

Universidade do Minho
Escola de Engenharia

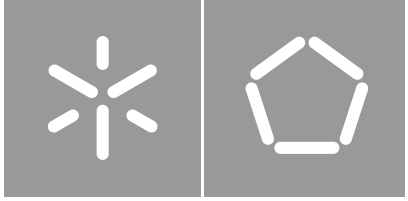
Carolina Resende Marques

Multiplatform Application for Monitoring Services in a Hospital Environment

Multiplatform Application for Monitoring Services in a Hospital Environment
Carolina Marques

UMinho | 2022

September, 2022



Universidade do Minho
Escola de Engenharia

Carolina Resende Marques

Multiplatform Application for Monitoring Services in a Hospital Environment

Master's Dissertation
Informatics Engineering

Work developed under the supervision of:
Doctor Hugo Daniel Abreu Peixoto

September, 2022

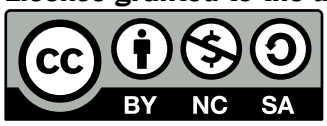
COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositoriUM of Universidade do Minho.

License granted to the users of this work



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International

CC BY-NC-SA 4.0

https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en

Acknowledgements

First and foremost, I would like to thank Hugo Peixoto for giving me the opportunity to develop a project in an area that he knew I was passionate about, for the advice and guidance provided over the last year, and for never allowing me to doubt my work.

Words cannot express my gratitude to my wonderful work partner and friend, Vasco Ramos, for being a major source of support over these years. Vasco Ramos, in parallel to this work, developed the back-end architecture for this solution, which allows metrics to be collected and managed across multiple institutions.

While not directly contributing to this work, I'd also like to thank Constança Elias, Maria Araújo, Pedro Machado, and Renata Ribeiro, who have shown their support throughout my academic career and helped me rediscover my love for coding.

I must express my heartfelt appreciation to my family. To my parents, Conceição and Fernando, for their sacrifices in educating and preparing me for the future, as well as for their unwavering support and encouragement throughout my years of study. To my grandparents, Armando, Carolina, and Rosa, for always being a major source of inspiration and being proud of me and my accomplishments. I have to thank Miguel, my boyfriend, for always being by my side, for all of his love and support. Thank you for always having faith in me and my dreams. Finally, I'd like to thank my brother Paulo for always encouraging me when I needed it the most and pushing me forward.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

“Truth can only be found in one place: the code.” (Robert C. Martin)

Abstract

Multiplatform Application for Monitoring Services in a Hospital Environment

Many people believe that information technology has the potential to change the way the healthcare industry approaches its current challenges by improving healthcare quality, safety, and efficiency by bringing decision support to the point of care and enabling routine quality measurement. In the medical field, healthcare information technology refers to any information technology tool or software that is intended to increase hospital and administrative productivity, provide new information about medications and treatments, or improve overall quality of care. Infrastructures in hospitals must manage both information technology and specialised healthcare systems and protocols. It is important in this type of structure to ensure that all operations and information technology run smoothly and one of the ways to achieve this is by continuously and automatically monitor the hospital environment's systems.

The right monitoring and reporting tools can help keep medical staff efficient without worrying about failing systems, provide visibility into usage trends, equipment performance, downtime, and much more, saving time and resources. A monitoring application of this type is regarded as an important source of information.

The main goal of this Dissertation within the scope of this project is to develop a monitoring web platform for healthcare information technology administrators that is based on a multi-site and multi-organisation scheme. The proposed solution will monitor various hospital services and is expected to provide the current state of the system in a timely manner through a set of graphs and reports, allowing for appropriate operational decisions and ensuring that the system functions as expected.

A web application for monitoring hospital services was developed, implemented, and evaluated during the dissertation work period. Using questionnaires, the platform was evaluated and validated in order to understand if this approach may improve information technology availability and, in the long run, alleviate some of the healthcare industry's pains. A formal evaluation of the solution was also performed, which comprised a strengths, weaknesses, opportunities and threats analysis and a risk assessment report, both of which gave helpful insights into the system's strengths and shortcomings, as well as potential improvement areas.

Keywords: Information Technology, Monitoring Systems, Web Application, Healthcare

Resumo

Aplicação Multidispositivo de Monitorização de Serviços em Ambiente Hospitalar

Muitas pessoas acreditam que a tecnologia da informação tem o potencial de mudar a forma como a indústria da saúde aborda os seus desafios atuais, melhorando a qualidade, segurança e eficiência dos cuidados de saúde, levando o apoio à decisão ao ponto dos cuidados e permitindo a medição da qualidade de forma rotineira. No campo médico, a tecnologia da informação sobre cuidados de saúde refere-se a qualquer ferramenta ou software informático que se destine a aumentar a produtividade hospitalar e administrativa, fornecer novas informações sobre medicamentos e tratamentos, ou melhorar a qualidade global dos cuidados de saúde. As infraestruturas dos hospitais devem gerir tanto as tecnologias da informação como os sistemas e protocolos especializados de cuidados de saúde. É importante, neste tipo de estrutura, assegurar que todas as operações e tecnologias de informação funcionem sem problemas e uma das formas de o conseguir é monitorizando-a de forma contínua e automática.

A escolha de ferramentas de monitorização e relatórios adequadas podem ajudar a manter o pessoal médico eficiente sem se preocupar com falhas nos sistemas, dar visibilidade às tendências de utilização, desempenho do equipamento, tempo de inactividade e muito mais, poupando tempo e recursos. Uma aplicação de monitorização deste tipo é considerada como uma importante fonte de informação.

O principal objectivo desta Dissertação no âmbito deste projecto é desenvolver uma plataforma web de monitorização para administradores de tecnologias de informação na área da saúde que se baseie num esquema multi-site e multi-organização. A solução proposta irá monitorizar vários serviços hospitalares e espera-se que forneça o estado actual do sistema em tempo útil através de um conjunto de gráficos e relatórios, permitindo decisões operacionais adequadas e assegurando que o sistema funciona como esperado.

Foi desenvolvida, implementada e avaliada uma aplicação web para monitorizar os serviços hospitalares durante o período de trabalho de dissertação. Utilizando questionários, a plataforma foi avaliada e validada de modo a compreender se esta abordagem pode melhorar a disponibilidade das tecnologias da informação e, a longo prazo, aliviar algumas das dores da indústria da saúde. Foi também realizada uma avaliação formal da solução, que incluiu uma análise de pontos fortes, pontos fracos, oportunidades e ameaças e um relatório de avaliação de risco, ambos dando uma visão útil dos pontos fortes e fracos do sistema, bem como de potenciais áreas de melhoria.

Palavras-chave: Tecnologia da Informação, Sistema de monitorização, Aplicação Web, Saúde

Contents

List of Figures xi

List of Tables xiii

Acronyms xv

1 Introduction 1

1.1 Motivation . . . 2
1.2 Goals . . . 3
1.3 Challenges . . . 4
1.4 Document Structure . . . 4

2 State of the Art 6

2.1 Information Technology . . . 6
2.2 Information Technology in Healthcare . . . 7
2.3 Information Systems . . . 8
2.4 Information Systems in Healthcare . . . 9
2.5 UI/UX Process . . . 9
2.6 Monitoring Applications . . . 11
2.6.1 Site24x7 . . . 11
2.6.2 PRTG . . . 12
2.6.3 Nagios . . . 13
2.6.4 Datadog . . . 15
2.6.5 Comparison . . . 16

3 Research Methodology and Tools 17

3.1 Design Science Research Methodology . . . 17
3.2 Technologies and Tools . . . 19
3.2.1 Progressive Web App . . . 19
3.2.2 Notifications . . . 20
3.2.3 Giraffe . . . 20

3.2.4 JWT . . . 21

4 Background **22**

4.1 React . . . 23
4.2 Angular . . . 23
4.3 Vue . . . 24
4.4 Svelte . . . 24
4.5 Background Overview . . . 25

5 Proposal **26**

5.1 Technical Evaluation . . . 26
5.2 Use Cases . . . 29
5.3 Functional and Non Functional Requirements . . . 31
5.4 Architecture . . . 32

6 Implementation **34**

6.1 Login . . . 34
6.2 Platform Template . . . 35
6.2.1 Navbar . . . 36
6.2.2 Menu . . . 38
6.3 Agents Overview . . . 39
6.4 Organisation Management . . . 39
6.5 Add Agent . . . 40
6.6 Agent Page . . . 43
6.7 PWA . . . 49

7 Evaluation **51**

7.1 Usability and User Satisfaction Evaluation . . . 51
7.1.1 Observation Reports . . . 52
7.1.2 Results . . . 52
7.1.3 Discussion . . . 53
7.2 SWOT Analysis . . . 54
7.3 Risk Assessment . . . 57

8 Conclusions and Future Work **59**

8.1 Main Contributions . . . 59
8.2 Future Work . . . 61
8.3 Scientific Contribution . . . 61

Bibliography **63**

Appendices

A First Questionnaire **68**

B Second Questionnaire **70**

C Publications **74**

C.1 Pervasive Monitoring System for Services and Servers in Healthcare Environment . . 74

C.2 IT Monitoring in Healthcare: A Case Study 75

List of Figures

1 IT components and functions [Adapted from [10]] . . . 7
2 Structure of organisational IS [Adapted from [17]] . . . 8
3 Design thinking process [Source [24]] . . . 10
4 Site24x7 Dashboard . . . 12
5 PRTG Dashboard . . . 13
6 NagiosXL Dashboard . . . 14
7 Datadog Dashboard . . . 15

8 DSR Methodology Process Model [Adapted from [31]] . . . 18
9 CPU values over time, an example plot using Giraffe . . . 21

10 Front-end frameworks experience ranking . . . 23

11 Percentage of participants that actively work for a company that actively employs monitoring systems . . . 27
12 Average and median response to the affirmations . . . 28
13 Web app use-case diagram . . . 31
14 High-level architecture . . . 33
15 Architecture of the proposed monitoring solution . . . 33

16 Login page . . . 35
17 Platform's template . . . 35
18 *Navbar* sub-components . . . 36
19 Create account form . . . 36
20 FAQ page . . . 37
21 Notification list . . . 37
22 Menu component . . . 38
23 Agents overview page . . . 39
24 Create organisation *dialog* . . . 40
25 Delete organisation dialog . . . 40
26 Create agent first step . . . 41

27 Create agent second step 42
28 Create agent error 43
29 Agent page 43
30 Create alert page 44
32 Empty agent alerts page 45
33 Agent alerts page with alerts 45
34 Edit alert dialog 45
35 Edit agent dialog 46
36 Agent information dialog 46
37 Statuses timeline 47
38 Statuses timeline dialog 47
39 Graph Visualisation 48
40 Average response graph from June 15 to July 15 49
41 Web app in a mobile phone 50

42 Average response to the affirmations 54
43 SWOT table [Adapted from [54]] 55
44 Platform SWOT analysis table 57

List of Tables

1 Software Comparison 16

2 Comparison between React, Vue, Angular and Svelte 25

3 Risk assessment results 58

List of Listings

1 Levels example 42
2 POST example of query when user wants to see the average response graph of an agent 49

Acronyms

API Application Programming Interface 3, 12, 13, 15, 19, 20, 21, 22
AWS Amazon Web Services 12

CHTS Centro Hospitalar do Tâmega e Sousa 2, 26, 51, 60
CPU Central Processing Unit xi, 12, 16, 21, 24

DK Design Knowledge 17
DOM Document Object Model 24, 25
DSR Design Science Research xi, 17, 18
DSRM Design Science Research Methodology 17, 18

EHR Electronic Health Record 9, 11
EMR Electronic Medical Record 15

FR Functional Requirements 31

HIS Health Information System 9, 11
HIT Health Information Technology 1, 7
HL7 Health Level-7 13
HTML HyperText Markup Language 23
HTTP Hypertext Transfer Protocol 31

IDE Integrated Development Environment 25
IS Information System xi, 4, 6, 8
IT Information Technology xi, 1, 2, 3, 4, 6, 7, 8, 11, 12, 13, 14, 15, 16, 32, 54, 59, 60

JSON JavaScript Object Notation 21
JWT JSON Web Token 21, 55

MVVM Model-View-ViewModel 24

NFR Non Functional Requirements 31, 32

PWA Progressive Web App 19, 20, 49

REST Representational State Transfer 12, 13

RPM Remote Patient Monitoring 7

SIEM Security Information and Event Management 29

SMS Short Message Service 20, 61

SPA Single Page Application 22, 24

SWOT Strengths, Weaknesses, Opportunities, and Threats xii, 51, 54, 55, 56, 57, 60

UI User Interface 2, 4, 5, 9, 20, 23, 24, 35, 53, 54, 60

UX User Experience 4, 9, 10, 25, 32, 53, 54, 60

Chapter 1

Introduction

The healthcare industry is under pressure to improve patient safety, operate more efficiently, reduce medical errors, and provide secure access to timely information while keeping costs under control, protecting patient privacy, and adhering to legal guidelines. This means that they are currently confronted with numerous challenges, but many believe that Information Technology (IT) can cause a paradigm shift.

Health Information Technology (HIT) is the use of information processing involving both computer hardware and software (ranging from simple charting to more advanced decision support and integration with medical technology) to store, retrieve, share, and use healthcare information. HIT is viewed as a means of improving healthcare quality, safety, and efficiency, as well as lowering operational costs and medical errors. IT offers new ways to boost patient confidence and reinforce patient trust in caregivers and healthcare facilities in the healthcare setting [1–3].

With this, and given its importance in the healthcare context, it is critical to make every effort to ensure that IT systems fail as little as possible because failures in this industry are frequently associated with life-or-death situations, resulting in higher mortality rates, poorer treatment outcomes, and lower patient trust and satisfaction. Therefore, monitoring these systems becomes a priority. As a result, this master's project consists in developing a web platform for monitoring various hospital services. This platform's core features include multi-platform compatibility and a multi-organisation scheme, implying that monitoring can be performed across multiple computing platforms and across multiple healthcare institutions.

In this light, the following questions arise: What constitutes a good IT monitoring tool? Can a multi-platform and multi-organisation monitoring application enhance IT availability in the healthcare industry? on top of others. As a result, the goal of this Dissertation is to answer and comprehend the above-mentioned concerns with an emphasis on a multi-organisation and multi-platform healthcare setting, give a systematic review of solutions for service monitoring in healthcare settings and the development of a

web application that allows hospital IT administrators to view and monitor services in several hospitals. The solution's primary goal is to ensure that hospital operations run smoothly and to improve overall IT availability. This work is part of a larger project developed by a team in collaboration with Vasco Ramos, who is in charge of the back-end architecture.

It is worth noting that by responding to two questionnaires at two separate occasions, the healthcare IT administrators at Centro Hospitalar do Tâmega e Sousa (CHTS) contributed to this dissertation results. The first moment occurred during the platform's early stages, with the purpose of knowing how various hospital IT administrators feel about monitoring solutions and how they believe it should function and aid them on a regular basis. This provided a clearer picture of the existing problems and what the solution should incorporate in order to solve them. On the other hand, the second moment took place following platform development to test and validate the developed solution. This collaboration made it possible to use a real-world case study to demonstrate how the work of this dissertation can be applied.

This chapter will discuss the driving forces behind the project, the objectives that were established, and finally the document structure.

1.1 Motivation

The dissertation seeks to create an User Interface (UI) that allows hospital IT administrators to monitor the several hospital services, thereby enhancing IT availability and decreasing overall response time to failures. The healthcare industry is highly heterogeneous, widely distributed, and fragmented, with a high degree of autonomy [4], and technologically progressing from isolated software systems to solutions that support a continuous medical process [5].

IT is increasingly being viewed as a means to improve healthcare quality, safety, and efficiency by bringing decision support to the point of care and enabling routine quality measurement to become a reality. It can not only lower operating costs, but it can also reduce the number of medical errors. In the healthcare industry, IT opens up new avenues for increasing patient confidence and reinforcing patient trust in caregivers and healthcare facilities [1]. Additionally, in this industry, information and information exchange are critical for the delivery of healthcare at all levels [6] and because there are so many aspects and components that go into creating a successful healthcare system, it is also prone to equipment malfunctions, minor technical delays, and so on. To counter this, a monitoring and reporting platform can be used to ensure greater visibility of potential issues that could escalate into larger and more damaging situations.

Therefore the right monitoring and reporting tools can help to keep medical staff efficient without having to worry about failing systems, provide visibility into usage trends, equipment performance, downtime, thereby enhancing time and resources. Without leaving the command centre, these tools provide insight into potential problems easing the maintenance process and when determining what's gone wrong or what's really going right, occupancy percentages, system failures, usage rates, and other metrics will

reveal them [7].

In other words, such monitoring applications are seen as an important information source as they can show the current system status in real-time through a set of visual indicators, graphs, and reports. The information allows appropriate operational decisions to be made, either automatically or on the operator's initiative, ensuring the system operates as expected.

Some solutions already exist that monitor various types of services but they do not offer multi-organisational monitoring. Most hospital IT administrators manage several systems and infrastructures across multiple institutions. The opportunity for these administrators to have a centralised view of all systems and readily access information such as their health, status, and metrics served as the motivation for this dissertation.

1.2 Goals

The project's driving question is "Can a multi-platform and multi-organisation monitoring application enhance IT availability in the healthcare industry?". With this, the main goal laid out was to build a monitoring web platform for healthcare IT administrators to monitor the services on which their hospital relies. The platform should provide timely feedback on failures and show the system status in real-time through visual indicators. Furthermore, the platform should be supported on multiple device types, so that all system information is available on the go. The application will monitor services such as APIs and web-services, databases, and servers performance and availability.
Alongside with the main goal, minor milestones where expected:

- Study different front-end technologies that allow the monitoring of services using dashboards in a multi-platform technology;
- Plan a solution that ensures the security and confidentiality of all data involved;
- Development of a solution prototype, which should include the following features:
  - Service management and creation;
  - User creation and management, making sure that only administrators use the platform;
  - Graph visualisation to understand service's behaviour;
  - In the event of a failure, provide feedback by notifying the user via various channels;
  - Multi device compatibility.
- Solution's prototype implementation;
- Use both subjective and formal analysis methods to test, validate, and evaluate the system.

1.3 Challenges

As previously mentioned, the main goal of this dissertation is to develop a service monitoring UI for several hospital services with multi-platform and multi-organisation functionalities at its core. When developing this type of solution, some difficulties arise that must be addressed. They are as follows:

- Multi-platform compatibility: The proposed UI should be adaptable to various devices/platforms. This functionality provides flexibility and allows users to easily access the monitoring platform regardless of what device or platform they are using. As a result, development should account for this functionality and make it possible for the platform to be accessible to any platform using a single codebase;

- Multi-organisation support: The solution should not only provide visibility to each of the organisations being monitored, but it should also make it simple to understand the current state of the various services available within each of them. Aside from the architectural challenges, the UI should be straightforward enough that this feature is understandable and simple to use. To make the most of this functionality, the proposed architecture and UI design must be carefully considered;

- Data security: Only authorised hospital IT administrators should be able to access this platform and view the various metrics about the various organisations and services. To accomplish this, proper security mechanics must be implemented and thought out;

- Alerting: When services fail, alerting the IT administrator is critical in order for them to understand what is wrong and respond as quickly as possible. This should be done in a timely manner and through notification channels. As a result, it is necessary to consider which channels and types of alerting should be implemented in order to notify users;

- User friendliness: Given that the proposed solution is primarily concerned with developing an UI, it is very important that it provides a positive User Experience (UX), is simple to use, and achieves its primary goal. It is critical to understand how to accomplish this during the interface design process.

1.4 Document Structure

The chapter presents and concisely explains the structure of the document. The dissertation is divided into eight sections where the first one, **Introduction**, describes the motivation for the dissertation and the objectives that the project seeks to achieve.

Following the introduction is a **State of the Art** analysis. This chapter aims to understand Information Technology and Information System by presenting their significance and structure, as well as how they function in the healthcare sector and the main benefits and drawbacks. Since the goal of this project is

to create a UI, it is also necessary to understand what it is and how the design process should be carried out. Finally, a market analysis is conducted, highlighting four existing solutions.

The third chapter, **Methodology**, discusses the methodologies utilised in the formulation and validation of the research for this dissertation and also the study of some technologies, and tools that may be useful during development. This chapter is then followed by the **Background** chapter, which discusses and compares several front-end development technologies in order to determine which framework will be used during the application's development.

The **Proposal** chapter aims to investigate the design, development, and implementation of the proposed multi-device monitoring application. The gathered requirements will be explored, along with the architecture that results from the requirements.

The sixth section, delves into each developed component and what it entails. The main purpose of the **Implementation** chapter is to describe the system implementation development based on the previous chapter's modelling.

The seventh chapter, **Evaluation**, describes the system's evaluation and analysis, and it is determined whether the solution is fit for the purpose for which it was designed. Finally, in the **Conclusions and Future Work** chapter, the obtained results are discussed, future work is identified, and conclusions about the entire solution are drawn.

Chapter 2

State of the Art

Healthcare technology is currently the fastest growing sector of the healthcare industry. This technology is one of the most important in healthcare supply chain management for improving their service quality and performance [8].

There are several aspects that must be studied to understand how this technology works, what goals they are attempting to achieve, and how we can improve them. The chapter explains how Information System (IS) work and why are they so crucial, with an emphasis healthcare industry, what a monitoring application is and how it should be structured. Additionally, solutions available on the market are presented.

2.1 Information Technology

Information Technology (IT) is typically used in the context of business operations and includes both computer technology and telecommunications equipment to send, receive, store, and manipulate data that can be textual, numerical, audio, or video data, or any combination of these [9, 10]. Due to its wide range of technologies, capabilities, and functions as IT advances, it creates newer and more complex environments [11]. Some components and functions like cloud computing, operating systems, telecommunications, and databases are included in IT. Some more examples are depicted in figure 1.

The modern world is highly digital, with IT allowing access to any type of information through multiple points of access, saving time by automating various types of tasks, making them more efficient, and empowering several industries. The healthcare industry has been using IT to improve patient safety, operate more efficiently, reduce medical errors, and provide secure access to timely information while controlling costs, protecting patient privacy, and so on [1]. The following section elaborates on this statement and demonstrates its significance in this setting.



Figure 1: IT components and functions [Adapted from [10]]

2.2 Information Technology in Healthcare

Patients can use the internet to research symptoms and communicate with their doctors via email, text, or video conferencing to determine whether they need to schedule an appointment or if their symptoms can be treated with over-the-counter medications [12]. As a result, Information Technology has improved medical care, reduced patient errors, and reduced healthcare costs in a variety of ways, including:

- Improving the accuracy of medical information delivery;
- Allowing for quicker, more informed decisions that can have an impact on individuals and the public;
- Remote Patient Monitoring[1] (RPM), that is thought to be more convenient and cost-effective than traditional institutional care because it allows patients to remain in their familiar surroundings while receiving professional care [14].

According to [15], Health Information Technology (HIT) is defined as the application of information processing involving both computer hardware and software dealing with the storage, retrieval, sharing, and use of health care information, data, and knowledge for communication and decision-making. Similarly to IT, HIT encompasses a wide range of technologies, from simple charting to advanced decision support and integration with medical technology [2]. As previously stated, HIT provides numerous opportunities

[1] RPM collects health data from individuals in one location, such as a patient's home, and electronically transmits the information to healthcare providers in another location for assessment and recommendations [13].

for improving and transforming healthcare, including the reduction of human errors, the improvement of clinical outcomes, the care coordination facilitation, the improvement of practice efficiencies, and so on.

2.3 Information Systems

The IT ability to perform effectively and reliably in today's workplace is critical.

Information Systems are an integrated set of components that collect, store, and process data as well as provide information, knowledge, and digital products. They include interactions between technology and algorithmic processes within the boundaries of an enterprise, organisational interaction with technology and vice versa, as well as between society and technology [16]. Their primary components are computer hardware and software, telecommunications, databases and data warehouses, human resources, and procedures. The overall structure of organisational IS can be divided into three layers: operational support, knowledge work support, and management support (figure 2).

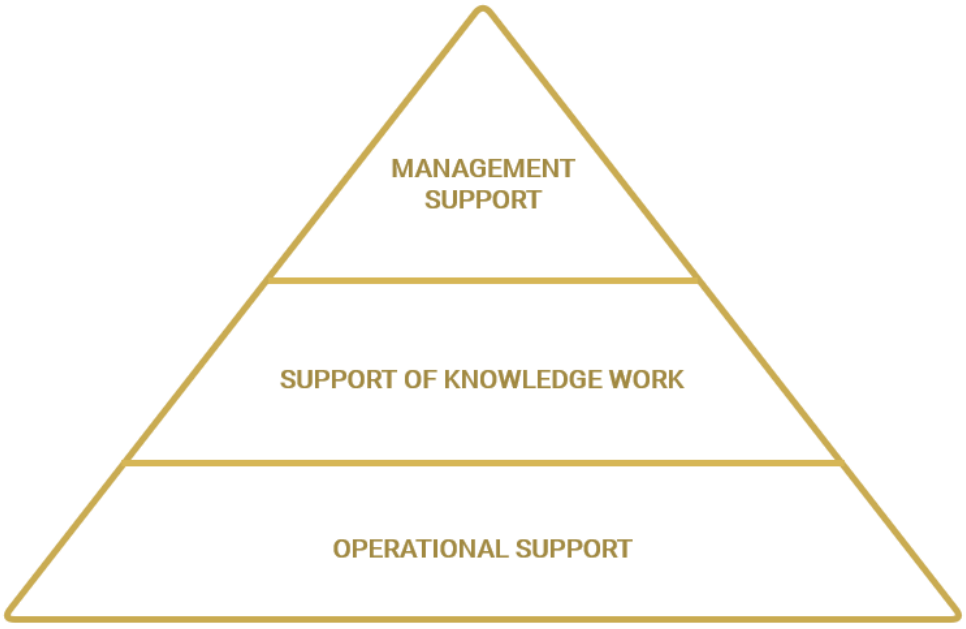


Figure 2: Structure of organisational IS [Adapted from [17]]

The foundation of an Information System is operational support, which includes several transaction processing systems for designing, marketing, producing, and delivering products and services. The middle layer is dedicated to knowledge work support, which includes subsystems for information sharing within an organisation. The top layer, management support, contains subsystems for managing and evaluating an organisation's resources and goals [17].

Many large corporations, such as eBay, Amazon, and Google, are based on IS.

2.4 Information Systems in Healthcare

Sharing health information provides a clearer picture of health and illnesses across the population, assisting in the prevention of disease spread and improving health outcomes. Sound and reliable information serves as the foundation for decision-making across all health system building blocks, and is required for health system policy development and implementation, governance and regulation, health research, human resource development, health education and training, service delivery, and financing [18].

A Health Information System (HIS) is a system that integrates data collection, processing, reporting, and use of information to improve health service effectiveness and efficiency through better management at all levels of health care. It includes all health data sources, such as health facility and community data; Electronic Health Record for patient care; population-based data; human resources information; financial information; supply chain information; and surveillance information, as well as their use and communication [19]. In this context it ensures the production, analysis, dissemination, and use of reliable and timely data by decision-makers at all levels of the health system.

A good HIS brings together all relevant partners to ensure that users of health information have access to data that is reliable, authoritative, usable, understandable, and comparable. They serve multiple users and a wide range of purposes that can be summarised as the generation of information to enable decision-makers at all levels of the health system to identify problems and needs, make evidence-based health policy decisions, and optimally allocate scarce resources [20].

These systems cross geographical boundaries because they are linked to a central database, allowing any patient file to be accessed from any hospital or clinic, increasing overall efficiency because there is no need to wait for the physical file. To access patient health records, authentication is required, ensuring that these systems are secure and confidential. Another advantage of these systems is that the patient files are electronic, which means that there is the possibility of increased storage capacity [21]. These systems, on the other hand, are expensive because they are highly sophisticated, and they are vulnerable to hacking, just like any other computer network these days.

2.5 UI/UX Process

User Interface (UI) is concerned with anticipating what users may need to do and ensuring that the interface contains elements that are simple to access, understand, and use to facilitate those actions. UI (UI) combines ideas from interaction design, visual design, and information architecture [22].

User Experience is concerned with having a thorough understanding of users, what they require, what they value, their abilities, and their limitations. It also considers the business goals and objectives of the project management team. UX practices encourage users to improve the quality of their interactions and perceptions of the product and any related services [23]. UX design is the process by which design teams create products that provide users with meaningful and relevant experiences. This includes the design of the entire acquisition and integration process, including branding, design, usability, and function.

Many factors influence the effectiveness of UX design, including complete and accurate contextual information, design approaches, and methods used. Design thinking, which is especially important in UX design, aids in addressing changes in users' environments and behaviours that are not always obvious. UX teams can use design thinking to conduct better research, prototyping, and usability testing to address modern-day human needs [24]. The design-thinking framework is divided into three steps:

1. Comprehend;
2. Investigate;
3. Materialise.

Within these larger buckets fall the 6 phases: empathise, define, ideate, prototype, test, and implement (figure 3).

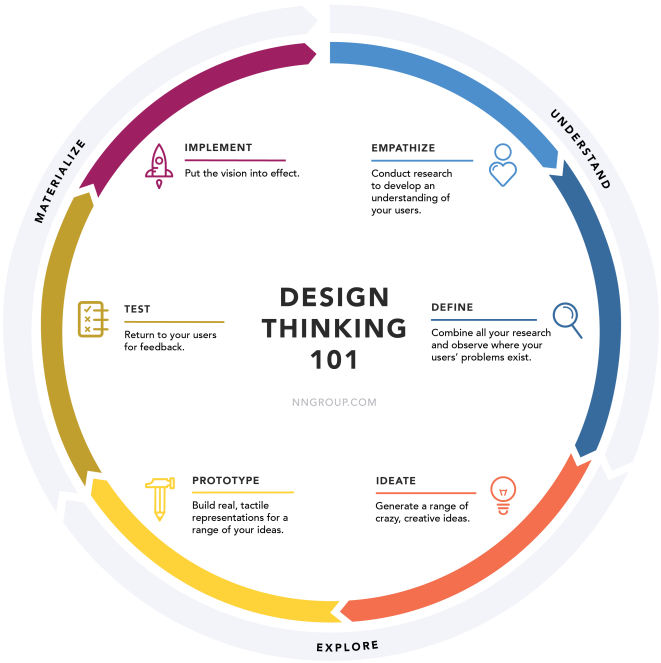


Figure 3: Design thinking process [Source [24]]

1. **Empathise**: Conduct research to learn about what users do, say, think, and feel. As the designer begins to identify the needs of their users, they also begin to highlight opportunities for innovation;
2. **Define**: Use the data collected during the empathise phase to gain insights and identify unmet user needs;
3. **Ideate**: Brainstorm a variety of solutions to the unmet user needs identified during the Define phase;

4. **Prototype**: Create representations for a subset of the concepts. The impact vs. feasibility of the gathered ideas is weighted in this phase through feedback on prototypes;

5. **Test**: Return to the users and solicit feedback. Determine whether the solution meets the needs of the users and, if so, whether it improves how they feel, think, or perform their tasks, ensuring that the goals are met;

6. **Implement**: Put the vision into effect. Ensure that the solution is materialised and touches the lives of the end users.

Design thinking is a non-linear process, meaning that designers can return to each phase iteratively.

2.6 Monitoring Applications

Hospital IT infrastructures must manage both "traditional" IT elements and specialised healthcare systems and protocols. It can be divided into three layers: "traditional" IT, the healthcare systems with their associated data formats and communication protocols, and medical devices [25]. "Traditional" IT is used to connect devices such as printers, servers, and is found in any IT environment. Standard devices such as routers, switches, and domain controllers are also included. The types of systems found in hospitals are referred to as healthcare systems (includes Health Information System, Electronic Health Record, and other similar systems). Finally, medical devices are the types of devices that are linked to the hospital's network.

It is critical to ensure that everything runs smoothly in this type of structure, as failures can not only cause inconvenient delays, but can also endanger human lives in extreme cases. Therefore, IT and hospital systems should be available at all times to ensure smooth hospital operations. As a result, a good monitoring application is required, and in the medical environment, it should monitor various aspects such as network speed, check for bottlenecks, monitor routers, servers, and storages, and so on. It should also provide alerts and notifications when certain thresholds are reached (for example, bandwidth is running low) and, most importantly, should display the status in a single dashboard view [25].

In the optics of understanding what are the current offers in IT monitoring solutions, different monitoring stacks were explored. An analysis was performed to the most interesting and compelling ones in this project's context: Site 24x7, PRTG-Network Monitor, Nagios and, Datadog. In the following sections their main services, key features, and how they can impact healthcare are identified and explained.

2.6.1 Site24x7

Site24x7[2] is a cloud-based website and server monitoring platform that allows small to large size businesses website performance, server, application performance, network, public and private cloud, real

[2] https://www.site24x7.com/

user monitoring and public status pages.

With the services provided, the client can continuously simulate and test user experience from multiple Site24x7 monitoring locations, and find problems before customers are impacted; report and alert on availability, content errors, or performance issues when there is a sign of potential trouble; and obtain a performance indicator from an end-users perspective (CPU, memory, services, disk, network usage, and so on) allowing to oversee health and availability.

In the **healthcare sector**, Site24x7 ensures that all devices are operational by monitoring the network. It can monitor REST APIs (which allow many applications and devices to communicate with one another and share patient data for easy analysis). Additionally, this software allows monitoring and optimisation of IT resource performance so physicians and technicians can quickly retrieve patient data for analysis and, in the event of a downtime, a text message or mobile app notification will be sent out immediately [26].

Figure 4 depicts a Site24x7 dashboard monitoring an AWS inventory for a hospital.

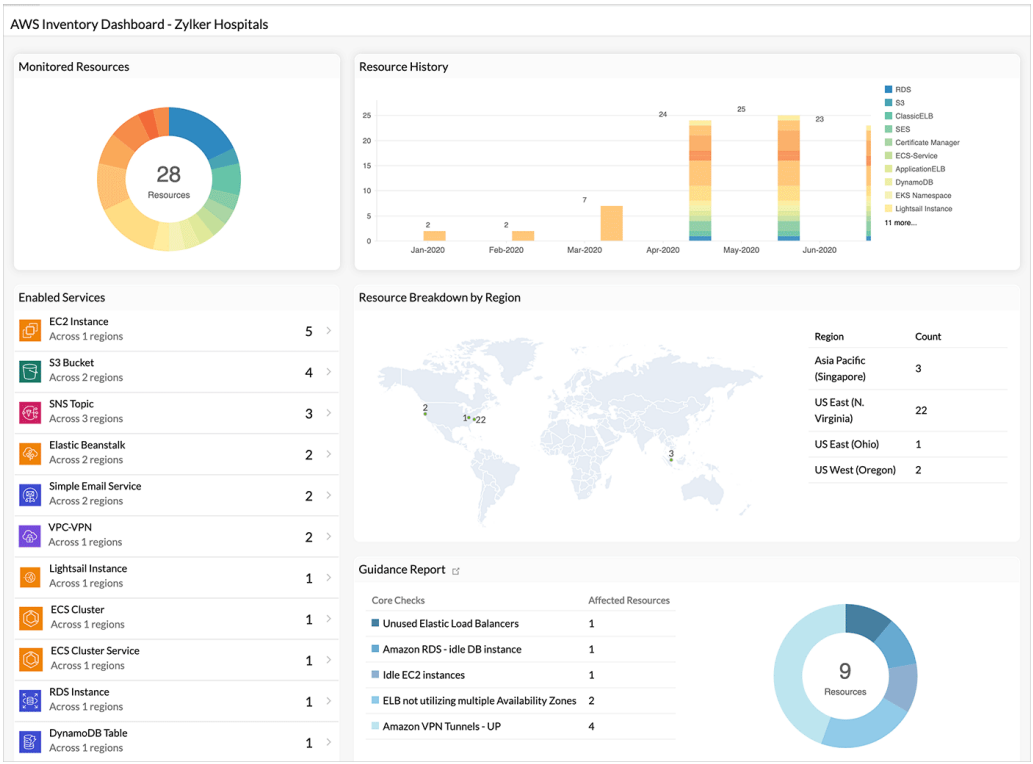


Figure 4: Site24x7 Dashboard

2.6.2 PRTG

PRTG Network Monitor[3] is a network monitoring software that collects statistics from defined hosts such as routers, servers, switches, and other critical devices and applications. It is a robust and simple-to-use solution that is appropriate for enterprises of all sizes.

[3] https://www.paessler.com/prtg

Its prime features are flexible alerting, multiple user interfaces, cluster failover (if the primary master of the cluster is not connected or is down, another node immediately takes over all its responsibilities, including the sending of notifications), maps and dashboards, distributed monitoring (monitor several networks in different locations and separated networks within the company) and in-depth reporting.

PRTG provides a centralised monitoring solution for medical equipment and IT in the **healthcare industry**. Users can create customised dashboards that visualise all hospital technology components; Because many hospitals have several location sites, using the PRTG remote probes feature is a quick and easy way to monitor multiple sites. Another adaptable feature allows the administrator to easily integrate all devices that support a REST API into PRTG (refrigerators, sensors for monitoring temperature, air quality, or humidity, and security systems are examples of such items); PRTG users can set thresholds to receive notifications before an outage occurs; patient data is managed and transferred using HL7, and PRTG employs HL7 to ensure secure data transfer and to monitor the machines involved [27].

Figure 5 illustrates an example of a PRTG dashboard for monitoring in a healthcare environment.

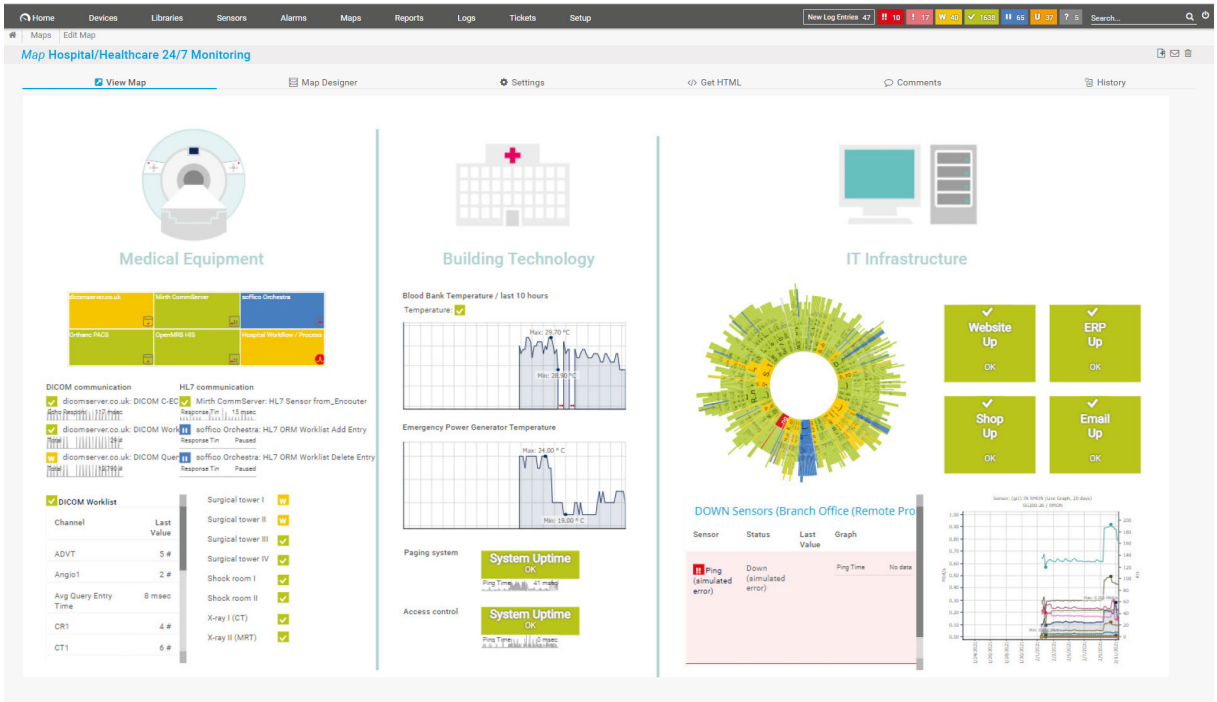


Figure 5: PRTG Dashboard

2.6.3 Nagios

Nagios[4] is a monitoring system that monitors the entire IT infrastructure to ensure that all systems, applications, services, and business processes are up and running. It can notify technical staff in the event of a failure, allowing them to begin remediation processes before outages disrupt business processes, end-users, or customers. Nagios monitors the network for problems caused by overloaded data links

[4] https://www.nagios.org/

or network connections, as well as routers, switches, and other network components, and can deliver results in a variety of visual representations and reports, as well as monitoring the availability, up-time, and response time of each node on the network.

With this tool, it is possible to monitor applications, services, operating systems, network protocols, system metrics, and infrastructure components; a unified view of the entire monitored IT infrastructure; web interface provides detailed status information; as soon as possible, infrastructure outages are detected; alert acknowledgements are used to communicate about known issues, problem resolution, and reporting: historical reports keep track of all alerts, notifications, outages, and alert responses.

Healthcare facilities can benefit from Nagios' comprehensive monitoring and alerting capabilities by deploying a Nagios solution. Performance graphing and capacity planning enable hospitals, clinics, and labs to keep their production IT systems running for as long as possible, allowing them to stretch their budgets as far as possible. Monitoring key performance indicators provides invaluable information for future system upgrades planning and budgeting. Medical organisations that use Nagios can receive immediate notifications of website and system outages that may have an impact on their public image, operational efficiency, and quality of care [28].

Figure 6 shows an example of a Nagios XL[5] dashboard.

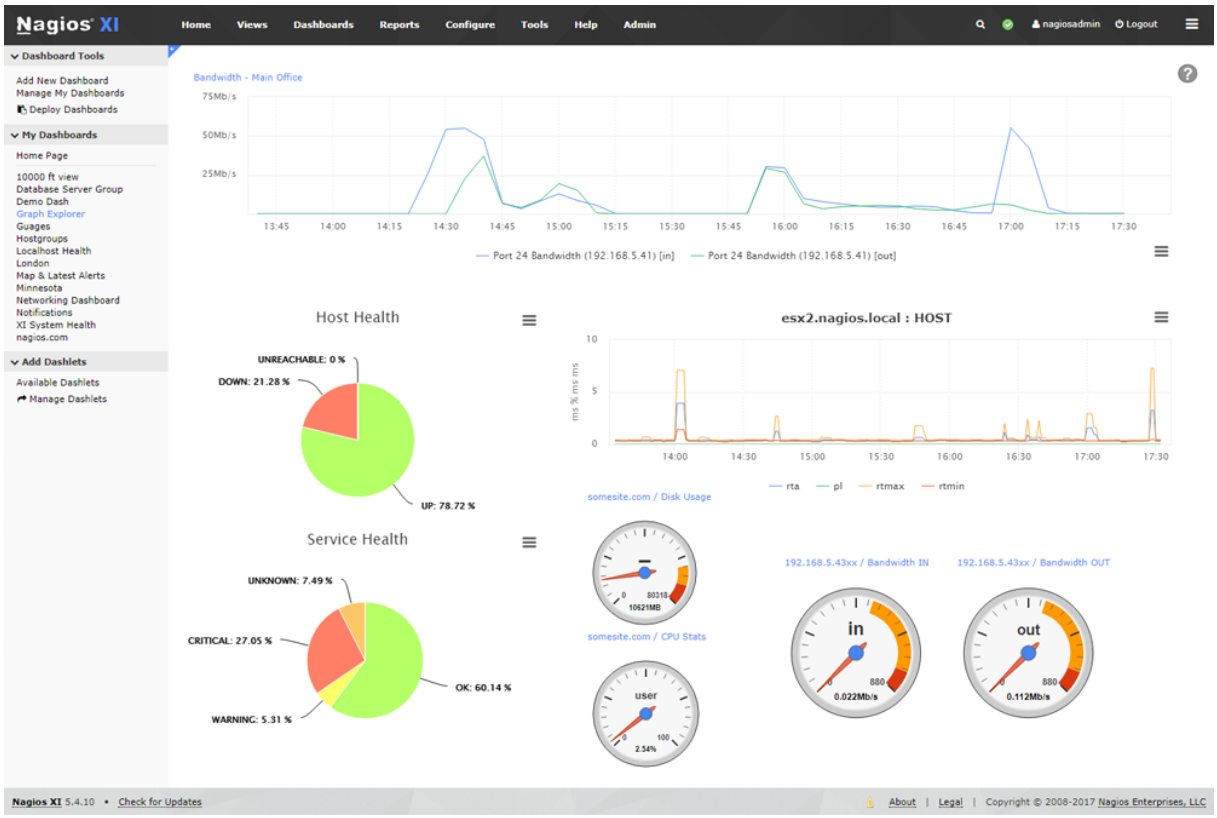


Figure 6: NagiosXL Dashboard

[5]Product from Nagios that monitors all mission-critical IT infrastructure components - applications, services, operating systems, network protocols, systems metrics, and network infrastructure

2.6.4 Datadog

Datadog is an IT and DevOps monitoring[6] and analytics tool that can be used to determine performance metrics as well as event monitoring for infrastructure and cloud services. Servers, databases, and tools can all be monitored by the software.

Datadog gives an IT/DevOps team a unified view of their infrastructure (including servers, apps, metrics and other services); Dashboards that can be customised; Alerts sent out in response to critical issues; Logs, latency, and error rates are automatically collected and analysed; Access to the API is granted and supports applications written in Java, Python, PHP,.NET, Go, Node, and Ruby.

In the **healthcare sector**, Datadog ensures healthy digital operations by gaining comprehensive visibility into application, infrastructure, and device performance; monitors digital assets on a global scale so that engineering teams can pinpoint issues in individual devices, applications, or lab systems and quickly correlate application and infrastructure data to determine the cause; accelerates DevOps initiatives, cloud migrations, EMR rollouts, and other modernisation efforts where visibility allows companies to establish performance baselines, validate that new technologies and processes are working as expected, and quickly address any issues that arise; and anticipate potential problems and outages: Datadog provides machine learning–based alerting that detects outliers or anomalies and forecasts resource usage and user demand, allowing engineers to prevent downtime in advance [29].

Figure 7 depicts an example of Datadog dashboard for an application and its infrastructure.

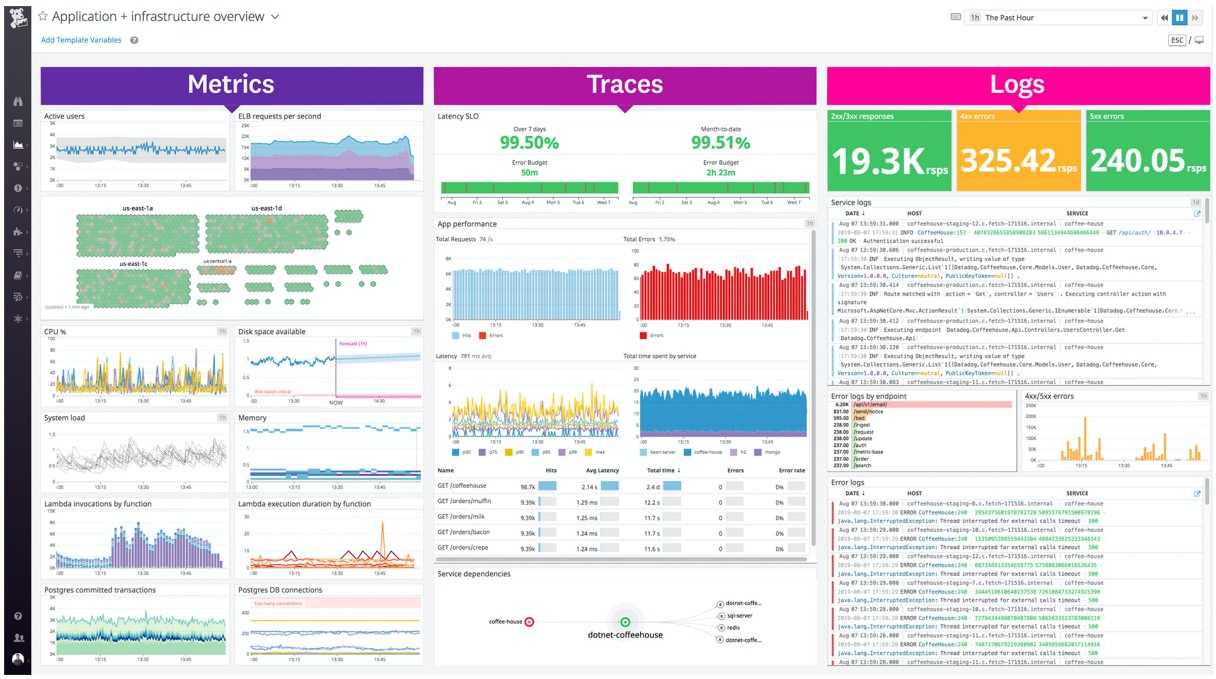


Figure 7: Datadog Dashboard

[6]Monitoring DevOps involves monitoring the entire development process, including planning, testing, development, integration and testing, deployment, and operations.

2.6.5 Comparison

By comparing the solutions, this section aims to reveal the key differences between them and to clarify which aspects/features are important in the proposed monitoring application.

Site24x7 is a website and server monitoring platform, PRTG is network monitoring software, Nagios and Datadog monitor IT infrastructure, but Datadog also monitors the DevOps process, as previously stated. This means that, while they are all monitoring applications, some features are only available on specific solutions. All the solutions can monitor servers and their performance, except Nagios, which cannot monitor networks. Only Site24x7 and Datadog can monitor CPU usage.

Aside from that, there are a few other things to consider: reporting/analysis, data visualisation, alerts, and dashboards. Reporting is an important feature that is very useful for data analysis since it allows users to ensure that everything is going as planned; Data visualisation enables business users to gain insight into massive amounts of data; When something goes wrong, users will be able to take action thanks to alerts; Dashboards enable users to gain quick insights into the most important data aspects.

As a result, only Nagios and Datadog can generate reports out of the four options. Each of the four software products includes data visualisation and the ability to send alerts/notifications. The solutions, except Nagios, have dashboards. One final important aspect is that the platform be cross-platform compatible; of the four solutions, only Nagios does not support mobile compatibility, whereas the others do.

Table 1 depicts an organised comparison with the previously mentioned aspects.

	Site24x7	PRTG	Nagios	Datadog
Server Monitoring	✓	✓	✓	✓
Network Monitoring	✓	✓	×	✓
Performance Monitoring	✓	✓	✓	✓
CPU Monitoring	✓	×	×	✓
Reporting/Analysis	×	×	✓	✓
Data Visualisation	✓	✓	✓	✓
Alerts/Notifications	✓	✓	✓	✓
Dashboard	✓	✓	×	✓
Mobile support	✓	✓	×	✓

Table 1: Software Comparison

Bearing the table 1 in mind and what makes a good monitoring tool, the developed platform should include the following features: performance monitoring, reporting/analysis, data visualisation, alerts/notifications, dashboards and mobile support. Other features may be added, but this study demonstrated that at least these points should be platform features.

Chapter 3

Research Methodology and Tools

This chapter discusses the methodologies utilised in the formulation and validation of the research for this dissertation. First, the applied research methodology - Design Science Research Methodology (DSRM - is introduced, followed by an introduction of different technologies and tools that could help overcome some challenges presented in 1.3.

3.1 Design Science Research Methodology

For most researchers, deciding on the best research methodology is one of the most hard and perplexing tasks. The research methodology that should underpin the research and data-collection methods to be used will be determined by the research type. The data-collection techniques used must be appropriate and capable of meeting the study's objectives, regardless of the method or methodology used for the study [30]. With this in mind, several methodologies were investigated, and it was determined that DSRM was the best fit. To comprehend what this methodology entails, there needs to be an understanding about the paradigm that underpins it.

Design Science Research (DSR) is a problem-solving paradigm that aims to improve human knowledge through the innovative artefacts' creation. Simply put, DSR seeks to improve technology and science knowledge bases through the development of innovative artefacts that solve problems and improve the environment in which they are instantiated. DSR aims to generate knowledge of how things can and should be designed, typically by human agency, to achieve a desired set of goals; this knowledge is referred to as Design Knowledge. For the production of unique solutions to important design challenges, DSR is a major research paradigm in many domains, including engineering, architecture, business, economics, and other information technology-related disciplines [31].

Design Science Research Methodology has six steps plus four entry points: problem-centered initiation,

objective-centered solution, design and development-centered initiation, and client/context initiation. The following is a brief description of each DSR activity:

1. **Problem Identification and Motivation**: Identify and justify the solution value to a specific research problem. Justifying the solution value serves two purposes: it drives the researcher and the research audience to pursue the solution, and it allows the audience to appreciate the researcher's problem knowledge. This task requires awareness of the current problem condition and its significance resolution;

2. **Specification of solution objectives**: The problem definition and knowledge of what is conceivable and doable can be used to infer the goals of a solution;

3. **Design and Development**: It is created an artefact. In terms of concept, it could be any designed object that incorporates a research contribution. This activity entails determining the desired functionality and the artefact architecture, as well as creating the actual artefact;

4. **Demonstration**: Shows how the artefact was used to solve one or more instances of the problem. It could be used in experiments, simulations, case studies, proofs, or any other appropriate activity;

5. **Assessment**: The researchers can decide whether to move on to communication and leave further improvement to future projects, or they can go back to step three to improve the artefact's effectiveness.

6. **Communication**: To the appropriate stakeholders, all aspects of the problem and the designed artefact are communicated.

Figure 8 depicts the DSR Methodology Process Model with the six steps previously mentioned.

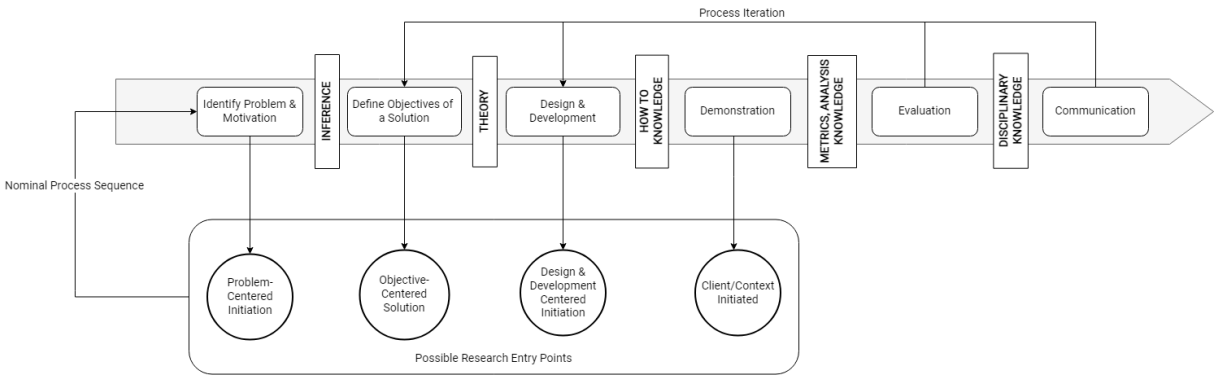


Figure 8: DSR Methodology Process Model [Adapted from [31]]

As previously stated, this dissertation began with the discovery of a problem and a lack of functionality in existing monitoring systems. As a result, it follows that the DSRM will be used with a problem-centred approach, completing the six phases of the technique.

3.2 Technologies and Tools

Based on the challenges presented in 1.3, it was determined that some tools should be integrated into the system and/or researched in order to aid in development. In the following subsections, some tools will be highlighted that can help answer questions like, "How can we make our platform multi-device?", "What is the importance of notifications in this type of platform?", "Is there a tool that can assist with visualisation?", and so forth.

3.2.1 Progressive Web App

The application should be cross-platform compatible, which means it should adapt to different devices. After some research, Progressive Web App appeared to be the best option since the platform should be web-based but needed to be cross-device compatible. Bearing this in mind, it becomes essential to first understand what it is and how it works.

According to [32], there are three pillars that transform PWAs into a platform-specific application-like experience.

- **Capable**: Only platform-specific apps could truly claim to be capable until recently. New and upcoming APIs are attempting to change that by expanding what the web can do with features such as file system access, media controls, and full clipboard support. All of these capabilities are built on the web's secure, user-centric permission model, ensuring that visiting a website is never a frightening prospect for users;

- **Reliable**: A dependable PWA feels fast and dependable regardless of the network. Reliable applications must be usable regardless of network connection. Users expect apps to launch even if their network connection is slow or unstable, or even when they are not connected to the internet. Even if sending a request to your server is difficult, they expect the most recent content with which they have interacted to be available and usable. When a request is not possible, they expect to be informed rather than failing or crashing silently;

- **Installable**: Installed PWAs run in a separate window rather than a browser tab. They can be accessed from the user's home screen, dock, taskbar, or shelf. PWAs can be configured to accept content from other applications or to be the default application for handling various file types.

In short, PWAs are created and enhanced with modern APIs to provide improved capabilities, reliability, and installability from a single codebase.

This means that it's a browser-based web application built with cutting-edge web technologies to provide users with a native app-like experience on any device or channel—mobile, desktop, or, of course, the web, reducing development effort while also allowing for a faster time-to-market.

Web apps can be installed and distributed without the use of app marketplaces, they can function without Internet access, they can receive push notifications, and they look like regular apps. The Service Worker API makes this possible by allowing developers to use a background-executed script to act as a network and device proxy. A PWA cannot function properly in browsers without Service Worker support. The worker is registered on the user's first page visit, and it consists of a JavaScript file with life-cycle hooks for business logic and cache management, handling background synchronisation, data and application shell caching mechanisms, and network request interception [33].

3.2.2 Notifications

A monitoring system serve as the foundation upon which alerting is built, where they collaborate to provide information about the performance of applications and infrastructure. They detect performance or usage anomalies, aid in troubleshooting and problem identification, reveal usage and behaviour patterns or trends, and aid in understanding the impact of any changes made to applications or infrastructure. Monitoring solutions send notifications whenever metrics meet conditions defined in alert rules, prompting users to investigate problems and even assisting in narrowing down and identifying potential root causes [34].

With this in mind, the several notification channels integration into the application, such as Slack[1], push notification, and SMS are crucial. A different approach should be taken for each notification channel because some libraries already offer this integration and others need to be studied to understand what is a better way to offer these services into the application. Push notifications will be the default notification channel in the beginning, but other options will be provided later.

3.2.3 Giraffe

One of the most important aspects in a monitor tool are the visual indicators. Some approaches can be taken to show them in a dashboard. InfluxDB will be used as the storage mechanism, which means that all metrics for consumption will be stored in this database. Giraffe[2] is an open source React-based visualisation library used to implement InfluxDB's UI (Example in Figure 9).

According to [35], giraffe uses algorithms to deal with the challenge of visualising the massive amounts of data that InfluxDB can ingest, query and strives to distinguish itself through a variety of features, including:

- Flux[3] language assistance;
- Easy reactivity and extensibility via React;

[1] https://slack.com
[2] Github repository: https://github.com/influxdata/giraffe
[3] Flux is a functional data scripting language developed by InfluxData for querying, analysing, and acting on data.

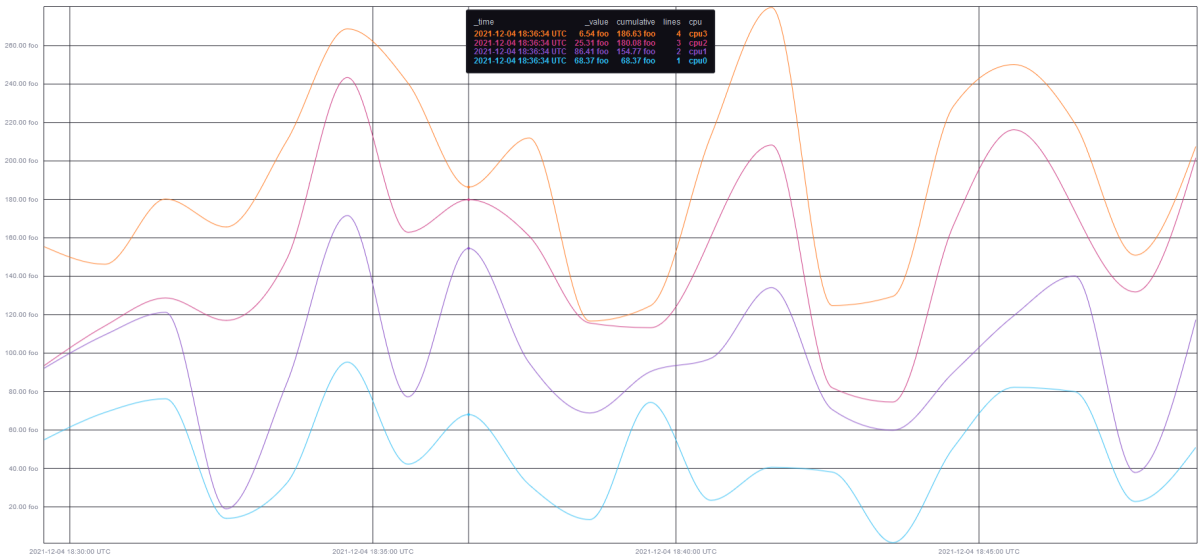


Figure 9: CPU values over time, an example plot using Giraffe

- Support for mapping column groupings to a single visual aesthetic;
- A high-level API capable of specifying a wide range of visualisations using a few simple concepts.

This library could be useful in implementing the various graphs and charts required for the agents.

3.2.4 JWT

JSON Web Token (JWT) is an open standard that defines a compact and self-contained method for securely transmitting information as a JSON object between parties. Because it is digitally signed, this information can be verified and trusted. JSON Web Token are useful for **security** and **information exchange**. The most common scenario (which will be the case in this project) in which JWT is used is for the security: once logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. They are an excellent way of securely transmitting information between parties. Since JWTs can be signed (for example, with public/private key pairs), there is a certainty that the senders are who they claim to be. Furthermore, because the signature is calculated using the header and payload, it ensures that the content has not been tampered with [36].

Chapter 4

Background

In recent years, the demand for highly sophisticated and complex web applications has significantly increased to replace the old desktop applications in all areas [37]. Even though web applications are more complex, their development should not be. As a result of the emergence of several front-end frameworks, web development has become much simpler in recent years and many frameworks have become available to web developers [38]. Each framework has its own strengths and weaknesses, and specific application areas, allowing the developer to choose one according to the project needs [39].

Today, there are two main approaches to building web applications: traditional web applications that perform the majority of the application logic on the server and SPAs that perform the majority of the user interface logic in a web browser, communicating with the web server primarily through web APIs. According to [40], when the application's client-side requirements are simple or even read-only, and the application must function in browsers that do not support JavaScript, we should use traditional web applications. On the other hand, we use an SPA when the application must expose a rich user interface with many features as well as an API for other (internal or external) clients.

Because of the project requirements, the application is SPA-based. The *State of JS* survey polled 23,765 developers from 137 countries in 2021, asking them questions about everything from front-end frameworks and state management to building tools and testing libraries. From 2016 to 2020, the most popular front-end frameworks are depicted in Figure 10.

As can be seen, React is the most widely used, followed by Angular and Vue. Svelte, a newer solution, is establishing itself as a potential new market leader [41]. Despite the several frameworks for developing SPAs, the next sections will be focus on the four leading frameworks, as shown in Figure 10: React, Angular, Vue, and Svelte.

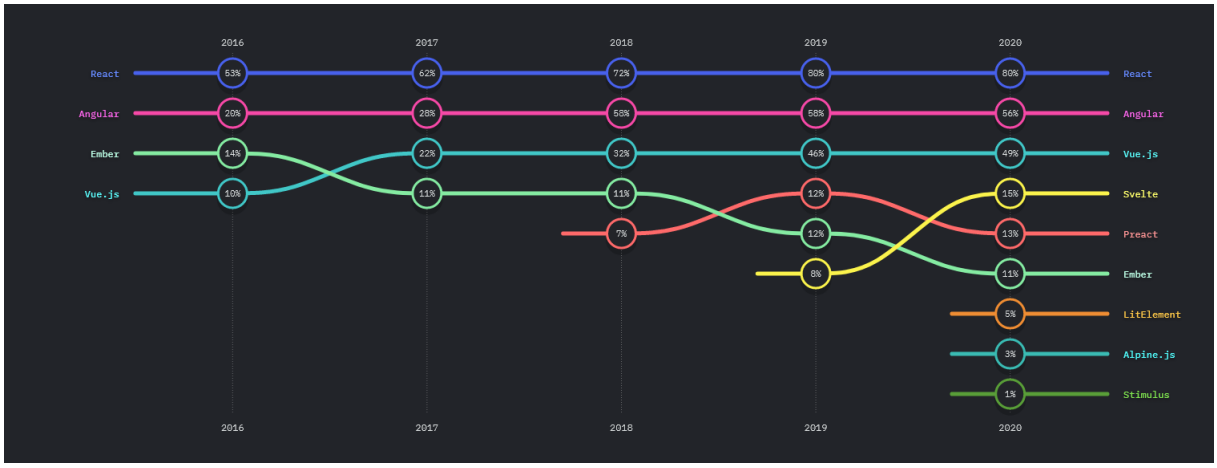


Figure 10: Front-end frameworks experience ranking

4.1 React

Facebook[1] created React[2], an open-source JavaScript library for interactive User Interface. It is used to manage the view layer in web and mobile apps. React enables the creation of reusable UI components. Developers can use components to create large web applications that can change data without reloading the page. React's primary goal is to be fast, scalable, and simple only working on the application's UI [42].

Since React has a component-based architecture, it is possible to break an application into many components. The components collaborate to create a powerful UI, allowing for code reuse to make the development process easier and less error-prone. "State"is an important concept in the React world. State represents data that is normally shown to users, or information about the component, subject to change. When the data changes, we can create simple components for each state of our application, and React will efficiently update and re-render only the necessary components, indicating that it is declarative. It is actively maintained and updated, and there is strong community support for it. This language has a lower learning curve than Angular [43]. React, on the other hand, relies on third-party libraries for advanced functionality.

4.2 Angular

Angular[3] is a framework and platform for developing single-page client applications in HTML and TypeScript create by Google. Angular applications are built on Angular modules (NgModules), which serve as a compilation context for components, and groups related code into functional sets. Angular applications are defined by a set of modules, with at least one root module that allows bootstrapping and many other feature modules [44].

[1] https://about.fb.com
[2] https://reactjs.org
[3] https://angular.io

Angular is a framework that includes a wide range of features. It has a MVVM architecture and separates data from design and visual representation, making even complex web applications easy to manage. Angular reduces the load on server CPUs by supporting caching and a variety of other processes, meaning that the server performs exceptionally well due to low traffic, only serving to static files and responding to API calls. The two-way data binding ensures that changes to the view are immediately reflected in the model and vice-versa [45].

Angular's main drawbacks include a steep learning curve and being too heavy and complex for simple applications due to the high number of features it includes.

4.3 Vue

Vue[4] is an open-source JavaScript framework which combines features from Angular and React.js. It is a progressive framework for creating UIs. Vue, in contrast to other monolithic frameworks, is built from the ground up to be incrementally adoptable. The core library is limited to the view layer and is simple to integrate with other libraries or existing projects. However, when combined with modern tooling and supporting libraries, Vue is capable of powering sophisticated SPAs [46].

Similarly to React, Vue has strong community support, its components are reusable, making its code also reusable, and it is actively maintained and updated with strong community support. Vue is good for unit testing because having components greatly simplifies this task and has a lower learning curve than Angular and React.

This framework is still in its early stages, there is a lack of support and limited resources for large-scale projects. The Vue community and development team are still small compared to React and Angular. These are the primary disadvantages.

4.4 Svelte

According to [41], Svelte[5] is quickly establishing itself as a very serious contender for the front-end crown. Svelte is a radical new approach to building user interfaces. Whereas traditional frameworks like React and Vue do the bulk of their work in the browser, Svelte shifts that work into a compile step that happens when we build our app. Instead of using techniques like virtual DOM diffing, Svelte writes code that surgically updates the DOM when the state of the app changes [47].

As said previously, Svelte performs the majority of the work when building the app and compiles it to efficient vanilla JavaScript, rather than doing it in the browser. Compile Svelte to vanilla JavaScript improves code readability, allows for re-use, and results in a much faster web app. Svelte does not require

[4] https://v3.vuejs.org
[5] https://svelte.dev

declarative, state-driven code that the browser must translate into DOM operations, meaning that there is no longer any need to use virtual DOM [48].

However, IDE support is not yet on par with popular frameworks, having a small open source ecosystem.

4.5 Background Overview

Front-end frameworks serve as the foundation blocks of the software development process. There are numerous options for developing visually appealing apps that provide a positive user experience, and aside from the UX, it is also critical to have tools that make the developer's life easier and more comfortable when developing web applications.

The previous sections discussed different framework options to achieve these goals, and this section will compare them. Leading frameworks such as React, Vue, and Angular, as well as newer solutions such as Svelte, have proven to be powerful tools for developing applications. React, Vue, and Angular are declarative and have re-usable components and thus re-usable code. React, Vue, and Svelte are lightweight and actively maintained frameworks. React and Angular compared to Vue and Svelte have the best documentation since they are older and more structured. Performance wise, Svelte is the best, followed by Vue, React, and Angular. Angular has the highest learning curve, followed by React, Vue, and Svelte, with Svelte having the lowest. All these distinctions can be illustrated in table 2.

	React	Vue	Angular	Svelte
Re-usable code	✓	✓	✓	×
Declarative	✓	✓	✓	×
Actively Maintained	✓	✓	×	✓
Lightweight	✓	✓	×	✓
Documentation	Very Good	Good	Very Good	Moderate
Performance	Medium	Good	Moderate	Very Good
Learning Curve	High	Moderate	Highest	Smallest

Table 2: Comparison between React, Vue, Angular and Svelte

When faced with these facts, the choice was between Vue and React, but it came down to which framework the developer was more familiar with and knew had more online resources, so React was chosen.

Chapter 5

Proposal

The main topics of this chapter will be the design, development, and implementation of the proposed multi-device monitoring application. It focuses on the decision-making process for design and development. An evaluation was performed in order to gain a better understanding of the problem at hand and how a monitoring application can be used to address these issues. The obtained results from this questionnaire are presented first, and then discussed. Following that, the various system requirements, including use cases as well as application functional and non-functional requirements, are examined. The proposed solution architecture is analysed based on the identified requirements.

5.1 Technical Evaluation

The first step toward gaining a better understanding of the problem and its associated needs was to conduct an anonymous online survey, the questions for which can be found in Appendix A. This questionnaire was distributed via Google form to system administrators of Portuguese hospitals in CHTS, and received 17 responses. The questionnaire is divided into two sections: one with closed questions about the respondent's background, context, and perspective to gain a more objective understanding of their ideas, and the other with open questions about requirements and functionality suggestions. This questionnaire was used to gain an understanding of the current state of monitoring services and their application, particularly in the healthcare setting. Making it possible to learn how and to what extent monitoring services can help with system/infrastructure management, as well as which direction the solution should take. This enabled the gathering of use cases as well as the various system requirements required for the application. As seen in figure 11, approximately 94% of the participants actively work for a company that actively employs monitoring systems.

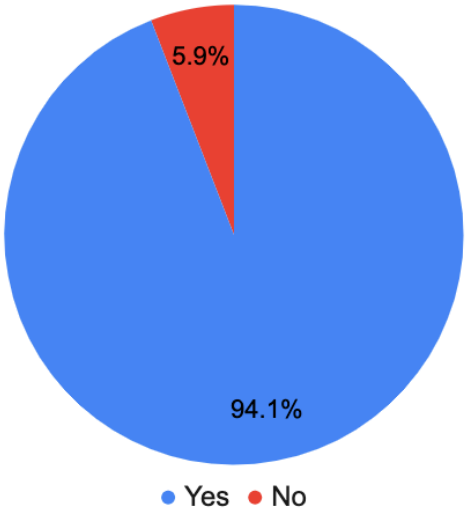


Figure 11: Percentage of participants that actively work for a company that actively employs monitoring systems

The six statements where participants had responses ranging from Strongly Agree to Strongly Disagreed were transformed into values ranging from zero to four, where zero means Strongly Disagree and four means Strongly Agreed. The main goal was to determine whether participants usually agree or disagree with some statements. The statements:

- **Statement 1**: Currently, whether there is a failure or not, you can check and control the system status.
- **Statement 2**: When there is a failure, that failure is noticeable.
- **Statement 3**: When a failure is detected, you can determine the implications and other services impacted.
- **Statement 4**: A monitoring system is important for systems management.
- **Statement 5**: A good monitoring system makes it possible to improve the systems availability.
- **Statement 6**: A monitoring system that can analyse and display, seamless, the targets status and organisations, streamlines the systems management process.

A graph with the average and median response for each statement was created and is depicted in figure 12.

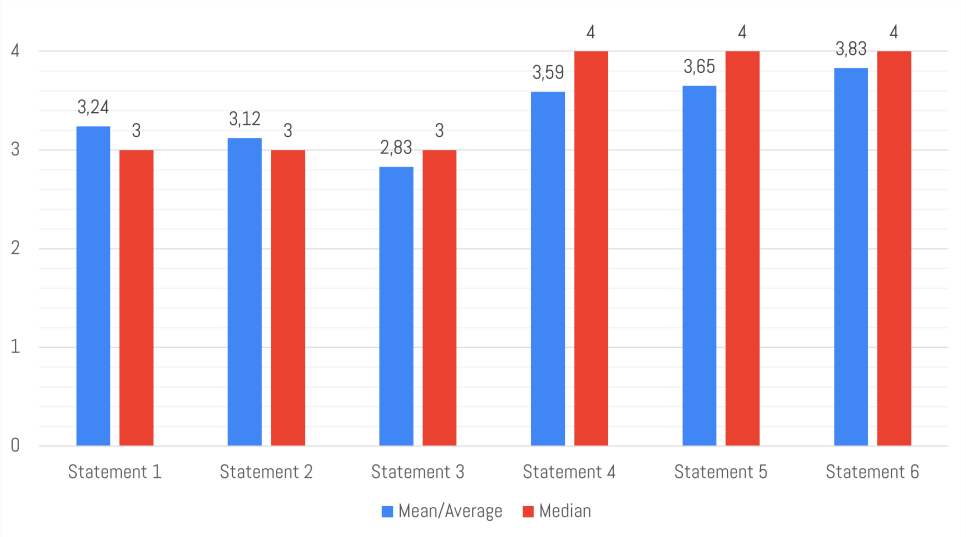


Figure 12: Average and median response to the affirmations

The results show that most administrators tended to agree to every statement, meaning that they are aware of the system's behaviour and that failures are not overlooked. However, the awareness time can be a critical factor in resolving the issue and ensuring that no other services become affected. An interesting finding was that, upon a failure, some administrators were unaware of the implications for other services. This could be due to a large number of services, which makes it easy to lose track of them, or to the fact that when one system fails, it is independent and does not affect the behaviour of other systems. It appeared clear how relevant a monitoring solution is for these types of services and how impactful they can be for improving the system's overall availability, providing a promising result to answer the dissertation's driving question.

Even though most people agreed that a failure is noticeable, the average time can be quite long, with 30 minutes being the longest, and the average response time also varying greatly. This could be due to the nature of the failure, where 5 minutes may be enough to resolve it, but if very complex, a response time of 120 minutes may be needed. This response is still concerning because the question asked about the average response time to an occurring failure, and 120 minutes as an average means that either most failures are very complex, or that the response time can be delayed because the failure was detected late, and the damage done is far too great and requires time to be properly repaired.

Finally, when participants were asked about what makes up a good monitoring system, the majority referred to:

- Being able to detect each failure:
  - Indicators about the service's health;
  - The possibility to define criteria to measure what is a healthy service or not;
  - Notifications about failures.

- Other features pointed out:
  - In the event of a failure, the system should be able to instruct them on how to resolve the issue that caused it to fail;
  - Indicators that can predict system failure and service load;
  - A solution that allows the events correlation generated by several information security applications (also known as SIEM[1]).

The questionnaire revealed some issues that need to be addressed. The most important is the failure awareness time. The platform should notify the user as soon as a failure occurs to ensure that no further failures occur, lowering the poll values. When the platform mockups were completed, some questions about the value of specific features began to surface. The survey was a great way to get answers to all of those questions, figure out what should be prioritised, and modify the mockups and the requirements to reflect the new model. Additionally , since the development was still in its early stages, shifting the application's focus to the new mockup was relatively simple.

5.2 Use Cases

A use case is a written description of how a website's users will complete certain tasks. It describes a system's behaviour as it responds to a request from the perspective of a user. Each use case is represented by a series of simple steps that begin with a user's goal and end when that goal is met. Use cases add value because they help explain how the system should behave while also brainstorming what could go wrong [50]. With this, and in order to make the scope of the web app more perceptible, use cases were gathered with the help of the technical questionnaire answers and then a use case diagram was drawn. This diagram depicts the system's functionalities and the actors with whom it should interact.

The list that follows delves into each use case and its definition.

- **Create Organisation** - An administrator must be able to create an organisation;
- **Delete Organisation** - If an organisation is no longer required, an administrator should be able to delete it;
- **Check Notifications** - An administrator can check notifications. When the agents' health is at stake, receiving feedback is critical to keeping track of them, and notifications will aid in understanding what went wrong and quickly resolving the underlying issues;
- **Create Agent** - An agent must be created so that an administrator can monitor what is required. In other words, to monitor the several organisation services, an administrator must be able to track them in our system;

[1] This type of security solution helps businesses identify potential security threats and vulnerabilities before they disrupt their operations [49]

- **Delete Agent** - An administrator can delete agents when they are no longer wanted;
- **Edit Agent** - An administrator must be able to edit an agent's configuration given that it could occasionally change;
- **Check Agent Configurations** - Each agent's main information should be accessible to an administrator;
- **Check Agent Health** - An administrator can monitor each agent's health through the most recent status;
- **Agent Graph Visualisation** - An administrator can use graphs to help understand an agent's behaviour;
- **Create Agent Alert** - Administrators should receive alerts if they want to keep abreast of a critical agent situation ;
- **Delete Agent Alert** - An administrator can delete alerts. This will be useful when administrators do not want to receive more agent updates;
- **Edit Agent Alert** - An administration can edit Agent alerts. Some aspects of an alert's configuration may change.
- **Search by Agent** - Administrators must be able to search for specific Agents. This feature is crucial considering the possibility of multiple monitoring agents.
- **Create a New Account** - An administrator can create accounts. Any administrator who already has an account must be able to create a new account for others;
- **Delete Account** - An administrator must delete their own account.

Figure 13 summarises the details of the system's users and their interactions. Based on these use cases, the system's functional and non-functional requirements were identified. These will be explored in the next section.

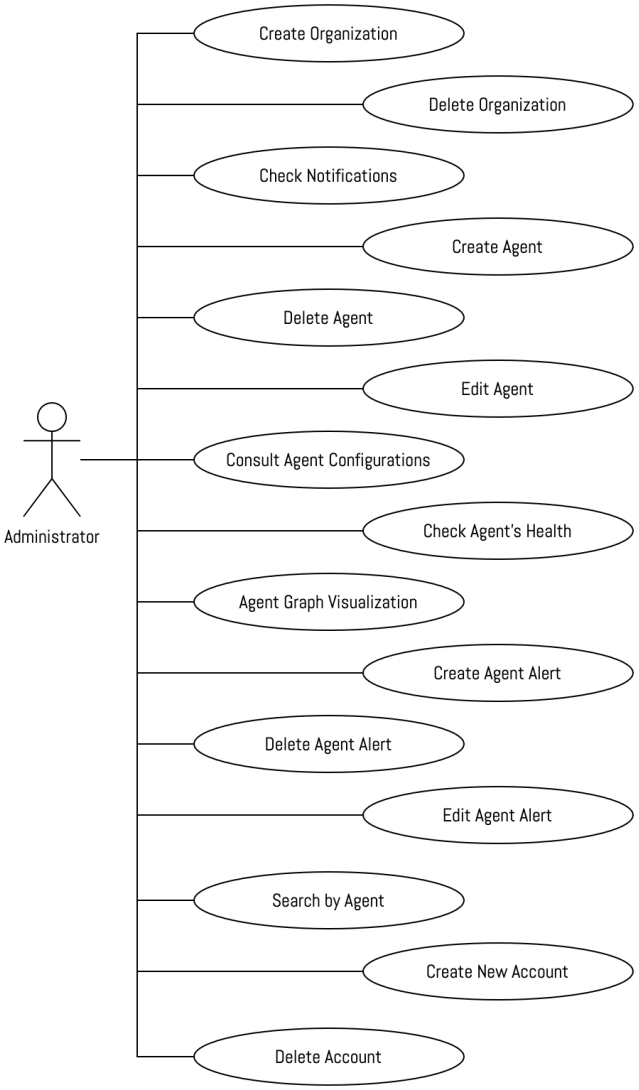


Figure 13: Web app use-case diagram

5.3 Functional and Non Functional Requirements

Requirements lay the groundwork for the product's vision, scope, cost, and timeline, and they must ultimately aim for finished product quality and performance [51]. Requirements are frequently classified as functional (FR) or non-functional (NFRs). FRs are typically used to describe system functionalities, whereas NFRs are used to describe system properties and constraints [52].

The FRs are:

- Monitoring agents and targets creation, including HTTP requests, health checks via ping, and database monitoring;
- Make adjustments to the previously mentioned monitoring agents;

- Delete monitoring agents;
- Create graph-based status monitors on agents (services). It will be shown the current service's health, as well as important data;
- Create alerts via notification channels, allowing users to connect to third-party notification services such as email, and Slack;
- Edit and remove the above mentioned alerts;
- Enable real-time push notifications to alert to changes in an agent's health.

Concerning the NFRs, the following are listed:

- **System Reliability**: Some issues can have a negative impact on dependability (unexpected usage, external failures, bugs, etc). The application's ability to provide a consistent functional experience to all users over time is essential;
- **Solution Scalability**: The application should be scalable, which means that its features should be small, maintainable, independent, and customise so that they can be served independently. This makes it critical in the long haul to maintain the codebase to aid performance-related tweaks without breaking the application [53];
- **Usability**: The system should be easy to use and understand with an user-friendly interface, aiming to improve overall UX. The system must also use terms and vocabulary appropriate to the context in which it is inserted to facilitate its understanding by IT administrators;
- **Security**: It is critical to ensure that only authorised users have access to the monitoring data;
- **Performance**: The administrator's interaction with the system should be smooth so that using the system is pleasant. The system should try to respond to administrators' requests as quickly as possible and should always be available so as not to impede its use by administrators.

5.4 Architecture

After gathering all the requirements, the project's architecture was set. As previously stated, a hospital has a plethora of different IT components that must be monitored to ensure that everything runs as smoothly as possible and to reduce the error rate throughout the entire infrastructure. This means the platform can translate each hospital into different agents that monitor each component. For instance, if Hospital A has a database and one server that needs monitoring, two agents will be created in the application to monitor both services. Figure 14 depicts the platform's high-level architecture, where it will display all monitoring data for every hospital that has services that need monitoring and have the client installed.

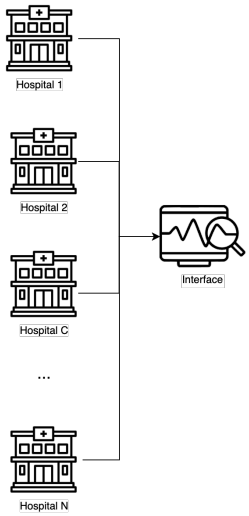


Figure 14: High-level architecture

Having in mind the work performed by Vasco Ramos, and the specificity of the integrated health network, this architecture had to be adjusted. Therefore, another one, that comprises local servers communicating into a guardian server, had to be followed. Local servers can be thought of as a locally installed data collection library that provides push and pull-based services in the hospital data centre. Their primary function is to collect and transmit various metrics (average response, status code, overall agent health, etc.) to the guardian server, which will then save and manage them. The platform will then display these metrics. Figure 15 shows the final architecture, where one can observe the different architecture components, making it possible to overcome the challenges of communication between endpoints overlapping firewalls and other barriers.

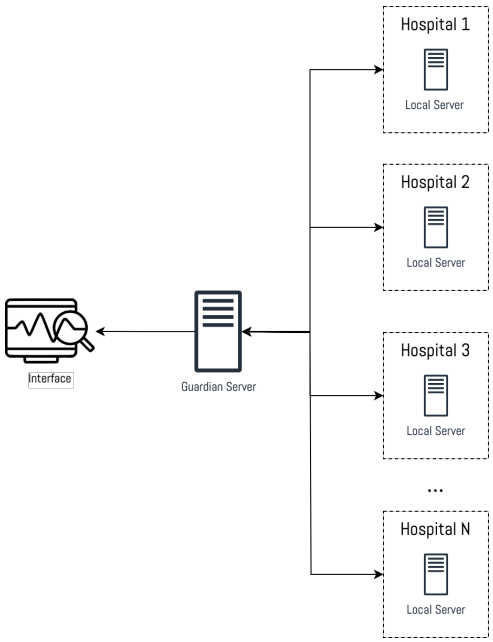


Figure 15: Architecture of the proposed monitoring solution

Chapter 6

Implementation

The chapter describes the proposed solution's implementation process. The platform was built with React, as previously stated, and is divided into four core components: Login, Agents Overview, Agent Creation, and Agent Dashboard, all making requests to the back-end server to build pages with relevant information, such as agent monitoring data. The implementation and functionality for each of these pages are described in the following sections.

6.1 Login

The Login page, (figure 16), is the platform's landing page. This page has a straightforward interface with two text inputs, one for the username and another for the password. Moreover, there's an icon that shows This page allows administrators to log in to the application and access the application's features. The user enters the password and presses the Sign In button. By clicking the button, the application sends a GET request to the server via the */login* route, along with the admin's username and password entered in the input text boxes. After successfully logging in, the administrator is redirected to the Agents Overview page, and the authorisation token and username are saved in the *SessionStorage*. The user can use all platform features as long as the token is valid. The expiration date is used to check the validity of the token whenever the application is re-rendered. If the token is invalid, the user is redirected to the Login page to re-enter their credentials. and hides what is written in.

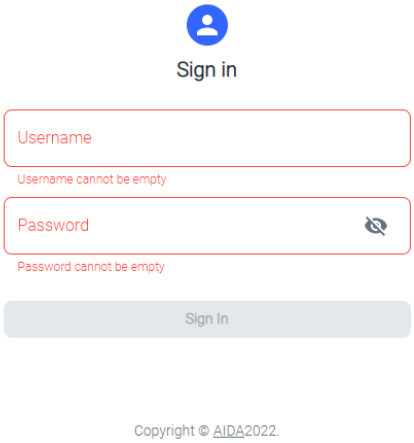


Figure 16: Login page

6.2 Platform Template

The application's UI is fairly simple, consisting of a *Navbar* at the top and a *Menu* on the left side of the screen. The only thing that can change are the pages that represent each component (figure 17). The following subsections will explain and detail the *Navbar* and the *Menu*, since they are the ones always visible.

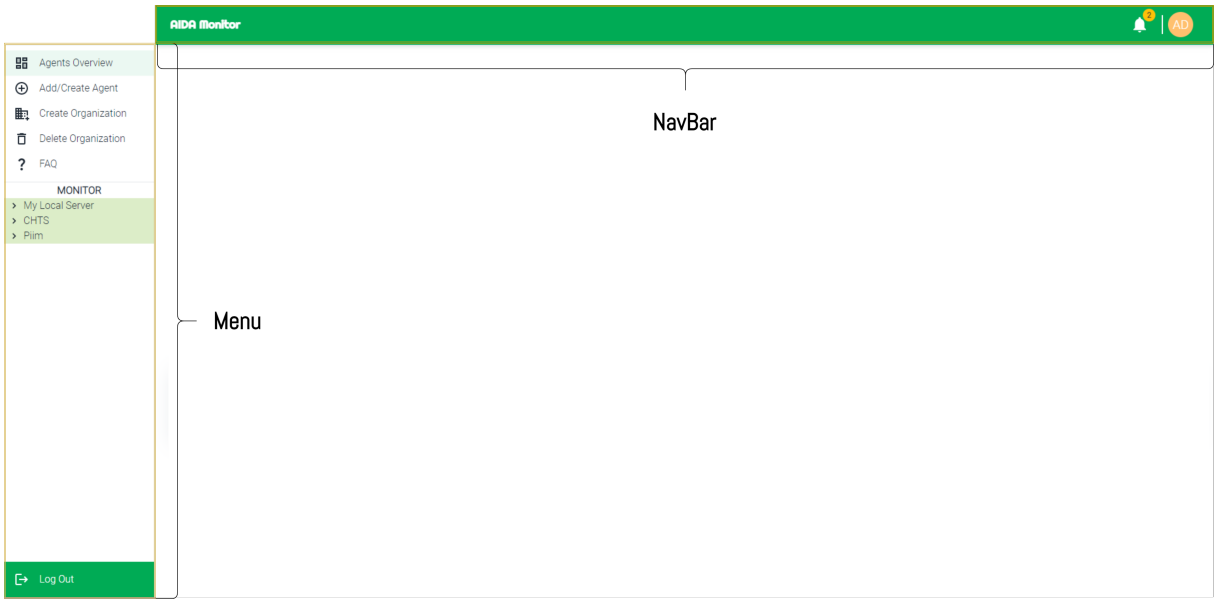


Figure 17: Platform's template

6.2.1 Navbar

The *Navbar* component, as shown in figure 18, is divided into two sub-components. The *Notifications*, represented by a notification icon with the unread messages number, and the *User Information*, represented as an avatar. Actions/pages will be triggered when clicking on these pages



Figure 18: *Navbar* sub-components

When clicking on the *User Information* area, there are four options:

Add an account: Since the platform can only be used by administrators, creating users should only be done within the application. When the "Add an account" button is clicked, a page with a form (figure 19) is displayed, and when the form is correctly filled out, the data is sent to the back-end;

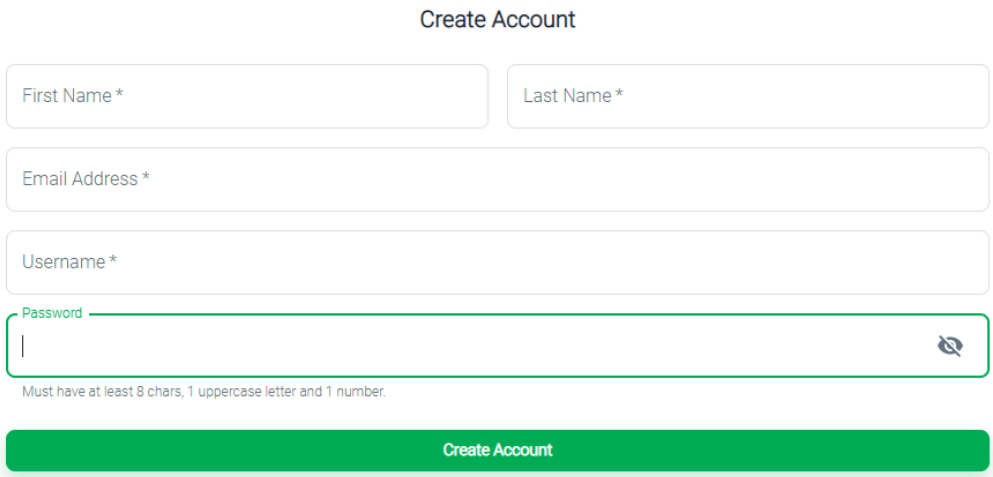


Figure 19: Create account form

Account settings: This page is very simple, only containing a button that allows the user to delete their account;

FAQ: A FAQ page (figure 20) is displayed when the FAQ button is clicked. This page can help new users to get around the platform. When a question is selected, an explanation and a video illustrating the solution are displayed;

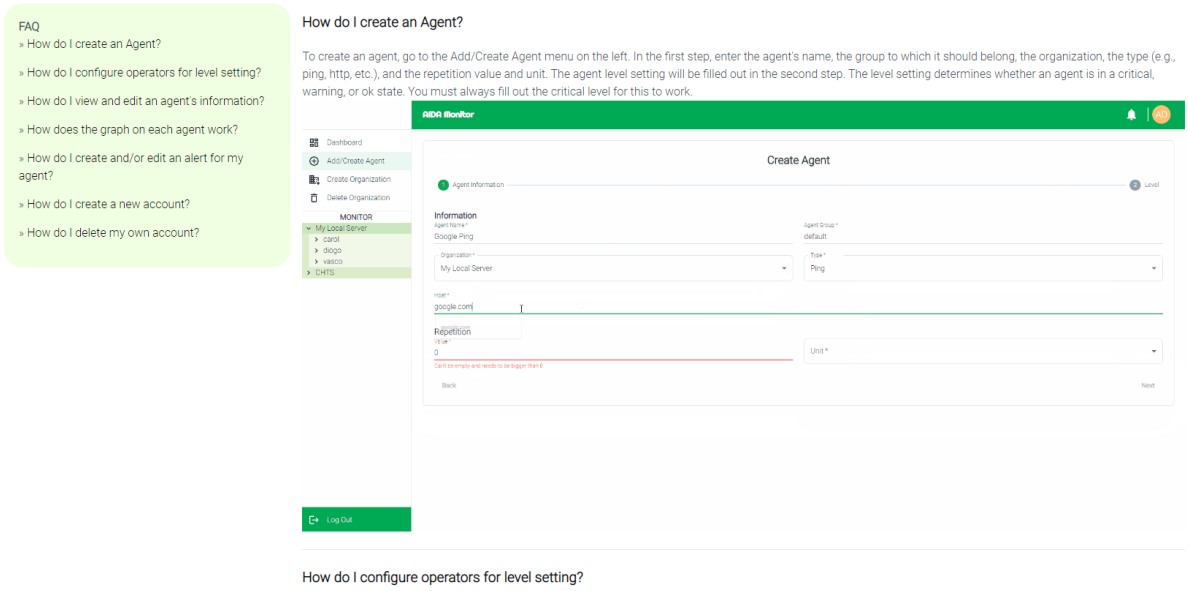


Figure 20: FAQ page

Logout: By selecting this option, the user logs out, and both the authorisation code and username are removed from the *SessionStorage*;

The Notifications feature (figure 21), shows the user how many unread notifications there are. When clicked, it shows a drawer with each received notification, broken down by read and unread (where unread can be marked as read).

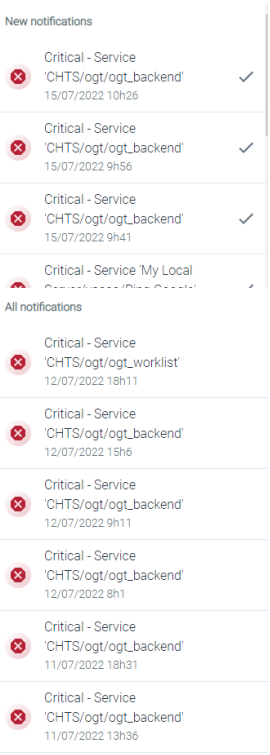


Figure 21: Notification list

6.2.2 Menu

The Menu component, which can be broken down into three sub-components: *Navigation*, *Tree Hierarchy*, and *Logout*, is shown in Figure 22. The *Navigation* component has five buttons that, when clicked, either launch a *dialog* element (the Create and Delete organisation buttons) or open a page (the Agent Overview, Add/Create Agent, and FAQ buttons). Since each component will be explained in the following sections, and since the *logout* sub-component behaves the same way as in the *Navbar*, the main focus will be on the *Tree hierarchy* sub-component.

A service intended to monitor within each hospital is referred to as an Agent. Since there may be numerous Agents requiring monitoring across multiple hospitals, the *Tree Hierarchy* condenses the data and provides visibility to everything on the platform in a tree format. This tree is divided into organisations containing different groups of agents. As a result, clicking on an agent takes the user to that agent's page. This tree sub-component serves as a visibility aid because when users look at this component, they can easily understand where their agents are and to what group and organisation they belong. This sub-component also functions as a navigation aid, as clicking on a tree agent takes you to that agent's page, making it easier to navigate to the desired agent page.

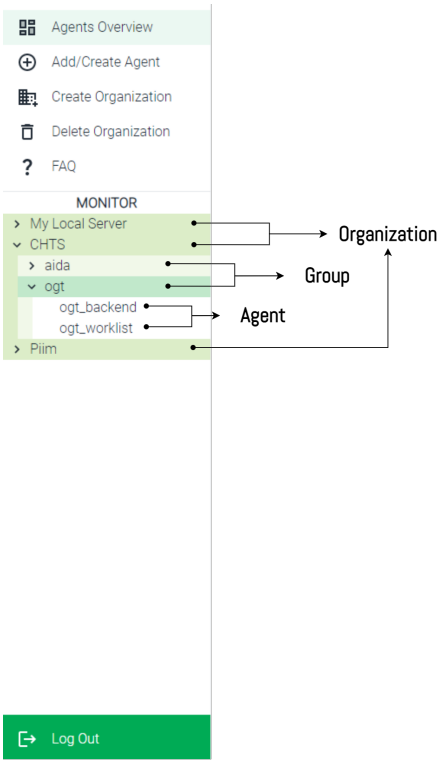


Figure 22: Menu component

6.3 Agents Overview

After a successful login, administrators are redirected to the agent's overview page, figure 23. This page features a table with all the agents and the most recent notifications. On the first component, each table row has the agent name, type group and which organisation it belongs. Users can easily search and sort agents by name, type, organisation, and the group they belong to, and then navigate to the desired agent page by clicking on the respective row. Finally, as the name implies, the other component displays the most recent notifications received. Each message does have a title, a description, and the time the notification was sent. The notifications make it easier to spot which agents' statuses have changed and which ones require attention.

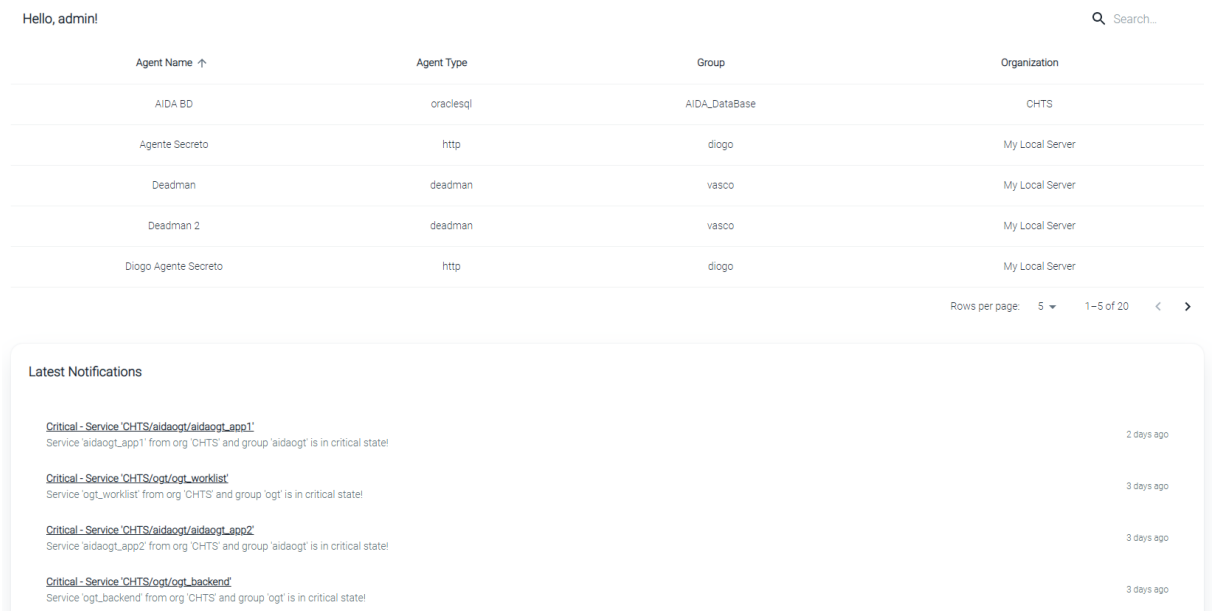


Figure 23: Agents overview page

6.4 Organisation Management

Before diving deeper into agent creation and monitoring, it is important to note how to create organisations, given that an agent must always belong to one. As will be seen in the next section, at least one organisation must be created before creating an agent. It is only necessary to click on the "Create Organisation"menu button, where a *dialog* element will pop up with an input field to enter the organisation's name, figure 24.

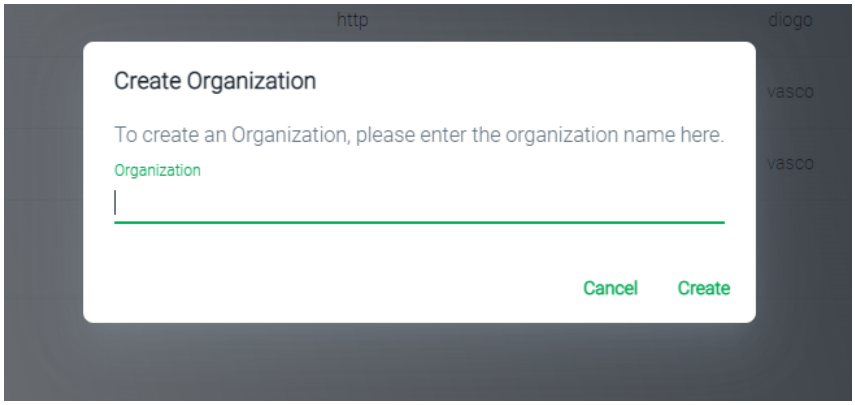


Figure 24: Create organisation *dialog*

To delete an organisation, the user must first click the Delete Organisation button; once clicked, a list of all currently existing organisations appears, from which the user can choose one, as shown in figure 25.

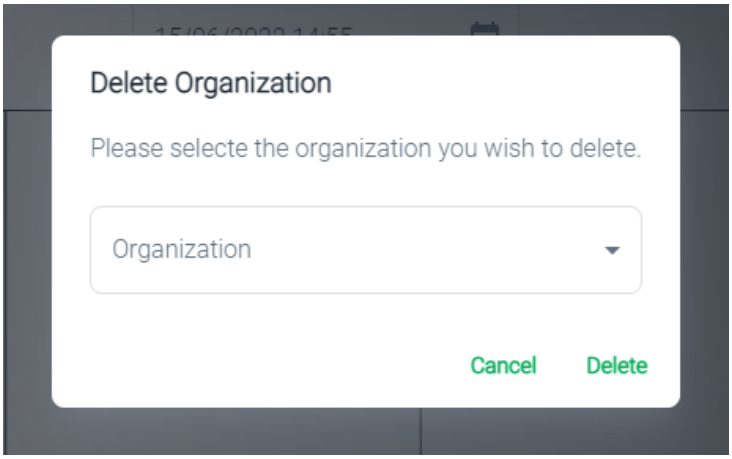


Figure 25: Delete organisation dialog

6.5 Add Agent

As previously stated, clicking a table row on the agents overview page redirects the user to the agent's page. However, this behaviour only happens if the agent has been created. The user should go to this page whenever a new service needs monitoring or when no agents are available.

When on this page, the user have to go through a two-step process to create an agent. As shown in Figure 26, the user is presented with a form that must be filled out with the following agent's primary information:

- Agent Name: the name that will represent the service that needs creation;
- Agent Group: an agent belongs to a specific organisation, but within each organisation, there may be additional divisions, known as groups. Therefore, the agents will be grouped into the default

group if not specified. Furthermore, this division exists only within the platform context to provide a more structured environment for the user;

- Organisation: the organisation to which the agent is assigned. Users can only choose from existing ones, so if none exist, they must first create one;

- Type: the agent's type can be Ping, HTTP, MySQL, PostgreSQL, Redis, Mongo, Deadman, and Nginx. After selecting one, several input boxes appear to allow further agent configuration. For example, to monitor a MySQL database, the user must enter the username, password, host, port, and database name;

- Repetition Value: the time interval between each repetition;

- Repetition Unit: each repetition's time unit can be seconds, minutes, hours, days, or months.

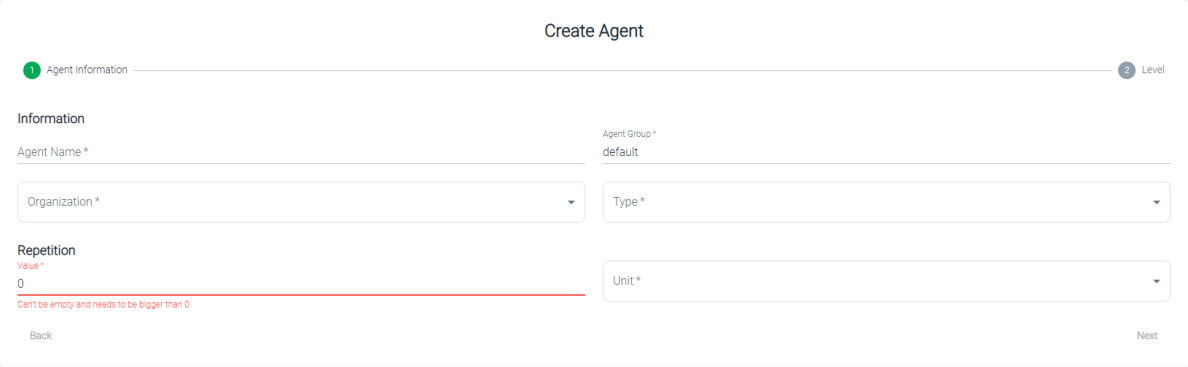


Figure 26: Create agent first step

Once filled out, the user can move on to the second step of the creation process: agent level configuration (figure 27). An agent level tells the system whether an agent is in critical, warning, or OK status, which is used to monitor the agents' health. At least the critical level must be filled in, but it is recommended that both critical and OK states be fulfilled. The user will see three different sections that define what is a critical, warning, and OK setting. When clicking the new criteria button, a row appears. The user can select the monitor level type, the condition, the value, the operator, or click the delete button to remove the criteria.

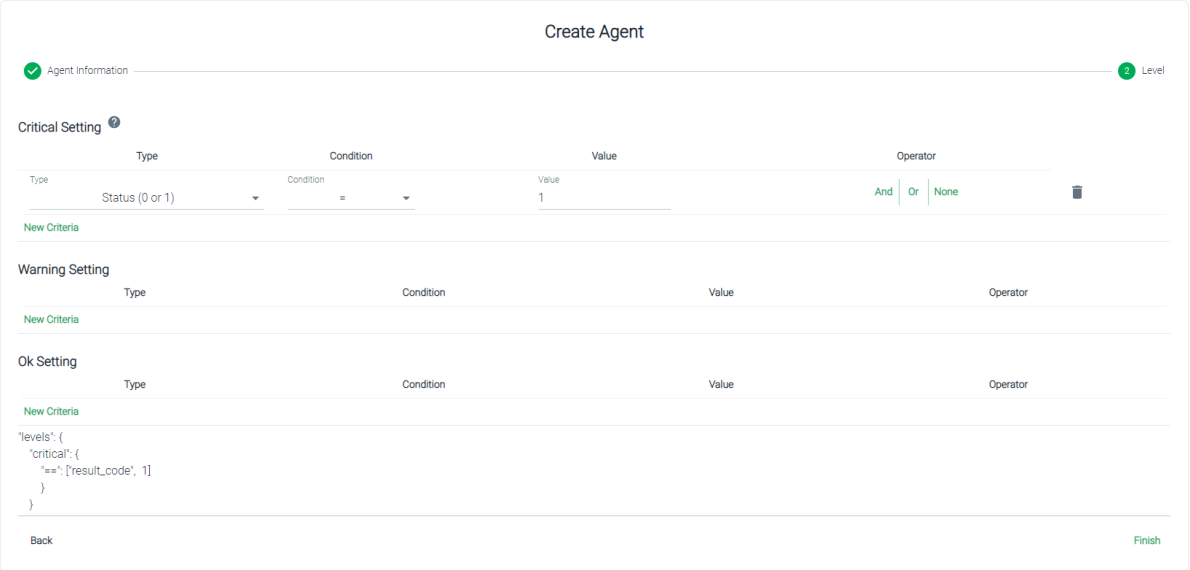


Figure 27: Create agent second step

To understand how a user should interact and define agent levels, the reader can consider the following:

- The user is creating a MySQL agent;
- The agent is in an OK state when the status is equal to zero;
- The agent is in a critical state when the status is equal to one;

To fill this out, the user will add a critical criterion and choose the type Status (0 or 1) with the condition "equal sign"with a value of 1. Then, the user will add the OK criteria and follow the process again, setting the value to 0 this time. A JSON object is generated at the bottom of the page as the user fills out this information. When all is filled out, the user can check if the object is correct. For this scenario, the user will see the object in 1. Next, the user creates the agent. Next, the user creates the agent. If everything is in place, the user is redirected to the Agent summary page; if not, *toasts* elements will appear with error messages providing feedback on what went wrong, figure 28.

```
1       "levels":{
2          "critical":{
3             "==":["result_code", 1]
4          },
5          "ok":{
6             "==":["result_code",0]
7          }
8       }
```

Listing 1: Levels example

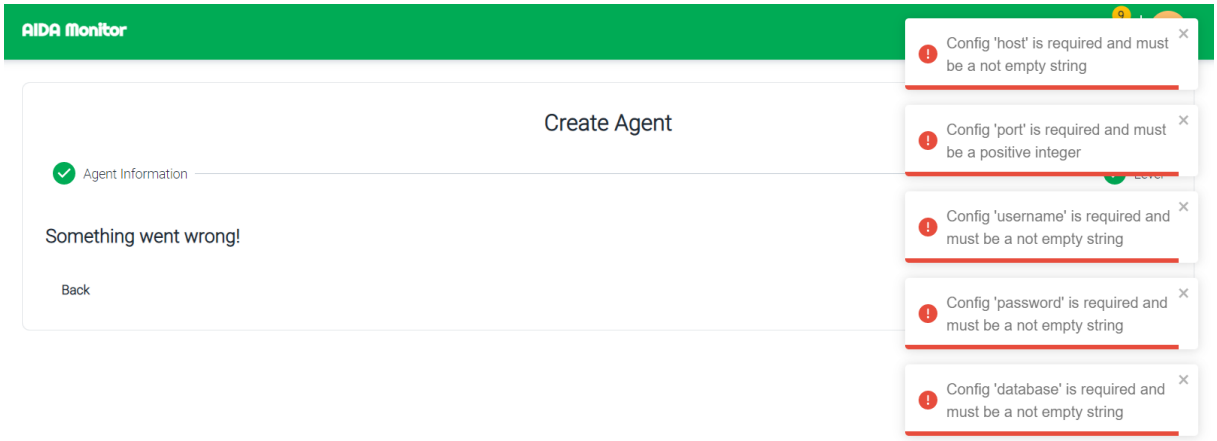


Figure 28: Create agent error

6.6 Agent Page

Successfully created, the user now has an agent that will monitor the service intended. To see the monitoring information and some other features, the user should access the agent page. This page is depicted in figure 29 with the four components that it comprises: Agent Menu, Agent Main Information, Agent Status and Agent Graph Visualisation.

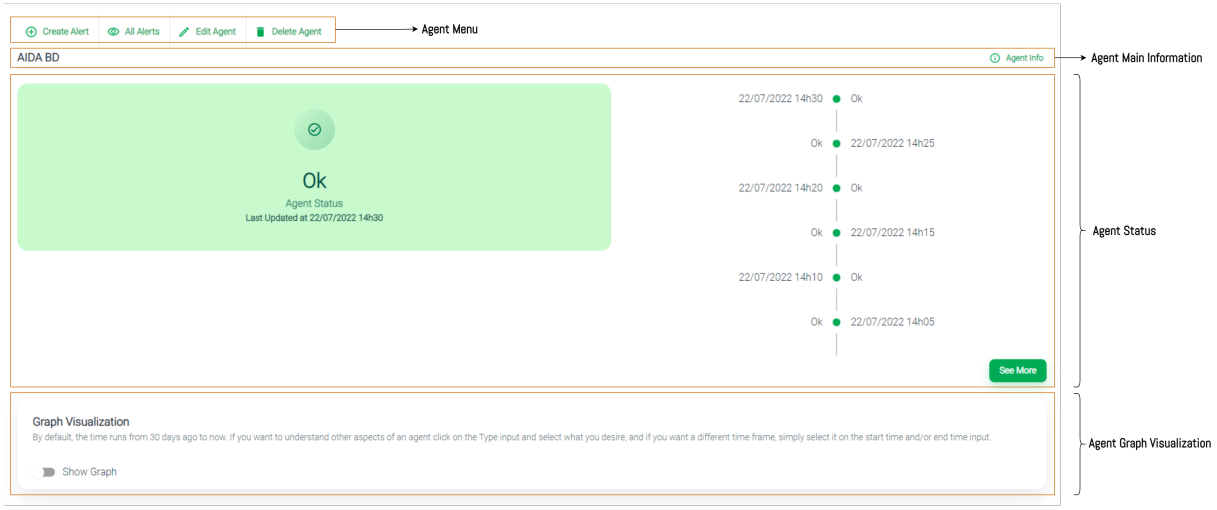


Figure 29: Agent page

Agent Menu is a menu at the top of the page with three options: create alert, all alerts, edit agent and delete agent. When the first option is selected, a page (figure 30) will open, allowing the creation of an alert to receive updates when the agent's status changes. Next, the user creates the agent. The alert can be sent via email, requiring the user to specify which mailing list will receive the alerts. If sent via Slack or Discord, it is only necessary to enter a webhook URL.

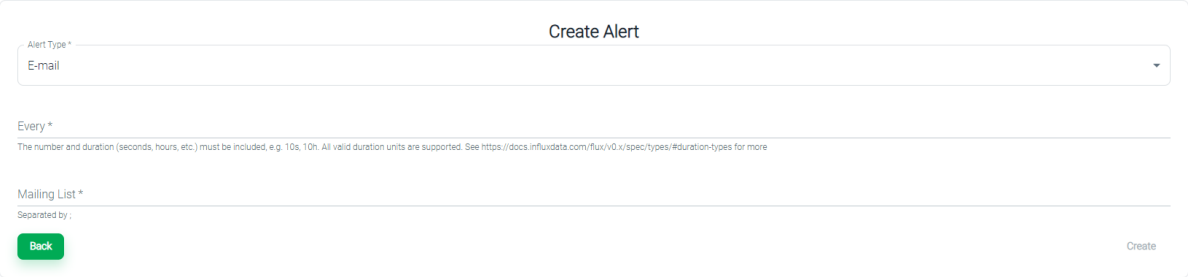
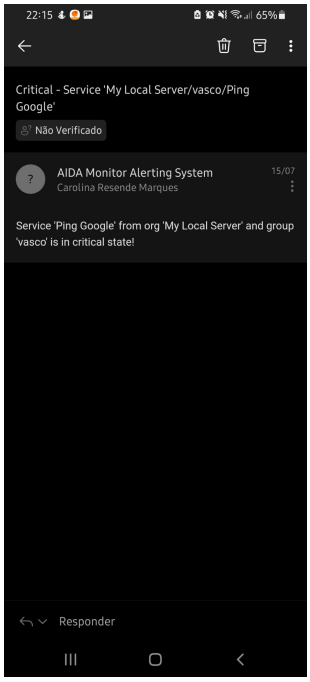
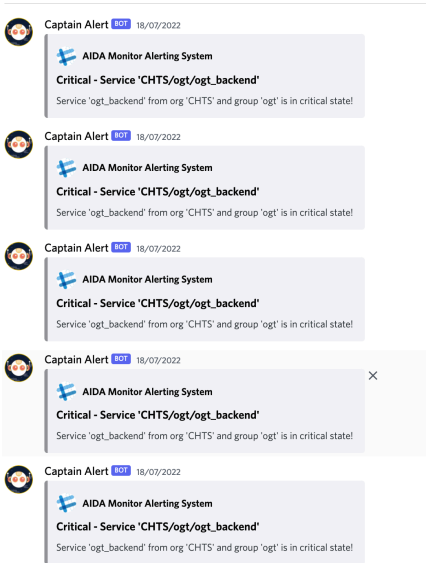


Figure 30: Create alert page

Figures 31a and 31b show what users see when they create an email alert and a Discord alert, and their agent's status changes.



(a) Alert received through email



(b) Alert received through discord

When the second option is selected, a page will be displayed that showcases all existing alerts; if none exist, a simple page will inform the user (figure 32), but if they do (figure 33), a list will appear. The alerts are listed here, along with each status window duration, the option to edit or delete the alert, and a collapsible button that, when clicked, displays more details about the alert, including the date it was created, and the last update.

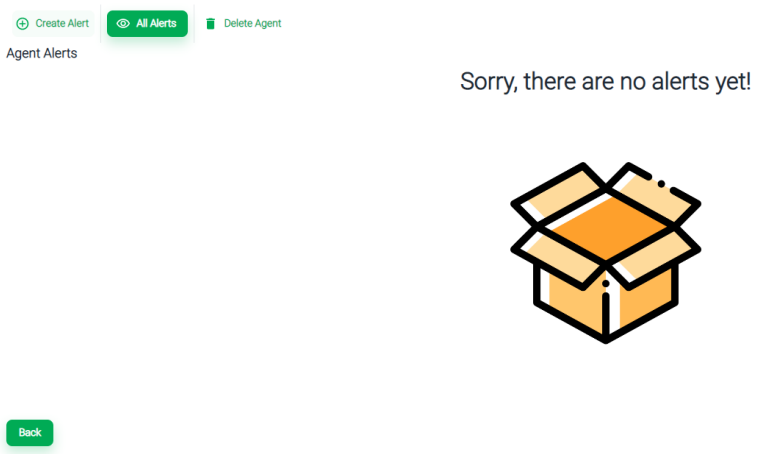


Figure 32: Empty agent alerts page

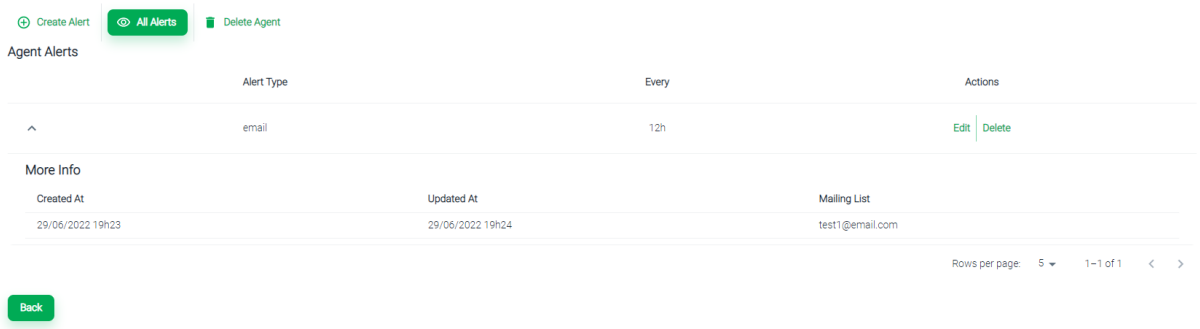


Figure 33: Agent alerts page with alerts

When a user clicks the edit button on an alert, a *dialog* (figure 34) opens in which they can edit the every field (duration of each status window) and the specific field for the alert (mailing list or webhook url). If satisfied, clicking the Save Changes button will update the alert information.

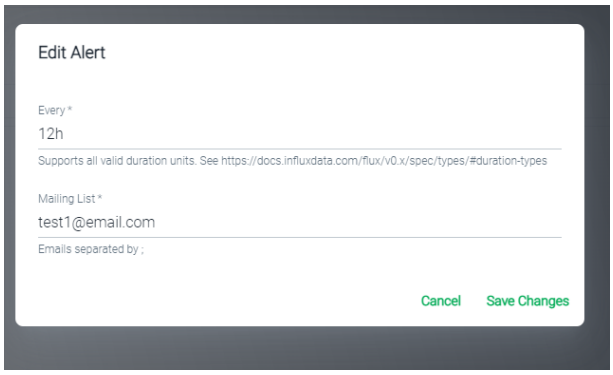


Figure 34: Edit alert dialog

The third option gives the user the option to edit an agent. When clicked, the system displays a form (figure 35) with the following information that can be changed: the agent's name, repetition value, and unit, the organisation and group to which it belongs. The final option, Delete Agent, as the name implies,

will delete the agent when clicked. A confirmation dialog will appear, redirecting the user to the Agents Overview page if approved.

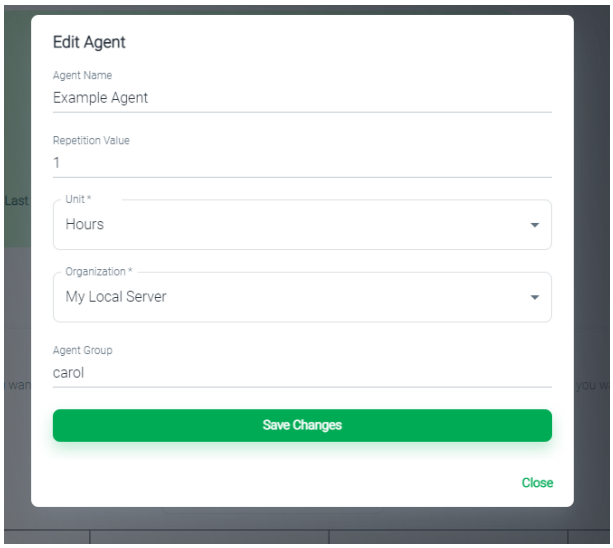


Figure 35: Edit agent dialog

Agent Main Information is a sub-component with the name of the agent and a button that, when clicked, will open a dialog with the agent's main information (figure 36). It includes information about the name, the kind (also known as type), the organisation, and the group to which it belongs. Moreover, it includes the agent configuration with the repetition value and unit, the agent specification (depending on the agent's kind), and the levels configured during the agent's creation.

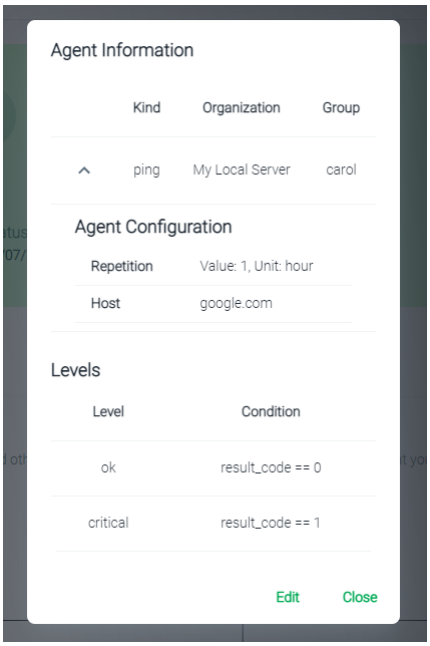


Figure 36: Agent information dialog

Apart from that, the dialog element allows the user to edit the agent, as explained before. **Agent Status** is one of the most important features on this page, since it shows the user how the agent is currently doing in a straightforward way. It consists of a card on the left displaying the agent's most recent status and the time it was collected. The card's colour will change depending on the agent's status: green for OK, yellow for warning, red for critical, and gray for unknown. On the right, it shows a timeline with the six most recent statuses, with the colour of the circles changing as it did in the card (figure 37).

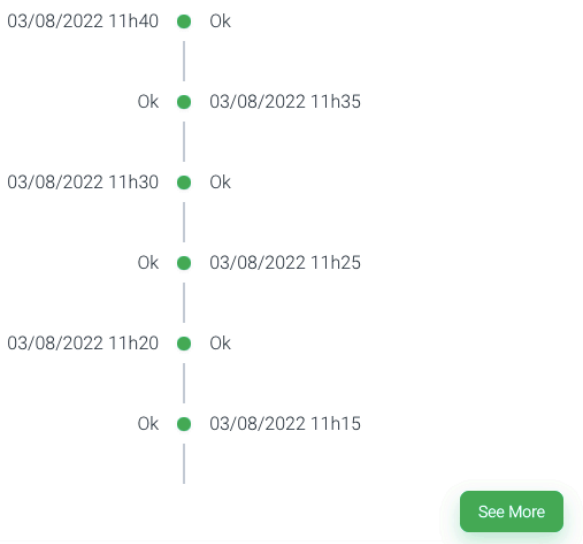


Figure 37: Statuses timeline

When the user clicks the "See More" button, a dialog element (figure 38) opens, displaying a search bar and a table ordered by latest time. The user can search the timeline closely through the dialog to find out when the agent's state changes.

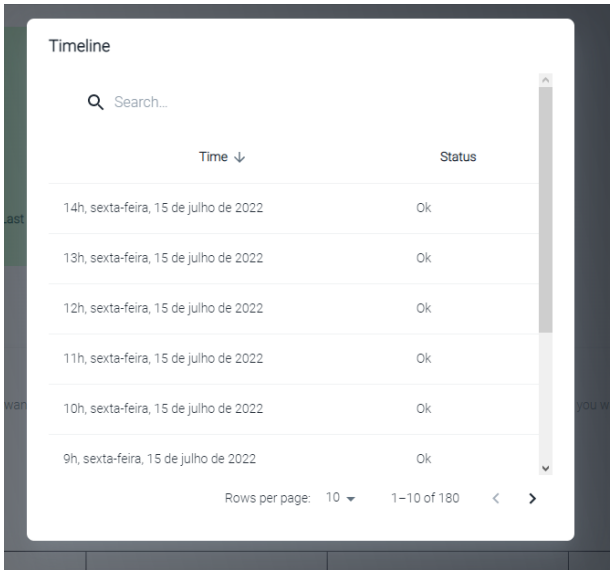


Figure 38: Statuses timeline dialog

As the name implies, **Agent Graph Visualisation** is a feature on the agent's page that allows users to understand specific aspects of the agent through a graph (figure 39). The user can see a selector for the type of information to search for/understand, along with the start and end times to be displayed on the graph. Since the agent represents a ping, the types shown in the figure appear. However, the data in the selector will vary depending on the agent type. By default, the graph's time interval is from 30 days to now.

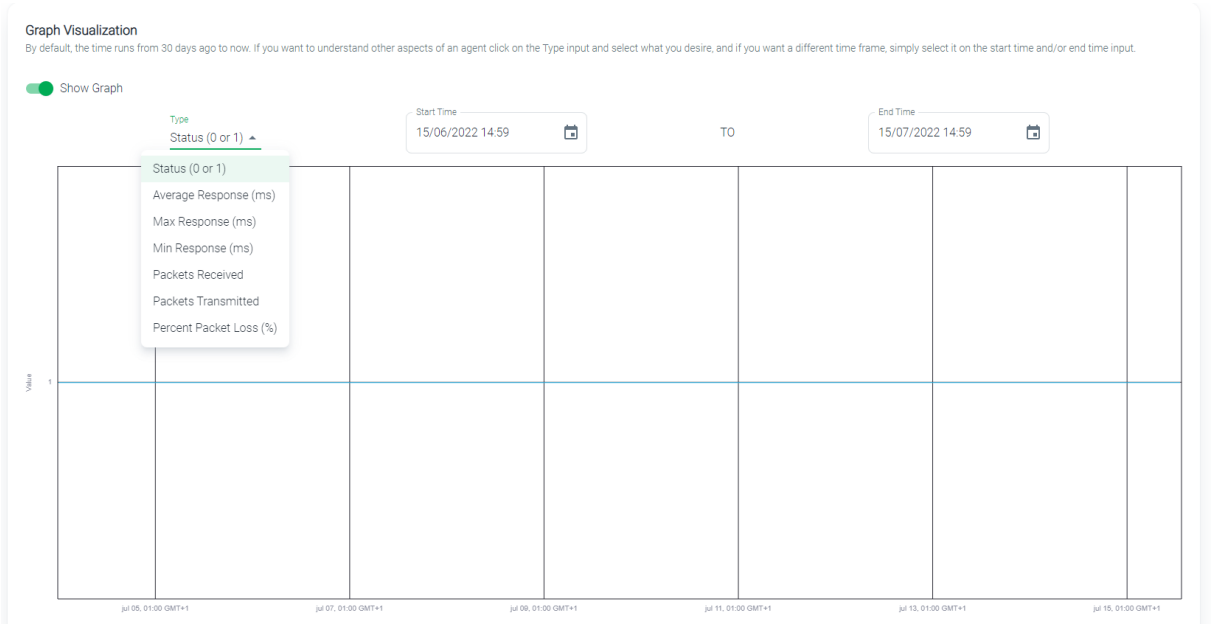


Figure 39: Graph Visualisation

Figure 40 shows an agent's average response from June 15 to July 15. With both the time frame and type selected, the user can see that there have been two peaks with average responses higher than usual, approximately at the same time on different days, suggesting that something needs further investigation. The agent was for an exemplification scenario, so there was no real alert. Yet, if used in a real situation, the user could easily be able to see patterns and figure out what went wrong.

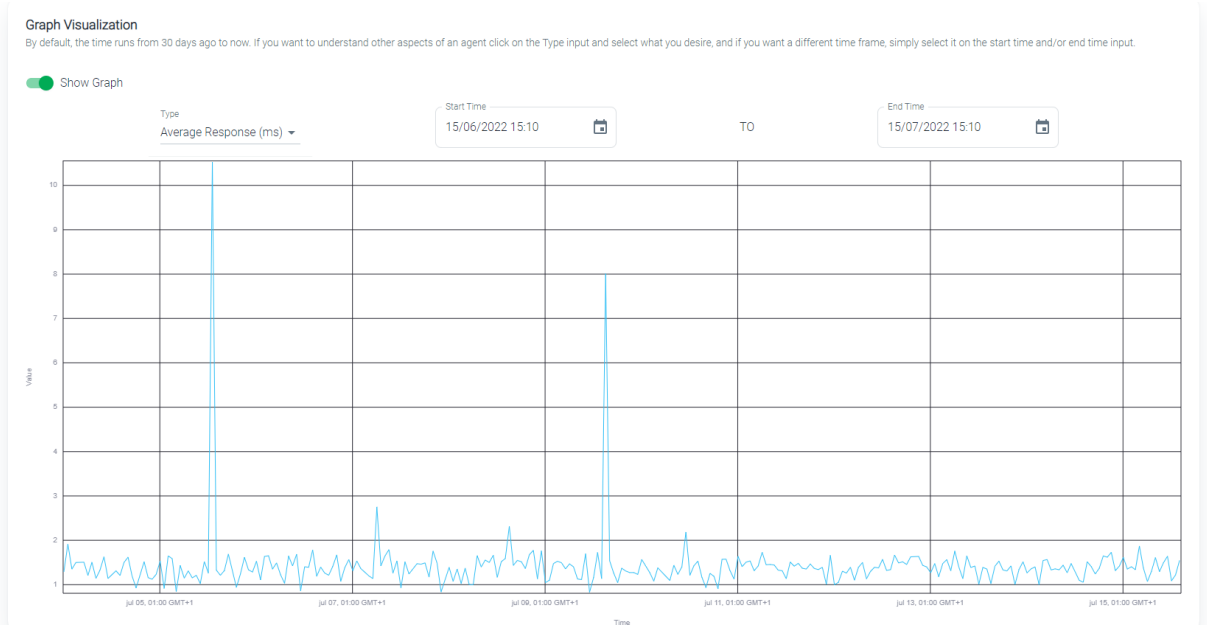


Figure 40: Average response graph from June 15 to July 15

The plots in the preceding figures were created with the Giraffe library (mentioned in 3.2.3). When any field (type, start time, and/or end time) is changed, a POST request is triggered with the start and end times and a query that is built based on the type selected on the Type selector (see 2 for example).

```
{
    "startTime": "2022-07-07T17:30:25.000Z",
    "stopTime": "2022-08-06T16:18:25.730Z",
    "query": "|> filter(fn: (r) => r[\"_field\"] == \"avg_response\")
    |> aggregateWindow(every: 5m, fn:mean, createEmpty: false)"
}
```

Listing 2: POST example of query when user wants to see the average response graph of an agent

The request response follows the CSV format, which the library will then interpret and use it to build the appropriate plot.

6.7 PWA

The implementation should be cross-device compatible, and it happened thanks to the use of PWA. As stated in 3.2.1, PWA are a browser-based web application built with cutting-edge web technologies to provide users with a native app-like experience on any device or channel—mobile, desktop, or, of course, the web. The system was designed with desktop sizes in mind, as the majority of users will use it on desktops or even tablets, so when viewed on smaller dimensions, like a mobile phone, the app may not be as appealing as when viewed on larger dimensions. To overcome this, the solution would be to change

the styling options to consider smaller dimensions and make the transition between desktop and mobile more seamless. Figure 41 depicts the web app use in a mobile phone.

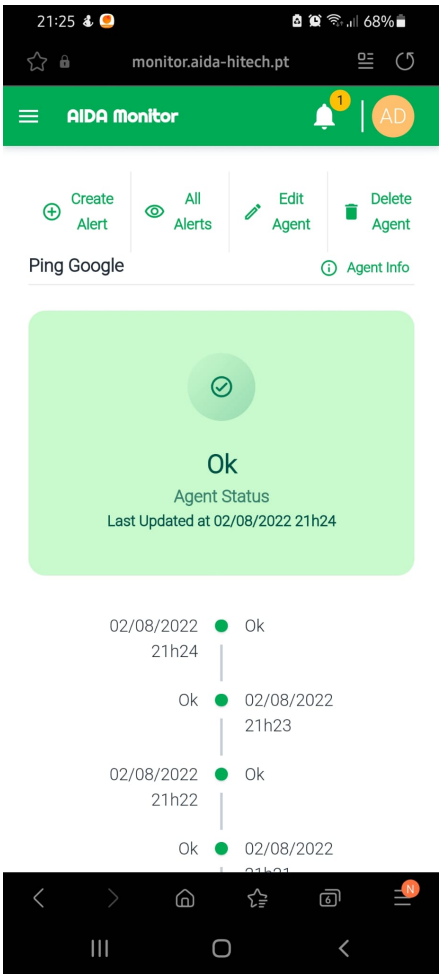


Figure 41: Web app in a mobile phone

Chapter 7

Evaluation

The platform was evaluated at two different moments. The first occurred during the initial platform's development phase, aiming to understand how different healthcare hospital administrators feel about monitoring solutions. The evaluation was crucial to discover how they believed it should behave and assist them on a daily basis. The second moment occurred after the platform development. A questionnaire-based usability test was conducted with users who had no, mild, or extensive experience in the field and type of product. Only the final questionnaire was completed at the CHTS with experienced users, while the rest were completed at the participant's home.

Since the first moment was already explored in section 5.1, this chapter will focus on the second moment of evaluation. At the end, a SWOT analysis will be presented to uncover and examine the overall platform, along with a risk assessment analysis.

7.1 Usability and User Satisfaction Evaluation

The second moment of evaluation occurred during and after development. The simplicity of use and the platform's value in the healthcare environment were assessed.

The questionnaire is divided into two parts. The first part aims to test the aforementioned application by presenting tasks to be completed. The second one includes questions designed to assess the user web platform satisfaction. The first part required users to test and experience the platform to complete tasks such as search, create, and edit agents; manage, create, and delete alerts; and check the agents' health using the graphical visualisation. The second section posed several questions to the user, including the importance of the information shown, whether the metrics made it easy to understand the agents' behaviour, how easy the system was to use and learn, and, importantly, the overall satisfaction with the system. The response scale in the first section varied between 5 response options (Very difficult, Difficult,

Neither difficult nor Easy, Easy, and Very Easy), while in the second section varied between the options, Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree. All questions were mandatory. The questionnaire used is presented in the Appendix B. Similarly to the previous survey, Google forms were used to generate statistical graphs.

7.1.1 Observation Reports

As previously stated, users lack extensive experience in the field, To overcome this, observation reports were conducted with these users. One observer was assigned to users with little to no experience, while the another was assigned to users with extensive experience. Observers observed and recorded user interactions with the system during the testing phase. The observations and reports provided valuable information about the behaviour of participants that questionnaires and interface digital monitoring alone cannot capture. The observation technique was particularly useful in identifying and comprehending system interaction issues, difficulties in understanding the content provided on the screens, device use issues, and user suggestions, among other things.

7.1.2 Results

Only ten participants responded to the questionnaire, with more people without a background than those who had. Given the nature of the population, administrators in a healthcare setting, it is understandable that few people were able to respond.

During the testing phase with users with no to little experience in the healthcare field and administration environment, the observer noticed that most users had little difficulty navigating the platform and performing the required tasks, but when they did, it was usually in the same tasks: searching for the agent's main information and the level creation while creating a new agent. During the testing phase with experts, the observer noticed that most users had difficulty performing some tasks compared to users with no experience. This occurred because the users involved typically had more experience with other platforms, such as Nagios, and to learn the implemented platform, they compared it to their usual tools. This states that they were used to the web app showing only the services and metrics about their hospital rather than showing details about every agent across multiple organisations, so navigation seemed strange at first.

Users also stated that they were unfamiliar with the structure type and that the learning curve was slightly different when compared to users who had never used this type of tool before, because this appeared to new users to be the only way to monitor agents and their metrics. Even though they struggled with the platform at first, once they got used to it, the responses to the tasks were usually around the Easy mark, whereas in the beginning, they were Neutral or even Hard. Even though this occurred, the difficulties in user creation persisted, but some valuable feedback on this was provided.

Participants could provide feedback at the end of the questionnaire if they wanted, to enhance the overall user experience. The following are some examples:

- Make the edit agent creation more intuitive;
- Keep user signed in;
- Latest notifications should appear by default;
- Show more statuses on the agent timeline;
- Display some type of information in the graph visualisation header to help users understand the graph's purpose and components.

7.1.3 Discussion

The questionnaire main goal was to understand how users would interact with the platform and what significance and relevance it had in the healthcare context for monitoring their services. The results may be skewed because not every user who tested had prior knowledge with this environment; yet, the satisfaction part of it may still be accessed when it came to their general view on user satisfaction after learning and utilising the platform. This means that even if they don't understand the platform's relevance, they can still evaluate it as a product. The results are still valuable since if users with little or no experience can use and understand the platform, users more knowledgeable in this area probably can as well.

It is worth remarking that because these results were so valuable, almost every piece of feedback regarding UI and UX was considered and implemented in the final product, and those that did not make it are still helpful for future improvement.

After testing the application, five affirmations were brought up to determine how users felt about the application and its use. The five statements with responses ranging from Strongly Agree to Strongly Disagreed were converted into values ranging from zero to four, with zero indicating Strongly Disagreed and four indicating Strongly Agreed. The following are the statements:

- **Statement 1**: The information provided about each agent is adequate.
- **Statement 2**: The presented metrics make it simple to understand the agent's behaviour to be monitored.
- **Statement 3**: Overall, I am satisfied with the system's user-friendliness.
- **Statement 4**: It was simple to learn how to operate the system.
- **Statement 5**: Overall, I'm pleased with this system.

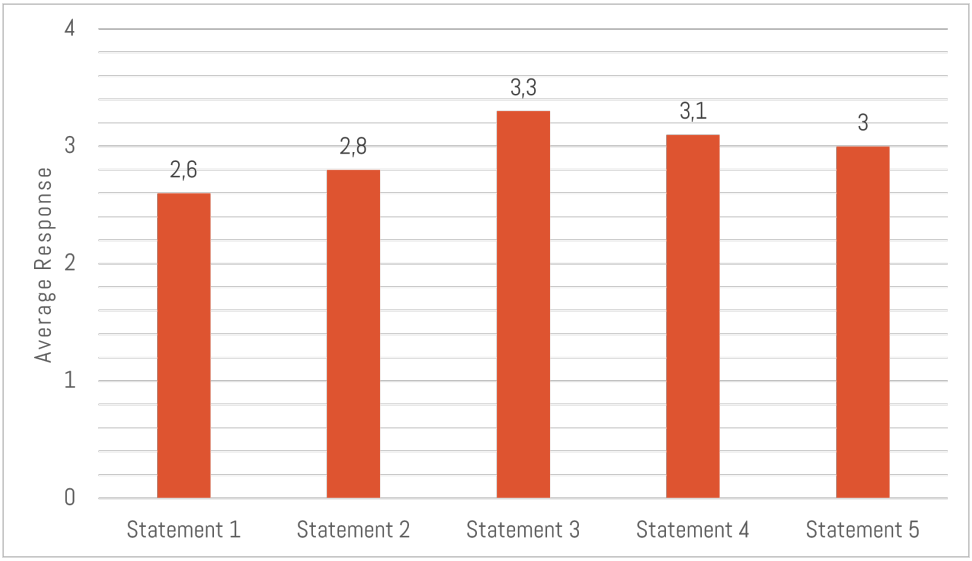


Figure 42: Average response to the affirmations

A graph with the average response for each statement was created and is depicted in figure 42.

The average response is almost always close to three, which translates to Agree. The first and second statement are the ones where more users neither agreed nor disagree. This probably happened because users who had no experience in the health field were not aware as experts of the notion of what is or is not sufficient. When looking at each participant's answer, only one administrator neither agreed nor disagreed, while the others agreed to this affirmation, making the agent information relevant and important to monitoring. For the second statement, most users agreed or strongly agreed with this, but for the same reason as before, the results on the graph do not reflect it. In the third and fourth statements, every user answered since it focuses more on the UI and UX side, and as such, the average response was above three, meaning that most either agreed or strongly agreed. In the final statement, almost every user agreed, indicating that the overall system was satisfactory and relevant to its purpose. All these findings confirmed what had been discussed in the first round of evaluation, indicating that participants saw value in using the platform and that it could be a useful tool for improving overall availability in the healthcare industry. This provides a straightforward answer to the thesis driving question: "Can a multi-platform monitoring application improve IT availability in the healthcare industry? ".

7.2 SWOT Analysis

The SWOT analysis is a strategic planning technique that allows for a critical evaluation of a solution to outline organisational and competitive strategies. A solution's internal analysis is used to identify its resources, capabilities, and inherent competencies to establish a comparison level with similar offerings. Its goal is to gain and apply knowledge about its internal environment and also about the environment in which it operates to best formulate production strategies. Furthermore, SWOT analysis is a conceptually

simplified technique that does not come at a high cost.

SWOTs are organisational influences that stand for strengths, weaknesses, opportunities, and threats. In this context:

- **Strength** is an internal enhancer of competence, a valuable resource, or an attribute;
- **Weakness** is an internal impediment to competence, resources, or attributes required for success;
- **Opportunity** is an external performance enhancer that can be pursued or exploited to gain benefit;
- **Threat** is an external performance inhibitor that has the potential to reduce achievements.

SWOTs are frequently arranged in a 2-by-2 matrix (figure 43), with internal performance enhancers classified as strengths and internal performance inhibitors classified as weaknesses. External enhancers are classified as opportunities, while external inhibitors are classified as threats [54].

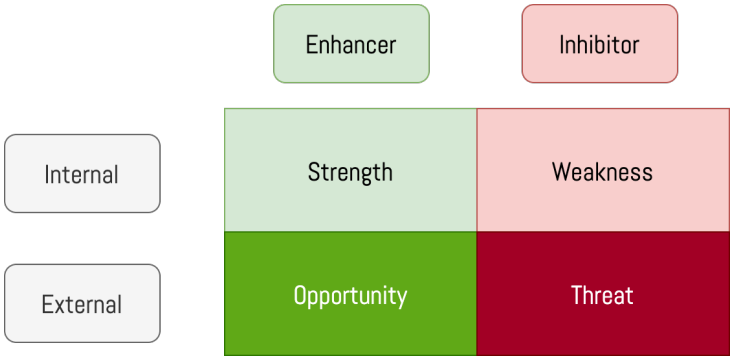


Figure 43: SWOT table [Adapted from [54]]

It is important to understand the platform's strengths and weaknesses, along with the opportunities and threats to be considered. The platform was analysed after being executed in a test environment similar to the real-world environment. The resulting analysis is listed below.

Strengths

- Web-based: Just needed to set up one browser, and it's ready to use;
- User-friendly design: Because a platform of this type should not be difficult to use, the application was designed to be simple to use and navigate.;
- Flux simplification: Professionals assisted and influenced the development process in order to improve the process of monitoring a service;
- Secure: JWT monitors and manages access control to routes and functions, ensuring that only authorised users access the platform;
- Unified View: Unlike the other solutions in the market, the platform shows every hospital and their respective monitoring services in a single dashboard;

Weakness

- Internet Connection: The platform's use is completely dependent on a network connection;
- Notifications: Alerts and notifications play an important role in alerting the user in real time. During development, only toasts were created to display notifications, but this should be done natively through push notifications since the user does not always need to have the platform open. If a failure occurs and no alerts are created, the user will only be able to see what happened if they go to the platform;
- Lack of staff: Compared to other market solutions, the solution only involved two developers with less experience and a short period of development time;

Opportunities

- Improve service quality: Platforms with simple and functional interaction flows enable the service quality improvement provided to users.
- Failure prevention: Identification and analysis of problems in the healthcare field is critical for making users more aware of the state of the system and preventing failures;
- Growth potential: Cross-pollination of ideas between the development team and end users is critical not only for the solution design but also for the increased likelihood of implementing more modern solutions as soon as they are suggested;
- Modernisation: Because the project is new, the platform makes use of the most recent and popular frameworks and languages, allowing it to be a good candidate and competitor in the market when compared to older solutions;

Threats

- Lack of acceptation: Following the implementation of the first version, platform users may not react as expected;
- Familiarisation: Users may be reluctant to switch to a newer platform that is not as well-established as their everyday tools.

Figure 44 depicts the aforementioned points in a SWOT table.

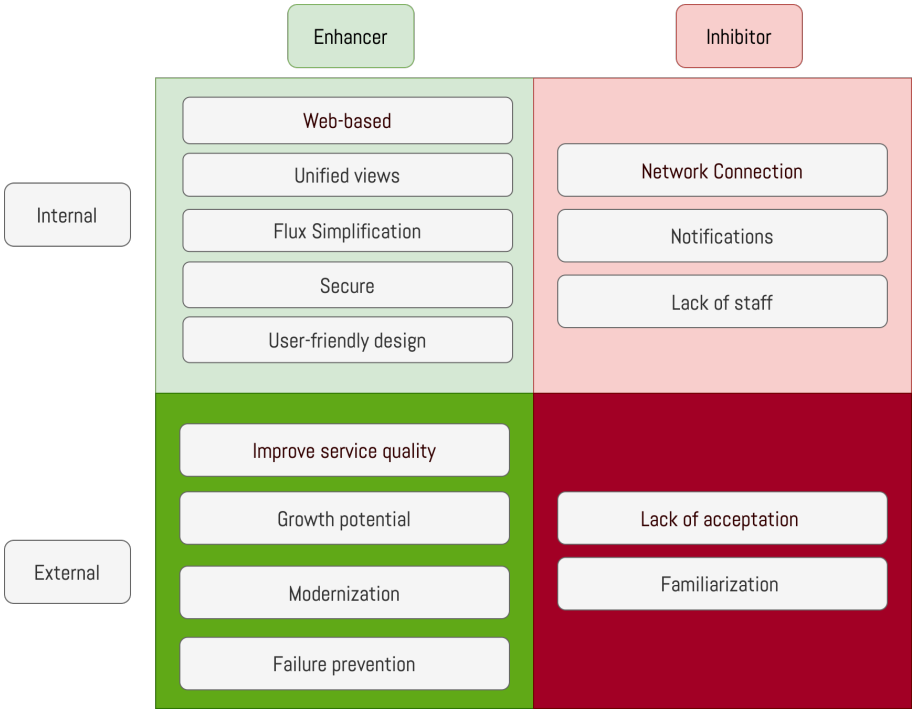


Figure 44: Platform SWOT analysis table

7.3 Risk Assessment

Besides a SWOT analysis, a risk assessment was performed. A risk matrix is built around two intersecting factors: the likelihood of the risk event occurring and the potential impact of the risk event. Simply put, it is a tool that allows to visualise the likelihood vs. severity of a potential risk. The likelihood rating can be divided into five categories, which are listed below:

1. Rare: It's possible, but it's extremely unlikely to happen;
2. Unlikely: The risk event could occur at some time but is unlikely.
3. Possible: A risk event would not be surprising;
4. Probable: A risk event is likely to occur more than once;
5. Almost certain: High likelihood of risk event happening several times.

The consequences can also be divided into five categories:

1. Insignificant: Unlikely to have an impact;
2. Minor: Short-term disruption;
3. Moderate: Significant disruption;

4. Major: Very significant disruption;

5. Catastrophic: Possibly irreversible.

With this, table 3 depicts a table with the risk, the likelihood (**L**) that the risk will occur, the consequence (**C**) that the risk will cause if it occurs, the severity (**S**) of the risk (product of likelihood and consequence), and a mitigation that explains how the risk will be/was overcome.

Table 3: Risk assessment results

ID	Risk	L	C	S	Occurred?	Mitigation
1	The results obtained were unsatisfactory	4	5	20	No	Understand why this happened and try to fix it. If time was needed ask to postpone the artifacts' expected delivery date.
2	Lack of experience with such projects and topics	4	4	16	Yes	Research documentation, articles, tutorials, books, and other materials that could aid in understanding and improving the overall development. Request assistance/suggestions from the advisor and/or colleagues.
3	Issues with application	4	4	16	Yes	Problem identification and subsequent analysis If the impact was significant, perform a rollback and try to resolve the issue so that the next release does not have the same issue.
4	Platform was not user-friendly	4	3	12	No	Go back a few steps and rethink a new mockup. Test mockups until approved.
5	Lack of responses to the inquiries	3	4	12	Yes	If possible, ask colleagues and friends to respond to these inquiries; if not, try to find other people who can fill the role and confirm their availability to respond to the inquiry.
6	This area of investigation is extremely complex	3	4	12	No	Divide a complex topic into less complicated subtopics. Investigate them and try to connect them at the end.
7	The findings did not support the thesis driving question	3	3	9	No	Hypotheses are sometimes proposed, but they are not always proven. This is still a result, so why it didn't work should be discussed.
8	Colleague refused to make the backend data available for consumption	2	4	8	No	Make the advisor aware of the issue. Talk and encourage that colleague to make those resources available. If nothing happens, mock results from the API and display them as is.
9	The chosen methodology was inadequate	2	4	8	No	Look for other available methodologies and select one that is appropriate for the project. Consult with the advisor.
10	Difficulties communicating with the adviser	1	4	4	No	Encourage weekly meetings with the supervisor and make use of appropriate communication channels.
11	Inadequate planning	1	3	3	No	Rethink plans to ensure that they meet expectations.
12	The machine used to work refused to turn on	1	2	2	No	Attempt it on a different machine. If the developer did not have another computer available, go to the university and work on computers there.

Chapter 8

Conclusions and Future Work

After presenting the scope of the project and the problem it seeks to solve, as after reporting the conceptualisation, implementation, and evaluation of the proposed solution, it is possible to summarise the dissertation and draw the main conclusions about the work completed.

8.1 Main Contributions

The healthcare industry is constantly under pressure to improve everything and make healthcare more secure with fewer errors while keeping costs in check. Faced with these numerous challenges, and with the growing popularity of IT around the world, many see an opportunity to use them in order to alleviate some of the sector's existing pains. IT has been proving its worth, but it is still error-prone with failures that can have catastrophic consequences, even leading to deaths. A monitoring solution could be a useful tool in attempting to mitigate such consequences and, ultimately, ensuring that everything runs smoothly. Aside from physical devices, there are critical services that, if shut down, could cause damage, such as web services, servers, and databases. This dissertation proposes a web-based monitoring solution to improve overall availability and, ultimately, ensure that there are as few failures as possible. If they do occur, the solution is hoped to reduce response time, making the failures appear seamless and not a disruption in hospital services.

Some research was conducted in order to gain a better understanding of the problem and to know how to approach and design the proposed solution. Therefore, existing solutions for monitoring tools in the healthcare sector have been investigated in order to learn about their key features and purposes, and how the healthcare sector can benefit from them. When comparing market solutions, it was discovered that in healthcare, they could potentially improve healthcare sector availability by ensuring that everything is operational and that alerts are sent immediately in cases of downtime, for example. Some fundamental

questions have been addressed throughout this research, providing a partial answer to the question of what comprises a good monitoring tool: user-friendly dashboards, timely alerts in the event of downtime or decreased performance, and so on.

Following this research, several front-end frameworks were studied and compared to decide which would be used to develop the web application, with React being chosen. Parallel to this study, a survey was conducted during the initial platform development phase to understand how several hospital administrators at CHTS feel about healthcare monitoring solutions and how they believe they should behave and assist them daily. When asked, users expressed how important a monitoring solution for these types of services is and how effective they can be in improving the overall availability of the system, providing a promising answer to the dissertation's thesis. These findings were not only encouraging, but they also revealed which features are important to have and which major issues the platform should address. Therefore,during this phase, it was possible to collect various system requirements such as use cases, functional and non-functional system requirements, and other information to serve as a guide for the development phase.

Bearing the requirements in mind, the architecture was proposed together with the other colleague in charge of developing the back-end, Vasco Ramos, where the solution's main components are made available through an off-site installation as well as an on-site local server responsible for the organisation's monitoring data collection. The proposed UI will provide a view of this collection by displaying monitoring data and their metrics in a simple and comprehensive manner, allowing different services across different organisations to be monitored. When the architecture was defined, it was critical to go through a UI/UX design process to create a platform that was not only functional but also intuitive and user-friendly, where mockups were created and tested. Once the architecture was defined, it was critical to go through a UI/UX mock-up-based design process to design a platform that was not only functional but also intuitive and user-friendly. The mock-ups were later used during the development phase after being approved.

Towards the end of development phase, when most of the functionality had been implemented an usability test and user satisfaction survey was conducted. However, this time the participants (these included not only hospital IT administrators) could try the application and provide feedback on the application's usability as well as on the platform in general. Overall, the results were promising, indicating that participants could benefit from this platform in service monitoring and improve overall availability in the healthcare industry. The platform was also subjected to formal evaluations, like a SWOT analysis, and a systematic risk assessment and mitigation approach. Both were important in understanding the solution's benefits and limitations more pragmatically, as well as identifying potential dangers and devising a plan to mitigate them.

Therefore, this dissertation contributes with a systematic review of solutions for service monitoring in healthcare settings, the development of a secure UI that allows hospital IT administrators to view and monitor services with multi-platform and multi-organisation functionalities at its core, as well as receive real-time alerts in the event of a status change, being then able to respond quickly to failures. Additionally, platform validation and results revealed that this solution can improve the availability, and subsequently the quality, of healthcare IT services, which was the thesis driving research question.

8.2 Future Work

Some feedback provided during the initial evaluation was not only useful, but also required for a good monitoring platform, so it was used during development. During the second phase of testing, feedback was also provided, and some of it was implemented. Aside from that, during the research, some other aspects were discovered to be interesting but not a priority. With this in mind, some suggestions for future work were gathered during these phases, namely:

- Use predictive models to predict when a failure might occur;
- Provide instructions for resolving specific failures in order to aid and shorten response time.;
- Include push notifications instead of in-app toasts;
- In the second step of the Create Agent page, make a clearer distinction between each setting by either delimiting each card or breaking each setting down into three different steps;
- Make the agent's name the page title on the Agent's Page, making sure it remains visible when you go to the All Alerts page, for example;
- When creating an alert, give it a name so that it is easier to find in the *All Alerts* table;
- Delete organisation could be in the Agent Page menu, alongside create alert, all alerts, edit agent, and delete agent;
- Show every agent that a user has created and be able to edit and delete them in the user settings page;
- Make it possible to read every unread notification at once;
- Create alert and edit alert page could become only one;
- For each institution, a page could be developed to see each agent status and quickly find what agents needs attention;
- Receive alerts through SMS;

8.3 Scientific Contribution

Besides the main contributions presented in section 8.1, this Master´s work also has scientific outcomes, namely:

- Marques, Carolina & Ramos, Vasco & Peixoto, Hugo & Machado, José. (2022). Pervasive Monitoring System for Services and Servers in Healthcare Environment. 10.1016/j.procs.2022.03.097.

- Ramos, Vasco & Marques, Carolina& Peixoto, Hugo & Machado, José. (2022). IT Monitoring in Healthcare: A Case Study.

Bibliography

[1] M. M. Helms, R. Moore, and M. Ahmadi. “Information technology (IT) and the healthcare industry”. en. In: *Int. J. Healthc. Inf. Syst. Inform.* 3.1 (Jan. 2008), pp. 75–92 (cit. on pp. 1, 2, 6).

[2] Y. Alotaibi and F. Federico. “The impact of health information technology on patient safety”. In: *Saudi Medical Journal* 38 (Dec. 2017), pp. 1173–1180. doi: 10.15537/smj.2017.12.20631 (cit. on pp. 1, 7).

[3] M. N. Alolayyan et al. “Health information technology and hospital performance the role of health information quality in teaching hospitals”. In: *Heliyon* 6.10 (2020), e05040. issn: 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2020.e05040. url: https://www.sciencedirect.com/science/article/pii/S2405844020318831 (cit. on p. 1).

[4] J. Grimson et al. “Sharing Health-Care Records over the Internet”. In: *Internet Computing, IEEE. 5. 49-58* 5.10. (2001) (cit. on p. 2).

[5] C. Kumar, C. Rao, and D. Govardhan. “ A Framework for Integration and Standardization of Data from Heterogeneous Sources ”. In: 2012 (Jan. 2012). doi: 10.5923/j.database.20120101.01 (cit. on p. 2).

[6] National Academy of Engineering and Institute of Medicine. *Building a Better Delivery System: A New Engineering/Health Care Partnership*. Ed. by P. P. Reid et al. The National Academies Press, 2005. isbn: 978-0-309-09643-0. doi: 10.17226/11378 (cit. on p. 2).

[7] A. Olson. *Why Health Care Organizations Need a Monitoring and Reporting System*. https://www.avisystems.com/blog/why-health-care-organizations-need-a-monitoring-and-reporting-system. Last Visited December 15,2021. 2018 (cit. on p. 3).

[8] N. Phichitchaisopa and T. Naenna. “Factors affecting the adoption of healthcare information technology”. In: *EXCLI J. 2013;12:413-436. Published 2013 May* 13. (2013) (cit. on p. 6).

[9] *A Dictionary of Physics*. Oxford University Press, 2009. isbn: 9780191726798. doi: 10.1093/acref/9780199233991.001.0001. url: https://www.oxfordreference.com/view/10.1093/acref/9780199233991.001.0001/acref-9780199233991 (cit. on p. 6).

[10] S. J. B. Rich Castagna. *What is information technology?* https://www.techtarget.com/searchdatacenter/definition/IT. Last Visited August 1. Aug. 2021 (cit. on pp. 6, 7).

[11] E. Stolterman and A. C. Fors. "Information Technology and the Good Life". In: *Information Systems Research: Relevant Theory and Informed Practice*. Ed. by B. Kaplan et al. Springer US, 2004. isbn: 978-1-4020-8095-1. doi: 10.1007/1-4020-8095-6_45. url: https://doi.org/10.1007/1-4020-8095-6_45 (cit. on p. 6).

[12] *What is information technology?* https://blog.udemy.com/importance-of-information-technology/. Last Visited August 2. Sept. 2021 (cit. on p. 7).

[13] A. Vegesna et al. "Remote Patient Monitoring via Non-Invasive Digital Technologies: A Systematic Review". In: *Telemedicine and e-Health* 23 (Apr. 2016). doi: 10.1089/tmj.2016.0051 (cit. on p. 7).

[14] J. Barlow et al. "Meeting government objectives for telecare in moving from local implementation to mainstream services". In: *Journal of telemedicine and telecare* 11 Suppl 1 (Feb. 2005), pp. 49–51. doi: 10.1258/1357633054461723 (cit. on p. 7).

[15] D. J. Brailer. "The decade of health information technology, Framework for Strategic Action". In: (June 2004) (cit. on p. 7).

[16] *Information Systems or Information Services (IS)*. https://www.techopedia.com/definition/1027/information-systems-or-information-services-is. Last Visited November 17,2021 (cit. on p. 8).

[17] *Information System*. https://www.britannica.com/topic/information-system/Computer-software. Last Visited November 17,2021 (cit. on p. 8).

[18] W. H. Organization. *Toolkit on monitoring health systems strengthening HEALTH SYSTEMS GOVERNANCE*. 2008 (cit. on p. 9).

[19] *Health information systems*. https://www.undp-capacitydevelopment-health.org/en/capacities/focus/health-information-systems/. Last Visited November 18,2021 (cit. on p. 9).

[20] W. H. Organization. *Health Metrics Network Framework and Standards for Country Health Information Systems*. 2008 (cit. on p. 9).

[21] S. Sinhasane. *What Is Health Information System & Its Importance?* https://mobisoftinfotech.com/resources/blog/importance-of-health-information-system/. Last Visited November 19,2021. 2019 (cit. on p. 9).

[22] *User interface design basics*. https://www.usability.gov/what-and-why/user-interface-design.html. Last Visited August 2 (cit. on p. 9).

[23] *User experience basics*. https://www.usability.gov/what-and-why/user-experience.html. Last Visited August 2 (cit. on p. 9).

[24] S. Gibbons. *Design Thinking 101*. https://www.nngroup.com/articles/design-thinking/. Last Visited August 2. June 2016 (cit. on p. 10).

[25] PAESSLER. “Healthcare IT – a monitoring primer”. In: (2019) (cit. on p. 11).

[26] *Site 24x7 -Healthcare Services Solution*. https://www.site24x7.com/solutions/healthcare-services.html. Last Visited November 01, 2021 (cit. on p. 12).

[27] *Monitor both your healthcare and classic IT infrastructure with PRTG*. https://www.paessler.com/healthcare_it_monitoring. Last Visited November 01, 2021 (cit. on p. 13).

[28] *Nagios Solutions For Healthcare and Medicin*. https://www.nagios.com/solutions/healthcare-medicine/. Last Visited November 01, 2021 (cit. on p. 14).

[29] *Datadog: Healthcare & Life Sciences*. https://www.datadoghq.com/solutions/healthcare/. Last Visited November 01, 2021 (cit. on p. 15).

[30] V. Ahmed, A. Opoku, and Z. Aziz. *Research Methodology in the Built Environment*. 2016. doi: https://doi.org/10.4324/9781315725529 (cit. on p. 17).

[31] J. v. Brocke, A. Hevner, and A. Maedche. “Introduction to Design Science Research”. In: Sept. 2020, pp. 1–13. isbn: 978-3-030-46780-7. doi: 10.1007/978-3-030-46781-4_1 (cit. on pp. 17, 18).

[32] S. Richard and P. LePage. *What are Progressive Web Apps?* https://web.dev/what-are-pwas/. Last Visited December 1,2021. 2020 (cit. on p. 19).

[33] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli. “Progressive Web Apps: The Possible Web-native Unifier for Mobile Development”. In: Jan. 2017, pp. 344–351. doi: 10.5220/0006353703440351 (cit. on p. 20).

[34] Sematex. *The Complete Guide to Metrics, Monitoring and Alerting*. https://sematext.com/blog/monitoring-alerting/. Last Visited December 2,2021. 2020 (cit. on p. 20).

[35] A. Dotis-Georgiou. *Giraffe Visualization Library and InfluxDB*. https://www.influxdata.com/blog/giraffe-visualization-library-and-influxdb/. Last Visited December 4,2021. 2020 (cit. on p. 20).

[36] *Introduction to JSON Web Tokens*. https://jwt.io/introduction. Last Visited December 4,2021 (cit. on p. 21).

[37] M. Kaluža, K. Troskot, and B. Vukelić. “Comparison Of Front-end Frameworks For Web Applications Development”. In: *Zbornik Veleučilišta u Rijeci* 6 (2018), pp. 261–282. doi: 10.31784/zvr.6.1.19 (cit. on p. 22).

[38] M. Laaziri et al. “Analyzing bootsrap and foundation font-end frameworks: a comparative study”. In: *International Journal of Electrical and Computer Engineering* 9 (Feb. 2019), pp. 713–722. doi: 10.11591/ijece.v9i1.pp713-722 (cit. on p. 22).

[39] D. Lakshmi and S. Mallika. “A Review on Web Application Testing and its Current Research Directions”. In: *International Journal of Electrical and Computer Engineering (IJECE)* 7 (Aug. 2017), p. 2132. doi: 10.11591/ijece.v7i4.pp2132-2141 (cit. on p. 22).

[40] *Choose Between Traditional Web Apps and Single Page Apps (SPAs)*. https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps. Last Visited October 26, 2021. 2021 (cit. on p. 22).

[41] *Front-end Frameworks*. https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/. Last Visited October 26, 2021. 2020 (cit. on pp. 22, 24).

[42] N. Pandit. *What And Why React.js*. https://www.c-sharpcorner.com/article/what-and-why-reactjs/. Last Visited October 26, 2021 (cit. on p. 23).

[43] K. Shah. *ReactJS - Introduction, Advantages And Disadvantages*. https://www.c-sharpcorner.com/article/react-introduction-advantages-and-disadvantages/. Last Visited October 26, 2021 (cit. on p. 23).

[44] *Introduction to Angular concepts*. https://angular.io/guide/architecture. Last Visited October 26, 2021 (cit. on p. 23).

[45] M. Rajput. *The pros and cons of choosing AngularJS*. https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html. Last Visited October 26, 2021 (cit. on p. 24).

[46] *Introduction – Vue.js*. https://v3.vuejs.org/guide/introduction.html. Last Visited October 26, 2021 (cit. on p. 24).

[47] *Svelte*. https://svelte.dev/. Last Visited October 26, 2021 (cit. on p. 24).

[48] N. Kramer. *Svelte pros and cons, ecosystem overview and top resources*. https://daily.dev/blog/building-with-svelte-all-you-need-to-know-before-you-start. Last Visited October 26, 2021 (cit. on p. 25).

[49] IBM. *Why is SIEM important?* https://www.ibm.com/topics/siem. Last visited July 24, 2022 (cit. on p. 29).

[50] *Use Cases*. https://www.usability.gov/how-to-and-tools/methods/use-cases.html. Last Visited September 19 (cit. on p. 29).

[51] M. Glinz and R. J. Wieringa. "Guest Editors' Introduction: Stakeholders in Requirements Engineering". In: *IEEE Software* 24.2 (2007), pp. 18–20. doi: 10.1109/MS.2007.42 (cit. on p. 31).

[52] Z. Kurtanović and W. Maalej. "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017, pp. 490–495. doi: 10.1109/RE.2017.82 (cit. on p. 31).

[53] H. Dhaduk. *11 Key Principles to Build Scalable Frontend Projects*. https://www.simform.com/blog/principles-of-scalable-front-ends/. Last visited January 3, 2022. 2021 (cit. on p. 32).

[54] D. Leigh. “SWOT Analysis”. In: *Handbook of Improving Performance in the Workplace: Volumes 1–3*. John Wiley & Sons, Ltd, 2009. Chap. 5, pp. 115–140. isbn: 9780470592663. doi: https://doi.org/10.1002/9780470592663.ch24. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470592663.ch24. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470592663.ch24 (cit. on p. 55).

Appendix A

First Questionnaire

Monitoring System for Services and Servers in Healthcare Environment - Survey

This survey comes in the course of research work in the context of a master's dissertation related to the monitoring of services and infrastructures in healthcare.

Its objective is to try to understand the current status of monitoring services and their use, particularly in the healthcare environment, as well as to understand to what extent monitoring services can simplify systems/infrastructure management processes and also in what way can this happen.

* Required

1. Do you, or your IT department, actively use monitoring systems for your services? *
   ☐ Yes ☐ No

2. **Please, classify the following statements.** Number each statement according to your perspective on it, from 0 to 4. * [0 - Strongly Disagree 1 - Disagree 2 - Neutral 3 - Agree 4 - Strongly Agree]

   a. Currently, whether there is a failure or not, you can observe and control the state of your system.

0	1	2	3	4
Strongly Disagree				Strongly Agree

   b. When there is a failure in the system, that failure is noticeable.

0	1	2	3	4
Strongly Disagree				Strongly Agree

   c. When a failure in a system is detected, you can determine what the implications are and what other services are impacted.

0	1	2	3	4
Strongly Disagree				Strongly Agree

   d. A monitoring system is important for systems management.

0	1	2	3	4
Strongly Disagree				Strongly Agree

e. A good monitoring system makes it possible to improve the availability of systems.

0	1	2	3	4
Strongly Disagree				Strongly Agree

f. A monitoring system capable of analyzing and displaying, in an integrated way, the status of different targets and organizations, streamlines the systems management process.

0	1	2	3	4
Strongly Disagree				Strongly Agree

3. Are you familiar with the detection time in the event of a failure? * (If yes, proceed to the next two questions; otherwise, skip ahead to question 6.)

☐ Yes ☐ No

4. In your current infrastructure, what is the average failure detection time, in minutes?

5. In your current infrastructure, what is the average response time to a detected failure, in minutes?

6. In your opinion, what constitutes a good monitoring system? *

Thank you.

Appendix B

Second Questionnaire

Monitoring System for Services and Servers in Healthcare Environment - Usability Test

In the scope of the master's thesis in Computer Engineering a platform for monitoring services in a hospital environment is being developed.

This questionnaire aims to test this application and this section presents tasks to be executed, where after each one you will answer how easy or difficult it was to complete that task, where 1 is Very difficult and 5 is Very easy. After these tests, a table will be presented in the second section to understand the satisfaction after using the system.

Url: https://monitor.aida-hitech.pt

Username: usabilityTester

Password: a123456B. (period included)

First Section

* Required

1. Find the agent with the name Ping Google.*

1	2	3	4	5
Very Difficult				Very Easy

2. Identify the actual state of the agent Ping Google. *

1	2	3	4	5
Very Difficult				Very Easy

3. Search for information of the agent Ping Google (repetition value, levels, etc).*

1	2	3	4	5
Very Difficult				Very Easy

4. Determine the average response time of the Ping Google agent on July 12th at 10:55 a.m.*

1	2	3	4	5
Very Difficult				Very Easy

5. Search all alerts of Ping Google.*

1	2	3	4	5
Very Difficult				Very Easy

6. Create an e-mail type alert for Ping Google.*

1	2	3	4	5
Very Difficult				Very Easy

7. Edit the previously created alert.*

1	2	3	4	5
Very Difficult				Very Easy

8. Delete the previously created alert.*

1	2	3	4	5
Very Difficult				Very Easy

9. Find the last notification the application has received.*

1	2	3	4	5
Very Difficult				Very Easy

10. Create an organization called Teste.*

1	2	3	4	5
Very Difficult				Very Easy

11. Create a ping agent, in the My Local Server organization, in the Testing Group for google.com, with a repetition value of 1 hour. In the second step, the agent will be in critical state when the status code is 1 or the average response is less than 1000ms, and it will be OK when the status code is 0.*

1	2	3	4	5
Very Difficult				Very Easy

12. Find the created agent in the platform.*

1	2	3	4	5
Very Difficult				Very Easy

13. Change the created agent group to the default group and the organization to Teste.*

1	2	3	4	5
Very Difficult				Very Easy

14. Delete the agent.*

1	2	3	4	5
Very Difficult				Very Easy

15. Delete the Teste organization.*

1	2	3	4	5
Very Difficult				Very Easy

16. Create a new account.*

1	2	3	4	5
Very Difficult				Very Easy

17. Log in to your newly created account.*

1	2	3	4	5
Very Difficult				Very Easy

18. Delete your account.*

1	2	3	4	5
Very Difficult				Very Easy

Second Section - User Satisfaction

* Required

19. **Please, classify the following statements.** Number each statement according to your perspective on it, from 0 to 4. * [0 - Strongly Disagree 1 - Disagree 2 - Neutral 3 - Agree 4 - Strongly Agree]

    a. The information provided about each agent is adequate.

0	1	2	3	4
Strongly Disagree				Strongly Agree

b. The presented metrics make it simple to understand the agents' behaviour to be monitored.

0	1	2	3	4
Strongly Disagree				Strongly Agree

c. Overall, I am pleased with the system's ease of use.

0	1	2	3	4
Strongly Disagree				Strongly Agree

d. Learning how to operate the system was simple.

0	1	2	3	4
Strongly Disagree				Strongly Agree

e. Overall, I'm pleased with this system.

0	1	2	3	4
Strongly Disagree				Strongly Agree

20. Thank you for your cooperation. If you have any suggestions, please answer them here.

Appendix C

Publications

C.1 Pervasive Monitoring System for Services and Servers in Healthcare Environment

Authors: Carolina Marques, Vasco Ramos, Hugo Peixoto, and José Machado

Title: Pervasive Monitoring System for Services and Servers in Healthcare Environment

Conference: The 5th International Conference on Emerging Data and Industry 4.0

Year of publication: 2022

Abstract: Information systems are continuously evolving in nature and complexity. In the healthcare environment, information and information exchange are critical for providing health care at all levels. Hence, the healthcare environment is particularly relevant when discussing IT infrastructure monitoring and disaster prevention since availability and communication are vital for the proper functioning of healthcare units, whether acting in isolation or on a network.

This work focuses on understanding what comprises a good monitoring solution, analyzing the monitoring solutions currently available in the optics of a multi-location healthcare environment and, finally, proposing a pervasive and comprehensive conceptual architecture for a monitoring system that is capable of handling such environments.

Keywords: Health Information Systems, IT Monitoring, Microservices

State of Publication: Published

C.2 IT Monitoring in Healthcare: A Case Study

Authors: Carolina Marques, Vasco Ramos, Hugo Peixoto, and José Machado
Title: IT Monitoring in Healthcare: A Case Study
Conference: The 6th International Conference on Information Technology & Systems
Year of publication: 2023 (Estimated)

Abstract: The healthcare environment is particularly relevant when discussing information technology infrastructure monitoring since availability and communication are vital for the proper functioning of healthcare units. It is important to be able to easily monitor and observe each unit from a single point of access so that actions can be swiftly taken when there is a problem.

This paper proposes a multi-site and multi-organization web and microservices-based information technology infrastructure monitoring solution. In addition to exploring the developed system and its architecture, it presents a case study resulting from the system's implementation in an organization and holds a discussion about the obtained results to determine whether a multi-platform monitoring system improves information technology availability in the healthcare industry.

Keywords: IT Infrastructure Monitoring, Microservices, Health Information Systems, Healthcare

State of Publication: Under Review