UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Mateus Gonçalo do Nascimento

A self-adaptive IoT architecture to support computational resource allocation in an e-health environment

Juiz de Fora

2024

Mateus Gonçalo do Nascimento

A self-adaptive IoT architecture to support computational resource allocation in an e-health environment

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: José Maria Nazar David

Coorientador: Mario Antonio Ribeiro Dantas

Juiz de Fora

2024

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Gonçalo do Nascimento, Mateus.
A self-adaptive IoT architecture to support computational resource allocation in an e-health environment / Mateus Gonçalo do Nascimento. – 2024.
106 f. : il.

Orientador: José Maria Nazar David
Coorientador: Mario Antonio Ribeiro Dantas
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2024.

1. Self-adaptive architecture. 2. Internet of Things. 3. Computational resource. 4. Artificial Intelligence. I. David, José Maria Nazar, orient. II. Dantas, Mario Antonio Ribeiro, coorient. III. Prof. Dr.

Mateus Gonçalo do Nascimento

A self-adaptive IoT architecture to support computational resource allocation in an e-health environment

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 21 de março de 2024.

BANCA EXAMINADORA

Prof. Dr. José Maria Nazar David - Orientador

Universidade Federal de Juiz de Fora

Prof. Dr. Mario Antonio Ribeiro Dantas - Coorientador

Universidade Federal de Juiz de Fora

Prof. Dr. Regina Maria Maciel Braga Villela

Universidade Federal de Juiz de Fora

Prof. Dr. Gabriella Castro Barbosa Costa Dalpra

Centro Federal de Educação Tecnológica de Minas Gerais

Juiz de Fora, 07/03/2024.

Documento assinado eletronicamente por **Jose Maria Nazar David**, **Professor(a)**, em 21/03/2024, às 16:09, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mario Antonio Ribeiro Dantas**, **Professor(a)**, em 22/03/2024, às 12:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gabriella Castro Barbosa Costa Dalpra**, **Usuário Externo**, em 23/04/2024, às 10:40, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Regina Maria Maciel Braga Villela**, **Professor(a)**, em 30/04/2024, às 12:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1741211** e o código CRC **0BEE7C38**.

To my friends and family.
To my love and my parents.
For your support and for believing in me

AGRADECIMENTOS

First, I thank God for always guiding me, illuminating my life, allowing me to go in search of my victories, giving me health, and blessing me with great people around me.

I would like to thank my family, who always supports me at different times, regardless of the difficulties and even from a distance. In particular, my mother and grandmother, who, even after my father's death, never left anything wanting and provided me with everything I needed to get me this far. Without them, none of this would have been possible.

I am grateful to my father. Even though he was no longer present, he always looked out for me and helped me become the man I am today. To my brothers, godfather, godmother, and cousins, thank you for all your support and encouragement, and never stop believing in me.

I would also like to thank my beloved for always supporting me and always being present, never giving up helping, and encouraging me to achieve all my goals.

I thank my friends, those who have been with me since I was a child, and those who have become close to me throughout my life for your constant companionship.

Special thanks to Professors José Maria Nazar David and Mario Antonio Ribeiro Dantas for their guidance and co-supervision since graduation. Thank you for your friendship, tips, advice, and, above all, your patience, without which this and other work would never have been done.

I am deeply grateful to the Postgraduate Program in Computer Science at the Federal University of Juiz de Fora for providing all the support, structure, and teaching material to make this work possible, even during the Covid-19 pandemic.

More special thanks go to the professors of the Department of Computer Science and the Postgraduate Program in Computer Science for their teachings, in addition to the course staff, who contributed in some way to our personal and professional enrichment during these years.

I would like to acknowledge Professor Fernando Antonio Basile Colugnati from the Department of Medicine at the Federal University of Juiz de Fora, who helped in the initial part of the work with data and experiments on intelligent environments of interest for study.

I express gratitude to NEC Latin America S.A. for providing access to the Aurora SX Tsubasa supercomputer, which allowed experiments and the collection of fundamental data for part of the work.

I am grateful to Repesq and Professor Marcelo Moreno, thank you for providing access to the area's computational resources to make experiments available to the academic

community and civil society in Juiz de Fora.

I am also grateful to the media outlets Globo, Tribuna de Minas, and the UFJF Portal for disseminating part of the experiment of this work for civil and academic society, entitled "Simula Covid-19 UFJF", as a tool for aid and decision-making in the fight against Covid-19.

Finally, I would like to thank Petrobrás, through Project RED, for allowing me to research with the company and promote resources for developing different works in the area.

Try it once, twice, three times and if possible try the fourth, the fifth and as many times as necessary. Just don't give up on the first few attempts, persistence is a friend of conquest. If you want to get to where most people don't go, do the that most don't (Bill Gates).

RESUMO

Ambientes inteligentes são espaços complexos de interação entre pessoas, sensores, dispositivos e sistemas. A Internet das Coisas (IoT) proporcionou, nos últimos anos, a exposição gradual da sociedade a esses ambientes. Porém, a Engenharia de Software requer técnicas específicas para lidar com o desenvolvimento desses sistemas. Portanto, a Engenharia de Software deve abordar características intrínsecas de dispositivos e sensores; interações complexas em ambientes inteligentes; aspectos relacionados à qualidade, como escalabilidade e interoperabilidade; e a capacidade de consolidar boas práticas de desenvolvimento. A utilização de recursos computacionais pode ajudar a apoiar a construção de sistemas IoT contemporâneos, lidando com as complexidades e requisitos de qualidade apresentados. O objetivo principal deste trabalho é apresentar uma arquitetura de IoT auto-adaptativa para alocação de recursos computacionais, com a ajuda de inteligência artificial, em um ambiente inteligente de e-health. A arquitetura aborda como diferentes módulos cooperam e interagem para permitir o desenvolvimento de aplicações de IoT. A avaliação do trabalho ocorreu através de um estudo de caso real em ambiente inteligente de e-health. A metodologia de pesquisa utilizada foi Design Science. Os resultados mostraram a eficiência da arquitetura auto-adaptativa proposta utilizando inteligência artificial. Através dos resultados obtidos por meio da avaliação conduzida: (i) foi possível observar como a arquitetura se comporta em um ambiente inteligente de e-health; (ii) como ocorre a auto-adaptação dos recursos computacionais de acordo com as necessidades do ambiente; (iii) como a inteligência artificial pode apoiar a utilização de recursos computacionais em ambientes inteligentes de e-health; (iv) como podem ser realizados estudos de caso reais em ambientes de e-saúde para ajudar as empresas a tomar decisões sobre a gestão do seu espaço físico. Ademais, os resultados obtidos demonstram como uma arquitetura auto-adaptável utilizando inteligência artificial pode apoiar a gestão de um espaço físico inteligente de e-health. Com isso, foi possível observar como pode ocorrer a coleta de dados, o monitoramento do ambiente, a previsão do uso de recursos computacionais e a otimização do gerenciamento do ambiente.

Palavras-chave: arquitetura auto-adaptativa; IoT; alocação de recursos computacionais; inteligência artificial; ambientes de e-health.

ABSTRACT

Intelligent environments are complex interaction spaces between people, sensors, devices, and systems. The Internet of Things (IoT) has gradually exposed society to these environments in recent years. However, Software Engineering requires specific techniques to deal with the development of these systems. Therefore, Software Engineering must address intrinsic characteristics of devices and sensors, complex interactions in intelligent environments, quality aspects, scalability and interoperability, and the ability to consolidate good development practices. Using computational resources can help support the construction of contemporary IoT systems, dealing with the complexities and quality requirements presented. The main objective of this work is to define a self-adaptive IoT architecture for allocating computational resources in an intelligent e-health environment with the help of artificial intelligence. The proposal concerns how architectural modules cooperate and interact to develop new applications. The work was developed through a real case study in an e-health environment. The research methodology used was Design Science. The results showed the efficiency of the proposed self-adaptive architecture using artificial intelligence. Through the results acquired: (i) it was possible to observe how the architecture behaves in an intelligent e-health environment; (ii) how computing resources self-adapt according to the needs of the environment; (iii) how artificial intelligence can support the use of computational resources in intelligent e-health environments; (iv) how real case studies can be carried out in e-health environments to help companies make decisions about the management of their physical space. Furthermore, the results demonstrate how a self-adaptive architecture using artificial intelligence can support the management of an intelligent physical e-health space. With this, it was possible to observe how data collection, environmental monitoring, prediction of the use of computational resources, and environmental management optimization can occur.

Keywords: self-adaptive architecture; IoT; computational resource allocation; artificial intelligence; e-health environments.

LISTA DE ILUSTRAÇÕES

Figure 1 – Study selection process 32
Figure 2 – First IoT architecture developed 46
Figure 3 – Siafu execution of Federal University of Juiz de Fora 47
Figure 4 – Dashboard application developed 48
Figure 5 – DSR three-cycle model 53
Figure 5 – DSR model proposed by Peffers et al. (2007) 54
Figure 7 – Architecture proposed in the first DSR cycle 57
Figure 8 – Developed application with dashboards 59
Figure 9 – Self-adaptive architecture proposed in the second DSR cycle 61
Figure 10 – Manager layer and its components 62
Figure 11 – Presentation of components to perform the change of computational resources 64
Figure 12 – Managed layer and its components 66
Figure 13 – Computational resource workflow 67
Figure 14 – Architecture DSR assessment 71
Figure 15 – Coworking environment 73
Figure 16 – Waiting room environment 73
Figure 17 – Totem and Tablet 74
Figure 18 – Temperature data stored model 76
Figure 19 – Hasura architecture 77
Figure 20 – Developed application 77
Figure 21 – Custom function as a service code 84
Figure 22 – YAML file 85
Figure 23 – Availability presence detector computational resource 88
Figure 24 – Developed application for the first scenario 89
Figure 25 – Developed application for the second scenario 90

LISTA DE TABELAS

Table 1 – PICOC method 30
Table 2 – Academic Work Base and Number of Works found 31

LISTA DE ABREVIATURAS E SIGLAS

E-health	Electronic health
IoT	Internet of Things
DSR	Design Science Research

SUMÁRIO

1 INTRODUCTION 15
1.1 MOTIVATION 15
1.2 PROBLEM 18
1.3 OBJECTIVE 19
1.3.1 Main objective 19
1.3.2 Specific objectives 19
1.4 RESULTS 20
1.5 ORGANIZATION 20
2 THEORETICAL BACKGROUND 21
2.1 INTERNET OF THINGS 21
2.1.1 Internet of Things and Software Engineering 21
2.2 SMART LIVING 22
2.2.1 E-health 23
2.3 COMPUTATIONAL RESOURCES 24
2.3.1 Cloud computing 24
2.3.2 Fog computing 25
2.3.3 Edge computing 25
2.4 CONTEMPORARY SOFTWARE ARCHITECT 26
2.4.1 Self-adaptive software architect 26
2.5 ARTIFICIAL INTELLIGENCE 27
2.6 FINAL CONSIDERATIONS OF THE CHAPTER 28
3 METHODOLOGY 29
3.1 SYSTEMATIC MAPPING 29
3.1.1 Planning 30
3.1.2 Conducting 31
3.1.2.1 Search process 31
3.1.2.2 Study selection 31
3.1.2.3 Quality assessment 34
3.1.2.4 Data extraction 34
3.1.3 Reporting 35
3.1.3.1 What IoT architectures focus on computational resources in intelligent e-health environments? 35
3.1.3.2 What quality attributes are essential for IoT architectures in intelligent e-health environments? 36
3.1.3.3 What computational resources have been applied to IoT architectures in intelligent e-health environments? 37

3.1.3.4 *How have Artificial Intelligence and computational resources been used in intelligent e-health environments?* . . . 37
3.1.4 **Discussion** . . . 38
3.1.5 **Threats to validity** . . . 39
3.2 RELATED WORKS . . . 40
3.3 FINAL REMARKS OF SYSTEMATIC MAPPING AND RELATED WORKS 43
3.4 EXPLORATORY STUDY . . . 44
3.4.1 **Scenario** . . . 44
3.4.2 **Proposed solution** . . . 46
3.4.3 **Evaluation** . . . 47
3.4.4 **Discussion** . . . 49
3.5 FINAL REMARKS OF THE CHAPTER . . . 51
4 **PROPOSED ARCHITECTURE** . . . **52**
4.1 DESIGN SCIENCE RESEARCH METHODOLOGY . . . 52
4.2 REQUIREMENTS . . . 54
4.2.1 **Functional requirements** . . . 55
4.2.2 **Non-functional requirements** . . . 55
4.3 FIRST CYCLE . . . 56
4.4 SECOND CYCLE . . . 59
4.4.1 **Architecture development** . . . 60
4.5 FINAL REMARKS OF THE CHAPTER . . . 69
5 **EVALUATION** . . . **70**
5.1 METHOD . . . 70
5.2 CONTEXT . . . 72
5.3 PLANNING . . . 72
5.4 CASE STUDY 1 - FIRT DSR CYCLE . . . 74
5.4.1 **Preparation** . . . 75
5.4.2 **Conducting** . . . 77
5.4.3 **Analyzing** . . . 78
5.4.4 **Results** . . . 79
5.5 CASE STUDY 2 - SECOND DSR CYCLE . . . 80
5.5.1 **Preparation** . . . 81
5.5.2 **Conducting** . . . 82
5.5.3 **Analyzing** . . . 86
5.5.4 **Results** . . . 86
5.6 CONCLUSIONS FROM CASE STUDIES . . . 91
5.7 THREATS TO VALILIDTY . . . 94
5.8 FINAL CONSIDERATIONS OF THE CHAPTER . . . 94
6 **CONCLUSIONS** . . . **96**

REFERÊNCIAS **99**
APPENDIX A – Journalistic publications about the exploratory study **106**

1 INTRODUCTION

This chapter presents the motivation, the problem to be addressed, the objective to be achieved, the acquired results, and the organization of the work.

1.1 MOTIVATION

The Internet of Things can be defined as a conceptual framework that unites the availability of heterogeneous devices and their interconnection with physical objects. This interconnection between physical objects and heterogeneous devices allows the representation of people, objects, and physical spaces. These representations occur through data generated by IoT devices and sensors, which portray the reality of the environment in which they are inserted. (Atzori, Iera and Morabito, 2017).

IoT has consolidated itself in people's daily lives and in the interactions they carry out with the environment in which they are (Fetahu, Maraj and Havolli, 2022). The impact becomes more relevant in people's lives as the population's expansion and integration into new environments of daily use occur, such as medical clinics, hospitals, and public agencies (Brous, Janssen and Herder, 2020). Environments that use IoT characterize intelligent environments or smart ecosystems (Pradhan et al., 2017). The IoT is responsible for allowing different devices and sensors to connect through the internet, allowing the sharing and collection of data about the environment (Gomez et al., 2019).

The expansion of intelligent environments has made new business opportunities and services available. E-health applications provide examples of intelligent environments. These include remote patient monitoring, telemedicine with consultations using video calls and communication systems, innovative medication to send patient reminders and notifications, and intelligent hospitals and clinics that can monitor these environments (Gomez et al., 2019).

Even though it is present in people's lives and brings many benefits, the introduction of IoT brought several complexities to software development. Teams have to deal with some challenges (Amershi et al., 2019), such as distributed devices and sensors, heterogeneous configuration of behavior, intrinsic characteristics, and integration and interoperability with other sensors and systems (Motta, De Oliveira and Travassos , 2018).

The challenges brought by IoT are more expansive than just the characteristics of these sensors and devices. The systems interacting with these sensors and devices are critical parts of the intelligent environments in which they are involved (Hajjaji et al., 2021). Software must deal with the connectivity, management, scalability, and integration of sensors and devices in different contexts and quantities. Finally, developers are also affected, given the non-consolidation of good development practices, causing the arrival of low-quality systems on the market (Larrucea et al., 2017).

The dynamics of intelligent environments are challenging topics for software development. The reasons are related to several factors, such as the large number of interconnected devices and the different people that interact with these environments, in addition to the challenges of ensuring that devices and sensors are always connected and communicating efficiently. It is also worth mentioning that adding and removing new devices and sensors in the space context and new types of interaction further increase the complexity of these environments (Ahmed et al., 2016).

Furthermore, the suitability of the computational resources needed for the systems dealing with sensors and devices is also challenging (Laroui et al., 2021). Using computational resources not suitable for applications can lead to the development of systems that fail to meet their objectives or requirements, generating dissatisfaction and problems in their users' daily lives. Approaches based on edge, fog, and cloud computing are examples of approaches to address these questions. Costa, Nassar and Dantas (2023) use edge, fog, and cloud to deal with the environment's needs and the nature of sensors and devices. The serverless paradigm is another example of computational resource applications for IoT. Cassel et al. (2022) present the benefits of using serverless. These benefits are related to development agility, scalability, high availability, fault tolerance, and development without a server.

With the expansion of IoT and intelligent environments, modern software engineering aims to support the development of systems for intelligent environments (Larrucea et al., 2017). For this, software engineer experts study how different elements of these environments interact, behave, communicate, and cooperate, seeking to provide techniques and methods to assist in developing applications and software (Samizadeh Nikoui et al., 2021). Examples of dealing with these challenges proposed by contemporary software engineering are IoT computational architectures, which define standards to be followed by developers to achieve quality attributes systems (Richards and Ford, 2020). Ray (2018) presents how an IoT architecture should behave in the face of the challenges presented, listing how architectures should behave and presenting functional blocks. These blocks are related to devices, communications, services, management, security, and applications.

Like IoT, smart cities have recently been developed worldwide (Khan et al., 2020). The concept of smart cities is related to using different technologies to improve the lives of the population and the efficiency of management and use of public and private spaces (Albino, Berardi and Dangelico, 2015). One of the concepts related to smart cities is smart living, which refers to the use of technologies by society to live with greater quality and convenience (Han and Kim, 2021). Examples of smart living are related to smart homes, intelligent lighting, health, and well-being, among others (Kumar, 2020).

The Internet of Things is one of the technologies that enabled the development of smart living. With devices and sensors, individuals and companies could equip different

environments with items for monitoring and managing spaces, such as cameras, thermostats, and presence sensors. The IoT paradigm allowed these environments to be built through network protocols, data collection and processing, and systems developed for the environment (Pathan et al., 2021).

In the context of smart living, e-health is an area that has an intersection between medicine, public health, business, and information and communication technologies. E-health aims to allow the creation of any service, platform, or tool that can be used electronically (Eysenbach et al., 2001). More specifically, in smart living, e-health is related to applications in complex, intelligent environments, which allow the area to be computerized (Wilson et al., 2021).

An e-health environment is where interactions between sensors, devices, systems, and people can be essential to a person's life. Applications in the context of e-health deal directly with other people's lives, seeking to facilitate and support the population's daily lives (Zadtootaghaj et al., 2019). In complex and vital scenarios, such as e-health, failure to establish efficient computational architectures for these environments can lead to low-quality software.

IoT systems can use data generated by sensors and devices to adapt to situations occurring in the environment. The self-adaptation characteristic may occur due to the user's need, the environment in which sensors and devices are found, or even to meet an environment's security and functioning requirements (Da Silva, Gonçalves and Dantas, 2019). For example, in an e-health system that monitors patients in real-time, devices and sensors can stop sending patient data when the system detects an electrical power outage. Due to the complexity of the patient monitoring scenario, the system needs to adapt to new conditions, or a patient may be at risk. Self-adaptive architectures are being used in other application domains. An example can be seen in Gomes et al. (2023), where the authors present a self-adaptive IoT architecture for the livestock domain. Artificial intelligence techniques were used to support decision-making.

Artificial intelligence (AI) has been used in applications for the e-health environment and self-adaptive architectures. AI has supported decision-making for different users in computer systems, such as the example seen in Kishor and Chakraborty (2022), in which the authors present a monitoring system using artificial intelligence. In self-adaptive architectures, the use of AI has helped to understand the needs of the environment and facilitate adaptation to new configurations, as in the example in Lahini et al. (2023), in which the authors present a dynamic product line for healthcare platforms, which adapts to the needs of the environment.

1.2 PROBLEM

Given how IoT computing architectures for intelligent e-health environments have been used, gaps can be observed in using computational resources related to cloud computing, fog computing, and edge computing. These computational resources have been used to load, process, and store large amounts of data, to allow the integration of different sensors and devices, to ensure low response time and adequate flow to sensors and devices, and to allow decision-making by interested parties appropriately.

Although used for different purposes, IoT architectures have used computational resources uninterruptedly. Regardless of the environment's needs, these computational resources are always available to applications, devices, sensors, and actors. Its indiscriminate use causes computational resources to be wasted without taking advantage of the capacity to be made available, resulting in more significant financial expenses and non-use of the available capacity.

Some examples of what the wasteful use of computing resources can result in are financial losses; excessive consumption of electrical energy; greater exposure of computing resources to privacy and security issues; infrastructure overload, in case of many parallel services required, increasing maintenance costs and decreasing service quality; risks of persistent attacks; resistance to scalability of resources, as they are always available at high capacity.

Integrating computational resources, self-adaptive capabilities, and artificial intelligence can provide new approaches to deal with the problems and challenges encountered in developing IoT architectures for e-health environments. This architecture, with several approaches together, seeks to allow (1) the development of applications for the e-health environment; (2) the use of computational resources must be carried out only on demand; (3) artificial intelligence assists in the management of computational resources; (4) process IoT data and sensors efficiently.

Consequently, based on the challenges presented, the research problem addresses the lack of support for building IoT architectures for e-health environments, focusing on using computational resources supported by artificial intelligence. The designed architecture aimed to adapt to the needs of the e-health environment and allow the efficient use of computing resources. The following research question was established to develop the architecture: "How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?".

1.3 OBJECTIVE

1.3.1 Main objective

This work seeks to support the management of intelligent e-health environments by developing a self-adaptive IoT architecture supported by artificial intelligence using computational resources. The intelligent environment used to evaluate the architecture was a corporate e-health environment.

Using computational resources and artificial intelligence, we seek to understand the complex dynamics in IoT e-health environments. Using computational resources aims to meet the quality attributes inherent to IoT architectures, such as scalability, extensibility, performance, and interoperability. Artificial intelligence seeks to support when these computational resources should be used to assist in processing data generated by devices and sensors.

1.3.2 Specific objectives

The specific objectives of the proposal address:

- Evaluate the use of a self-adaptive architecture in e-health environments.
- Evaluate the use of computational resources together with self-adaptive architectures in intelligent e-health environments
- Evaluate the use of artificial intelligence to support the management of computational resources in intelligent e-health environments.
- Evaluate the proposal's suitability for a real corporate e-health environment.

To achieve the main goal proposed for the architecture, some specific goals were defined before the architecture proposal, they were:

- The architecture must be capable of processing data close to the generation source.
- The architecture must provide computational resources for IoT devices and sensors when needed.
- The architecture must allow the processing of different data types generated by sensors and IoT devices.
- The architecture must receive new sensors and IoT devices and allow the reception of new computational resources.
- The architecture must have a component responsible for making the necessary adaptations to computational resources.

- Computational resources should not be active when not used.
- Implementing the architecture in a real corporate e-health environment must be possible.

1.4 RESULTS

Following the Design Science Research (DSR) methodology, we developed the current work. This methodology assumes artifacts (solutions) are designed to solve a practical problem (Simon, 1996). Two DSR cycles were executed using the presented methodology, and artifacts were developed, generating results through the architecture development and two case studies.

The results of the case studies made it possible to verify how e-health environments use an IoT architecture. Through the implementation of the proposed architecture and case studies, the results were related to data capture and analysis, physical space management, self-adaptation to environmental scenarios, and optimization of the use of computational resources.

As a main contribution, we present the architecture to support the development of IoT applications in an e-health environment. In addition to architecture, the analysis of computational resources and artificial intelligence related to architecture also made it possible to establish ways of using them. This architecture aims to use computational resources according to the demands of the environment in which it is inserted and allow efficient data processing. Furthermore, quality attributes related to the architecture are presented and discussed, which assist in building knowledge of the state of the art.

1.5 ORGANIZATION

This work is organized into six chapters. The following chapter discusses the theoretical reference background. Chapter 3 presents the methodology. Chapter 4 discusses the architecture proposal. Chapter 5 shows how we developed an application detailing a case study. Finally, the last chapter presents the conclusion and future work.

2 THEORETICAL BACKGROUND

This chapter presents the theoretical background that provides the conceptual knowledge necessary to understand the developed work. The concepts related to self-adaptive architecture, the internet of things, smart living, computational resources, and artificial intelligence are addressed, as they are related to the main scope of this work.

2.1 INTERNET OF THINGS

The Internet of Things, or IoT, can be defined as a conceptual framework that unites the availability of heterogeneous electronic devices and sensors and their interconnection with physical objects in different environments. The interconnection between physical objects, devices, and sensors allows for representing physical environments (spaces), people, and objects, such as cameras, presence sensors, and temperature sensors. These representations occur through data generated by IoT devices and sensors, which portray the reality of the environment in which they are inserted. (Atzori, Iera and Morabito, 2017).

The use of the Internet of Things has gained prominence in recent years. IoT is used in various areas of everyday life, impacting the lives of people, public bodies, and private companies. Examples include but are not limited to smart cities, industry 4.0, electricity, transport and logistics, and the health sector (Hassija et al., 2019). The healthcare sector is having to deal with IoT in its daily life. Employing IoT, for example, to monitor patients, intelligent medical equipment, medication management, hospital bodies, and monitored clinics, among others, are just some of the usage demonstrations (De Michele and Furini, 2019).

2.1.1 Internet of Things and Software Engineering

To meet IoT's diverse areas of interest, Computer Science must also deal with several challenges. Software engineering has been impacted by the large-scale introduction of the Internet of Things (IoT) paradigm (Spinellis, 2017). The characteristics of IoT devices and sensors brought complexities to software development. Examples include the heterogeneity of each device, communication protocols, the number of computational resources available, scalability, interoperability, and integration with computational systems. Therefore, software designed for intelligent environments must deal with these challenges (Motta, De Oliveira and Travassos , 2018).

Chen et al. (2019) present challenges regarding the heterogeneity of IoT devices and sensors. The work presents that devices and sensors use different hardware resources and operating systems. In addition to the characteristics of devices and sensors, they can use different network protocols to communicate with systems and other devices. Finally,

the processing capacity of sensors and devices can be different, which can lead to different response times, data generation, and resource delivery.

Ahmed et al. (2016) show the difficulties related to scalability and interoperability in developing IoT applications. The authors demonstrate that the difficulties became even more evident with the massive increase in devices used daily. These devices exist in different environments, such as homes, hospitals, medical centers, and buildings. One of the significant challenges for modern software is the expansion of the number of devices and sensors, which generate large amounts of data and need to use various systems that already exist or are yet to be built. When adding new devices, scalability from computational resources and systems is necessary. Interoperability is present when devices and sensors need to communicate with other systems.

Therefore, integrating IoT devices and sensors with computational systems is complex (Larrucea et al., 2017). The individual characteristics of each device or sensor are a critical factor. Furthermore, these sensors and devices often need to connect with legacy systems and existing computational infrastructures. The limited computing capacity of devices and sensors and the amount of available computing resources they must use can cause problems in software delivery.

The challenges related to the systems that must deal with IoT devices and sensors are related not only to their characteristics but also to studies and opportunities to use better quality attributes (Kashani et al., 2021). The expansion of IoT, along with other technologies such as big data, artificial intelligence, edge, fog, and cloud computing, has made software development face new challenges to satisfy the demands of software consumers.

The demand for large-scale data processing is one of the needs that systems have to deal with. IoT devices and sensors in intelligent environments generate large amounts of data. Methodologies and computational approaches are necessary to extract, process, and store this data for decision-making. Of of the goals of software engineering is to deal with those methodologies and computational approaches. Examples of these approaches are ETL (extract, transform and load), ELT (extract, load and transform) and data streaming. To tackle this situation, researchers and software developers must propose solutions capable of handling significant data processing and storage (Shandilya, Chun and Shandilya, 2022).

2.2 SMART LIVING

Smart cities can be defined as urban areas that use information technology to improve the lives of their citizens, allowing efficiency in the use of public and private resources and providing effectiveness in urban services. These cities comprise various sensors, devices, and platforms that collect and send data to their stakeholders (Albino,

Berardi and Dangelico, 2015).

Given their technological aspect, smart cities are presented as spaces with six main components: smart people, smart mobility, smart economy, smart environment, smart government, and smart living. Smart people are related to citizens who can interact, contribute, and understand how the smart urban environment works. Smart mobility is related to the sustainable use of public or private transport that adapts to the population's needs. The smart economy is about an environment conducive to innovation, allowing the creation of qualified, sustainable jobs that can help improve the population's quality of life. Environmental monitoring and management technologies ensure that intelligent environments can be built. These technologies allow us to create more sustainable cities with a better quality of life, helping us understand the environment that the population lives in. The concept of smart government is directly related to using information and communication technologies to assist efficiency in public management, provide transparency and accountability, and expand public policies in people's daily lives. Finally, smart living is related to using information and communication technologies in public and private spaces in the best way. These spaces include homes, schools, public and private buildings, constructions, and health organizations. These environments are enriched by the use of IoT devices and sensors (Kumar, 2020).

One of the technologies that makes smart living environments possible is the Internet of Things. The application of devices and sensors in intelligent environments helps develop innovative spaces that support the daily life of society (Hossinian-Far, Ramachandran and Slack, 2018). Thus, examples of smart living applications are smart homes, which have devices and sensors allowing home automation (Sovacool and Del Rio, 2020); urban and residential security, which use tracking sensors and cameras to monitor urban and residential environments (Toma et al., 2019); medical environments, which have devices and sensors to monitor patients, environments and the dynamics of environments (Selem, Fatehy and Abd El-Kader, 2019).

2.2.1 E-health

Electronic health, or e-health, is defined as an area that has an intersection between medicine, public health, business, and information and communication technologies, which aims to create any service, platform, tool, or method of intervention that can be used electronically (Eysenbach et al., 2001) (Wilson et al., 2021).

E-health and smart living are connected through IoT devices and sensors. E-health has benefited from the growth of smart living in recent years, allowing intelligent environments to use health platforms, services, and tools to assist professionals in the field and the population (Semwal et al., 2019). An example can be seen in Larcher et al. (2020), which uses an assisted home environment, using IoT devices and sensors to

monitor patients and assist in their medical monitoring.

IoT has been used in the e-health sector to support patients, healthcare professionals, hospital environments, and medical clinics, but it is not restricted to these environments. Nizetic et al. (2020) present areas of initiatives that use IoT within e-health. Some of these initiatives address innovations to support patients, such as remote monitoring and medication management. New services, such as intelligent clinics, have been created in other cases. The development of e-health software and the existing challenges related to devices, sensors, and systems have evidenced even more complexities (Maksimovic and Vujovic, 2017). In case of sensors, devices, or systems failure, a patient's life can be put at risk, jeopardizing a person's vital health care.

2.3 COMPUTATIONAL RESOURCES

Computational resources can process and execute operations and carry out tasks. These resources are related to hardware, software, computer networks, data storage, and security. These computational resources are available in computing systems (Chang et al., 2020).

Computational resources are essential for IoT and e-health. They provide the interface for devices and sensors to communicate with systems, allowing data collection and processing of what is happening in an intelligent environment. Examples of what computational resources can help are related to edge, fog, and cloud processing. They allow data collection close to the generated source and the cloud infrastructure needed to scale the number of devices and integrated sensors, ultimately storing the data for stakeholders. This stored data can still be processed and used for decision-making. Computational resources provide the computational structure for these processes (Ortiz et al., 2022).

Edge, fog, and cloud computing are layers of technologies that support the development of IoT applications, responsible for the necessary computational resources (Laroui et al., 2021). The edge shows the location of all the sensors and devices with a large amount of data to process. Fog receives data already processed by the edge and on a smaller scale. The cloud receives this data and stores the different applications (Malliga, Kogilavani and Nadhini, 2021).

2.3.1 Cloud computing

Cloud computing is defined as a set of computer network services that provide scalability, and quality of service (QoS), which can be customized to the interests of people, companies, and public bodies, offering infrastructure of computing resources on demand to which can be accessed in a simple and pervasive way (Wang et al., 2010).

IoT and cloud computing have been linked to allow applications to be developed

in different areas of interest, such as smart cities. Cloud computing offers the IoT the necessary infrastructure for storing, reading, processing, and using data (Park, 2018). Other benefits of cloud computing for IoT are related to resource scalability, remote infrastructure management, and aspects related to platform security, in addition to infrastructure cost controls (Aazam et al., 2014).

Cloud computing is also related to developing IoT applications in intelligent environments. Liu and Xiao (2021) present a solution for an IoT application for smart living that uses essential characteristics, such as processing, storage, and collection of data, as well as availability, to guarantee the correct functioning of the application. Thus, this is an example of how cloud computing enables the development of IoT and smart living applications.

2.3.2 Fog computing

Fog Computing is proposed based on dealing with low latency, greater bandwidth usage efficiency, and allowing data processing close to the origin of their generation. In this way, fog is defined as a technology that must provide data movement between the cloud and the edge to continuously meet the quality of service demands of applications (Berbach et al., 2018).

IoT devices and sensors rely on low latency, efficient bandwidth usage, and data processing, collection, and storage. To deal with these challenges, Fog Computing presents these characteristics in its definition (Yi, Li and Li, 2015). Like cloud computing, Smart Living and IoT applications have also been proposed. Vora et al. (2017) use Fog Computing to help monitor patients through application-generated data.

2.3.3 Edge computing

Like Fog computing, edge computing aims to promote data processing closer to the generation source. In edge computing, the objective is for computational processing capacity to be carried out at the edge, using the resources of network devices and data generators (Varghese et al., 2016).

Using edge computing with IoT allows for reduced latency, rapid responses to devices and sensors, and data processing close to the source of generation (Li, Ota and Dong, 2018). These facilities allow the creation of applications for IoT and smart living. In Rajavel et al. (2022), the authors present the creation of a video monitoring system for an intelligent healthcare environment using edge computing technology. This approach allowed the construction of the application through data processing close to the source and provided a low response time, given the degree of complexity involved in monitoring patients.

2.4 CONTEMPORARY SOFTWARE ARCHITECT

In addition to challenges related to IoT and intelligent environments, software architectural approaches have been used to mitigate and assist in developing efficient solutions that use cloud, fog, and edge (Razzaq, 2020). Contemporary software engineering needs to deal with the challenges brought by IoT and guarantee the quality of application development.

Given the challenges of developing systems for intelligent environments, such as availability, low latency, processing, collection, and data storage, modern software engineering seeks to support technology teams and the research area that delivers software (Motta, De Oliveira and Travassos , 2018). Gubbi et al. (2013) show that one of the ways to support software development is through computational architectures. These architectures are intrinsic to all software, but they establish a set of characteristics that software must have to deal with most of the challenges and meet the interests of its end users. The authors also state that architectures must ensure that it is possible to capture, process, and display data on a large scale, integrate and interoperability between devices, sensors, and systems, and have scalability and flexibility to enter new devices. Furthermore, data security and privacy are topics of interest for these systems.

2.4.1 Self-adaptive software architect

Intelligent IoT environments are changeable, where unplanned situations occur, generating the need for responses to these events. This changeable nature incurs difficulties that must be dealt with by software architectures that use IoT devices and sensors (Muccini et al., 2018).

Self-adaptive software architecture solutions can be helpful for environments that have a changing nature and need different software configurations. Garces et al. (2020) present the goals of self-adaptive systems as related to automatically allowing the software to change its settings to meet a demand from the environment. These demands, for example, may be related to the needs of the environment, users, and even resilience and fault tolerance.

According to Garces et al. (2020), self-adaptive architectures must comprise management and managed systems. The management system is responsible for adapting the managed system to the momentary demands of the software, for example, switching between a managed system module capable of doing more data collection and another capable of doing more data processing. The management system follows defined and mapped metrics to adapt the managed system to different configurations. These metrics may be related to response time, number of requests received, and number of failures. Managed systems are the systems that must handle the end use of the application. For example, an e-health application that handles patient medication and provides prompts

about when to take medication. This patient does not use the management system. This patient uses the managed system that displays the medication times.

Weyns et al. (2020) define a self-adaptive system as dealing with changes and uncertainties in the environment, the system itself, and the objectives defined for them. In addition, these systems must also interact with the environment and have a feedback loop to collect what is happening with the system. Finally, the authors state that self-adaptive systems have emerged as a fundamental element of modern software engineering.

Alfonso et al. (2021) present the topics of most significant interest for self-adaptive IoT architectures. According to the authors, monitoring, visualization of scenarios, software delivery, and machine learning are relevant topics that help guarantee the objectives and quality of these architectures. In addition, the authors also point out how computational resources, such as cloud, edge, and fog, are essential elements.

2.5 ARTIFICIAL INTELLIGENCE

Artificial intelligence is related to the ability of a system or machine, be it a computer, to perform tasks that require human intelligence (Wang, 2019). Acts of intelligence can be related to learning, problem-solving, perception, and acts of reasoning. Such acts are related to the capabilities of algorithms developed to support intelligence development (Kok et al., 2009).

Artificial intelligence in IoT applications is becoming essential (Amershi et al., 2019). Patel et al. (2023) present areas where artificial intelligence can help. Some examples are data processing, predicting patterns, and warning about situations that may happen based on machine learning techniques. Some benefits of artificial intelligence with IoT are related to assisting in decision-making, allowing new automation, personalization, and recommendation based on data. Finally, Artificial Intelligence can use devices and sensors to understand the intelligent environment better.

The use of artificial intelligence in intelligent environments has expanded in recent years. Applications in education, energy, agriculture, health, politics, finance, mobility, sustainability, and IoT are just a few examples (Allam and Dhunny, 2019). Gomes et al. (2023) use artificial intelligence in an intelligent agriculture context, allowing the authors to develop an application that supports physical space management and animal production and predicts diseases through data generated through environmental monitoring. Neves et al. (2020) present using artificial intelligence to identify risk cases in healthcare environments. To do this, the authors used data generated by sensors and devices, processed this data, and used artificial intelligence to make predictions about risk identification.

Given the use of artificial intelligence techniques in several areas, it is possible to identify how some AI approaches have supported the development of IoT applications

in smart environments, such as health and medical assistance. This support provides higher-quality applications that positively impact the lives of the population

2.6 FINAL CONSIDERATIONS OF THE CHAPTER

This chapter presented essential concepts related to the understanding of work, including the Internet of Things and software engineering, smart living, e-health, computational resources, edge computing, fog computing, cloud computing, contemporary software architecture, and artificial intelligence. These concepts portray the state of the art in the theoretical framework topics of this work.

3 METHODOLOGY

This chapter presents a systematic mapping of the literature on the research topic, the related works, and an exploratory study. The systematic mapping was conducted to support our research conjectures and identify related works in the literature. The exploratory study reinforces our conjectures and presents the challenges and obstacles encountered in the area of interest.

3.1 SYSTEMATIC MAPPING

Through the current systematic mapping, the aim is to identify and list IoT architectures for intelligent e-health environments, focusing on using computational resources and artificial intelligence. Therefore, it will be possible to present which quality attributes have been considered in developing IoT architectures, alongside how computational resources technologies and artificial intelligence techniques are used.

Given the current scenario of IoT, e-health, intelligent environments, computational resources, artificial intelligence, and software engineering, this systematic mapping addresses the following research question: **"How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?"**. To answer the research question, a systematic literature mapping was conducted to understand the current state of the art and to present how IoT, computational resources, and e-health are connected. Finally, the mapping results are presented in this work.

To conduct the systematic mapping of literature, based on the research question some research questions were derived, they are:

RQ1: What IoT architectures focus on computational resources in intelligent e-health environments? This question seeks to extract which computational architectures have been used regarding computational resources, presenting the applications of edge computing, fog computing, and cloud computing.

RQ2: What quality attributes are essential for IoT architectures in intelligent e-health environments? This question presents which quality attributes have been considered for developing architectures.

RQ3: What computational resources have been applied to IoT architectures in intelligent e-health environments? This question presents how edge computing, fog computing, and cloud computing have been used in architectures.

RQ4: How have Artificial Intelligence and computational resources been used in intelligent e-health environments? This question seeks to present whether artificial intelligence is used in architecture and how it is applied.

The current work followed the systematic literature mapping methodology proposed

by Kitchenham and Charters (2007). This methodology defines three main activities that should be followed: planning, conducting, and reporting. According to Kitchenham and Charters (2007), the planning phase is essential for reviewing the proposed research questions and defining a protocol for the systematic mapping process. The conduct phase is responsible for defining and executing the search process, selecting the related studies, enabling a quality assessment, and, finally, data extraction. Finally, the reporting phase seeks, in an organized manner and aligned with the execution protocol, to group and demonstrate the results found in the literature on the subject to be researched.

Parsifal[1] software was used to support the execution of all phases of systematic mapping.

3.1.1 Planning

Based on the derived research questions, we used the PICOC (Population, Intervention, Comparison, Outcome, and Context) method to define the scope of the work and the terms used in the search string(Kitchenham and Charters, 2007). The PICOC can be viewed in Table 1.

Table 1 – PICOC method

Population	Works that deal with computational resources, IoT, and e-health
Intervention	Computational resources
Comparison	Not applicable
Outcome	Software architecture
Context	IoT, Cloud Computing, Fog Computing, Edge Computing, Artificial Intelligence, Computational resources, E-health

Source: Elaborated by the author (2024).

Through PICOC, the work's keywords and synonyms were defined. Two specialists in the area were consulted to validate the PICOC. This validation occurred through discussions and evaluations regarding adherence to the research topic in Software Engineering and Computational Resources. Finally, the following search string was obtained:

("Cloud Computing" OR "Cloud" OR "Edge" OR "Edge Computing" OR "Fog" OR "Fog Computing") AND ("Computational resources") AND ("Internet of things" OR "IoT") AND ("Software architecture") AND ("Artificial intelligence") AND ("e-health")

After defining the search string, one Software Engineering specialist and one Computational Resources specialist were consulted to validate the words and their synonyms. Discussions and validations were made with the specialists, and preliminary database searches were conducted to verify the research topic's adherence to the words. The work

[1] https://parsif.al/

of Mahmud, Koch and Buyya (2018) was used as a control article to calibrate the search string during consultations with experts in the area. To calibrate the search string while modifying it, the control study guaranteed that we would constantly be identifying articles related to the terms used in the search string. Furthermore, this approach also allowed the identification of synonyms and other keywords (Budgen et al., 2008).

3.1.2 Conducting

3.1.2.1 Search process

After defining the search methodology, the protocol, the definitions of the research questions, and the search string, academic work bases that process search strings in an automated manner were mapped. In addition to this requirement, the work base must be accessible through the authors' institutional access. With this, we arrived at the result of three available databases: the ACM Digital Library[2], Scopus[3], and Springer Link[4].

Once in possession of the databases, the search string was executed during the month of November 2023, and works were found. We found 49 works in ACM Digital Library, 52 in Scopus, and 381 in Springer Link. Table 2 presents the quantity found per base.

Table 2 – Academic Work Base and Number of Works found

Works base	**Number of works**
ACM Digital Library	49
Scopus	52
Springer Link	381
Total	**482**

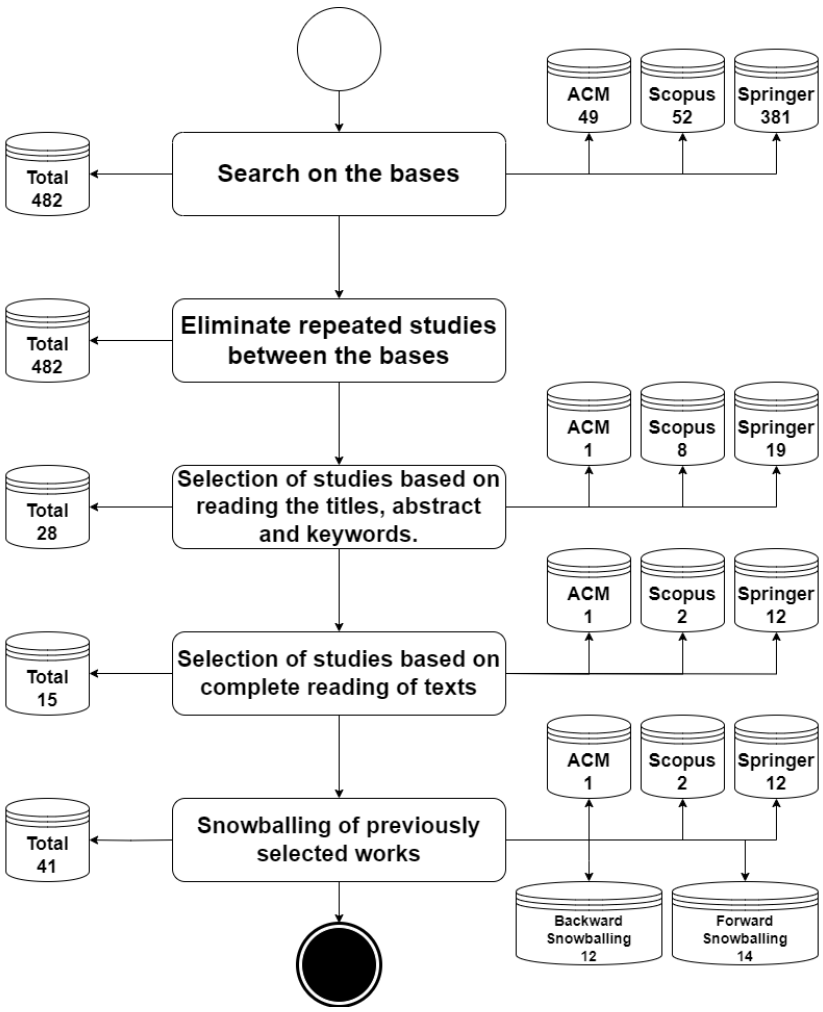
Source: Elaborated by the author (2024).

3.1.2.2 Study selection

The step-by-step presented in Figure 1 was followed for the study selection.

[2] https://dl.acm.org/
[3] https://www.scopus.com/home.uri
[4] https://link.springer.com/

Figure 1 - Study selection process



Source: Elaborated by the author (2024).

Step 1 – Eliminate repeated studies between the bases: When evaluating the studies, we did not find repeated studies. At the end of this stage, all 482 articles found in the databases were maintained.

Step 2 - Selection of studies based on reading the titles, abstract, and Keywords: This step classified the works as accepted or rejected. This classification is essential to identify which works relate to the area of interest and provide an overview.

The inclusion criteria ensure that works of interest can be located and identified for reading (Sanfilippo et al., 2020). Reading the title and abstract allowed us to identify whether the work relates to the research questions. Other fundamental conditions are related to how the work was reviewed, whether it was carried out by peers or not. Another premise is that the works are written in English. Finally, with the research group's access via the educational institution we had, the work must be accessible.

The exclusion criteria allow works that do not adhere to the research questions to be excluded from the investigation by reading the title and abstract (Meline, 2006). Furthermore, it allows studies not in the computing field to be filtered. Another exclusion criterion adopted was not considering books, sections, or chapters of books. Given the institutional access to the databases, it was also excluded if it was impossible to access the work. Another exclusion criterion used was only to consider works written from 2017 onwards. This choice was because the number of works found in the literature became relevant after this period. Before this period, few works were found with the search string. Finally, other secondary studies, such as systematic mappings, systematic reviews, and surveys, were also removed.

As inclusion criteria, we used:

IC1: Title and abstract are related to the research questions; **IC2:** The work was peer-reviewed; **IC3:** The work was written in English; **IC4:** The work is available for access. The exclusion criteria allow works that do not adhere to the research questions to be excluded from the investigation by reading the title and abstract.

As exclusion criteria, we used:

EC1: The work does not adhere to the research questions; **EC2:** Studies that are not in the Computer Science area; **EC3:** Books, sections, or chapters of books; **EC4:** Unable to access the study; **EC5:** Studies published before 2017; **EC6**: Other secondary studies.

At the end of this stage, 28 works were accepted for the next phase.

Step 3 - Selection of studies based on complete reading of texts: Given the complete reading of the 28 articles selected by the inclusion and exclusion criteria, a new filter was applied to these works. This complete reading explored aspects related to the research questions in the papers. If a work failed to address any essential terms for the research, such as IoT, e-health, software architecture, and some computational resources, the work was removed from the results. When performing the filter through the complete reading of the works, 13 of the 28 were removed, leaving 15 works after executing this phase.

Step 4 – Snowballing of previously selected works: The backward snowballing technique used by the researchers evaluated the references used in the 15 articles selected up to this stage. The forward snowballing technique evaluated the citations received from the 15 articles selected. To assess the references, a reading of the titles of the works was carried out to identify similarities with the research questions. If the title matched the research questions, the abstract was read. After reading the abstract, if the work was still adherent, a complete reading was carried out applying the filters that If a work failed to address any essential terms for the research, such as IoT, e-health, software architecture, and some computational resources, the work was removed from the results. Thus, at the

end of the snowballing phase, 26 papers were identified. Forty-one works of interest for systematic mapping were totaled, 15 before and 26 after the snowballing. The works found on snowballing were not on the list of rejected works.

3.1.2.3 Quality assessment

Each work found can contribute to the answers to the research questions differently. Therefore, evaluating the quality of these selected final studies is necessary. To achieve this, we proposed eight questions to estimate the quality of the contribution of each of the works. This approach to assessing the quality of studies was proposed by Kitchenham and Charters (2007).

For the constructed questions, each of them accepted three types of answers. The first was adherent, adding 8 points; partially adherent, totaling 4 points; non-adherent, not adding points. To guarantee the quality of the work, they should score at least 28 points between questions.

The questions related to quality are: (**QA1**) Does the study present an architecture? (**QA2**) Does the study present the use of quality attributes? (**QA3**) Does the study present the use of cloud computing? (**QA4**) Does the study present the use of fog computing? (**QA5**) Does the study present the use of edge computing? (**QA6**) Does the study present the use of artificial intelligence in architecture or solutions? (**QA7**) Does the study propose an application?

When conducting the quality assessment, the lowest score was 28, and the highest was 48 points. The mean value was 38, and the median was 40. No other studies were excluded at this time.

3.1.2.4 Data extraction

We carried out the data extraction process from the selected final works to answer the established research questions. For this extraction, a set of questions was formed, and their answers will help answer the research questions and understand the current state of the art.

The questions for data extraction were: **(1)** What are the characteristics of the architectures used? **(2)** What are the existing concerns with the architectures? **(3)** What are the quality attributes used? **(4)** What concerns exist with quality attributes? **(5)** Have cloud, fog, and edge been used in architecture? **(6)** Are cloud, fog, and edge used in the same architectural proposals? **(7)** Is there any artificial intelligence approach to use computational resources? **(8)** What AI approaches are used in architecture? **(9)** What are the areas of applications developed?

3.1.3 Reporting

This section condenses the results obtained by answering the research questions proposed in the systematic mapping.

3.1.3.1 What IoT architectures focus on computational resources in intelligent e-health environments?

The first objective related to the research questions is identifying architectures being employed in the architectures. For this reason, the term architecture was one of the words chosen for the search string. The inclusion criteria also considered reading the abstract and the title, which must contain the word architecture. Furthermore, reading the complete works still focused on identifying which architectural approaches were used. Finally, the questions related to the quality of the work also sought to understand the use of architectures in intelligent e-health environments.

Given the structure of the work and the reasons previously presented, all the chosen works propose an architecture or framework for developing IoT applications in e-health. Furthermore, a focus of interest in the work is to understand which computational resources are being used in these architectures.

The computational architectures and frameworks presented in the works were characterized by division into at least three different layers (modules). The first layer had sensors and IoT devices in monitored intelligent e-health environments. These environments include patients, healthcare professionals, hospital environments, and homes related to IoT and intelligent environments. The proposed second layer commonly performs the processing, loading, and storing data generated in the first layer. In jobs that use cloud computing, this processing occurs in the cloud. Data processing begins in these computational resources in works that use fog and edge computing. The third layer presents developed applications that consume the stored data, generating platforms for monitoring intelligent spaces and allowing data analysis for decision-making, which can result in changes in the intelligent environment. Some examples of this layered architecture are the works of Kumar et al. (2023), Said et al. (2020), Arual et al. (2021), Verma and Sood (2018), Ghorbani et al. (2023) and Tuli et al. (2020).

The variations seen in the selected works are related to introducing other layers in the architecture. These layers can involve distributed processing and storage of data, as seen in Gia et al. (2019), involve new Edge and Fog layers, as seen in Mahmud, Koch and Buyya (2018), and may be related to security aspects Rahman et al. (2020)

Through this analysis, architectures used in the selected works commonly use a layer for data generation, a second layer for processing, loading, and storage, and a layer for developing applications responsible for the display.

Among the 41 selected works, all used the computational cloud structure. Seventeen of the works made use of the fog computing resource. Nineteen of the works made use of edge computing. Through the visibility of this data, it is concluded that the cloud approach is expected to architectures in the area. Fog and edge have also been used.

Another conclusion is that Cloud, Fog, and Edge are computational resources commonly used in these architectures, causing variations between the number of layers; given that where Fog and Edge are used, new layers are found in the applications so that data processing occurs closer to the edge, reducing application response time.

Therefore, after analyzing the selected works, the architectures that use computational resources in e-health environments are highly related to Cloud, Edge, and Fog approaches. Given the choice of those responsible for the architectures, the number of layers to employ the processing techniques varies. In summary, among the works found, all IoT architectures in e-health environments collect, process, and store data and use at least three layers. Depending on the criticality, some applications have alerts about unexpected situations, such as patient problems and interruptions in sending signals. Applications for data visualization and decision-making support are commonly found in these architectures.

3.1.3.2 What quality attributes are essential for IoT architectures in intelligent e-health environments?

The second objective through systematic mapping is to understand which quality attributes are being addressed in the architectural proposals. Quality attributes were extracted from the works using data extraction questions. Thus, those considered were performance, efficiency, interoperability, usability, scalability, security, maintainability, availability, and reliability.

The most cited quality attribute among the works was performance; 36 of 41 works deal in some way with performance. In order, security was mentioned in 24 papers, efficiency in 23, reliability in 18, scalability in 14, availability in 12, interoperability in 10, usability in 5, and the last mentioned was maintenance capacity; only 4 of 41 works deal with this quality attribute.

Thus, through analysis of the works, it was possible to observe that the quality attributes presented are the most cited and should be considered when proposing architectures for IoT applications in intelligent e-health environments.

When analyzing the works, another concern regarding quality attributes was related to the quality of the service. In 10 of the 41 selected works, the authors are concerned with the quality of the service, whether in the final application or the architecture while it was being developed. Topics related to the delivery of applications that meet the needs of devices, sensors, and the target audience are highlighted by the authors as an object of concern.

3.1.3.3 What computational resources have been applied to IoT architectures in intelligent e-health environments?

Through the analysis of the selected works and the answer to the first research question presented, concepts and technologies of Cloud Computing, Fog Computing, and Edge Computing are being applied in intelligent e-health environments, sensors, and IoT devices. In all selected works, Cloud Computing was used. Fog Computing was used in 23 of the 41 selected works. Finally, Edge Computing was used in 28 of the 41 selected works.

The investigation of related work demonstrated that the main advantage of using these technologies is their processing capacity, whether closer to the origin with Fog and Edge or in the cloud. A second advantage is data collection, which is carried out after sending data from sensors and devices. A third advantage is associated with the large storage capacity and cost of cloud storage. A fourth advantage can be observed in the cloud computational resource interfaces for building and making applications available. A fifth advantage observed in the studies is related to maintenance costs, which are controllable and understandable through the configuration of technologies. The sixth advantage refers to the portability of these resources to other cloud, fog, and edge structures, which are presented.

Furthermore, the analysis of the works allows us to see that the use of computational resources is a reality in the applications of intelligent e-health environments. Its advantages are presented, and the authors defend it as a facilitator in software architecture and a means for creating robust applications that meet the demands of healthcare companies, public bodies, patients, and healthcare professionals.

3.1.3.4 How have Artificial Intelligence and computational resources been used in intelligent e-health environments?

Only 11 of the 41 selected works used some artificial intelligence in software architectures for intelligent IoT environments in the context of e-health. When reading the work, it was impossible to directly identify a relationship between computational resources and the use of artificial intelligence in the expected context.

Muthu et al. (2020), Verma and Sood (2018), Kaur, Kumar and Kumar (2019), Verma, Agarwal and Gupta (2022), and Tuli et al. (2020) present the use of artificial intelligence based on data generated by sensors and devices, predict what types of diseases a person may be suffering or being affected by. Once again, AI is used in the final application, not alongside computational resources.

Said et al. (2020) present the use of artificial intelligence to identify attacks on the computational infrastructure used by the architecture proposed. This technique seeks to create defense mechanisms both in the computational part and to understand what can be done in the physical part of the hospital.

Gupta, Al-Naime and Al-Anbuky (2021) present the proposed architecture where artificial intelligence could be used. However, they do not present the use made in their work. AI is related to the application's data processing, not computational resources.

Mahmud, Koch and Buyya (2018) do not use AI but propose that it can be used in the Fog framework to predict when it will be necessary to perform data processing and prepare for processing. This approach links computational resources and the intelligent e-health environment for which the architecture was proposed.

Talaat (2022) proposes using artificial intelligence to predict real-time resource allocation. The work focuses on the algorithms that can support this prediction. Therefore, the author aims to verify which artificial intelligence and machine learning algorithms are best suited to reduce latency, improve service quality metrics, reduce response time, and ensure energy efficiency. The author focuses on demonstrating how algorithms work and not necessarily linking them to an architecture. Finally, the authors test the algorithms in a patient environment.

Dai et al. (2019) propose using AI to help predict and measure the energy expenditure that some aspects of the architecture can use. In this way, the authors hope there will be less energy expenditure.

Dhasarathan et al. (2023) propose using deep learning techniques to help predict privacy metrics in e-health environments. Deep learning techniques are applied at the application level rather than at the architectural level.

Upon completing the analysis of the selected works, we saw that the use of computational resources and artificial intelligence are not intrinsically linked in the selected works. Only in two approaches was it possible to identify the use of artificial intelligence to assist computational resources, proposing AI to predict when data processing would be used in the architecture.

3.1.4 Discussion

After carrying out the entire systematic mapping process and answering the research questions, architectural proposals were identified for developing IoT applications in intelligent e-health environments. The proposed architectures and their use demonstrate the impact of a good solution for creating quality and robust applications. Computational architectures are one of the fundamental factors for developing these applications with quality.

The second important factor is quality attributes. After reading and exploring the selected works, it was possible to observe that quality attributes are fundamental parts of the proposed architectures, so the development of applications is possible. Attributes such as performance, security, and efficiency are cited and highlighted as directly impacting reasonable solutions.

The third related factor is Cloud, Fog, and Edge. All the works presented use these resources, and given the systematic mapping scenario, all architectures benefit from using computational resources. In the works read, the authors present several advantages, such as loading, processing, and displaying large amounts of data as one of the main benefits.

Artificial intelligence is being used in some applications, such as predicting situations with patients, diseases, and scenarios. However, AI is not being applied in architecture, how quality attributes and computational resources behave.

Using computational resources in conjunction with artificial intelligence still needs to be explored. The only note was related to using fog to predict moments of data processing stress. Exploring the use of computational resources can bring benefits to architectures. Benefits can be helpful in the use of computational resources, adaptation to the needs of the intelligent environment that are inserted, and scalability according to existing demands. This section condenses the results obtained by answering the research questions proposed in the systematic mapping.

All data relating to the systematic mapping can be seen in Zenodo[5].

3.1.5 Threats to validity

As threats to the validity of the present study, they are related to:

Study selection: Given the limited access to some databases, studies that could have been useful in the systematic mapping analysis may have yet to be explored. Alternative search strategies can be adopted to address this threat, such as contacting authors and requesting their work. Another possibility is to increase the databases of work to be researched and accessed via an educational institution.

Unavailable studies: Some works could not be accessed due to non-availability or limited access to the databases by researchers. To address this threat, authors can be notified to request access to their work.

Exclusion criteria: Works published in books were disregarded, as they were classified as secondary works. This filter can influence the selection of jobs. To address this threat, exclusion criteria can be re-evaluated. Mainly, the part that concerns book chapters assesses whether these chapters were peer-reviewed or resulted from some event in which the work had to be peer-reviewed before publication.

Study approach: Although the protocol was faithfully followed, some work may have been unduly excluded, given the amount of work to be filtered and the researchers' critical reading. To address this threat, reviewing works by two or more persons can help

[5] https://zenodo.org/records/10971320?token=eyJhbGciOiJIUzUxMiJ9.eyJpZCI6ImE0N2VkZGMxLWI5NmMtNGVjNS1iZjk3LWQ5YTk1YWZmMTAzOSIsImRhdGEiOnt9LCJyYW5kb20iOiI2MjEyMzljNmNhYWYwYTgwMWQwZmZiOGFhYTNmZWQzZSJ9.u772ml6RAQs8QQ2TznsbLmYavIl7_SBoeBfTYrGgXF3IZD6Q_jBGGkN6VZ5-jamaflNwQARFwg2A1TZ56feU8A

identify works that one person has mistakenly filtered during the execution process. One mitigation strategy that can be adopted is double-checking the same work.

3.2 RELATED WORKS

This subsection presents related works with the highest scores obtained in the quality assessment of systematic mapping. In total, seven works scored 48 out of a possible 70.

Rahmani et al. (2018) present a gateway to communicate between sensors and IoT devices with the computational resources available in the fog and cloud computing structure. For developing this gateway, the authors present the needs related to IoT devices and sensors, such as data storage, low response time, and quick decision-making. The authors also present the difficulties related to developing gateways of this type in e-health environments.

To solve the problem, the study presents an architecture full of sensors and IoT devices that can be used in hospitals or smart homes. This architecture proposes that sensors and devices generate data about the environment, using the gateway as an intermediate target between computational resources for communication and ultimately enabling data processing, storage, and use. The authors list several quality attributes explored, such as interoperability, security, performance, and availability. The authors also present efficiency aspects related to the electrical energy expenditure of sensors and IoT devices, stating that gateways allow processing in layers closer to the origin, saving energy in sizeable computational resource structures.

Although they point out the use of computational resources and how the architecture benefits from using the gateway, the authors do not address how computational resources can be used more efficiently as a significant focus of the work. The authors point out that resources can benefit from architecture. However, they do not present the real benefits obtained. Furthermore, there is no use of artificial intelligence in this approach.

Ben Hassen, Dghais and Hamdi (2019) discuss a computational architecture that uses fog and cloud computing to monitor older people. The authors justify the construction of architecture given the growing number of older people worldwide and to guarantee the quality of life for people at this stage of life.

For the development of the proposed architecture, the authors present an architecture based on four main agents. The first is related to people's devices and sensors present in an intelligent environment, the second to the fog computing structure, and the third to the cloud computing structure and put an end to those interested in monitoring patients who use applications. The authors address aspects related to trust, performance, and especially usability, given the elderly population that may be affected by the application.

Although it is an architecture that deals with data generation, processing, and storage, the authors do not present how computational resources can be used more efficiently. The authors present computational resources as enablers of the solution and do not focus on them. Therefore, the work does not address improving computational resources, only opportunities with fog computing. Finally, aspects related to artificial intelligence are not covered.

Kumar et al. (2023) present a framework for heart attack prediction using the RF-LRG algorithm. To propose the framework, the authors present an architecture that uses computational resources, such as edge and cloud computing. According to the work, the architecture was proposed to assist in the population's daily life, given the change in the lives of people who began to have habits that led to a higher incidence of heart problems in recent years.

A framework was proposed to provide safety and a better quality of life for the population by predicting heart attacks. This framework encompasses user aspects, data generation through devices and sensors used, and the processing and storage of this data in edge and cloud computing structures. This framework uses predictive artificial intelligence algorithms to support the solution. The work focuses mainly on availability, security, and scalability.

Although it proposes a solution that uses artificial intelligence, it is applied to support the monitoring of people rather than dealing with computational resources. In this way, the application uses computational resources, and how they provide the infrastructure for its operation. However, it does not focus on how they can help develop the solution.

Mahmud, Koch and Buyya (2018) present the main difficulties in using IoT in healthcare as aspects related to latency problems, data loading, the diversity of user expectations, and the heterogeneity of each application. The work addresses how interoperability between cloud and fog computing resource approaches can assist these applications in dealing with and addressing these problems.

To deal with the problems presented, the authors present an architecture with IoT devices and sensors, computational resource managers, and cloud structures to enable aspects related to the hardware infrastructure necessary for applications. In this way, the authors present the role of the cloud and fog approach in dealing with aspects related to quality attributes, such as performance, scalability, interoperability, and security.

The authors do not use artificial intelligence. However, the authors present the use of AI in the proposed fog structure as much as possible. The use proposed by the authors is related to predicting when it will be necessary and at what time of day to carry out large-scale data processing to prepare the computational infrastructure better. Although the authors propose, they do not develop or experiment with the proposed solution. For this reason, it is impossible to assess the quality of the proposal for using AI.

Rodrigues et al. (2021) present a computational architecture used in an intelligent hospital focusing on availability and performance. The architecture proposal derives from the need for IoT applications with low latency and high resource availability. According to the authors, the application service quality is guaranteed by proposing reasonable solutions to these issues.

This way, the work presents an architecture with stakeholders, such as sensors and IoT devices, network devices, humans, and local and remote servers. The authors present how to synchronize all services and argue how to guarantee high availability and the necessary performance. With this, the hospital would present the necessary quality for the application.

Although the presented solution deals with aspects related to computing resources, such as the cloud and quality attributes, the focus was on something other than using computing resources but on providing a solution that serves intelligent hospitals. Aspects related to computational resources are not discussed nor presented in detail. Finally, aspects related to artificial intelligence are also not covered.

Talaat (2022) addresses how the Internet of Things has expanded recently. This growth also led to the expansion of the Fog Computing approach and its use in various applications related to digital transformation. One of the problems the author highlighted is the great need for data processing by IoT applications in different spaces of intelligent environments. To deal with this problem, the author presents the use of Fog Computing as one of the solutions.

In addition to using Fog Computing, the work presents a methodology called the Effective prediction and resource allocation method. The author points out how managing computational resources is a complex task. To mitigate this arduous task, the author proposes an architecture divided into three modules, which perform data pre-processing, one module to allocate RAM and another to predict how much computational resources will be needed. The author uses artificial intelligence algorithms to predict how many resources are needed.

Although the author presents it as a practical approach, in addition to helping to reduce latency and improve the quality of service in e-health applications, the approach focuses only on the resources needed in Fog Computing. Furthermore, the authors focus on memory usage, CPU usage, disk space, and network latency. This way, the focus is not on cloud, fog, and edge.

Tuli et al. (2020) argue that many applications can be developed in different areas with the growth of cloud computing. However, the author's work focuses on the electronic health niche. The authors present the various challenges facing IoT, such as large data generation and low latency. They argue that to deal with this problem, there is a need for frameworks to deal with data processing close to the origin.

Faced with the challenges, the authors propose a new framework called HealthFog to generate data close to the source, focusing on predicting cardiac anomalies in patients. Given the characteristics of the problem, the authors need to reduce the application's response time. To this end, they propose using Fog Computing, which provides computational capacity and helps reduce latency.

Even though Fog Computing is highly effective in reducing latency and supporting the management of critical patients, the approach focuses only on using computational resources to support the application, not on how to best use them. Although it deals with aspects related to performance and other quality attributes, the authors' focus is not on optimizing the use of computational resources but rather on how to develop an application through their use.

3.3 FINAL REMARKS OF SYSTEMATIC MAPPING AND RELATED WORKS

The systematic mapping identified and analyzed different IoT architectural proposals in the e-health area and computational resources and their use. Quality attributes and using artificial intelligence were also the subject of study. Initially, 482 works were identified. After the entire conducting process, 41 works of interest related to the research topic were finally found.

With the results of the research questions, it was possible to observe how IoT architectures have been developed and used. Computational resources and their use in e-health architectures were also identified. In this way, it was possible to glimpse which quality attributes are used and how artificial intelligence is used. Finally, through systematic mapping, it was possible to advance knowledge about the research question, making it possible to identify opportunities and the basic requirements for these architectures in terms of computational resources and quality attributes.

Based on related work, it was possible to identify how computational resources are used in IoT and e-health. The use of cloud, fog, and edge is a common theme in related works. Such computational resources are used directly to process, load, and store large amounts of data, focusing on low response time and ensuring the correct functioning of applications. Although their use is well established, these computational resources have been directly available for IoT applications, sensors, and devices. The indiscriminate use of these resources can result in waste, resulting in more significant financial expenses and the non-use of available computing capacity. These problems can result in financial losses; excessive consumption of electrical energy; greater exposure of computing resources to privacy and security issues; infrastructure overload, in case of many parallel services required, increasing maintenance costs and decreasing service quality; risks of persistent attacks; resistance to the scalability of resources, as they are always available at high capacity.

Architectures and quality attributes have also been explored in related work. Performance, security, efficiency, reliability, and scalability are some quality attributes found. These quality attributes are related to the problems presented by the architectures in using computational resources. For example, the problem of infrastructure overload can degrade application performance if all services are using computational resources and are not being used by devices, sensors, and systems. A second example deals with the uninterrupted availability of computing resources, which can also cause problems involving greater exposure of applications, leading to security problems. The third example deals with electrical energy consumption, which can hinder the efficient use of computing resources. The fourth example deals with reliability, given that services can be affected by unavailability due to problems related to the performance or scalability of resources already using all available computing capacity. Finally, a fifth example deals with how resources can scale, given that they are constantly consumed 100% of the time. If there was management of who uses computing resources, other resources could benefit from the availability generated of resources not being consumed 100% of the time.

After conducting the systematic mapping and analyzing related works, with the knowledge obtained and based on the research question "How computational resources have been used in intelligent e-health environments that use IoT?", we establish the following conjecture. Suppose we propose a self-adaptive architecture for developing IoT applications in e-health. In this case, can we improve the use of computational resources by systems, supporting data collection from intelligent e-health environments and supporting the management of these spaces?

3.4 EXPLORATORY STUDY

In possession of the research question, systematic mapping, and conjecture, an exploratory study was conducted and is presented in this subsection. With the exploratory study, in addition to the difficulties and opportunities found in the literature, the aim was to observe other challenges that may arise during the development of IoT architecture environments and evaluate the use of computational resources. Furthermore, the objective was to validate and verify the challenges found in the literature in practice.

The exploratory study is related to one of our works, which can be seen in Do Nascimento et al. (2021). The exploratory study in this subsection is divided into four subsections: the scenario, the proposed solution, the evaluation of the solution, and the discussion based on the research question and conjecture.

3.4.1 Scenario

During the Covid-19 pandemic, all countries worldwide have had to deal with the same challenges. From a computing point of view, solutions, and proposals were made to

deal with the contamination, transmission, and spread of the disease worldwide. One of the main items of concern to be addressed was how the spread of the disease would occur in different locations, such as neighborhoods, cities, hospitals, and universities, among others.

With the expansion of Big Data and Digital Transformation in recent years, the amount of data has become increasingly more significant for society. One of the ways to support the decisions of public and private bodies regarding the spread of the disease was to use available data generated by sensors and IoT devices.

We presented an architecture for building IoT applications in pervasive environments. The architecture presented featured sensors, computational resources, and IoT devices in an intelligent environment at the Federal University of Juiz de Fora. The main objective of the architecture was to present disease contamination scenarios and support decision-making (Do Nascimento et al., 2021).

Based on the quality attributes found in the systematic mapping, the problem, and the established conjecture, the following non-functional requirements are established for the exploratory study architecture:

NFR01: The architecture must be capable of dealing with different models and databases (Interoperability and scalability).

NFR02: The architecture must be capable of processing and storing large volumes of data (Scalability and performance).

NFR03: Processed and collected data must be protected to avoid unauthorized access or leakage (Security).

NFR04: The architecture must be scalable to handle the increase in IoT devices and sensors. In addition to supporting the increase in the amount of data generated (Scalability and reliability).

NFR05: The solution must be extensible to allow the entry of new sensors and IoT devices (Scalability).

NFR06: The solution must be capable of using computational resources efficiently, providing their use only when necessary (Efficiency and performance).

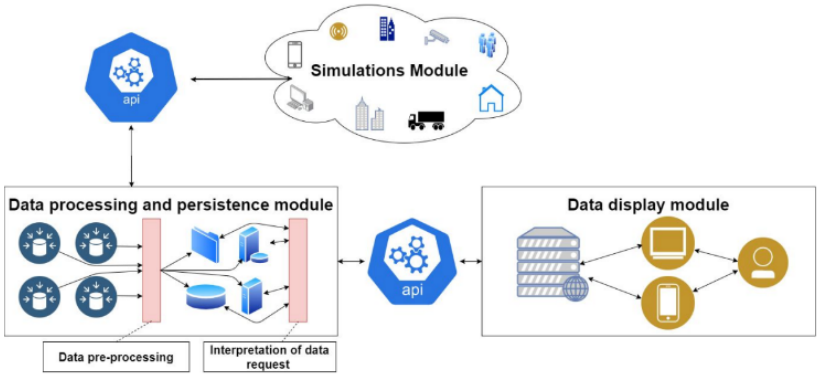
NFR07: The developed system must provide clear and intuitive views of what the data represents (Usability).

NFR08: The solution must offer ways of accessing data for interoperability with other applications (Interoperability).

3.4.2 Proposed solution

While the Covid-19 pandemic evolved, understanding the spread and contamination in different city locations was necessary for decision-making. Intelligent environments, at the time, already made use of IoT devices and sensors, which generated data about the movement of people in different settings. The architecture presented in Figure 2 was proposed to support decision-making based on the collection of generated data.

Figure 2 - First IoT architecture developed



Source: Elaborated by the author (Do Nascimento et al., 2021).

The presented architecture was divided into three modules, which communicated with each other through APIs. The proposed modules were:

Simulations module: Given the risks linked to testing the solution with people and in different physical spaces at the time, the first module proposed was a simulation. This module simulates data generation by different sensors and IoT devices at the Federal University of Juiz de Fora. The data generated was representative of the interaction and traffic of different people with the environment to support this generation. All generated data was sent to an API with a communication interface with the Data Processing and Persistence Module. All the data generated can be seen in the website[6].

Data processing and persistence module: This module had a data pre-processing layer, which performed initial filtering of the data received. Then, this data was processed according to the type of data storage required, such as database servers and file servers such as images and texts. The module also had a data request interpreter to access the stored data, checking whether the data could be available. Communication with this module was also done via API.

Data display module: The last module provided applications that support decision-making. These applications could be for computers and cell phones and would be

[6] `http://200.17.70.211:10081/simulacoes/lista`

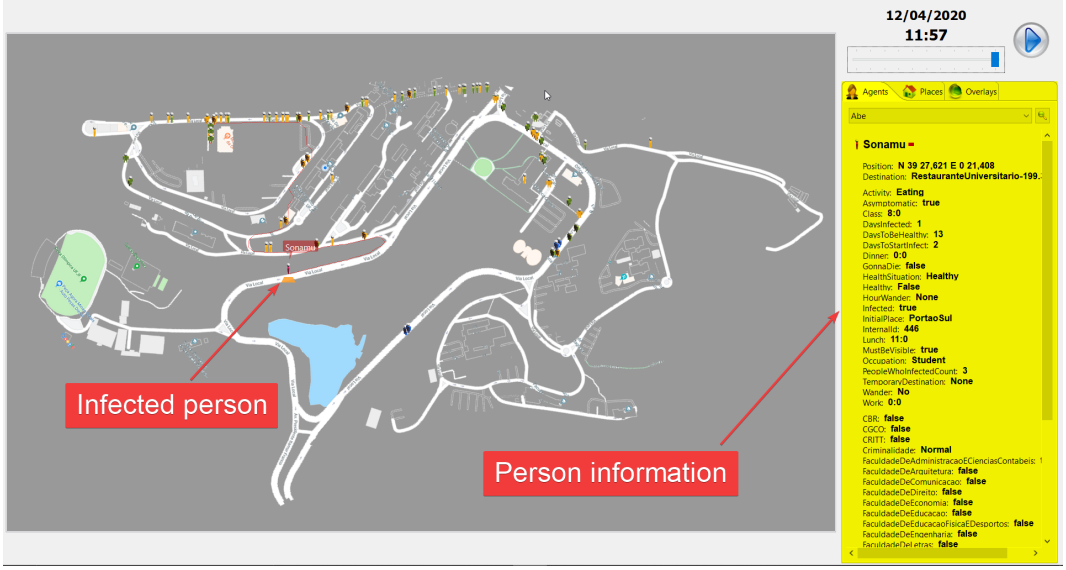
on servers. End users, such as decision-makers from health regulatory bodies, accessed this module to analyze data to make decisions.

3.4.3 Evaluation

The present evaluation analyzes how the proposed solution addressed the objective of present disease contamination scenarios and supports decision-making. Furthermore, it evaluates how non-functional requirements were used and identifies challenges and opportunities in line with the research question and the established conjecture. As a result of evaluating the exploratory study, opportunities and challenges were found and validated, as presented in the literature.

The architecture was developed using the context of the Federal University of Juiz de Fora to evaluate the proposed solution. Thus, the Siafu[7] simulator was used in the simulation module to build interactions between people and environments. The simulator created different contamination scenarios through the spread of the Covid-19 disease. These scenarios comprised people, IoT devices and sensors, physical spaces, and mapping of interactions on the university campus. Figure 3 presents a graphical view of how the simulation was performed.

Figure 3 - Siafu execution of Federal University of Juiz de Fora



Source: Elaborated by the author (Do Nascimento et al., 2021).

The NEC-SX-Aurora Tsubasa[8] supercomputer was used to execute the simulations, given the need to generate data for several days and complex interaction scenarios.

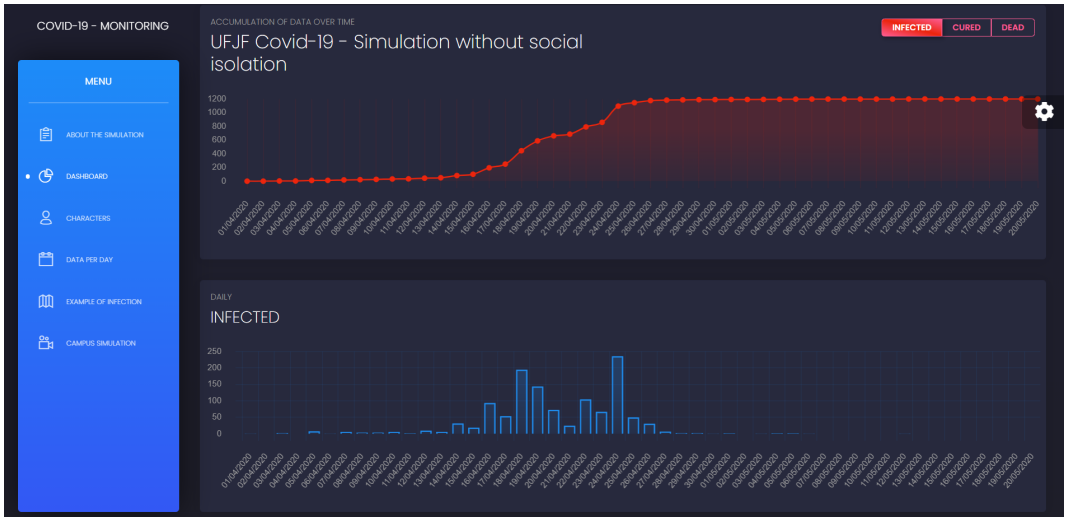
[7] https://github.com/miquelmartin/Siafu
[8] https://www.nec.com/en/global/solutions/hpc/sx/vector_engine.html

Scenarios composed of different people, sensors, devices, physical spaces, and interactions are characterized as complex interactions. In them, different actions happen every second by different people. For example, in the simulated scenarios executed, 20 thousand people interacted with the university campus daily. Each physical space at the university was mapped with presence and temperature sensors—each interaction with the simulated space needed to be analyzed using data generated by sensors and IoT devices. A sizeable computational capacity was necessary for this processing and the extensive data generation found in the NEC-SX-Aurora Tsubasa.

The objective of the developed API was to load, process, store, and list the data generated by sensors and IoT devices in the simulated scenarios. This API used only the computing cloud resource, which housed the database. The API was built on a monolith, with an API route for each device and sensor to process, store, and list the data. Therefore, a new API route must be created for each new sensor mapped. The computational resources were all consumed by this API and the database.

Decision-makers used the developed visualization tool to support the understanding of different scenarios. Built to visualize simulated scenarios, the application also used the cloud computing resource to communicate with the previous API. Figure 4 demonstrates the developed application.

Figure 4 - Dashboard application developed



Source: Elaborated by the author (Do Nascimento et al., 2021).

The developed architecture could connect to a database and a file server to deal with different models and databases. If necessary, the architecture could connect to other banks and servers.

The developed architecture was capable of processing and storing large volumes of

data. However, when there are many generations of data from IoT devices and sensors, especially at times of high movement on the university campus in the simulated scenario, the application response time increases, and in some moments, slowness in the processing and storage of data is noticed. Furthermore, using a single API concentrated all processing in a single computing resource environment, which did not distribute processing and increased the load on computing resources.

Given the expansion of IoT sensors and devices, the architecture also had difficulties in handling the large scale of data processing and loading, which led to more significant slowness problems.

Both for processing data and storing it, the architecture did not prove to be efficient, given the increase in response time and the appearance of slowness, which ended up causing efficiency problems.

To protect the data, only the API had access to the database. User access settings to the database through the API were created in order to guarantee the privacy and security of the processed data. Furthermore, if another application required data access, this was permitted through user and password authentication established as a database access policy.

Data visualization occurred clearly and supported decision-making. Although processing could suffer from slowness in processing and storing data, as the scenarios were simulated, this did not impede decision-making. Appendix A presents journalistic news related to the decision-making tool.

3.4.4 Discussion

The execution of the exploratory study allowed us to learn more about the research question and our conjecture. As presented in the evaluation, in summary, the exploratory study allowed us to learn about how IoT data generation dynamics occur in intelligent environments, how IoT devices and sensors behave, how computational resources are used, how architectures are developed for the area; communication of IoT sensors and devices with systems; data access; interoperability and scalability of devices and sensors; interoperability and scalability of computational resources; software development and validation challenges; complex interactions within these environments; complexities involving e-health environments.

At the end of the exploratory study, we created a list of necessary improvements to the proposed architecture:

- Improve the use of computational resources. Only one module in the proposed architecture concentrates all the data processing, listing, and updating in a single module. The same architecture module always uses all the necessary data. With

intensive data generation and application use, response times for requests were relatively high and increased memory and processing consumption. This problem showed that scalability, interoperability, and performance aspects needed to be privileged. An example to be presented is the high use of a presence sensor, which consumed all available computational resources during parts of a day. This problem meant that other devices and sensors needed more resources available for their use, failing to interoperate or scale. Modules such as visualization were also impacted, as the structure that handled the data was already under a high workload; it could not meet the needs of the visualization module.

- Distribution of processing load between different computational resources. Cloud, fog, and edge distribution approaches can filter, process, and store only data relevant to the application context to achieve this.

- Use of data processing techniques. Data processing algorithms can be used to reduce response time, increasing the efficiency of the architecture and processing only the most essential data.

- The use of simulated sensors and IoT devices could not faithfully portray the needs of these elements. Using simulation, complexities related to the characteristics of IoT devices and sensors were alleviated. In this way, all processing of all sensors occurred similarly. For example, a presence detector and temperature devices performed a data request similarly.

- Access to available data layer without access policies. The data processing, listing, and updating module was integrated with the storage module without a clear access policy. It meant that different applications could consume the same data without having data usage traceability.

- Monitoring the architecture allows for adapting computational resources according to the environment's needs. Computing resources were available 100% of the time, which can lead to waste, resulting in more significant financial expenses and the non-use of available computing capacity. These problems can result in financial losses; excessive consumption of electrical energy; greater exposure of computing resources to privacy and security issues; infrastructure overload, in case of many parallel services required, increasing maintenance costs and decreasing service quality; risks of persistent attacks; resistance to the scalability of resources, as they are always available at high capacity.

3.5 FINAL REMARKS OF THE CHAPTER

The systematic mapping of literature allowed the identification, classification, and analysis of the use of computational resources in IoT architectures for e-health. The 41 articles from the systematic mapping made it possible to observe how cloud, fog, and edge are used. Furthermore, it was also possible to observe how quality attributes are used and point out which ones are currently most used. Finally, observing how artificial intelligence is used in these architectures was possible.

After systematic mapping, based on the research question, "How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?"the following conjecture was established. Suppose we propose a self-adaptive architecture for developing IoT applications in e-health. In this case, can we improve the use of computational resources by systems, supporting data collection from intelligent e-health environments and supporting the management of these spaces?

With the mapping, research question, and conjecture in hand, an exploratory study was conducted to experience in practice the challenges listed in the literature and identify opportunities. The exploratory study validated the difficulties encountered related to computational resources and the quality attributes intrinsic to IoT architectures in e-health environments. Furthermore, with the architecture proposed in the exploratory study, it was possible to identify opportunities for improvements in the proposed architecture and related to the established conjecture.

To answer the research question, validate the conjecture, and execute the opportunities found in the exploratory study, we carried out two implementations of cycles of the design science research methodology. In the DSR cycles developed, the development of the architecture is presented in detail, as well as the difficulties, opportunities, and challenges mitigated through its development.

Given the above, this work presents an architecture to deal with the use of computational resources in intelligent e-health environments that use IoT, exploring self-adaptation and artificial intelligence to make resources available according to the demand of the environment in which the solution is inserted. To this end, the use of artificial intelligence will help to understand the dynamics that occur in the generation of data in the environment, providing learning when resources should be available and making them available through the computational infrastructure proposed in the architecture. The following chapter presents the proposed architecture and the execution of the research methodology for development.

4 PROPOSED ARCHITECTURE

The previous chapter presented a systematic mapping, related work, an exploratory study, the research question, and the work conjecture. Based on the analysis of the previous chapter, it was possible to identify opportunities in the literature related to computational resources for IoT architectures in e-health. Such opportunities are related to how computational resources are being used. One of the examples is related to its use being carried out 100% of the time, resulting in unnecessary resource expenditure. Additional opportunities are related to computational resources and exploring quality attributes for developing an architecture that provides large-scale data processing. Finally, artificial intelligence was observed in only one of the works to support the management of computational resources. Therefore, this chapter presents an architecture that deals with computational resources and quality attributes and applies artificial intelligence to support management. The architecture was developed to use sensors and IoT devices in a corporate e-health environment. In order to conduct the research process, the Design Science Research methodology was used.

The following subsections present the use of the methodology and also the cycles for building the architecture.

4.1 DESIGN SCIENCE RESEARCH METHODOLOGY

Design Science Research assumes that artifacts (solutions) are designed to solve a practical problem. Artifacts are objects whose construction followed scientific methods in different steps to generate knowledge about a specific item or area of knowledge. The focus of the DSR methodology is on the artifact and the relevance of its application (Simon, 1996).

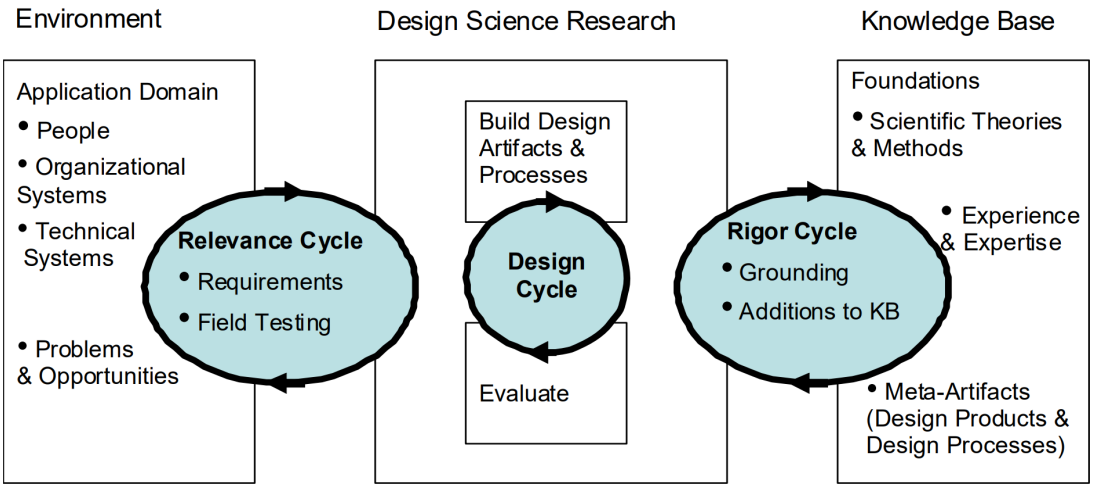
According to Wieringa (2009), research conducted using the DSR methodology deals with two types of problems: practical and knowledge problems. Practical problems require changes in artifacts that are related to decision-makers. Knowledge problems portray necessary changes to existing knowledge about the world and state of the art. Still, according to Wieringa (2009), a practical problem is responsible for safeguarding the research; therefore, other practical problems and questions about existing knowledge will arise from it. Both problems and questions chain a cycle of knowledge construction.

One of the design science research methodology proposals is its use in information systems and computer science. Its use allows software construction with well-established steps and establishes methods for constructing, evaluating, and generating scientific knowledge (Hevner et al., 2010).

Hevner (2007) exemplify using the DSR methodology in cycles. The three cycles are relevance, design, and rigor. The relevance cycle establishes that research must begin with

real-world applications. In this way, both research success requirements and acceptance criteria must be established. Another critical factor in the relevance cycle is defining the application domain, problems, and opportunities. The design cycle is responsible for establishing the construction of the artifact, focusing on obtaining feedback, and improving the artifact with the results obtained. Finally, the rigor cycle seeks to carefully ensure the application of developed theories and methods and evaluate established conjectures. In the last cycle, scientific knowledge is generated and provides advances in the research area. Figure 5 below presents the three-cycle model proposed by Hevner (2007).

Figure 5 - DSR three-cycle model



Source: Hevner (2007)

Added to the ideas proposed by Hevner (2007) is the work of Peffers et al. (2007). According to Peffers et al. (2007), the authors propose that the design science research methodology has a set of sequential and iterative milestones and steps, namely:

1. **Identification of the problem and motivation:** This phase aims to define the problem and demonstrate its importance.

2. **Defining the objectives of a solution:** Based on the question "What would a better artifact accomplish?"the authors propose that the objectives to be achieved with the artifact be defined here.

3. **Design and development of a project:** Responsible for leading the project and designing the artifact to be developed.

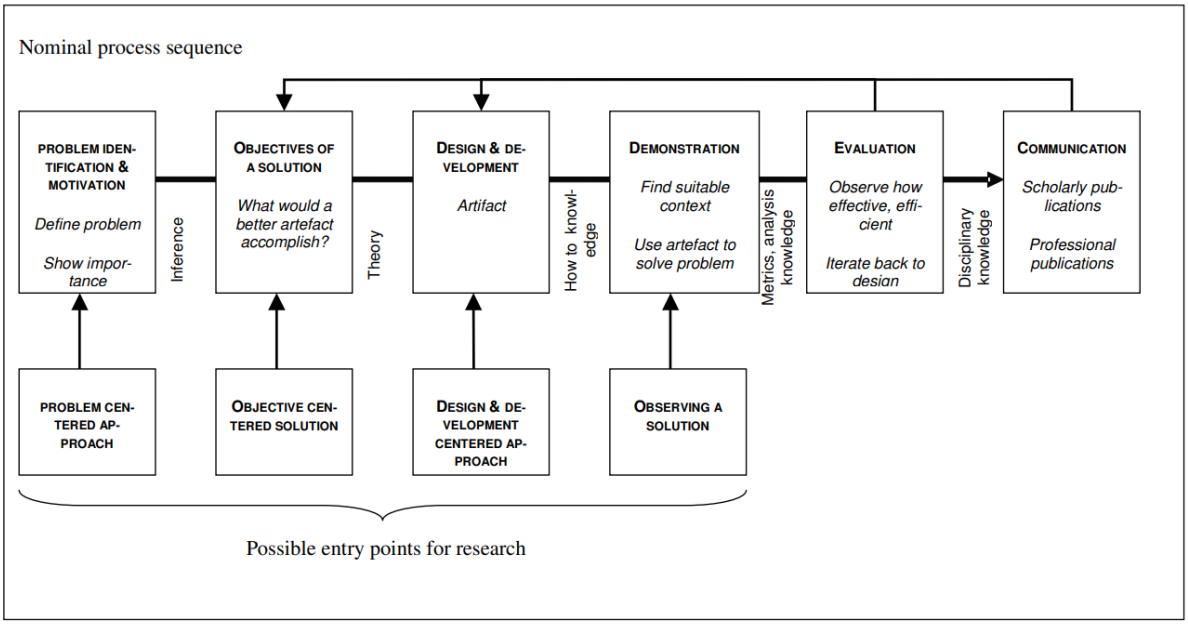
4. **Demonstration:** Presents the artifact solving the problem in its use context.

5. **Assessment:** Observe how effective and efficient the artifact was, which can result in the generation of new objectives and the execution of a new stage of design and development of the project.

6. **Communication and dissemination:** Generation of academic publications, contributing to the state of the art and generation of professional knowledge for the context in which it operates.

Figure 6 below presents the cyclical view of Peffers et al. (2007).

Figure 6 - DSR model proposed by Peffers et al. (2007)



Source: Peffers et al. (2007)

In the present work, the step-by-step instructions defined in the literature were followed to apply the DSR methodology. Systematic mapping and exploratory study helped to identify the problem, as well as to define the first objectives for a solution. Subsequently, in the next section, the design and development of the project are presented, as well as the demonstration, evaluation, communication, and dissemination. The following section presents the requirements established for implementing the architecture.

4.2 REQUIREMENTS

Defining functional and non-functional requirements is essential for developing the proposed architecture. For this, the application context (artifact) is that of a corporate

e-health environment. It must be capable of dealing with a large amount of data generated by different sensors and IoT devices. Furthermore, the architecture must be able to deal with the available computing resources and use artificial intelligence to predict the use of computing resources. Finally, the architecture must self-adapt to the environment's needs.

Next, the functional and non-functional requirements are established through the knowledge obtained in the systematic mapping and the exploratory study and the needs of the e-health environment.

4.2.1 Functional requirements

FR01: The architecture must be capable of processing, storing, and loading data from different IoT devices and sensors in an e-health environment.

FR02: The architecture must be capable of allowing the integration of different databases, as well as the storage of this data.

FR03: The architecture must display the stored data to support the management of the physical e-health space.

FR04: The architecture must adapt to the needs of the e-health environment, enabling and disabling the use of computational resources.

FR05: The architecture must use artificial intelligence to support computational resources.

FR06: The architecture must be capable of using computing resources such as cloud, fog, and edge.

4.2.2 Non-functional requirements

NFR01: The architecture must be capable of dealing with different data models and databases, at least two, relational and non-relational models. Each IoT device and sensor has characteristics and data storage needs. Therefore, the solution must be flexible and extensible to address this characteristic.

NFR02: The architecture must be capable of processing and storing large volumes of data. IoT devices and sensors generate large-scale data. For this reason, processing performance must be guaranteed.

NFR03: The architecture must maintain low response time and adequate data processing. Some sensors and IoT devices can help with decision-making. For this reason, the performance and reliability of data processing can be crucial in an e-health environment. Added to this is the need for some data processing to be carried out quickly.

NFR04: Processed and collected data must be protected to avoid unauthorized access or leakage. Security and privacy must be guaranteed so that user and system data are not exposed to undue people.

NFR05: The architecture must be scalable to handle the increase in IoT devices and sensors. In addition to supporting the increase in the amount of data generated. Environments are changeable and can receive new sensors and IoT devices; for this reason, the architecture must be scalable.

NFR06: The solution must be capable of using computational resources efficiently, providing their use only when necessary. The solution must adapt to the environment's needs to avoid wasting computational resources. In this work, efficiency is addressed by making computational resources available when necessary.

NFR07: The solution developed via architecture must be easy to use. Monitoring and decision-making dashboards must be built in such a way that they can be used without problems. In other words, users must be able to visualize data and use decision-making tools without any problem

NFR08: The developed system must provide clear and intuitive views of what the data represents. Usability regarding applications that use data is essential when developing the architecture.

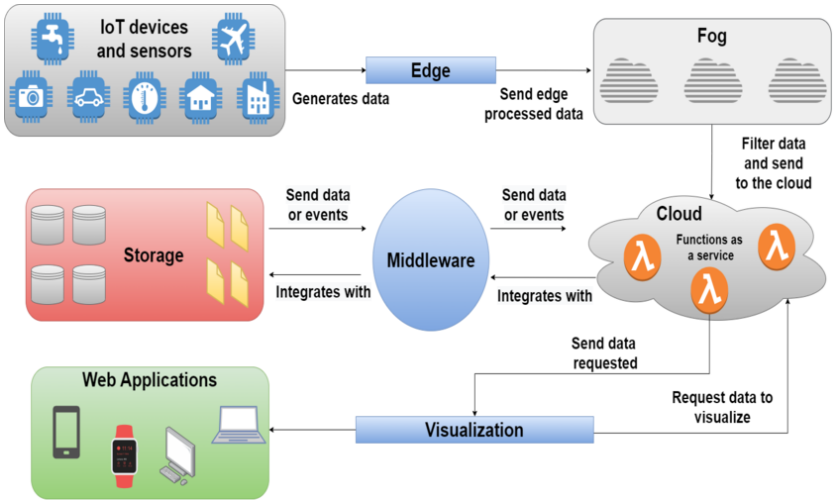
NFR09: The solution must offer ways of accessing data for interoperability with other applications. Integrating data generated with other applications may be necessary as the architecture expands. In this way, the database must interoperate with other solutions.

4.3 FIRST CYCLE

With the conjecture, the research problem, the systematic mapping, the exploratory study, and the functional and non-functional requirements, the first cycle of the DSR methodology was executed. This subsection presents the architecture construction process for the first cycle. The first cycle is related to one of our works, which can be seen in Do Nascimento et al. (2023).

An intelligent corporate e-health environment was considered to develop an architecture. The proposed architecture was divided into modules related to different functional and non-functional requirements for its development. Figure 7 presents the proposed architecture.

Figure 7 - Architecture proposed in the first DSR cycle



Source: Elaborated by the author Do Nascimento et al. (2023)

The architecture is divided into some modules, namely:

IoT Devices and sensors: Module responsible for all IoT devices and sensors in an intelligent environment. These sensors are responsible for generating data and can operate together or individually. Sensors and devices communicate directly with the Edge infrastructure.

Edge: Part of the architecture responsible for carrying out the first data processing using the capacity of devices on the edge. This layer performs the initial filtering of the data. Data filtering at the edge layer is performed based on rules. The data to be filtered must have a specific format, comply with necessary field rules, and define acceptable ranges for each data attribute.

Fog: Part of the architecture responsible for continuing data processing after the Edge layer. Fog's infrastructure already has greater processing capacity. Edge and fog allow greater data processing capacity and reduced response time, given their use close to the origin of data generation.

Cloud: The cloud infrastructure used functions as a service, which used snippets of code responsible for dealing individually with each sensor and device. Each sensor and device uses a specific function as a service in the proposed architecture. Therefore, when a new sensor or device arrives, a new function must be built to be used as a necessary resource. This approach is the part of the architecture responsible for dealing with the scalability of the use of devices and sensors. Functions as a service use the data access policy established by the middleware. At this point, the function transforms the data received by IoT sensors and devices into the data format required for storage. If necessary, the data is processed into the appropriate format and finally stored. Functions as a service

serve as a mapping and formatting layer to adapt data to the access policy established via middleware.

Middleware: Responsible for coordinating access to data, writing, and loading data servers. Furthermore, it connects functions as a service with the data layer. The use of middleware was necessary to guarantee security and data access policies and to allow coordination of functions as a service, ensuring that schemas are created for data consumption. These schemas deal with information about how functions can consume and access data.

Storage: Responsible for storing data on file servers and databases.

Visualization: Part of the architecture is responsible for requesting and interpreting data that has been stored. This layer ensures that data is available for decision-making based on its display.

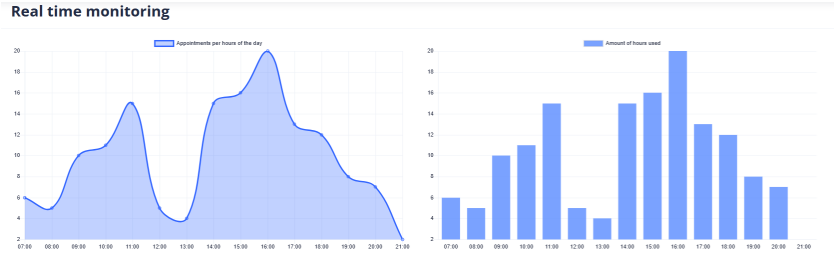
Web applications: Various web applications for viewing data generated from the environment. These applications can be on cell phones, computers, and even watches.

To deal with FR01, FR02, NFR01, NFR04, and NFR09, the architecture featured middleware, which maps the need for each IoT sensor and device. The middleware approach allows a function as a service for each sensor and device. Furthermore, the middleware has a layer that maps access to the database, generating access and control policies according to the needs of each application.

Given that each device and sensor has a function as a service, data processing began to be distributed between different parts of the computing resources. This meant the load did not just remain in a monolith, as in the exploratory study. Furthermore, using edge and fog allowed the filtering and processing of data close to the origin of its generation. This edge and fog approach also allows for reduced application response time. In this way, the needs of FR06, NFR02, and NFR03 were addressed.

The proposed architecture also deals with the proposition of a data visualization platform. For this, an application with dashboards was proposed. In this way, FR03, NFR07 and NFR08 were treated. Figure 8 demonstrates the application with dashboards.

Figure 8 - Developed application with dashboards



Source: Elaborated by the author Do Nascimento et al. (2023)

Although the first cycle of the architecture deals with several functional and non-functional requirements, some aspects can still be explored. They are:

- At certain times of the day, we notice that the computational resources are idle or receive a large load of data to process. The serverless paradigm supported these characteristics, but the computational resources could be better used according to interactions in the scenario. An example can be seen in the intelligent e-health environment operation; it operated between 6 am and 11 pm. However, all computational resources were available without real need throughout the day. Therefore, functional requirements FR04, FR05, and NFR06 were not addressed.
- The context in which the sensor is inserted directly influences the computational resources required. Various behaviors and actions can occur within an environment, for example, people walking, a sensor generating data, and a lack of electricity. In this way, the mapping of essential interactions to this environment must be privileged and mapped so that the computational resources are always prepared to receive this data. Functional requirements FR04, FR05, and NFR06 were not addressed as before.
- The usability of the dashboards and the tools to help the decision-makers could be improved, given the limitations encountered with the proposed visualization module. The NFR07 and NFR08 can still be better explored in a new architecture.
- The architecture presented how to deal with the entry of new IoT devices and sensors but did not present aspects related to artificial intelligence and the management of these computational resources. FR04, FR05, NFR05, and NFR06 were not treated.

4.4 SECOND CYCLE

This sub-section presents the second cycle of the artifact development phase proposed in the DSR method. This section presents the improvements needed after

collecting the results from the first cycle.

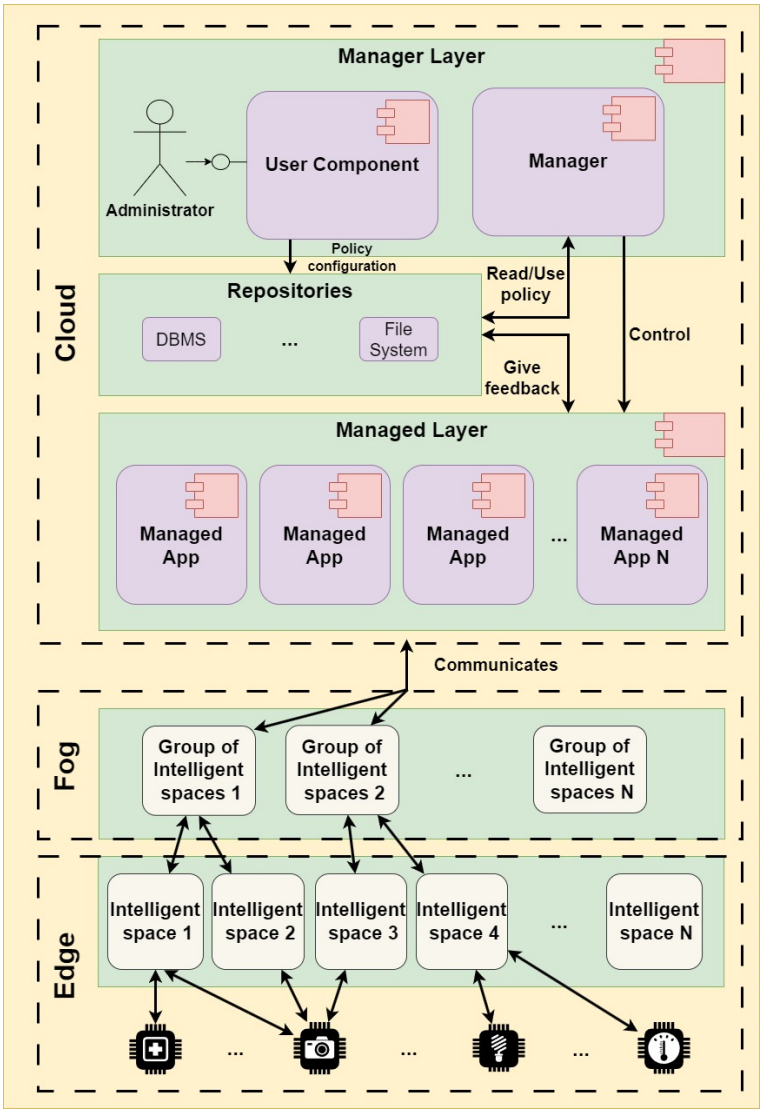
Given the expansion of intelligent environments and their presence in people's lives, computational solutions have become increasingly evident (Wilson et al., 2021). The development of these applications requires flexible computational architectures that can handle the input of new IoT devices and sensors, scalability to ensure that it functions correctly in times of large-scale use and interoperability with other systems and devices (Motta, De Oliveira and Travassos , 2018). Another need is supporting the interactions in these environments, adapting, and ensuring computational resource availability (Ortiz et al., 2022). The use of serverless allows the creation of flexible computational applications adaptable to other cloud structures and can scale according to existing needs. In addition to these benefits, serverless allows computational resources to handle the most minor parts of an environment, such as sensors and IoT devices. The use of edge and fog allows data processing close to the origin of their generation. Allowing pre-processing and filtering of data to be performed (Laroui et al., 2021).

In addition to the challenges above, we considered the lessons from the systematic mapping, exploratory study, and first DSR cycle to develop the artifact presented in this section. Furthermore, aspects related to scalability and interoperability of computational resources, data access policy, sensors, and real devices must be addressed. It is also necessary that the architecture can deal with the availability of computational resources according to the demands of the day and ensure that they are adaptable to complex interactions in the environment. We evolved the architecture developed in the first cycle to present a self-adaptive IoT architecture.

4.4.1 Architecture development

A self-adaptive IoT architecture is proposed to deal with FR04, FR05, NFR06, NFR07, and NFR08. The self-adaptive IoT architecture proposed provides a mechanism for choosing computational resources that meet the needs of e-health environments – mapping the use of sensors and IoT devices and exemplifying how processes occur from data capture to use across the applications concerned. The details of the architecture are illustrated in Figure 9.

Figure 9 - Self-adaptive Architecture proposed in the second DSR cycle



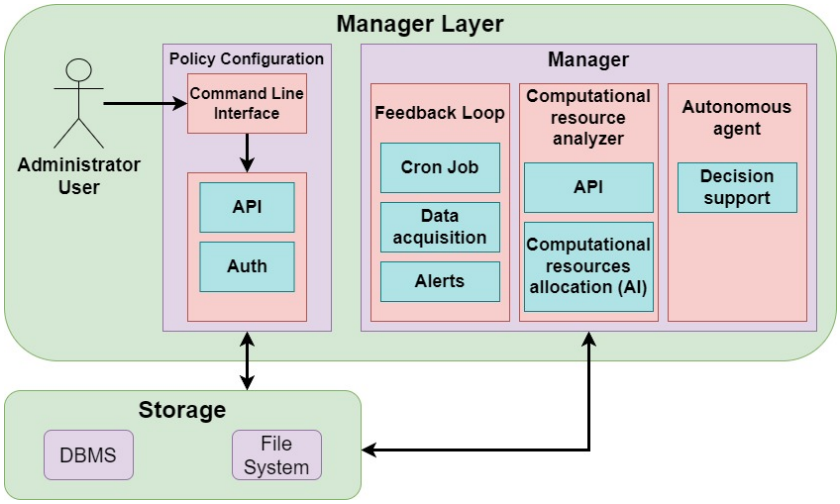
Source: Elaborated by the author (2024).

In the cloud, the first layer is called the Manager Layer. It is responsible for managing whether the managed system meets the non-functional requirements established by the developers and stakeholders, such as scalability, interoperability, response time, memory usage, and processing usage. It is also responsible for changing the computational resources according to these definitions. The second component is the Managed Layer. It is the system the final users use to monitor the intelligent e-health environment. The Managed Layer will also be self-adapted according to the definitions established by the Manager Layer. Connected to the Manager Layer and the Managed Layer, the repositories store helpful data for the Manager Layer and also about the health of the managed system and the environment, such as response time metrics, amount of memory used, and amount

of processing. We have sensors and IoT devices that generate data in edge and fog. The edge serves as an initial data processor in smaller intelligent environments. Fog already performs another data processing. However, it performs the processing of two or more intelligent environments together. Finally, the generated data is sent to the managed systems.

The proposed self-adaptive architecture aims to adapt the use of computational resources according to the needs established through non-functional requirements and scenarios of using sensors and IoT devices. An example of adaptation is that at a specific time of day, a computational resource is necessary for data processing, with up to 5 seconds of response time, from a temperature sensor to determine whether the environment is comfortable. For this, the architecture divides the Manager Layer into some internal components. To support our arguments, Figure 10 presents the Manager Layer in detail.

Figure 10 - Manager layer and its components



Source: Elaborated by the author (2024).

The first component of the Manager Layer is the user component. This user is a system administrator. The administrator must establish which IoT devices and sensors are necessary to allow the architecture to adapt to different situations in the intelligent environment. Next, the user must establish which serverless computational resources are required for these IoT devices and sensors to function correctly. The set that involves IoT items and computational resources characterizes the use scenarios of the intelligent environment.

An example could be a doctor's office using a presence sensor, temperature, humidity, and devices to monitor a medical appointment. The system administrator defines the IoT devices and sensors to be (i) an occupancy sensor, (ii) a thermometer, (iii) a humidity sensor, (iv) a tablet, and (v) an intelligent door lock. Then, the administrator defines

the serverless computing resource required for each IoT device and sensor. In this case, a function is a service for each device and sensor. Usage scenario configurations are an essential part of the architecture. The user configures all the settings in the Policy Configuration. The repository component stores the scenarios in a database, relating IoT devices and sensors with computational resources. The administrator must be able to create, change, and delete configurations, given the needs of the scenarios.

The Feedback Loop is another essential factor in architectural design. It monitors the usage of the scenarios, devices, sensors, and computational resources used. The architecture has a structure to store data from executing adaptive scenarios. It is responsible for logging scenario data, issuing alerts for misconfigurations, and managing the necessary computational resources for usage scenarios for the managed system.

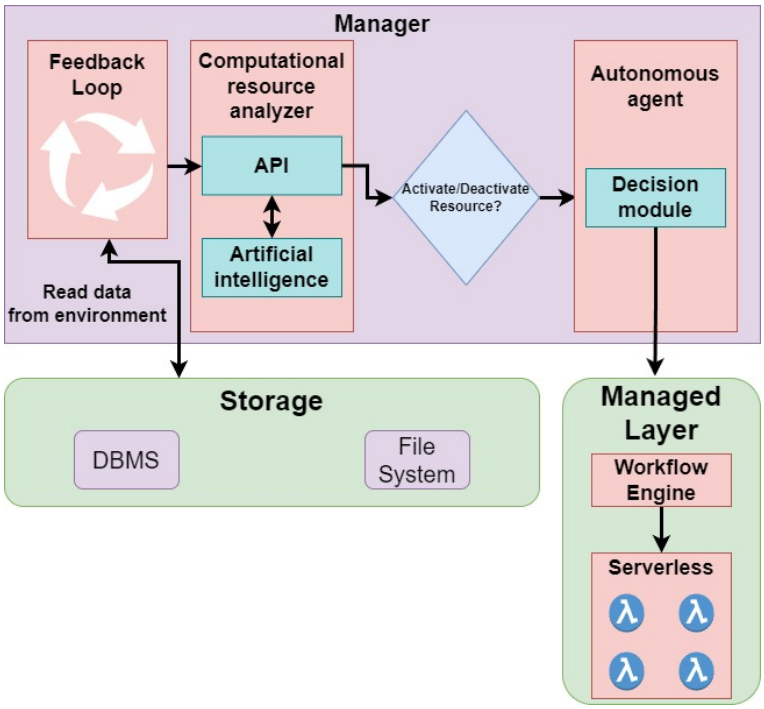
When reading scenario data, the Feedback Loop establishes whether the managed system's operation is following the needs of the intelligent environment. The serverless framework is essential for this, as it evaluates each component required for an IoT device or sensor at a lower granularity. This approach allows the Manager Layer to execute a modification individually based on historical configuration if a computational resource that is not working is identified. Furthermore, Feedback Loop also issues alerts on standards defined. An example may be the response time required for a computational resource to monitor a patient. A sensor that monitors heartbeats may require a low response time. If the sensor is not reaching, the administrator must issue alerts.

The Manager visualizes how healthy the managed system is and adapts it to the environment's needs. It assesses whether scenario configurations follow those defined by administrators, manages new configurations based on situations experienced in intelligent e-health environments, and focuses on using computational resources.

To adapt to the environment's needs, the architecture uses artificial intelligence. In the context of intelligent environments, artificial intelligence is essential to optimize the allocation of computational resources. By analyzing usage patterns, it decides when to activate or deactivate resources, dynamically adapting to the environment's needs. Furthermore, predictive analysis techniques predict future demand, reduce waste, and improve operational efficiency for more sustainable and economical management. The computational resource analyzer must evaluate when the scenarios' computational resources should be activated or deactivated. For this, the Feedback Loop must also record the times of use of computational resources and which scenario performed their use. In this way, a resource usage history is necessary, and the computational resource analyzer should be able to assess when the applications must be adapted to activate the computational resources. Artificial intelligence can help to improve the management of the intelligent environment, reduce the unnecessary use of computational resources, and, consequently, allow more efficient management of computational resources. The use of loosely coupled

cloud resources that can be easily activated and related to sensors is necessary for the proposed architecture. The loosely coupled resources allow artificial intelligence to define which computational resource is vital for the scenario and understand when its use is necessary for the environment. The data to be consumed by the Feedback Loop and the Computational Resource Analyzer can be stored in relational databases or file systems. Figure 11 demonstrates how the allocation of a computational resource works in the proposed architecture.

Figure 11 - Presentation of components to perform the change of computational resources



Source: Elaborated by the author (2024).

Based on Figure 11, the autonomous agent can turn a computational resource on and off according to the time of day and stored data. The Feedback Loop executes a request to the Computational Resource Analyzer. The Computational Resource Analyzer connects to artificial intelligence to analyze whether it is time to activate or deactivate a computational resource. If it is necessary to activate or deactivate a computational resource, the Computational Resource Analyzer invokes an autonomous agent. This autonomous agent is responsible for acting on the managed system, changing the configurations of computational resources. An example is a humidity sensor that needs a computational resource at a specific time of day. The Feedback Loop executes from time to time calls to the Computational Resources Analyzer requesting the assessment of the need to use the computational resource for the sensor. The artificial intelligence evaluates that the sensor

resource must be activated and triggers an event for the autonomous agent. Then, the autonomous agent acts on the managed system, configuring it for the need.

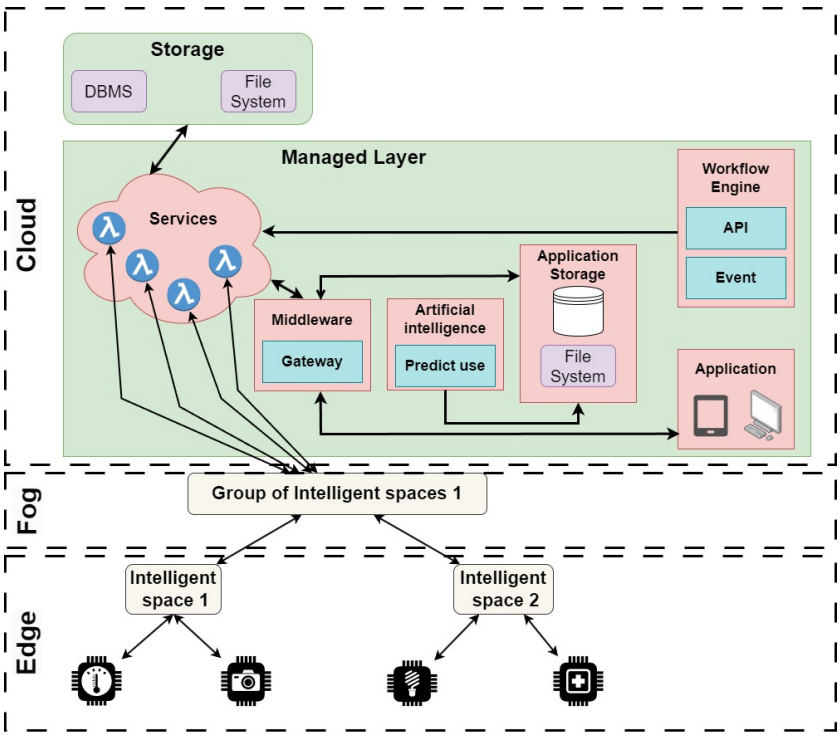
The proposed architecture allows the use of different artificial intelligence techniques. Examples are machine learning, expert systems, recommendation systems, and automatic learning (AutoML). Some algorithms that can be used for these techniques are decision trees, random forests, linear regression, logistic regression, and collaborative filtering. One example is the use of AutoML. In the AutoML process, the decision on which algorithm to use is made through established criteria that evaluate the adequacy of the model to a given algorithm. The first phase of the selection process is preparing the test data. This phase executes the data cleaning and transformation. In the next phase, different algorithm models are evaluated on the data set. Each model's performance is evaluated using precision, recall, and F-1 score metrics. Given the models' performances, the worst are discarded. The models' hyperparameters are optimized to extract better performance from the models. Finally, in the last phase, there is the final choice of the model and which algorithm suits the chosen model (Karmaker et al, 2021).

Artificial intelligence trains with historically recorded data, allowing training sessions to be executed occasionally and allowing model learning and validations to be performed according to the time of year. One of the critical aspects is that one or more computational resources can interoperate, causing computational resources to run simultaneously. Furthermore, given the characteristics of the environment, computational resources can be scaled because of the loosely coupled cloud resources into the architecture.

In the proposal for using artificial intelligence, the characteristics of the model chosen via AutoML are taken into account when choosing an algorithm. This approach allows for greater assertiveness in using the algorithm with the model, allowing metrics related to the model to be optimized and explored. Given that the entire AutoML process checks the adequacy of the model to the algorithm, this use together brings benefits in terms of metrics related to precision, assertiveness and F1-score.

The Managed Layer is responsible for running the application used by users. It has all the necessary computational infrastructure defined in the Manager Layer. It is composed of a serverless component that uses functions as a service to communicate with IoT device sensors; a middleware component to isolate access to the database from the application and allow the definition of usage rules; the database to store application data, the application component to follow the mapped intelligent environment; a workflow engine component for changing serverless component settings. Figure 12 demonstrates the proposed architecture for the Managed Layer component.

Figure 12 - Managed layer and its components



Source: Elaborated by the author (2024).

In our proposal, the serverless component uses a computing framework that should use functions as a service. When executed, the function distributes data to the middleware. In addition to functions as a service, an API Gateway receives data generation events and looks for the function that should be activated. If the function does not exist, the gateway generates an error.

Middleware is the computational structure of the architecture responsible for integrating different data consumers, such as serverless and databases. One of the main goals of middleware is to ensure that only it has access to the application storage. This way, the middleware defines data access policies in its structure, and for each new interested service, the middleware must map the use. This approach allows the coordination of application storage access. As in the first cycle, the middleware's function is to centralize access to databases and coordinate communication, access, and cooperation between them. Within the managed layer, artificial intelligence can be used to support decision-making regarding the application. For example, a sensor stopped generating data at a specific time. Artificial intelligence can help predict the use of this data.

The application component deals with visualizing the data. Applications are constructed in this component to serve those interested in the data generated by IoT sensors and devices. In this way, the IoT systems in intelligent environments perform

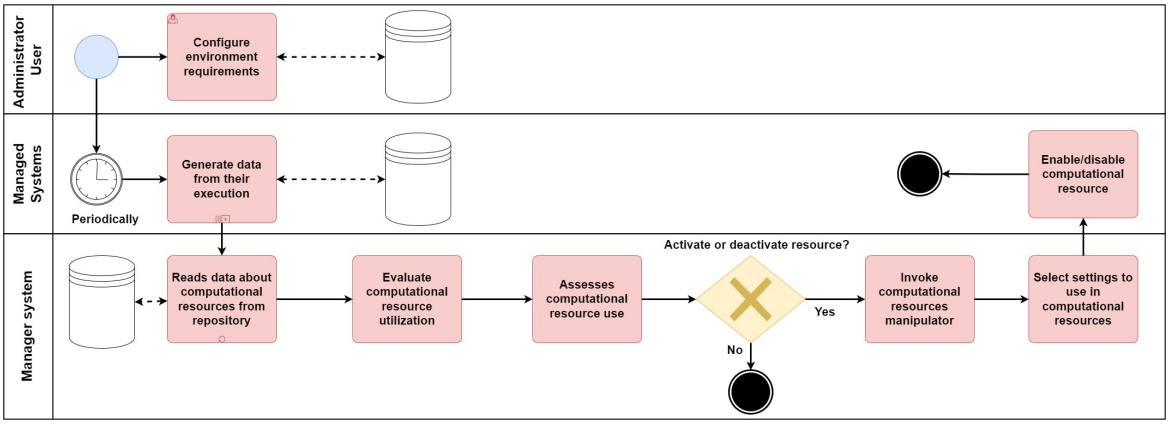
requests for middleware to obtain data. These functions communicate with the middleware and receive the requested data in return.

The last part of the managed system is the workflow engine. It is responsible for changing the configurations of computational resources according to the needs of the scenario. It executes the configurations defined by the managed system. The workflow engine has an API, and whenever it needs to change some computational resources, it sends an event to the cloud to execute the changes.

Finally, the self-adaptive architecture still proposes the environment, edge, and fog. It is the part of architecture that is responsible for the intelligent environment. In it are available devices, sensors, and people interacting with each other and the devices. In an e-health context, an assisted home environment that uses presence sensors, temperature, and cameras is an example of an environment.

The environment, through the intelligent environment, generates the data. Each device and sensor acts in this context, generating data related to its purposes. For example, a temperature sensor generates data about the environment. The data needs to be used by the managed system and end users. Computational resources are essential for this data's loading, processing, and visualization. Figure 13 demonstrates the proposed workflow.

Figure 13 - Computational resource workflow



Source: Elaborated by the author (2024).

The heterogeneity of IoT devices and sensors is one of the challenges that computational resources must deal with. In addition to heterogeneity, health applications in the field of IoT need data processing with low time and adequate response time. To address these needs, the architecture proposes using data processing with edge computing. Edge processing decreases the response time, increases the data throughput, and reduces the delay. Furthermore, edge processing allows the first data filtering and verifies data

integrity. At the edge, the amount of data is abundant. Applications must define what data is important to them after processing with the edge framework. An example is an e-health application that deals with a hospital emergency and has a temperature sensor and a presence sensor. When a new patient enters, a presence sensor notifies that a new person has entered the environment. In this case, the data that notifies the arrival of a new patient is more important than the one that monitors the environment's temperature. So, at the edge, there is many data. The edge ends up filtering, but it needs to be even more filtered. The architecture also proposes using fog computing after the edge to support data processing and prioritization of essential data. We have less data when analyzing the fog due to the processing performed at the edge. Fog technology helps data processing. It works as a mediator between data generators and data storage software. The distributed structure of the fog layer works close to the data generators even though the fog layer does not offer significant computational resources or edge computing.

The main objective of using the edge and fog proposal is to ensure we process more data close to their origin. As a result, we ensure that only application-relevant data gets into the storage structures, reducing response time and distributing data processing in the different available structures.

Even using the Edge and Fog structure, each IoT device and sensor still needs computational resources to store and process data. In this way, computational resources are essential to correctly use data generated from the environment. A differential of the proposed architecture is in the managed system with serverless computing, providing computational resources that provide flexibility and scalability for sensors and devices. Architectures for IoT systems often have a computational structure to handle different data. For example, a software layer with an API or a web service will receive data from different devices and perform the treatment, filter, and storage. In this way, an approach like the one presented in the architecture can enable the construction of loosely coupled services, such as serverless technologies that allow the distribution of data collection and processing tasks.

The second cycle of the DSR aimed to deal with FR04, FR05, and NFR06. For this, the entire component-based architecture was presented. In it, the use of computational resources is presented according to the needs of the e-health environment in which they are inserted. Furthermore, the use of artificial intelligence was presented to enable processing and self-adaptation of the architecture to different scenarios existing in e-health environments. The approach makes computational resources predictable, supporting the avoidance of inappropriate use. This approach allows for reducing costs and waste of resources and a more adequate management of the environment's needs, whether or not they are available.

In addition to FR04, FR05, and NFR06, NFR07 and NFR08 were explored. Within

the Managed Layer, an application component was used so that different applications could be built and their use adapted according to the needs of decision-makers in the intelligent e-health environment.

4.5 FINAL REMARKS OF THE CHAPTER

Based on systematic mapping, exploratory study, research question, and established conjecture, this chapter presented two development cycles of an artifact based on the design science research methodology.

The functional and non-functional requirements of the architecture were defined to conduct the DSR cycles. Then, cycles were executed to meet functional and non-functional requirements and objectives. During the cycles, practical and theoretical knowledge was acquired, resulting in improvements and new features in the architectures.

Two architectures were developed. The first cycle architecture did not address computational resources and artificial intelligence issues. The architecture only used computational resources. The architecture of the second cycle dealt with these themes and was developed after experiments in the first cycle. Its proposal aims to use computational resources and aspects of self-adaptation to the environment's needs and use the artificial intelligence approach to support the architecture. Therefore, the objective of the architecture is to allow computing resources to be available when IoT devices and sensors are needed. The e-health environment was used, given the previous environments in which the first and second cycles carried out experiments.

The chapter also presents communication between different actors and layers of architecture. Each layer has a responsibility, which must guarantee the functioning of the architecture. No layer is more important than the other, as only interoperability will guarantee correct functioning. Among the objectives of the architecture is to ensure that aspects related to the fair use of computing resources are carried out, not resulting in indiscriminate availability problems or wasting too many resources.

Finally, the first and second cycles present two architectures for developing IoT applications in e-health environments, focusing on computational resources. In this way, the proposed architectures respond to the conjecture that architecture can support data collection and the management of intelligent e-health spaces. The next chapter presents two case studies conducted in an intelligent corporate e-health environment to answer the conjecture.

5 EVALUATION

This chapter presents a real-world case study to explore the following research question to evaluate the artifacts: How can self-adaptive architecture support automated computational resource allocation in e-health environments? As previously presented, this study followed the Design Science Research (DSR) methodology. This chapter will present the architectures proposed in the first and second cycles through two case studies.

In Hevner (2007), Wieringa (2009), Peffers et al. (2007), evaluation is presented as one of the fundamental parts of the DSR methodology. To create practical and theoretical knowledge, all methods and practices to develop the artifact must be presented and discussed at this stage. This way, scientific knowledge can be beneficial and built from evaluating case studies and DSR cycles.

In addition to the DSR methodology, the objective is to evaluate whether the proposed architecture can support allocating computational resources in intelligent e-health environments. In this way, the scientific knowledge generated can be evaluated and included in the literature and the state of the art.

Next, the phases that precede the case studies are demonstrated. Finally, the results of the case studies are presented in detail.

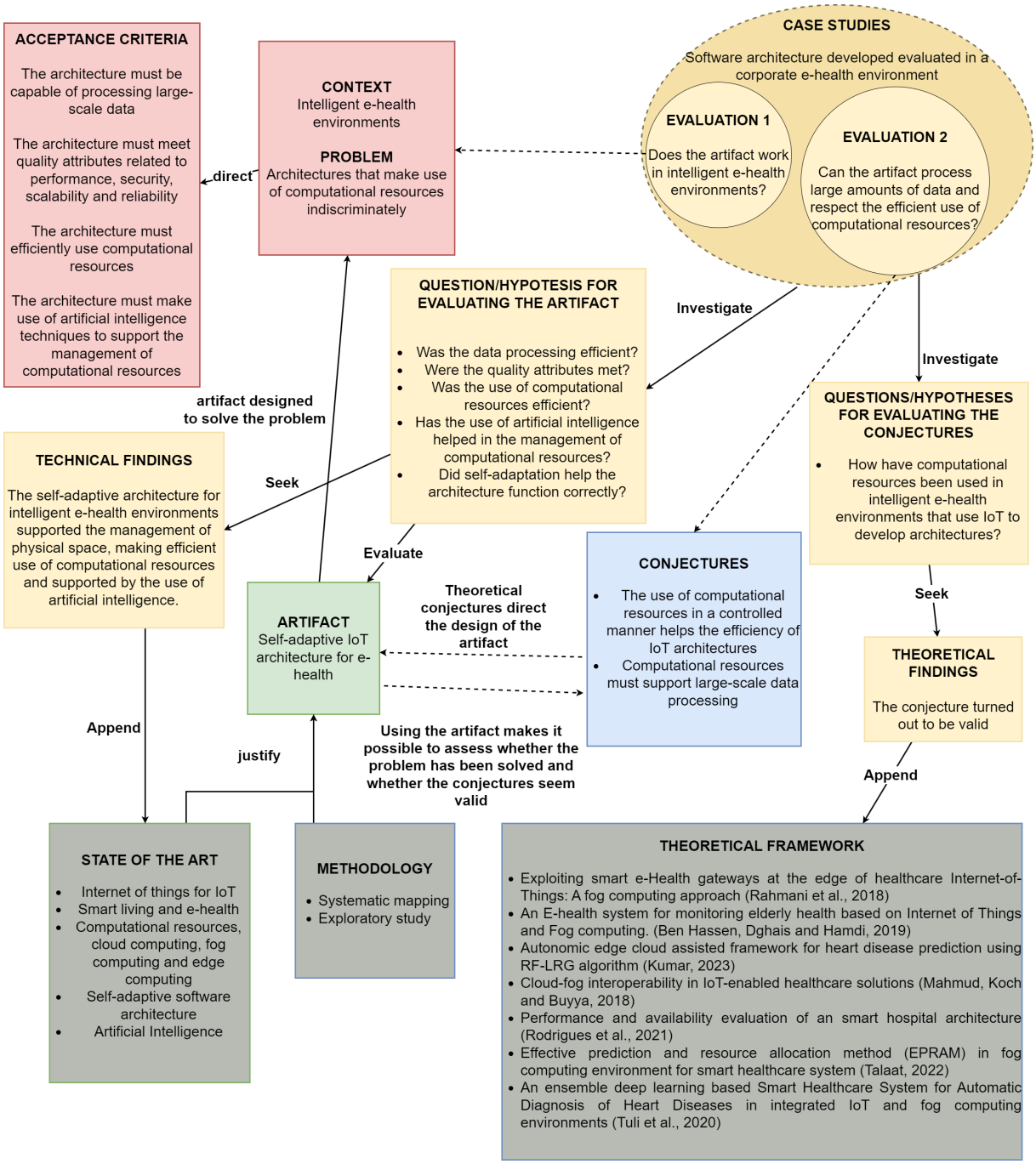
5.1 METHOD

This work aims to present the efficient use of computational resources in intelligent e-health environments, relying on artificial intelligence and self-adaptive architectures. Given the objective of the work, the research question was discussed: "How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?".

A systematic mapping was conducted to address the research question. Based on what was found in the literature, an exploratory study was carried out to visualize the challenges and opportunities and identify new items of interest in the research. Finally, two cycles of DSR were conducted.

Given the above, to follow the DSR methodology, a summary of the steps is shown in Figure 14.

Figure 14 - Architecture DSR assessment



Source: Adapted from Pimentel, Filippo and Dos Santos (2020).

Based on the theoretical foundation found in the literature, state of the art, and the methodology applied in the research (systematic mapping and exploratory study), we developed and evaluated an artifact capable of supporting intelligent e-health spaces. The evaluations sought to verify whether the artifact contributed to solving the problem, mapping the theoretical and practical contributions. This way, existing knowledge was

compared, and new knowledge was generated through interactions in two case studies.

5.2 CONTEXT

We designed the proposed architectures and tested the applications developed in a corporative e-health environment. The intelligent environment was a medical clinic with a coworking, waiting room, and medical offices for patient care.

5.3 PLANNING

We conducted a planning step to understand the needs of the intelligent environment. We mapped the sensors and devices used in the medical clinic. Then, we talked with users of the intelligent environment to understand people's interactions, in addition to sensors and IoT devices. Finally, we identified the users of the environment.

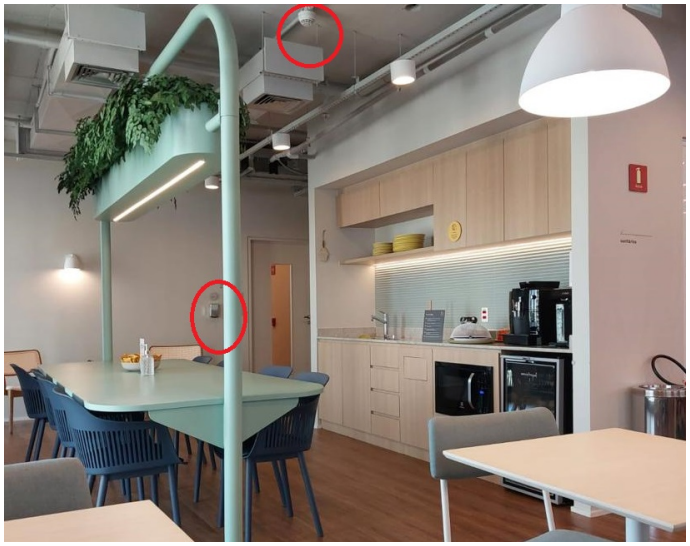
In the study's planning phase, we mapped the existing technologies, spaces, interactions, and needs. The devices and sensors in the spaces of the clinic are:

- Coworking: Presence sensor, temperature sensor, and humidity sensor.
- Waiting room: Presence sensor, cameras, and totem.
- Doctor's office: Tablet.

The functioning of the presence sensor works to detect a person in the environment periodically. The presence sensor is available in all medical clinic areas, except the doctor's office.

Temperature and humidity sensors, for example, perform coworking measurements every hour. Figure 15 demonstrates the coworking environment.

Figure 15 - Coworking environment



Source: Elaborated by the author (2024).

The cameras are used in real time to monitor the flow of people in the waiting areas. Figure 16 shows the waiting room area.

Figure 16 - Waiting room environment



Source: Elaborated by the author (2024).

The totem is responsible for recording the arrival of patients for consultations and receiving payment for the services – finally, the tablets in the doctors' offices record which

health professionals are using the environment. Figure 17 illustrates the tablet and totem devices used.

Figure 17 - Totem and Tablet



Source: Elaborated by the author (2024).

As a result, all devices and sensors of the intelligent e-health environment were presented.

5.4 CASE STUDY 1 - FIRT DSR CYCLE

The first architecture was built with systematic mapping, exploratory study, research question, conjecture, and the scientific method of DSR. The architecture was built through modules to support medical clinics in understanding the dynamics between patients and healthcare professionals. Each module proposed in the architecture had an objective, as presented in the previous chapter, and supported achieving the established functional requirements.

The architecture of the first DSR cycle met the following functional requirements, namely FR01, FR02, FR03, and FR06. As the case study scenario stipulated, the environment had IoT devices and sensors. The developed architecture communicated with these sensors and devices, which were integrated, and their generated data was stored (FR01). In addition to the data being stored, some of the data generated was stored in non-relational databases, given the specific processing needs of the sensors (FR02). The entire proposed architecture used cloud, fog, and edge resources to assist in processing data close to the source and filtering it (FR06). Finally, an application for visualizing what data from sensors and devices were generating was also developed (FR03).

The architecture could meet non-functional requirements NFR01, NFR02, NFR03, NFR04, NFR07, NFR08 and NFR09. As presented, the architecture supported using

different sensors and IoT devices in the environment, integrating different data models and databases (NFR01). The data models used were relational and non-relational. This way, application growth is guaranteed. Although the sensors and devices generated a large amount of data and were stored, the first case study did not generate stress on the application. This way, large-scale data was stored, and response time was not an issue (NFR02 and NFR03). This characteristic meant that aspects related to performance and efficiency were addressed.

Regarding security and privacy (NFR04), introducing middleware with database access policies ensured that data was protected. The middleware not only served the security aspect, it also addressed the interoperability of the application by allowing access to data by different applications and integrating different generated data models (NFR09). The aspects related to the solution (NFR07 and NFR08) were partially met in the first cycle, as improvements were still found to deal with specific IoT data listed on a large scale, which hindered the system's usability.

The case study analysis is divided into four parts: preparation, conduction, analysis, and obtaining results.

5.4.1 Preparation

Given the developed corporate scenario, any impact from the operation of the physical space should be mitigated. For this reason, the first case study extracted the data from where it is stored to access IoT sensors and devices. This way, a script to obtain the data was created and stored according to the necessary databases. In this case, a relational database, a non-relational database, and a file server.

Before the data was stored, it was cleaned to ensure that only the data of interest were used in the application. This cleaning consisted of an essential phase for the architecture, given that the sensors and devices in the corporate environment generated diverse data for different applications. In the case study to be carried out, the data of interest were related to the arrival of patients and the environment in which these patients would be cared for. In this way, data unrelated to this need was discarded and cleaned.

With data cleaning, again, the data was stored in its respective necessary databases. Again, relational and non-relational databases and file servers were used. In the case of the relational database, data relating to which room in the physical space was used for service was stored. In the non-relational database, general presence and environmental temperature data were stored. Finally, file servers stored photos and videos of the environment. In order to maintain the privacy of people in the environment, data used was anonymized, and images from file servers had blurred faces of people who frequented the environment. Figure 18 shows how the temperature data were stored.

Figure 18 - Temperature data stored model

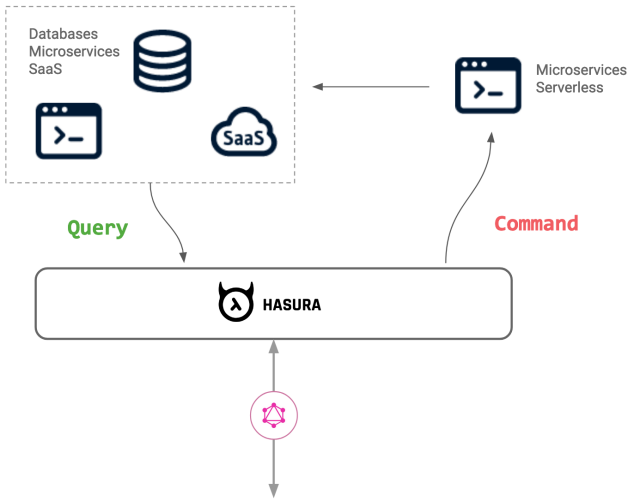


Source: Elaborated by the author (2024).

After storing the data, the middleware began to be used. Using the Hasura tool, different databases were connected, and access to them was established through the access policy for these databases. Hasura[9] is a software development tool that acts as middleware, which can connect different databases, create access policies, and create basic API actions, such as HTTP Post, Get, Put, Patch, and Delete requests. Figure 19 below demonstrates an example of how Hasura works.

[9] https://hasura.io/

Figure 19 - Hasura architecture



Source: Extracted from https://hasura.io/blog/introducing-actions/.

With the data stored and processed, a visualization tool was proposed to support operation and decision-making within the physical space. The following subsection details how the tool was developed and how end users benefited.

5.4.2 Conducting

One of the fundamental steps of the case study is to understand how the developed application supports the management and functioning of the physical space. In this way, those impacted by the proposed application and the architecture development are the workers who work in such an intelligent environment.

With the stored data, an application using the proposed architecture was developed. This application allows data visualization on what occurred in the environment and supports physical space management. An example of the developed application can be seen in Figure 20.

Figure 20 - Developed application architecture



Source: Elaborated by the author (Do Nascimento et al., 2023).

Figure 20 shows that the application is divided into some parts. The first part deals with information about a room and what purpose it is used for. The second information presents what is happening in real-time, which doctor is treating a patient, and whether the room is occupied.

The application developed aimed to support understanding what was happening in that environment. Environmental employees were asked to use the application and reported that, through what was developed, they could understand the dynamics between patients, environments, and healthcare professionals.

Initially, the data generated by IoT sensors and devices was stored. Then, a script processed the data of interest and stored this data again. With the data of interest stored in relational and non-relational databases and file servers, the middleware could be connected to these databases to make the data available and create applications. Access to data from each sensor used a function as a service that connected with API endpoints automatically generated by Hasura (middleware) to list, store, update, and delete data. The application developed allowed data visualization and was used by employees who care for the intelligent environment. The stored data, the middleware, and the developed application used the cloud computing resource to execute correctly. The edge occurred in processing close to the origin carried out by IoT sensors and devices. The use of fog occurred through filtering on the data right after such devices generated the data.

5.4.3 Analyzing

Through case study 1, the developed architecture was evaluated to understand how much it supports data processing and management of the physical environment through computational resources. In addition to this assessment, functional and non-functional requirements were also assessed whether they were in accordance with the established conjecture and the research question.

Given the exposed scenario of interactions in the corporate e-health environment, the first architecture aimed to visualize how interactions between healthcare professionals, patients, and physical spaces occur. This way, the necessary databases were sought, and the developed architecture was applied to measure its functioning.

Through the development of the architecture and monitoring the use of the corporate intelligent environment by employees, it was possible to observe a satisfactory result in terms of data processing, scenario visualization, and also aspects related to middleware to guarantee connection with different databases and integration with different functions as a service in other contexts. Finally, the cloud application assisted in processing and ensuring the functioning of the application and middleware.

5.4.4 Results

At the end of the first DSR cycle, the entire architecture and the use by employees of the corporate e-health environment could be evaluated. Through the application shown in Figure 20 by analyzing the application, it was possible to observe how the dynamics of the environment occur, as well as gain experience with IoT interactions, devices, and sensors in a way that allows improvements and learning to be carried out in a new DSR cycle.

The learnings from the first case study and the first DSR cycle allow us to understand how IoT devices and sensors behave in intelligent corporate e-health environments. Furthermore, it allowed the understanding of how middleware works to enable aspects related to security, scalability, and extensibility. Security was addressed through data access policies. Scalability allows the simultaneous execution of several accesses to the database and file servers. Extensibility was addressed by creating standard methods to handle HTTP requests that do Post, Get, Put, Patch, and Delete.

Regarding performance, functions as a service isolated all data processing according to the area of interest for which they were used. This way, computational resources were divided between IoT devices and sensors and their data processing purpose.

Although it supported the management of physical space, the application developed through the architecture presented some points of improvement related to the use of computational resources, the usability of the tool for data visualization, and how new sensors and IoT devices would be introduced. Furthermore, the first cycle architecture did not address some of the proposed functional and non-functional requirements, such as FR04, FR05, NFR05, and NFR06. Next, the aspects of improvement found are discussed.

With the developed architecture and the use of cloud, fog, and edge, it was possible to observe that computing resources were available 100% of the time. The corporate environment only operated from 7 am until 10 pm. What can result are financial losses; excessive consumption of electrical energy; greater exposure of computing resources to privacy and security issues; infrastructure overload, in case of many parallel services required, increasing maintenance costs and decreasing service quality; risks of persistent attacks; resistance to scalability of resources, as they are always available at high capacity. Using computational resources could be made more efficient by employing artificial intelligence to adapt the use of computational resources according to the environment's needs. Another benefit that could be noted is self-adaptive architecture, allowing resources to be adapted through artificial intelligence.

Another problem noted was that data about the environment could be missing in the event of network or electrical power failures. The architecture was unprepared to deal with these situations, which caused data visualization errors and hindered the application's reliability. Therefore, employees in the intelligent environment found themselves having

problems understanding what was happening in the environment. A solution may be to predict this data when it is missing or adapt visualization tools if this data does not exist.

In addition to the abovementioned problem, another challenge is that the visualization application was quite simple. In this case, it was impossible to see historical data of what happened in the environment and rooms, which only allowed the understanding of what was happening at the time of viewing and not the past. One way to solve this is by displaying historical and current data, supporting management and decision-making.

Another aspect is that although the middleware offers scalability and extensibility for new IoT devices and sensors to be added, the application needed to prepare to allow the visualization of new data generated by sensors and devices. A more scalable tool that allows visualization would need to be used.

Finally, using artificial intelligence, one of the architecture's functional and non-functional requirements still needs to be realized. In this way, artificial intelligence can be used in the next cycle to support the management of computational resources and help with data predictability in the case of problems with the unavailability of sensors and devices.

The module-based architecture met most of the proposed functional and non-functional requirements. However, through the improvements presented, a new DSR cycle is necessary to provide improvements and new scientific and practical knowledge. The new cycle was necessary mainly to address aspects of artificial intelligence, computational resources, and usability improvements. Furthermore, the established conjecture must be revisited in the new cycle through artificial intelligence and computational resources.

5.5 CASE STUDY 2 - SECOND DSR CYCLE

With the scientific knowledge from the first cycle of DSR, the proposed artifact was improved, thus proposing a new architecture and a new round of evaluations to measure the practical and scientific knowledge acquired. The architecture of the second DSR cycle was separated into some components related to aspects not addressed in the first cycle, such as using computational resources and artificial intelligence.

The architecture of the second cycle of the DSR, in addition to supporting the functional requirements already presented and discussed in the first cycle, FR01, FR02, FR03, and FR06, must also meet FR04 and FR05. FR04 deals with adapting the architecture to the environment's needs, turning computational resources on and off according to environmental conditions. FR06 deals with the use of artificial intelligence to support computational resources. As well as functional requirements, the architecture must support non-functional requirements not discussed in the first cycle, namely NFR05 and NFR06. NFR05 addresses the architecture's scalability for adding sensors and IoT

devices and allows increased processing of new sensors and devices without compromising application performance. NFR06 addresses the efficient use of computing resources and their use only when necessary. Finally, aspects related to improving usability are also addressed to provide more significant support for managing physical space.

To support resource management, artificial intelligence is presented in a way that can support the use of computational resources. In the following subsections, the development of artificial intelligence is presented.

Finally, this case study is divided into 4 phases: the first is planning, the second is conducting, the third is analysis, and the last is results.

5.5.1 Preparation

Considering the sensors and IoT devices, as well as the environments present in the medical clinic, a mapping of the use of the environment was carried out and divided into two usage scenarios to be supported using the self-adaptive architecture.

The first usage scenario ensures the professionals' coworking environment has a pleasant temperature. The medical clinic is open from 7 am to 10 pm. Before and after providing care, health professionals use the coworking space to study, contact other professionals, or wait for their patients to arrive. In this way, the administrator of the medical clinic hopes that the coworking environment will always be as pleasant as possible for health professionals. Temperature and humidity sensors generate data to support the management of this environment and ensure that it is comfortable. These sensors generate data about the environment every hour and notify the medical clinic coordinators. The coordinators are responsible for managing the temperature and humidity of the clinic.

The second scenario deals with professionals' essential use of consultation offices to provide care to their patients. Given that the medical clinic operates from 7 am to 10 pm, health professionals can attend to one or several patients. Whenever a health professional arrives at the clinic, he waits for his patient in the coworking space. The coworking presence sensor periodically sends whether an environment has people or not. Therefore, if a health professional arrives at the coworking space, the presence sensor sends an alert to the environment coordinators, warning them about using the area. When a patient is going to attend, the health professional leaves the coworking space and goes to the doctor's office designated for use. Upon arriving at the doctor's office, the tablet will have the health professional photo informing that the room is free. The tablet indicates the environment used when entering the doctor's office. Still, in the current scenario, a patient attending must notify the health professional of their arrival using the totem in the waiting room.

The totem consists of software in which patients search for their names and that they are in the waiting room. The use of the totem alerts the health professional that the

patient is already waiting in the room. A presence sensor in the waiting room identifies whether people are in that environment to support this alert. Health professionals and patients meet in offices. When the medical appointment finishes, a health professional clears the room, and the patient leaves. At that moment, the tablet in the doctor's office change the picture, the presence sensor of waiting room, and coworking space sent data on the release of the premises.

Although the first and second scenarios presented differ, both occur concurrently in the intelligent medical clinic. For this reason, the developed applications follow the same self-adaptive architecture.

Unlike the first cycle, the data used in this phase of the architecture was generated in real-time. The developed architecture was evaluated in real-time during application in the intelligent scenario. To this end, a one-week cycle was carried out to measure its operation not to harm employees' daily lives or the business environment's policies.

5.5.2 Conducting

Given the intelligent environment composed of sensors and IoT devices, they are responsible for generating data about what is happening in the environment. The clinic environment comprises patients, health professionals, sensors, devices, offices, waiting rooms, and coworking. The interactions between all these actors form the studied intelligent environment. The temperature, humidity, presence, camera, tablet, and totem sensors use edge computing techniques to perform the first data processing. The use of edge techniques allows the devices present there to carry out data processing with their computational capacity near the origin of the data. Edge usage is essential to help filter important data, assess the integrity of this data, and reduce the amount that must be stored. After processing the data initially at the edge, the data goes to the fog. The Fog structure is used in doctors' offices, waiting rooms, and medical coworking. Its use allows the data generated to be even more filtered, given the greater processing capacity of fog compared to the edge.

By following the path of the data, after passing through the Fog structure, it will be processed in the managed system. The managed system has a Serverless, Middleware, Storage, and Application component. The serverless component makes use of functions as a service. Functions as a service offers flexibility to configure and allow each IoT device's and sensor's needs according to their characteristics. The serverless paradigm also allows for scalability using computational resources in daily situations that generate high demand. To verify the scalability of computational resources, the clinic's day-to-day experimentation allowed the visualization of scenarios with the need to expand computational resources and others where resources were not required.

For the implementation of the architecture, we used some technologies. OpenFaas[10] was used as the serverless technology approach. OpenFaas can be used in different clouds and allows the construction and use of functions as a service in several programming languages. Given the need for IoT devices and sensors, in addition to the need for experimentation, functions as a service were written using Python[11]. The use of serverless with OpenFaas also made it possible to monitor the performance of functions as a service. Information such as response time, hour of use, number of requests, processing time, and memory usage are monitored and stored in a database. OpenFaas uses Prometheus[12] to manage and monitor performance metrics.

OpenFaas allows the migration of the architecture's serverless structure to any cloud. This feature is essential to ensure computational resources' flexibility, scalability, and portability. To use the functions as a service in OpenFaas, it is enough to access the function's source code and create a YAML file. YAML files establish guidelines for using a computational resource, such as port, amount of memory, and processing capacity. Based on these settings in the YAML file, containers use functions as a service. Using containers allows the portability of functions and their activation and deactivation when necessary. In a scenario of different sensors and IoT devices, such as the clinic, the characteristics of each of the sensors can be privileged and met via the source code of the functions and through the configuration file that establishes the use of computational resources.

Using OpenFaas makes using different programming languages possible, together with the concept of serverless and functions as a service. Creating a new function and configuration file is enough if new devices and sensors enter the clinic. This flexibility allows the expansion of the use of computational resources for any needs of the environment.

Still, in the managed system, the architecture makes use of middleware. The middleware is responsible for guaranteeing access, insertion, and query in a coordinated way to the data of the intelligent environment. To use the Middleware component, the architecture used Hasura. Hasura is a tool that offers integrations to different databases and allows the creation of data schemas. Different services, such as functions as a service, can consume their data and generate schemas. We integrated Hasura along with the Storage component. The storage component is responsible for storing databases. In the context of the medical clinic, we used a PostgreSQL[13] relational database to store scenario event data.

We developed the application component to display scenario data. The Thingsboard[14] was used to demonstrate the displayed data, generate alerts, and display scenarios

[10] https://www.openfaas.com/
[11] https://www.python.org/
[12] https://prometheus.io/
[13] https://www.postgresql.org/
[14] https://thingsboard.io/

for those interested. Thingsboard is a tool that integrates IoT devices and sensors and allows the construction of scenario visualizations. Another explored aspect of Thingsboard is that it allows the creation of visualization of joint entities of environments. For example, in the intelligent medical clinic environment, Thingsboard allows the visualization of data from a specific doctor's office and all rooms. This tool allows users to understand what is happening individually in an environment and the clinic. Thingsboard uses middleware to consume data and receive data needed for display.

The central element of the architecture is related to the management component. In addition to managing the health of the managed system, the management component is responsible for ensuring that sensors and IoT devices can use computational resources. The Manager Layer has a user who must configure the usage scenarios. To configure the scenarios, the user must use the Feedback Loop component. Given the two clinical usage scenarios, we registered the computational resources in the management system, namely the functions as a service and their respective configuration files. Each sensor has a relationship with a function as a service and its configuration in YAML files. We stored the relationship in a Postgres database. If we had to deal with a new sensor or IoT device, we should have created a new mapping in the database and its addition to a repository. Below, Figure 21 presents an example of a function as a service:

Figure 21 - Custom function as a service code

```
import requests
import json

def handle(req):

    requestObject = json.loads(req)
    baseUrl = 'http://host.docker.internal:8081/api/v1/'
    key = requestObject['key']
    finalUrl = '/telemetry'
    url = baseUrl+key+finalUrl

    del requestObject['key']

    #Define custom logical here to the function as a service.

    requests.post(url, json = requestObject)
```

Source: Elaborated by the author (2024).

The presented function as a service is a template for generating functions for new IoT devices and sensors. The exposed code points to loading, processing, and sending data for visualization.

The built function as a service has a mapping through a YAML file. Figure 22 below shows how a function as a service is configured and how the use of computational resources is established.

Figure 22 - YAML file

```yaml
version: 1.0
provider:
  name: openfaas
  gateway: http://127.0.0.1:8080
functions:
  coworking-function:
    lang: python
    handler: ./resources/coworking/coworking-function
    image: mateusgon/coworking-function:latest
    memory: 256MB
    cpu: 0.1
```

Source: Elaborated by the author (2024).

The defined YAML file has a service provider name, in this case, OpenFaas, and a gateway, which will establish the connection and activate it to receive requests. Furthermore, the function is defined. In this case, the function of the coworking space has the mapping presented. As shown in Figure 23, the language used was Python. The file path is found in the handler. An image is generated to be used by the gateway, and this image presents the compilable code for execution. Finally, the amount of memory is defined, as well as the amount of processing of a processor as well. The example memory was 256MB, and the processing power was 10

In addition to configuring computational resources by the management system, one of the critical factors is when we should have activated the computational resources. For this, data from the use of sensors and scenarios were essential. Based on the data stored from the execution of the scenarios in the managed system, it was possible to predict the hours of the day that the computational resources were used. With this data, we developed an artificial intelligence model using Python's SciKit Learn[15]. The model used in the SciKit Learn library was the Linear Support Vector Classifier (Linear SVC). It is a linear classification model which uses a linear function for classifications. One of its characteristics is that it allows the separation of data according to their characteristics based on a linear hyperplane. Artificial intelligence aims to establish whether a computational resource should be available during a specific time of day. To train the artificial intelligence model to construct the model, we used historical data from the execution of sensors and IoT devices.

The Feedback Loop is the component of the management system that queries the repository. The repository stores all information about computational resources. The computational resource analyzer component provides an interface to execute the model we developed. We created an API in Python to execute the prediction of our model. This API

[15] https://scikit-learn.org/stable/

receives Feedback Loop queries every 5 minutes for the artificial intelligence to establish whether a computational resource should be available for use or not. The Feedback Loop uses a Job Scheduler to run the computational resource analyzer component API.

The API calls the self-adaptative executor if a computational resource must be available or unavailable according to the time of day. An autonomous agent generates an event for the workflow engine by an API request. The API request into the workflow engine generates an event to the OpenFaas to activate or deactivate a computational resource.

Configurations based on YAML files that generate containers allow computing resources to be activated or deactivated at any time. Through the management system, the workflow engine alternates the characteristics of the managed system concerning computational resources. This approach allows activating or deactivating resources, allowing their use only when necessary, saving computational resources.

Executing the sensors and the proposed architecture in the corporate environment generated some data. A sample of this data, regarding execution over a few days, can be seen on GitHub[16]

5.5.3 Analyzing

Through case study 2, the self-adaptive architecture was developed and evaluated to understand how much it supports data processing and management of the physical environment through computational resources. Artificial intelligence is also presented alongside computational resources, and its impact is discussed in the results of the following subsection.

Given the exposed scenario of interactions in the corporate e-health environment, the objective of the second architecture was to support the use of computing resources in the intelligent e-health environment and improve the usability of applications. In this way, the data generated by the intelligent corporate e-health environment was used, and the built application was evaluated.

The results obtained through developing the architecture and monitoring of computational resources supported by artificial intelligence were satisfactory and are discussed in the following subsection.

5.5.4 Results

This subsection presents the last phase of the DSR method. It aims to discuss and evaluate the knowledge and results obtained in the previous phases of the study case.

[16] https://github.com/mateusgon/architecture/

Through the implementation of the architecture for the experimental scenario, we were able to obtain results and observations about the proposed architecture.

The scenario mapping was critical and conducted with professionals working in the e-health environment. These professionals helped to map when a computational resource was needed according to user interaction. Interactions by time of day were studied and passed to artificial intelligence based on historical data from the medical clinic. Based on the information provided by professionals and historical usage bases, we also established weights for when a computational resource should be used. Therefore, at times of greater use of the environments, the chance of a computational resource being available was greater than at other times. These weights were passed on to artificial intelligence training and helped predict whether a resource should be available or unavailable.

To store the interactions of the intelligent environment with sensors and devices, the approach used with a relational database and resource mapping in Yaml files was effective and satisfactory. When it was predicted by artificial intelligence that a resource should be available or should be unavailable, information about the computational resource was easily accessed in the files that configured its use. In this way, computing resources were quickly made available in the cloud, and the cloud could quickly scale its use.

The flexibility provided by serverless allowed computational resources to be easily made available and unavailable to use sensors and devices. With this flexibility, the architecture could scale resource usage and allow for interoperability. One of the most apparent benefits of the serverless approach was allowing computing resources to be deployed quickly and not needing a server to start operating.

The use of middleware made it possible to have clear policies for processing and accessing data by computational resources. The mapping of what was necessary for the computational resource and the middleware allowed the entire structure of reading, writing, updating, and data privacy to be preserved. This way, we could guarantee data security and establish clear access policies.

The main objective of the implemented architecture was to ensure that the necessary computational resources were adapted according to the time of day and the characteristics of an intelligent environment. In addition, the architecture should guarantee that it would be possible to address aspects of scalability, interoperability, and adequate access to the database, which are challenges addressed and resolved in other cycles.

One of the most significant challenges was the self-adaptation of computational resources to use sensors in clinical medicine. To make self-adaptation possible, we had to select and train a machine-learning technique with historical data from sensors. This training involved breaking down historical running data from sensors and devices by time and day of the week.

When evaluating whether a computational resource should be available or disabled,

we found the challenge that the predicted artificial intelligence did not always have assertiveness. At times when the resource should be available, artificial intelligence predicts that it should not. The opposite scenario also occurred. In Figure 23 when it was true, it was available and should be. When it was false, it was not available, and it should have been.

Figure 23 - Availability presence detector computational resource

Hour	Monday	Tuesday	Wednesday	Thursday	Friday
7AM	TRUE	TRUE	TRUE	TRUE	FALSE
8AM	FALSE	TRUE	FALSE	TRUE	TRUE
9AM	TRUE	TRUE	TRUE	TRUE	TRUE
10AM	TRUE	TRUE	TRUE	TRUE	TRUE
11AM	TRUE	TRUE	TRUE	TRUE	FALSE
0PM	FALSE	TRUE	TRUE	FALSE	FALSE
1PM	FALSE	TRUE	TRUE	FALSE	TRUE
2PM	TRUE	TRUE	TRUE	TRUE	TRUE
3PM	TRUE	TRUE	TRUE	TRUE	TRUE
4PM	TRUE	FALSE	TRUE	TRUE	TRUE
5PM	TRUE	TRUE	TRUE	TRUE	TRUE

Source: Elaborated by the author (2024).

While mispredicting usage causes some data not to be obtained, this has not impeded the testing of the intelligent e-health environment. Even though evaluating possible artificial intelligence or the one that would best fit was not the focus of our work, it was necessary to choose a model. For it, data was generated on its effectiveness. As a result of our experiment, we obtained some satisfactory data about the artificial intelligence model. They are precision, recall, and F1-Score. Precision was about 70.9%, the recall was 61.1%, and F1-Score was 65.6. As a significant benefit, this approach made it possible to guarantee support for the clinic's day-to-day activities. One of the ways to evaluate the impacts of predicting the use of computational resources was by talking to the workers who used our developed application. The reports of these workers showed that when there was no data, and data was necessary, the workers made themselves an inspection of the environment.

Although it is a problem that the computational resource is not available at certain times of the day, the impact caused could have been more relevant, according to the reports of workers in the corporate environment. In this way, exploring other machine learning and artificial intelligence methods can support the resolution of the problem.

Another concern was ensuring that the usage scenarios were mapped and, based on them, the computational resources were also activated and deactivated. This way, the

humidity and temperature sensors were the only resources needed between 10 pm and 7 am. When evaluating the data of when the computational resources were available during the period above, it was seen that only one day, the computational resource for the tablet was activated at an inappropriate time. This approach demonstrated a reduction in the use of computational resources when interactions were not taking place.

Scalability and interoperability occurred at times when artificial intelligence predicted that it would be necessary to use several computational resources together. This prediction caused computational resources to adapt to use. Examples were seen during clinic day when many patients walked through the environment. In addition to the need to scale computational resources, aspects such as interoperability became evident when scenarios using several sensors occurred—computing resources needed to interoperate to ensure data was effectively collected from the environment.

Following the presented implementation of the architecture, we will develop two applications. Each application reflects the scenarios for using the environment explained above. The coordinators of the medical clinic used both applications. Figure 24 shows the results of building the support application for coordinating the environment.

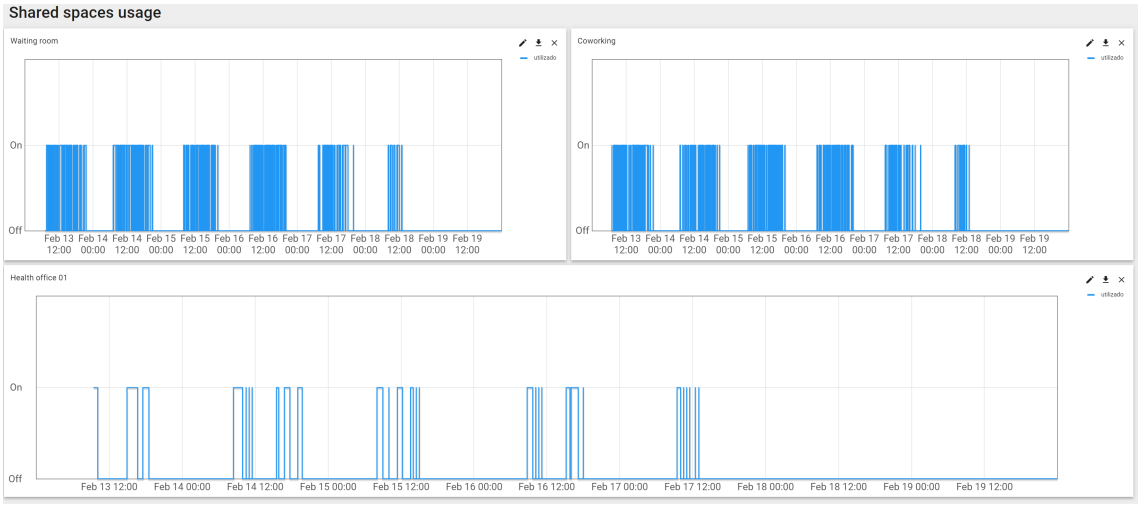
Figure 24 - Developed application for the first scenario



Source: Elaborated by the author (2024).

The second application was developed using the self-adaptive architecture to help clinic management in a scenario rich in interactions. Figure 25 demonstrates the application developed for the scenario.

Figure 25 - Developed application for the second scenario



Source: Elaborated by the author (2024).

The end user's medical clinic use was another critical factor in evaluating the proposed architecture. Three employees from the corporate environment were interviewed to evaluate the application developed. Two employees are clinic coordinators, and one of the interviewees is a manager. The interviews took place while the application was running in the corporate environment. Employees had a form to evaluate the operation, which was answered at the end of each day of execution of the application. These results were condensed and presented below at the end of the tests. The users reported that the data necessary for managing the intelligent environment, for the most part, were available for decision-making. Users also reported that when data was not available, manual verification of environments was required. Ultimately, the environment managers reported satisfactory experience with the developed application.

Through the proposed architecture for developing IoT applications in e-health environments, we achieved the objective of improving the management of the use of computational resources and allowing data collection from the intelligent e-health environment and its management. This was demonstrated through the discussion, implementation, and results of the implementation of the artifact, as the results obtained are qualified as satisfactory. The self-adaptation of computational resources brought great benefits to the environment, allowing better management of resources and ensuring that they will only be used when needed. This approach can allow the construction of IoT applications to be increasingly focused on the complex interactions of existing environments and establish clear usage rules for computational resources. This approach allows, for example, financial savings and even electrical consumption in case these computational resources are not used when not necessary.

5.6 CONCLUSIONS FROM CASE STUDIES

The research objective was addressed through the case studies carried out in the first and second cycles of DSR, as well as the discussions on the architecture implementation presented in this chapter. In this way, the architectures presented demonstrated how an intelligent space could benefit from an approach that uses computational resources and is supported by artificial intelligence. Different quality attributes and functional and non-functional requirements were addressed to demonstrate this support. Finally, the solution developed supported the management of the intelligent physical space presented.

In the first cycle of DSR, the architecture was built to support the intelligent medical clinic. The initial objective was to support space management and understand how an architecture could support receiving data from sensors and devices present there. Furthermore, the complex interactions of these environments were mapped. The Functional Requirements addressed included integration and communication with IoT devices and sensors (FR01), data storage in relational and non-relational databases (FR02), and visualization of generated data (FR03). Furthermore, the architecture used cloud, fog, and edge resources for close-to-source processing and data filtering (FR06). Regarding Non-Functional Requirements (NFRs), the architecture met data security and privacy (NFR04), scalability and processing efficiency (NFR02 and NFR03), and interoperability (NFR09).

At the end of the first DSR cycle, improvement opportunities were identified. These were related to the more efficient use of computing resources and the preparation of computing resources for problems in physical space, for example, lack of electricity or problems generating data due to intermittent or interrupted computer networks (NFR05 and NFR06). In addition to these points highlighted, improvements in the usability of the data visualization tool were necessary, offering support for historical data. The use of computational resources could have been optimized. Finally, another relevant factor was the architecture's ability to adapt to events and the needs of the physical environment (FR04, FR05, NFR07, and NFR08). Thus, the need for a new DSR cycle arose.

The second case study presented the development of a self-adaptive architecture to support using computational resources in the corporate e-health environment. To meet the established functional and non-functional requirements, the architecture focused mainly on supporting the management of computing resources. The functional requirements addressed dealt with adaptation to the needs of the intelligent environment and the use of artificial intelligence to support computational resources (FR04 and FR05). Non-functional requirements addressed aspects related to quality attributes, such as, for example, scalability and the introduction of new IoT devices and sensors. Furthermore, aspects related to computational resources and the use of AI were presented to deal with NFRs 06 and 07. The architecture was implemented using technologies such as OpenFaas for

serverless computing, Hasura for middleware, and PostgreSQL for data storage. Analysis of the results showed that the architecture allowed a compelling adaptation of computing resources according to the use of the environment, supported by artificial intelligence. However, it faced challenges in accurately predicting the availability of resources.

The second case study also presented the importance of environmental usage scenarios and the importance of mapping the devices and sensors used; only in this way could self-adaptive architecture be built. Although there have been challenges in the accuracy of resource availability predictions, the architecture has proven effective in managing computational resources, ensuring their availability only when necessary. Furthermore, the applications developed based on the architecture were well received by users, demonstrating their usefulness in managing the corporate e-health environment. This approach provided a solid foundation for the future development of IoT applications in complex e-health environments.

Thus, with the research question "How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?"with the conjecture "Suppose we propose a self-adaptive architecture for developing IoT applications in e-health. In this case, can we improve the use of computational resources by systems, supporting data collection from intelligent e-health environments and supporting the management of these spaces?"based on the results of the case studies, it was possible to conclude that the self-adaptive architecture can support automated computational resource allocation in e-health environments through its ability to adjust resource provisioning based on changing conditions and requirements dynamically. In an e-health environment scenario, where the demands for computing resources are varied and depend on several environmental factors, such as criticality for patients and health professionals and which area of health is affected, the self-adaptive architecture presented several benefits, such as those listed below.

Large-scale data processing: The architecture enabled large-scale data processing through a function-as-a-service approach for each IoT device and sensor. This approach made it possible to isolate the computational resource needs of each sensor and device, resulting in processing according to the characteristics of each IoT item.

Choice of computing resources according to the demands of the environment: Through the integration of artificial intelligence and machine learning techniques, the self-adaptive architecture can adapt its computing resources according to the needs of the corporate environment used. This resulted in improved energy costs, reduced waste, and more efficient use of resources for data processing.

Real-time monitoring of the environment: The self-adaptive architecture allowed continuous monitoring of everything happening in the environment. In this way, interactions, data generation, and events were generated and used by workers to manage the physical space in the corporate e-health environment. Real-time analysis of

environmental data also allowed the training of artificial intelligence and the development of the self-adaptation technique.

Data protection and privacy: Data protection and privacy were guaranteed through the use of the middleware present in the architecture and the establishment of the data access policy. This approach mitigated potential risks related to security attacks and data theft.

Automated workflow management: By automating the allocation of computing resources, it was possible to guarantee the application's scalability and ensure that the entry of new IoT devices and sensors does not cause problems. In addition to this benefit, the automatic scaling of computational resources also has advantages, as it avoids human work and allows the application to self-manage.

Scalability and reliability: The proposed self-adaptive architecture allowed scalability using computational resources and integrating new IoT devices and sensors. Furthermore, the architectural approach allowed alerts to be generated in applications if problems and recovery policies exist, allowing for increased application reliability.

In short, the self-adaptive architecture for intelligent e-health environments provided a development model to support allocating computational resources in an efficient, adaptive, and responsive way to the needs of the environment in which it is inserted. Furthermore, by leveraging adaptive capabilities and advanced analytics, such an architecture can drive improvements in patient care, operational efficiency, and resource optimization in e-health facilities.

Although many benefits and advantages have been observed, there are opportunities for improvements and future work; examples are:

Improving the accuracy of the artificial intelligence model and adopting new model training policies. These improvements can increase efficiency in managing computing resources and intelligent physical spaces. For this reason, additional investigations into the AI model can be adopted.

Exploration of other functional and non-functional requirements. The work focused on functional and non-functional requirements by those found in the literature through systematic mapping and exploratory study. Therefore, other attributes may be necessary in other approaches to e-health.

Improvements in the way non-functional requirements were handled. Individual studies for each functional requirement can be proposed and explored to identify other gaps in the literature and lead to the execution of new studies.

Evaluation in other e-health environments. The architecture was validated in a specific e-health environment. Other e-health environments can be evaluated, and new functional and non-functional requirements may appear.

Assessment in other intelligent environments: Besides e-health environments, other intelligent environments can use the architecture and understand their needs according to the environment. This future work will be able to evaluate the suitability of the architecture for different intelligent scenarios.

5.7 THREATS TO VALILIDTY

In this subsection, we discuss threats to validity that may limit or affect the results obtained.

Internal validation: During the case study, data was generated by specific sensors in the corporate environment. Although the initial results are valuable, a more detailed study in other intelligent corporate e-health environments is necessary to identify other learnings and possible improvements. Adding new sensors and devices and competition for network resources can affect the data generated and obtained.

External validation: In a corporate scenario with an even more significant number of sensors, new techniques to predict the need to use computational resources will be necessary. The context, sensors, devices, number of rooms, and shared spaces in which the work was developed will directly influence how computational resources should be used. New e-health environments can generate different results using computational resources and data generation, bringing new learning and discoveries.

Construct validation: Different artificial intelligence models were not tested for computational resource selection during the case study. The use of other artificial intelligence can generate different results. Given the construction of the architecture and the division into modules, new artificial intelligence techniques can be used. New studies can be carried out to improve the results obtained with artificial intelligence, and researchers can compare results.

Validity of selection: During the study period, the selected corporate environment had similar people dynamics and use of computing resources. Using computational resources, people moving through the environment can generate new results on days with different behaviors. Therefore, validating the architecture on specific operating days may be necessary.

5.8 FINAL CONSIDERATIONS OF THE CHAPTER

This section presented how the proposed architecture was tested, generated results, evaluated, and constructed knowledge according to the step-by-step process proposed by the design science research methodology. The results obtained from the experiment were satisfactory, and the objectives of dealing with computational resources were achieved, ensuring that resources were not wasted unnecessarily. This was possible given the use of

artificial intelligence and a self-adaptive architecture, which provided an understanding of the developed scenario and necessary adaptations to the dynamics of the environment.

6 CONCLUSIONS

The expansion of the Internet of Things and intelligent environments has led to new opportunities for Software Engineering. Intelligent e-health environments address different areas of society that seek to automate or make life more accessible through technologies. Linked to the use of new technologies and the opportunities, challenges, and problems that have arisen for Software Engineering, the use of machine learning techniques and the development of computing resource technologies, such as cloud, edge, and fog, have allowed the development of a new range of applications.

This work presented a set of concepts related to IoT, intelligent environments, smart living, e-health, computational resources, and artificial intelligence. Through the presentation of these concepts, a systematic mapping of the literature was conducted to identify improvements, challenges, and opportunities. The challenges showed that computational resources, large-scale data processing, quality attributes, and artificial intelligence could be used in IoT architectures in intelligent e-health environments. Based on the systematic mapping and opportunities found, the following research question was presented: "How have computational resources been used in intelligent e-health environments that use IoT to develop architectures?". Next, a conjecture was raised about the research question. Suppose we propose a self-adaptive architecture for developing IoT applications in e-health. In this case, can we improve the use of computational resources by systems, supporting data collection from intelligent e-health environments and supporting the management of these spaces?

An exploratory study was conducted to answer the research question and evaluate the established conjecture. During the exploratory study, an IoT architecture was developed to support the management of the spread of the Covid-19 disease and demonstrate and gain knowledge about how IoT architectures work. Based on the exploratory study results, the hypotheses raised were validated, presenting new challenges and complexities and reinforcing the research question and the established conjecture.

Considering the knowledge obtained in the systematic mapping and exploratory study, the work focused on intelligent e-health environments. IoT architectures for intelligent e-health environments were proposed to answer the research question and the established conjecture. The first architecture did not deal with aspects directly related to computational resources and artificial intelligence, unlike the second architecture, which supported these themes. Finally, both architectures addressed quality attributes and functional and non-functional requirements.

Both architectures were validated in an intelligent corporate e-health environment using the DSR methodology to evaluate the proposed architectures. This environment had interaction dynamics between sensors, devices, people, and generated data. Exploring

this data generated through architecture allowed the development of an application made available to employees in an intelligent environment and supported them in day-to-day tasks.

As contributions to this work, the exploratory study of the topic was published (Do Nascimento et al., 2021), and the first version of the architecture was also published (Do Nascimento et al., 2023). Other related published works (Do Nascimento et al., 2020) and (Do Nascimento et al., 2020) In addition to these contributions to the literature, other contributions can be seen in:

- Systematic mapping of the literature on how IoT architectures have been used with e-health. This systematic mapping contributes to the literature by demonstrating state-of-the-art opportunities, challenges, and problems encountered in the research area.
- Development of three architectures for the development of IoT applications. These architectures contribute to the scientific community through the knowledge generated and providing solutions that can be employed.
- Presentation of the use of computational resources and approaches to improve efficiency. This contribution allows computing resources to be used more efficiently, reducing cost, using them on demand, and allowing unused resources to be available to other devices needing them.
- Development of an artificial intelligence model to assist the use of computational resources by application architectures for IoT and e-health. This contribution highlights this technology's advantages, problems, and challenges.
- Exploration of the quality attributes of the architecture, such as, for example, performance and scalability. As a contribution to the literature, the architectures presented demonstrate how to deal with these topics and present techniques for using them.

In future works, evaluating the architecture in environments other than e-health is essential to understanding adaptations that may be necessary. The main focus of the work was not on which artificial intelligence to choose but on validating the functioning of the architecture with artificial intelligence, evaluating other artificial intelligence to predict situations with more scenarios, and ensuring that resources are available in the most crucial moments. In the future, an autoML approach can be studied to deal with better scenario adaptations. Finally, evaluating how self-adaptive architecture can help to reduce energy costs and how much it supports green computing.

Furthermore, as explained in the previous chapter, other future work may address improving the accuracy of the artificial intelligence model and adopting new model

training policies, exploration of other functional and non-functional requirements, improvements in the way non-functional requirements were handled, evaluation in other e-health environments, assessment in other intelligent environments.

REFERÊNCIAS

AAZAM, Mohammad et al. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In: Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014. IEEE, 2014. p. 414-419.

AHMED, Ejaz et al. Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges. IEEE Wireless Communications, v. 23, n. 5, p. 10-16, 2016.

ALBINO, Vito; BERARDI, Umberto; DANGELICO, Rosa Maria. Smart cities: Definitions, dimensions, performance, and initiatives. Journal of urban technology, v. 22, n. 1, p. 3-21, 2015.

ALFONSO, Iván et al. Self-adaptive architectures in IoT systems: a systematic literature review. Journal of Internet Services and Applications, v. 12, n. 1, p. 1-28, 2021.

ALLAM, Zaheer; DHUNNY, Zaynah A. On big data, artificial intelligence and smart cities. Cities, v. 89, p. 80-91, 2019.

AMERSHI, Saleema et al. Software engineering for machine learning: A case study. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2019. p. 291-300.

ARUL, Rajakumar et al. Multi-modal secure healthcare data dissemination framework using blockchain in IoMT. Personal and Ubiquitous Computing, p. 1-13, 2021.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. Ad Hoc Networks, v. 56, p. 122-140, 2017.

BEN HASSEN, Hafedh; DGHAIS, Wael; HAMDI, Belgacem. An E-health system for monitoring elderly health based on Internet of Things and Fog computing. Health information science and systems, v. 7, p. 1-9, 2019.

BERMBACH, David et al. A research perspective on fog computing. In: Service-Oriented Computing–ICSOC 2017 Workshops: ASOCA, ISyCC, WESOACS, and Satellite Events, Málaga, Spain, November 13–16, 2017, Revised Selected Papers. Springer International Publishing, 2018. p. 198-210.

BROUS, Paul; JANSSEN, Marijn; HERDER, Paulien. The dual effects of the Internet of Things (IoT): A systematic review of the benefits and risks of IoT adoption by organizations. International Journal of Information Management, v. 51, p. 101952, 2020.

BUDGEN, David et al. Using Mapping Studies in Software Engineering. In: Ppig. 2008. p. 195-204.

CASSEL, Gustavo André Setti et al. Serverless computing for Internet of Things: A systematic literature review. Future Generation Computer Systems, v. 128, p. 299-316, 2022.

CHANG, Zheng et al. Dynamic resource allocation and computation offloading for IoT fog computing system. IEEE Transactions on Industrial Informatics, v. 17, n. 5, p. 3348-3357, 2020.

CHEN, Ying et al. Survey of cross-technology communication for IoT heterogeneous devices. IET Communications, v. 13, n. 12, p. 1709-1720, 2019.

COSTA, Felipe S.; NASSAR, Silvia M.; DANTAS, Mario AR. A Sensor Ranking Approach for Edge–Fog–Cloud Environments. SN Computer Science, v. 4, n. 4, p. 392, 2023.

DA SILVA, Madalena Pereira; GONÇALVES, Alexandre Leopoldo; DANTAS, Mário Antônio Ribeiro. A conceptual model for quality of experience management to provide context-aware eHealth services. Future Generation Computer Systems, v. 101, p. 1041-1061, 2019.

DAI, Shijie et al. An adaptive computation offloading mechanism for mobile health applications. IEEE Transactions on Vehicular Technology, v. 69, n. 1, p. 998-1007, 2019.

DE MICHELE, Roberta; FURINI, Marco. Iot healthcare: Benefits, issues and challenges. In: Proceedings of the 5th EAI international conference on smart objects and technologies for social good. 2019. p. 160-164.

DHASARATHAN, Chandramohan et al. A nomadic multi-agent based privacy metrics for e-health care: a deep learning approach. Multimedia Tools and Applications, p. 1-24, 2023.

DO NASCIMENTO, Mateus Gonçalo et al. An Architecture to Support the Development of Collaborative Systems in IoT Context. In: 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2023. p. 1722-1727.

DO NASCIMENTO, Mateus Gonçalo et al. Towards an IoT architecture to pervasive environments through design science. In: International Conference on Advanced Information Networking and Applications. Cham: Springer International Publishing, 2021. p. 28-39.

DO NASCIMENTO, Mateus Gonçalo et al. Covid-19: A Simulation-Based Architecture Proposal for IoT Application Development. In: 2020 International Conference on High Performance Computing & Simulation (HPCS).

DO NASCIMENTO, Mateus Gonçalo et al. Covid-19: A digital transformation approach to a public primary healthcare environment. In: 2020 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2020. p. 1-6.

EYSENBACH, Gunther et al. What is e-health?. Journal of medical Internet research, v. 3, n. 2, p. e833, 2001.

FETAHU, Lorik; MARAJ, Arianit; HAVOLLI, Abdullah. Internet of Things (IoT) benefits, future perspective, and implementation challenges. In: 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE, 2022. p. 399-404.

GARCES, Lina et al. Architectural solutions for self-adaptive systems. Computer, v. 53, n. 12, p. 47-59, 2020.

GIA, Tuan Nguyen et al. Energy efficient fog-assisted IoT system for monitoring diabetic patients with cardiovascular disease. Future Generation Computer Systems, v. 93, p. 198-211, 2019.

GOMES, Jonas et al. A scientific software ecosystem architecture for the livestock domain. Information and Software Technology, v. 160, p. 107240, 2023.

GOMEZ, Carles et al. Internet of Things for enabling smart environments: A technology-centric perspective. Journal of Ambient Intelligence and Smart Environments, v. 11, n. 1, p. 23-43, 2019.

GHORBANI, Fatemeh et al. A decision-aware ambient assisted living system with IoT embedded device for in-home monitoring of older adults. Sensors, v. 23, n. 5, p. 2673, 2023.

GUBBI, Jayavardhana et al. Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, v. 29, n. 7, p. 1645-1660, 2013.

GUPTA, Akash; AL-NAIME, Khalid; AL-ANBUKY, Adnan. Iot environment for monitoring human movements: Hip fracture rehabilitation case. In: Information and Communication Technologies for Ageing Well and e-Health: 6th International Conference, ICT4AWE 2020, Prague, Czech Republic, May 3–5, 2020, Revised Selected Papers 6. Springer International Publishing, 2021. p. 44-63.

HAJJAJI, Yosra et al. Big data and IoT-based applications in smart environments: A systematic review. Computer Science Review, v. 39, p. 100318, 2021.

HAN, Min Jee Nikki; KIM, Mi Jeong. A critical review of the smart city in relation to citizen adoption towards sustainable smart living. Habitat International, v. 108, p. 102312, 2021.

HASSIJA, Vikas et al. A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access, v. 7, p. 82721-82743, 2019.

HEVNER, Alan et al. Design science research in information systems. Design research in information systems: theory and practice, p. 9-22, 2010.

HEVNER, Alan R. A three cycle view of design science research. Scandinavian journal of information systems, v. 19, n. 2, p. 4, 2007.

HOSSEINIAN-FAR, Amin; RAMACHANDRAN, Muthu; SLACK, Charlotte Lilly. Emerging trends in cloud computing, big data, fog computing, IoT and smart living. Technology for smart futures, p. 29-40, 2018.

KARMAKER, Shubhra Kanti et al. Automl to date and beyond: Challenges and opportunities. ACM Computing Surveys (CSUR), v. 54, n. 8, p. 1-36, 2021.

KASHANI, Mostafa Haghi et al. A systematic review of IoT in healthcare: Applications, techniques, and trends. Journal of Network and Computer Applications, v. 192, p. 103164, 2021.

KAUR, Pavleen; KUMAR, Ravinder; KUMAR, Munish. A healthcare monitoring system using random forest and internet of things (IoT). Multimedia Tools and Applications, v. 78, p. 19905-19916, 2019.

KHAN, Huma H. et al. Challenges for sustainable smart city development: A conceptual framework. Sustainable Development, v. 28, n. 5, p. 1507-1518, 2020.

KITCHENHAM, Barbara and Stuart M. Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

KISHOR, Amit; CHAKRABORTY, Chinmay. Artificial intelligence and internet of things based healthcare 4.0 monitoring system. Wireless personal communications, v. 127, n. 2, p. 1615-1631, 2022.

KOK, Joost N. et al. Artificial intelligence: definition, trends, techniques, and cases. Artificial intelligence, v. 1, p. 270-299, 2009.

KUMAR, Mohit et al. Autonomic edge cloud assisted framework for heart disease prediction using RF-LRG algorithm. Multimedia Tools and Applications, p. 1-25, 2023.

KUMAR, T. V. Smart living for smart cities. Smart living for smart cities, p. 3-70, 2020.

LAHIANI, Nesrine et al. Self-Adaptive Healthcare Platforms Development: A Dynamic Product Line Approach Driven by Ontologies. In: 2023 International Conference on Decision Aid Sciences and Applications (DASA). IEEE, 2023. p. 354-359.

LARCHER, Lucas et al. Event-driven framework for detecting unusual patterns in AAL environments. In: 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS). IEEE, 2020. p. 309-314.

LAROUI, Mohammed et al. Edge and fog computing for IoT: A survey on current research activities & future directions. Computer Communications, v. 180, p. 210-231, 2021.

LARRUCEA, Xabier et al. Software engineering for the internet of things. IEEE Software, v. 34, n. 1, p. 24-28, 2017.

LI, He; OTA, Kaoru; DONG, Mianxiong. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, v. 32, n. 1, p. 96-101, 2018.

LIU, Yunpeng; XIAO, Fei. Intelligent monitoring system of residential environment based on cloud computing and internet of things. IEEE Access, v. 9, p. 58378-58389, 2021.

MAHMUD, Redowan; KOCH, Fernando Luiz; BUYYA, Rajkumar. Cloud-fog interoperability in IoT-enabled healthcare solutions. In: Proceedings of the 19th international conference on distributed computing and networking. 2018. p. 1-10.

MAKSIMOVIĆ, Mirjana; VUJOVIĆ, Vladimir. Internet of things based e-health systems: ideas, expectations and concerns. Handbook of large-scale distributed computing in smart healthcare, p. 241-280, 2017.

MALLIGA, S.; KOGILAVANI, S. V.; NANDHINI, P. S. A comprehensive review of applications of Internet of Things for COVID-19 pandemic. In: IOP conference series: materials science and engineering. IOP Publishing, 2021. p. 012083.

MELINE, Timothy. Selecting studies for systemic review: Inclusion and exclusion criteria. Contemporary issues in communication science and disorders, v. 33, n. Spring, p. 21-27, 2006.

MOTTA, Rebeca C.; DE OLIVEIRA, Káthia M.; TRAVASSOS, Guilherme H. On challenges in engineering IoT software systems. In: Proceedings of the XXXII Brazilian symposium on software engineering. 2018. p. 42-51.

MUCCINI, Henry et al. Self-adaptive IoT architectures: An emergency handling case study. In: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings. 2018. p. 1-6.

MUTHU, BalaAnand et al. IOT based wearable sensor for diseases prediction and symptom analysis in healthcare sector. Peer-to-peer networking and applications, v. 13, p. 2123-2134, 2020.

NEVES, Felipe et al. Heath-PRIOR: an intelligent ensemble architecture to identify risk cases in healthcare. IEEE Access, v. 8, p. 217150-217168, 2020.

NIŽETIĆ, Sandro et al. Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. Journal of cleaner production, v. 274, p. 122877, 2020.

ORTIZ, Guadalupe et al. Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing. Computer Standards & Interfaces, v. 79, p. 103550, 2022.

PARK, D. S. Future computing with IoT and cloud computing. the Journal of Supercomputing, v. 74, p. 6401-6407, 2018.

PATEL, Abhishek et al. Review of artificial intelligence and internet of things technologies in land and water management research during 1991–2021: A bibliometric analysis. Engineering Applications of Artificial Intelligence, v. 123, p. 106335, 2023.

PATHAN, Al-Sakib Khan et al. Internet of Things for smart living. Wireless Networks, v. 27, p. 4293-4295, 2021.

PEFFERS, Ken et al. A design science research methodology for information systems research. Journal of management information systems, v. 24, n. 3, p. 45-77, 2007.

PIMENTEL, Mariano; FILIPPO, Denise; DOS SANTOS, Thiago Marcondes. Design Science Research: pesquisa científica atrelada ao design de artefatos. RE@ D-Revista de Educação a Distância e eLearning, v. 3, n. 1, p. 37-61, 2020.

PRADHAN, Madhvi A. et al. IoT for smart city: Improvising smart environment. In: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). IEEE, 2017. p. 2003-2006.

RAHMAN, Mohamed Abdur et al. Secure and provenance enhanced internet of health things framework: A blockchain managed federated learning approach. Ieee Access, v. 8, p. 205071-205087, 2020.

RAHMANI, Amir M. et al. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. Future Generation Computer Systems, v. 78, p. 641-658, 2018.

RAJAVEL, Rajkumar et al. IoT-based smart healthcare video surveillance system using edge computing. Journal of ambient intelligence and humanized computing, p. 1-13, 2022.

RAY, Partha Pratim. A survey on Internet of Things architectures. Journal of King Saud University-Computer and Information Sciences, v. 30, n. 3, p. 291-319, 2018.

RAZZAQ, Abdul. A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture. SN Computer Science, v. 1, n. 6, p. 350, 2020.

RICHARDS, Mark; FORD, Neal. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020.

RODRIGUES, Laécio et al. Performance and availability evaluation of an smart hospital architecture. Computing, v. 103, p. 2401-2435, 2021.

SAID, Abd Mlak et al. Machine learning based rank attack detection for smart hospital infrastructure. In: The Impact of Digital Technologies on Public Health in Developed and Developing Countries: 18th International Conference, ICOST 2020, Hammamet, Tunisia, June 24–26, 2020, Proceedings 18. Springer International Publishing, 2020. p. 28-40.

SAMIZADEH NIKOUI, Tina et al. Internet of Things architecture challenges: A systematic review. International Journal of Communication Systems, v. 34, n. 4, p. e4678, 2021.

SANFILIPPO, Filippo et al. Importance of inclusion criteria in systematic reviews. British journal of anaesthesia, v. 125, n. 5, p. e398-e399, 2020.

SELEM, Enas; FATEHY, Mohammed; ABD EL-KADER, Sherine M. E-Health applications over 5G networks: Challenges and state of the art. In: 2019 6th International Conference on Advanced Control Circuits and Systems (ACCS) & 2019 5th International Conference on New Paradigms in Electronics & information Technology (PEIT). IEEE, 2019. p. 111-118.

SEMWAL, Neelkamal et al. An IoT based smart e-health care system. Journal of Information and Optimization Sciences, v. 40, n. 8, p. 1787-1800, 2019.

SHANDILYA, Shishir K.; CHUN, Soon Ae; SHANDILYA, Smita (Ed.). Internet of Things Security: Fundamentals, Techniques and Applications. CRC Press, 2022.

SIMON, Herbert A. The sciences of the artificial. MIT press, 1996.

SOVACOOL, Benjamin K.; DEL RIO, Dylan D. Furszyfer. Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies. Renewable and sustainable energy reviews, v. 120, p. 109663, 2020.

SPINELLIS, Diomidis. Software-engineering the Internet of Things. IEEE Software, v. 34, n. 1, p. 4-6, 2017.

TALAAT, Fatma M. Effective prediction and resource allocation method (EPRAM) in fog computing environment for smart healthcare system. Multimedia Tools and Applications, v. 81, n. 6, p. 8235-8258, 2022.

TOMA, Cristian et al. IoT solution for smart cities' pollution monitoring and the security challenges. Sensors, v. 19, n. 15, p. 3401, 2019.

TULI, Shreshth et al. HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. Future Generation Computer Systems, v. 104, p. 187-200, 2020.

VARGHESE, Blesson et al. Challenges and opportunities in edge computing. In: 2016 IEEE international conference on smart cloud (SmartCloud). IEEE, 2016. p. 20-26.

VERMA, Ankit; AGARWAL, Gaurav; GUPTA, Amit Kumar. A novel generalized fuzzy intelligence-based ant lion optimization for internet of things based disease prediction and diagnosis. Cluster Computing, v. 25, n. 5, p. 3283-3298, 2022.

VERMA, Prabal; SOOD, Sandeep K. Cloud-centric IoT based disease diagnosis healthcare framework. Journal of Parallel and Distributed Computing, v. 116, p. 27-38, 2018.

VORA, Jayneel et al. FAAL: Fog computing-based patient monitoring system for ambient assisted living. In: 2017 IEEE 19th international conference on e-health networking, applications and services (Healthcom). IEEE, 2017. p. 1-6.

WANG, Lizhe et al. Cloud computing: a perspective study. New generation computing, v. 28, p. 137-146, 2010.

WANG, Pei. On defining artificial intelligence. Journal of Artificial General Intelligence, v. 10, n. 2, p. 1-37, 2019.

WEYNS, Danny. An introduction to self-adaptive systems: A contemporary software engineering perspective. John Wiley & Sons, 2020.

WIERINGA, Roel. Design science as nested problem solving. In: Proceedings of the 4th international conference on design science research in information systems and technology. 2009. p. 1-12.

WILSON, Jessica et al. Barriers and facilitators to the use of e-health by older adults: a scoping review. BMC public health, v. 21, p. 1-12, 2021.

YI, Shanhe; LI, Cheng; LI, Qun. A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 workshop on mobile big data. 2015. p. 37-42.

ZADTOOTAGHAJ, Parang et al. Internet of Things: A Survey for the Individuals' E-Health Applications. Journal of Information Technology Management, v. 11, n. 1, p. 102-129, 2019.

APPENDIX A – Journalistic publications about the exploratory study

Projeto: Simula Covid-19 UFJF. Disponível em: https://www2.ufjf.br/pgcc/curso/projetos/simulacovid19ufjf/

Simulações computacionais demonstram impactos da Covid-19 em campus da UFJF. Disponível em: https://www2.ufjf.br/noticias/2021/08/18/simulacoes-computacionais-demonstram-impactos-da-covid-19-em-campus-da-ufjf/

Simula Covid-19 | UFJF. Disponível em: https://www.youtube.com/watch?v=M00YCpWMvvI

Covid-19: através de simulação, pesquisadores da UFJF demonstram impactos da doença no campus. Disponível em: https://g1.globo.com/mg/zona-da-mata/noticia/2021/08/19/covid-19-atraves-de-simulacao-pesquisadores-da-ufjf-demonstram-impactos-da-doenca-no-campus.ghtml

Estudo da UFJF simula cenários de uma nova onda da Covid-19. Disponível em https://tribunademinas.com.br/noticias/cidade/11-11-2021/estudo-da-ufjf-simula-cenarios-de-uma-nova-onda-da-covid-19.html