Occurrence Management System for Smart Cities João Afonso Póvoa Marques

1 2 9 0

UNIVERSIDADE D
COIMBRA

João Afonso Póvoa Marques

Occurrence Management System for Smart Cities

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Catarina Helena Branco Simões da Silva and Engineer João Garcia and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2021

Faculty of Sciences and Technology

Department of Informatics Engineering

Occurrence Management System for Smart Cities

João Afonso Póvoa Marques

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering advised by Prof. Catarina Helena Branco Simões da Silva and Engineer João Garcia and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2021

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Abstract

City life poses many challenges to its citizens and administrations. One of these challenges is the reporting and handling of incidents. From reporting of incidents to their resolution administrations and authorities face difficulties in receiving new reports and resolving the incidents. The goal of this internship is the development of the Occurrence Management System, a solution that will face these challenges through the implementation of functionalities the assist in dealing with these kind of situations. The system allows citizens to report occurrences directly to the city, regardless of their scale, while also providing tools to manage and provide reports with relevant information.

This document presents all the steps taken during the internship to develop this product. Starting with the State of the Art, a study was conducted on competing solutions and other systems that could support the information gathering needed to achieve its goals. This step identified the key areas of occurrence management that have systems to support and the ones where improvements could be done. The results of this study were crucial to the elicitation of the requirements of the project and an architecture capable of meeting the needs of the system. The architecture was designed to support the development of the system and allow the company to easily improve the product after the end of this project. The C4 model was used to represent the architecture, which was based on a Microservices-oriented pattern. During this phase, the technologies to be used were also defined and Django was the selected framework to develop the system.

With the definition of the architecture finished the development phase started. During this process, several issues came up, the biggest one being the change in authentication and role management planned at the start not being capable of meeting the project's needs. Nevertheless, these obstacles were surpassed leading to a backend capable of managing occurrence reports and provide Authorities with reports with more information while also easier to consume. The document closes with the final remarks as well as a view on the future of the product, where the steps that could be taken next, and the possible new features that could be added to the system are described.

The present document elaborates on the work produced by the student João Afonso Póvoa Marques, in the context of the Internship in Software Engineering for the Masters of Informatics Engineering of the Department of Informatics Engineering of FCTUC.

Keywords

Occurrence Handling, Occurrence Reporting, Urban Platform, Smart Cities

This page is intentionally left blank.

Resumo

A vida na cidade apresenta muitos desafios para seus cidadãos e administrações. Um desses desafios é o relato e o tratamento de ocorrências. Desde a notificação de ocorrências até a sua resolução, as administrações e autoridades enfrentam dificuldades em receber novos relatórios e resolver as ocorrências. O objetivo deste estágio é o desenvolvimento do Occurrence Management System, a solução que vai fazer face a estes desafios através da implementação de funcionalidades e assistência no tratamento deste tipo de situações. Um sistema que permita que o cidadão comunique as ocorrências diretamente à cidade, independente de sua escala, ao mesmo tempo que fornece ferramentas para gerir as ocorrências e fornecer relatórios com informações relevantes.

Este documento apresenta todas as etapas realizadas durante o processo de desenvolvimento deste produto. Começando pelo Estado da Arte, um estudo foi realizado às soluções concorrentes e outros sistemas que pudessem apoiar a recolha de informações necessárias para atingir os objetivos. Esta etapa identificou as principais áreas de gestão de ocorrências que possuem sistemas de suporte e aquelas onde melhorias poderiam ser feitas. Os resultados deste estudo foram fundamentais para a elicitação dos requisitos de projeto e arquitetura capaz de corresponder às necessidades do sistema. A Arquitetura foi projetada para suportar o sistema e permitir que a empresa continue o trabalho no produto com facilidade. O modelo C4 foi usado para representar a arquitetura e um padrão orientado a microsserviços foi usado. Nesta fase, também foram definidas as tecnologias a serem utilizadas onde o Django foi o framework selecionado para desenvolver o sistema.

Concluída a definição da arquitetura, deu-se início à fase de desenvolvimento. Durante esse processo, vários problemas surgiram, sendo o maior deles a mudança no sistema de autenticação e gerenciamento de funções planeada no início não estar preparada para suportar o projeto. No entanto, esses obstáculos foram superados levando a um backend capaz de gerir as ocorrências e de fornecer relatórios com maior riqueza de informação e de mais fácil compreensão às Autoridades. O documento termina com as considerações finais e também uma visão sobre o futuro do produto, onde são descritos os passos que podem ser dados a seguir e as possíveis novas funcionalidades que podem ser adicionadas ao sistema.

O presente documento desenvolve a obra do aluno João Afonso Póvoa Marques, no âmbito do Estágio em Engenharia de Software do Mestrado em Engenharia Informática do Departamento de Engenharia Informática da FCTUC.

This page is intentionally left blank.

Acknowledgements

The past year posed many challenges, work and stress that lead to many sleepless nights. This work represented the culmination of a journey that now feels more fulfilling than I ever could have imagined. However this journey could not be made alone, it was only possible with the help and company of some exceptional people who have my unending gratitude.

To Ubiwhere, thank you for the opportunity to complete my journey here, thank you for helping my development as a professional and person, and for providing me the tools needed to complete this work.

I would like to thank Professor Catarina Silva, your precious input got me through this difficult task. Your teachings and valuable help were much appreciated.

To João Garcia, an outstanding advisor, I would like to express my gratitude for the help and guidance provided through this year, even when the time was short your readiness to help was remarkable and a crucial factor for the success of this work.

I must also thank my friends, all that were there through thick and thin, you that taught me, that laughed with me and that have helped me in so many ways during these last six years, without you this journey would have been impossible.

Daniela, words can not express the help you provided me during these tough times. Thank you for all the care, love, support and for staying with me throughout this arduous times. I hope to one day repay the immeasurable amount of help you provide me every day.

To my family, you are the cornerstone of the person I have become. Thank you for your support, teachings and understanding, you made this journey not only possible but also as easy as you could manage. I will use all that you have taught me my entire life, once again thank you for making me who I am.

This page is intentionally left blank.

Acronyms

AQI Air Quality Index. 44

GDPR General Data Protection Regulation. 29

IoT Internt of Things. 2, 6, 7

OMS Ocurrences Management System for Smart Cities. xiv, 1, 2, 3, 8, 20, 32, 33, 35, 36, 50, 51

RDMS Relational Database Management System. 34

UFRN Federal University of Rio Grande do Norte. 7

UN United Nations. 1

Contents

This page is intentionally left blank.

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

In any inhabited area, occurrences may result from human interactions, wear, and tear of the location's infrastructure, or external factors. These can result in losses such as small damages in roads, worrying concerns for public safety, or more pressing matters such as crimes or disasters. Those occurrences can be more prevalent in urban areas due to the density of population and more intensive use of infrastructure which can lead to an increase in problems within cities. The capability of a given city's administration to identify and respond to all types of occurrences (as well as the quality of this response) has a great impact on citizen satisfaction, making it crucial for a city to prove performant at handling situations.

In today's world a great part of its population lives in urban areas, the search for a better quality of life led people to congregate in areas where infrastructures and services provided were abundant and of better quality. These urban areas keep growing at an accelerated pace; according to the United Nations (UN) in 2017 55% of the world's population lived in urban areas and this value is estimated to increase by 18% by 2050[1]. To accommodate this ever-growing number of people cities have to grow and the challenge of managing a city becomes even harder. To bridge this challenge would be an effort requiring manpower and high costs which is often hard to fit in a city's budget.

Another challenge that cities face is collecting citizens' feedback. While there are emergency lines in place to handle the most pressing matters, events that do not yet class themselves as emergencies can not be reported through the same mechanisms. The process in place for a citizen to report his/her concerns related to the city can be time-consuming and cumbersome. Ignoring these problems can lead to escalation and therefore should be addressed by the administration. It is agreed that small disturbances can escalate rapidly into compound crises due to cities' complexity and tight coupling (Turner, 1978; Perrow, 1999). This means that due to the scale of a city and how it is intricately connected a small incident can affect much more than just the key area where the incident started.

To tackle this rising challenge of managing a city, many administrations use smart city products. These platforms give a multitude of information related to a city and its impact on the population. Depending on the extent of the service's implementation these technologies can help in maintaining the city organized and healthy.

The goal of this project is to further expand the Urban Platform, a smart city product, and tackle a different area of city management, occurrence handling, with the objective of expanding the area to meet needs expressed by city officials. It is here that the Ocurrences Management System for Smart Cities (OMS) comes into play, a product

that aims to improve the reporting and resolution of occurrences, incidents, or smaller events, for all involved. Providing a system for users to create and view incident reports on a wide variety of topics, built upon a platform to allow city administrations to better manage and analyze each occurrence and a system for the authorities responsible for the resolution to access better information more easily.

This document presents the work done during the internship which took place in the academic year 2020/2021, in the context of Dissertation of Masters Degree in Informatics Engineering in the branch of Software Engineering. The internship was hosted by Ubiwhere a company that focuses on developing intelligent systems for cities and state-of-the-art technology for the modern city.

1.1 Context and Opportunities

Ubiwhere is a company established in 2007 in Aveiro. Their work is focused on "Telco and Future Internet", working to innovate networks and infrastructures to grow alongside the use of the internet around the world. More relevant for this project is their work on smart cities, which encompasses several areas of city life like mobility, environment, energy, and many others.

Smart cities provide detailed information from several fields, gathered through a set of Internt of Things (IoT) devices connected in a network that feeds the system with data. Further data analysis provides a deeper understanding of the information gathered, allows for the exploration and development of new products and features. This allows the tackling of new challenges in a city and provides innovative solutions. Intending to make cities connected and sustainable environments smart cities aim to connect humans and institutional components with technology, it is in this environment that new challenges appear [2].

Smart cities aim to tackle challenges in various city domains such as waste management, mobility, and environmental concerns. Another challenge that cities face is managing occurrences, where the processes for reporting them can vary with the nature of each problem. For incidents of a pressing nature, emergency lines are available with a simple phone call, but for minor concerns such as potholes, damaged city property, or organizational problems, the process of filing a complaint or report can be long and tedious. This can lead to a lack of interest of the public to report and aggravation of problems due to the delay of the response.

With this challenge rises the opportunity to use Ubiwhere's product Urban Platform and with it create a new way to report the problems mentioned before, the OMS. The Urban Platform [3], launched in 2018, gives a view of a city like a single integrated system, providing a global view of cities' data from traffic to air quality and waste collection.

With this information, the goal is to tackle the occurrence reporting challenge with the intent to make the process easier for the citizens and authorities. This will be achieved by providing a single platform where reporting occurrences is easy and the information for the resolution is relevant and accessible. Improving resolution will be achieved by adding relevant information, that may lead to incident escalation or help in the resolution, to the original reports.

1.2 Objectives

The goal of the OMS is to help citizens become active participants in the city's management, giving users the possibility to express their concerns more easily, directly, and with more detailed information. This in turn makes it easier for administrations to identify problems by giving them accurate information regarding all problems, saving time and resources for all parts involved. To achieve these goals the backend of a system will be planned and implemented by the intern. This system and its different modules will be responsible for:

- **Receiving new incident reports**, citizens should be capable of submitting new reports to the platform to alert the city of issues.
- **Allowing management of reported occurrences**, the managers should be able to visualize, sort, search and update the occurrences, as well as assign them to someone responsible for solving them.
- **Adding information to the reports**, when the resolution of an occurrence begins the system should take the initial report that was submitted by the citizens and add information regarding traffic, air quality, and weather to it automatically when the resolution phase starts. This information is directly connected to the occurrence and can impact it or its resolution in some capacity. By attaching this information to the report, it is possible to do better decision-making during the resolution phase.

These goals shall be completed during two phases:

- The first phase, spanning the first semester, consists of **studying the concept** and the **state of the art**, which will give a view of the work already done on this subject. With the information gathered from this process, it will be possible to **establish the product** including its **Requirements**, **Architecture** and the overall **Planning**.
- The second phase, which lasts the remaining time of the internship's duration, is reserved for **product development** and **integration with the Urban Platform**. This product consists of a **backend** application capable of **taking information provided by citizens**, **integrating it with contextual data from the city** (such as traffic, weather, and events that may impact the incident's resolution). Following this, it should be possible to **return a report**, complete with all the information and an overview of the incident in natural language, for the use of the city's administration. During this period the system will also be **tested** to ensure that the products behave as expected.

  The report is elaborated during all phases of the work.

1.3 Document Structure

- **Chapter 2**, State of the Art, presents the results of the study on the concept where similar products were studied to learn good practices and features from them. Proceeding to describe Ubiwhere's concept, the result of all desired features, and the technologies used to materialize the system.

- **Chapter 3**, Planning and Methodology, describes the planned work for the whole project's duration, as well as the processes implemented for work and development.
- **Chapter 4**, Requirements Specification, presents the functional and non-functional requirements established for the product as well as the actors of the system. User Stories were utilized for the specification of the functional requirements. The non-functional requirements are represented in the form of restrictions imposed on this project as well as the quality attributes that the product must meet and which will impact the architectural decision.
- **Chapter 5**, Architecture and Technology, describes the system architecture, details the technologies to be used, and gives a review of the design decisions.
- **Chapter 6**, Development, presents the development environment, the requirements that were implemented, and the details and decisions made during the implementation of each one.
- **Chapter 7**, Testing, presents the implemented tests to assure that the requirements and quality attributes are met. A description of the test and the conclusions drawn from them are also described.
- **Chapter 8**, Conclusion, elaborates on the thoughts of the entire internship. Providing an overview of the problems faced and a critical analysis of the work done, the lessons learned, and the work for the future.

This page is intentionally left blank.

Chapter 2

State Of The Art

In this section, the results of the background research around products capable of creating a real-life scenario of event reporting are presented. It begins by contextualizing the scope around smart cities as a support system to event reporting systems down to the challenges of the area. To complete this chapter the standards of the industry are analyzed, this research allowed for the project's conception resulting in the idea and features to be developed.

2.1 Smart Cities

The term "Smart City" is becoming more prominent, the term has multiple definitions yet in all of them have matured significantly, advances in technology and a surge in interest from cities make these solutions a desirable tool for city administrations. Smart cities, despite the numerous definitions, are a desired product in order to help management as cities grow and aim to achieve a common objective: to improve the quality of life of its citizens and the quality of the services [2].

Technological advances in areas such as Internt of Things (IoT) and connectivity are improving smart city products, providing better and new information, making a smart city environment richer with data. This is the key to making a city smarter, this gathering process can only lead to innovation when it is coupled with communication with other intelligent systems that can all understand each other. These exchanges lead to the development of new software applications that use these new resources gathered to transform them into knowledge that impacts the decision-making process of city management[4].

2.1.1 Smart Cities and Occurrence Reporting

Occurrences, may they be small events or incidents when they escalate, are part of the city's information, and like seen previously they compose yet another one of these pieces of information circulating in the smart city environment. Companies like IBM[5] already have a solution that can be used alongside their smart city product; in their suite information is already exchanged in the system and they allow for automatic responses, this is a great benefit to cities since with an automated process there are guarantees that no occurrence is left unattended since their resolution can be started manually or automatically when needed.

Systems for event reporting can and have been implemented without the support of a smart city product behind them. Products such as FixMyStreet or "Na Minha Rua Lx" simply present the user with a platform to report issues in their local areas. This is a possible solution and one that people have been using more and more, with Lisbon's solution "Na Minha Rua Lx" growing in usage significantly, in 2018 the number of reports grew from nearly 13000 in April to 15000 in May[6]. These solutions while not as complete as IBM's solution are a great starting point for cities with no support from smart city products and create a bridge between administrations an citizens allowing the mitigation of some of the problems related to occurrence reporting. Nonetheless, pairing them with a smart city solution that gathers extensively more information relevant to the occurrences rather than just the occurrence itself can be a great asset to further improve on the city's occurrence handling capabilities.

2.1.2 Smart Campuses and Occurrence Reporting

Some campuses around the world have started implementing services that align with those of smart cities and much like them are starting to develop services that are generated through this new source of information. Such is the case of the Federal University of Rio Grande do Norte (UFRN) where an application to report incidents and IoT support for the betterment of the app have already been developed. The development of this product was born from a necessity of improving campus security as it was found lacking in quality and responsiveness. A study elicited three issues with the established process, these were a managerial issue, where they found that traditional methods of communication between authorities couldn't provide information regarding the authorities and the incident in a quick and detailed manner, a communications issue, where the conservation of essential and accurate information could be affected through the communication process and an informational issue, information was being collected and stored in an inadequate way and provided no means of analysis or data visualization.

In order to tackle these issues and provide a more efficient way of reporting occurrences the system **SIGOc**[7] was created. This product provided a mobile app for incident reporting that allowed students on campus to report occurrences with all the basic information relating to the incident. This report is then sent to a guard that has access to his own application where he can access the incident's information and give status updates on them. All this information is also accessible through a dashboard where a supervisor can visualize guard location and incoming reports in real-time as well as create new occurrences and assign specific guards to the occurrence. This system was later improved through its integration with the smart campus program, now IoT devices can send information like the number of students in a given space, temperature and motion directly to the supervisor to provide a quicker and better understanding of ongoing occurrences and emergency situations. The system was regarded as a great asset to campus security and a good response to the issues mentioned before, facilitating management of resources, making communication easier and more accurate, and creating a better way to store and visualize information allowing for data analysis to identify critical points in security.

2.1.3 ArcGIS and Occurrence Reproting

Occurrence reporting is not limited by smart city aid, as it will be explored further in this chapter other solutions that are not backed by smart city products make use of other systems for the handling of geographical information. One of these systems is the ArcGIS[8] platform, used in products like Citizen Problem Reporter. To understand how it could help in the product's development there is a need to understand first what GIS is. GIS or Geographic Information System is a set of tools for analysis, management, and display of geographical info, composed of a series of geographical datasets. GIS has three major views:

- **Geodatabase** view is a collection of datasets that represent geographical information such as features and topology of the land;
- **Geovisualization** view is a series of map views divided into layers (transportation, postal codes, land use, raster images, and others) that when combined provide a complete view of the information stored in the Geodatabases allowing for a user-friendly interpretation and manipulation of the data;
- **Geoprocessing** view is a set of tools that apply analytic functions to the information in order to derive new information from it.

With the progress of computing and networking the vision and role of GIS also evolved. In addition to the existing GIS Desktop, GIS software can be deployed in application and Web servers. GIS can be embedded and deployed in custom applications and even mobile applications that allow new ways of using GIS. These new uses extend to event reporting where several products already use GIS software to aid in managing events and incidents.

ArcGIS is a response to this new scope and uses for GIS software. A product line was built by Esri in order to satisfy these new requirements and provide a scalable and comprehensive GIS platform with access to all of GIS's software and features. The use of ArcGIS for an event reporting system is not new and can be a great feature for improving these systems. However, the Ocurrences Management System for Smart Cities (OMS) is projected to work in tandem with Ubiwhere's Urban Platform which, as a smart city product, already possesses information relevant to the reports there is no need to use the ArcGIS suit even though it is a suitable alternative.

2.2 Competitor Analysis

The solution to be presented should be of public interest and should be a good fit for the necessities of the regions encompassed by the Urban Platform. To ensure that these requirements were met without the need to make inquiries and research outside the scope of the project some outlines were defined at the start of the internship, allowing a research phase on similar products in order to bolster the solution.

A set of questions was prepared in order to narrow down the key factors for the solution:

- What degree of event severity is acceptable in the solution?
- Which information is being gathered relating to the occurrence and the user?

- What further additional features the solution implements to benefit event handling?

To ensure that the product covered as much of event reporting as possible research was done on products supported by smart city solutions and solutions that are not supported by these systems. This separates them into two groups, where the group not supported by other information-gathering systems is excluded for the third question.

After studying several platforms/products three solutions stood out, below is a description of their functioning.

- **IBM Intelligent Operation Center**[5] - Working worldwide, allows for the reporting of events, these events are typically on a small severity scale and can be scaled to incidents manually if they aggravate. This product is heavily backed by IBM's suite and this support allows it to keep events in check, events that are reported to the platform are monitored and through analysis of other data collected by their products like the weather the system can determine if an event can escalate and even deploy standard protocols of resolution.

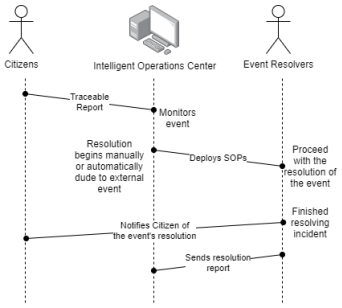

Figure 2.1: Intelligent Operation Center's process

- **Citizen Problem Reporter**[9] - has support to work worldwide and deals only with non-emergency events, this system is based on the ArcGIS software for occurrence reporting gathers information relative to the event as well as a photo, location input can be given either through address or map pinning which is easy for all users and the basic information on the event, this system while giving citizens the possibility to follow the resolution status deploys no other action regarding the occurrence. The friendly user interface and availability as a mobile app are great factors to improve citizen engagement.

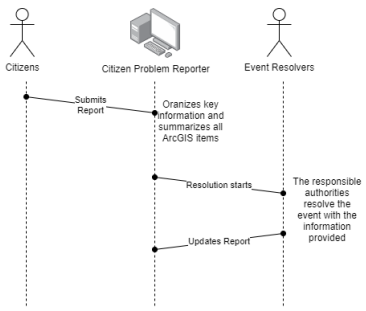

Figure 2.2: Citizen Problem Reporter's process

- **Fix My Street**[10] - this solution is browser based and limited to the UK with the possibility of using their API to set up the system anywhere, citizens can give an address and are then presented with a map focused on the address where they can specify location and give basic information on the incident as well as a photo. The status of the occurrence is updated for users yet no further work is done thought the platform.

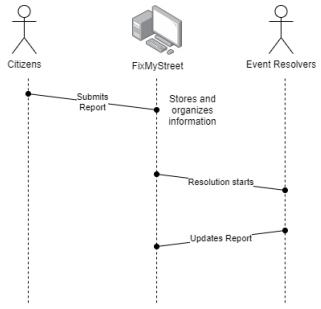

Figure 2.3: FixMyStreet's process

After evaluating both the concept and available market solutions studied, focusing primarily on the three selected products, and with the support from Ubiwhere's advisor the table 2.1 was elaborated as an answer to the three questions posed on the solutions.

		IBM Intelligent Operations Center	Citizen Problem Reporter	FixMyStreet
Region		Worldwide	Worldwide	UK (Worldwide API)
Event Scale		Starts as non-emergency and administrators can scale to incident	Non-Emergency Events	Non-Emergency Events
Information Gathered from Users	Type of Occurrence	Yes	Yes	Yes
	Location	Yes	Defined through giving an address and pining the exact location on a map	Address
	Description	No	Yes	Yes
	Time of Occurrence	Yes	Yes	Yes
	Severity	Yes	No	No
	Photos or Attachments	No	Yes	Yes
	Name of the Reporter	No	Yes	Yes
	Contact of the Reporter	No	Yes	Yes
Additional Tasks		The system monitors events with the help of other information in the platform and can deploy resolution protocols	No additional tasks performed	No additional tasks performed

Table 2.1: Comparison of Competing Solutions

Table 2.1 goes into more detail on the analysis of each competing product. In terms of availability by region, no product is limited to a specific region, the adoption of the products suite or API can be used regardless of the region of the client. An important piece of research was the "Event Scale" this is the scale of accepted reports from more to less serious issues, here we see that at an initial stage all platforms only accept occurrences that are not considered emergencies, in the specific case of the Intelligent Operations Center these events can be scaled up to incidents after the report.

Next, there was a need to study the information gathered by each product and how this information was being collected from the citizens. The information gathered is similar across all products where one would take a different approach compared to the others. For the location of the occurrence, each product adopted a different manner of specification yet the information is key to the process and it is present in all products. All products allow for the specification of the type of occurrence from a wide range of possibilities, most allow

for a description to provide extra details while the Intelligent Operations Center uses the type of occurrence as a description which can be limiting in information, this is balanced with the presence of severity which is not available in other products. Severity allows users to specify the state of the occurrence in order to access dangers and prioritize since more severe incidents can lead to more problems. The presence of media is common, both the Citizen Problem Reporter and FixMyStreet allow for the upload of photos or other media like voice messages with each report. Finally, mostly all products gather some type of personal information of the citizen who creates the report, the products that collect this type of data all request the name of the citizen yet the contact varies from phone numbers to e-mail.

Lastly and one of the most important aspects of analysis is the additional tasks performed, this aspect focused on any actions performed on the report or with the report after the report beyond the managerial tasks such as deletion and set as complete. In this aspect, products do not offer much outside the notable example of IBM which, if the rest of the suite is deployed together with the Intelligent Operations Center, can automatically start the incident's resolution if the system, through external information, detects that there is a chance that the situation can escalate and cause further damage.

Chapter 3

Planning and Methodology

This chapter presents the methodology adopted during this internship, as well as the plans laid out for both semesters.

3.1 Process Management

During the entire internship, an Agile-inspired methodology was adopted to take from it its benefits and adapt it to the internship's needs. The Agile framework has the benefit of allowing adaptive planning and fast delivery which in turn gives fast feedback on the product. These characteristics not only help with fast product delivery but are also flexible to change and aim to focus mainly on the product[11]. Since Agile methodologies tie their development with the prioritization of User Stories this will also be the method adopted for the Requirements in Chapter 4. These benefits were especially valuable when working with the advisor, with ease of scheduling meetings and keeping him updated on the work at hand. These meetings served as regular updates where the work done in the past week was reviewed and feedback was provided. The work to be done in the following week was established in these meetings as well, based on the past week's work and the overall progress of the project. This methodology will also allow the intern to work autonomously but sustain a controlled environment where advisor feedback in the weekly meetings can help the intern with his difficulties and needs and identify mistakes that may cause problems down the line if left unchecked. The adopted methodology is visually represented in figure 3.1.

In this process the following actors participated:

- **Product Owner:** Ubiwhere (represented by João Garcia)
- **Project Manager:** João Garcia
- **Developer:** João Marques

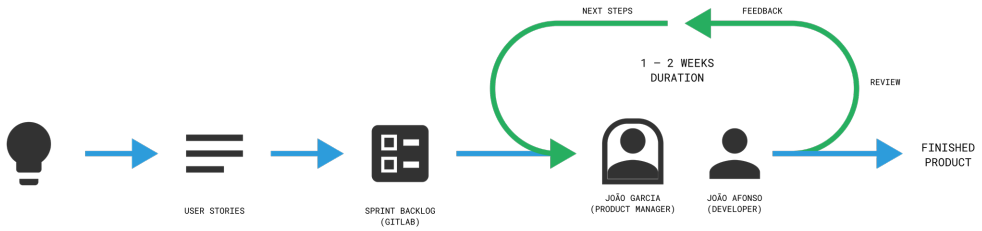

Figure 3.1: Adopted Agile-Inspired Methodology

Figure 3.1 goes into more detail of the steps taken during the internship. Despite the Agile nature of the project it was possible to define User Stories early which leeds to changes in requirements having a low impact on the project and an early creation of a backlog of tasks to be done. After this the sprints are visible, each sprint concluded with a meeting with the advisor where a review of the work was done and feedback was provided. During these meetings issues were also discussed if any had came up during the sprint, changes and other solutions to mitigate these issues were discussed in this meeting.

In order to keep track of task and time management the platforms BambooHR[1] and Clockify[2] were used, these platforms are hosted by Ubiwhere. BambooHR allows for the tracking of the total time spent on all tasks and Clockify (figure 3.3) lets the intern register the time spent on each individual task. Gitlab was also used not only for version control but also as a platform to keep track of the project's development (figure 3.2). The issues on Gitlab were used firstly as a tracker of what tasks are done and what are missing and also for updates and discussion on the tasks currently being worked on. The issues feature allowed discussion on the task as well as report bugs, raise problems, and review the code.

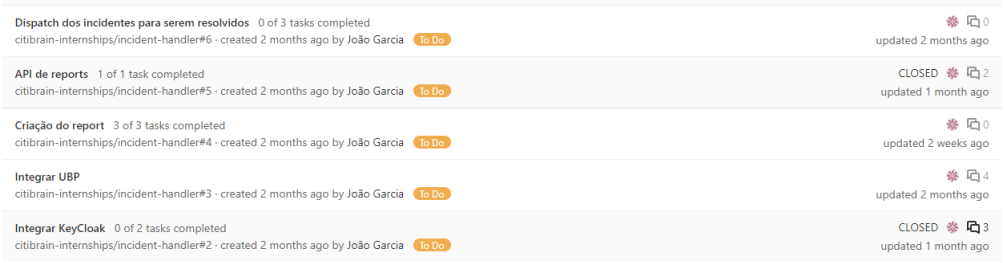

Figure 3.2: Gitlab Issues

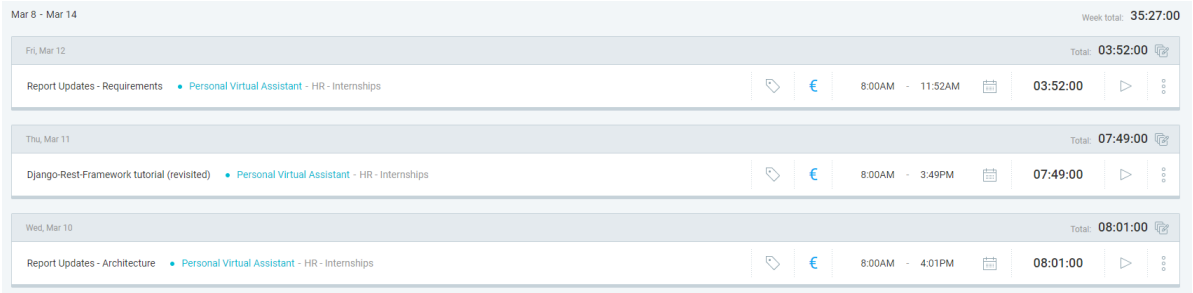

Figure 3.3: Clockify Example

[1] https://www.bamboohr.com/
[2] https://clockify.me/

For the planning of both semesters, Gantt charts were used. There is much debate on the utilization of these charts in Agile methodologies while some claim there is no use for Gantt charts in Agile, such as Jeff Sutherland who banned Gantt when he first implemented Scrum[12], there are a growing number of companies that use these diagrams. In Agile Gantt charts are typically used to illustrate releases or sprints, these charts are simple and help to bridge differences between all parties involved in the project. In the classical sense of Agile Gantt charts' use is limited and would need to be updated continuously, where Gantt comes in strongly into Agile is from the management standpoint, providing a quick and easy to understand overview of the time spent on tasks helping managers to oversee the progress made against estimates[13].

3.2 First Semester

The following section presents the planning for the first semester, from September of 2020 to January of 2021, which was focused on concept study, architecture definition, and requirement specification. In this section, the Gantt chart (3.4) produced for this semester is also presented.

- **Intermediate Report:**
  - **Concept Study:** Study of the products and platforms available, the progress of event reporting systems worldwide, and their adoption by cities as well as external platforms that could aid in concept improvement.
  - **Report Writing:** Elaboration of the report with the guidance of both advisors.
  - **Intermediate Report Delivery:** 18 January 2021
- **Requirement Specification:**
  - **User Stories:** Specification of the requirements and production of the Chapter 4.
  - **Requirements Approval:** Approval of the specified requirements from Ubiwhere's advisor.
- **Architecture Design:**
  - **Architecture Patterns:** Research on the patterns to use when implementing the product.
  - **Technologies:** Research on the technologies to use when implementing the product.
  - **Report:** Adding to the report the studies and conclusion of the research.

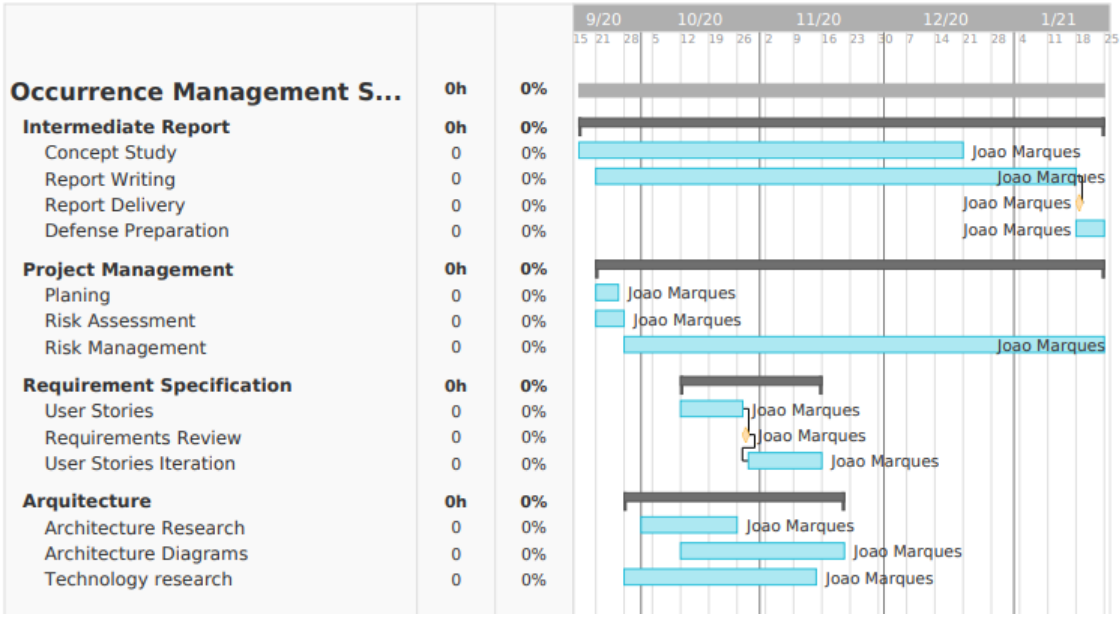

Figure 3.4: First Semester Gantt Diagram

3.3 Second Semester Planning

In this section is presented the planning for the second semester of work. This semester focuses on two tasks, product development and writing the final internship report. A Gantt chart (3.5) was also produced to better visualize the span of both these tasks.

- **Product Development:** In this phase focus will be directed to implementing and testing the product of this internship.
- **Report:** In the second semester this stage will be composed of two parts, firstly the intern will act on the feedback provided in the intermediate defense after this is concluded the focus will be on writing the implementation and testing chapters of the report.

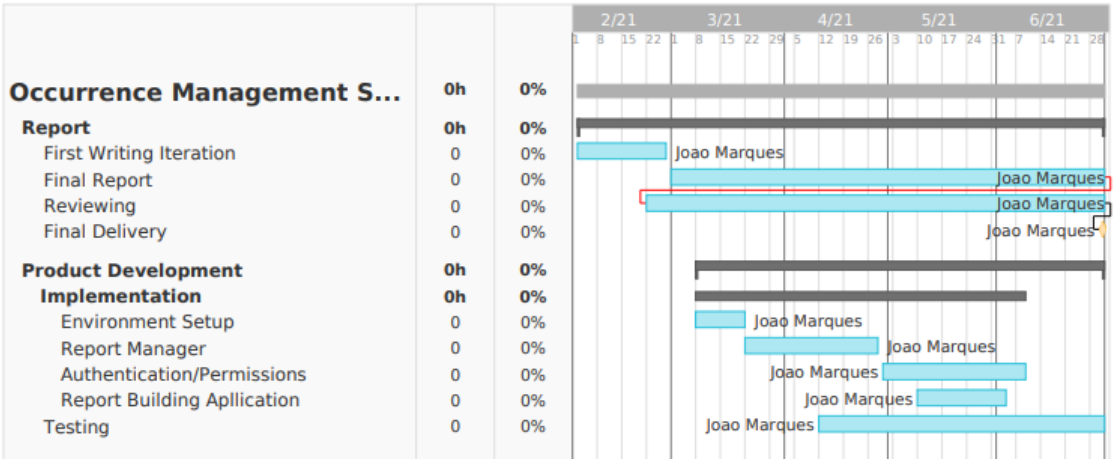

Figure 3.5: Second Semester Gantt Diagram

3.4 Internship Success Criteria

The success criteria for this internship represents the agreement between the intern and Ubiwhere and ensure the guidelines imposed for the internship are respected. To evaluate the success of the project the following criteria were established.

- The requirements specified for the product and the subsequent architecture design are approved by the hosting company.
- Requirements classified as "Must Have" must be fully implemented and tested at the time of final delivery. To note that while the user stories presented in chapter 4 were designed with the frontend of the project in mind, this component is considered out of scope for the internship. Therefore the classification "Must Have" implies that only the backend of the feature must be implemented and tested.
- The final product must meet the quality attributes proposed in the section 4.6 of the chapter 4.

3.5 Risks Management

The Threshold of Success (ToS) is a project's boundary between success and failure and defining it needs to be done from both the perspective of the failure or success at the end of the project[14]. The success factors are defined in section 3.4, the perspective of failure can be done while eliciting risks through imagining a picture of failure and using these reasons behind the supposed failure of the project as an initial point for the definition of the risks. In order to ensure a successful project and that the Threshold of Success (ToS) is reached attention must be paid to all factors involved. Looking at a project from the point of view of what problems can be felt further down the project's development or in the picture of failure can eliminate the chance of oversight and unforeseen circumstances. By identifying these risks it is also possible to also evaluate their probability of occurrence and its possible impact on the project. Based on these factors of probability and impact it is also possible to create mitigation plans for each risk to be deployed when a risk becomes a problem in the project. To classify each risk, according to their impact on the project, a scale was implemented on a scale of 1 to 3 where:

- **1 - Marginal:** ToS can be reached without great difficulty.
- **2 - Critical:** ToS can be reached but with greater effort/cost.
- **3 - Catastrophic:** ToS can not be reached

In addition to the impact scale there is also a probability scale, also from 1 to 3, where:

- **1 - Low Probability:** $<40\%$
- **2 - Medium Probability:** between 40% and 70%
- **3 - High Probability:** $>70\%$

ID	Risk	Probability	Impact
R01	The product relies on Urban Platform, problems with integration may delay the implementation.	1	3
R02	Time spent on a task exceed the expected.	2	2
R03	Lack of experience in the technologies and tools chosen.	2	2
R04	Changes in the functional requirements or quality attributes.	1	2
R05	The product relies on Urban Platform's new authentication which may not be functional at the time of the implementation.	1	3

Table 3.1: Risk Management Analysis

With this system, it is possible to classify each risk identified by its impact on the project and likelihood of happening. The following table (3.1) presents all the risks as well as their classification on impact and probability according to the scales defined before.

For each of these risks, a mitigation plan was established. These plans are established in the table 3.2 below.

Risk ID	Mitigation Plan
R01	The intern should take time to get acquainted with the platform before starting the integration.
R02	This should be reported in the next sprint meeting and the relevance of the task should be re-analysed. If there are more important tasks to do the intern should focus on them first.
R03	Dedicate time to learn these new technologies before the start of the project.
R04	Reevaluate the requirement's priority in order to keep it valuable for the company and feasible for the available time.
R05	The intern should implement an independent mechanism for authentication outside the Urban Platform's scope or the product should be developed as an Django Application inside the Urban Platform (similar to how the platform handles Smart Cities verticals)

Table 3.2: Risk Mitigation Plan

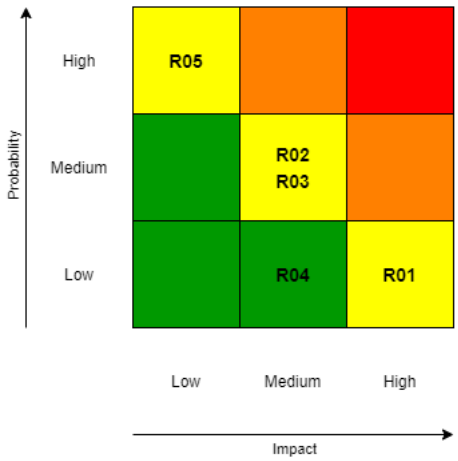

Figure 3.6: Risk matrix

Figure 3.6 displays the risks in a matrix by their probability and impact. This is

typically done in order to visualize the "cut line", a line that separates the low impact risks from the high impact ones, in order to create mitigation plans for those that will impact the project, normally risks below the cut line have a negligible impact on the project and can be addressed easily at the moment when the risk starts being a problem. Since there are no risks above the cut line mitigation plans were developed for all risks, as seen in Table 3.2, as a mechanism to prepare the solutions to these problems ahead of time.

With the establishment of the success criteria, the definition of the methodology, and the evaluation of the risks, the planning of the internship is complete. This plan spans both semesters and allows a view of the internship on each individual semester. The evaluation of the risks that may hinder the success of the project allowed planning of means of mitigation or correction should they arise. With this global view of the project and the challenges that will have to be surpassed, this will serve as a guideline for the remainder of the internship and the development of the product.

Chapter 4

Requirements Specification

This chapter presents all the requirements identified for the proposed solution, the restrictions imposed on the product, and its quality attributes. For a better understanding of the requirements the actors, the tasks that each can perform, and the terminology used for the specification of the requirements are described.

4.1 Actors

The actors are the different types of users that will engage with the system. When planning the system and after defining how information flows through the system 3 types of users were identified. These users will act upon the information in different ways from information treatment to information input, the 4 actors are:

- The **"Citizen"** is a regular user in the platform and has access to the service in order to report occurrences without the need of login, this interaction with the platform, as addressed in the architecture, is done through the App.
- The **"City Administrator"**, this type of user, who interacts with the service through the Urban Platform, is a user that is a person responsible for the handling of incoming occurrence reports, appointed to work through the platform as part of public office job, should review and dispatch incoming occurrence reports to the relevant authorities (i.e. Firefighters, Police Officers, etc).
- The **"Authorities"** that represent the users that will receive the occurrence report after the dispatch in order to respond.

4.2 Report Life-cycle

In section 2.2 the processes of the three chosen products were analyzed. A similar analysis was done to the expected life-cycle of the report in the OMS. This was done to help elicit functional requirements based on:

- The life-cycle of the reports;
- How the occurrence reports are handled and used by all actors;
- The flow of information between the actors and the system;

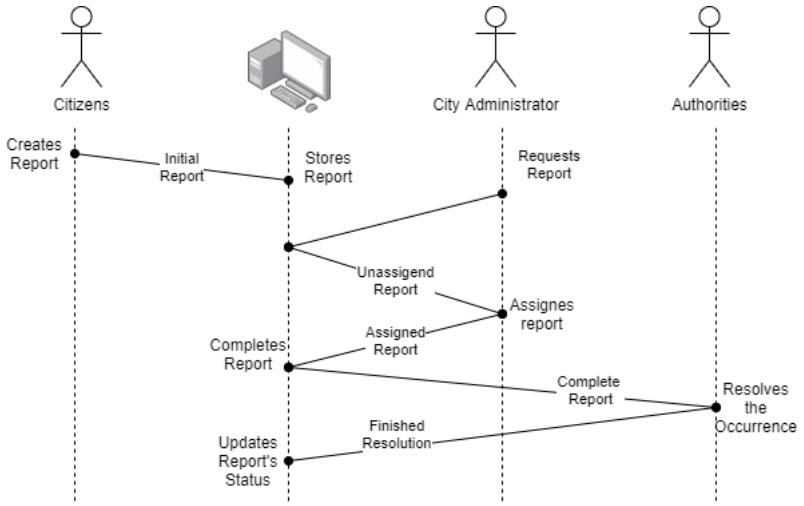

Figure 4.1: Report Life-cycle

Upon the report's creation by the user, the system simply stores it and will wait until a City Administrator initializes the resolution process. This process begins when a City Administrator assigns the report to an available Authority, this action starts an internal process to improve the Citizen's report and the system builds a new report that includes the occurrence's information provided by the user and additional environment information, gathered from the Urban Platform, this information is tailored to better fit the occurrence. This process also includes a conversion of the newly built report into natural speech format for easier accessibility by the Authorities. When this process is finished, it is sent to the authority which will automatically be displayed on his mobile phone and the natural speech will be played. After this, all that remains is for the authority to complete the occurrence's resolution and send the confirmation that the work has finished, the report will continue to be stored in the system with its status changed to "resolved". Since the same technique was used to define the life-cycle of the OMS and the competing products in section 2.2 a few comparisons can be made between them. While all of the products allow the creation and management of reports, visible in the life-cycles presented, all products have a different focus. The FixMySteet and Citizen Problem Reporter both focus heavily on report ingestion and management, with the later having an added bonus of being based on the ArcGIS software which allows it to access Geographical tools that can aid on analysis. IBM's Intelligent Operations Center while also being a great tool for report management draws great benefits from having the capability of starting resolutions automatically, this focus beyond management is also shared by the proposed solution in a different capability. The OMS shares with the competition the management aspect, yet much like IBM's solution it goes further and tries to benefit other aspects of the occurrence's resolution process in this case the improvement of the report.

4.3 Requirement Structure

The requirements created for this internship adopt the following structure:

US#ID

- **Description:** As an <actor>, I want <feature> so that <reason>
- **Acceptance Criteria:**
  - **Scenario:** A determinable business situation
    - **Given** some precondition
      And some other precondition
    - **When** some action preformed by the actor
    - **Then** some measurable outcome is achieved
- **Dependencies:** US#ID
- **Priority:** Must have, Should Have, Nice to Have

Each story can also be composed of multiple scenarios and each scenario can have several preconditions, actions, or outcomes.

4.4 Functional Requirements

For the elicitation of the functional requirements, the Competitor Analysis done in section 2.2 was used together with the report's life-cycle analysis and Ubiwhere's view of the project as a baseline for the product's functioning. After understanding what features should be available for each actor, based on Ubiwhere's needs and other products features, the following requirements were defined:

US01

- **Description:** As a Citizen, I want to report an occurrence so that the problem is sent to the authorities.
- **Acceptance Criteria:**
  - **Scenario:** Citizen provides basic information on an active occurrence
    - **Given** that I can access the platform
    - **When** I press the report occurrence button
      And I provide the necessary information for the handling of the occurrence
    - **Then** the occurrence is registered
  - **Scenario:** The Citizen provides insufficient information on an active occurrence
    - **Given** that I can access the platform
    - **When** I press the report occurrence button
      And I fail to provide the basic information on an active occurrence
    - **Then** I should see an error message
- **Dependencies:** N/A
- **Priority:** Must Have

US02

- **Description:** As a Citizen, I want to see all occurrences reported on a map view so that a more geographical view of occurrence distribution is visible.
- **Acceptance Criteria:**
  - **Scenario:** The Citizen wants to see all reported occurrences in a map
    - **Given** that I can access the platform
    - **When** I press the map view button
    - **Then** I am presented with a map with all the reported occurrences marked
- **Dependencies:** N/A
- **Priority:** Should Have

US03

- **Description:** As a Citizen, I want to see a list of all reported occurrences with more detailed information.
- **Acceptance Criteria:**
  - **Scenario:** The Citizen wants to see a list of all reported occurrences with details on each
    - **Given** that I can access the platform
    - **When** I press the list occurrences button
    - **Then** I am presented with a list of all occurrences reported
      And detailed information on each of them
- **Dependencies:** N/A
- **Priority:** Must Have

US04

- **Description:** As a Citizen, I want to sort all reported occurrences by date, type or status
- **Acceptance Criteria:**
  - **Scenario:** The Citizen want to sort occurrences by status
    - **Given** that I am on the occurrence report list page
    - **When** I press the sort button
      And I select the parameter I want to sort by
    - **Then** I get a list of reports sorted by status
- **Dependencies:** US01
- **Priority:** Must have

US05

- **Description:** As a Citizen, I want to search for a specific occurrences
- **Acceptance Criteria:**
  - **Scenario:** The Citizen wants to search for occurrences
    - **Given** that I am authenticated **And** on the occurrence report list page
    - **When** I type in the search bar a keyword
      And I press the search button
    - **Then** I get a list of reports whose descriptions contain the search parameters
- **Dependencies:** US01, US08
- **Priority:** Must Have

US06

- **Description:** As an Authority, I want to login in the Authority App so that I can recieve reports.
- **Acceptance Criteria:**
  - **Scenario:** The Authority has access to the platform and inputs valid credentials
    - **Given** that I am unauthenticated
      And I have permission to access the platform
    - **When** I input a valid e-mail and password on the login form
    - **Then** My session is initiated
      And I can now receive reports
  - **Scenario:** The Authority has access to the platform and inputs invalid credentials
    - **Given** that I am unauthenticated
      And I have permission to access the platform
    - **When** I input an invalid e-mail and password on the login form
    - **Then** I get an error message
      And the session is not initiated
  - **Scenario:** The Authority does not have access to the platform
    - **Given** that I am unauthenticated
      And I do not have permission to access the platform
    - **When** I input any combination of e-mail and password on the login form
    - **Then** I get an error message
      And the session is not initiated
- **Dependencies:** N/A
- **Priority:** Must Have

US07

- **Description:** As an Authority, I want to logout from the Authority App so that my session is terminated.
- **Acceptance Criteria:**
  - **Scenario:** The Authority wants to end his session
    - **Given** that I am authenticated
    - **When** I press the logout button
    - **Then** my session is terminated
      And I can no longer work on the platform
- **Dependencies:** US05
- **Priority:** Must Have

US08

- **Description:** As an Authority, I want to archive an occurrence so that is status is set as "resolved".
- **Acceptance Criteria:**
  - **Scenario:** The Authority finishes resolving the occurrence
    - **Given** that I am authenticated
      And the occurrence has been resolved
    - **When** Press the resolve button
    - **Then** the occurrence's is archived
      And its status is set as resolved
      And an e-mail is sent to the reported notifying him of the occurrence's resolution.
- **Dependencies:** US01, US05, US15
- **Priority:** Must Have

US09

- **Description:** As a City Administrator, I want to login in the Urban Platform so that I can work on reports.
- **Acceptance Criteria:**
  - **Scenario:** The City Administrator has access to the platform and inputs valid credentials
    - **Given** that I am unauthenticated
      And I have permission to access the platform
    - **When** I input a valid e-mail and password on the login form
    - **Then** My session is initiated
      And I can now receive reports

  - **Scenario:** The City Administrator has access to the platform and inputs invalid credentials
    * **Given** that I am unauthenticated
      And I have permission to access the platform
    * **When** I input an invalid e-mail and password on the login form
    * **Then** I get an error message
      And the session is not initiated

  - **Scenario:** The City Administrator does not have access to the platform
    * **Given** that I am unauthenticated
      And I do not have permission to access the platform
    * **When** I input any combination of e-mail and password on the login form
    * **Then** I get an error message
      And the session is not initiated

- **Dependencies:** N/A
- **Priority:** Must Have

US10

- **Description:** As a City Administrator, I want to logout from the Urban Platform so that my session is terminated.
- **Acceptance Criteria:**
  - **Scenario:** The Authority wants to end his session
    * **Given** that I am authenticated
    * **When** I press the logout button
    * **Then** my session is terminated
      And I can no longer work on the platform
- **Dependencies:** US09
- **Priority:** Must Have

US11

- **Description:** As a City Administrator, I want to see all occurrences reported on a map view so that a more geographical view of occurrence distribution is visible.
- **Acceptance Criteria:**
  - **Scenario:** The City Administrator wants to see all reported occurrences in a map
    * **Given** that I am authenticated
    * **When** I press the map view button
    * **Then** I am presented with a map with all the reported occurrences marked
- **Dependencies:** US09
- **Priority:** Should Have

US12

- **Description:** As a City Administrator, I want to see a list of all reported occurrences so that I can access more detailed information.
- **Acceptance Criteria:**
  - **Scenario:** The City Administrator wants to see a list of all reported occurrences with details on each
    - **Given** that I am authenticated
    - **When** I press the list occurrences button
    - **Then** I am presented with a list of all occurrences reported
      And detailed information on each of them
- **Dependencies:** US09
- **Priority:** Must Have

US13

- **Description:** As a City Administrator, I want to sort all reported occurrence by date, type or status
- **Acceptance Criteria:**
  - **Scenario:** The City Authority wants to sort occurrence by status
    - **Given** that I am authenticated **And** on the occurrence report list page
    - **When** I press the sort button
      And I select the parameter I want to sort by
    - **Then** I get a list of reports sorted by status
- **Dependencies:** US01, US09
- **Priority:** Must have

US14

- **Description:** As a City Administrator, I want to search for a specific occurrence
- **Acceptance Criteria:**
  - **Scenario:** The City Authority wants to search for occurrence
    - **Given** that I am authenticated **And** on the occurrence report list page
    - **When** I type in the search bar a keyword
      And I press the search button
    - **Then** I get a list of reports whose descriptions contain the search parameters
- **Dependencies:** US01, US09
- **Priority:** Must Have

US15

- **Description:** As a City Administrator, I want to assign an occurrence so that its resolution can begin.
- **Acceptance Criteria:**
  - **Scenario:** The City Administrator wants to assign an occurrence to an Authority to being its resolution
    - **Given** that I am authenticated **And** I am on the occurrence report details page
      And there are available Authorities
    - **When** I press the assign button
      And I select an Authority to assign the occurrence to
    - **Then** the selected occurrence changes its status to "in progress"
      And an Authority gets assigned to the occurrence
      And an e-mail is sent to the Citizen that created the report with the status update
- **Dependencies:** US01,US06, US09, US12
- **Priority:** Must Have

US16

- **Description:** As a City Administrator, i want to delete a fallacious report so that it is no longer present on the platform
- **Acceptance Criteria:**
  - **Scenario:** The City Administrator encounters a report that has been determined false
    - **Given** that I am authenticated
      And I am on the occurrence report details page
    - **When** I press the delete button
    - **Then** the report is removed from the system
- **Dependencies:** US01, US08, US12
- **Priority:** Must Have

US017

- **Description:** As a Citizen, I want to use speech recognition to report an occurrence to the authorities so that the problem is sent to the authorities.
- **Acceptance Criteria:**
  - **Scenario:** Citizen provides basic information on an active occurrence and is comprehended by the system
    - **Given** that I can access the platform
    - **When** I press the report occurrence button
    - **And** And I press the speech option
      And I provide the necessary information for the handling of the occurrence
    - **Then** the occurrence is registered
  - **Scenario:** The Citizen provides insufficient information on an active occurrence and is comprehended by the system
    - **Given** that I can access the platform
    - **When** I press the report occurrence button
    - **And** And I press the speech option
      And I fail to provide the basic information on an active occurrence
    - **Then** the assistant request the remaining information
  - **Scenario:** Citizen provides basic information on an active occurrence and is not comprehended by the system
    - **Given** that I can access the platform
    - **When** I press the report occurrence button
    - **And** And I press the speech option
      And I provide the necessary information for the handling of the occurrence
      And the system fails to understand me
    - **Then** the occurrence is not registered
      And I am asked to repeat the occurrence
- **Dependencies:** N/A
- **Priority:** Nice to Have

Note that the functional requirements were written with the front-end in mind, however, this is considered out of scope for this thesis and only the back-end of each requirement will be implemented to support each feature.

4.5 Restrictions

The restrictions impose constraints on how the system will be built, three different types of restrictions were identified and are presented in the present section.

4.5.1 Legal Restrictions

Legal Restrictions relate to the constraints imposed by European Legislation.

- **R01: GDPR Compliance:** As a result of the study of competing products it was defined that the report could be filled with personal information such as the name and e-mail of the reporter. As such, in conformity with the rules imposed by the European General Data Protection Regulation (GDPR)[15], this information should be protected and unavailable to all unessential users.

4.5.2 Business Restrictions

Business restrictions are constraints imposed on a project related to business resources such as the time constraints imposed and the quantity and quality of the team working on the project. Due to the nature of this internship, there is no need to be concerned with restrictions related to the team and therefore the focus should be on time restrictions.

- **R02: Development Time Frame:** The product must be developed by the intern for the duration of the semester.

4.5.3 Technical Restrictions

Technical restrictions are restrictions imposed by which technologies are used, the systems that will be integrated with the product, and the platforms that should be able to interact with the product.

- **R03: Integration with Urban Platform:** The product developed must be integrated with the preexisting solution developed by Ubiwhere.

4.6 Quality Attributes

Quality attributes or non-functional requirements are the guidelines that impact the system's architecture and the functioning of several of the system's features. The quality attributes elicited for this project are presented in this section.

4.6.1 Accuracy

This attribute aims to provide consistency and reliability to the reports generated by the system. The reliability of the information collected by the system is the key to providing the solution with its benefits, it is this information that impacts the process of

the occurrence's resolution and provides more details concerning the occurrence than the ones available to the reporter.

Source of Stimulus	Authority
Stimulus	Needs to have the necessary information for the resolution of the occurrence
Environment	Production
Artifact	System
Response	The system gathers all information required for the occurrence
Response Metric	The report has all fields of additional information related to the occurrence filled

Table 4.1: Accuracy Scenario

4.6.2 Confidentiality

The confidentiality attribute derives from attributes of security regarding users' personal data, ensuring privacy for all users of the system, and keeping any personal information away from all non-essential people.

Source of Stimulus	Citizens
Stimulus	The citizens right to not have their personal information displayed to other users on the platform or third parties
Environment	Production
Artifact	System
Response	Personal data can only be accessed by city administrators
Response Metric	The system omits the citizen's personal data in all reports

Table 4.2: Confidentiality Scenario

4.6.3 Interoperablity

This attribute ensures that the developed system can communicate with the necessary external services. Since the product relies on information gathered from Urban Platform it is essential to ensure seamless communication between the system and Urban Platform.

Source of Stimulus	Report
Stimulus	A report is submitted and needs to gather information from Urban Platform
Environment	Production
Artifact	System
Response	The system communicates successfully with the Urban Platform and gathers the required information
Response Metric	The report is completed with the necessary information

Table 4.3: Interoperability Scenario

4.6.4 Modifiability

This attribute aims to ensure that future work on the product is feasible. Maintaining legacy features while being able to easily work on new features is important, for Ubiwhere's developers to maintain and work on the system it is important to design it and implement it in a way that new features have little impact on the existing ones without much effort.

Source of Stimulus	Developers
Stimulus	Need to implement a new feature
Environment	Maintenance and Development
Artifact	System
Response	The system maintains correct behaviour after the new feature is implemented
Response Metric	The new feature is successfully developed, integrated and tested within its estimated time

Table 4.4: Modifiability Scenario

4.7 Final Overview

In Agile methodologies typically there is no need to have the requirements closed before moving on to the implementation. However, in this instance, good communication with the project owner allowed the requirements to be well developed and approved early in the project. These requirements are always open to change as per the methodology adopted. However, their strong definition at an early stage allowed for a better view of the overall project leading to the implementation. This, in turn, allowed planing the development phase to have a clearer sight of end-goals, making the next phases more focused on the development and with fewer worries regarding requirement changes.

Chapter 5

Architecture and Technology

5.1 Architectural Pattern

In this chapter are presented a concept and design of the project's architecture. It starts by describing the model chosen to represent the architecture followed by the pattern it will follow then the technologies employed to develop the system are presented. Finally, at the end of this chapter, the components of the system's architecture are explained and an analysis of the quality attributes is done to ensure these are met. For the design and representation of the system architecture, the C4 model[16] for software architecture was chosen. This model specifies the procedure for designing the architecture through the use of up to 4 types of diagrams increasing in levels of specification. The model provides an easy way to read and understand the architecture that all stakeholders can understand as well as providing the developers that will implement the system with valuable content. As an additional benefit, this model is well known by Ubiwhere's collaborators which in addition to its ease of understanding will not pose a problem for anyone who must grasp the architecture to analyze it in the future. This model was chosen because of its different levels of complexity, allowing all stakeholders to understand the architecture. Other models for the representation of architectures such as UML can introduce a higher level of complexity for those not familiarized with the model. In addition to its ease of understanding the c4 model is also flexible and if there is a need to detail the system further other models such as UML or Entity-relationship models can be added to add more clarity to the system.

To specify the OMS's architecture two of the C4's diagrams will be used. First, the *System Context* diagram (5.1) will present a high-level view of the different actors that will interact with the system and the external software systems the product makes use of. Then the *Containers* diagram (5.2) is presented, this diagram presents the different components that make up the OMS. These components represent an organized view to better understand which ones provide the necessary functions of the product.

With this it remains to define the overall structure of the architecture; whether it would be monolithic or microservices oriented. These structures are described as:

- **Monolithic:** in this orientation the system is built as a single component and its development, testing, and deployment are done as a whole. This is considered to be the traditional approach, typically easier to develop and deploy a single application that is responsible for every step necessary to complete each function. This approach usually posed fewer problems with tasks such as integration and dependencies while suffering from a lack of modularity, the biggest drawback for this pattern, which

implies that changes in requirements have a greater impact on the system since it is implemented as a single component changes affect the system as a whole, leading to greater development times.

- **Microservices:** in this orientation the system is split into different and independent components that interact between themselves. This granularity of components provides the benefit of isolation, each component works separately from the others, which means that changes in components can be done without affecting the rest of the system. This benefit can also be the biggest problem with the microservices pattern, on one hand, too few components can end up sacrificing the benefits of the pattern, on the other hand, the system's complexity increases with each service added, this rise in complexity can difficult the processes of integration and testing. It is key in this pattern to find the right level of granularity in order to maintain the benefits of the microservices pattern without making the system unnecessarily complex.

Having studied both these patterns it was decided to develop a microservices-oriented architecture. Ubiwhere's desire to study the product and to further develop it is better served with this approach, decoupling the several parts of this system grants modularity to further adapt the product to the company's needs allowing it to make use of the services created at will without needing to change the entire system to achieve the same results.

5.2 Technologies

In this section are presented the technologies chosen to build the OMS and the reason behind each decision. To help determine each technology the following restrictions were considered:

- The chosen technologies must be open-source
- The chosen technologies must have good community support and documentation
- The time spent learning each technology and implementing each component must not compromise the delivery of the project

Having considered the previous restrictions the choices for the technologies some were eliminated, and the research resulted in the following technologies/frameworks as options:

- **Ruby on Rails** - an open-source web MVC framework written in Ruby.
- **Django** - A Python Framework using the MTV[17] (Model-Template-View) framework with a lot of community support.
- **Flask** - a simple and lightweight Python microframework.
- **Spring** - A Open-Source Java Framework, growing in popularity as a full-stack development framework.

The above technologies were considered tools capable of meeting the necessary requirements and within the imposed restrictions. However, as any of them could fit the project and allowed for its completion, Django was elected as the best fit, not only does this framework match all the restrictions it is part of the company tech-stack and brings the following benefits to the project:

- The Django REST Framework allows for the development of a REST API easily without compromising quality, security, and good procedures.

- Python, while not the best regarding performance speeds, enables easy and fast development, debugging, and testability.

- The Django ORM (Object-Relational Mapping) allows for easy interaction with databases. Django is regarded as a powerful technology to support systems with geographical components.

- This framework has great support not only from documentation but also from its community.

As for the remaining technologies, they were also chosen considering the restrictions and considering how they would work with Django. For the Relational Database Management System (RDMS) the choice made was **PostgreSQL**, this open-source technology is already part of Ubiwhere's tech stack and the intern is already familiarized with the technology. This RDMS is also accompanied by good community support and documentation not only for its implementation but for its integration with Django as well. For the building of the reports, there was a need to handle several building requests to meet this need **Celery** was chosen this Python technology is also part of Ubiwhere's knowledge base is open-source and offers RabbitMQ support and proper documentation to achieve so. **RabbitMQ** was the chosen message broker to manage the several requests of report building, it employs an open-source methodology, is part of Ubiwhere's knowledge base, and supports integration with both Django and Celery. For the processing of reports to natural speech the Python library **IBMWatson** a technology integrates several of IBM Cloud services and will allow the usage of the text-to-speech service. This decision compromises one of the restrictions considered to elicit technologies, it is not an open-source technology. However, this choice was made after testing several libraries, such as the Python library pyttsx3 and other APIs, that while open-source did not yield the expected outcomes and quality of the speech. This decision was made in order to meet Ubiwhere's quality requirements. Finally, **Docker** was chosen as the orchestrator of all services, this technology was again chosen due to its presence in Ubiwhere's tech stack and good documentation.

PostgresSQL: is a robust data management system that provides reliability and good performance. An open-source technology supported by the community through many years meets all storing requirements needed for this project.

Docker[18]: is a microservices orchestration technology that allows for the individual implementation of the several components of the system and allows communication between each of them, these components can also be replicated in order to provide elasticity to the system or to replicate them in another context. In addition to this docker allows the use of the same environment for development, quality assurance, and production. This technology utilizes images with the required technologies configured to create each isolated component and provides extensions to orchestrate each component (*docker-compose*) and to replicate components when needed (*docker-swarm*).

RabbitMQ[19]: is an open-source message broker compatible with several programming languages. Its asynchronous messaging functionalities are an important feature for this project which will allow the system to gather data and create reports without impacting its performance. In addition to this, it has easy integration with the other technologies chosen for this project and has many extended functionalities access able through its plugin system developed by the community.

Celery[20]: is an open-source task queue in Python. It utilizes messages to communicate and consists of several workers and brokers that provide high availability and horizontal scaling. These workers constantly check for a new task to perform and are responsible for the building of each report. This technology is regarded as simple but also highly customizable.

IBMWatson[21]: Watson Machine Learning Python client is a library that enables work with IBM's Watson Machine Learning services. This library is used to Authenticate and communicate with IBM's cloud services, it is through this API that the conversion to speech is done.

5.3 Architecture

The main purpose of this system is to provide the necessary endpoints to meet the requirements stated in the previous chapter. For this, it is needed that the server-side component that interacts with the external software system is implemented. The development of all interfaces that would give users a visual response to all requests is considered out of scope.

The following diagram (figure 5.1), the first in the C4 model, shows the relationships between the system, its actors, and the external software systems it interacts with.

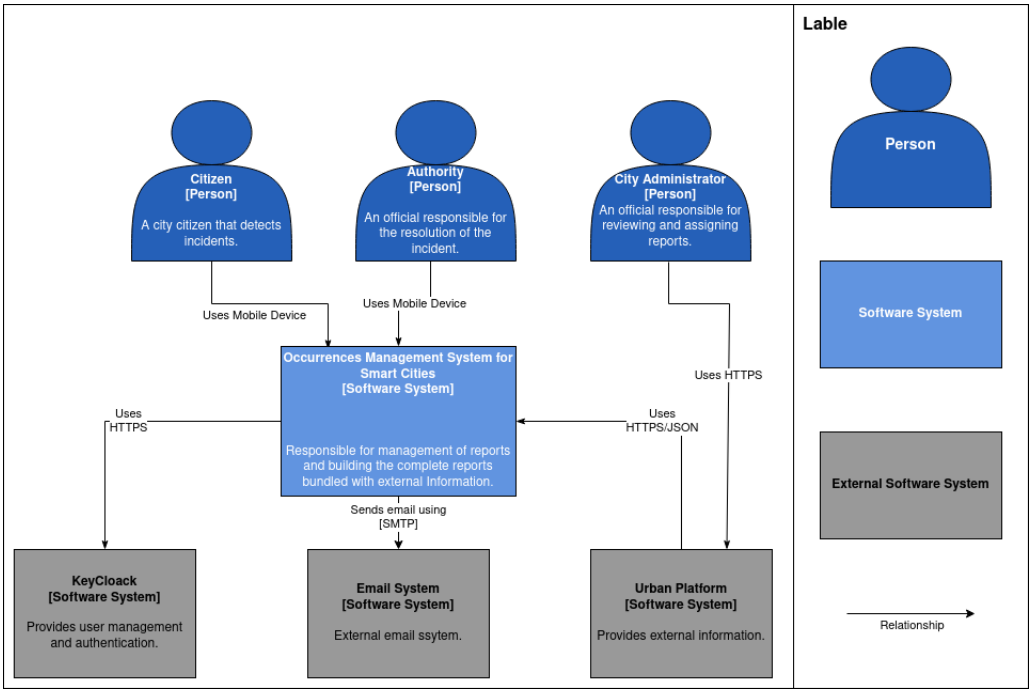

Figure 5.1: Context Diagram of the OMS

In figure 5.2 it is possible to see the whole system and its services more in-depth whose analysis reveals that the architecture is composed of several different containers each responsible for its own set of tasks.

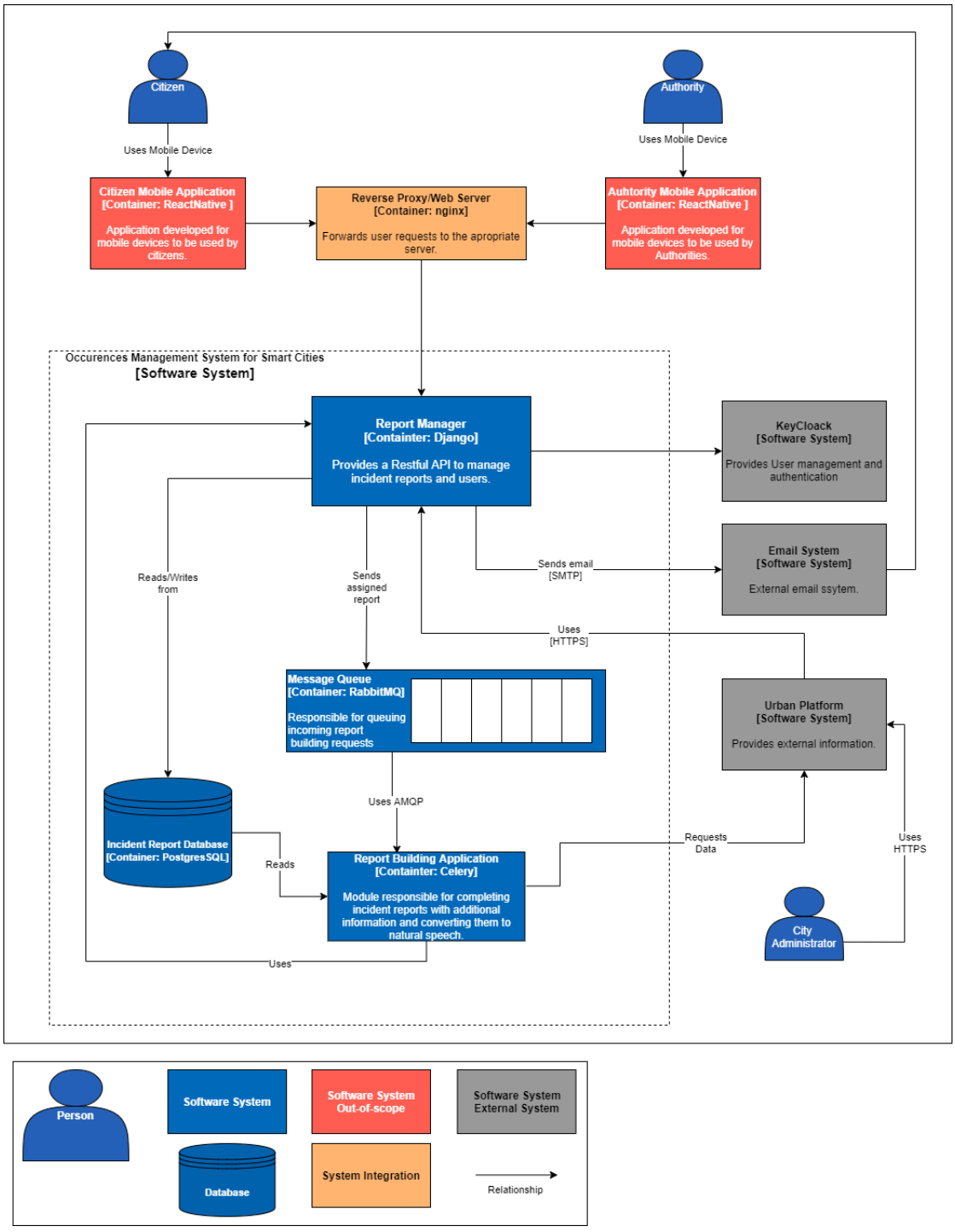

Figure 5.2: Container Diagram of the OMS

Entry Points: The system is composed of three entry points, one for each of the types of users that interact with the System. These entry points are the **Citizen Mobile Application**, the **Authority Mobile Application** and the **Urban Platform**. These systems should be tailored to meet the needs of each type of user. The Citizen Mobile Application will allow citizens to view and create occurrence reports, the Authority Mobile Application should focus on the display of information to minimize user interaction while maximizing the information access, this application aims to provide authorities with information relating to the report they have been assigned to with no effort and allow them to update the occurrence's status. The last entry point for the city administrator is the Urban Platform it is there that management and assignment of reports should be done.

Reverse Proxy: This system is responsible for the handling of requests incoming from the citizen and authority entry points to the Report Manager. The usage of the reverse proxy mandates the need to use HTTPS communications and enables the possibility of implementing load balancing functionalities should the need arise.

Report Manager: This component is responsible for the resolution and forwarding of all requests. It will be responsible for the authentication of authorities using Keycloack, the creation of new reports, assigning users to reports, and starting the report building action when users are assigned to reports. In addition to this, it is this component that will provide functions of listing and sorting reports for user access.

Report Building Application: This component is responsible for the creation of the final report before it is sent to the text to speech module and then to the authorities. Whenever a worker is available a request for the creation of a report is removed from the **Message Queue** and the worker will begin the building of the report. This implies communication with the Urban Platform, it is from there that the information required for each specific type of report will be sent to the worker this information will then be added to the report. The collected data will be filtered to better adapt it to each occurrence and then a message will be built based on the original report and the relevant information gathered, this message will be an overview of the occurrence in natural speech for the authorities to listen to without needing to spend time reading the details on the report.

5.4 Final Overview

The architecture defined and the technologies chosen in this chapter both had in mind the assurance of all requirements stated in the previous chapter. However, in order for the system to work as intended the non-functional requirements were also met. To ensure this the following measures were taken:

- **Accuracy:** Since the product is being developed internally for Ubiwhere it is up to the product owner, represented by João Garcia, to validate if the information added is sufficient for the resolution process.
- **Confidentiality:** Citizens need to provide personal data such as an e-mail for the creation of a report. In order to ensure that their information is kept safe, any personal data not essential for the occurrence's resolution will be omitted to all users that do not need access to the information in order to perform their tasks.
- **Interoperability:** The **Report Building Application** is responsible to ensure that the data received from the Urban Platform can be used in the system, disregarding how the systems operate. Besides building the report this component must

also process the information gathered into a usable format.

- **Modifiability:** This attribute is met by the choice in architectural pattern, as stated before the choice of a microservices oriented pattern gives the ability to work in the several containers composing the system without affecting others as well as adding new ones ensuring that maintaining legacy code does not impact any existing or new components of the system and the creation of new features can be easily done through the usage of a new container that, once again, will not impact the work of the existing containers.

Chapter 6

Development

This chapter elaborates on the environment used for the development of the project and the implementation of each functional requirement. This chapter also details the relevant characteristics of the system and the decisions made during the implementation.

6.1 Environment

As stated in chapter 5, Docker was used to run the different components in a local development system. This platform allows the virtualization of all the components of the system in a controlled environment. The following image (6.2) presents the configurations necessary to run the system. In the image it is visible the configurations of the *handler*, the component with the API responsible for report management, and the *handler_ celery*, the component where the celery tasks responsible for building reports and sending e-mails run. Both components use different contexts with their separate *Dockerfiles*, while the component's isolation could be similarly guaranteed with a single *Dockerfile* two were used to meet the requirements of each component without compromising compatibility with the other component's requirements. These *Dockerfiles*, despite their differences, both use the same Python image from Docker Hub and install the necessary packages for each container according to the requirements. The source code is then replicated into the container and the dependencies are installed. For the container running the database an image of Postgres with Postgis installed was used.

```
1  FROM fmonsanto/python3-gdal:alpine
2
3  ENV PYTHONUNBUFFERED 1
4  WORKDIR /code
5
6  RUN apk update && apk add postgresql-client libffi-dev
7
8  # Add requirements and install them
9  COPY ./requirements /requirements
10 RUN pip install --no-cache-dir -r /requirements/requirements.txt
11 RUN pip install --no-cache-dir -r /requirements/app-requirements.txt
12
13 # Add the code and create static files directory
14 ADD . /code/
15 RUN mkdir -p /code/static
```

Figure 6.1: handler Dockerfile

Figure 6.2: Docker Configurations

Docker represented a challenge for the intern since the intern did not have much practice with the technology and only possessed a basic knowledge of the tool. However, after some time spent on learning the tool, the initial ramp-up was overcome. The usage of Docker and Docker-Compose allowed for better coding practices and ensured that the development during the internship, or any further development after, is not affected by the choice of operating system. With the elimination of these compatibility problems and with this technology being part of Ubiwhere's tech stack it guarantees that there are no barriers for future work on the components.

6.2 Report Manager

The Report Manager is the component responsible for the managerial tasks that the system must perform. This section describes the features implemented for the Report Manager component, including the models used for the occurrence reports, the administration platform implemented and the authentication systems.

6.2.1 Report Management Component

For the implementation of the Report Management component *django-rest-framework* was used. Some Django concepts are key to the development and are explained here to help the understanding of the details described further.

- **ViewSets** - This class combines a set of related views into a single class allowing the quick creation of a CRUD interface for the database[22].
- **Serializer** - are a toll to translate the data structures sent by request into Python data structures[23].
- **Mixins** - The mixin classes provide basic view behaviour without the need of defining handler methods[24].

These tools allow the programmer to define REST endpoints quickly and with few lines of code.

To implement the Report Management Component two GenericViewSets were used. Since tasks such as update and destroy should be done through endpoints protected by authentication and permissions the GenericViewSet used here was configured with the necessary mixins in order to only perform the tasks that were not constrained by these requirements. This set of views allowed the creation of the endpoints needed to create a new report, list all reports, search for reports, and sort reports. The following table lists all endpoints created by this Viewset and which User Stories they fulfill.

Resource	Create (POST)	Read (GET)
List Reports /api/incidents/	-	US03, US12
Search Reports /api/incidents/?serach=<keyword>	-	US05, US14
Sort Reports /api/incidents/?ordering=<keyword>	-	US04, US13
Create Report /api/incidents/	US01	-

Table 6.1: Implemented Methods with no need for authentication

The remaining report management User Stories were exclusive to City Administrators and Authorities, therefore, needed to be restricted by authentication. This time the views needed to create these endpoints were created through another ViewSet configured in order to implement the destroy and update methods. The destroy method was used to remove malicious or mistakenly submitted reports while the update method was overridden in order to, when called, perform the additional functions mentioned next.

In accordance with the User Stories when the report is updated, in order to begin or end resolution, additional tasks need to be carried out. According to the update being done to the report an e-mail is generated and one or two celery tasks are initiated to perform the following jobs when:

- **Begining Resoltion** - two celery tasks are initiated in order to send the generated update e-mail to the citizen and to initiate the process of complementing the report with additional information and converting it to natural speech;
- **Finalizing Resolution** - a single celery task is initiated in order to send the generated update e-mail to the citizen.

In order perform theses jobs automatically, when the occurrence is assigned or is status is changed, the update method was overridden. This is also protected by the user Groups to ensure that only City Administrators can begin the resolution phase or archive the report and only Authorities can set the occurrence as resolved.

6.2.2 Models

Two models were used to support the system, the user model and the report model. For the first one the base Django user model was used, which already provides the necessary features needed to store user information, manage users and define their roles and permission within the platform.

The report model is a custom Django model created to store all the necessary information regarding the occurrences. This model is composed by the following fields:

- **Severity:** Severity of the occurrence being reported;
- **Incident Type:** Type of occurrence being reported;
- **Location:** Location of the occurrence;
- **Description:** Description of the occurrence;
- **Time of Incident:** Time when the occurrence took place;
- **Attachments:** Media file related to the occurrence (optional);
- **Reporter Name:** Name of the person reporting the occurrence (optional);
- **Reporter E-mail:** Contact e-mail of the person reporting the occurrence (optional);
- **Status:** Status of the occurrence's resolution;

6.2.3 Administration Platform

Managing reports and resolving occurrences is not possible without first registering users on the platform. Allowing users to create their own accounts in an product like this can have its disadvantages, the administrator can only give permissions if the user is valid. A mistake in this process could threaten the integrity of the information in the system. To ensure that this problem could not occur user registration is done by a system administrator in the administration platform.

Through this platform occurrence reports can also be handled, users with access to this platform can change report information, create new reports and delete existing reports. These management tasks available in the administration platform were created with the intent of being used to fix mistakes and problems with occurrences' information. For example if an occurrence report is incorrectly assigned, or its resolution was accidentally started, it is possible to revert these changes in the administration platform. However, changes in occurrence status done in this platform will not trigger the celery tasks responsible for building reports. To make performing these jobs easier, if there is a need to navigate through the reports to change them in anyway it is possible to filter and sort them. Also in this platform it is possible to access the status update e-mails sent by the system in order to see the status of each e-mail and errors if the system fails to send them.

6.2.4 Authentication

As stated in chapter 5 user management was supposed to be handled by Keycloack. However, since there is no mature integration with *django-rest-framework* and Ubiwhere's

own implementation not being prepared to handle a new project, the usage of Keycloak would not be possible without compromising the project's deadline. This situation was foreseen and accounted for during the Risk Management (table 3.1) identified as risk R06 whose mitigation plan was activated. To replace Keycloak, Django user models, Django groups, and JSON Web Tokens (JWT) were used. As it was mentioned previously the creation of users is done through the administration platform, where users are created and given a username and password they can use to login into the system. Upon entering a valid username and password combination the system will provide the user with a JWT which should be used in the header of all requests made from this point on. In figure 6.3 examples of requests made with valid and invalid JWT are presented.

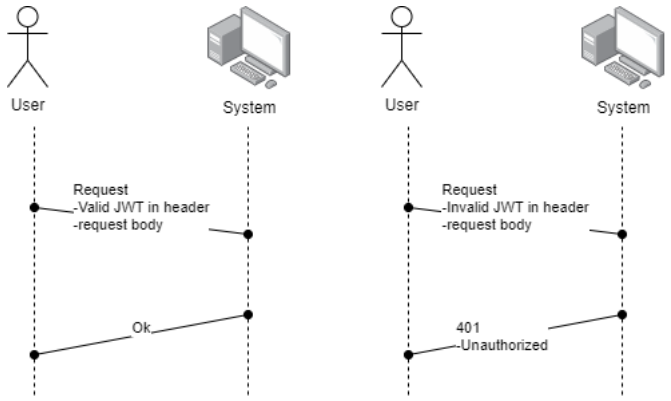

Figure 6.3: Request Examples

6.2.5 Celery Tasks

For this project two tasks were created. The first task is called whenever a status update e-mail is generated in order to send it to the occurrence report. The second task is used to build the final report that is presented to the Authority. This task uses the report model described in this section to collect the information that is relevant to the occurrence and transforming the complete report into natural speech. The second tasks is better detailed in section 6.3.

6.3 Report Building Application

This section further elaborates on the Report Building Component and the work done by the task.

6.3.1 Construction of the Report

In order to build a more complete report firstly the task queries the database to get all information on the occurrence whose report is being built. Then the task must request information from the Urban Platform. For this an authentication request is done in order to get the access token to authenticate in all further requests. With the access token, three requests will be made. In the context of this project the version of the Urban Platform used was the one deployed in Barcelona, and the following data is requested:

- **Air Quality** - Index rating the quality of the air and the particles that influence

this rating;

- **Traffic** - The status of all roads monitored by the platform, rated on the chance of jam in 0 to 10;

- **Events** - These events are events that impact traffic in the area such as holes in the road or closed roads;

The first request to be made relates to air quality, the request is made to receive only the latest readings from all stations, after this the readings from the closest station to the occurrence's location are selected. The next request made is to get traffic data, all the streets are collected and then filtered by distance to the event and level of traffic, a list of all streets with conditioned traffic is returned. Finally, Event that may impact the resolution process are collected. This information is also filtered by event severity and distance to the occurrence and a list of events that are encompassed in the acceptable distance and severity is returned. For both traffic and events, a radius of 500 *meters* was considered as the distance in which these factors may impact the resolution process.

This information will be given to the authorities in their complete report and is used in order to create a message. This message is what the Authority will listen to after it is converted to natural speech. It includes the street name where the occurrence took place, the type of occurrence, and its description. After this the air quality information is added, including the Air Quality Index (AQI). The AQI is directly received from the Urban Platform where it is calculated using the readings of four dangerous particles, *NO2*, *SO2*, *O3* and *PM10*. If the received AQI passes the level where the air quality is considered unhealthy, the reading of the particles that are causing this deterioration are also added to the message. Following this the traffic information is added. To pass the message as efficiently as possible the task firstly adds all streets whose traffic is considered "intense" by the Urban Platform and then the streets whose flow is considered "conditioned". Finally, if the events are considered "Major", they will be listed in the message, naming explicitly their type and the address. The following image shows the task performing the report building.

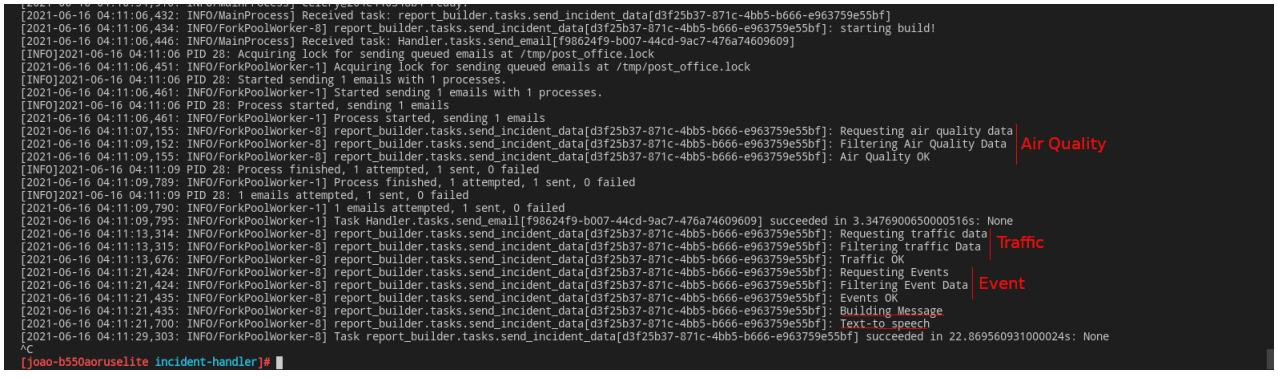

Figure 6.4: Report Building Application Task

6.3.2 Text-to-Speech Conversion

With the message to be sent to the Authorities built, the task needs to convert it to natural speech. IBMWatson was used to communicate with the text-to-speech service provided by IBM Cloud Services.

```python
def speak(message,name):
    authenticator = IAMAuthenticator(os.environ.get('IBM_KEY'))
    text_to_speech = TextToSpeechV1(
        authenticator=authenticator
    )
    text_to_speech.set_service_url(os.environ.get('IBM_URL'))
    path = "incident_"+str(name)+".mp3"
    audio = text_to_speech.synthesize(message,accept="audio/mp3").get_result().content
    with open(path,"wb") as audiofile:
        audiofile.write(
            text_to_speech.synthesize(message,
            accept="audio/mp3").get_result().content
        )
```

Figure 6.5: Usage of IBMWatson

IBM Cloud Services provide a key to use in the IAMAuthenticater from IBM's SDK core, this will authenticate the user for all further requests made. Then with the URL also provided by the Cloud Services, the text-to-speech service is set. All that remains is to make the request to the API to convert the message into natural speech, the audio file is then stored to be sent to the Authorities.

6.4 Final Considerations

The development of the product was perhaps the biggest challenge of this internship. Much of the technologies and strategies used during this phase were either new to the intern or the experience with them was low. However the difficulties were overcome and all the components needed to ensure the requirements were implemented. The table 6.2 presents the functional requirements implemented.

User Story	Priority	Implemented
US01	Must Have	Yes
US02	Should Have	Handled by the frontend
US03	Must Have	Yes
US04	Must Have	Yes
US05	Must Have	Yes
US06	Must Have	Yes
US07	Must Have	Handled by the frontend
US08	Must Have	Yes
US09	Must Have	Yes
US10	Must Have	Handled by the frontend
US11	Should Have	Handled by the frontend
US12	Must Have	Yes
US13	Must Have	Yes
US14	Must Have	Yes
US15	Must Have	Yes
US16	Must Have	Yes
US17	Nice to Have	No

Table 6.2: Implemented User Stories

As it was mentioned previously the User Stories were created with the frontend of the product in mind with the goal for this internship being the development of the backend

to support it. As it is detailed on table 6.2 most of the "Must Have" user stories were developed, only two were not implemented. The remaining user stories with "Must Have" priority, as well as the ones with "Should Have", were stories that only made sense in the context of the frontend. In more detail the missing user stories with these priorities regarded the logout from the platform and the display of occurrences on a map. Both of these types of action are actions typically performed on the frontend therefore, they were not developed during this internship. Nonetheless, these missing requirements that should be handled by the frontend have no barriers for their implementation and the system is capable of behaving correctly with these added features.

Chapter 7

Testing

In this chapter the test done to the system are described and the results presented. In order to ensure the quality of the implemented features a unit testing suite was developed. Other testing strategies were considered such as end-to-end testing or load testing. These tests were not performed because at this stage of the project it would not be worth addressing them. Finally the code covered by the tests is presented.

7.1 Unit Testing

This software testing method's goal is to isolate small portions of the system and verify that they work as intended. These tests are performed in an isolated environment where a simulated input is given and the response is evaluated against the expected outputs. The following subsections detail how each component was tested.

7.1.1 Report Manager

In this component a group of tests were performed on the API for managing reports and users. The goal of these tests were to access the following:

- The system only accepts Authenticated requests (when authentication is needed);
- The system correctly behaves given the different roles;
- The system only creates or updates occurrences when the data is validated;

The following table (7.1) presents all tests performed on this component.

ID	Description	Passed
U01	Logging in with correct credentials should be accepted	Yes
U02	Logging in with incorrect credentials should be rejected	Yes
RM01	Creating a report with all fields provided should be accepted	Yes
RM02	Creating a report with only required fields should be accepted	Yes
RM03	Creating a report without required fields should be rejected	Yes
RM04	Creating a report with no fields provided should be rejected	Yes
RM05	Updating a report with valid information as a City Administrator should be accepted	Yes
RM06	Updating a report with valid information as an Authority should be accepted	Yes
RM07	Updating a report with no information should be rejected	Yes
RM08	Requesting all reports should be accepted	Yes
RM09	Requesting all reports with no reports created should be accepted	Yes
RM10	Searching for a report with reports created should be accepted	Yes
RM11	Searching for a report with no reports created should be accepted	Yes
RM12	Sorting reports with reports created should be accepted	Yes
RM13	Sorting reports with no reports created should be accepted	Yes
RM14	Fetching a report that exists should return its information	Yes
RM15	Fetching a report that does not exist should return and empty result	Yes
RM16	Deleting a report should be accepted	Yes
RM17	Deleting an nonexistent report should return an empty result	Yes

Table 7.1: Report Manager - Unit Tests

7.1.2 Report Builder

The Report Builder component was tested to ensure that the data received from the Urban Platform was correctly filtered and the calculated distances are correct. The responses from the Urban platform were simulated to avoid doing requests to this platform during the execution of the tests. Table 7.2 presents the tests done to this component.

ID	Description	Passed
RB01	Calculating the distance between to coordinates should return the correct result	Yes
RB02	Filtering Air Quality data should return the data nearest to the occurrence's location	Yes
RB03	Filtering Traffic data should return all roads within the acceptable radius with moderate or worst jam factor	Yes
RB04	Filtering Events should return all roads within the acceptable radius and "Major" severity	Yes

Table 7.2: Report Builder - Unit Tests

7.2 Code Coverage

Code coverage is a measurement of how many lines of code are executed during the execution of the tests. For this the *coverage* Python library was used, this tool not only gives the percentage of lines of code covered during the test it also registers which lines are not visited.

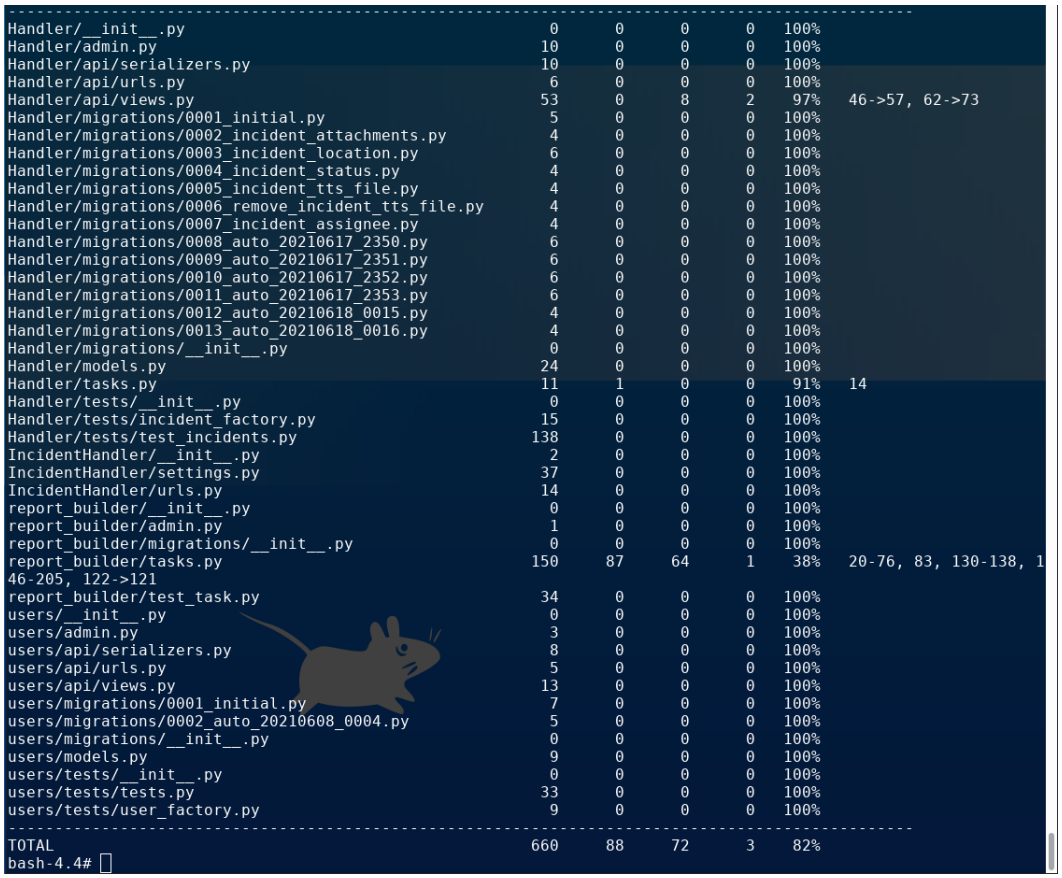

Figure 7.1: Code Coverage Results

As it is shown in figure 7.1 the tests performed cover 82% of the developed code. The exact percentage of code that should be covered is a theme that is still discussed nowadays and no concrete answer exists. While complete coverage is always desired this goal is not only unrealistic, because it depends on the quality and extent of the tests. Despite avoiding some test cases reaching a coverage of 82% is positive since most projects aim for 70-80% coverage across the whole project[25].

7.3 Final Considerations

The testing phase is crucial for any project. This phase ensures that the developed product is formally tested to ensure the fulfillment of the requirements and the quality of the developed product. For this thesis the goal of the tests was to validate the endpoints created to meet the requirements and if these worked correctly in all possible situations. After the testing was complete it was ensured that the endpoints created could perform all "Must Have" requirements and could meet all "Should Have" requirements when the frontend of the product is developed.

Chapter 8

Conclusion

This chapter concludes the report of this internship. Firstly there is a summary of the conclusions drawn from the work done, followed by the lessons learned during this internship and concluding with the work the could be done to further improve the product.

8.1 Work Done

This internship explored the theme of occurrence management and the challenges cities face in order to efficiently handle this issue. It was understood that communication and information were the focus points that could be improved to make reporting and resolving occurrences a more efficient process. With a clear understanding of the problem to be tackled an analysis of the work done to improve occurrence management was conducted in chapter 2. In this analysis it was possible to get a better grasp of the existing solutions for the problem and what part of it they focus on. Each of this products brought advantages to one or more issues regarding the subject, ranging from improving communication with all parties involved in occurrence handling to improving the management aspect. Communication and its quality is an important part of the process, but equally important is the information being shared. For this a study was done on how information gathering could improve the overall process and how its delivery to each actor can impact the quality and efficiency of occurrence resolution. The OMS was then planned in order to create means of communication between cities and its citizens and deliver better information to those who are charged with resolving each occurrence.

In chapter 3 the Success Criteria was established. This stated that the requirements, specified in 4, and the architecture, specified in chapter 5 must be approved by the company. While ensuring that he backend that would fulfill the needs of all "Must Have" requirements must be developed by the end of the internship and that the restrictions and quality attributes must be respected. In chapter 1 a set of goals were also established, stating that the product needed to receive new reports, provide the necessary management functions to City Administrators and collect additional information to be delivered to the Authorities. Reflecting on the product developed during this internship it is possible to say that both the goals and Success Criteria established were fulfilled making this internship a success.

8.2 Lessons Learned

This internship provided opportunities to learn new skill and work on and solidify existing skills. Ubiwhere has been a great influence in this learning process, through their feedback on the work done many lessons were learned. The trust deposited on the intern to make his own decisions and work independently gave the intern a greater sense of responsibility which lead to many existing skills being improved.

The internship provided a chance to work on soft-skills such as communication skills through the several communications inside of a company as well as giving the opportunity to learn how to negotiate his decisions and opinions with each discussion with the advisors. The writing of an academic deliverable was by far the most important hard-skill gained as this process involves a critical analysis to ensure it delivers the intended message and makes use of preexisting skills. These preexistence skills also make up the set of hard-skills the intern had a chance to work on, these skills learnt during the Masters Course such as requirements elicitation and architectural design were progressively improved during the first stage of this internship.

8.3 Future Work

Although the internship is considered a success and the system is developed, the OMS can still be improved. The first aspect of improvement is the development of the frontend of the application, as an interface is necessary for users to interact with the system. Next, an external validation of the reports created could be done. And finally, as more information is added to the reports, a revision to the report's message building process could also benefit the product to ensure that the quality is maintained without overloading the Authorities with information.

All of this work can lead not only to the betterment of the developed product but also yield greater benefits to occurrence resolution. The validation of the report with the authorities that will eventually be responsible for resolving the occurrences can provide a better view of the next steps for the product. With end-user feedback it is possible to identify new information that can be added to the report. This extra information should, in turn, lead to a more refined message building process that is better tailored to handle the new information added. The interface developed would then bring all information stored and generated by the platform together in an a way that users can easily interpret and access.

References

[1] U. Nations, *Press release on population*, https://population.un.org/wup/, 2018.

[2] E. F. Z. Santana, A. P. Chaves, M. A. Gerosa, F. Kon, and D. Milojicic, "Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture," 2017. arXiv: 1609.08089 [cs.CY].

[3] Ubiwhere, *Urban platform*, https://urbanplatform.city/, 2018.

[4] C. C. tech Cluster, "Danish smart cities : Sustainable living in an urban world : An overview of danish smart city competencies," 2012.

[5] IBM, *Ibm intelligent operations center*, https://www.ibm.com/developerworks/industry/library/ind-intelligent-operations-center/index.html, 2012.

[6] C. M. de Lisboa, *Informação escrita do presidente*, https://www.am-lisboa.pt/documentos/1529934530I0wDR8zb1Yy52HT0.pdf, 2018.

[7] França, N. Araújo, A. Gomes, N. Cacho, F. Lopes, J. Lima, and E. Adachi, "Sigoc: A smart campus platform to improve public safety," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020.

[8] ESRI, *Arcgis9: What is arcgis*, http://downloads.esri.com/support/documentation/ao/698WhatisArcGis.pdf, 2018.

[9] ——, *Introduction to citizen problem reporter*, https://doc.arcgis.com/en/arcgis-solutions/reference/introduction-to-citizen-problem-reporter.htm.

[10] FixMyStreet, *Fix my street*, https://fixmystreet.org/.

[11] A. v. B. A. C. W. C. M. F. J. G. J. H. A. H. R. J. J. K. B. M. R. C. M. S. M. K. S. J. S. Kent Beck Mike Beedle and D. Thomas, *Manifesto for agile software development*, http://agilemanifesto.org/, [Online; accessed 10-June-2021], 2001.

[12] J. Sutherland, *Why gantt charts were banned in the first scrum*, https://www.scrum.org/, [Online; accessed 21-May-2021], Feb 12, 2006.

[13] *Should you use gantt charts in agile project management?* https://content.intland.com/blog/project-management-en/should-you-use-gantt-charts-in-agile-project-management, [Online; accessed 8-June-2021], December 04, 2020.

[14] M. Keeling, *Reflections on software engineering*, https://www.neverletdown.net/2010/01/threshold-of-success.html, [Online; accessed 21-May-2021], January 15, 2010.

[15] E. Parliament, *Directive 2014/45/ec of the european parliament and of the council.*

[16] *The c4 model for visualising software architecture*, https://www.c4model.com/, [Online; accessed 17-January-2021].

[17] A. N. Oyom, *"understanding the mvc pattern in django"*, https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f, [Online; accessed 15-February-2021], Jul 19, 2017.

[18] *Docker*, https://www.docker.com/, [Online; accessed 17-January-2021].

[19] *Rabbitmq*, https://www.rabbitmq.com/, [Online; accessed 17-January-2021].

[20] *Celery - distributed task queue*, https://docs.celeryproject.org/en/stable/getting-started/introduction.html#what-s-a-task-queue, [Online; accessed 17-January-2021].

[21] https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/python-client.html, [Online; accessed 12-May-2021], Jan 04, 2021.

[22] django-rest framework, *"viewsets"*, https://www.django-rest-framework.org/api-guide/viewsets/#viewsets, [Online; accessed 13-March-2021].

[23] ——, *"serializers"*, https://www.django-rest-framework.org/api-guide/serializers/, [Online; accessed 17-March-2021].

[24] ——, *"mixins"*, https://www.django-rest-framework.org/api-guide/generic-views/#mixins, [Online; accessed 17-March-2021].

[25] S. Cornett, *"minimum acceptable code coverage"*, https://www.bullseye.com/minimum.html, [Online; accessed 25-June-2021].