SEVEN
PUBLICAÇÕES ACADÊMICAS
2024

ARCHITECTING IOT FOR HEALTHCARE

KEY CONSIDERATIONS

Itamir de Morais Barroca Filho

SEVEN
PUBLICAÇÕES ACADÊMICAS
2024

ARCHITECTING IOT FOR HEALTHCARE

KEY CONSIDERATIONS

Itamir de Morais Barroca Filho

EDITOR-IN-CHIEF
Prof. Me. Isabele de Souza Carvalho

EXECUTIVE EDITOR
Nathan Albano Valente

BOOK AUTHOR
Itamir de Morais Barroca Filho

EDITORIAL PRODUCTION
Seven Publications Ltd.

ART EDIT
Alan Ferreira de Moraes

TEXT EDITING
Natan Bones Petitemberte

LIBRARIAN
Bruna Heller

COVER IMAGES
AdobeStok

International Cataloguing in Publication Data (CIP)
(Brazilian Book Chamber, SP, Brazil)

B277a

Barroca Filho, Itamir de Morais.
Architecting IoT for Healthcare [recurso eletrônico] : Key Considerations / Itamir de Morais Barroca Filho. – São José dos Pinhais, PR: Seven Editora, 2024.
Dados eletrônicos (1 PDF).

Inclui bibliografia.
ISBN 978-65-6109-045-2

1. Ciências da saúde. 2. Internet das coisas. I Título.

CDU 61

Indexes for systematic catalog:

1. CDU: Ciências da saúde 61

Bruna Heller-Librarian-CRB10/2348

DOI: 10.56238/livrosindi202439-001

Seven Publications Ltd.
CNPJ: 43.789.355/0001-14
editora@sevenevents.com.br
São José dos Pinhais/PR

AUTHOR'S STATEMENT

The author of this work DECLARES, for the following purposes, that:

Do not have any commercial interest that generates a conflict of interest in relation to the published content;

Declares to have actively participated in the construction of the respective manuscripts, preferably under the following conditions: "a) Study design, and/or data acquisition, and/or data analysis and interpretation; b) Preparation of the article or revision to make the material intellectually relevant; c) Final approval of the manuscript for submission";

Certifies that the published text is completely free of data and/or fraudulent results and authorship defects;

Confirms the correct citation and reference of all data and interpretations of other data Research;

Acknowledges having informed all the sources of funding received to carry out the research;

Authorizes the editing of the work, including catalog records, ISBN, DOI and other indexers, visual design and creation of cover, internal layout, as well as its release and dissemination according to the criteria of Seven Eventos Acadêmicos and Editora.

EDITOR'S STATEMENT

Seven Publications DECLARES, for the purposes of rights, duties and any methodological or legal meanings, that:

This publication constitutes only a temporary transfer of copyright, constituting a right to publication and reproduction of the materials. The Publisher is not co-responsible for the creation of the published manuscripts, under the terms established in the Copyright Law (Law 9610/98), in article 184 of the Penal Code and in article 927 of the Civil Code; The author(s) is solely responsible for verifying such copyright and other issues, holding the Publisher harmless from any civil, administrative, and criminal damages that may arise.

Authorizes the DISCLOSURE OF THE WORK by the author(s) in lectures, courses, events, shows, media and television, provided that there is due recognition of the authorship and edition and without any commercial purpose, with the presentation of the due CREDITS to SEVEN PUBLICATIONS, being the author(s) and publisher(s) responsible for the omission/exclusion of this information;

All eBooks are open access, so don't sell them on your website, partner sites, e-commerce platforms, or any other virtual or physical medium. Therefore, it is exempt from copyright transfers to authors, since the format does not generate rights other than the didactic and advertising purposes of the work, which can be consulted at any time.

All members of the editorial board are PhDs and linked to public institutions of higher education, as recommended by CAPES to obtain the Qualis book;

Seven Eventos Acadêmicos does not attribute, sell or authorize the use of the authors' names and e-mails, as well as any other data of them, for any purpose other than the dissemination of this work, in accordance with the Civil Rights Framework for the Internet, the General Data Protection Law and the Constitution of the Federative Republic.

This book is dedicated to my parents, Itamir and Ilona, my brother, Diego, and girlfriend, Bruna. They are the framework for the architecture of my soul.

SEVEN
publicações acadêmicas

ACKNOWLEDGEMENTS

Thank God, for everything! Thanks, family!

Thanks to my advisor and friend for life.

Thanks to my friends, the oldest and the newest. Thanks to my team for the big help in this research.

Many thanks to everyone that directly or indirectly contributed to this research.

AUTHOR OF THE E-BOOK

Itamir de Morais Barroca Filho

Researcher, professor and entrepreneur. Doctor and Master from the Graduate Program in Systems and Computing at the Federal University of Rio Grande do Norte (UFRN). Computer engineer from UFRN, and specialist in computer networks. He is a professor at the Digital Metropolis Institute (IMD/UFRN), and a permanent professor at the Graduate Program in Information Technology (PPGTI/UFRN). He is the Deputy Director of Projects at IMD/UFRN, as well as Coordinator of the unit of the Brazilian Company for Industrial Research and Innovation (Embrapii) at IMD/UFRN. He works in the areas of development of information systems and mobile applications, Internet of Things (IoT), Cloud Computing and Artificial Intelligence (AI). Since 2015 he has been carrying out research and development projects in Information Technology applied to Health, which gave rise to his doctoral thesis and technological products. He has extensive experience in Research, Development and Innovation (RDI) with public and private partnerships, as he has coordinated more than 20 projects, having raised and managed 20 million reais from this portfolio of projects in the last three years alone (2021-2023). The products resulting from these research emerge competitively not only in the domestic market, but in Latin America and Europe. He also led dozens of other impact initiatives, structured in teaching, research and extension projects, with multiple local and international teams. He is the author of award-winning scientific publications, with a worldwide reach, and has supervised several academic works. As for the intellectual property of innovations, he is the author of more than 30 computer program registrations and remains active in the innovation and entrepreneurship ecosystem.

PRESENTATION

The Internet of Things (IoT) is creating a wave of connected devices that collect data, and this data is being used to develop applications that improve healthcare. These applications have the potential to make hospitals more efficient, treatments more effective, and healthcare overall more affordable. However, there are challenges to overcome in developing these applications, such as making sure the devices can work together and keeping data secure. This document looks at how to design these healthcare applications to address these challenges.

Itamir de Morais Barroca Filho

SUMMARY

ABSTRACT 13

1 INTRODUCTION 14

1.1 PROBLEM STATEMENT 16

1.2 OBJECTIVE 18

1.3 METHODOLOGY 19

1.4 BOOK OUTLINE AND SUMMARY OF CONTRIBUTIONS 20

2 BACKGROUND 21

2.1 SOFTWARE ARCHITECTURE 21

2.1.1 Architectural Structures and Views 22

2.1.2 Software Architecture Terminology 23

2.1.3 Quality Attributes or Nonfunctional Requirements 26

2.2 INTERNET OF THINGS 28

2.2.1 IoT Vision and Scope 30

2.2.2 IoT Characteristics 31

2.3 INTERNET OF THINGS FOR HEALTHCARE 32

2.4 FINAL REMARKS 33

3 RELATED WORK 34

3.1 REFERENCE ARCHITECTURES FOR IOT 34

3.1.1 IoT-A - Reference Architecture 34

3.1.2 IIRA - Industrial Internet Reference Architecture 37

3.1.3 WSO2's Reference Architecture 39

3.2 IOT ARCHITECTURES 41

3.2.1 Three Layers Architectures 41

3.2.2 Middleware Based Architectures 42

3.2.3 Five Layers Architecture 44

3.3 DISCUSSION 45

3.4 FINAL REMARKS 46

4 STATE-OF-THE-ART REVIEW **47**
4.1 METHOD **47**
4.1.1 Research Questions **47**
4.1.2 Search Process **48**
4.1.3 Inclusion and exclusion criteria **48**
4.1.4 Quality assessment **48**
4.1.5 Data collection **49**
4.2 RESULTS **49**
4.2.1 Search Results **49**
4.2.2 Papers Overview **51**
4.2.3 Quality evaluation results **52**
4.3 DISCUSSION **52**
4.3.1 What are the main characteristics of healthcare applications based on IoT infrastructure? **52**
4.3.2 What are the protocols used in healthcare applications based on IoT infrastructure? **54**
4.3.3 What are the challenges related to healthcare applications based on IoT infrastructure? **56**
4.3.4 A technology view for IoT-based healthcare applications **58**
4.3.5 Limitations of this study **59**
4.4 FINAL REMARKS **60**
5 RAH - A SOFTWARE REFERENCE ARCHITECTURE FOR IOT-BASED HEALTHCARE APPLICATIONS **61**
5.1 REQUIREMENTS OF A REFERENCE ARCHITECTURE FOR IOT-BASED HEALTHCARE APPLICATIONS **62**
5.1.1 Functional requirements **63**
5.1.2 Quality attributes **65**
5.1.3 Architecture qualities **66**
5.2 RAH - REFERENCE ARCHITECTURE FOR IOT-BASED HEALTHCARE APPLICATIONS **67**
5.3 FINAL REMARKS **79**
6 ARCHITECTURAL EVALUATION OF RAH **81**
6.1 CASE STUDY DESIGN **81**
6.1.1 Objective **81**

6.1.2 Research Questions (RQs) **82**

6.1.3 Procedures for Data Collection **83**

6.1.4 Methods for Data Analysis **84**

6.2 COLLECTING EVIDENCE **84**

6.2.1 Procedure 1 - Documenting the platform software architecture design **85**

6.2.2 Procedure 2 - Specifying and documenting quality scenarios **102**

6.2.3 Procedure 3 - Implementing the platform based on software architecture designed **107**

6.2.4 Gateway.jar **108**

6.2.5 IoTDataCollector.jar **110**

6.2.6 Intelligence.war **110**

6.2.7 BodyMonitor.war and EnvironmentMonitor.war **111**

6.2.8 AuthService.war **111**

6.2.9 Hospital.war, Ambulance.war and Mobile App **112**

6.2.10 Monitor.jar and MonitorWebInterface.jar **112**

6.3 ANALYSIS OF COLLECTED DATA **113**

6.3.1 RQ1 - RAH allows to design software architectures of IoT-based healthcare applications **113**

6.3.2 RQ2 - RAH address interoperability in IoT-based healthcare applications **114**

6.3.3 RQ3 - RAH address availability in IoT-based healthcare applications **115**

6.3.4 RQ4 - RAH address security in IoT-based healthcare applications **116**

6.3.5 RQ5 - RAH address performance in IoT-based healthcare applications **117**

6.3.6 RQ6 - RAH allows to design and implement software architectures of IoT- based healthcare applications **117**

6.4 DISCUSSION OF RESULTS **118**

6.5 THREATS TO VALIDITY **119**

6.6 FINAL REMARKS **121**

7 CONCLUSIONS **121**

REFERENCES **123**

APPENDIX A - PAR - AN IOT-BASED....132

1 HEALTHCARE PLATFORM FOR REMOTE MONITORING OF PATIENTS IN CRITICAL CONDITIONS. VERSION 1.1....132

1.1 USE CASES SPECIFICATIONS....132

1.1.1 Patient's Data Management....132

1.1.2 Clinical Staff Data Management....136

1.1.3 Health Insurance Data Management....140

1.1.4 Patient and Health Professional Association....143

1.1.5 Patient's Critical Values Configuration....147

1.1.6 Health Data Management....150

1.1.7 Emergency Alert Data Management....153

1.1.8 Ambulance Data Management....155

1.1.9 Health Data Monitoring and Reporting....159

ABSTRACT

The myriad of connected things promoted by the Internet of Things (IoT) and the data captured by them are making possible the development of applications in various markets, such as transportation, buildings, energy, home, industrial, and healthcare. Concerning the healthcare market, it is expected the development of these applications will be part of the future since it can improve e-health to allow hospitals to operate more efficiently and patients to receive better treatment. The IoT can be the main enabler for distributed healthcare applications, thus having a significant potential to contribute to the overall decrease of healthcare costs while increasing health outcomes. However, there are a lot of challenges in the development and deployment of this kind of application, such as interoperability, availability, performance, and security. The complex and heterogeneous nature of IoT-based healthcare applications makes their design, development, and deployment difficult. It also causes an increase in the development cost, as well as an interoperability problem with the existing systems. To contribute to solving the aforementioned challenges, this book aims to improve the understanding and systematization of the IoT-based healthcare applications' architectural design. It proposes a software reference architecture, named Reference Architecture for IoT-based Healthcare Applications (RAH), to systematically organize the main elements of these applications, their responsibilities, and their interactions, promoting a common understanding of these applications' architecture. To establish RAH, a systematic mapping study of existing publications regarding IoT-based healthcare applications was performed, as well as the study of quality attributes, tactics, architectural patterns, and styles used in software engineering. As a result, RAH presents domain knowledge and software architectural solutions (i.e., architectural patterns and tactics) documented using architectural views. To assess RAH, a case study was performed by instantiating it to design the software architecture of a computational platform based on the Internet of Things (IoT) infrastructure to allow the intelligent remote monitoring of the patient's health data (biometrics). With this platform, the clinical staff can be alerted of the health events that require immediate intervention and then prevent unwanted complications. Results evidenced that RAH is a viable reference architecture to guide the development of secure, interoperable, available, and efficient IoT-based healthcare applications, bringing contributions to the areas of e-health and software architecture.

Keywords: Internet of Things (IoT), Healthcare, E-health, Reference architecture, Software architecture.

New technologies can change lives! That is what is happening with the use of the Internet of Things (IoT). The IoT denotes a trend where many embedded devices employ communication services offered by Internet protocols. Many of these devices, often called "smart objects" or "things", are not directly operated by humans but exist as components in buildings or vehicles, or are spread out in the environment (ARKKO et al., 2015). Thus, the basic idea of this paradigm is the pervasive presence, around all of the users, of a variety of things - such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. - which, through unique addressing schemes, can interact with each other and cooperate with their neighbors to reach common goals (ATZORI; IERA; MORABITO, 2010).

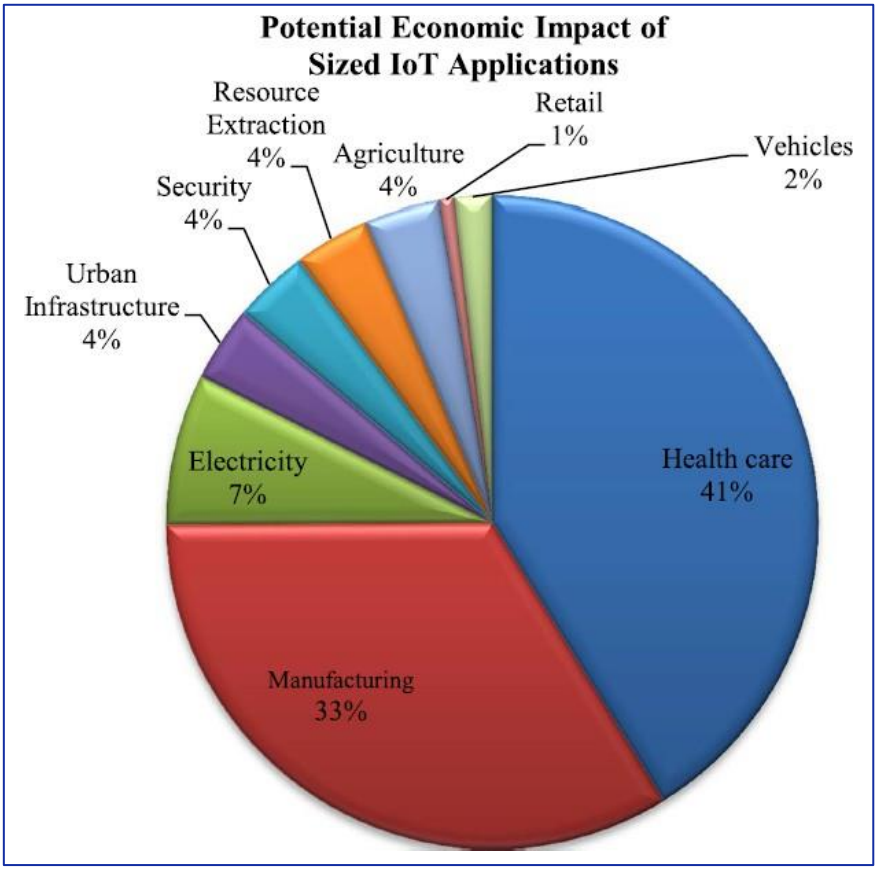
It is estimated that by 2025, 80 billion IoT devices will be online, creating 180 ZB of data (IDC, 2017). This myriad of connected things, the data captured by them, and the connectivity between them will make possible the development of IoT applications in various markets, such as transportation, buildings, energy, home, industrial, and health care. Regarding these applications, six elements are needed to deliver their functionalities: identification, sensing, communication, computation, services, and semantics (AL-FUQAHA et al., 2015), as illustrated in Figure 1. The identification element is crucial for the IoT to name and match services and demands. The sensing element gathers data from related objects, such as smart sensors and actuators. The data, essential to the IoT-based applications, is analyzed and used to direct the applications to perform specific actions. The communication element, in turn, connects different things to deliver the IoT-based applications' requirements.

These applications use processing units, such as microcontrollers and microprocessors, which are the computational elements responsible for processing the data from the sensing elements and sending it to the service's database in the cloud. The services and semantics elements implement the IoT-based applications' requirements by extracting the knowledge from the received data. This extraction includes recognizing and analyzing data that will base the decision-making process required for the application to provide the exact service needed (BARNAGHI et al., 2012).

Figure 1: The IoT elements (AL-FUQAHA et al., 2015).

Therefore, the potential for change in the quality of life that can be promoted by IoT is unquestionable. The IoT has become the major disruptive technology changing software and society (EBERT et al., 2016). Creating integrated utilities will lead to a qualitative change in services to integrate information systems, computing, and communication with extensive control (CHEN, 2016). The main strength of IoT applications is the high impact that has on changing aspects and behaviors of the potential users' everyday lives. Regarding the users, there are two points of view: private and business users. To private users, the application's impact will be related to areas such as home and healthcare, making their lives more comfortable. To business users, it will highly impact aspects such as transportation and industry, changing automation and industrial manufacturing, logistics, business/process management, and smart transportation of people and goods.

Figure 2: Projected market share of dominant IoT applications by 2025 (AL-FUQAHA etal., 2015).



Concerning the healthcare market, it is expected the development of these applications will be part of the future since it can improve e-health to allow hospitals to operate more efficiently and patients to receive better treatment. This paradigm is reshaping modern healthcare, connecting everything to the Internet, shifting "from any time, anyplace connectivity for anyone" to "connectivity for anything". The IoT can be the main enabler for distributed healthcare applications, thus having a

significant potential to contribute to the overall decrease of healthcare costs while increasing health outcomes. In other words, it has the potential to open entire new paths to generate benefits for the patients, health systems, and society at large (COUTURIER et al., 2012).

A type of IoT healthcare application on which developers will focus is the mobile health application (mHealth). The primary goal of mHealth is to allow remote monitoring of the patient's health status (biometrics) and treatment from anywhere in the world (JARA; ZAMORA-IZQUIERDO; SKARMETA, 2013). Moreover, IoT-based healthcare applications are projected to provide the biggest economic impact, as presented in Figure 2. These applications, such as mHealth and telecare, which help to afford medical wellness, prevention, diagnosis, treatment, and monitoring services to be delivered efficiently through electronic media, are expected to create about $ 1.1 - $ 2.5 trillion annually in global economy growth by 2025 (AL-FUQAHA et al., 2015).

On the other hand, population aging and the rise of chronic diseases are becoming a global concern since they might result in an increase in the number of patients at hospitals. Several studies indicate the need for strategies to minimize the institutionalization process and the effects of the high cost of patient care (HOCHRON; GOLDBERG, 2015). Intending to reduce this concern, a promising trend in health treatments is to move the medical check routines from the hospital (hospital-centric) to the patient's home (home-centric). Nowadays, this trend is supported by e-health technologies and can be improved with IoT, with the promotion of distributed healthcare, helping to enhance the outcome of health services and decrease related costs. The progress in wireless technologies with related performance improvements heavily supports real-time monitoring of physiological parameters, thus easing the uninterrupted care of chronic diseases, enabling early diagnosis, and the management of medical emergencies (ISLAM et al., 2015).

1.1 PROBLEM STATEMENT

There is a variety of IoT-based applications that do not contemplate interoperation with other existing systems and devices. Research trends in IoT-based healthcare include network architectures and platforms, new services and applications, interoperability, and security, among others (ISLAM et al., 2015). Moreover, as presented in the previous section, there is also a projection of the development of technologies and applications related to IoT infrastructure for healthcare.

However, there are a lot of challenges in the development and deployment of this kind of application, such as (i) interoperability (DOUKAS; MAGLOGIANNIS, 2012) (KHATTAK et al., 2014) (SEBESTYEN et al., 2014): there are heterogeneous sources of data, the devices' protocol is not open, so a given device cannot be integrated to another (or multiple) applications, and there are also different studies and proposals for patient monitoring at hospitals or personal monitoring at

home; (ii) availability (DOUKAS; MAGLOGIANNIS, 2012): the proposed applications do not provide a way to ensure that the systems are available when needed; (iii) usability (KEVIN et al., 2014): the existing home healthcare systems have drawbacks, such as simple and few functionalities, weak interaction and poor mobility; (iv) security (DOUKAS; MAGLOGIANNIS, 2012): the existing proposed systems lacks of permission control, privacy and data anonymity, etc; (v) flexibility (EBERT et al., 2016): the existing products can not autonomously adapt to usage scenarios, such as assisted living, intelligent buildings, smart transportation, energy, healthcare, transportation, or entire supply chains; (vi) productivity (EBERT et al., 2016): IoT services need to extend toward predictive maintenance and proactive enhancements, improving uptime and thus productivity.

There are also challenges related to data storage and management (DOUKAS; MAGLOGIANNIS, 2012) since the vast volume of data produced by the sensors is in an unstructured format, which is very complicated to understand and requires data storage mechanisms that are different from the typical database management system (DBMS) (MOHAMMED et al., 2014).

In short, the complex and heterogeneous nature of IoT-based healthcare applications makes its design and development difficult. It also causes an increase in the development cost, as well as an interoperability problem with the existing systems. Thus, a strategy to design a software reference architecture to systematically organize the main elements of IoT-based healthcare applications, their responsibilities, and interactions, promotes a common understanding of these applications' architecture. Software reference architectures have emerged as abstractions of concrete software architectures from a certain domain (ANGELOV; GREFEN; GREEFHORST, 2012). Reference architecture (RA) is used to design concrete architectures in multiple contexts, serving as an inspiration or standardization tool (MULLER, 2008). Nowadays, the increasing complexity of software, the need for efficient and effective software design processes, and the need for high levels of system interoperability lead to an increase in the importance of reference architectures in the software design process. IoT architecture and modeling solutions must connect heterogeneous communities to understand and work together (EBERT et al., 2016).

For the existing and emerging IoT applications, it is very well known that they have different architectural requirements such as scalability, flexibility, interoperability, diverse QoS support, and security, to name a few (YAQOOB et al., 2017). Aiming for guidelines to develop these applications, several reference architectures have been proposed considering the necessity to address these requirements. Examples of reference architectures are the three layers architectures proposed by Yang et al. (YANG et al., 2011) and Gubbi et al. (GUBBI et al., 2013); the middleware-based architectures proposed by Tan et al. (TAN; WANG, 2010) and Atzori et al. (ATZORI; IERA; MORABITO, 2010); the five layers architecture proposed by Wu et al. (WU et al., 2010); the wearable architectures

proposed by Hiremath et al (HIREMATH; YANG; MANKODIYA, 2014) and Sharma et al. (SHARMA et al., 2014); IoT-A Reference Architecture; Industrial Internet Reference Architecture (IIRA); and WSO2's Reference Architecture. Despite the existence of reference architectures to guide the development of IoT-based applications, they are too abstract and none of them is focused on supporting the development of IoT-based healthcare applications.

Finally, the definition of a Software Reference Architecture (SRA) for IoT-based healthcare applications could facilitate and standardize the design of concrete architectures, as well as the development of interoperable, secure, efficient, and available systems for healthcare. Thus, the problem addressed in this book is the lack of guidelines to conduct the development of interoperable, secure, efficient, available, and standardized IoT-based healthcare applications.

1.2 OBJECTIVE

Considering the challenges associated with developing IoT-based healthcare applications, mentioned in Section 1.1, the main objective of this book is to propose a reference architecture, named Reference Architecture for IoT-based Healthcare Applications (RAH), to improve the understanding and systematization of the IoT-based healthcare applications' architectural design, and offer guidelines for the development of these applications.

Hypothesis: A software reference architecture for IoT-based healthcare applications is a suitable approach to address the challenges of security, interoperability, availability, and performance, found in developing this kind of applications.

To confirm the hypothesis, the following specific activities were executed:

- **Perform a mapping study based on the Systematic Mapping Study (SMS) methodology:** the study described in Chapter 4 was able to find the main characteristics of IoT-based healthcare applications, their elements, and how they relate to each other.
- **Establishment of a reference architecture for IoT-based healthcare applications (RAH):** it was defined the architecturally significant requirements (functional and non-functional) for RAH, described in Chapter 5, and the design decisions that allow it to achieve such requirements.
- **Evaluation of RAH:** The proposed reference architecture was evaluated through the conduction of a case study, presented in Chapter 6, to obtain shreds of evidence that allowed to confirm the hypothesis and discover improvements to be made.

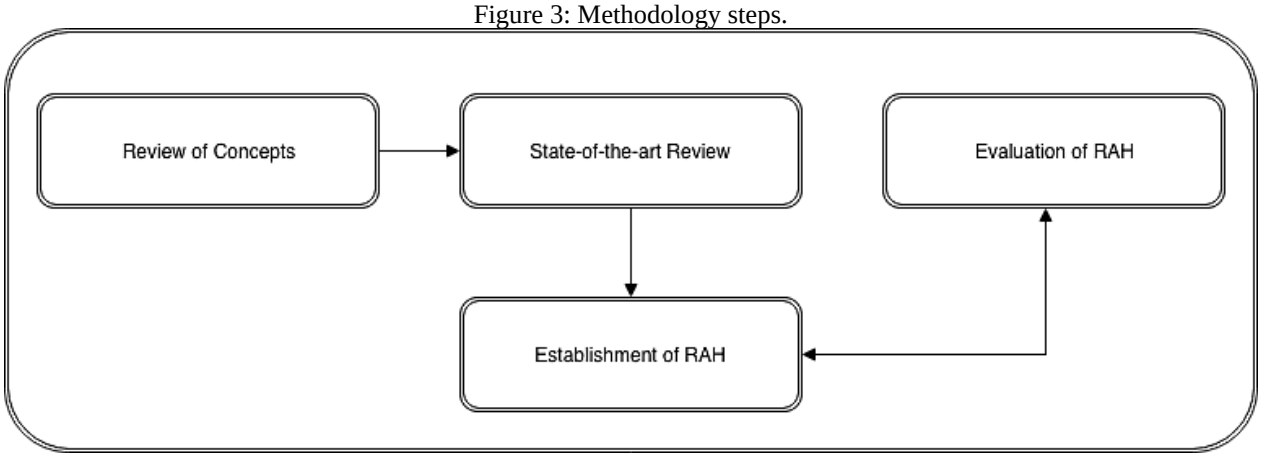
1.3 METHODOLOGY

Aiming to achieve the objective of this research, the methodology, which is presented in Figure 3, consisted of the following steps: Review of Concepts, State-of-the-art Review, Establishment and Evaluation of RAH. In the step of the Review of Concepts, it was reviewed the concepts of software architecture mainly, but not exclusively, on the Software Engineering Institute Books (BASS; CLEMENTS; KAZMAN, 2003) (BASS; CLEMENTS; KAZMA, 2013) (CLEMENTS et al., 2010) (BACHMANN et al., 2011).

In the step of the State-of-the-art Review, it was performed a mapping study based on the Systematic Mapping Study (SMS) methodology (PETERSEN et al., 2008) (KITCHENHAM; BUDGEN; BRERETON, 2011) aiming to comprehend the current state and future trends for IoT-based healthcare applications, as well as to find areas for further investigations. With this study, it was possible to determine the main characteristics, functional requirements, quality attributes or non-functional requirements, challenges, and opportunities of IoT-based healthcare applications. Moreover, in this step, it was performed the study of the related works focused on reference architectures for IoT-based applications.

With the result found, it was started the step of the Establishment of the software reference architecture for IoT-based healthcare applications, named RAH, considering the elements and the relationships between them discovered by the SMS. To establish and document this reference architecture, the concepts of software architecture design methods, views, styles, patterns, and tactics were used.

Figure 3: Methodology steps.



Finally, after the establishment of RAH, it was performed the step of the Evaluation of RAH, accomplished through the conduction of a case study, to obtain evidence that allowed to confirmation of the hypothesis and discovery of improvements to RAH.

1.4 BOOK OUTLINE AND SUMMARY OF CONTRIBUTIONS

This book is structured as follows. Chapter 2 brings an overview of the background information that supports the topics covered in this book. Initially, the terminology and key concepts related to Software Architecture, such as Architectural Structures and Views and Quality Attributes are discussed. Continuing with the background, the Internet of Things (IoT) concepts, such as Vision, Scope, and Characteristics are presented. Then, concepts of IoT for Healthcare and e-health are discussed.

In Chapter 3, the related works regarding reference architectures for IoT-based applications are presented. To find the studies of reference architectures for IoT-based applications, an exploratory review of the literature was performed. Continuing, Chapter 4 describes the state-of-the-art IoT-based healthcare applications, presenting a mapping study based on the Systematic Mapping Study (SMS) methodology. This study was performed by Barroca and Aquino (BARROCA; AQUINO, 2017a) and updated to be used in this book.

Chapter 5 presents the proposed software reference architecture, describing its elements and the relationship between them. This reference architecture was proposed by Barroca and Aquino (BARROCA; AQUINO, 2017b) (BARROCA; AQUINO, 2018). Chapter 6 presents the evaluation of RAH, describing the case study performed to answer if RAH is a suitable approach to address the challenges of interoperability, performance, security, and availability found in developing IoT-based healthcare applications.

The IoT-based healthcare platform used in this case study was defined by Barroca and Aquino (BARROCA; AQUINO, 2017b) (BARROCA; AQUINO; LIMA, 2018). Finally, Chapter 7 presents the conclusions and future works, revisiting the achieved contributions, summarizing limitations, and presenting perspectives of future research.

In this chapter, the theoretical background containing the main topics embraced in this book, namely, software architecture, and the Internet of Things (IoT) is given. Section 2.1 details the main concepts of software architecture, such as architectural structures and views, software architecture terminology, reference architecture, quality attributes, and architecture and requirements. Section 2.2 presents the concepts related to the Internet of Things (IoT), such as IoT vision, scope, and characteristics. Finally, Section 2.3 presents key concepts related to the usage of IoT in healthcare, such as e-health and m-health.

2.1 SOFTWARE ARCHITECTURE

Software systems are developed to satisfy organizations' business goals. The architecture is a bridge between those (often abstract) business goals and the final (concrete) resulting system. While the path from abstract goals to concrete systems can be challenging, the good news is that software architectures can be designed, analyzed, documented, and implemented using known techniques that will support the achievement of these business and mission goals. The complexity can be tamed and made tractable (BASS; CLEMENTS; KAZMA, 2013).

Thus, according to Bass et al. (BASS; CLEMENTS; KAZMA, 2013), the software architecture of a computing system is the set of structures needed to reason about the system, which comprises software elements, the relations between them, and properties from both. When it comes to structures, it is simply a set of elements held together by a relation. Software systems are composed of many structures. There are three categories of architectural structures, which will play an important role in the architectures design, documentation, and analysis:

1. **Module structures:** The systems are partitioned into implementation units called modules. Modules are assigned specific computational responsibilities and are the basis of work assignments for programming teams (Team A works on the database, Team B works on the business rules, Team C works on the user interface, etc.) (BASS; CLEMENTS; KAZMA, 2013). In some projects, the modules are subdivided to assign work to sub-teams, for example, the subsystems of a system can be decomposed into many parts. The module structure that captures the decomposition is the decomposition module structure. Another kind of module structure emerges as an output of object-oriented analysis and design class diagrams. If the modules are aggregated into layers, another module structure is created. Module structures are static structures that focus on the way the system’s functionality is divided up and assigned to implementation teams (BASS; CLEMENTS; KAZMA, 2013).

2. **Component-and-connector (C&C) structures:** The elements of a system interact with each other at runtime to carry out the system's functionalities. To capture the system's runtime characteristic, it is used the component-and-connector structures. Thus, a component is always a runtime entity that interacts with another component by using a connector.
3. **Allocation structures:** This structure describes the mapping from the software structures to the system's organizational, developmental, installation, and execution environments. For example, modules are assigned to teams to be developed and assigned to places in a file structure for implementation, integration, and testing. Components are deployed onto hardware to be executed. These mappings are called allocation structures (BASS; CLEMENTS; KAZMA, 2013).

Still regarding structures, a structure supports reasoning about the system and the system's properties. The reasoning should be related to an attribute of the system that is important to some stakeholders. These include functionalities achieved by the system, the system's availability when it comes to faults, the difficulty of making specific changes to the system, the system's responsiveness to user requests, and many others. Finally, other important definitions of software architecture presented in the software engineering community are (CLEMENTS et al., 2010): architecture is a high-level design; architecture is the system's overall structure, and architecture is the components and connectors.

2.1.1 Architectural Structures and Views

As presented in the previous section, a structure is a set of elements as they exist in software or hardware. To represent a software architecture, the structures are related to views. A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them (BASS; CLEMENTS; KAZMA, 2013). Therefore, a view is a representation of a structure. For example, a module structure is the set of the system's modules and their organization. A module view is the representation of that structure, documented according to a template in a chosen notation, and used by some system's stakeholders. Thus, architects design structures and document views of those structures.

The module structures embody decisions of how the system has to be structured and what set of code or data units have to be constructed. In this structure, the elements are modules of some kind, for example, classes or layers, all of which are units of implementation. The modules represent a static way of considering the system and are assigned to areas of functional responsibility. It allows

us to reason about the functional responsibility assigned to each module; the software elements that a module is allowed to use; the dependencies with other software; and the relationships between the modules by generalization or specialization (i.e., inheritance). Thus, the module structures convey this information directly, but they can also be used by extension to ask questions about the impact on the system when the responsibilities assigned to each module change. In other words, examining a system's module structures that is, looking at its module views, is an excellent way to reason about a system's modifiability (BASS; CLEMENTS; KAZMA, 2013).

The component-and-connector structures embody decisions of how the system will be structured as a set of elements that have runtime behavior (components) and interactions (connectors). In these structures, the elements are runtime components (which are the main units of computation and could be services, peers, clients, servers, filters, or many other types of runtime elements) and connectors (which are the vehicles of communication between components, such as call-return, process synchronization operators, pipes, or others) (BASS; CLEMENTS; KAZMA, 2013). These views allow us to reason about the main executing components and how they interact at runtime; the shared data stores; data progress through the system; and the parts of the system that can run in parallel. By extension, component-and-connector views are crucially important for asking questions about the system’s runtime properties such as performance, security, availability, and more.

The allocation structures embody decisions of how the system will relate to nonsoftware structures in its environment, such as CPUs, file systems, networks, and development teams. These structures show the relationship between software elements and elements in one or more external environments in which the software is created and executed. Thus, allocation views help us to reason about the processor in which each software element executes; directories or files in which each element is stored during the system’s development, testing, and building; and the assignment of each software element to development teams.

2.1.2 Software Architecture Terminology

Architectural patterns, reference models, and reference architectures are some important terms related to software architecture that represent the outcome of a set of architectural decisions. Thus, according to Bass et al. (CLEMENTS et al., 2010), an architectural pattern is a description of element and relation types together with a set of constraints on how they may be used. A pattern can be thought of as a set of constraints on an architecture and the element types and their patterns of interaction. These constraints define a set or family of architectures that satisfy them.

The terms architectural style and architectural pattern are used in similar ways (HOFMEISTER; NORD; SONI, 2000). For example, client-server is a common architectural pattern.

Client and server are two element types, and their coordination is described regarding the protocol that the server uses to communicate with each of its clients. The use of the term client-server implies only that multiple clients exist; the clients themselves are not identified, and there is no discussion of what functionality other than the implementation of protocols has been assigned to any of the clients or the server. Countless architectures follow the client-server pattern under this (informal) definition, but they are different from each other.

Another example of an architectural pattern is the pipes and filters, composed of two types of elements - pipes and filters. A pipe can be connected to a filter, but not to other pipes, nor filters to other filters. In this style, processing is mapped to filters, and pipes act as data conduits. Finally, an architectural pattern or style is not an architecture, but it conveys a useful image of the system and imposes useful constraints on the software architecture.

One of the most useful aspects of patterns is that they present known quality attributes. This is why the architect chooses a particular pattern and not one at random. Some patterns represent known solutions to performance problems, while others lend themselves well to high-security systems; other patterns have been successfully used in high-availability systems. Choosing an architectural pattern is often the architect's first significant design choice. The term architectural style has also been widely used to describe the same concept.

Regarding the reference model, according to Bass et al. (CLEMENTS et al., 2010), it is a division of functionality together with data flow between the pieces. A reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem. Arising from experience, reference models are a characteristic of mature domains, for example, they can be named the standard parts of a compiler or a database management system, and that is why it has been taught about these applications' reference models.

Finally, still according to Bass et al. (CLEMENTS et al., 2010), a reference architecture is a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them. While a reference model divides the functionality, a reference architecture is the mapping of that functionality onto a system decomposition. The mapping may be, but by no means necessarily is, one-to-one. A software element may implement part of a function or several functions.

The terms "reference architecture" and "domain-specific software architecture" are used in similar ways (HOFMEISTER; NORD; SONI, 2000). Thus, it defines how the domain functionality is mapped to the architecture elements. An example of a reference architecture is a compiler. There is a general notion of the basic elements of a compiler, for example, the lexical syntax and semantic analyzers (parsers).

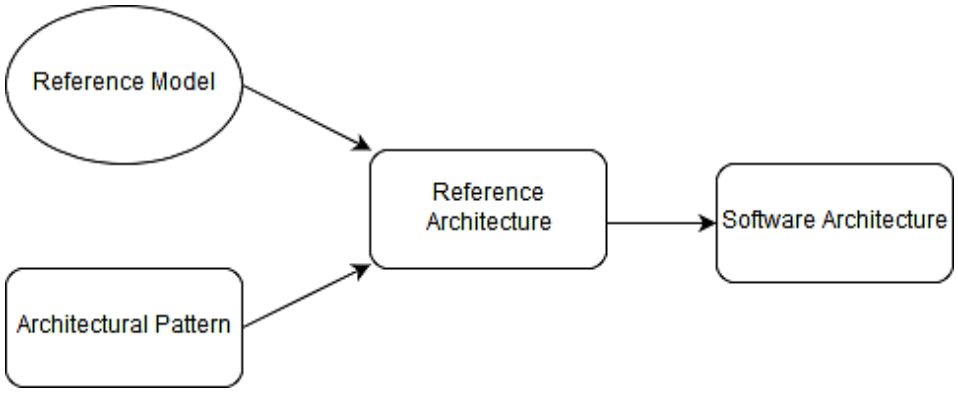
Reference models, architectural patterns or architecture styles, and reference architectures or domain-specific software architecture are not concrete software architectures; they are useful concepts that capture elements of an architecture. Each one is the outcome of early design decisions, and the relationship among them is presented in Figure 4.

2.1.2.1 Comparison of Concrete and Reference Architectures

There are several differences between reference architectures and concrete architectures (ANGELOV; TRIENEKENS; GREFEN, 2008):

1. Reference architectures are generic. A reference architecture is designed to address the functionalities and qualities desired by all stakeholders in their specific contexts, as presented in Figure 5.
2. There is not a clear group of stakeholders of a reference architecture. As stakeholders can be seen all companies from the domain, all companies developing software for the domain, etc. However, it is not possible to involve all these stakeholders in the definition of a reference architecture (due to logistic, political, etc. reasons).

Figure 4: Relationship between reference models, architectural patterns, reference architectures, and software architectures (CLEMENTS et al., 2010).



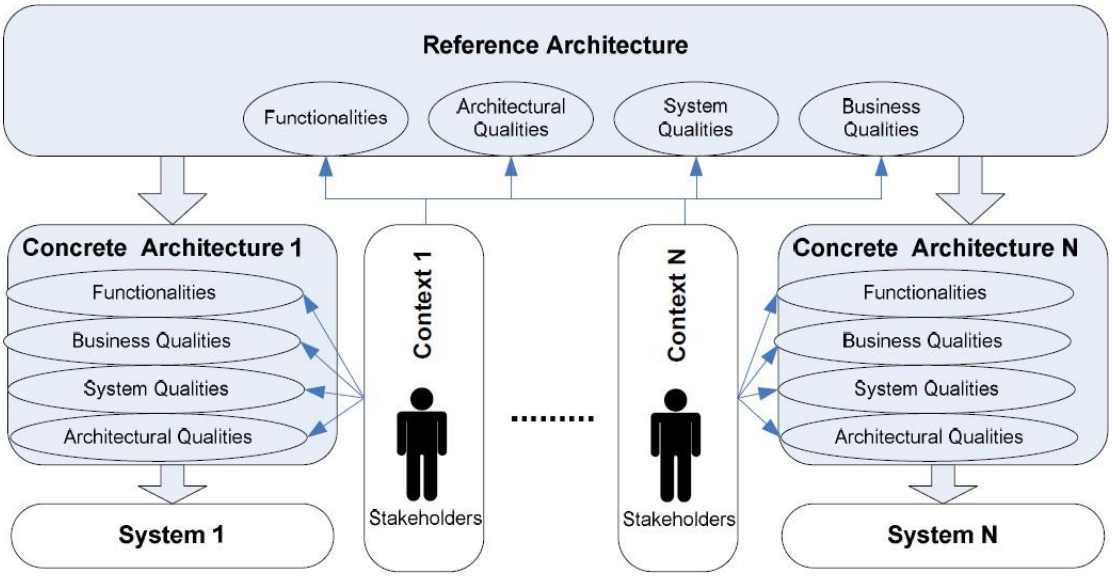
3. Due to their generic nature, reference architectures are defined on a high level of abstraction. They may provide details only for specific elements.
4. A reference architecture has to address more functional requirements and quality attributes than a concrete architecture. These additional architectural qualities are due to the generic nature of reference architectures and their wider audience. For example, an "applicability" quality would be of importance for a reference architecture to indicate the level of applicability of the architecture to different contexts in the domain. This quality is superfluous for concrete architecture as concrete architecture is designed to apply to a specific context.

Because of these differences between concrete and reference architectures, reference architectures are considered by some authors as very distant from concrete architectures: "Reference architectures are not architectures; they are useful concepts that capture elements of an architecture" (CLEMENTS et al., 2010) (ANGELOV; TRIENEKENS; GREFEN, 2008).

2.1.3 Quality Attributes or Nonfunctional Requirements

The qualities of a software architecture go beyond functionality, which is the primary statement of the system's capabilities, services, and behavior. For example, systems are often redesigned not because they are functionally deficient, but because they are difficult to maintain, port, or scale or they are too slow. Thus, when it refers to these characteristics, it is referring to quality attributes. According to Bass et al. (BASS; CLEMENTS; KAZMA, 2013), a quality attribute (QA) is a measurable or testable property of a systemused to indicate how well the system satisfies the needs of its stakeholders. This way, itcan be understood that a quality attribute is a measure of "how good" a system is along with some dimension of how interesting it is to a stakeholder.

Figure 5: The role of stakeholders and contexts for reference and concrete architectures (ANGELOV; TRIENEKENS; GREFEN, 2008).



2.1.3.1 Architecture and Requirements

Therefore, requirements for a system come in a variety of forms: textual requirements, mockups, existing systems, use cases, user stories, etc. All requirements encompass the following categories (BASS; CLEMENTS; KAZMA, 2013):

1. **Functional requirements:** These requirements state what the system must do and how it must behave or react to runtime stimuli.

2. **Quality attributes or Nonfunctional requirements:** These requirements are qualifications of the functional requirements or the overall product. A qualification of a functional requirement is an item such as how fast the function must be performed, or how resilient it must be to erroneous input. Qualification of the overall product is an item such as the time to deploy the product or a limitation on operational costs.
3. **Constraints:** A constraint is a design decision that should be made with no freedom at all. That is, it's a design decision that's already been made. Examples include the requirement to use a certain programming language or to reuse a certain existing module, or a management fit to make the system service-oriented. These choices are arguably in the architect's scope, but external factors such as not being able to train the staff in a new language, having a business agreement with a software supplier, or pushing business goals of service interoperability) have led those with the power to dictate these design outcomes.

The software architecture has “responses” for each of these requirements:

1. Functional requirements are satisfied by assigning an appropriate sequence of responsibilities throughout the design. As it will be seen later in this chapter, assigning responsibilities to architectural elements is a fundamental architectural design decision.
2. Quality attributes or Nonfunctional requirements are satisfied by the various structures designed into the architecture, and the behaviors and interactions of the elements included in those structures.
3. Constraints are satisfied by accepting a design decision and reconciling it with other affected design decisions.

2.1.3.2 Specifying Quality Attribute Requirements

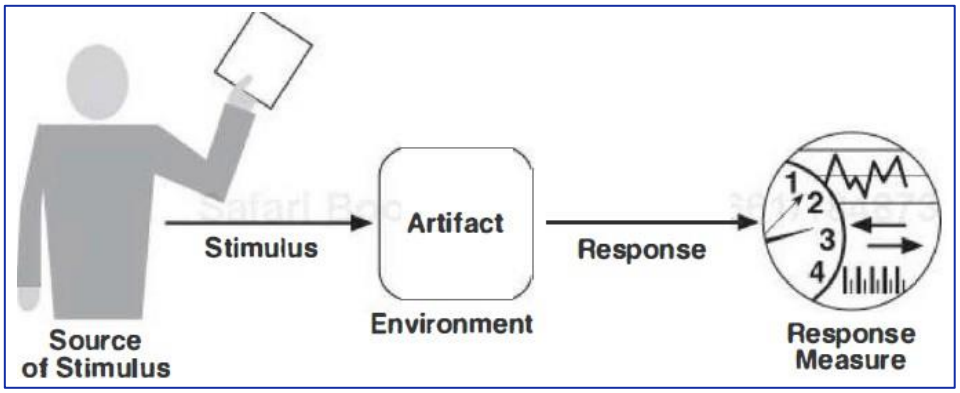
A quality attribute requirement should be unambiguous and testable (BASS; CLEMENTS; KAZMA, 2013). There is a common form to specify all quality attribute requirements, which has the advantage of emphasizing the commonalities among all quality attributes. This common form of quality attribute expression is formed of:

1. **Source of stimulus:** An entity (a human, a computer system, or any other actuator) that generated the stimulus.
2. **Stimulus:** The stimulus is a condition that requires a response when it arrives at a system.

3. **Environment:** The stimulus occurs under certain conditions. The system may be in an overload condition in normal operation, or some other relevant state. For many systems, "normal" operation can refer to different modes. For these kinds of systems, the environment should specify in which mode the system is executing.
4. **Artifact:** An artifact is stimulated. This may be a collection of systems, the whole system, or some piece or pieces of it.
5. **Response:** The response is the activity undertaken as the result of the stimulus arrival.
6. **Response measure:** When the response occurs, it should be measurable in any way so that the requirement can be tested.

These parts are presented in Figure 6.

Figure 6: Parts of a quality attribute scenario (BASS; CLEMENTS; KAZMA, 2013).



2.2 INTERNET OF THINGS

The Internet of Things (IoT) is a paradigm that is rapidly gaining ground in the modern wireless telecommunications scenario. This is mainly due to the growing number of physical objects that are being connected to the Internet. These objects achieve the idea of the Internet of Things (IoT), which is the pervasive presence of a variety of things or objects such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc., around the users, which, through unique addressing schemes, can interact with each other and cooperate with their neighbors to reach common goals (ATZORI; IERA; MORABITO, 2010). Examples of these things include thermostats and HVAC (Heating, Ventilation, and Air Conditioning) monitoring and control systems which make smart homes possible. The "things" are related to systems that have different domains, such as transportation, healthcare, industrial automation, and emergency response. Thus, in this context, the IoT can play a remarkable role and improve the quality of our lives (AL-FUQAHA et al., 2015). Moreover, according to Atzori et al. (ATZORI; IERA; MORABITO, 2010), the main strength of the

IoT concept is the high impact it will have on several aspects of everyday life and the behavior of potential users.

From a private user's point of view, the most apparent effects of the use of the IoT will be visible in both working and domestic fields. In this context, domotics, assisted living, healthcare, and enhanced learning are only a few examples of possible application scenarios in which the new paradigm will play a leading role shortly. Similarly, from the business users' perspective, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, and the intelligent transportation of people and goods.

IoT enables things to see, hear, think, and perform tasks, as well as to share data and coordinate decisions. The IoT transforms these objects from plain and traditional things to smart objects by exploiting its underlying technologies such as ubiquitous and pervasive computing, embedded devices, communication technologies, sensor networks, Internet protocols, and applications (ALFUQAHA et al., 2015). Smart objects, along with their supposed tasks, constitute specific domain applications (vertical markets) while ubiquitous computing and analytical services form independent domain services (horizontal markets). Figure 7 illustrates the overall concept of the IoT in which every specific domain application interacts with independent domain services, whereas in each domain sensors and actuators communicate directly with each other.

Figure 7: Overall picture of the IoT emphasizing the vertical markets and the horizontal integration between them (ALFUQAHA et al., 2015).

2.2.1 IoT Vision and Scope

From a conceptual vision, the IoT is built on three pillars, related to the ability of things to (i) be identifiable (anything identifies itself), (ii) communicate (anything communicates) and (iii) interact (anything interacts), either among themselves, building networks of interconnected objects, or with end users or other entities in the network (MIORANDI et al., 2012). Therefore, the development of technologies and solutions to enable such a vision is the main challenge for us. Regarding the definition of "things", they are entities that (MIORANDI et al., 2012):

- Have a physical embodiment and a set of associated physical features (e.g., size, shape, etc.);
- Have a minimal set of communication functionalities, such as the ability to be discovered and to accept incoming messages and reply to them.
- Own a unique identifier;
- Have some basic computing capabilities;
- May hold means to sense physical phenomena (e.g., temperature, light, electromagnetic radiation level) or to trigger actions affecting the physical reality (actuators).

From a system perspective, the IoT can be looked at as a highly dynamic and radically distributed networked system composed of a substantial number of smart objects producing and consuming information. The ability to interface with physical reality is achieved through the presence of devices able to sense physical phenomena and translate them into a stream of information data (thereby providing information on the current context and/or environment), as well as through the presence of devices able to trigger actions that have an impact on the physical field (through suitable actuators).

By using these technologies combined it will be possible to create what is referred to as a smart world. For example, nowadays, many buildings already have sensors in an attempt to save energy; home automation is occurring; cars, taxis, and traffic lights have devices to try and improve safety and transportation; people have smartphones with sensors to run many useful apps; industrial plants are connecting to the Internet, and healthcare services are relying on increased home sensing to support remote medicine and wellness. Finally, from a human perspective, it will often be an integral part of the IoT system, and consequently, in the future, the scope of IoT will be enormous and will affect every aspect of our lives (STANKOVIC, 2014).

2.2.2 IoT Characteristics

From a research perspective, according to Pereira et al. (PERERA et al., 2014) (SUNDMAEKER et al., 2010), the IoT has seven major characteristics: intelligence, architecture, complex system, size considerations, time considerations, space considerations, and everything-as-a-service. These characteristics need to be considered throughout all the phases of the development of IoT solutions, which are design, development, implementation, and evaluation.

Regarding intelligence, it means the generation of knowledge. First, to generate knowledge, it needs to collect the raw data. The data is transformed into knowledge (high-level information) mainly by modeling and reasoning the context of the application. The context can be used to fuse the sensor data to infer new knowledge. Once it is known, it can be applied to more intelligent interaction and communication.

IoT should be facilitated by a hybrid architecture which comprises many different architectures. Primarily there would be two architectures: an event-driven and a time-driven (PERERA et al., 2014). Some sensors produce data when an event occurs (e.g., door sensor), while the rest provide data continuously, based on specified time frames (e.g., temperature sensor).

The complex system characteristic is related to the fact that the IoT comprises a large number of objects/things (sensors and actuators) that interact autonomously. Currently, there are millions of sensors deployed around the world (LE-PHUOC et al., 2010), and a projection of billions of things connected in the next years. The size considerations is that IoT needs to facilitate the interaction between these objects. These numbers will grow continuously and will never decrease, and similar to the number of objects, the number of interactions may also increase significantly. These interactions may differ considerably depending on the objects' capabilities. Moreover, some objects may have very few capabilities, as well as store insufficient information and do not process it at all or, in other cases, may have more significant memory, processing, and reasoning capabilities, which make them more intelligent.

The time considerations regard the fact that IoT could handle billions of parallel and simultaneous events due to the massive number of interactions; therefore, real-time data processing is essential. Beyond time, space considerations regard the locations of the objects. These locations play a significant role in context-aware computing. When the number of objects gets larger, tracking becomes an essential requirement. Interactions are highly dependent on the objects' positions, their surroundings, and the presence of other entities (e.g., objects and people) (PERERA et al., 2014).

Finally, the everything-as-a-service characteristic concerns the popularity of cloud computing, consuming resources as a service such as Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), and Software-as-a-Service (SaaS). The everything-as-a-service model is highly efficient,

scalable, and easy to use (BANERJEE et al., 2011). IoT demands significant amounts of infrastructure to be put in place to make its vision a reality, where it would follow a community or crowd-based approach. Therefore, sharing would be essential and an everything-as-a-service model would suit sensing-as-a-service the most (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2013).

2.3 INTERNET OF THINGS FOR HEALTHCARE

Traditionally, the motivation for utilizing modern Information and Communication Technologies (ICT) in the healthcare system is to offer promising solutions for efficiently delivering all kinds of medical healthcare services to patients, named as e-health, such as electronic record systems, telemedicine systems, personalized devices for diagnosis, etc (QI et al., 2017). E-health involves a broad group of activities that use electronic means to deliver health-related information, resources, and services. It encompasses a range of standards, tools, and activities that use electronic means to deliver information, resources, and services about health and social care. At the heart of e-health is a vision of improving the quality of health information, strengthening national health systems, and ensuring accessible, high-quality health care for all (ORGANIZATION, 2018).

Driven by a sustained increase in longevity, many developed countries are now facing the fact that their fast-growing demographics are the over 80s (QI et al., 2017). This trend brings with it some key concerns about the economic viability of traditional healthcare systems, and thus it needs to design and develop more coherent and ubiquitous ICT-enabled solutions for delivering high-quality patient-centered healthcare services. In this context, the use of IoT technology will enable faster and safer preventive care, lower overall cost, improved patient-centered practice, and enhanced sustainability. This technology will play a prominent role in patient telemonitoring in hospitals and more importantly at home (AHMADI et al., 2018).

There are several applications for healthcare in IoT, which can potentially deliver comprehensive patient care in various settings, including acute (in hospital), long-term (nursing homes), and community-based (typically, in-home). With patients attached to sensors to measure vital signs and other biometric information, problems could be more rapidly diagnosed, a better quality of care given, and resources used more efficiently (LA- PLANTE; LAPLANTE, 2016). The healthcare sector always seeks new approaches to service delivery, reducing costs and improving healthcare quality; therefore, the reliance of this sector on IoT technology will be increased.

IoT-based healthcare applications can be used in a diverse array of fields, including care for pediatric and elderly patients, the supervision of chronic diseases, and the management of private health and fitness, among others. Most of the current IoT-based applications for healthcare were proposed for home healthcare monitoring (AHMADI et al., 2018). In this regard, Ambient-Assisted

Living (AAL) technologies can create suitable solutions for disabled and elderly people suffering from different disabilities and chronic diseases.

AAL using a dynamic and interconnected environment has the potential to improve people's quality of life (CALVARESI et al., 2017). Based on Blackman et al. (BLACKMAN et al., 2016), three generations of AAL systems can be distinguished. The first generation includes wearable devices, usually alarms for emergencies. The second generation is home sensors that provide automatic response to detection of hazards. Finally, the third generation is based on the integration of wearable devices and home sensors, applicable for monitoring patient situations and prevention of health risks.

Another e-health and IoT-related component is mHealth. To date, no standardized definition of mHealth has been established. The World Health Organization (WHO) (ORGANIZATION, 2011) defined mHealth or mobile health as medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices. mHealth involves the use and capitalization on a mobile phone's core utility of voice and short messaging service (SMS) as well as more complex functionalities and applications including general packet radio service (GPRS), third and fourth generation mobile telecommunications (3G and 4G systems), global positioning system (GPS), and Bluetooth technology (ORGANIZATION, 2011).

Finally, the main scenarios of usage of IoT-based healthcare applications, supported by e-health and ICT, are monitoring physiological and pathological signals; self-management, wellness monitoring and prevention; medication intake monitoring; personalized healthcare; cloud-based health information systems; disease monitoring and telepathology; assisted living; and rehabilitation.

2.4 FINAL REMARKS

This chapter presented the essential concepts related to this book, such as software architecture and the Internet of Things. In the software architecture section, it was presented the concepts of architecture structure and views; the software architecture terminology; and quality attributes. In the Internet of Things section, it was defined the IoT vision, scope, and characteristics. Moreover, In the Internet of Things for Healthcare subsection, some concepts of home healthcare, e-health and m-health, and scenarios of IoT-based applications were presented. Finally, The next chapter will present the related works of this book.

In this chapter, a review of the existing architectures and reference architectures for the Internet of Things (IoT) is provided. These architectures were identified through the conduction of an exploratory review. Currently, to the best of our knowledge, there is no specific reference architecture for IoT-based healthcare applications. The chapter is organized as follows: in Section 3.1, the reference architectures for IoT are detailed. Section 3.2, in turn, presents the existing architectures for IoT applications.

3.1 REFERENCE ARCHITECTURES FOR IOT

Identifying and structuring an architecture or model is a long and tedious process with much negotiation to abstract from specific needs and technologies. Such a reference can serve as an overall, generic guideline; not all domain applications will require each detail for real-life implementation (EBERT et al., 2016). In the IoT context, the applications have been based on fragmented software implementations for specific systems and use cases, and usually do not follow reference architectures. The need for reference architectures in the industry has become tangible with the fast-growing number of initiatives working toward standardized architectures. These initiatives aim to facilitate interoperability, simplify development, and ease implementation (EBERT et al., 2016). There are three major reference architectures found in the literature for IoT: IoT-A, IIRA, and WSO2. In the following sections, these reference architectures will be presented.

3.1.1 IoT-A - Reference Architecture

The lack of standardization in the IoT domain has resulted in the fragmentation of the approaches in IoT systems design and implementation. To address this problem, the IoT-A project of the EU (BASSI et al., 2016) proposed an Architecture Reference Model (ARM) defining the principles and guidelines for generating IoT architectures, providing the means to connect systems in the communication (how devices interact with the system) and service (how services are integrated).

The IoT-A Reference Architecture is, among others, designed as a reference for the generation of compliant IoT concrete architectures that are tailored to one's specific needs (BAUER et al., 2013). It is an abstract framework that comprises a minimal set of unifying concepts, axioms, and relationships for understanding significant relationships between entities of the IoT domain. It consists of several submodels that set the scope for the IoT design space (STRAVOSKOUFOS; SOTIRIADIS; PETRAKIS, 2016):

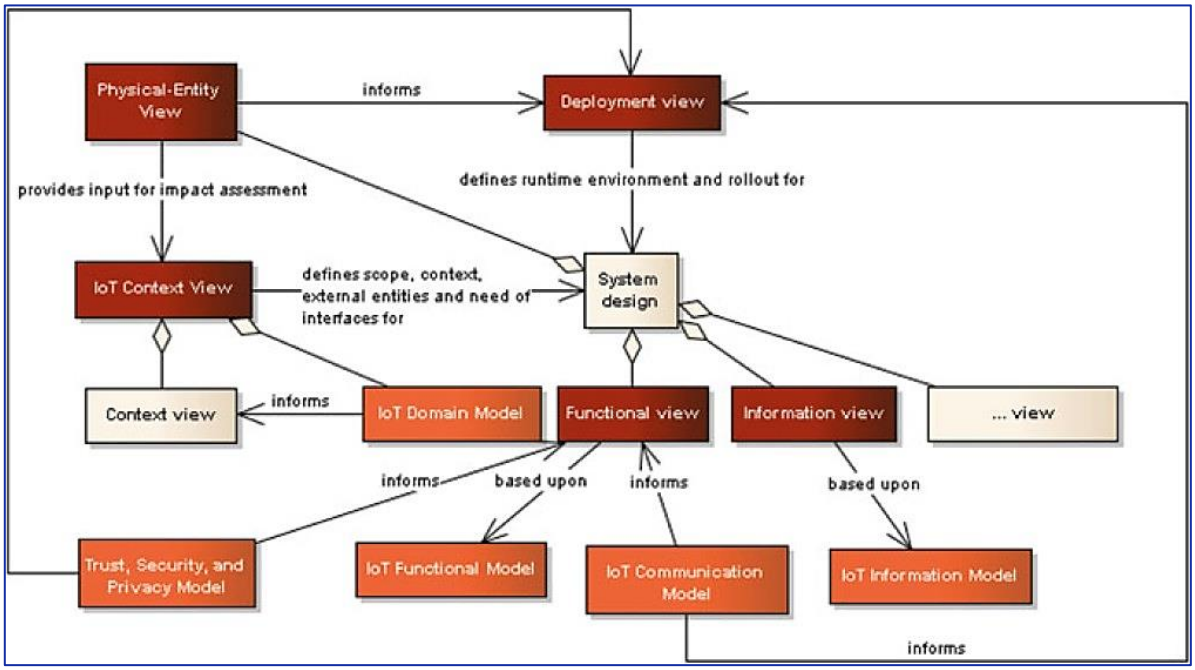
1. **IoT domain model:** It is a top-level description (a UML diagram or ontology) of the IoT domain that describes the main concepts of the IoT like Devices, IoT Services, and Virtual Entities (VEs) that is, anything that has a distinct existence, and also relations between these concepts.
2. **IoT Information model:** An abstract description (UML diagram or ontology) for explaining information about elements or concepts defined in the IoT Domain Model (e.g., applicability of concepts).
3. **IoT Functional model:** It identifies Functional Groups (FGs) that is, groups of functionalities, grounded in key concepts of the IoT Domain Model.
4. **IoT Communication Model:** Introduces concepts for handling the complexity of communication in an IoT environment. It is one FG in the IoT Functional model.
5. **Trust, Security, and Privacy (TSP) model:** Introduces functionality related to Trust, Security, and Privacy. TSP is also one FG in the IoT functional model.

This reference architecture is based on the concepts of architectural views and architectural perspectives. The IoT-A RA's views addressing one aspect of the architectural structure are (STRAVOSKOUFOS; SOTIRIADIS; PETRAKIS, 2016):

1. **Physical Entity View:** It describes all physical entities and their relations (e.g., sensors, actuators, environment measurements) in an IoT system. This view is not covered by IoT-A because it is use-case-independent.
2. **IoT context View:** It provides context information about physical entities such as the Physical Entity View, this view is also not covered by IoT-A as it is use case-independent.
3. **Functional View:** It describes the system's runtime Functional Components, their responsibilities, default functions, interfaces, and primary interactions. The Functional View derives from the Functional Model and reflects the developer's perspectives on the system.
4. **Information View:** Provides an overview of how (a) static information (i.e., VEs using hierarchies, semantics) and (b) dynamic information (i.e., information processing, storage, flow) is represented.
5. **Deployment View:** It explains the operational behavior of the functional components and their interplay of them.

Finally, Figure 8 demonstrates the relationship between IoT-A architectural views and model in the process of designing the actual system architecture, and Figure 9 presents the Functional decomposition viewpoint of the IoT Reference Architecture.

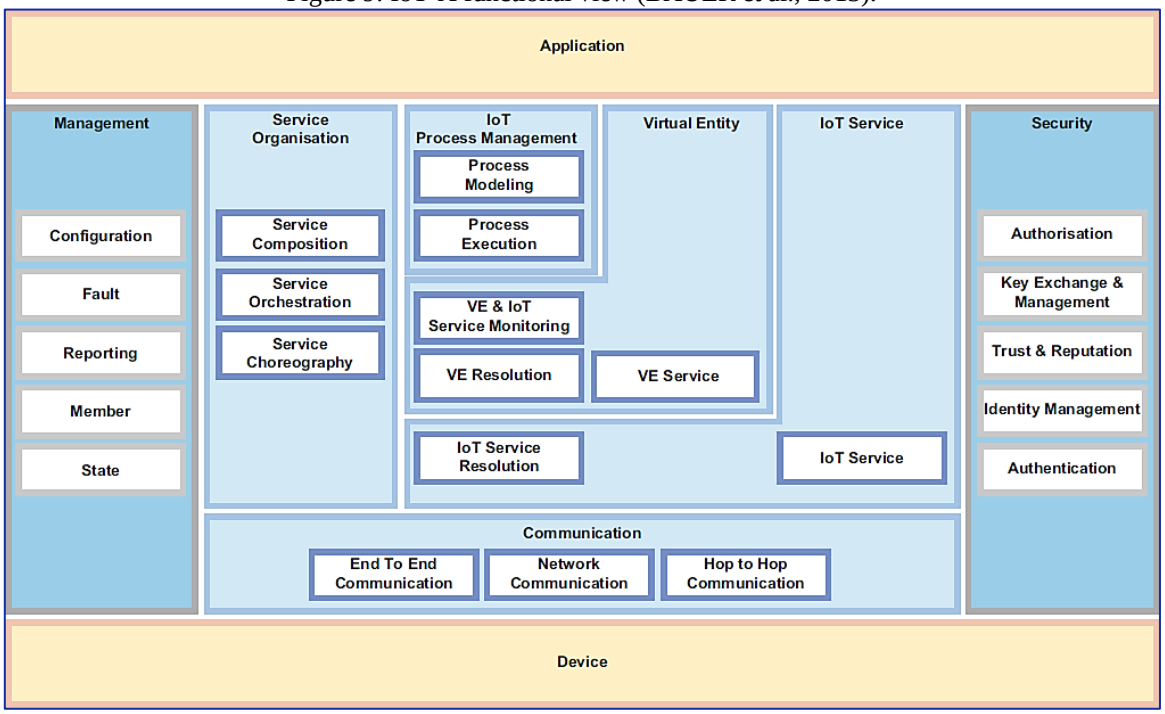
Figure 8: Relationship of IoT-A architectural views and models (BASSI et al., 2016) (ROZANSKI; WOODS, 2012).



The IOT RA's stakeholder requirements clearly show the need to address nonfunctional requirements. Based on them, the perspectives which are most important for IoT- systems are (BAUER et al., 2013):

- Evolution and Interoperability;
- Availability and Resilience;
- Trust, Security, and Privacy;

Figure 9: IoT-A functional view (BAUER et al., 2013).



- Performance and Scalability.

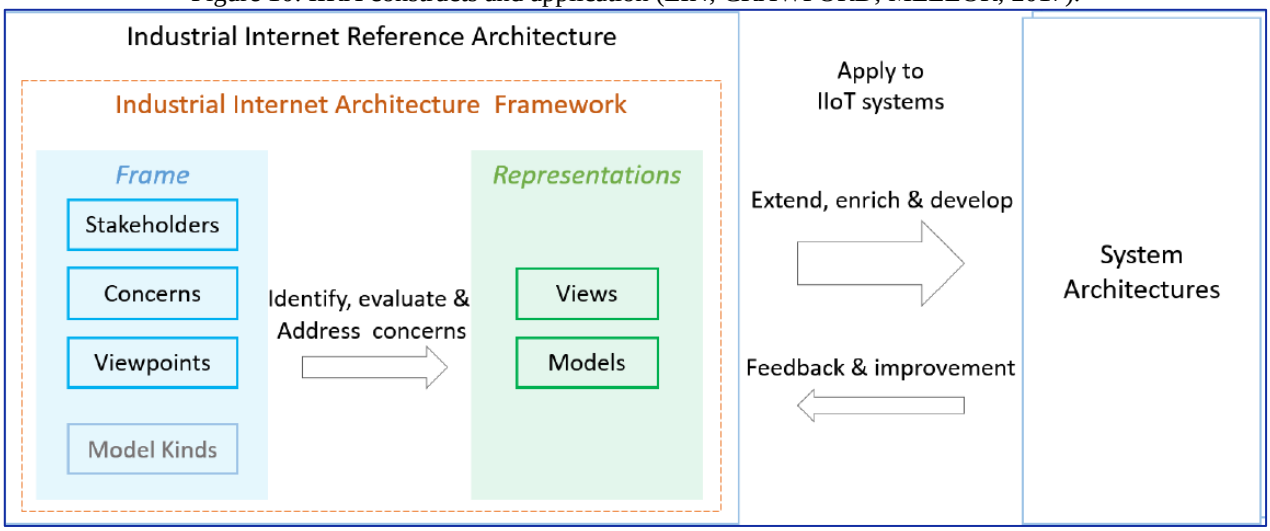
However, according to Bauer et al. (BAUER et al., 2013), as these requirements require some kind of quality for a real system, the perspectives aim more on the concrete system architecture, than at a Reference Architecture. Thus, although identified, these requirements are not evaluated.

3.1.2 IIRA - Industrial Internet Reference Architecture

The IIRA is a standards-based open architecture for Industrial IoT (IIoT) systems. The IIRA maximizes its value by having broad industry applicability to drive interoperability, map applicable technologies, and guide technology and standard development. The architecture description and representation are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA distills and abstracts common characteristics, features, and patterns from use cases defined in the Industrial Internet Consortium (IIC) as well as elsewhere. It will be refined and revised continually as feedback is gathered from its application in the testbeds developed in IIC as well as real-world deployment of IoT systems. The IIRA design is also intended to transcend today's available technologies and so can identify technology gaps based on the architectural requirements. This will in turn drive new technology development efforts by the IIC (LIN; CRAWFORD; MELLOR, 2017).

The IIRA documents the outcome of applying its framework to the intended class of systems of interest: Industrial Internet of Things systems. It first identifies and highlights the most important architectural concerns commonly found in IIoT systems across industrial sectors and classifies them into viewpoints along with their respective stakeholders. It then describes, analyzes, and, where appropriate, guides to resolve these concerns in these viewpoints, resulting in a certain abstract architectural representation. Figure 10 illustrates the key ideas about the constructs of the Industrial Internet Reference Architecture and its application.

Figure 10: IIRA constructs and application (LIN; CRAWFORD; MELLOR, 2017).

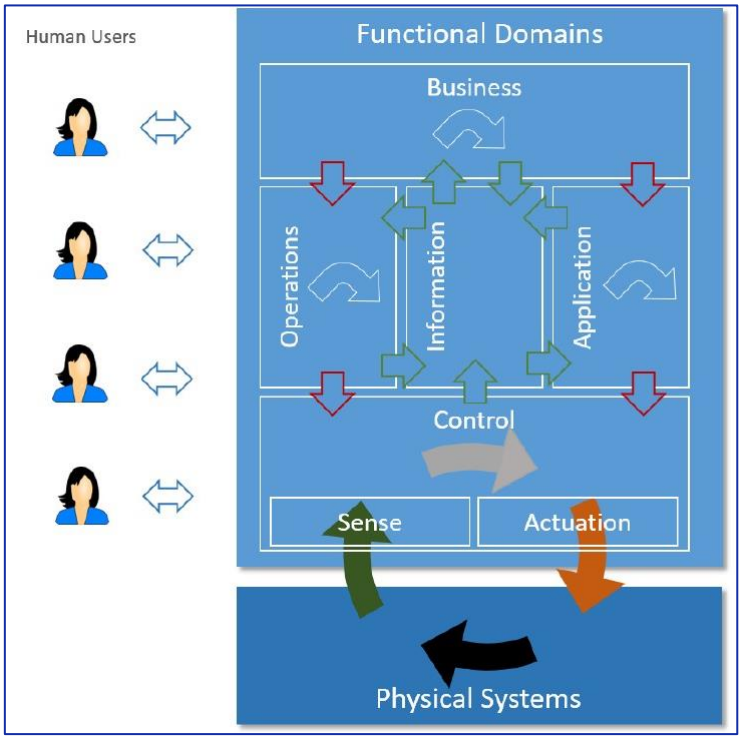


The IIRA viewpoints are defined by analyzing the various IIoT use cases developed by the IIC and elsewhere, identifying the relevant stakeholders of IIoT systems, and determining the proper framing of concerns. These four viewpoints are (LIN; CRAWFORD; MELLOR, 2017):

- **Business:** The business viewpoint attends to the concerns of the identification of stakeholders and their business vision, values, and objectives in establishing an IIoT system in its business and regulatory context.
- **Usage:** The usage viewpoint addresses the concerns of expected system usage. It is typically represented as sequences of activities involving human or logical (e.g. system or system components) users that deliver its intended functionality in ultimately achieving its fundamental system capabilities
- **Functional:** The functional viewpoint focuses on the functional components in an IIoT system, their structure and interrelation, the interfaces and interactions between them, and the relation and interactions of the system with external elements in the environment, to support the usages and activities of the overall system.
- **Implementation:** The implementation viewpoint deals with the technologies needed to implement functional components (functional viewpoint), their communication schemes, and their lifecycle procedures.

Considering the Functional Viewpoint, a typical IIoT system is decomposed into five functional domains: control, operation, information, application, and business. These domains and their data flow are presented in Figure 11.

Figure 11: IIRA Functional Domains (LIN; CRAWFORD; MELLOR, 2017).



Green Arrows: Data/Information Flows; Grey/White Arrows: Decision Flows; Red Arrows: Command/Request Flows

The IIRA is at a level of abstraction that excludes architectural elements and requirements whose evaluation requires specificities only available in concrete systems (LIN; CRAWFORD; MELLOR, 2017). It does not describe all the architecture constructs as outlined in Figure 10.

3.1.3 WSO2's Reference Architecture

The WSO2's reference architecture, presented in Figure 12, consists of a set of components organized in layers and cross-cutting layers (FREMANTLE, 2014). The layers are

- **Device:** The bottom layer of the architecture is the device layer. Devices can be of various types, but to be considered as IoT devices, they must have some communications that either indirectly or directly attach to the Internet. Examples of devices are Arduino and Raspberry PI.
- **Communications:** The communication layer supports the connectivity of the devices. There are multiple potential protocols for communication between the devices and the cloud. Examples of communication protocols are HTTP/HTTP and MQTT.
- **Aggregation/Bus:** This layer that aggregates and brokers communications. It provides the following abilities: to support an HTTP server and/or an MQTT broker to talk to the devices; to aggregate and combine communications from different devices and to route communications to a specific device (possibly via a gateway); to bridge and transform

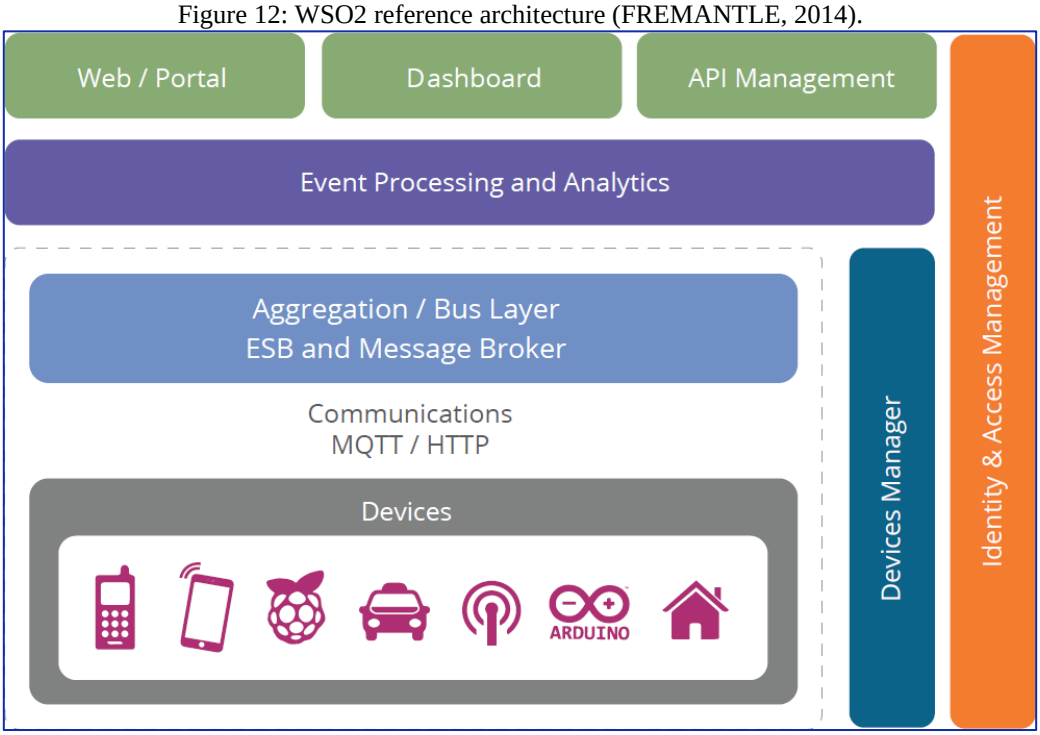
between different protocols, e.g. to offer HTTP-based APIs that are mediated into an MQTT message going to the device.

- **Event processing and analytics:** This layer takes the events from the bus and provides the ability to process and act upon these events. It is the layer related to the big data analytics platform.
- **Client/external communications:** The reference architecture needs to provide a way for these devices to communicate outside of the device-oriented system. This includes three main approaches: web/portal, dashboards, and API management. This layer is responsible for these approaches.

The cross-cutting layers are:

- **Device management:** This layer is handled by two components. A server-side system (the device manager) communicates with devices via various protocols and provides both individual and bulk control of devices. It also remotely manages software and applications deployed on the device. It can lock and/or wipe the device if necessary. The device manager works in conjunction with the device management agents. There are multiple different agents for different platforms and device types.
- **Identity and access management:** This layer is responsible for identification and access management. It provides the following services: OAuth2, SAML2 SSO, XACML, LDAP, etc.

Figure 12: WSO2 reference architecture (FREMANTLE, 2014).



This reference architecture highlights the scalability and security requirements, mapping it to components of the WSO2 platform. The WSO2 platform is based on a technology called WSO2 Carbon, which is in turn based on OSGi. Each product in the platform shares the same kernel based on Carbon. In addition, each product is made from features that are composed to provide the required functionality (FREMANTLE, 2014). Thus, although the requirements are identified, they are not evaluated and it is not clear how to instantiate them differently from the technologies provided by the WSO2 platform.

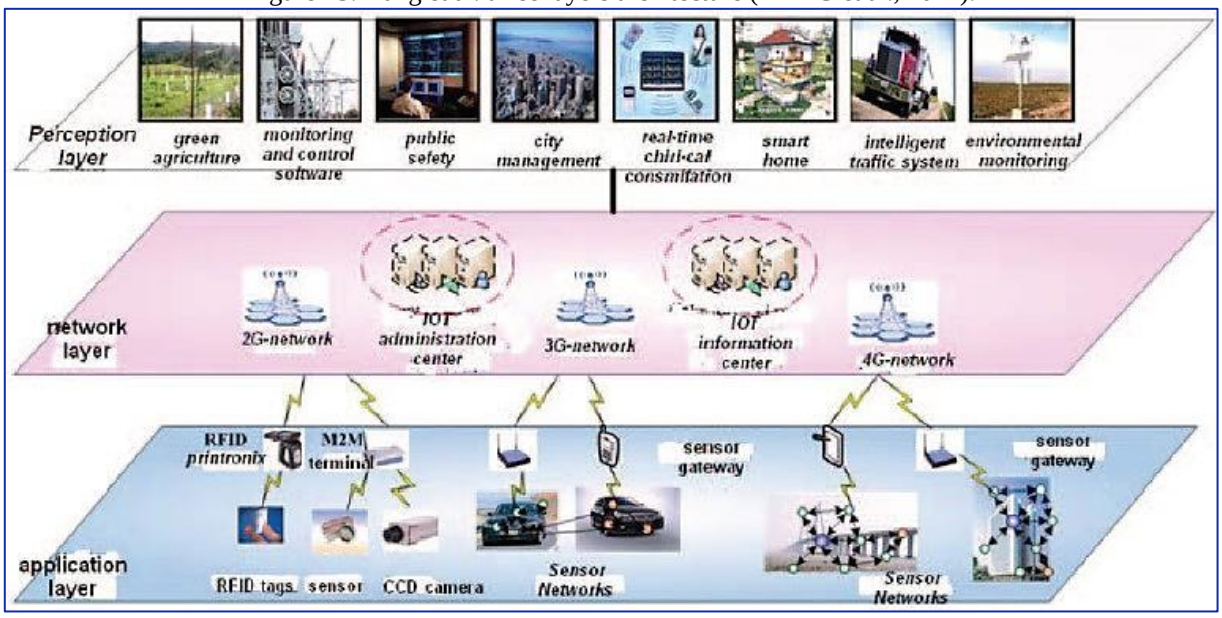
3.2 IOT ARCHITECTURES

There are many architectures found in the literature for IoT applications. The major architectures are proposed by Yang et al. (YANG et al., 2011), Gubbi et al. (GUBBI et al., 2013), Tan and Wang (TAN; WANG, 2010), Atzori et al. (ATZORI; IERA; MORABITO, 2010), and Wu et al. (WU et al., 2010). Although they are too abstract, they define important layers and components regarding IoT applications in various domains. In the following sections, these architectures will be presented.

3.2.1 Three Layers Architectures

The architecture proposed by Yang et al. (YANG et al., 2011), presented in Figure 13 is a three-layer architecture composed of the following layers: perception, network, and application. The perception layer's main task is to collect and process information. It consists of a traditional wireless sensor network, WSN radio frequency identification, RFID, and the final controlling element.

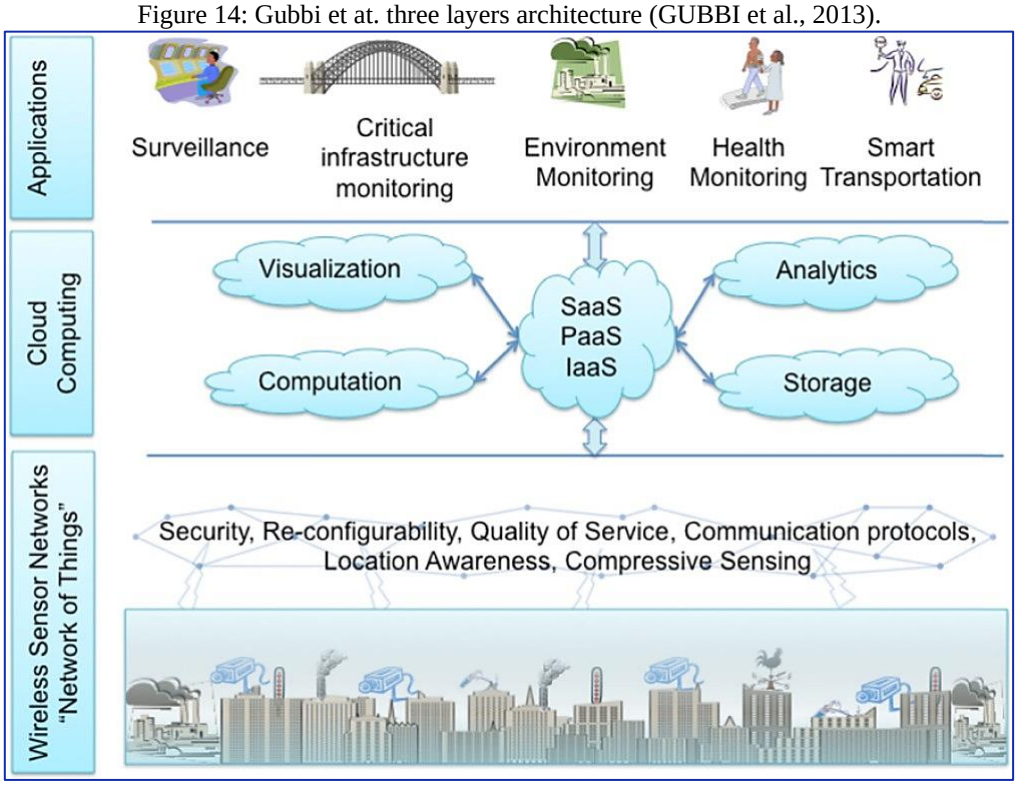
Figure 13: Yang et al. three layers architecture (YANG et al., 2011).



IoT's network layer is established based on the current mobile telecommunication and Internet. Its main feature is to convey information over a long distance. The network layer comprises various communication networks and integrated networks based on the Internet, which is generally regarded as the most mature part. The main task of the application layer is to provide services. The application layer is a connection of IOT technologies and sector professional technologies and a layer to realize the wide intelligent application by providing various solutions (GUBBI et al., 2013).

Another three-layer architecture, presented in Figure 14, is proposed by Gubbi et al. (GUBBI et al., 2013). This architecture is similar to the architecture proposed by Yang et al. (YANG et al., 2011). The wireless sensor networks ("network of things") layer has the same responsibility as the perception layer. The cloud computing layer has the same responsibility as the network layer but with the use of cloud computing concepts (SaaS, IaaS, and PaaS). The application layer contains the IoT applications and services.

Figure 14: Gubbi et at. three layers architecture (GUBBI et al., 2013).



3.2.2 Middleware-Based Architectures

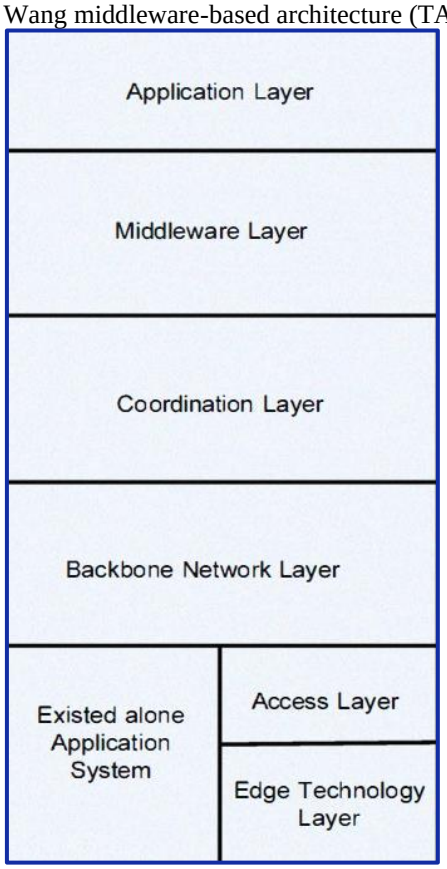
The architecture proposed by Tan and Wang (TAN; WANG, 2010), presented in Figure 15, is a middleware-based architecture composed of six layers. The access and edge technology layers are responsible for the intelligent objects (things) and how to access them. The backbone network layer allows the things to connect to the Internet. The coordination layer responds to process the structure

of packages from different application systems and reassemble them to a unified structure that can be identified and processed by every application system (TAN; WANG, 2010). The middleware layer allows the interaction between objects and services with different hardware specificities. Finally, the application layer is responsible for executing applications for the users.

Another architecture, presented in Figure 16, is proposed by Atzori et al. (ATZORI; IERA; MORABITO, 2010). This architecture is an Server Oriented Architecture (SOA) middleware composed of the following layers: applications, service composition, service management, object abstraction, objects, management of trust, privacy, and security. The applications layer is on the top of the architecture, exporting all the system's functionalities to the final user.

The service composition layer is on top of a SOA-based middleware architecture. It provides the functionalities for the composition of single services offered by networked objects to build specific applications. On this layer, there is no notion of devices and the only visible assets are services. The service management layer provides the main functions that are expected to be available for each object and that allow for their management in the IoT scenario. A basic set of services encompasses object dynamic discovery, status monitoring, and service configuration (ATZORI; IERA; MORABITO, 2010).

Figure 15: Tan and Wang middleware-based architecture (TAN; WANG, 2010).

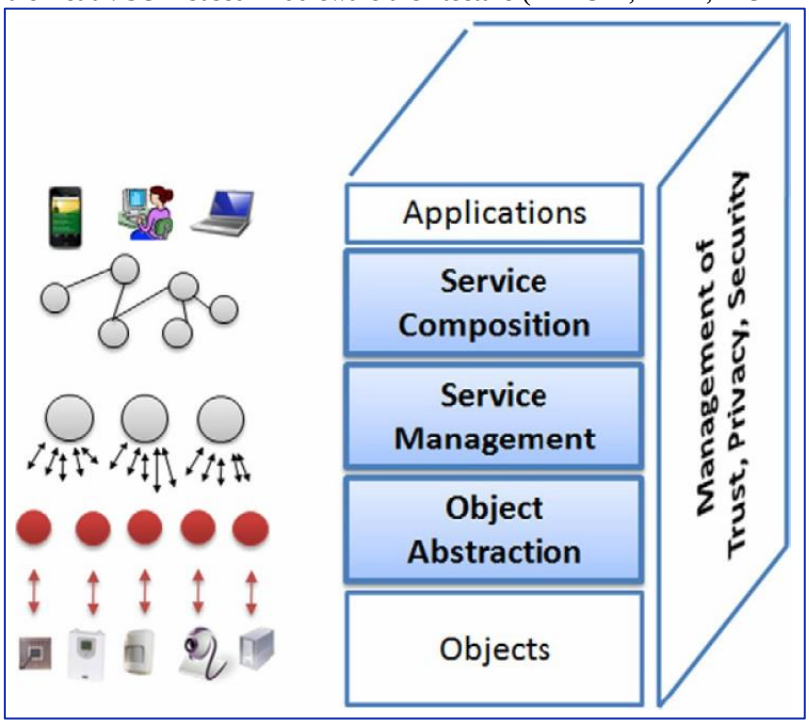


The IoT relies on a vast and heterogeneous set of objects, each one providing specific functions accessible through its dialect. There is thus the need for an abstraction layer capable of harmonizing the access to the different devices with a common language and procedure (ATZORI; IERA; MORABITO, 2010). Finally, the layer of management of trust, privacy, and security is responsible for the security and privacy of the data exchanged between the other layers.

3.2.3 Five Layers Architecture

The architecture proposed by Wu et al. (WU et al., 2010), presented in Figure 17, is composed of the following five layers: business, application, processing, transport, and perception. The perception layer is responsible for perceiving the physical properties of objects (such as temperature, location, etc.) by various sensors (such as infrared sensors, RFID, 2-D barcode), and converting this informations to digital signals which is more convenient for network transmission.

Figure 16: Atzori et al. SOA-based middleware architecture (ATZORI; IERA; MORABITO,2010).

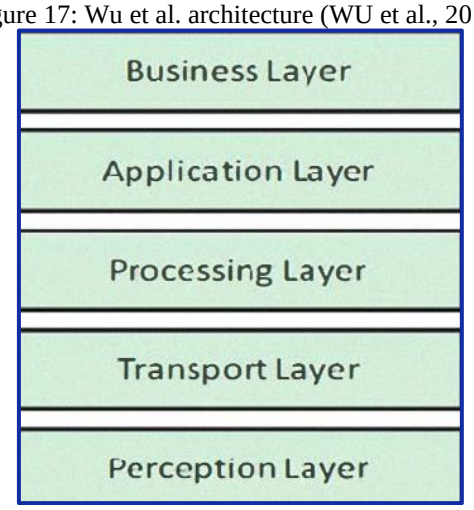


The Transport Layer (or Network Layer) is responsible for transmitting data received from the Perception Layer to the processing center through various networks, such as wireless or cable networks, even the enterprise Local Area Network (LAN). The Processing Layer mainly stores, analyzes, and processes the information of objects received from the transport layer.

The task of the Application Layer is based on the data processed in the Process Layer and develops diverse applications of the Internet of Things, such as intelligent transportation, logistics management, identity authentication, location-based service (LBS), safety, etc.

The Business Layer is a manager of the Internet of Things, including managing the applications, the relevant business model, and other business. The Business Layer not only manages the release and charging of various applications but also the research on the business model and profit model (WU et al., 2010).

Figure 17: Wu et al. architecture (WU et al., 2010).



3.3 DISCUSSION

As presented in the previous sections, IoT RA, IIRA, and WSO2 RA are reference architectures for IoT. These reference architectures work with concepts, such as IoT domain and IoT service, trying to address as many IoT application scenarios as possible. The IoT RA presents the need to address the following quality attributes: interoperability, availability, security, performance, and scalability. However, as these attributes require some kind of quality for a real system, they are not evaluated or it is not presented what are the components that address them. IIRA is at a level of abstraction that excludes architectural elements and requirements whose evaluation requires specificities only available in concrete systems, not presenting what are the addressed quality attributes. The WSO reference architecture highlights the scalability and security quality attributes mapping them into proprietary components of the WSO2 platform.

Moreover, it was not found examples of how to instantiate these reference architectures into concrete architectures, and their evaluation. On the other hand, the previous sections also presented IoT architectures with three layers, middleware-based, and five layers. Although these architectures have definitions of layers' responsibilities, they are too abstract and focused on general scenarios of

IoT-based applications. There are no definitions of components or how they address requirements and quality attributes.

Finally, the idea to address as many IoT application requirements and scenarios as possible without specifying the quality attributes required for the IoT-based healthcare applications or the components that address these requirements makes it difficult to use these reference architectures as guidelines for the development of these applications. In this context, even though there are existing reference architectures for IoT-based applications, they are too general and abstract and do not focus on IoT-based healthcare applications. Therefore, currently, to the best of our knowledge, there is no reference architecture for these specific applications.

3.4 FINAL REMARKS

This chapter presented the related works, providing a review of existing architectures and reference architectures for the Internet of Things (IoT). Thus, the described reference architectures were IoT RA, IIRA, and WSO2 RA. The described architectures were three, five layers, and middleware-based architectures. In this chapter, it was possible to note that, to best the knowledge, there is no reference architecture for IoT-based healthcare applications. Finally, the next chapter will present the state-of-the-art of IoT-based healthcare applications.

Before the proposal of the software reference architecture for IoT-based healthcare applications, it is essential to understand the state-of-the-art of this area and to realize that, it was performed a study based on Systematic Mapping Study (SMS) methodology (PETERSEN et al., 2008). According to Wohlin et al. (WOHLIN et al., 2012), SMS searches a broader field for any kind of research, to get an overview of the state-of-art or state-of-practice on a topic. It follows the same principled process as Systematic Literature Reviews (KITCHENHAM; CHARTERS, 2007), but it has different criteria for inclusions/exclusions and quality. Due to its broader scope and varying types of studies, the collected data and the synthesis tend to be more qualitative than for Systematic Literature Reviews (WOHLIN et al., 2012).

Therefore, this chapter presents a study based on the SMS methodology that was performed aiming to comprehend the current state and future trends for IoT-based healthcare applications, and also to find areas for further investigations. This Chapter is structured as follows: in section 4.1, it is presented the method for this study, focusing on the research questions, search process, inclusion and exclusion criteria, quality assessment, and data collection. Continuing, in section 4.2, it is presented the results for this method, regarding search results, quality evaluations, and factors. Section 4.3 presents the discussion about the results, and in section 4.4, the conclusions and future works of this research are presented.

4.1 METHOD

This study has been undertaken as a systematic mapping study based on the guidelines proposed by Petersen et al. (PETERSEN et al., 2008). In this case, the goal of the study is to comprehend the current state and future trends in IoT-based healthcare applications. The steps in the systematic mapping study method are documented in the following subsections.

4.1.1 Research Questions

Considering the context of IoT-based healthcare applications, the research questions addressed by this study are:

- **RQ1:** What are the main characteristics of healthcare applications based on IoT infrastructure?
- **RQ2:** What are the protocols used in healthcare applications based on IoT infrastructure?
- **RQ3:** What are the challenges related to healthcare applications based on IoT infrastructure?

Regarding RQ1, about the characteristics of healthcare applications, it intends to analyze the functional and nonfunctional requirements, and for which area of healthcare the applications are intended.

4.1.2 Search Process

The study selection was made on Scopus from Elsevier1, as it indexes the main sources of computing in the academic area. The examples of sources indexed by Scopus are presented in Table 1.

Table 1: Example of sources indexed by Scopus.

Source	Link
ACM Digital Library	http://dl.acm.org
explorer	http://ieeexplore.ieee.org
Science Direct	http://www.sciencedirect.com
Springer Link	http://link.springer.com

To define the search string, it was used terms related to health and the Internet of Things (IoT). The main goal was to obtain a major number of research on this particular application. Thus, the defined search string was: ("Internet of Things" OR "IoT") AND health.

4.1.3 Inclusion and exclusion criteria

This review included works published in any year because it was intended to find the biggest number of research regarding the development of healthcare applications based on IoT infrastructure. The duplicated works, those that do not present IoT-based healthcare applications, and, those that the researchers did not have access to were excluded from this review.

4.1.4 Quality assessment

Each selected study was evaluated according to the following quality assessment (QA) questions:

- **QA1:** Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
- **QA2:** Is there a clear statement of the aims of the research?
- **QA3:** Is there an adequate description of the context in which the research was carried out?
- **QA4:** Is the study of value for research or practice?

[1] http://scopus.com

- **QA5.** Is there a clear statement of findings?

These criteria were based on Dyba and Dingsoyr (DYBÅ; DINGSØYR, 2008) and they are grounded in three points that need to be addressed in the appreciation of the studies of the literature review:

- **Rigour:** Has a thorough and appropriate approach been applied to key research methods in the study?
- **Credibility:** Are the findings well-presented and meaningful?
- **Relevance:** How useful are the findings to the software industry and the research community?

These five quality assessment questions were scored as follows: 0 - in case of not attending the criteria; 0.5 - in case of partially attending the criteria; and 1 - in case of fully attending the criteria.

4.1.5 Data collection

The data extracted from each study were: authors' country, publication year, venue (journal of the conference), goal, app characteristics, functional requirements, nonfunctional requirements, transfer protocols, formatting pattern, IoT platform, and defined ontologies. Communication protocols, application domain, hardware, interoperability with other systems, application deployment, challenges, and additional comments.

4.2 RESULTS

This section summarizes the results of this study. It specifies each stage of its execution and also presents an overview of the papers that were useful for answering the research questions. Finally, it describes the quality evaluation results of the read studies.

4.2.1 Search Results

With the execution of the search with the string described in Section 4.1.2 at Scopus (stage 1), 1355 papers were retrieved. It was performed the analysis of the titles and abstracts of each one of them (stage 2). After the analysis, 46 papers presented in Table 2 remained. Finally, it was performed a careful read of these 46 papers, and 33 of them were useful in answering the proposed research questions (stage 3). Figure 18 presents these stages of the study selection process. The results of the extraction of the 46 studies are presented in https://goo.gl/skZmns.

Figure 18: Stages of the paper's selection process.

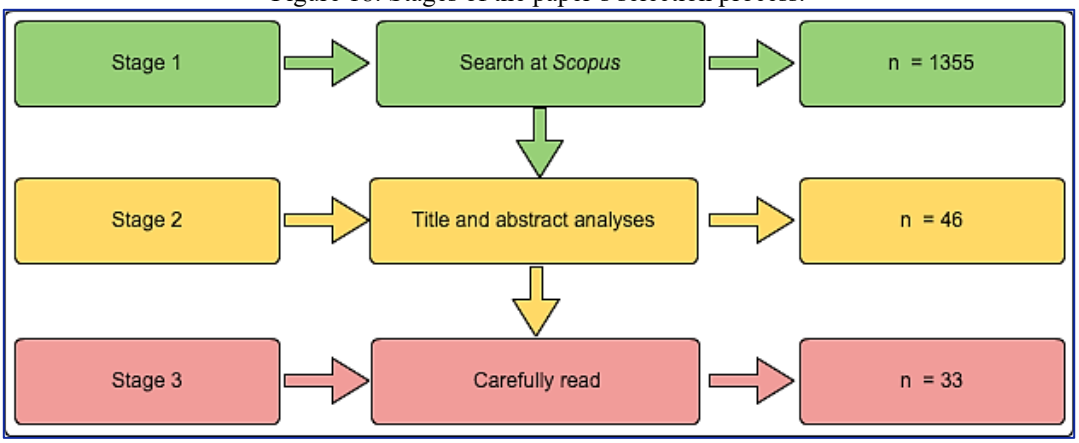


Table 2: The 46 carefully read papers of stage 2.

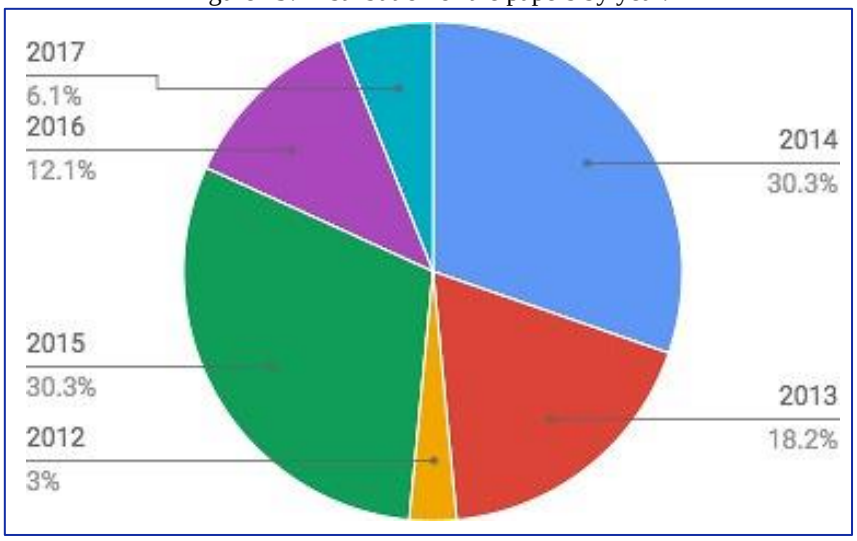
Id	Authors	Year	Venue
S1	Geng Yang et al (YANG et al., 2014a)	2014	Conference
S2	Antonio J. Jara et al (JARA; ZAMORA IZQUIERDO; SKARMETA, 2013)	2013	Journal
S3	Yuan Jie Fan et al (FAN et al., 2014)	2014	Conference
S4	Pedro Castillejo et al (CASTILLEJO et al., 2013)	2013	Journal
S6	Charalampos Doukas and Ilias Maglogiannis (DOUKAS; MAGLOGIANNIS, 2012)	2012	Conference
S7	Elena Poenaru and Calin Poenaru (POENARU; POENARU, 2013)	2013	Conference
S8	Chao-Tung Yang et al (YANG et al., 2013)	2013	Conference
S9	Pawel Swiatek and Andrezej Rucinsky (SWIATEK; RUCINSKI, 2013)	2013	Conference
S10	Pablo Lopez et al (LÓPEZ et al., 2013)	2013	Conference
S11	Denis Trcek and Andrej Brodnik (TRCEK; BRODNIK, 2013)	2013	Journal
S12	Fang Hu et al (HU; XIE; SHEN, 2013)	2013	Conference
S13	Yannick Le Moullec et al (MOULLEC et al., 2014)	2014	Conference
S14	Junaid Mohammed et al (MOHAMMED et al., 2014)	2014	Conference
S18	Lin Yang et al (KEVIN et al., 2014)	2014	Conference
S19	Gheorghe Sebestyen et al (SEBESTYEN et al., 2014)	2014	Conference
S20	Mohammad Mehedi Hassan et al (HASSAN; ALBAKR; AL-DOSSARI, 2014)	2014	Conference
S21	Iuliana Chiuchisan et al (HASSAN; ALBAKR; AL-DOSSARI, 2014)	2014	Conference
S24	Hasan Ali Khattak et al (KHATTAK et al., 2014)	2014	Conference
S25	Antonio J. Jara et al (JARA; ZAMORA; SKARMETA, 2014)	2014	Journal
S26	Rohan Tabish et al (TABISH et al., 2014)	2014	Conference
S27	Tuan Nguyen Gia et al (GIA et al., 2014)	2014	Conference
S28	Partha Pratim Ray (RAY, 2015)	2015	Conference
S29	Muhammad Wasim Raad (RAAD; SHELTAMI; SHAKSHUKI, 2015)	2015	Conference
S30	Ruiling Gao et al (GAO et al., 2015)	2015	Conference
S34	Steven van der Valk et al. (VALK et al., 2015)	2015	Conference
S35	Mirjana Maksimovic et al (MAKSIMOVIĆ; VUJOVIĆ; PERIŠIĆ, 2015)	2015	Conference
S37	Wan Abdullah et al (YAAKOB et al., 2016)	2016	Conference
S38	Jemal Abawajy and Mohammad Hassan (ABAWAJY; HASSAN, 2017)	2017	Journal
S39	JMin Chen et al (CHEN et al., 2017)	2017	Journal
S40	Zhe Yang et al (YANG et al., 2016)	2016	Journal
S41	Alexandru Archip et al (ARCHIP et al., 2016)	2016	Conference

S42	Ravi Kodali et al (KODALI; SWAMY; LAKSHMI, 2015)	2015	Conference
S43	Tuan Gia et al (GIA et al., 2015)	2015	Conference
S44	Majid Al-Taee et al (AL-TAEE et al., 2015)	2015	Conference
S45	Soumya Datta et al (DATTA et al., 2015)	2015	Conference
S46	Shamim Hossain and Ghulam Muhammad (HOSSAIN; MUHAMMAD, 2016)	2016	Journal

4.2.2 Papers Overview

Considering the venue (journal or conference) of the selected papers, 72.7% are from conferences and 27.3% are from journals. Moreover, 6.1% of these papers are from 2017, 12.1% are from 2016. Only 3% are from 2012 and it was not found IoT-based healthcare applications before 2012. It is believed that this is because of the maturity of the Internet of Things area, and the not inclusion of the terms used before "IoT" in the search string, such as cyber-physical systems. Figure 19 presents this distribution of the selected papers by year.

Figure 19: Distribution of the papers by year.



Regarding the applications described in the papers, 63.7% of the papers do not specify where the healthcare application is deployed. Considering the other 36.3%, 12.1% present solutions deployed at hospitals, and 24.7% deployed at home. Moreover, these studies describe that the main characteristics of these healthcare applications are the body and ambient monitoring. From the applications presented, only two studies, S6 and S34, presented the use of IoT Platforms, in this case, the ThingSpeak Platform2. Another observation is that seven of them define ontologies, they are S2, S3, S4, S10, S19, S25, and S45. One important point of these applications is that only S1 and S2 present interoperability with other systems, in the case of S1, with the medical supply chain,

[2] https://thingspeak.com/

emergency center, and hospital, and S2 with clinical devices. So, the consequence is that the use of most of the presented healthcare applications in 93% of the selected papers would demand a change in the existing systems of the hospitals.

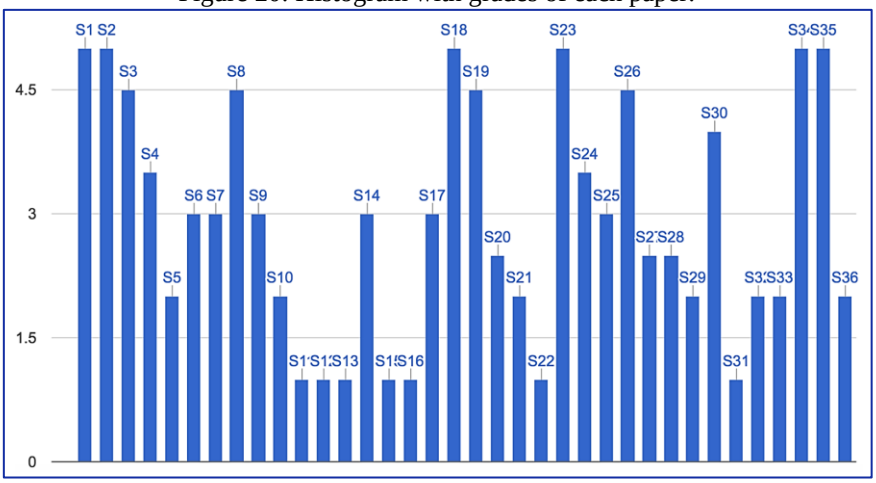
4.2.3 Quality Evaluation Results

The papers were evaluated using the criteria described in section 4.1.4. The score of each paper is presented in https://goo.gl/skZmns. The results show that all studies scored more than 1, and only 7 of them had the maximum score (5): S1, S2, S18, S34, S35, S40 and S41. Figure 20 presents the histogram of the grades of each paper.

4.3 DISCUSSION

In this section, it is discussed the answers to the research questions and then, it is presented the limitations and conclusions of this study.

Figure 20: Histogram with grades of each paper.



4.3.1 What are the main characteristics of healthcare applications based on IoT infrastructure?

Regarding the main characteristics of healthcare applications based on IoT infrastructure, it was collected their functional and non-functional requirements from the papers. The functional requirements described in the papers are related to the patient's body and environment monitoring. Considering the body monitoring, the data monitored by sensors attached to the patient's body are the pulse oximeter, heart rate, galvanic skin, transpiration, muscle activity, body temperature, oxygen saturation, blood pressure, airflow, body movement, blood glucose, breathing rate, and ECG. Moreover, environment monitoring is about sensors deployed in the patient's environment that capture data from temperature, light, humidity, location, body position, motion data, SPO2, atmospheric pressure, and CO2. Table 3 presents the papers and the patient's body and environment

data captured by the IoT-based healthcare applications, and Figure 21 presents a word cloud regarding it. It can be noted that the most frequently captured data of the IoT-based healthcare applications are related to ECG, body temperature, heart rate, and blood pressure.

Figure 21: Word cloud of body and environment captured data of the IoT-based healthcare applications.

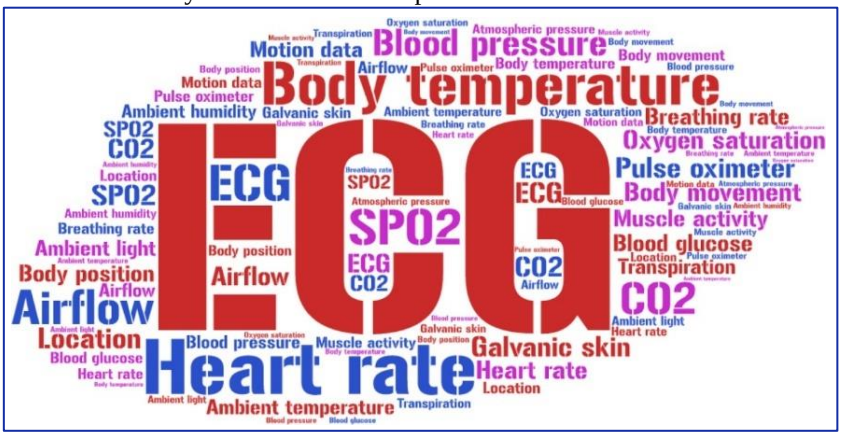


Table 3: Patient's body and environment data captured by the IoT-based healthcare applications and the papers.

Data	Freq.	Papers
ECG	22	S1, S2, S6, S7, S10, S14, S19, S20, S26, S27, S28, S29, S30, S35, S37, S38, S39, S40, S41, S42, S43, S46.
Body temperature	17	S1, S2, S4, S6, S7, S19, S20, S21, S24, S26, S28, S34, S35, S39, S41, S42, S45.
Heart rate	11	S2, S4, S6, S18, S21, S24, S27, S29, S34, S39, S46.
Blood pressure	9	S2, S8, S19, S20, S22, S28, S29, S35, S44.
Oxygen saturation	6	S6, S19, S21, S24, S29, S39.
Ambient temperature	5	S6, S8, S21, S30, S34.
Body movement	4	S28, S30, S34, S37.
SPO2	4	S27, S35, S41, S42.
Pulse oximeter	3	S24, S28, S35.
Breathing rate	3	S4, S21, S27.
Muscle activity	2	S19, S35.
Galvanic skin	2	S34, S35.
Blood glucose	2	S7, S36.
Ambient humidity	2	S8, S21.
Airflow	2	S19, S35.
Body position	2	S19, S35.
Motion data	2	S6, S21.
CO2	1	S8.
Transpiration	1	S19.
Ambient light	1	S30.
Location	1	S6.
Atmospheric pressure	1	S21.

About the features of IoT-based healthcare applications, there are some important nonfunctional requirements (quality attributes) that represent a concern in this kind of application. The non-functional requirements cited by the papers are scalability, reliability, ubiquity, portability, interoperability, robustness, performance, availability, privacy, integrity, authentication, and security.

Table 4 specifies the nonfunctional requirements in the studies, and Figure 22 presents a word cloud regarding it. It can be noted that the most cited nonfunctional requirements are security, interoperability, reliability, and privacy.

Finally, it can be concluded that the main characteristics of IoT-based healthcare applications in terms of functional requirements are the patient's body and environment monitoring, with the main capture of data from ECG, body temperature, heart rate, and blood pressure. Concerning nonfunctional requirements, the most important are security, interoperability, reliability, and privacy.

Table 4: Nonfunctional requirements of IoT-based healthcare applications and the papers.

NFR	Freq.	Papers
Security	13	S2, S3, S6, S9, S10, S14, S19, S20, S28, S30, S38, S45, S46.
Interoperability	10	S2, S3, S4, S6, S24, S27, S35, S38, S43, S45.
Reliability	8	S2, S9, S20, S27, S30, S35, S37, S40.
Privacy	8	S2, S6, S14, S19, S20, S28, S35, S38.
Scalability	6	S2, S6, S14, S39, S44, S46.
Availability	4	S2, S6, S9, S45.
Performance	2	S14, S20.
Authentication	2	S35, S46.
Ubiquity	1	S10.
Portability	1	S14.
Robustness	1	S2.
Integrity	1	S35.

Figure 22: Word cloud of nonfunctional requirements of IoT-based healthcare applications.

4.3.2 What are the protocols used in healthcare applications based on IoT infrastructure?

Concerning protocols, the collected data of the papers showed that there are two categories of protocols: communication, regarding network protocols, and application, regarding data transfer protocols. The communication protocols cited by the studies on healthcare applications are 6LoWPAN, IEEE 802.15.4, Zigbee, Bluetooth, RFID, WIFI, Ethernet, GPRS, IEEE 802.15.6, 3G/4G, NFC, and IrDA. Regarding the application protocols, the studies cited: REST, YOAPY, HTTP, CoAP, XML-RPC, and Web Services. Table 5 presents the communication protocols and the papers, and

Figure 23 presents a word cloud regarding it. It can be noted that the most used communication protocols are Bluetooth, WIFI, 6LoWPAN, Zigbee, and 3G/4G.

Table 5: Communication protocols of IoT-based healthcare applications and the papers.

Com. Protocols	Freq.	Papers
Bluetooth	19	S1, S2, S4, S6, S10, S14, S18, S19, S20, S28, S34, S35, S38, S39, S40, S43, S44, S45, S46.
WIFI	17	S1, S3, S6, S19, S21, S29, S31, S34, S35, S37, S38, S39, S40, S43, S44, S45, S46.
6LoWPAN	11	S2, S10, S17, S21, S24, S25, S26, S27, S28, S30, S43.
Zigbee	11	S1, S2, S4, S8, S18, S28, S35, S40, S42, S43, S45.
3G/4G	10	S1, S7, S20, S26, S31, S35, S37, S38, S40, S44.
RFID	7	S1, S2, S3, S18, S25, S29, S37.
IEEE 802.15.4	6	S4, S7, S9, S26, S35, S41.
GPRS	3	S21, S35, S40.
NFC	2	S10, S25.
Ethernet	2	S1, S21.
IEEE 802.15.6	1	S7.
IrDA	1	S25.

Figure 23: Word cloud of communication protocols of IoT-based healthcare applications.

Table 6 presents the application protocols and the papers, and Figure 24 presents a word cloud regarding it. It can be noted that the most used application protocols are REST, HTTP, and CoAP.

Regarding the data format, the studies presented that the healthcare applications use HL7, XML, EHR, CSV, JSON, and PHR. Table 7 presents the data format and the papers, and Figure 25 presents a word cloud regarding it. It can be noted that the most used are JSON, XML, HL7, and EHR.

Table 6: Application protocols of IoT-based healthcare applications and the papers.

Com. Protocols	Freq.	Papers
REST	7	S4, S6, S14, S20, S35, S41, S45.
HTTP	5	S34, S35, S40, S44, S45.
CoAP	4	S2, S24, S30, S45.
Web services	2	S27, S46.
YOAPY	1	S2.
XML-RPC	1	S9.

Figure 24: Word cloud of application protocols of IoT-based healthcare applications.

4.3.3 What are the challenges related to healthcare applications based on IoT infrastructure?

The papers presented that there are many challenges related to healthcare applications based on IoT infrastructure. In S6, the authors presented that health information management through mobile devices introduces several challenges: data storage and management (e.g., physical storage issues, availability, and maintenance), interoperability and availability of heterogeneous resources, security and privacy (e.g., permission control, data anonymity, etc.), unified and ubiquitous access are a few to mention. According to S6, the vast amount of sensor data generated by the capture of these applications needs to be managed properly for further analysis and processing. Another challenge regarding the data is the unstructured format, according to S14, the huge volume of data produced by the sensors is in an unstructured format, which is very complex to understand and requires different data storage mechanisms than the typical database management system (DBMS).

Table 7: Data format and the papers.

Data format	Freq.	Papers
JSON	9	S4, S6, S18, S24, S34, S35, S41, S44, S45.
XML	6	S6, S8, S18, S19, S27, S45.
HL7	3	S2, S8, S24.
EHR	3	S2, S25, S45.
CSV	2	S6, S34.
PHR	1	S25.
HTML	1	S40.

Figure 25: Word cloud of data format of IoT-based healthcare applications.

Still about challenges, in S18, the authors highlight that the existing home healthcare systems have drawbacks such as simple and few functionalities, weak interaction, and poor mobility, and IoT is considered an effective method for the healthcare monitoring system of the disabled and elderly people by the people-object interaction. Moreover, the authors, in S18, describe that their future work is focused on the wireless body area networks combined with social networks, exploring the mobility-impaired healthcare services based on social networking, and sharing the information of smart objects.

The authors in S19 describe interoperability, political, and administrative challenges since the communication protocol of the devices is not open and a given device cannot be integrated into other (or multiple) applications. Moreover, according to S19, the implementation of these applications is a technical as well as political and administrative challenge, as it implies not only a technical infrastructure but also several regulatory measures, such as standards, regulations, and institutional reorganization. Any regional or national implementation of such a system must fulfill not only quality and safety requirements but also economic efficiency conditions.

In S20, the authors present the need for the development of new protocols that are reliable and energy efficient in data transmission, since routing protocols are critical for the system to work efficiently. In addition, they say that even though several protocols have been proposed for various domains, none of them has been accepted as a standard, and with the growing number of things, further research is required. Still, in S20, the authors also describe the need for the development of efficient data mining techniques for extracting useful knowledge from IoT data. Moreover, sometimes IoT-generated data are not always ready for direct consumption using visualization platforms, and, therefore, new visualization schemes need to be developed. Another key challenge described by the authors in S20 regards the need to protect private information. They say that more innovative solutions need to be developed in privacy and security aspects.

The authors in S24 highlight the interoperability challenge, once there are different studies and proposals for patient monitoring at the hospital or home for personal monitoring, a shared goal to produce an interoperable system adopting open standards for healthcare, for example, HL7 and a seamless framework to be easily deployed in any given scenario for healthcare is still missing. Table 8 presents a summary of the found challenges.

Table 8: Summary of the challenges of IoT-based healthcare applications.

Paper	Challenges
S6	Data store and management, interoperability, and availability of heterogeneous resources, security, and privacy, unified and ubiquitous access.
S14	The huge volume of data produced by the sensors in an unstructured format.
S18	The current home healthcare applications have few functionalities, weak interaction, and poor mobility.
S19	The interoperability, political, and administrative.
S20	Reliable protocols and energy efficient in data transmission. Efficient data mining techniques for extracting useful knowledge from IoT data, and privacy and security.
S24	Interoperability and a shared goal to produce an interoperable system adopting open standards for healthcare.

4.3.4 A technology view for IoT-based healthcare applications

Through the discussion of the results of this study, it was possible to define layers and organize the technologies used in IoT-based healthcare applications into them. The defined layers were users, requirements, systems and services, communication, middleware, monitoring, and patients. The users layer is composed of the users of IoT-based healthcare applications. They are physicians, hospital administrators, nurses, family, pharmaceutical, clinical staff, and patients.

The requirements cross-cutting layer is composed of nonfunctional requirements (quality attributes). These requirements are scalability, reliability, ubiquity, portability, interoperability, robustness, performance, availability, privacy, integrity, authentication, and security. This is a cross-cutting layer because of the importance of these requirements to the other layers. Another cross-cutting layer is the communication cross-cutting layer, that is composed of communication protocols. These protocols are 6LoWPAN, IEEE 802.15.4, Zigbee, Bluetooth, RFID, WI-FI, Ethernet, GPRS, IEEE 802.15.6, 3G/4G, NFC, and IrDA.

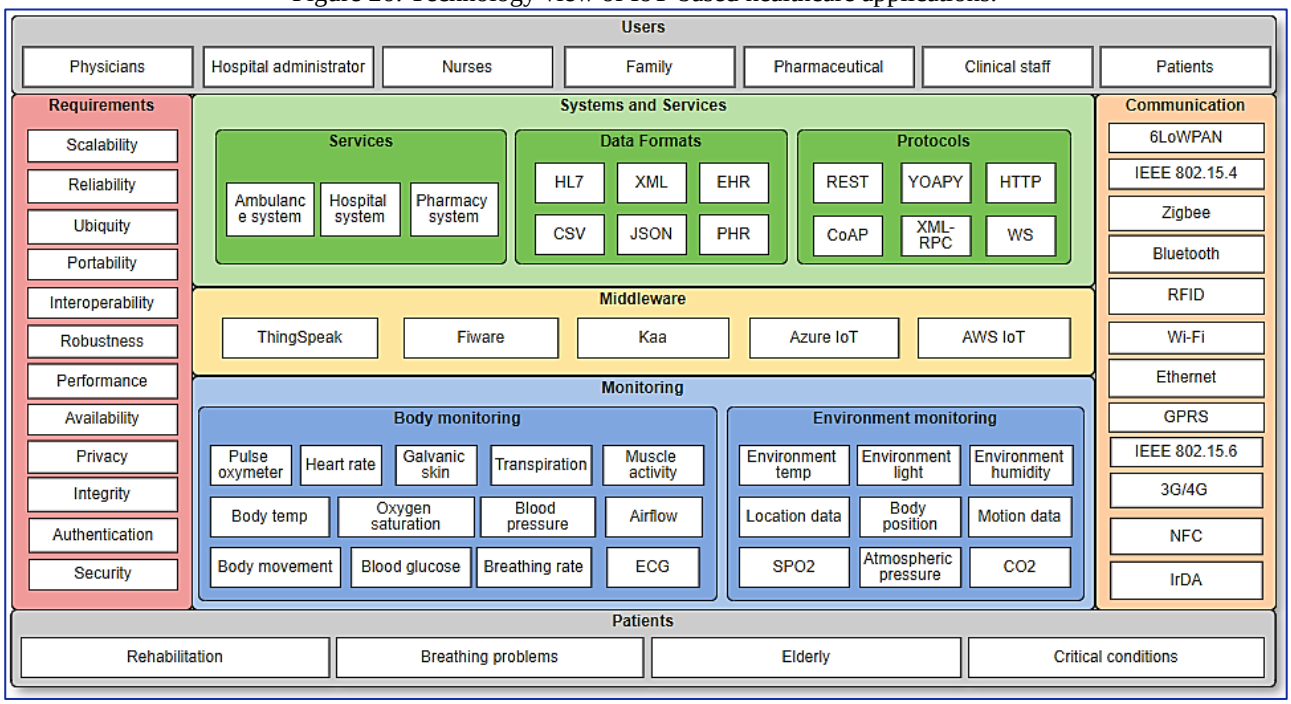
The patient layer is composed of the usual patients of IoT-based healthcare applications. They are usually patients in rehabilitation, with breathing problems, elderly, and with critical conditions. The monitoring layer is composed of body and environment monitoring. Body monitoring is related to pulse oximeter, heart rate, galvanic skin, transpiration, muscle activity, body temperature, oxygen saturation, blood pressure, airflow, body movement, blood glucose, breathing rate, and ECG. Environment monitoring is related to environment temperature, light and humidity, location data, body position, motion data, SPO2, atmospheric pressure, and CO2.

The middleware layer is composed of middleware, such as ThingSpeak, Fiware, Kaa, Azure IoT, and AWS IoT. The systems and services layer is composed of services, data formats, and application protocols. The services are ambulance, hospital, and pharmacy systems. The data formats are HL7, XML, EHR, CSV, JSON, and PHR. The application protocols are REST, YOAPY, HTTP, CoAP, XML-RPC, and web services. Finally, this technology view is presented in Figure 26.

4.3.5 Limitations of this study

The main limitation of this study is the bias in the selection of publications and inaccuracy in data extraction. However, it strictly followed the defined protocol, described in section 4.1, to ensure that the selection process was unbiased. Another limitation is the search string, described in section 4.1.2, although it was defined as guided by the research questions, there is a risk that some studies were omitted. Another limitation of this study is that it used Scopus from Elsevier to proceed with the search of the papers and, although it indexes other scientific repositories, inclusion in Scopus once a paper has been published takes some time, and so, there is a risk that some already published studies were not yet included. The final limitation of this study is that it did not consider the works realized by companies, such as patents, software, etc (gray literature).

Figure 26: Technology view of IoT-based healthcare applications.



4.4 FINAL REMARKS

This study was made aiming to comprehend the current state and future trends of healthcare applications based on IoT infrastructure, and also to find areas regarding it for further investigations. Started this study by defining the method with research questions, search process, quality assessments, and data collection. Then, was performed the search using the defined search string at Scopus from Elsevier (stage 1), resulting in 1355 papers. After this search, it was performed the analyses of the titles and abstracts of the papers (stage 2). Then, 46 papers remained in this study and they were carefully read (stage 3). For these 46 selected papers, were evaluated according to the quality assessment and 7 of them had the maximum score. Of these 46 selected papers, 33 papers were useful to answer the research questions.

Using the extraction data, it was possible to answer the research questions and provide the characteristics of healthcare applications based on IoT infrastructure (section 4.3.1). It also described the protocols and data formats used in the papers (section 4.3.2). Moreover, using the extracted data from studies, it was possible to find some challenges for healthcare applications (section 4.3.3). The challenges are related to the development of new solutions to resolve interoperability problems, data mining techniques for extraction of knowledge for IoT data, and privacy and security problems. There is also an industry opportunity for companies that develop IoT-based healthcare applications since the healthcare industry is estimated to be more than $2 trillion by 2020 with an annual consumer market for remote/mobile monitoring devices at $40 billion globally (POENARU; POENARU, 2013). Besides, it also defined a technology view for IoT-based healthcare applications (section 4.3.4) that relates the technologies used in the papers and the layers of these applications.

Finally, with this study, it was possible to define a layered architecture for healthcare applications based on IoT Infrastructure. It considers the characteristics of these applications, functional requirements, nonfunctional requirements (quality attributes), and used protocols, and is composed of a layer of monitoring, quality attributes, middleware, and services. In Chapter 5 this architecture will be described, and it will be used for the development of IoT-based healthcare applications that will address issues like security and interoperability.

As presented in Chapter 4, there are a lot of challenges related to the development and deployment of IoT-based healthcare applications, such as interoperability (DOUKAS; MAGLOGIANNIS, 2012) (KHATTAK et al., 2014) (SEBESTYEN et al., 2014), availability (DOUKAS; MAGLOGIANNIS, 2012), usability (KEVIN et al., 2014), security (DOUKAS; MAGLOGIANNIS, 2012), flexibility (EBERT et al., 2016), and productivity. Regarding interoperability, the overview of the papers presented in section 4.2.2, showed that 93% of the described new solutions would demand a change in the existing healthcare hardware and software.

Although there are many proposed protocols and different studies about IoT-based healthcare applications, as presented in Chapter 4, a shared goal to produce an interoperable system adopting open standards for healthcare, for example, HL7, and a seamless framework to be easily deployed in any given scenario for healthcare is still missing (KHAT- TAK et al., 2014). On the other hand, even though there are reference architectures for IoT-based applications, as presented in Chapter 3, they are too general and abstract and do not focus on IoT-based healthcare applications. The outcome of this situation is the development of independent IoT-based healthcare applications that do not interoperate and communicate with each other, making their deployment difficult in scenarios with existing healthcare solutions (hardware and software). With the perspective of expanding these applications market, and consequently the development of new solutions, this problem will grow significantly.

In this context, one of the possible causes for this lack of interoperability and communication between IoT-based healthcare applications is the absence of a software reference architecture (SRA) to serve as a guideline for the design of their architectures. SRA facilitates the development process, acting as a tool for standardization and making modular configuration and interoperability with IoT-based healthcare solutions from different suppliers possible. Furthermore, with an SRA, different vendors would be able to provide specific modules that can be integrated among themselves. Finally, its existence would provide a standardized view for these applications which promotes communication between the potential stakeholders (business professionals, software developers).

Therefore, in this book, it was designed a software reference architecture for IoT-based healthcare applications, which was named RAH, to serve as a guideline for the design of these applications' architectures. This SRA systematically organizes the main elements of these applications, their responsibilities, and interactions, promoting a common understanding of these applications' architecture and addressing the challenges of interoperability, security, performance, and availability related to their development. RAH is defined based on a set of functional and nonfunctional requirements (quality attributes) related to IoT-based healthcare applications. These

requirements were extracted from existing publications collected through the study presented in Chapter 4.

Some of the benefits of an SRA (MARTÍNEZ-FERNÁNDEZ et al., 2017) that are expected to achieve with RAH are: (i) standardizing concrete software architectures by using the SRA as a template for designing a portfolio of applications that use the standardized design; (ii) facilitating the design of concrete software architectures by providing guidelines and inspiration for the applications' developers; (iii) systematically reusing standard functionalities and configurations throughout the applications' development; (iv) reducing risks through the use of proven and partly pre qualified architectural elements included in the SRA; (v) enhancing quality by facilitating the achievement of software quality aspects already addressed by the SRA; (vi) allowing interoperability between different applications and their software components by establishing common mechanisms for information exchange; (vii) creating a knowledge repository, since SRA inherently acts as a repository of applied knowledge such as architectural and design principles; (viii) improving communication in the organization and with multiple suppliers since stakeholders share the architectural mindset established in the SRA. Preliminary, SRAs are usually designed to provide innovative design solutions concerning the existing state of the art.

Finally, in this chapter, it is presented the proposed software reference architecture for IoT-based healthcare applications. It is structured as follows: Section 5.1, describes the requirements of a reference architecture for IoT-based healthcare applications, specifying the requirements (functional and nonfunctional), and architecture qualities. Section 5.2 describes RAH, the software reference architecture for IoT-based healthcare applications. Finally, in Section 5.3, the final remarks of this chapter are presented. This chapter's organization was inspired by the structure defined by Angelov and Grefen to describe their software reference architecture (ANGELOV; GREFEN, 2008).

5.1 REQUIREMENTS OF A REFERENCE ARCHITECTURE FOR IOT-BASED HEALTHCARE APPLICATIONS

In this section, it is discussed the functional and nonfunctional requirements that must be addressed in RAH. The functional requirements express the functionalities that must be supported by an IoT-based healthcare application. The study presented in Chapter 4 was used to define these requirements. The quality attributes or nonfunctional requirements, in turn, are separated into two groups: applications' nonfunctional requirements and architecture qualities. The applications' nonfunctional requirements must be addressed in the development of an IoT-based healthcare application. The architecture qualities are directly related to the architecture itself and are important for the design of a reference architecture.

Therefore, to define the nonfunctional requirements of a reference architecture for IoT-based healthcare applications, it was used the list of quality attributes of information systems presented by Bass et al. (BASS; CLEMENTS; KAZMA, 2013), as well as the existing publications presented in Chapter 4. Based on this list of requirements, the first version of RAH was defined.

Finally, section 5.1.1 presents the functional requirements of IoT-based healthcare applications. In section 5.1.2, it is presented the quality attributes (nonfunctional requirements) of these applications, and in section 5.1.3, it is presented the architecture qualities for a software reference architecture.

5.1.1 Functional requirements

According to Bass et al., the functional requirements state what the system must do, and how it must behave or react to runtime stimuli (BASS; CLEMENTS; KAZMA, 2013). Thus, considering the evidence collected in the study presented in Chapter 4, the functional requirements of IoT-based healthcare applications consist of monitoring the patient's body and environment. Regarding body monitoring, the applications use sensors attached to the patient's body and capture data from:

1. **Electrocardiogram** (ECG) (DOUKAS; MAGLOGIANNIS, 2012) (JARA; ZAMORA-IZQUIERDO; SKARMETA, 2013) (YANG et al., 2014b) (MAKSIMOVIĆ; VUJOVIĆ; PERIŠIĆ, 2015) (YANG et al., 2016) (ABAWAJY; HASSAN, 2017): recording of the electrical activity of the heart in the form of specific waves. The ECG monitoring can be used to monitor the heart rate of a patient, assess the effects of an illness or injury on the function of the pacemaker, and evaluate the response after a physician's procedure. The ECG can give information about the orientation of the heart, conduction disturbances, electrical effects of medications and electrolytes, the mass of the heart muscle, and the presence of ischemic damage. However, to evaluate the effectiveness of the mechanical activity of the heart the pulse and blood pressure of the patient are evaluated (AEHLERT, 2012).
2. **Blood pressure (**JARA; ZAMORA-IZQUIERDO; SKARMETA, 2013) (RAAD; SHELTAMI; SHAKSHUKI, 2015) (MAKSIMOVIĆ; VUJOVIĆ; PERIŠIĆ, 2015): recorded as a ratio between two numbers, systolic - the top number, which is also the higher of the two numbers, measures the pressure in the arteries when the heart beats (when the heart muscle contracts); diastolic - the bottom number, which is also the lower of the two numbers, measures the pressure in the arteries between heartbeats (when the heart muscle is resting between beats and refilling with blood) (ASSOCIATION, 2017b). For example: read as 140X90 mmHg (millimeters of mercury). This measure can be decisive for a patient's life in the early identification of cardiac and vascular problems (VIDAL-PETIOT et al., 2016). In

cases of high pressure, it control reduces the risk of cardiovascular events and death (ZANCHETTI; THOMOPOULOS; PARATI, 2015).

3. **Blood glucose** (POENARU; POENARU, 2013): monitoring is the main tool you have to check patient diabetes control (ASSOCIATION, 2017a). Population data indicate that 30-40% of people with type 1 diabetes experience an average of 1 to 3 episodes of severe hypoglycemia each year. With self-monitoring and patient education and care, the patient may benefit from a controlled glycemic rate with individual goals set by the team of health professionals. During the last decade, the introduction of continuous glucose monitoring to facilitate self-administration has shown an improvement in glucose control and reduced exposure to hypoglycemia (BOLINDER et al., 2016). Experience shows the beneficial effect of continuous monitoring of blood glucose (THABIT; BALLY; HOVORKA, 2016).
4. **Heart rate** (VALK et al., 2015) (RAAD; SHELTAMI; SHAKSHUKI, 2015) (GIA et al., 2015) (KHATTAK et al., 2014) (YANG et al., 2016) (CHEN et al., 2017): the number of heartbeats per unit of time, usually per minute (ASSOCIATION, 2017b). The heartrate is based on the number of contractions of the ventricles (the lower chambers of the heart) (MEDICINET, 2017). Heart rate variability has been used as a noninvasive means of assessing the neural control of the heart and is used to identify hemodynamic problems.
5. **Oxygen saturation** (DOUKAS; MAGLOGIANNIS, 2012) (SEBESTYEN et al., 2014) (CHIUCHISAN; COSTIN; GEMAN, 2014) (RAAD; SHELTAMI; SHAKSHUKI, 2015): it is especially useful to detect hypoxemia associated with critical problems such as cardiovascular ones (EWER, 2014). The oxygen uptake occurs primarily in the lungs, constituting the first step in the process of oxygen to the tissues. The oxygen taken up in the lungs is transported in the blood in two ways: by dissolving in plasma and also combining with hemoglobin. Hemoglobin is capable of carrying 98-99% of all oxygen in the blood and can be viewed through the oxygen saturation measured by pulse oximetry. The arterial oxygen saturation is determined as a percentage, on average it is in the range of 95% to 100%. There may be some changes and false readings of oxygen saturation, which are usually caused by chills, hypotension, low perfusion, and edema (BAZERBASHI et al., 2014).
6. **Temperature** (RAY, 2015) (TABISH et al., 2014): the human being is homeothermic, i.e. can maintain body temperature within a certain predetermined range despite variations in the thermal environment - thermal homeostasis (GASPARRINI et al., 2015). Increased body temperature may indicate increased cell metabolism, consumption of O2 and CO2 production, demands on the heart and lungs, and additional stress to the cardiopulmonary system and infectious processes, and therefore may justify a continued investigation in critically ill

patients (CAHILL; PRENDERGAST, 2016). Measured in degrees Celcius (C), it is taken using a catheter close to the skin in the axillary region continuously (HALL, 2011).

7. **Breathing rate** (CASTILLEJO et al., 2013) (CHIUCHISAN; COSTIN; GEMAN, 2014): it is measured by the respiratory motion for one minute, measured in rpm. It demonstrates not only lung function but can denote problems in other systems, such as neurological and cardiac (CAHILL; PRENDERGAST, 2016).

When it comes to monitoring the environment, the applications use sensors deployed in the patient's environment to capture data from temperature, light, humidity, location, body position, motion data, SpO2, atmospheric pressure, and CO2. They are important because the environment measures can directly affect the patient's treatment.

5.1.2 Quality attributes

The quality attributes or nonfunctional requirements are qualifications of the functional requirements or the overall product. A qualification of a functional requirement is an item such as how fast the function must be performed, or how resilient it must be to erroneous input. Qualification of the overall product is an item such as the time to deploy the product or a limitation on operational costs (BASS; CLEMENTS; KAZMA, 2013). Thus, the main nonfunctional requirements/quality attributes of IoT-based healthcare applications evidenced in the study presented in Chapter 4 are:

- **Availability:** refers to a property that is found in the software when it is available to be used and ready to carry out its task when you need it to be (BASS; CLEMENTS; KAZMA, 2013). In the healthcare context, the availability quality can be decisive between a patient's life or death. This quality is also related to dependability, which is the ability to avoid failures that are more frequent and more severe than it is acceptable (BASS; CLEMENTS; KAZMA, 2013).
- **Interoperability:** it is related to the degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context. The definition includes not only having the ability to transfer data (syntactic interoperability) but also having the capacity to interpret the data exchanged correctly (semantic interoperability) (BASS; CLEMENTS; KAZMA, 2013). Interoperability is an important quality in the healthcare context but, as presented in Chapter 4, the existing applications usually do not interoperate with each other, thus creating isolated solutions.
- **Performance:** it is about time and the software system's ability to meet timing requirements (BASS; CLEMENTS; KAZMA, 2013). In the healthcare context, the information should

reach stakeholders as fast as possible. Moreover, the performance quality can also be decisive between a patient's life or death.

- **Security:** the measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized (BASS; CLEMENTS; KAZMA, 2013). Thus, this attribute is related to privacy, integrity, and authentication. In the healthcare context, the patient's data should be protected and available only to authorized and authenticated personnel.

5.1.3 Architecture qualities

In Bass et al. (BASS; CLEMENTS; KAZMAN, 2003), several architecture qualities are presented. Inspired by them and the quality attributes defined by Angelov and Grefen (ANGELOV; GREFEN, 2008), it was elaborated the following list of architectural qualities expected for a software reference architecture for IoT-based healthcare applications:

- **Completeness:** it is essential for the architecture to allow all of the system's requirements and runtime resource constraints to be met (BASS; CLEMENTS; KAZMAN, 2003). Thus, completeness is necessary for RAH, as it has to serve as a guiding model for designing concrete software architectures of IoT-based healthcare applications regardless of the business and technological context. A quality closely related to completeness is the scope (ANGELOV; GREFEN, 2008). A reference architecture for IoT-based healthcare applications must give a clear scope description of the business aspects that it addresses.
- **Buildability:** an architecture specification must be implementable (buildable), preferably in an easy and timely manner. Another aspect of buildability is the knowledge about the problem to be solved (BASS; CLEMENTS; KAZMAN, 2003). Further - more, being a reference architecture, RAH must have a clear structure and coherent design (conceptual integrity).
- **Applicability** (ANGELOV; GREFEN, 2008): RAH must be able to be applied for the design of a new IoT-based healthcare application as well as for the analysis of existing applications. Thus, RAH must be applicable in different kinds of IoT-based healthcare applications.
- **Usability** (ANGELOV; GREFEN, 2008) (BASS; CLEMENTS; KAZMA, 2013): to be successfully adopted for the development of IoT-based healthcare applications, RAH must be easy to understand by both business and IT professionals. It must foster communications between the IoT-based healthcare applications' stakeholders. Furthermore, architecture designers should be able to use RAH as a starting step in the design of concrete IoT-based healthcare architectures.

5.2 RAH - REFERENCE ARCHITECTURE FOR IOT-BASED HEALTH CARE APPLICATIONS

From the requirements of a reference architecture for IoT-based healthcare applications, which were previously defined, RAH was designed, as a SRA for IoT-based health care applications. The stakeholders for this reference architecture are systems analysts, software architects, and developers of IoT-based healthcare applications. Thus, RAH, presented in Figure 27, is organized in layers according to its requirements. Layers help to bring the modifiability and portability quality attributes to a software system (BACH- MANN et al., 2011). A layer is an application of the principle of information hiding, whose main theory is that a change to a lower layer can be hidden behind its interface and will not impact the layers above it.

RAH is composed of five layers: sensing, middleware, services, applications, and quality attributes. Interacting with the sensing layer, there are patients with devices to capture their biometrics and environment data. Interacting with the applications layer, there are the users, such as physicians, hospital administrators, nurses, family, patients, pharmaceuticals, and clinical staff, who can be using an IoT-based healthcare application integrated with cloud-based health information systems, e-Health, and mHealth applications. Hospital, ambulance, and pharmacy systems are examples of cloud-based health information systems. Moreover, applications for assisted living, personalized healthcare, monitoring physiological and pathological signals, disease monitoring, self-management, wellness monitoring and prevention, rehabilitation, telepathology, and medication intake are examples of e- Health and mHealth applications.

The Sensing Layer is responsible for monitoring the patient's body and environment, and is composed of the following components:

- **Devices:** it is a hardware component that represents the devices used for monitoring the patient's body and environment. The patient's body monitoring involves sensors to capture heart rate, temperature, oxygen saturation, blood pressure, blood glucose, breathing rate, and ECG. Regarding environment monitoring, the devices are sensors that capture data related to temperature, light, humidity, location, body position, motion, SpO2, pressure, and CO2.
- **Gateway Component:** it is a software component that receives the data from the Devices and makes it available to the Middleware Layer. This component is composed of the Raw Data Receive Service, Raw Data Send Service, Filter Service, and Network Service.

Figure 27: RAH reference architecture.

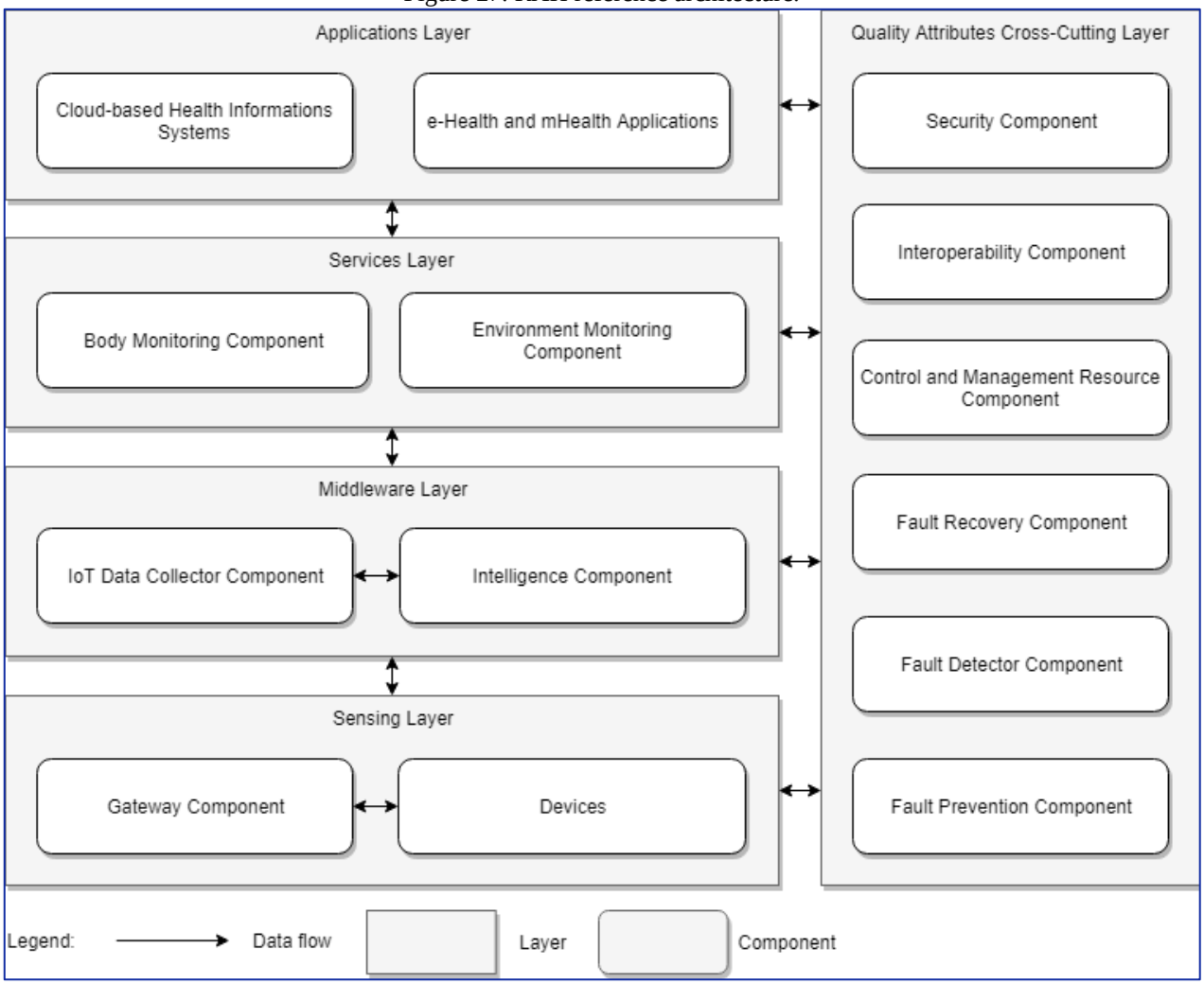


Table 9 presents the Sensing Layer's components and services.

Table 9: Sensing Layer's components and services.

Component Name	Service Name	Description
Devices	EGC Hardware	Hardware to measure the ECG of the patients.
Devices	Blood Pressure Hardware	Hardware to measure the blood pressure of the patients.
Devices	Blood Glucose Hardware	Hardware to measure the blood glucose of the patients.
Devices	Temperature Hardware	Hardware to measure the temperature of the patient's body and environment.
Devices	Location Hardware	Hardware to locate the patients.
Devices	Heart Rate Hardware	Hardware to measure the heart rate of the patients.
Devices	Oxygen Saturation Hardware	Hardware to measure the oxygen saturation of the patients.
Devices	Breathing Rate Hardware	Hardware to measure the breathing rate of the patients.
Devices	Light Hardware	Hardware to measure light conditions of the patients' environment.

Gateway Component	Raw Data Receive Service	Responsible for receiving the raw data of the devices. In this receiving, it uses the authorization service to verify if the device sending the data is authorized. Moreover, it uses the filter service to eliminate noises in the received data.
Gateway Component	Raw Data Send Service	Responsible for sending the raw data to the Middleware Layer. In this sending, it locates the data receive service of the IoTDataCollector using the discovery service.
Gateway Component	Filter Service	Responsible for eliminating possible noises of the signals measured by the devices. It applies the filter according to the characteristic of the signal (e.g., ECG hardware and low pass filter).
Gateway Component	Network Service	Responsible for defining the method of remapping the IP address space of the devices (e.g., NAT, application level gateway).

The Middleware Layer is responsible for receiving the patient's sensors and environment data from the Sensing Layer, processing it, persisting it, and making it available for the Services Layer. This layer is composed of the following components:

1. **IoT Data Collector Component:** the collector is a software component that receives the raw data sent by the Gateway component. It is composed of Data Receive, Data Persistence, Transformation, and IoT Data Send services. Thus, this component is responsible for persisting, processing, and transforming the raw data in a data format that is understandable by the Intelligence component.
2. **Intelligence Component:** it is a software component that receives data from the IoT- Data Collector and uses its inference engines to classify and persist the information in a repository. This information can be specific alerts configured by the clinical staff for the patients, or automatic alerts detected by the use of Artificial Intelligence techniques. This component also sends this information to the Services Layer and is composed of IoT Data Receive, Inference Engine, Information Persist, and Information Send services.

Table 10 presents the Middleware Layer's components and services.

Table 10: Middleware Layer's components and services.

Component Name	Service Name	Description
IoT Data Collector Component	Data Receive Service	Responsible for receiving the data in the IoT Data Collector Component. It verifies the authorization of the origin component using the authorization service and sends the received data to the transformation service.
IoT Data Collector Component	Data Persistence Service	Responsible for persisting the received raw data. This persistence is necessary to lower the possible loss of patients' monitored data. Moreover, this data is consumed by the Intelligence Component.

IoT Data Collector Component	Transformation Data Service	Responsible for transforming the raw data in a format understandable by the Intelligence Component (e.g., JSON, XML, and HL7). It uses the data format service to parse and format the received data. Moreover, this transformation, adds a semantic of health to the data.
IoT Data Collector Component	IoT Data Send Service	Responsible for sending the data to the Intelligence Component. In this sending, it locates the IoT data receive service of the Intelligence component using the discovery service.
Intelligence Component	IoT Data Receive Service	Responsible for receiving the data of the IoT Data Collector Component. In this receiving, it verifies the authorization of the origin component using the authorization service and sends the received data to the inference engine.
Intelligence Component	Inference Engine Service	Responsible for applying logical rules to the knowledge base to deduce intelligent information (e.g., Expert Systems and Deep Learning).
Intelligence Component	Information Persist Service	Responsible for persisting the classified data after the process of the inference engine. This persistence is necessary to lower the possible loss of patients' classified data. Moreover, the classified and nonclassified data is presented and consumed by the Services Layer.
Intelligence Component	Information Send Service	Responsible for sending the information (classified and nonclassified data) to the Services Layer. In this sending, it locates the components of the service layer using the discovery service.

The Services Layer is responsible for establishing a set of available operations related to the consumption of the patient's monitored data (body and environment) by the applications. It centralizes access to this data providing a bridge between the applications in the Application Layer and the Middleware Layer. Thus, this layer is composed of the following components:

1. **Body Monitoring Component:** it is a software component composed of services that provide information about patients' biometrics data. This component is composed of ECG, blood glucose, oxygen saturation, breathing rate, blood pressure, heart rate, and temperature services.
2. **Environment Monitoring Component:** it is a software component composed of services that provide information about patients' environment data. This component is composed of light, humidity, location, body position, motion, SpO2, atmospheric pressure, CO2, and temperature services.

Thus, the data about patients' biometrics and environment are available to the Applications Layer. Table 11 presents the Services Layer's components and services.

Table 11: Services Layer's components and services.

Component Name	Service Name	Description
Body Monitoring Component	ECG Service	Responsible for making patients' ECG data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the ECG data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read ECG data.
Body Monitoring Component	Blood Glucose Service	Responsible for making patients' Blood Glucose data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Blood Glucose data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Blood Glucose data.
Body Monitoring Component	Oxygen Saturation Service	Responsible for making patients Oxy- gen Saturation data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Oxygen Saturation data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Oxygen Saturation data.
Body Monitoring Component	Breathing Rate Service	Responsible for making patients' Breathing Rate data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Breathing Rate data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Breathing Rate data.
Body Monitoring Component	Blood Pressure Service	Responsible for making patients' Blood Pressure data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Blood Pressure data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Blood Pressure data.
Body Monitoring Component	Heart Rate Service	Responsible for making patients' Heart Rate data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Heart Rate data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Heart Rate data.
Body Monitoring Component	Temperature Service	Responsible for making patients' Temperature data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Temperature data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Temperature data.
Environment Monitoring Component	Light Service	Responsible for making patients' Light data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Light data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Light data.

Environment Monitoring Component	Humidity Service	Responsible for making patients' Humidity data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Humidity data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Humidity data.
Environment Monitoring Component	Location Service	Responsible for making patients' Location data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Location data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Location data.
Environment Monitoring Component	Body Position Service	Responsible for making patients' Body Position data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Body Position data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Body Position data.
Environment Monitoring Component	Motion Service	Responsible for making patients' Motion data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Motion data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Motion data.
Environment Monitoring Component	SPO2 Service	Responsible for making patient's SPO2 data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the SPO2 data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read SPO2 data.
Environment Monitoring Component	Atmospheric Pressure Service	Responsible for making patients' Atmospheric Pressure data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Atmospheric Pressure data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Atmospheric Pressure data.
Environment Monitoring Component	CO2 Service	Responsible for making patients' CO2 data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the CO2 data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read CO2 data.
Environment Monitoring Component	Temperature Service	Responsible for making patients' Temperature data available to the Applications Layer. This service verifies if the Intelligence Component is authorized to send the Temperature data and if the applications trying to consume this data are authorized. Moreover, it can access the Middleware Layer to read Temperature data.

The Applications Layer contains the primary usage scenarios of IoT-based healthcare applications. Therefore, these examples of applications are grouped into cloud-based health information systems, e-Health, and mHealth applications. The Cloud-based health information systems are for hospitals, ambulances, and pharmacy systems. The e-Health and mHealth applications

are for assisted living, personalized healthcare, self-management, wellness monitoring and prevention, disease monitoring, medication intake monitoring, telepathology, and rehabilitation. Table 12 presents the Applications Layer's with applications group and name.

Table 12: Applications Layer's components and services.

Application Group	Application Name	Description
Cloud-based Health Information Systems	Hospital Systems	Hospital information systems.
Cloud-based Health Information Systems	Ambulance Systems	Ambulance service information systems.
Cloud-based Health Information Systems	Pharmacy Systems	Pharmacy service information systems.
e-Health and Health Applications	Assisted Living Apps	Ambient assisted living applications.
e-Health and Health Applications	Personalized Health- care Apps	Applications for personalized health- care.
e-Health and mHealth Applications	Self-management, Wellness Monitoring, and Prevention Apps	Applications for wellness monitoring and prevention.
e-Health and mHealth Applications	Disease Monitoring Apps	Applications for specific disease monitoring.
e-Health and Health Applications	Medication Intake Monitoring Apps	Support applications and follow medication intake.
e-Health and Health Applications	Monitoring Physiological and Pathological Signals Apps	Applications for monitoring physiological and pathological signals.
e-Health and Health Applications	Telepathology Apps	Applications for practicing pathology at a distance.
e-Health and Health Applications	Rehabilitation Apps	Application to support rehabilitation of patients.

Finally, the Quality Attributes Cross Cutting Layer is responsible for features that make IoT-based healthcare applications secure, interoperable, available, and efficient. Its components address availability, interoperability, performance, and security. It is important to emphasize that because of the responsibility of this layer, it interacts with the Applications, Services, Middleware, and Sensing layers. Therefore, it is composed of the following components:

- **Security Component:** it is a software component responsible for protecting patients' data and information from unauthorized access while still providing access to people (patients, clinical staff, family, and physicians), systems, and services that are authorized. It is composed of authentication, authorization, encryption, audit, security information, and intrusion detection services.
- **Interoperability Component:** it is a software component responsible for allowing the IoT-based healthcare applications to have the ability to exchange data (syntactic interoperability) with the devices, and also to interpret the data being exchanged (semantic interoperability). It is composed of data format, discovery, and driver services.

- **Control and Management Resource Component:** it is a software component responsible for the performance of IoT-based healthcare applications. This performance regards the time and IoT-based healthcare applications' ability to meet timing requirements. It is composed of event response, event prioritization, and execution limiter services.
- **Fault Recovery Component:** it is a software component related to the availability and fault recoveries of IoT-based healthcare applications. It is composed of retry, redundancy, exception handler, and state resynchronization services.
- **Fault Detector Component:** it is a software component related to availability and fault detections of IoT-based healthcare applications. It is composed of monitor, self-test, and exception detection services.
- **Fault Prevention Component:** it is a software component related to the availability and fault prevention of IoT-based healthcare applications. It is composed of exception prevention and removal services.

Table 13 presents the Quality Attributes Cross-Cutting Layer's components and ser- vices.

Table 13: Quality Attributes Cross-Cutting Layer's components and services.

Component Name	Service Name	Description
Security Component	Authentication Service	Responsible for users' (person, device, application, component, or service) authentication. Authentication means ensuring that a user (person, device, application, component, or service) is actually who or what it purports to be. Therefore, this service validates the submitted credentials, e.g., passwords, one-time passwords, tokens, digital certificates, and biometric identification, querying its database of users.
Security Component	Authorization Service	Responsible for users' (person, device, application, component, or service) authorization. Authorization means ensuring that an authenticated actor has the right to access and modify either data or services. It uses the authentication service to authenticate these users.
Security Component	Encryption Service	Responsible for data encryption. It encrypts each message exchanged by the components. Data should be protected from unauthorized access. Confidentiality is usually achieved by applying some form of encryption to data and communication. Encryption provides extra protection to persistently maintained data beyond that available from authorization (e.g., a virtual private network (VPN) or a Secure Sockets Layer (SSL) can be used for data encryption).
Security Component	Audit Service	Responsible for keeping the record of users' (person, device, application, component, or service) actions in the IoT- based healthcare application and their effects. It extracts user, input, output date, and time data of the action and persists it into a log of information about the operations performed in these applications.

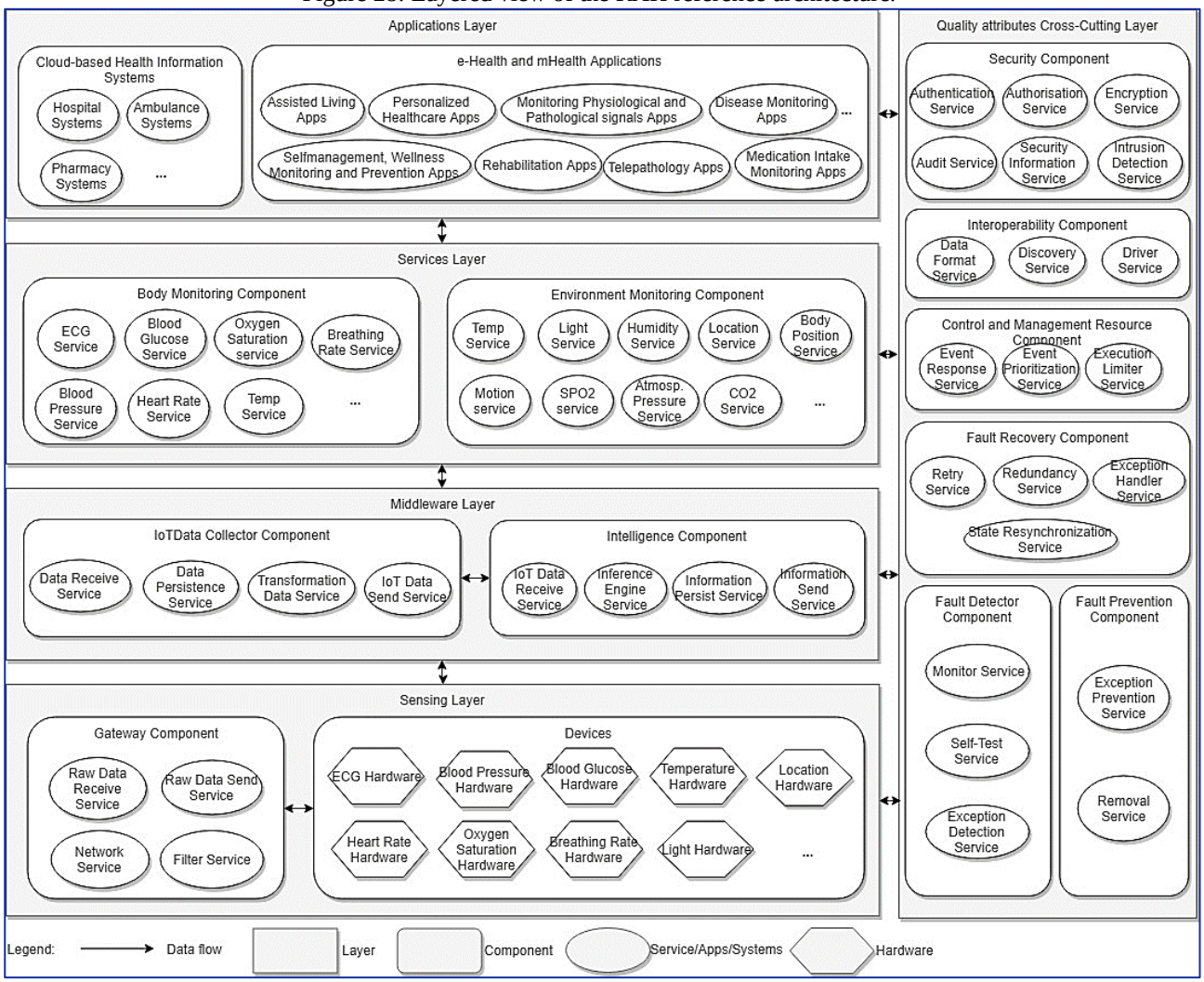
Security Component	Security Information Service	Responsible for notifying the administrators of the IoT-based healthcare application of ongoing attacks on security information. These attacks may require action by these administrators, other personnel, or cooperating systems. It uses the intrusion detection service to detect a security problem and notifies them.
Security Component	Intrusion Detection Service	Responsible for monitoring the network traffic to detect intrusion (e.g., detect intrusion by the comparison of the net- work traffic or service request patterns within a system to a set of signatures or known patterns of malicious behavior stored in a database. The signatures can be based on protocol, TCP flags, payload sizes, applications, source or destination address, or port number).
Interoperability Component	Data Format Service	Responsible for parsing and formatting the data exchanged in the IoT- based healthcare application's components and devices, according to defined standards (e.g., JSON, XML, and HL7). Thus, the driver and transformation data services use this service.
Interoperability Component	Discovery Service	Responsible for maintaining a directory of the components services. These services can be located by type of service, name, component, location, or some other attribute. Moreover, it registers new services using these attributes.
Interoperability Component	Driver Service	Responsible for defining the drivers used to allow communication between the devices and the IoT-based health care applications. For example, suppose that a device needs to send raw data to the Gateway. To process the raw data, the Gateway needs to have a driver for this kind of device. A driver is software that understands the device protocol and converts the received data into a format understandable by the IoT DataCollector. Thus, this service locates the driver for the device, understands its protocol, and converts the data for the IoT DataCollector. It is necessary to emphasize that this service uses the data format service in the process of data conversion.
Control and Management Resource Component	Event Response Service	Responsible for queuing the arrived events in a service/component of an IoT-based healthcare application. When discrete events arrive at a service/component too rapidly to be processed, then the events are queued until they can be processed. Because these events are discrete in IoT-based health care applications, it is typically not desirable to "downsample" them. Thus, this service creates a queue for the events that were not processed when the component was busy. For each new event, the service verifies if the queue is empty and processes the possible existing events in it.
Control and Management Resource Component	Event Prioritization Service	Responsible for prioritization of services/components' events. If not all events are equally important, it can impose a priority scheme that ranks events according to how important it is to service them. If there are not enough resources available in the component to service them when they arise, low-priority events might be queued or ignored. Thus, it uses the event response service to bypass the queue if the component is busy for a high-priority event. Moreover, it also can be used to define events for priority patients depending on their health condition.

Control and Management Resource Com ponent	Execution Limiter Service	Responsible for placing a limit on how much execution time is used by the services/components of an IoT-based healthcare application to respond to an event. If this limit is passed, it defines the component as busy, using the event response service to process the next events.
Fault Recovery Component	Retry Service	Responsible for defining the limit on the number of retries to process and send events of a service/component of an IoT-based healthcare application that is attempted before a permanent failure is declared. This service assumes that the fault (exception) that caused a failure can be transient and retrying the operation may lead to success. Thus, if this limit is reached, it uses the removal service to set the component in an out-of-service state.
Fault Recovery Component	Redundancy Service	Responsible for defining the redundancy configuration of services/components of IoT-based healthcare applications. It can allow the use of configurations such as active redundancy (hot spare), passive redundancy (warm spare), and spare (cold spare). Active redundancy refers to a configuration where all of the nodes (active or redundant spare) in a protection group receive and process identical in- puts in parallel, allowing the redundant spare(s) to maintain a synchronous state with the active node(s). A protection group is a group of processing nodes where one or more nodes are "active," with the remaining nodes in this group serving as redundant spares. Passive redundancy refers to a configuration where only the active members of the protection group process input traffic, requiring that the active members provide the redundant spare(s) with periodic state updates. Cold sparing refers to a configuration where the redundant spares of a protection group remain out of service until a fail over occurs, at which point a power-on-reset procedure is initiated on the redundant spare before its being placed in service.
Fault Recovery Component	Exception Handler Service	Responsible for handle exceptions in components/services of an IoT-based healthcare application. The mechanism employed for exception handling depends largely on the programming environment employed, ranging from simple function return codes (error codes) to the use of exception classes that contain information helpful in fault correlation, such as the name of the exception thrown, the origin of the exception, and the cause of the exception thrown. The software can then use this information to mask the fault, usually by correcting the cause of the exception and retrying the operation.
Fault Recovery Component	State Resynchronization Service	This service is used with the redundancy service and is responsible for synchronizing the state of the nodes in a protection group of the service/component of an IoT-based healthcare application. When used with the active redundancy configuration (hot spare), the state resynchronization occurs naturally, because the active and standby components each receive and process identical inputs in parallel. The states of the active and standby components are periodically compared to ensure synchronization. This comparison may be based on a cyclic redundancy check calculation (checksum) or, message digest calculation (a one-way hash function). When used with the passive redundancy configuration (warm spare), state resynchronization is based only on periodic state information transmitted from the active node to the standby node, via check pointing.

Fault Detector Component	Monitor Service	Responsible for monitoring the health status of the services/components of an IoT-based healthcare application. This service can detect failure or congestion in the network or other hard ware resources. It periodically queries the services/components of these applications for the current health status. This health status is a message mainly composed of information, such as detected exceptions, usage of processors, disk, memory, tasks, load average, uptime, threads, network, and server status (up or down). Moreover, if this service detects an anomaly status, it alerts the administrators of the possible or imminent failure providing information about the component/service health status.
Fault Detector Component	Selftest Service	Responsible for allowing self-test of services/components of IoT-based health care applications. It can run procedures to test itself for correct operation, verifying information, such as detected exceptions, usage of processors, disk, memory, tasks, load average, uptime, threads, network, and server status (up or down). These self-test procedures can be initiated by the component/service itself or invoked from time to time by the monitor service.
Fault Detector Component	Exception Detection Service	Responsible for exception detection in IoT-based healthcare applications. An exception refers to a condition that alters the normal flow of execution. (e.g., division by zero, address faults, illegal instructions, array bounds, time- outs, etc.). Therefore, this service detects and logs the exceptions so that it can be used by the self-test and monitor services.
Fault Prevention Component	Exception Prevention Service	Responsible for exception prevention in components/services of IoT-based healthcare applications. It defines techniques employed to prevent system exceptions from occurring. It can use exception classes to threat values, which allows an application to transparently recover from its exceptions, abstract data types, and the use of wrappers to prevent faults, and defensive programming techniques.
Fault Prevention Component	Removal Service	Responsible for removing components/services of IoT-based healthcare applications. It places an IoT-based healthcare application component/service in an out-of-service state to mitigate potential application failures. Thus, its use is associated with the redundancy service that defines which redundancy configuration and what node in the protection group is going to replace the out-of-service node.

RAH is presented in the layered and decomposition view in Figures 28 and 29. The decomposition view describes the organization of the software into modules and submodules and shows how the system's responsibilities are partitioned across them. The layered view is based on the layered style, which reflects a division of the software into layers that represent a group of modules that offer a cohesive set of services (BACHMANN et al., 2011).

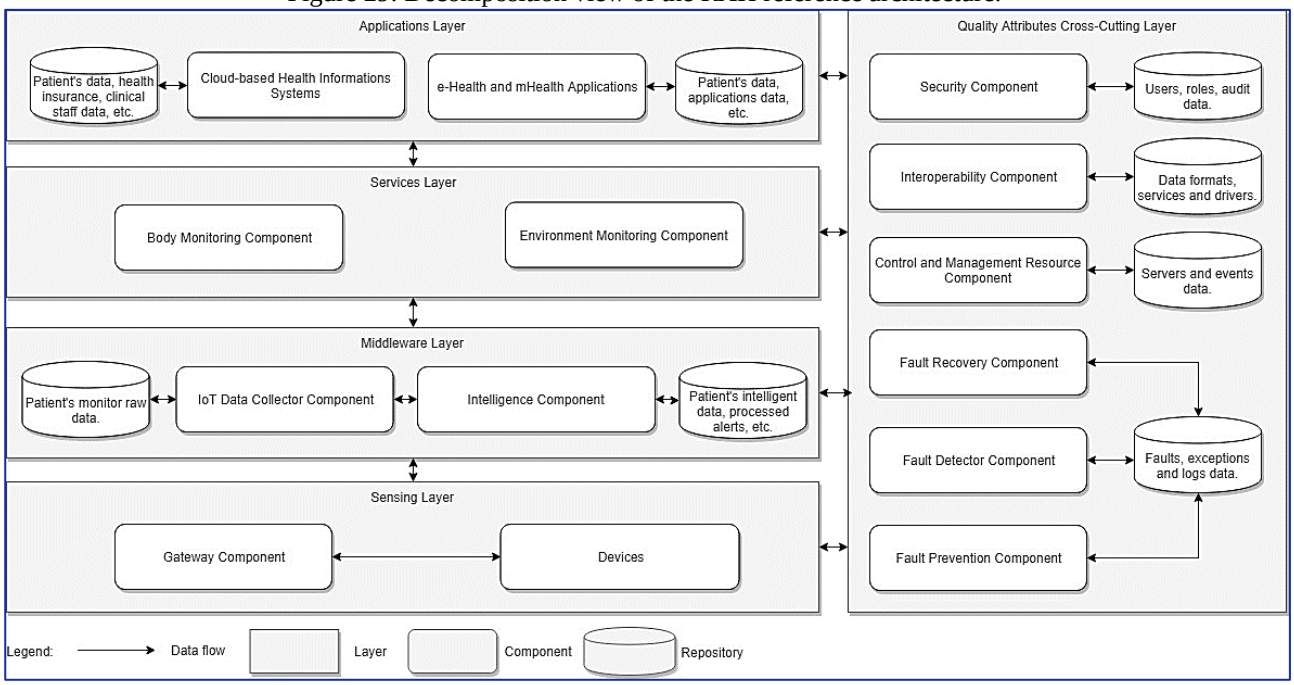
Figure 28: Layered view of the RAH reference architecture.



Continuing with the RAH documentation, Figure 30 presents the view of the use based on the use styles. The uses style results when the depends on the relation is specialized to uses. A module uses another module if its correctness depends on the correctness of the used module. Thus, this style goes one step further to reveal which modules use which other modules, enabling incremental development and the deployment of useful subsets of full systems (BACHMANN et al., 2011).

Therefore, the Gateway uses the Devices, and the IoTDataCollector uses the Gateway. The Intelligence uses the IoTDataCollector and is used by the Body and Environment Monitoring components. The Body and Environment Monitoring components are used by Cloud-based health information systems and e-health and mHealth applications. All theses components use the components of the Quality Attributes Cross-Cutting Layer.

Figure 29: Decomposition view of the RAH reference architecture.

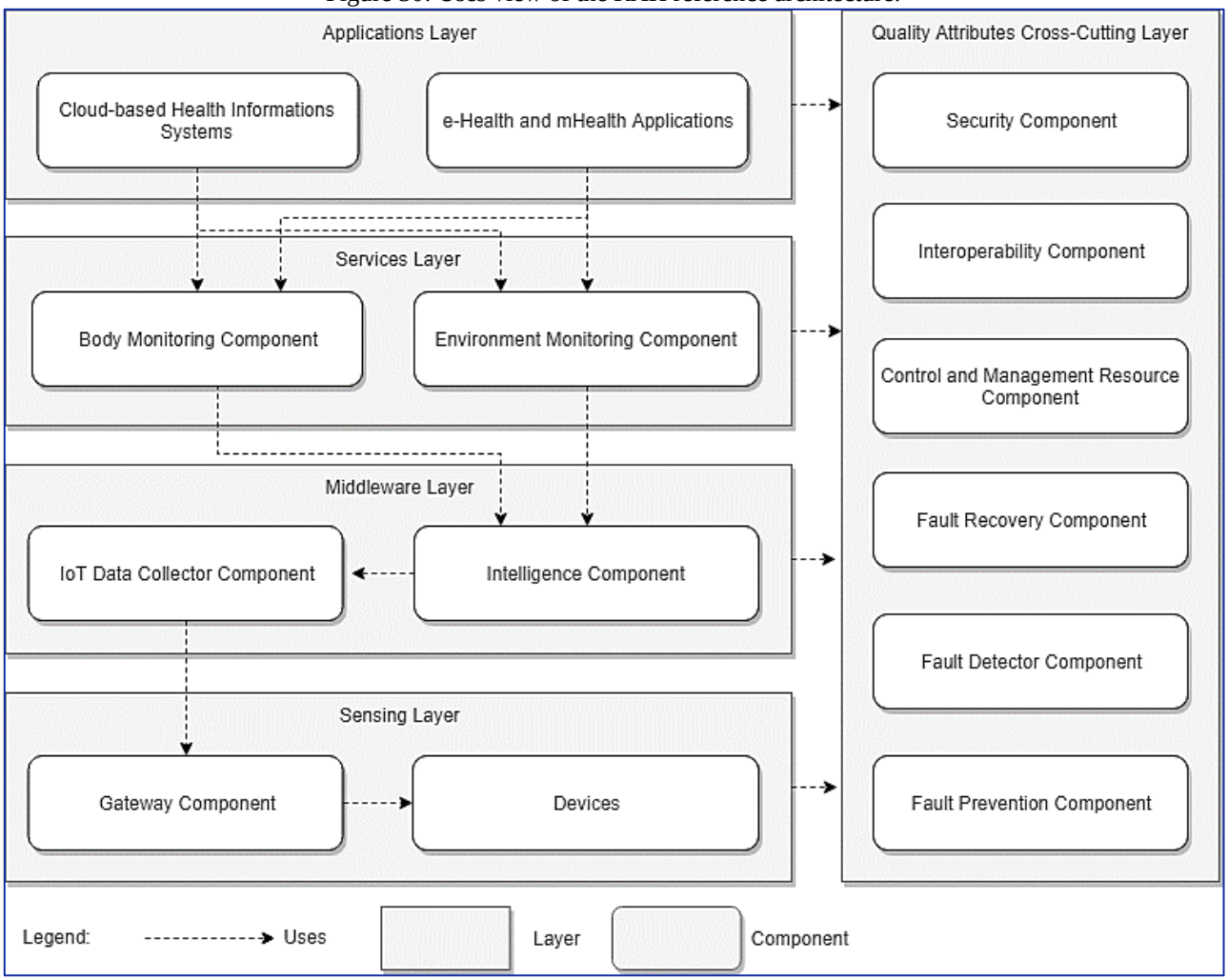


5.3 FINAL REMARKS

This chapter presented RAH, a software reference architecture for IoT-based health care applications, which was designed to serve as a guideline for the design of the architectures of these applications. This SRA systematically organizes the main elements of these applications, their responsibilities, and their interactions, promoting a common understanding of these applications' architecture and addressing the challenges of interoperability, security, performance, and availability related to their development. RAH is defined based on a set of functional and nonfunctional requirements (quality attributes) related to IoT-based healthcare applications. These requirements were extracted from existing publications collected through the study presented in Chapter 4.

The functional requirements of IoT-based healthcare applications involve the patient's body and environment monitoring. The nonfunctional requirements (quality attributes) of these applications are availability, interoperability, performance, and security. The architecture qualities of RAH are completeness, buildability, applicability, and usability. When it comes to functional requirements, the Gateway, the IoTDatacollector, and the Intelligence, Body, and Environment Monitoring components are responsible for providing them. Considering the nonfunctional requirements (quality attributes), - availability, interoperability, performance, and security -, Table 14 specifies the strategies that were used to provide these qualities.

Figure 30: Uses view of the RAH reference architecture.



Finally, RAH was detailed, presenting its modules and elements, and documenting them using the decomposition, layered, and uses views. Finally, in Chapter 6 the architectural evaluation of RAH will be presented.

Table 14: Quality attributes and strategies.

Quality attribute	Strategies
Availability	The definition of fault recovery, detector, and prevention components, are composed of retry, redundancy, exception handler, state resynchronization, monitor, self-test, exception detection, exception prevention, and removal services.
Interoperability	The definition of interoperability component is composed of data format, discovery, and driver services.
Performance	The definition of control and management resource component is composed of event response, prioritization, and execution limiter services.
Security	The definition of security component is composed of authentication, authorization, encryption, audit, security information, and intrusion detection services.

This chapter presents the evaluation of RAH. For this evaluation, a case study was conducted aiming to search evidence to test the hypothesis that a software reference architecture for IoT-based healthcare applications is a suitable approach to address the challenges of security, interoperability, availability, and performance, found in developing this kind of applications. For this, the case study research process presented by Runeson and Host (RUNESON; HOST, 2009) was followed. The results of conducting such a process are presented in this chapter. Section 6.1 describes the case study design and planning, detailing objectives, hypothesis, research questions, and methods to collect data. Section 6.2 presents the collected data used to bring the required evidence to answer each research question. Section 6.3 presents the analysis and synthesis, based on collected data, to resolve the research questions, hypothesis, and objective. A discussion of the results obtained through this case study is detailed in Section 6.4. Threats of validity are discussed in Section 6.5. Finally, Section 6.6 concludes this chapter.

6.1 CASE STUDY DESIGN

To assess RAH, the software architecture of a platform for intelligent remote monitoring of patients, named PAR, was designed as an instance of such reference architecture. PAR is an IOt-based healthcare platform to integrates patients, physicians, and ambulance services (BARROCA; AQUINO, 2017b) to promote better care and fast preventive and reactive urgent actions for patients in a critical situation. It is composed of five modules: Remote Patient and Environment Monitoring, Patient Healthcare Data Management, Patient Health Condition Management, and Emergency and Crisis Management. This platform was developed considering the need to transfer healthcare from the hospital (hospital-centric) to the patient's home (home-centric) and is based on RAH (Reference Architecture for IoT-based Healthcare Applications).

In this section, the case study plan is presented, detailing the research questions (RQs) that were proposed to confirm the hypothesis stated in this book. Moreover, methods to collect data and bring evidence to answer the RQs are also defined in this section.

6.1.1 Objective

The main objective of this case study is to validate the suitability of RAH to support the software architecture design of IoT-based healthcare applications capable of addressing their requirements and overcoming the challenges of interoperability, security, performance, and availability presented in its domain.

6.1.2 Research Questions (RQs)

To achieve the general objective, six RQs were proposed. For each RQ, a hypothesis is intended to be confirmed or refuted through the assessment of units of analysis, which are assessed using collected data. Table 15 presents the RQs and related hypotheses, units of analysis, and data to be collected during the conduction of this case study.

Table 15: Research questions, hypothesis, units of analysis, and data collected.

RQ	Hypothesis	Unit of Analysis	Data Collected
RQ1 - Can a software architecture of an IoT-based healthcare application be designed by using RAH?	RAH allows to design of software architectures for IoT-based healthcare applications.	Instantiation of RAH to design and implement the software architecture of a platform for intelligent remote monitoring of patients.	Documents resulting from conducting the instantiation of RAH (diagrams, models, etc). Analysis of time spent and people involved in conducting the process for creating the architectural design of a platform for intelligent remote monitoring of patients.
RQ2 - Is RAH an alternative to address interoperation issues of IoT-based healthcare applications?	Using RAH, an architecture of an IoT-based healthcare application can address the interoperability of services provided by the components, devices, and applications.	Interoperability scenario.	Information from interoperability scenario template. Architectural views of RAH. Diagrams and models of the platform for intelligent remote monitoring of patients.
RQ3 - Is RAH an alternative to address availability issues of IoT-based healthcare applications?	By using RAH, software architectures of IoT- based healthcare applications can address the availability of components and services.	Availability scenario	Information from availability scenario template. Architectural views of RAH. Diagrams and models of the platform for intelligent remote monitoring of patients
RQ4 - Is it possible to instantiate software architectures of secure IoT-based healthcare applications using RAH?	By using RAH, software architectures of IoT-based healthcare applications can address security requirements.	Security scenario.	Information from security scenario template. Architectural views of RAH. Diagrams and models of the platform for intelligent remote monitoring of patients.
RQ5 - Is RAH an alternative to address performance issues of IoT-based healthcare applications?	By using RAH, software architectures of IoT-based healthcare applications can address the performance of components and services.	Performance scenario.	Information from performance scenario template. Architectural views of RAH. Diagrams and models of the platform for intelligent remote monitoring of patients.
RQ6 - Can a software Architecture of IoT-based healthcare applications, designed using RAH, be implementable?	By using RAH, it is possible to design and implement software architectures for IoT-based healthcare applications.	Instantiation of RAH to design and implement the software architecture of a platform for intelligent remote monitoring of patients.	Code resulting from conducting the implementation process. Analysis of time spent and people involved in conducting the process for implementing the platform for intelligent remote monitoring of patients.

6.1.3 Procedures for Data Collection

To obtain valid information to investigate the established units of analysis, answer the RQs, and confirm or refute the pre-defined hypothesis, the following procedures were performed.

6.1.3.1 Procedure 1 - Documenting the platform software architecture design

To support the investigation of the RQ1, the documentation of the platform architecture design was performed, and the effort (time, people) required to conduct each activity was collected. In summary, the data collected and documented in this process were:

- Requirements document containing functional and nonfunctional requirements of PAR;
- Services, their related health data, and functional requirements that each service is involved in. Services and components conforming to PAR are represented as instances of services defined in RAH.

Moreover, a mapping between PAR requirements and RAH architecture was made. This mapping gives evidence that all requirements are addressed by at least one architectural element of RAH. Elements shown in Tables 16 and 17 were used to document such mapping, and registering the ID of the functional or nonfunctional requirement specified in the requirements document of PAR. Following, the element (e.g., component or service) responsible for each requirement is described.

6.1.3.2 Procedure 2 - Specifying and documenting quality scenarios

Aiming to answer the research questions RQ2, RQ3, RQ4, and RQ5, quality scenarios specifications were proposed. Scenarios help to understand how the system behaves, and which is the system's response when a stimulus is given in determined environmental settings (BASS; CLEMENTS; KAZMA, 2013). In this context, scenarios assist the validation of architectural decisions made to address quality attribute requirements. In the context of this case study, general scenario templates such as those provided by Clements et al. (CLEMENTS et al., 2003) were used to establish scenarios to assess the architecture of PAR regarding interoperability, security, availability, and performance attributes.

Therefore, for analyzing how the software architecture of PAR (as an instance of RAH) addresses these qualities, one scenario was defined for each attribute. In summary, a scenario specification is composed of eight parts as defined by Clements et al. (CLEMENTS et al., 2003):

- **Scenario identities:** Detailing the ID number and scenario objective;
- **Attribute(s):** Specifying the quality attribute(s) with which the scenario is concerned;

- **Environment:** Detailing relevant assumptions about the environment in which the system resides, and the relevant conditions when the scenario is carried out;
- **Stimulus:** Describing a precise statement of the quality attribute stimulus embodied by the scenario;
- **Response:** Exposing a precise statement of the designed quality attribute response. Such response should be measurable in some way to further test the quality attribute requirement;
- **Architectural decision(s):** Describing architectural decisions relevant to the scenario that affect the quality attribute requirement;
- **Reasoning:** Explaining the rationale (qualitatively or quantitatively) behind the architectural decisions, detailing why such decisions support the achievement of quality attribute requirements; and
- **Architectural diagram:** Illustrating architectural information to support the above reasoning.

6.1.3.3 Procedure 3 - Implementing the platform based on software architecture designed

To investigate research question RQ6, PAR was implemented from the concrete software architecture instantiated by RAH. The code and effort (time, people) required to conduct the development activity were collected.

6.1.4 Methods for Data Analysis

Qualitative data analysis is used to generate evidence for confirming or denying the established hypothesis. Hence, to answer each RQ and validate the respective hypothesis, conclusive statements were made, as proposed by Runeson and Host (RUNESON; HOST, 2009).

6.2 COLLECTING EVIDENCE

Nine people participated in this case study during its conduction: (i) The software architect of RAH, in charge of verifying the correct conduction of the instantiation process of RAH, and responsible for collecting and analyzing the evidence to answer the RQs; (ii) the software architect of PAR, responsible for conducting and documenting the instantiation process; (iii) five developers responsible for supporting the requirements elicitation and implementation of PAR; and (iv) two registered nurses assisting the domain analysis activity. The remainder of this section presents the information collected at conducting each procedure described in 6.1.

6.2.1 Procedure 1 - Documenting the platform software architecture design

In this procedure, the scope and architectural design of PAR were established. PAR is an IoT-based healthcare platform for intelligent remote monitoring of patients in a critical situation and was developed considering the necessity to transfer the healthcare from the hospital (hospital-centric) to the patient's home (home-centric). This platform integrates patients, physicians, and ambulance services to promote better care and provide fast preventive and reactive urgent actions. It addresses challenges like interoperability, performance, security, and availability.

The two registered nurses involved in the case study were responsible for defining with the developers and architects the requirements of PAR. In total, 05 functional requirements and 12 nonfunctional requirements were defined for PAR. These functional requirements are summarized in Table 16. The software architecture of PAR identified what are RAH's services and components that address these functional requirements. Stakeholders identified in the context of PAR are the patient, family, physician, nurse, hospital, and ambulance operators.

Table 16: Functional requirements of PAR and RAH's components and services.

Id	Functional requirements	RAH's component and services
FR01	Remote body monitoring of patients: ECG, heart rate, oxygen saturation, temperature, breathing rate.	RAH's body monitoring component: ECG, heart rate, oxygen saturation, temperature, and breathing rate services.
FR02	Remote environment monitoring of patients: temperature and humidity.	RAH's environment monitoring component: temperature and humidity services.
FR03	Patient healthcare data management: records data of patients, physicians, nurses, health insurance, health conditions, history of monitoring, and emergency alerts.	RAH's cloud-based health information systems: Hospital systems.
FR04	Patient's health condition management: definition of critical levels for the values read by the sensors.	RAH's cloud-based health information systems: Hospital systems.
FR05	Emergency and crisis management: patient's health condition and the services that should be alerted in case of emergency.	RAH's cloud-based health information systems: Ambulance systems; RAH's health and health applications.

To attend to these functional requirements, the following twenty-seven use cases were specified and refined through several iterations conducted during group meetings:

- **Patient's Data Management (FR03):** A.1.1.1 - Create patient data, A.1.1.2 - Read patient data, A.1.1.3 - Update patient data, Appendix A.1.1.4 - Delete patient data;
- **Clinical Staff Data Management (FR03):** Appendix A.1.2.1 - Create health professional data, Appendix A.1.2.2 - Read health professional data, Appendix A.1.2.3 - Update health professional Data, Appendix A.1.2.4 - Delete health professional Data;

- **Health Insurance Data Management (FR03):** Appendix A.1.3.1 - Create health insurance data, Appendix A.1.3.2 - Read health insurance data, Appendix A.1.3.3 - Update health insurance data, Appendix A.1.3.4 - Delete health insurance data;
- **Patient And Health Professional Association (FR03):** Appendix A.1.4.1 - Associate patient with health professional, Appendix A.1.4.2 - Disassociate patient with health professional,
- **Patient's Critical Values Configuration (FR04):** Appendix A.1.5.1 - Create patient critical values, Appendix A.1.5.2 - Read patient critical values, Appendix A.1.5.3 - Update patient critical values;
- **Health Data Management (FR03):** Appendix A.1.6.1 - Create patient evolution data, Appendix A.1.6.2 - Read patient evolution data;
- **Emergency Alert Data Management (FR05):** Appendix A.1.7.1 - Receive patient emergency alert, Appendix A.1.7.2 - Read patient emergency alert;
- **Ambulance Data Management (FR03):** Appendix A.1.8.1 - Create ambulance data, Appendix A.1.8.2 - Read ambulance data, Appendix A.1.8.3 - Update ambulance data, Appendix A.1.8.4 - Delete ambulance data;
- **Health Data Monitoring and Reporting (FR01, FR02):** Appendix A.1.9.1 - Real time health monitoring, and Appendix A.1.9.2 - Read health data report.

These use case specifications are detailed in Appendix A. Regarding nonfunctional requirements (quality attributes), the software architecture of PAR identified what are RAH's services and components that address these requirements. Thus, the nonfunctional requirements and RAH's components and services are summarized and presented in Table 17.

Table 17: Nonfunctional requirements of PAR and RAH's components and services.

Id	Nonfunctional requirements	RAH's component and services
NFR01	The platform must be able to interface (exchange data and interpret it) with an OMNI 612 Multiparametric Monitor using HL7 v2.6, and an eHealth Shield using a hashmap (interoperability).	Interoperability component: driver service.
NFR02	Each device of the platform must be able to be located by its type, protocol, and IP (interoperability).	Interoperability component: discovery service.
NFR03	The platform must allow standard communication between participating services (interoperability).	Interoperability component: data format service.
NFR04	The platform must allow access to patient data only for authorized users (security).	Security component: authorization service.
NFR05	The platform must authenticate users and participating services (security).	Security component: authentication service.

NFR06	The platform must offer authorization mechanisms for users and participating services (security).	Security component: authorization service.
NFR07	The platform must respect patients' privacy and protect its data with confidentiality and integrity (security).	Security component: encryption service, authorization, and authentication services.
NFR08	The platform must detect failures in the participating services (availability).	Fault detector component: Exception detection service.
NFR09	The platform must provide error handling (availability).	Fault recovery component: Exception handler service.
NFR10	The platform must monitor the participating services and devices (availability).	Fault detector component: monitor service.
NFR11	The platform must be aware of its situation, and prevent and correct internal faults and failures (availability).	Fault detector component: monitor service.
NFR12	The platform must be able to monitor 19 patients and handle 133 transactions per second (performance).	Control and management resource component: event response, prioritization, and execution limiter services.

The number of monitored patients (19) proposed in NFR12 is based on the current capacity of the Intensive Care Unit (ICU) of the Onofre Lopes Hospital. The nurses participating in this case study work at this hospital and suggested that this platform should be able to handle the current capacity of this ICU. The number of transactions (133) proposed in NFR12 is based on the 19 patients and the necessity of PAR to monitor ECG, heart rate, oxygen saturation, temperature, breathing rate, temperature, and humidity (NFR01 and NFR02). This results in 19 patients with 7 monitored data per second (one for each sensor).

In the discussion, the software architect and the developers chose to use JSON as the standard data format to exchange clinical data between the platform components and services. JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write. Moreover, it is also easy for machines to parse and generate JSON data (JSON, 2016). Based on the requirements documents, and RAH reference architecture, defined in section 5.2, PAR's software architect designed PAR architecture instantiating the identified components and services of RAH. Figure 31 presents the RAH layered view highlighting the instantiated elements for PAR. Descriptions of RAH components and services responsibilities were detailed in Section 5.2.

Figure 31: Layered view of the RAH reference architecture with highlighted elements for PAR.

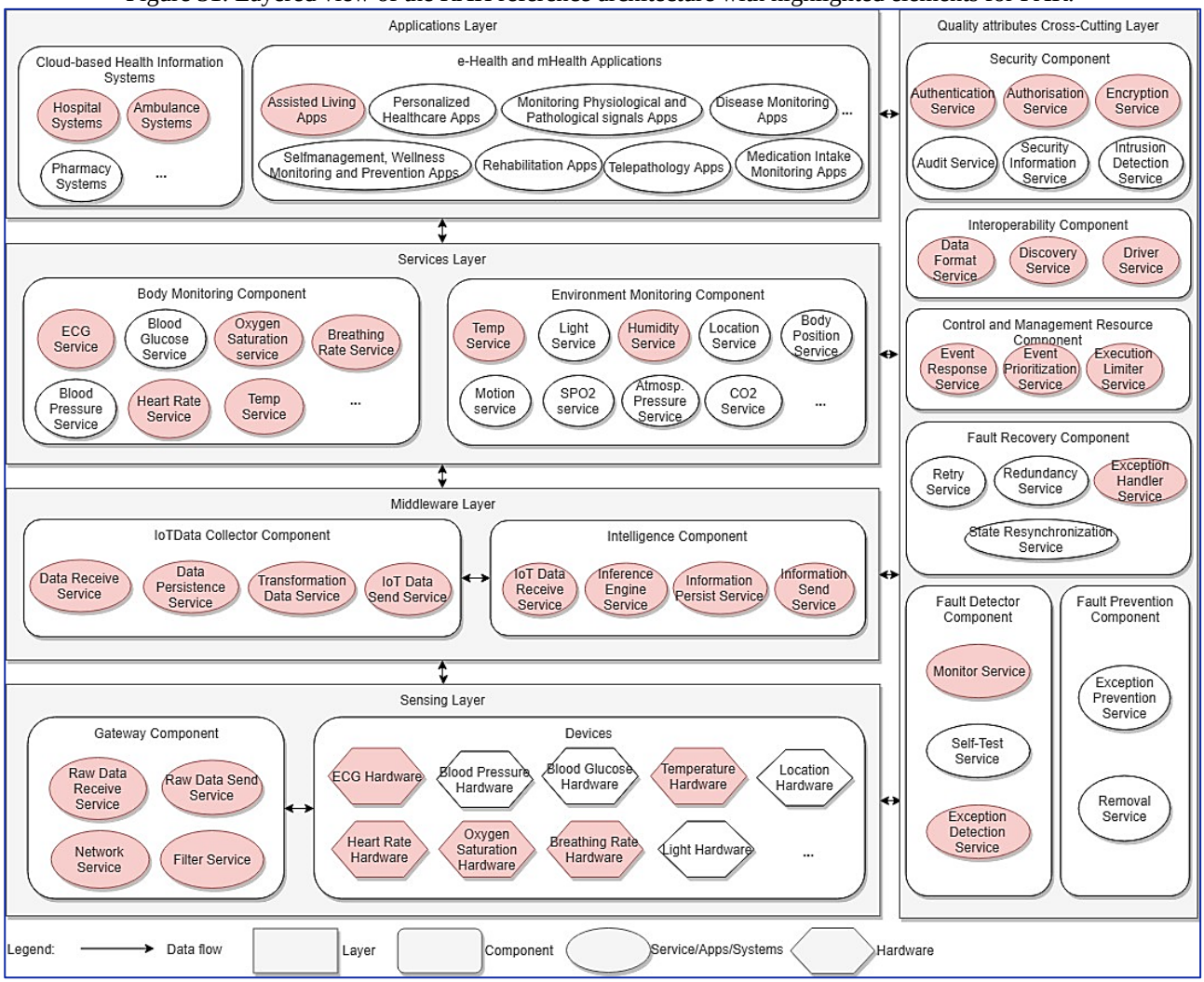


Figure 32 presents a PAR layered view as an instance of RAH. The layered style (BACHMANN et al., 2011) was used to design the layered view. This view was used to represent PAR according to layers' division (stereotype layer) and its components (stereotype segment), according to RAH's structure, presented in Figure 28. Another motivation for this view is to present PAR abstractly, without detailing the components and services, permitting the developer's team to have a simple view of the layers and their responsibilities in PAR.

Moreover, this layered view presents the interactions between the layers of PAR, representing it through the stereotype allowed to be used. The top layer is only allowed to access the next lower layer, except the cross-cutting layer which can be accessed by any other layer presented in this view. Thus, the Application Layer accesses only the Service Layer, which accesses only the Middleware Layer. This last layer accesses only the Sensing Layer, and the Quality Attributes Cross-cutting Layer can be accessed by the other layers. With this layer division, the PAR software architect intends to achieve portability and maintainability.

Figure 32: PAR layered view as an instance of RAH.

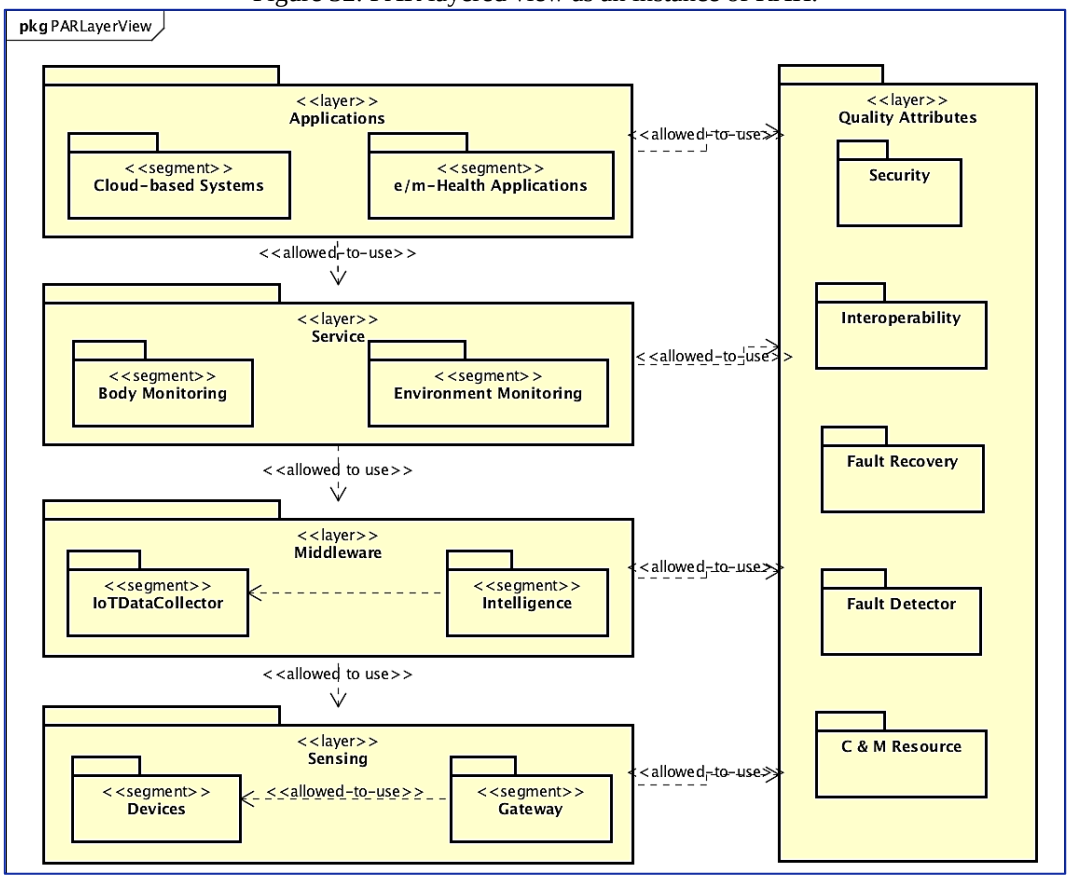
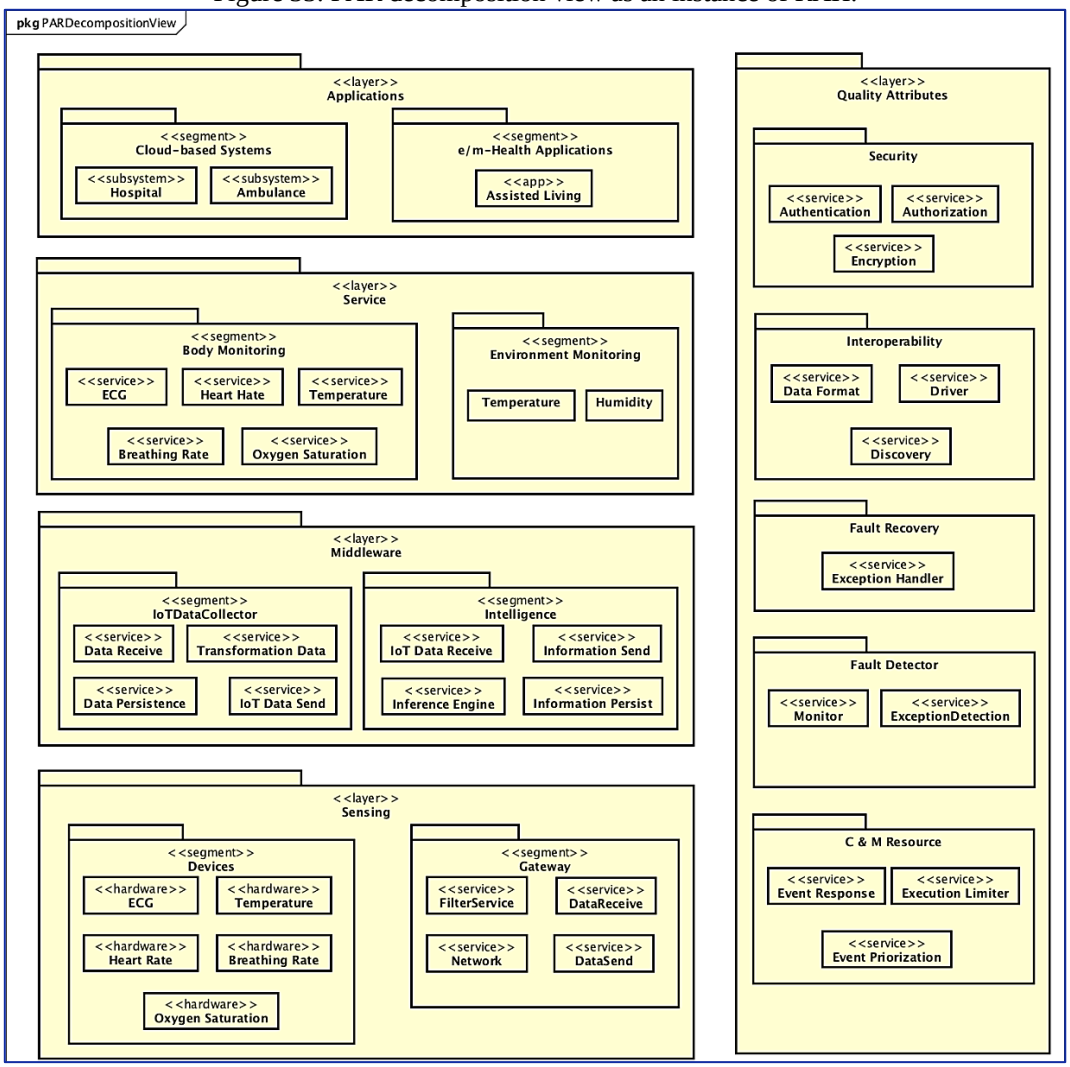

pkg PARLayerView
<<layer>>
Applications
<>
Cloud-based Systems
<>
e/m-Health Applications
<<allowed-to-use>>
<<layer>>
Quality Attributes
Security
<<allowed-to-use>>
<<layer>>
Service
<>
Body Monitoring
<>
Environment Monitoring
<<allowed-to-use>>
Interoperability
<<allowed to use>>
Fault Recovery
<<layer>>
Middleware
<>
IoTDataCollector
<>
Intelligence
<<allowed-to-use>>
Fault Detector
<<allowed to use>>
<<layer>>
Sensing
<>
Devices
<<allowed-to-use>>
<>
Gateway
<<allowed-to-use>>
C & M Resource

Figure 33 presents the PAR decomposition view. This view presents PAR in a fragmented way, beyond the layers and components presented in PAR Layered View, it details the PAR's services. This view was created by the PAR software architect following the requirements, and selecting which services of RAH would be instantiated for PAR. The importance of the PAR decomposition view is the simplicity of the presentation of PAR fragmented in services, without showing its relationships, which are the focus of the following architecture views, designed from this decomposition. The services of RAH selected for the components are necessary to achieve the functional and non-functional requirements of PAR.

Figure 33: PAR decomposition view as an instance of RAH.


pkg PARDecompositionView
<<layer>>
Applications
<>
Cloud-based Systems
<<subsystem>>
Hospital
<<subsystem>>
Ambulance
<>
e/m-Health Applications
<<app>>
Assisted Living
<<layer>>
Service
<>
Body Monitoring
<<service>>
ECG
<<service>>
Heart Hate
<<service>>
Temperature
<<service>>
Breathing Rate
<<service>>
Oxygen Saturation
<>
Environment Monitoring
Temperature
Humidity
<<layer>>
Middleware
<>
IoTDataCollector
<<service>>
Data Receive
<<service>>
Transformation Data
<<service>>
Data Persistence
<<service>>
IoT Data Send
<>
Intelligence
<<service>>
IoT Data Receive
<<service>>
Information Send
<<service>>
Inference Engine
<<service>>
Information Persist
<<layer>>
Sensing
<>
Devices
<<hardware>>
ECG
<<hardware>>
Temperature
<<hardware>>
Heart Rate
<<hardware>>
Breathing Rate
<<hardware>>
Oxygen Saturation
<>
Gateway
<<service>>
FilterService
<<service>>
DataReceive
<<service>>
Network
<<service>>
DataSend
<<layer>>
Quality Attributes
Security
<<service>>
Authentication
<<service>>
Authorization
<<service>>
Encryption
Interoperability
<<service>>
Data Format
<<service>>
Driver
<<service>>
Discovery
Fault Recovery
<<service>>
Exception Handler
Fault Detector
<<service>>
Monitor
<<service>>
ExceptionDetection
C & M Resource
<<service>>
Event Response
<<service>>
Execution Limiter
<<service>>
Event Priorization

For PAR architecture, the needed to represent through the architectural view of the "use" of the services. Figure 34 presents the PAR Uses View. In this view, the services that are presented in the decomposition view are now related to the dependencies between them. This relation is understood as depending on if a stereotype arrow "uses" exits from one service to another. It is important to emphasize that this view does not present the data flow between the services, but only the dependencies between them, which obey the definitions of allowed to use presented PAR layer view (Figure 32).

Dependencies start with the Gateway Component, which depends on the Devices, which are collecting patient biometrics and environment data, and Quality Attributes services. Internally to the Gateway, there is a dependency between services: the Network service is required for the Filter and DataReceive services, which also depends on the Filter. Finally, the DataSend service of Gateway depends only on the DataReceive of the IoTDataCollector. Moreover, the service of the Gateway depends on the Authorization Service of the Security Component, and the DataReceive service of the

IoTDataCollector Component depends on all services of the Interoperability Component (Driver, DataFor- -mat, and Discovery).

The IoTDataCollector Component, being the "entrance" of the Middleware Layer, depends on the Gateway Component since the IoTDataCollector's DataReceive service depends on the Gateway's DataSend service. Then, there is the DataReceive as a dependency of the DataPersist and the TransformationData, which is the dependency of the IoTDataSend. Also, DataReceive depends on the Driver and Authorization services, and the TransformationData depends on the DataFormat.

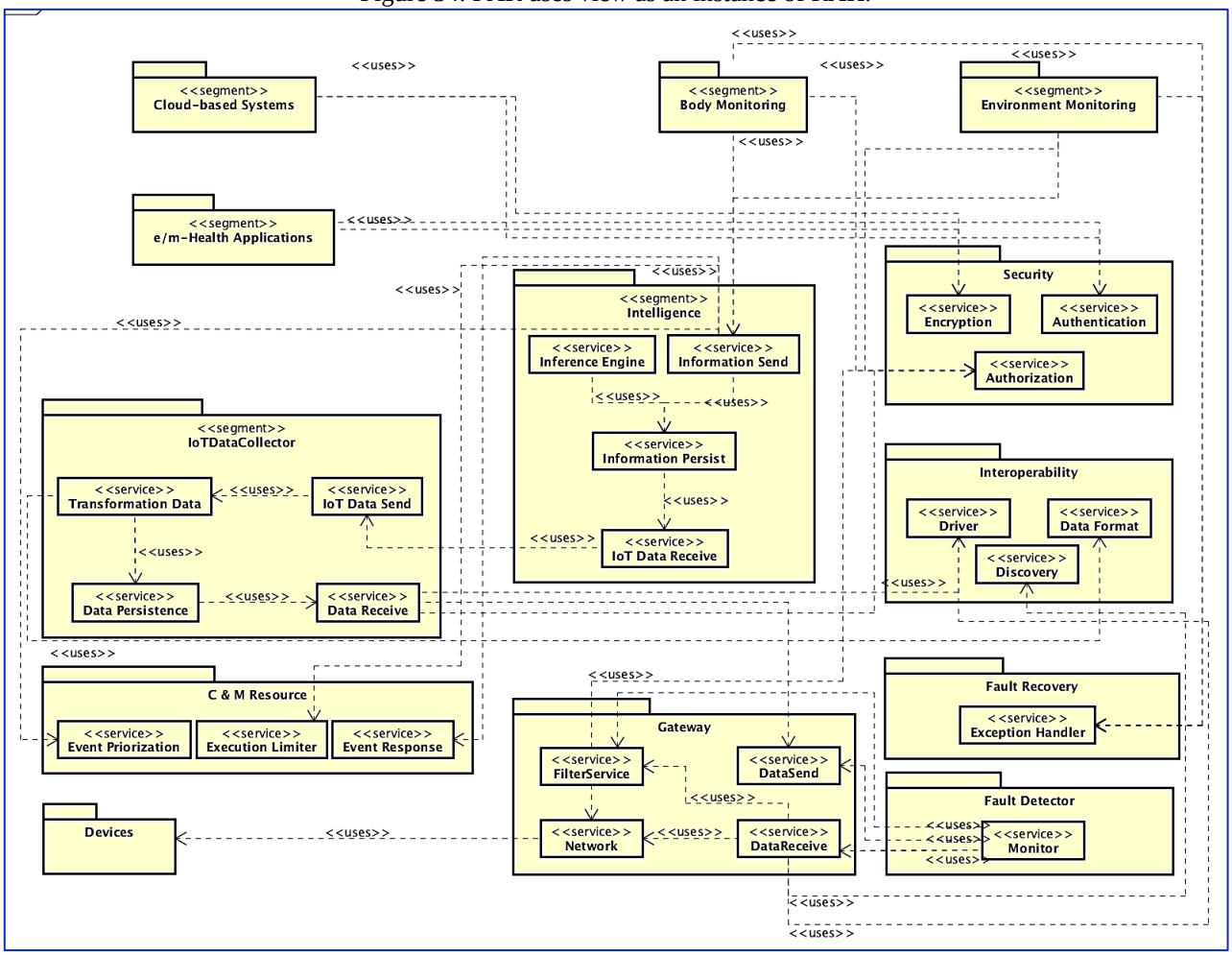
In the Intelligence Component, the dependency starts with the IoTDataReceive service, which depends on the IoTDataSend of the IoTDataCollector Component. Information Persist depends on IoTDataReceive and is dependent on IntelligentEngine and InformationSend. In its turn, InformationSend relies on the Control and Management of Resource services (EventPriorization, ExecutionLimiter, EventResponse).

The services of the Quality Attributes Layer, being cross-cutting, may have a dependency association for the services of the other layers. The Monitor service of the Fault Detector Component will check the availability of the Gateway services, so it depends on them, except the Network Service, which is independent of the other services of this component. Finally, the components of the Service Layer (environment and body monitoring) depend on the InformationSend service of the Middleware Layer and are dependencies for the components of the Application Layer. The Service Layer services also depend on the ExceptionHandler and Authorization services. In the Application Layer, the services depend on Encryption and Authentication services.

PAR software architect designed the PAR data model view, presented in Figure 35. This view was chosen to represent the PAR domain entities, as well as the relationship between them, and it was built according to the use cases specification presented in Appendix A. The Patient entity represents a person who is under the care of a health professional, and it is the central entity of this model since practically all other entities have a direct or indirect relationship with it.

A health professional entity is the representation of a person who acts professionally in the health area, working in a hospital, clinic, or ambulance, and may have a specific specialization, which is represented through the generalization relationship of Physician and Nurse. A HealthProfessional relates to a Patient through the HealthProfessionalCaring entity, which corresponds to a professional performing health procedures on a patient.

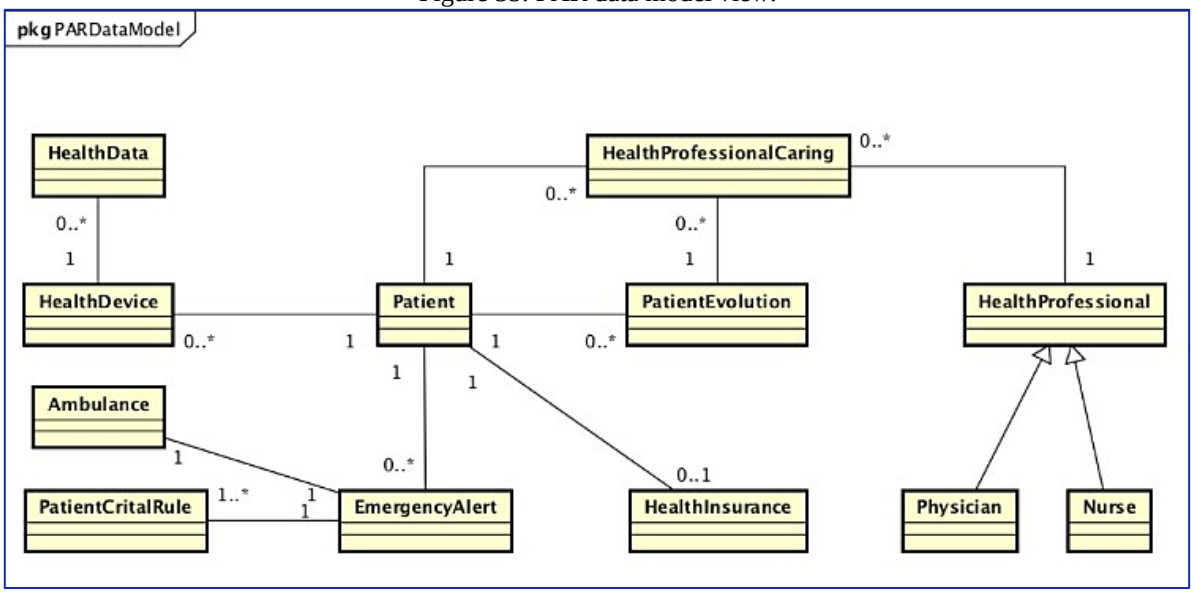
Figure 34: PAR uses view as an instance of RAH.


<>
Cloud-based Systems
<>
Body Monitoring
<>
Environment Monitoring
<>
e/m-Health Applications
<>
Intelligence
<<service>>
Inference Engine
<<service>>
Information Send
<<service>>
Information Persist
<<service>>
IoT Data Receive
Security
<<service>>
Encryption
<<service>>
Authentication
<<service>>
Authorization
<>
IoTDataCollector
<<service>>
Transformation Data
<<service>>
IoT Data Send
<<service>>
Data Persistence
<<service>>
Data Receive
Interoperability
<<service>>
Driver
<<service>>
Data Format
<<service>>
Discovery
C & M Resource
<<service>>
Event Priorization
<<service>>
Execution Limiter
<<service>>
Event Response
Gateway
<<service>>
FilterService
<<service>>
DataSend
<<service>>
Network
<<service>>
DataReceive
Fault Recovery
<<service>>
Exception Handler
Fault Detector
<<service>>
Monitor
Devices
<<uses>>

The PatientEvolution entity refers to these procedures that are being recorded and the evolution of the clinical condition of a patient. HealthData is the entity that represents data (biometrics) from the patients. This data is gathered through the devices represented by the HealthDevice entity. Patient data can generate alerts for the EmergencyAlert entity, which may vary according to patient alert rules defined in the PatientCriticalRule entity. EmergencyAlert may allow the use of the Ambulance service and other services, depending on the patient's health insurance entity.

The Component and Connector view, presented in Figure 36, exposes the PAR components connected according to the data flow between them, informing the types of data that flow in this platform, and the communication interfaces between the components.

Figure 35: PAR data model view.



This view was constructed from the Uses and Decomposition views, presented in Figures 33 and 34, following the rules established in it regarding the communication between components. The rationale for this view is to allow the PAR implementation team to have an idea of how the data should enter and exit each component, and promote more clarity in the technical aspects that they should consider when developing each component.

Moreover, in this view, it is possible to note the following PAR quality attributes: interoperability and availability. Interoperability can be achieved in the communication between the devices and the Gateway component, which allows connection with different device types and different data flows. Availability can be achieved through the presence of the Fault Detector component, which monitors the PAR components, to identify any anomalies in their behavior. Regarding the data flow presented in this view, it starts with the devices sending the raw data (HL7 V2.6 and HashMap) to the Gateway. The Gateway packets the data, defines the packet headers, and sends them to the IoTDataCollector (IDC). The IDC will receive the data packets, persist, and treat them so that the output to the Intelligent Component is like IoTData.

Therefore, the Intelligent Component will apply its rules of inference about the IoT- Data, so that this data is semantically understood and presents information about the health status of a patient. The service layer components (Body and Environment Monitoring) act as interfaces that abstract the requests for information about patients' health and the environment in which they are accommodated. Finally, this information reaches the applications and is presented to the end users of PAR. Next, the component and connector views are presented from the internal perspective of the PAR components, which are shown in Figure 36. It is important to note that the services in the component-specific views have stereotypes that represent the RAH services that are being implemented.

Figure 36: PAR component and connector view.

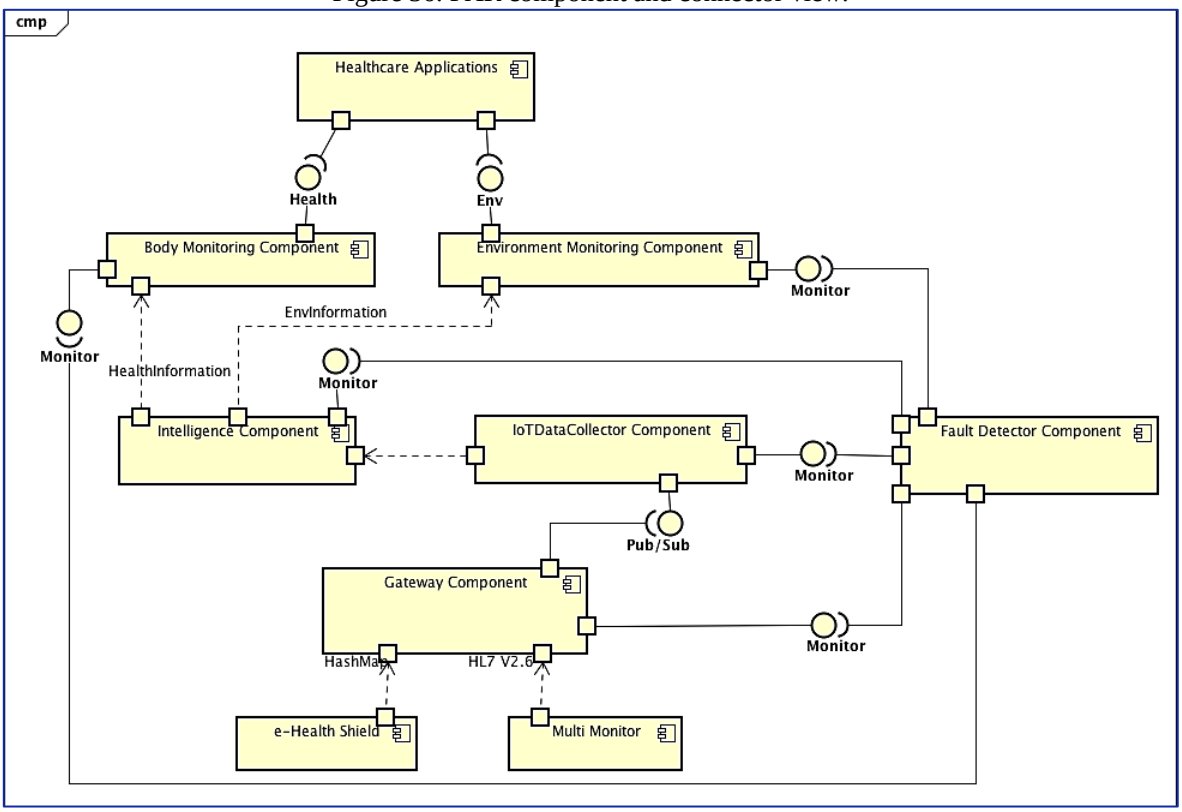


Figure 37 presents the component and connector view of the PAR Gateway component. This view details the data flow between services in the Gateway component. With this view, it is possible to note the achievement of security quality attributes with the existence of the AuthService, which deals with the authentication and authorization of the devices connected to the Gateway. Another important aspect is the presence of the Monitor interface, which allows the monitoring of Gateway services by the Fault Detector Component.

The data stream starts with the data (HL7 V2.6 and HashMap) of the devices entering the Gateway. First, it passes through the NATService without experiencing change and going to the RawDataService. RawDataService, in turn, uses the FilterService, DeviceDiscoveryService, and AuthService services to perform its filter operations on the signals, device discovery, authentication, and authorization, respectively. Then the data goes to the driver service, which classifies the data according to the device. Finally, the data goes to the RawDataSendService, which is responsible for sending the data to the IoTDataCollector.

Figure 37: Gateway component and connector view.

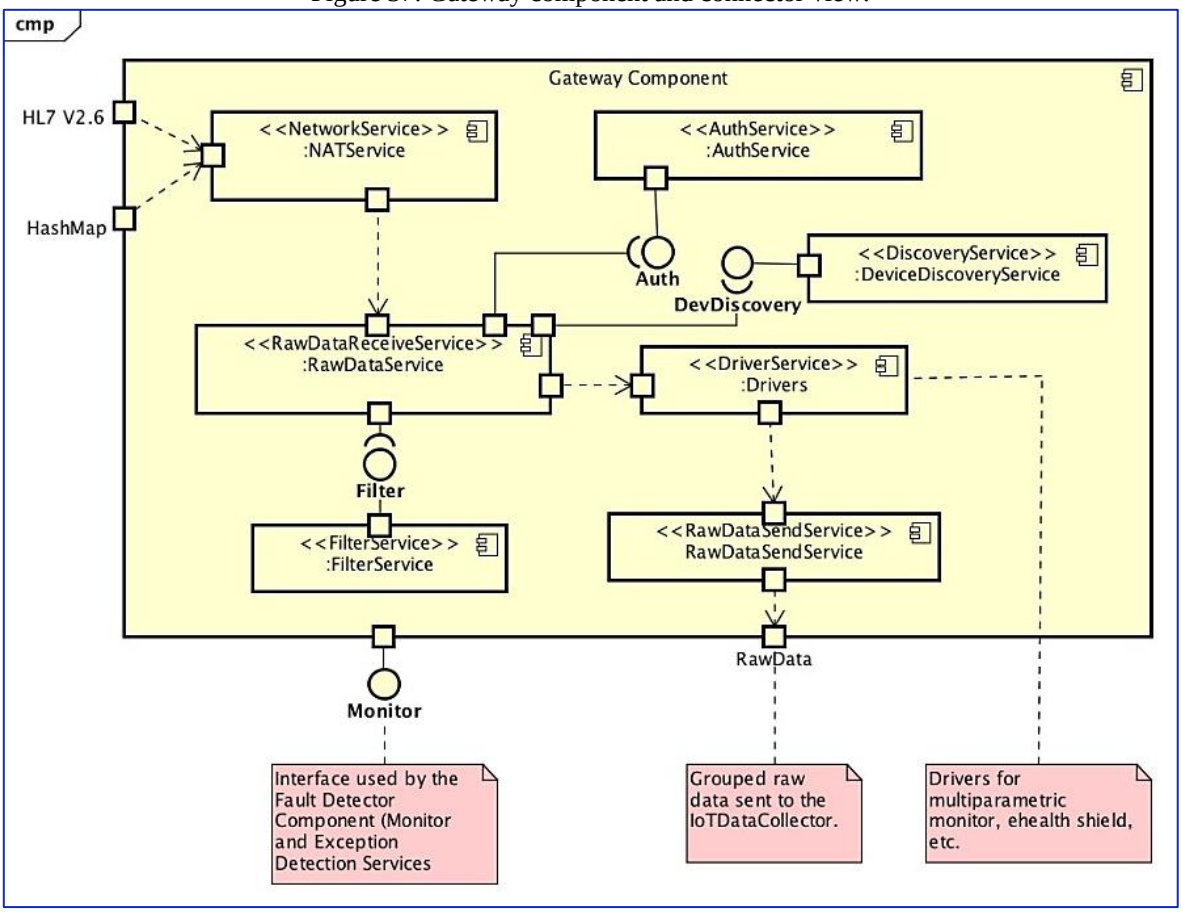
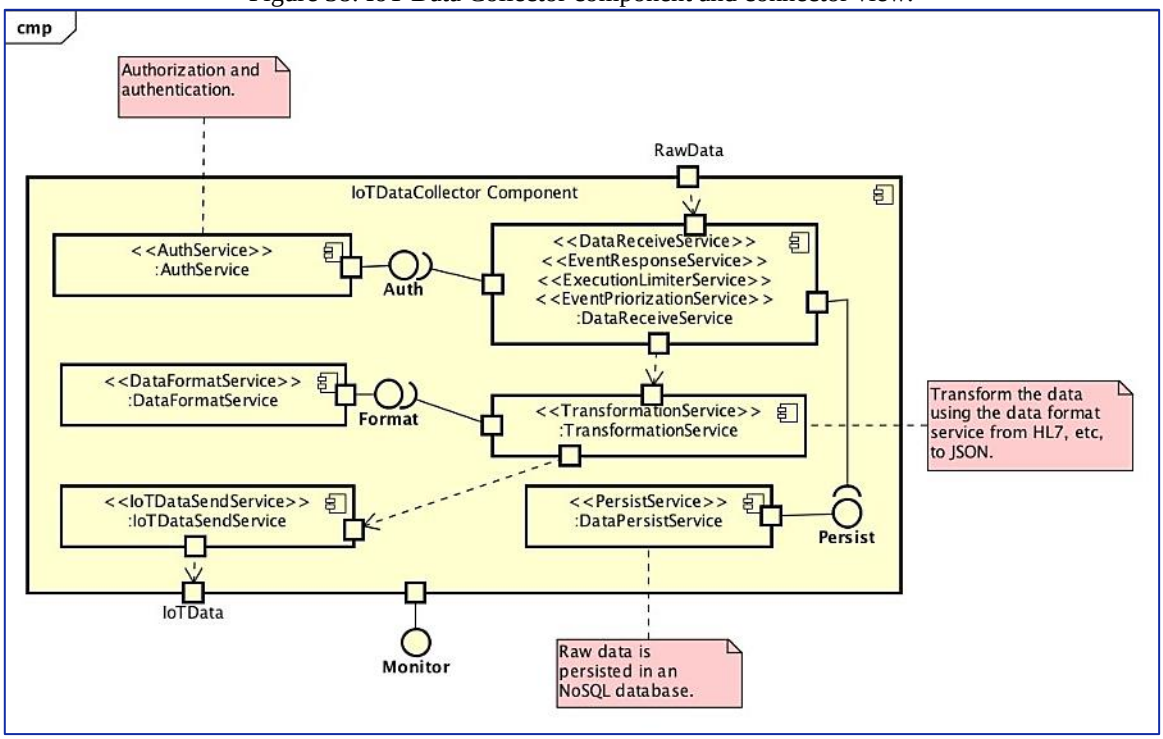


Figure 38 presents the detailed component and connector view of IoTDataCollector (IDC). In this view, it is possible to note that, in addition to the information on the input and output data types of this component, it has the data flow between the services that compose it. Starting the stream, the DataReceiveService uses the AuthService and DataPersistService services to perform authentication, and authorization of data sources and persists the raw data in the Gateway component.

Also, the DataPersistService has the service responsibilities (represented by the stereotypes) of EventResponseService, ExecutionLimiterService, and EventPriorizatonService. Following the flow, TransformationService is responsible for transforming (or converting, parsing) RawData into IoTData, which has the semantics of the device context. To perform this transformation, TransformationService uses the DataFormatService, which functions as a dictionary that assists in the translation of RawData. Finally, the stream ends with the IoTDataSendService, which is responsible for sending the IoTData to the Intelligence component.

Figure 38: IoT Data Collector component and connector view.



The component and connector view of the Intelligence component (IC) is presented in Figure 39. It receives the data (IoTData) sent by the IDC and classifies them according to the type of output information, which may be health or environmental information. The data flow between the Intelligence component services starts with the IoTDataReceiveService, which similar to the Gateway's DataReceiveService, has the responsibilities of the EventResponseService, ExecutionLimiterService, and EventPriorizatonService defined in the stereotypes. It also uses an AuthService to authenticate and authorize the component of the data source. Then, the InferenceEngineService (IE) is responsible for performing possible classifications of the data and can generate health alerts related to patients being monitored by the platform. For this, the IE uses the DataPersistService to read and write the sort data. As a final step, the flow goes on with the InformationSendService receiving the information from IE and sending it to health or environment monitoring services.

The component and connector view of the patient's body and environment monitoring components are presented in Figure 40. These components have as input the information that arrives from the Intelligent component. In the case of the Body Monitoring component (BMC), the health information is received and, in the case of the Environment Monitoring Component (EMC), environmental information is received. For these components, the input information is the same as the output, but at the output, this information is made available through interfaces that represent the component's use call.

The services of these components do not communicate with each other, including those internal to the same components. This is because each service handles the information regarding your domain, to avoid the coupling and ensure the maintainability of them.

Figure 39: Intelligence component and connector view.

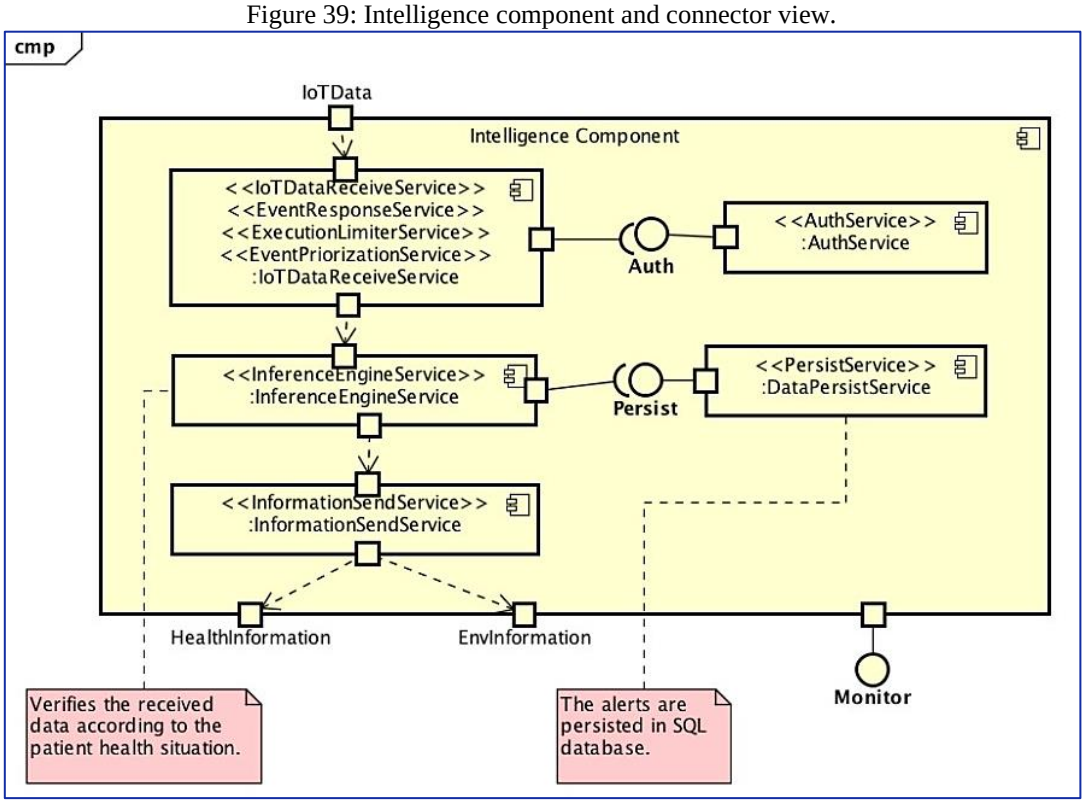
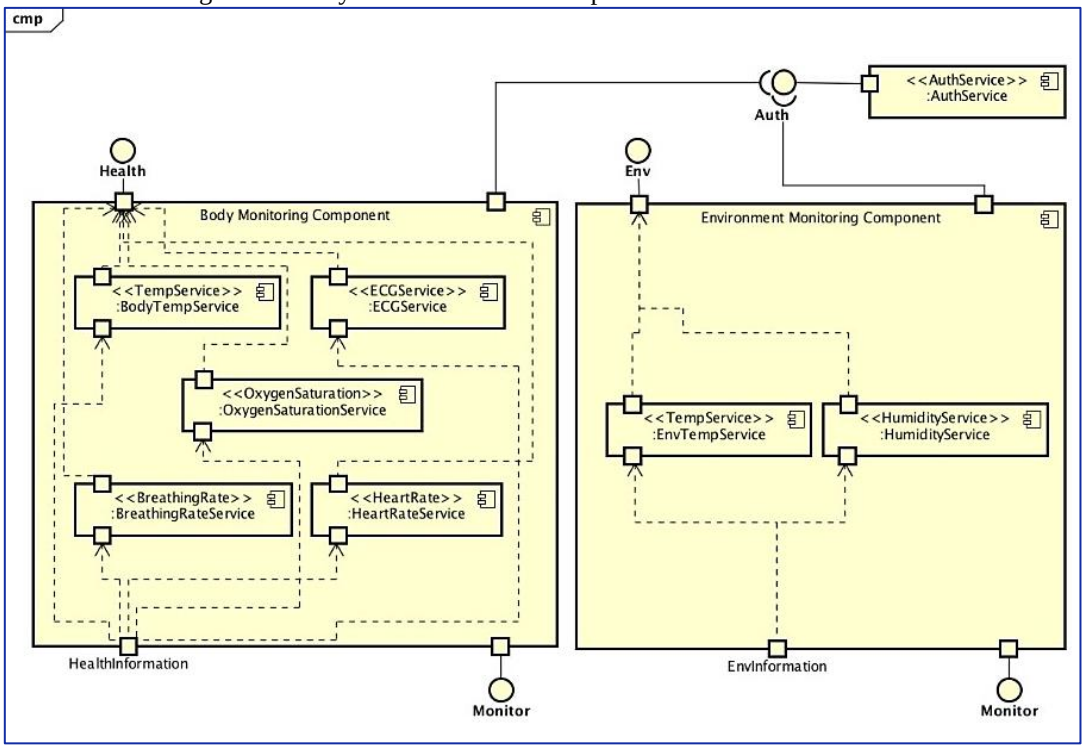


Figure 40: Body and environment components and connectors view.



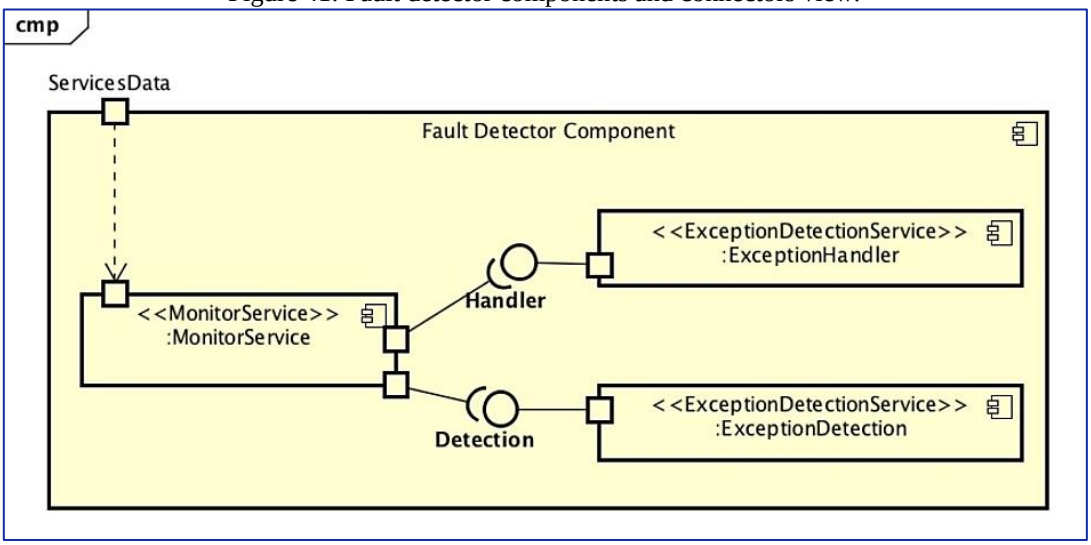
In this way, the information flow becomes very simple, since the information enters the component, through the services of its respective domain and goes to the applications that request them, through the interfaces.

These components also use AuthService, but in this case, it is used to authenticate and authorize the applications that request patient information, and since the same applications can require information from the two components, they use the same AuthService. Finally, the Monitor interface is used to allow the monitoring of these components by the FaultDetector component.

Figure 41 presents the component and connector view of the Fault Detector Component (FDC), which belongs to the cross-cutting layer of PAR, as explained in the PAR decomposition view (Figure 33). In this way, the FDC can communicate with the components of all other layers and, for this reason, the input data is represented by the ServicesData, which contains data about the PAR components, such as Gateway, IoT- DataCollector, Intelligence, Body and Environment Monitoring.

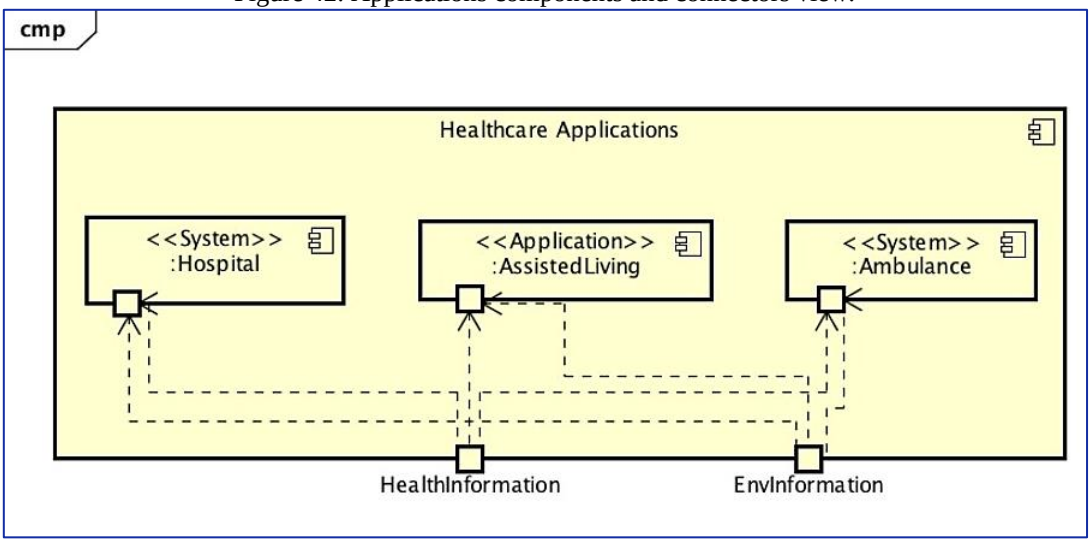
This monitoring by the FDC is possible through the use of the Monitor interface, which is presented in Figures 37, 38, 39, and 40. In this view, it is noted that the quality attributes of performance and availability are addressed, since FDC is constantly monitoring the PAR components to identify anomalies in their behavior, such as the execution time above the desired or unavailability.

Figure 41: Fault detector components and connectors view.



The FDC data stream starts with the SeviceData arriving at the component, according to the constant requests that are made to the monitored components in search of services status. The MonitorService then receives this data and uses the ExceptionDetection and ExceptionHandler services to perform the appropriate treatment for unusual situations. FDC does not have a data output since no function needs to consume the results of its services.

Figure 42: Applications components and connectors view.



The last component and connector view, presented in Figure 42, is about PAR applications. They are the final destiny of the data, so, this view does not display output data, only the input information, which comes from the health and environment monitoring components. The application services consume this information.

The PAR architect designed the repository views, based on the RAH decomposition view presented in Figure 29. The instanced repositories are presented in Figure 43. These repository views represent interactions between PAR services and the databases specific to the components to which they belong. Thus, the team involved in the project has an artifact that presents the platform databases, which component they are located in, the services they access, and how this access is made (read and write).

The services are represented by the blocks, the databases by the disks, and the types of database access are represented by the arrows that connect the services to the databases. The relations can be of three types: just write, the arrow leaving the service destined to the database; read-only, the arrow leaving the database to the service, and read and write, a bi-directional arrow.

The repository view of the Gateway component is presented in Figure 44. The Gateway services use an in-memory DB (H2DB) because of the hardware limitations in which they are running. The DeviceDiscoveryService stores device-specific information to identify them at the moment they connect to the Gateway to send data. The AuthService accesses this database to authenticate and authorize devices attempting to connect to the Gateway.

In the IoTDataCollector, presented in Figure 45, services access two databases: a nonrelational database (NoSQL) for storing data from devices, and a relational database for authentication and authorization from the devices that attempt to connect to it for data transmission. The use of the

nonrelational database is necessary because of the variety of data that can be received from the devices and, hence, it is possible to maintain its state before being transformed into IoTData.

Figure 43: Decomposition view of the RAH reference architecture with highlighted elements for PAR.

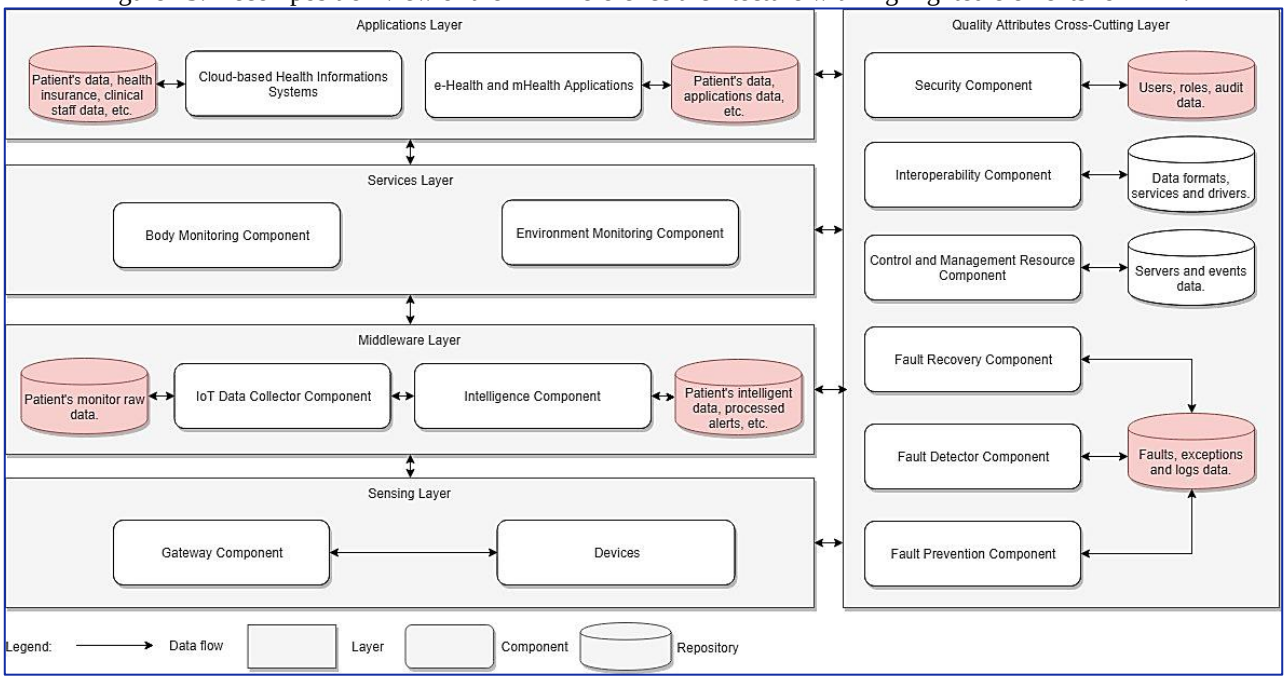


Figure 44: Repository view of the PAR Gateway component.

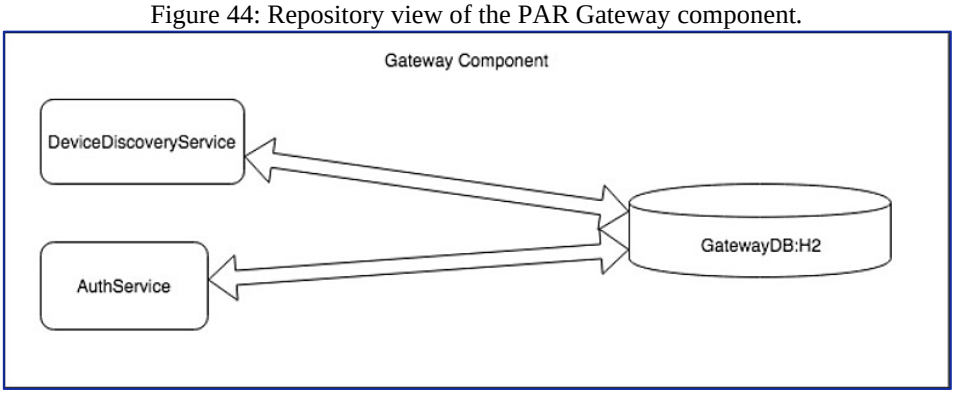
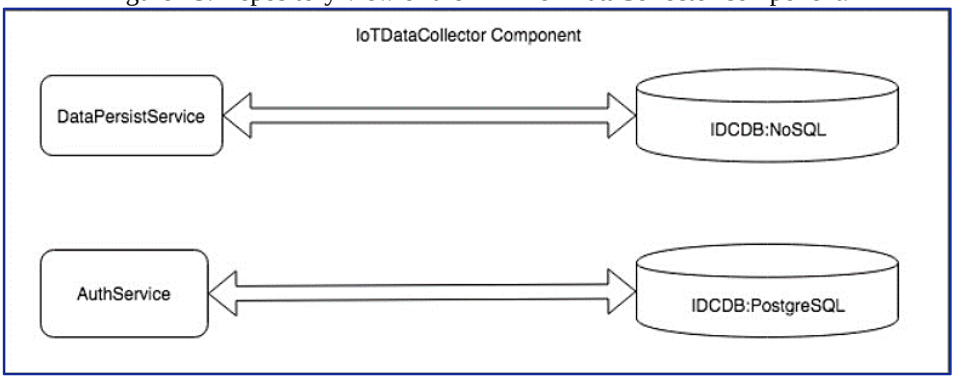
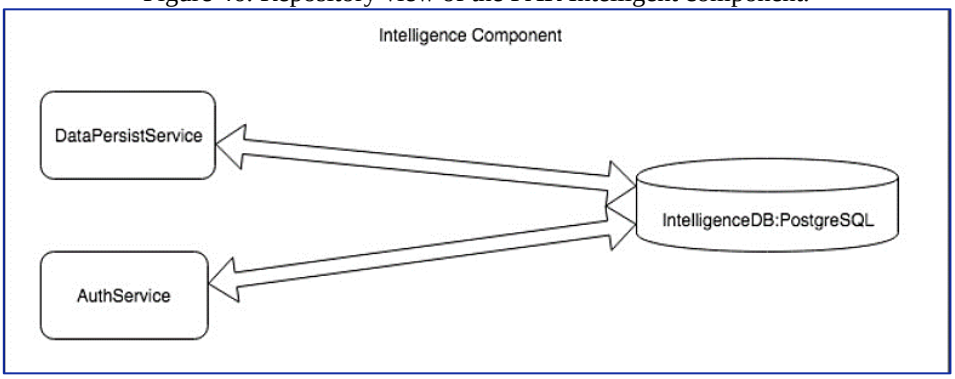


Figure 45: Repository view of the PAR IoTDataCollector component.



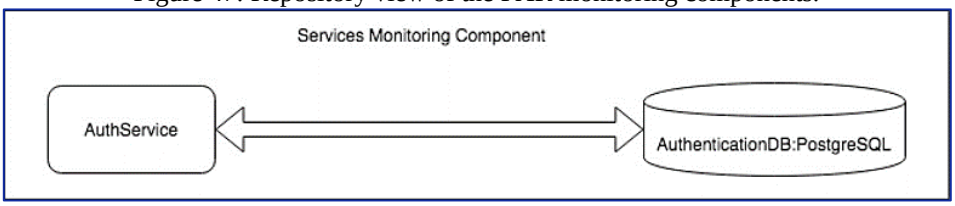
The Intelligence component uses a single database in your services, that is described in the repository view presented in Figure 46. The DataPersistService is the main service of access to the database IntelligenceDB since it is used by other Intelligence services, which require the data to carry out their operations. For example, the inference engine constantly needs to classify the data of the devices. AuthService also makes access to the database to authorize and authenticate those who try to connect to the Intelligence Component for sending data.

Figure 46: Repository view of the PAR Intelligent component.



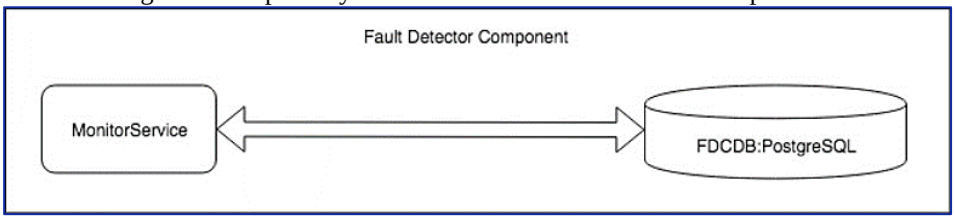
The repository view of the monitoring components (body and environment) is presented in Figure 47. These components do not require operations that imply the storage of information. However, they have a database of an AuthService that performs authorization and authentication of applications that request information from patients.

Figure 47: Repository view of the PAR monitoring components.



The Fault Detector Component (FDC) has its relational database, which is accessed by the MonitorService, and it is used for registering the status information about the running components of PAR. The FDC repository view is presented in Figure 48.

Figure 48: Repository view of the PAR Fault Detector component.



Resources - This procedure took 118 hours:

1. Nine persons, of which two were nurses, one was the RAH's software architect, one was the PAR's software architect and five were developers, were involved in the PAR's requirements and use cases specifications and documentation. This activity took 26 hours to complete.
2. Two persons, the software architect of PAR and the software architect of RAH were involved in the identification of what are RAH's services and components that address the defined requirements. This activity took 9 hours to complete.
3. Two persons, the software architect of PAR and the software architect of RAH were involved in the design of PAR's services and components to attend to the defined requirements, based on RAH's instantiation. Most of the work made in this activity was under the responsibility of the architect of PAR. The PAR's software architect spent 67 hours to complete this activity. The RAH's software architect spent 16 hours in this activity, to resolve doubts and in the reviewing meetings made jointly with the software architect of PAR. Considering the time spent by both architects, this activity demanded 83 hours to be completed.

6.2.2 Procedure 2 - Specifying and documenting quality scenarios

This procedure is oriented to support the activity of architectural evaluation of PAR. Specifically, this procedure gives evidence to assess architectural decisions regarding quality attribute requirements. In this context, four quality scenarios were used to validate the software architecture of PAR regarding quality attributes of interoperability, availability, security, and performance. For each quality attribute one scenario was proposed following the guidelines offered by Clements et al. (CLEMENTS et al., 2003). The four quality scenarios are presented as follows.

6.2.2.1 Scenario 1 - Interoperability scenario

Attribute: Syntactic and Semantic Interoperability between Devices and the Platform;

Environment: The platform is connected to a multi-parametric monitor that sends data using HL7 V2.6. This data goes through the Gateway, and it sends the raw data to the IoTDataCollector. This data is transformed by the IoTDataCollector (syntactic) and it is classified according to the patient's critical values configuration (semantic) by the Intelligent Component. Finally, the data is presented to the applications by the Service Layer.

Stimulus: An e-health shield needs to be connected to the platform to monitor new patient biometrics. This data is sent by the e-health shield using a hashmap (key, value structure) where the key is the sensor's name and the value is the captured data. This data needs to be received by the

Gateway, transformed by the IoTDataCollector Component, and classified by the Intelligent Component. Finally, this data needs to be presented to the applications by the Service Layer.

Response: The data is received and sent by the Gateway. It is transformed in the IoTDataCollector, classified according to the patient's critical values configuration by the Intelligent component, and presented in the applications by the Service Layer.

Architectural decisions:

- Development of a driver for the e-health shield in the Gateway Component. To create this driver, it is used the data format service responsible for defining the data format used in the platform components (syntactic interoperability). This e-Health shield driver understands its protocol (hashmap) and converts the received data in the Gateway into a format understandable by the IoTDataCollector (JSON).
- Definition of this driver in the Driver service for e-health shields. This driver can be reused for the communication of a new e-health shield.
- Registration of the e-health shield in the Authorization service. With this authorization, the raw data of this device can be received and processed in the Gateway.
- Usage of transformation data service in the IoTDataCollector to transform the raw data received and persisted in a format understandable by the Intelligent Component, to be classified according to the patient's critical values configuration (semantic interoperability).

Reasoning:

- **Benefits:** (i) Allows to put new and change existing devices. (ii) Syntactic and semantic interoperation between devices and the platform components. (iii) Reusability of drivers, transformation, and data formats services.
- **Liabilities:** (i) Necessity to define new drivers and data formats for unknown devices. (ii) Definition of a transformation rule for the transformation data service.

Architectural diagrams: PAR layered, decomposition, and component and connector views, in Figures 32, 33, 36, 37, and 38.

6.2.2.2 Scenario 2 - Availability scenario

Attribute: Availability;

Environment: The platform is connected to a multi-parametric monitor and e-health shield that sends data through the Gateway. The Gateway sends it to the IoTDataCollector. This data is transformed by the IoTDataCollector, and it is interpreted by the Intelligent Component. Finally, this data is presented to the applications by the Service Layer.

Stimulus: The IoTDataCollector stops sending the data to the Intelligent Component, presenting an increase in the memory load average.

Response: The monitor service detects the possible failure and alerts the administrators of the application providing the health status of the IoTDataCollector; the retry service attempts three times to process and send received data to the Intelligence component. The limit is reached and this service uses the removal service to declare a failure of the IoTDataCollector and put it in an out-of-service state. The exceptions are captured and logged by the exception services. The redundancy service activates another node in the protection group of the IoTDataCollector.

Architectural decisions:

- Definition of a monitor, retry, exceptions, removal, redundancy, and state resynchronization services;
- Registration of the platform services and components to be monitored by the monitor service;
- Definition in the retry service to perform three attempts to process and send data before declaring failure of a component/service;
- Definition in the redundancy service of the redundancy configuration (hot, warm, or cold spare), and the location of the protection group with redundant nodes of the components and services;
- Based on the redundancy configuration of the redundancy service, the definition of the strategy of the state resynchronization service (checksum, hash-function, or check-pointing).

Reasoning:

- **Benefits:** (i) Detection of failures is automatic. (ii) Provision of mechanisms to capture exceptions for analyzes. (iii) Provision of redundancy of the components.
- **Liabilities:** (i) Negative Impact on the performance of the components and services monitored since the response requires information processing demanding additional time to answer the monitor service. (ii) The necessity of additional infrastructure and computational resources to the copy of nodes of the components for redundancy.

Architectural diagrams: PAR layered, decomposition, and component and connector views, in Figures 32, 33, 36, 38, and 41.

6.2.2.3 Scenario 3 - Security scenario

Attribute: Security;

Environment: The platform is connected to a multi-parametric monitor and e-health shield that sends data through the Gateway. The Gateway sends it to the IoTDataCollector. This data is transformed by the IoTDataCollector, and it is interpreted by the Intelligent Component. Finally, this data is presented to the applications by the Service Layer.

Stimulus: A hacker connects a device in the network and tries to send data to the platform posing as an existing patient.

Response: The hacker's attached device is not achieve to send data to the platform.

The administrators of the platform are notified of the attempt of intrusion.

Architectural decisions:

- Definition of authorization and authentication services to the devices and components of the platform;
- Register the devices and components on the authorization service. The authorization and authentication services contain a database of registered devices and components specifying the IP, type, and authorization token. In each exchanged packet of data in the platform, the authorization token of each device/component is present. The authorization service verifies this token, and if it is not valid, the packet is discarded;
- Definition of encryption service for the packet of data exchanged by the platform components/services and devices. Encryption is essential to ensure the confidentiality of the data, protecting the token and the patient's data. The packet of data is encrypted and sent using HTTP protocol;
- Definition of security information and intrusion detection services to detect the attempt of intrusion and notification of the platform administrators.

Reasoning:

- **Benefits:** (i) Improvement of detection of unauthorized access. (ii) Provision of mechanisms to register and authenticate the authorized devices. (iii) Provision of encryption to protect patient's data. (iv) Provision of mechanisms to detect and notify intrusion attempts.

- **Liabilities:** Negative impact on performance, since every sent data is verified if it comes from an authorized device.

Architectural diagrams: PAR layered, decomposition, and component and connector views, in Figures 32, 33, 36, 37, and 38.

6.2.2.4 Scenario 4 - Performance scenario

Attribute: Performance;

Environment: The platform is monitoring 1000 patients. They are connected to multiparametric monitors, e-health shields, and environment sensors that send data through the Gateway. The Gateway sends it to the IoTDataCollector. This data is transformed by the IoTDataCollector, and it is interpreted by the Intelligent Component. Finally, this data is presented to the applications by the Service Layer.

Stimulus: A network instability occurs in the Gateway damning the packets of patients' monitored data. All the packets that should be sent during the network instability are queued and sent at once to the Intelligent Component too rapidly to be processed by it.

Response: The Intelligent Component queues the packets of monitored data until they can be processed by it. All the packets are processed by this component and sent to the Services layer components to be available to the applications Applications Layer.

Architectural decisions:

- Definition of an event response service to control resource demand in the Intelligent Component;
- Definition of a limit of the maximum rate of received packets to process. If the rate of the received packets passes the set maximum rate, the number of excess packets is queued to be processed later with the next packets.

Reasoning:

- **Benefits:** (i) Performance controlled. (ii) Guarantees that even in a stressful situation in the platform, it will attend to the requests. (iii) No "downsample" of packets of patients' monitored data, since it is not acceptable to lose any packet.

- **Liabilities:** (i) The necessity of additional computational resources to the queues of the components. (ii) The necessity to ensure that the queues are large enough to handle the excess of packets of patients monitored data in the worst case.

Architectural diagrams: PAR layered, decomposition, and component and connector views, in Figures 32, 33, 36, 38, 39, and 41.

Resources - This procedure took 32 hours:

1. To establish quality scenarios, both software architects, of PAR and RAH, were involved.
2. 20 hours spent by the architect of PAR analyzing the architectural decisions, benefits, and liabilities for the scenarios.
3. 12 hours spent by the architect of RAH designing and validating the scenarios.

6.2.3 Procedure 3 - Implementing the platform based on software architecture designed

This procedure is oriented to collect evidence of the development of PAR based on the concrete architecture instantiated from RAH. Thus, in this procedure, PAR was implemented, and its deployment view is presented in Figure 49. This view details the physical structure of virtual machines and servers for this platform. Moreover, it also shows the layout of PAR software artifacts, developed in this procedure, based on the software architecture of PAR.

The devices and servers allocated for this platform are presented in Table 18.

Table 18: Devices and servers allocated for PAR.

Devs/servers	Responsibility	OS	Resources
Raspberry PI 3 - Model B	Gateway	Raspbian	Quad-core, 1GB of RAM
e-Health Shield	Shield sensor	-	-
Omni 6122	Multi-parametric monitor	-	-
chi.imd.ufrn.br	Components of middleware and service layer	CentOS	8vCPUs, 8GB of RAM, and 20GB of storage
imam.imd.ufrn.br	Fault components	CentOS	8vCPUs, 8GB of RAM and 20GB storage
par.imd.ufrn.br	Applications	CentOS	8vCPUs, 8GB of RAM and 20GB storage

For the development of the software artifacts of PAR, the technologies presented in Table 19 were used.

Table 19: Technologies used in the development of PAR.

Technology	Frameworks
Languages	Java, JavaScript
Frameworks	Spring MVC, Hibernate
IDEs	Eclipse
Databases	H2, PostgreSQL and MongoDB
Broker	ActiveMQ

The software artifacts created by the developer's team of PAR are presented and detailed as follows.

6.2.4 Gateway.jar

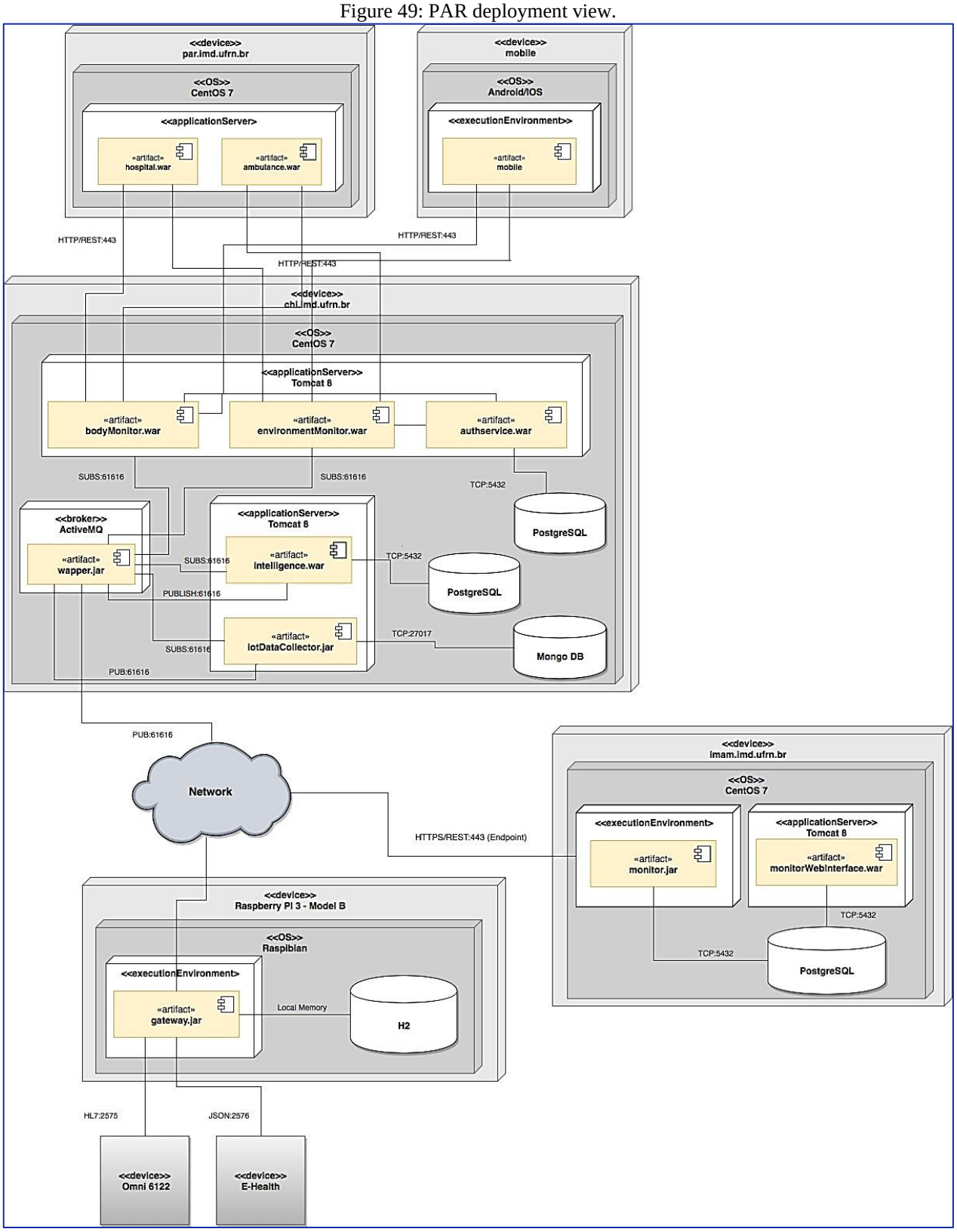
This artifact is a stand-alone component that is responsible for performing the data capture of the monitoring equipment and sending them to the IotDataCollector artifact. It is deployed in the PAR platform infrastructure on the Raspberry PI 3 - Model B. This equipment has two network interfaces, one connected to a network that enables external access to the Internet and another connected to the subnet in which the devices for monitoring patients are located. In this way, this device can communicate with both networks, which allows the capture of data from patient monitoring devices and the subsequent sending of this data to the IotDataCollector artifact, available in the PAR infrastructure.

It was necessary to verify the involved formats of the data in the devices. Thus, the HL7 format was used for the Omni 6122 Multi-Parameter Monitor, and the JSON format for the E-Health Shield device. The implementation of the connection ports for data capture was performed using the DataReceiver service, which provides sockets in the following ports: 2575 / TCP (Omni 6122) and 5000 / TCP (e-HealthShield). All of these formats are carried over the network through the TCP protocol and are made available in the layers above the transport layer of the TCP/IP stack. Thus, after the establishment of the connection through the sockets, the Driver component, which can understand each specific format, is triggered for the consolidation of the data capture. This component knows the specificities of each format and implements the necessary routines for adequate data capture. In gateway.jar, these services are implemented by methods in the DriverEHealth and DriverHL7 classes.

This data is captured and sent, through the RawDataSender service, to the IotDat- aCollector artifact using a publish-subscribe pattern. The authorization service was implemented, which has methods that perform authentication and authorization control for the establishment of a connection between the devices and the gateway. In some cases, the need for the implementation of the Filter service was verified, which will be responsible for filtering the data before sending it to IoTDataCollector. The filter was implemented as a Python application whose function is to remove

noise from digital-analog sensors and converters that are present in the signal provided by e-HealthShield. Thus, its goal is to provide better signal quality for better data extraction in the intelligence component.

Figure 49: PAR deployment view.



6.2.5 IoTDataCollector.jar

The IotDataCollector artifact is the component responsible for receiving the data sent by the Gateway, persisting it in a non-relational database, and then converting it to a format known by the other platform components. This artifact is deployed in server chi.imd.ufrn.br, receives the data coming from the gateway through the publish-subscribe pattern. This communication is established through Apache ActiveMQ1, which is a multi-lingual Message Broker capable of managing a bus for message exchange. This component also implements gateway authentication and authorization to publish data in a particular subscriber topic.

For the IoTDataCollector artifact to receive the data from the Gateway, it is necessary to register and authorize this component to collect the raw data. Once the raw data is received, it is persisted in a non-relational MongoDB2 database through the DataPersistence service. Then it is sent to the Transformation service that performs the conversion of it, through the identification and characterization of different formats (HL7, JSON, etc.) performed by the DataFormat service. This transformation results in a JSON object known by the platform components. After this transformation, the IoTDataSend service publishes the data in the intelligence topic, which will be consumed by the Intelligence artifact.

6.2.6 Intelligence. war

The intelligence artifact, available from the chi.imd.ufrn.br server is responsible for applying inference rules to the data so that it can be semantically understood and present relevant information about the health status of a patient. Thus, in this component, a set of rules is defined and applied directly to this data. The data of the sensors arrive in the Intelligence component through the component IoTDataCollector. The pattern Publish Subscribe does the communication of messages between these components. It is also used ActiveMQ to manage these messages and to implement this component authentication and authorization for the publication of data on a particular topic in the subscriber.

It is necessary to register and authorize the Intelligence component to receive the data from the IoTDataCollector. Once received, the data is sent to the Intelligence Engine service, which is responsible for classifying and applying rules to the data about the patient's condition. If it finds any value outside the defined rules for the sensors, the component generates an alert and sends it to the InfomationSend and Persist services to be stored in a database. The values that are inside the rules that set limits for the sensors are also sent to the InformationSend service.

[1] http://activemq.apache.org/
[2] https://www.mongodb.com

Finally, the Persist service is responsible for storing the alerts generated by the IntelligenceEngineService component in a PostgreSQL[3] relational database. The InformationSend service is responsible for receiving the data analyzed by the IntelligenceEngine service and sending it to the Body Monitoring component, if it is patient sensor data, and to the Environment Monitoring component if it is environment sensor data.

6.2.7 BodyMonitor.war and EnvironmentMonitor.war

The BodyMonitor and EnvironmentMonitor artifacts are available from chi.imd.ufrn.br server is responsible for providing interfaces to applications located on the par.imd.ufrn.br server. In this way, this artifact sends data of HTTP / REST or Publish-Subscribe requests to real-time data related to information about the health status of patients and the environment in which they are accommodated. For the applications to consume the information delivered by these artifacts, they must send in the request for an authentication token produced by the AuthService. With this token, they will be able to validate if the applications that are requesting the data are properly authorized and what the patient's data can be consumed.

6.2.8 AuthService.war

The AuthService artifact is available from the chi.imd.ufrn.br server is responsible for the authentication and authorization of the applications, mobile or web, which will consume data through the interfaces provided by the components BodyMonitor and EnvironmentMonitor. The service will also check which users have permission to access the patient's data history and real-time data captured by the monitoring devices. This authentication of the services is provided through the JSON Web Token (JWT)4 standard.

For the applications to communicate with the BodyMonitor and EnvironmentMonitor components, they will have to authenticate themselves in the AuthService artifact by informing a client-id and a secret-id. When performing the authentication procedure, the AuthService will check whether the requesting application is allowed access to the monitoring components. If the application has the necessary access permissions, a valid token will be generated so that it can connect with the monitoring components.

[3] https://www.postgresql.org
[4] https://jwt.io

6.2.9 Hospital.war, Ambulance. war, and Mobile App

Hospital, Ambulance, and Mobile applications, available from par.imd.ufrn.br, are responsible for displaying information collected from sensors to users. It is important to emphasize that the use cases that concern the presentation of the data to the users defined in the requirements phase are implemented in these components. These applications present, in real-time, data and alerts of the monitored patient's health situation, making it possible for them to be monitored remotely by the clinical staff. For these artifacts to consume data provided through the BodyMonitor and EnvironmentMonitor components in the service layer, registration and permission are required. This authorization is implemented through queries that use the JSON Web Token standard.

The Hospital. war artifact implements the functionalities related to the management of patients and clinical staff, as well as monitoring, reporting, and configuration of critical patient data. The development of this artifact follows the Model View Control pattern. The Mobile artifact presents the monitoring, and display of alerts and reports based on the patients' health data. The Ambulance artifact is responsible for notifying critical alerts due to altered health conditions of monitored patients.

6.2.10 Monitor.jar and MonitorWebInterface.jar

The monitor artifact is available from the imam.imd.ufrn.br server, performs availability checking and collects application and equipment infrastructure monitoring data. Also, this component sends alerts in case of failures to the user and, in certain situations, acts proactive by executing commands for the recovery of failed services and applications. This component implements four main functionalities, which are: periodic verification of the availability of applications and equipment through the implementation of the ping/echo strategy; collection of monitoring data from the SNMP protocol for equipment and servers (virtual or physical) and, for applications, through endpoints made available by applications developed from the Spring framework and application servers such as JBoss/Tomcat; in case of fault identification in the services, this component receives instructions to perform actions of recovery and re-adaptation of computational resources in the environment; and send alert with information to the user in case a fault is identified.

The monitor web interface artifact is complementary to the monitor and will allow users to register the applications and equipment that will be monitored, as well as related endpoints of applications, and attributes that will have data collected through SNMP, among others. Through panel dashboards, the user can check reports, messages, graphs, data processing results, and analysis of the monitored environment. Also, through the web interface, it is possible to define the control steps. In

this way, the user can define commands to be executed by the monitor, in cases of anomalies being identified.

The source code is available in the repositories[5], and the developers made a video[6] that presents PAR execution.

Resources - This procedure took 358 hours:

1. Seven persons, of which five were developers, one was the RAH's software architect, and one was the PAR's software architect, were involved in PAR development.
2. Five persons, all developers, were involved in the Gateway component development. This activity took 63 hours to complete.
3. Five persons, all developers, were involved in the IoTDataCollector component development. This activity took 58 hours to complete.
4. Five persons, all developers, were involved in the Intelligence component development. This activity took 45 hours to complete.
5. Five persons, all developers, were involved in the Body and Environment monitoring component development. This activity took 62 hours to complete.
6. Five persons, all developers, were involved in the Monitor component development. This activity took 48 hours to be completed.
7. Five persons, all developers, were involved in the Hospital, Ambulance, and Mobile applications development. This activity took 70 hours to complete.
8. Two persons, of which one was the RAH's software architect and the other one was the PAR's software architect, were involved in the review of the implemented artifacts to avoid software architecture deviations. This activity took 12 hours to complete.

6.3 ANALYSIS OF COLLECTED DATA

In this section, conclusions are derived based on the collected evidence in Section 6.2. For each research question, conclusive statements are proposed offering evidence to support or refute the related hypothesis.

6.3.1 RQ1 - RAH allows to design of software architectures of IoT-based healthcare applications

To answer RQ1 - *Can a software architecture of an IoT-based healthcare application be designed by using RAH?,* time and people required to conduct and document the instantiation of RAH

[5] https://projetos.imd.ufrn.br/iothealthcareplatform
[6] https://par.imd.ufrn.br/video

was registered. Therefore, at the end of Procedure 1, presented in Section 6.2.1, information about the time spent and people involved in designing the software architecture of PAR was detailed.

To support the hypothesis that *RAH allows to design of software architectures of IoT- based healthcare applications,* the concrete software architecture of PAR was designed, as an instantiation of RAH. In this procedure, presented in Section 6.2.1, the requirements and architectural elements of PAR were mapped into the requirements and architectural elements of RAH. This mapping, presented in Tables 16 and 17, shows that each requirement of PAR is under the responsibility of at least one component and service of RAH. Figures 31 and 43 present the services, components, and repositories instanced of RAH for PAR.

Moreover, the architectural views of PAR were created following the guidelines and views of RAH, as presented in Figures 32, 33, and 34. With this evidence, it is possible to affirm that RAH facilitated the design of the software architecture of PAR, an IoT-based healthcare application. However, additional instantiations of RAH for the design of concrete architectures of IoT-based healthcare applications must be performed to offer more evidence to support this hypothesis.

Finally, as presented in Procedure 1, the RAH's software architect spent 16 hours resolving doubts and in the reviewing meeting made jointly with the software architect of PAR. The PAR's software architect spent 67 hours completing this instantiation of RAH and documentation of PAR. It is possible that this time could be less if there were a specific instantiation process to use with RAH.

6.3.2 RQ2 - RAH addresses interoperability in IoT-based healthcare applications

To answer RQ2 - *Is RAH an alternative to address interoperation issues of IoT-based healthcare applications?*, an interoperability scenario detailed in Procedure 2, Section 6.2.2, was analyzed. This scenario was intended to support the hypothesis that *by using RAH, an architecture of an IoT-based healthcare application can address the interoperability of services provided by the components, devices, and applications.* In IoT-based healthcare applications, interoperability is mainly related to the capacity of integration of new devices and standard communication between participating components and services. Thus, the proposed scenario involved the necessity of connection for a new unknown device (e-health shield) in PAR.

To address interoperability, architectural decisions made and identified by analyzing the proposed scenario include (i) the Development of a driver for an e-health shield in the Gateway Component. This driver uses the data format service responsible for defining the data format used in the platform components (syntactic interoperability), understands its protocol (hashmap), and converts the received data in the Gateway into a format understandable by the IoTDataCollector; (ii) Definition of this driver in the Driver service for e-health shields. This driver can be reused for the

communication of a new e-health shield; (iii) Registration of the e-health shield in the Authorization service. With this authorization, the raw data of this device can be received and processed in the Gateway (security); (iv) Usage of transformation data service in the IoTDataCollector to transform the raw data received and persisted in a format understandable by the Intelligent Component, to be classified according to the patient's critical values configuration (semantic interoperability). The scenario-related architectural diagrams were: PAR layered, decomposition, and component and connector views, presented in Figures 32, 33, 36, 37, and 38.

The proposed scenario also presented the benefits of the architectural decisions, such as syntactic and semantic interoperation between devices and the PAR components, and the possibility of reusability of drivers, transformation, and data formats services. The evidence obtained allows arguing that architectural decisions made in RAH and instantiated in PAR support the hypothesis that by using RAH, an architecture of an IoT-based healthcare application can address the interoperability of services provided by the components, devices, and applications. Therefore, it is possible to address semantic and syntactic interoperability between devices, components, and services in these applications.

6.3.3 RQ3 - RAH address availability in IoT-based healthcare applications

To answer RQ3 - *Is RAH an alternative to address availability issues of IoT-based healthcare applications?,* an availability scenario detailed in Procedure 2, Section 6.2.2, was analyzed. This scenario was intended to support the hypothesis that *by using RAH, software architectures of IoT-based healthcare applications can address the availability of components and services.* In IoT-based healthcare applications, availability refers to a property of services and components that is there and ready to carry out their task when you need it to be.

To address availability, architectural decisions made and identified by analyzing the proposed scenario include (i) Definition of a monitor, retry, exceptions, removal, redundancy, and state resynchronization services; (ii) Registration of the platform components to be monitored by the monitor service; (iii) Definition in the retry service to perform three attempts to send data before declare failure of a component/service; (iv) Definition in the redundancy service of the redundancy configuration (hot, warm or cold spare), and the location of the protection group with redundant nodes of the components and services; (v) Based on the redundancy configuration of the redundancy service, definition of the strategy of the state resynchronization service (checksum, hash-function or check-pointing). The scenario-related architectural diagrams were: PAR layered, decomposition, and component and connector views, presented in Figures 32, 33, 36, 38, and 41.

The proposed scenario also presented the benefits of architectural decisions, such as the detection of failures is automatic, the Provision of mechanisms to capture exceptions for analyzes, provision of redundancy of the components. The evidence obtained allows arguing that architectural decisions made in RAH and instantiated in PAR support the hypothesis that by using RAH, software architectures of IoT-based healthcare applications can address the availability of components and services.

6.3.4 RQ4 - RAH addresses security in IoT-based healthcare applications

To answer RQ4 - *Is it possible to instantiate software architectures of secure IoT- based healthcare applications using RAH?,* a security scenario detailed in Procedure 2, Section 6.2.2, was analyzed. This scenario was intended to support the hypothesis that *by using RAH, software architectures of IoT-based healthcare applications can address security requirements.* In IoT-based healthcare applications, security is related to the application's ability to protect user's (patients, clinical staff, etc.) data and information from unauthorized access while still providing access to people and systems that are authorized.

To address security, architectural decisions made and identified by analyzing the proposed scenario include (i) the Definition of authorization and authentication services to the devices and components of the platform; and (ii) the Register of the devices and components on the authorization service. The authorization and authentication services contain a database of registered devices and components specifying the IP, type, and authorization token. In each exchanged packet of data in the platform, the authorization token of each device/component is present. The authorization service verifies this token, and if it is not valid, the packet is discarded; (iii) Definition of encryption service, and usage of encryption between the components and devices. Encryption is essential to ensure the confidentiality of the data, protecting the token and the patient's data. The packet of data is encrypted and sent using HTTP protocol; (iv) Definition of security information and intrusion detection services to detect the attempt of intrusion and notification of the platform administrators. The scenario-related architectural diagrams were: PAR layered, decomposition, and component and connector views, presented in Figures 32, 33, 36, and 37.

The proposed scenario also presented the benefits of the architectural decisions, such as the improvement of detection of unauthorized access, provision of mechanisms to register and authenticate the authorized devices, provision of encryption to protect patient data and provision of mechanisms to detect and notify intrusions attempts. The evidence obtained allows arguing that architectural decisions made in RAH and instantiated in PAR support the hypothesis that by using RAH, software architectures of IoT-based healthcare applications can address security requirements.

Therefore, it is possible to provide authentication, authorization, intrusion detection, and encryption between devices, components, and services in these applications.

6.3.5 RQ5 - RAH addresses performance in IoT-based healthcare applications

To answer RQ5 - *Is RAH an alternative to address performance issues of IoT-based healthcare applications?,* a performance scenario detailed in Procedure 2, Section 6.2.2, was analyzed. This scenario was intended to support the hypothesis that *by using RAH, software architectures of IoT-based healthcare applications can address the performance of components and services.* In IoT-based healthcare applications, performance is related to time and the application's ability to meet timing requirements. This attribute is critical since time can be decisive in a life-and-death situation.

To address performance, architectural decisions made and identified by analyzing the proposed scenario include (i) a Definition of an event response service to control resource demand in the Intelligent Component; and (ii) a Definition of a limit of the maximum rate of received packets to process. If the rate of the received packets passes the set maximum rate, the number of excess packets is queued to be processed later with the next packets. The scenario-related architectural diagrams were: PAR layered, decomposition, and component and connector views, presented in Figures 32, 33, 36, 38, 39, and 41.

The proposed scenario also presented the benefits of architectural decisions, such as performance controlled, guarantees that even in a stress situation the platform will attend to the requests, and no "downsample" of packets of patients monitored data, since it is not acceptable to lose any packet. The evidence obtained allows arguing that architectural decisions made in RAH and instantiated in PAR support the hypothesis that by using RAH, software architectures of IoT-based healthcare applications can address the performance of components and services.

6.3.6 RQ6 - RAH allows to design and implement software architectures of IoT- based healthcare applications

To answer RQ6 - *Can a software architecture of IoT-based healthcare application, de- signed using RAH, be implementable?,* time and people required to implement PAR, as an instantiation of RAH, was registered. Therefore, at the end of Procedure 3, presented in Section 6.2.3, information about the time spent and people involved in the development of the artifacts of PAR was detailed. To support the hypothesis that *By using RAH, it is possible to design and implement software architectures of IoT-based healthcare applications,* The concrete software architecture of PAR was implemented.

In this procedure, presented in Section 6.2.3, the Gateway (Gateway.jar), IoTDataCollector (*IoTDataCollector.jar*), Intelligence (*Intelligence. war*), Body and Environment monitoring (*BodyMonitor.war* and *EnvironmentMonitor.war*), FaultDetector (*Monitor.jar and MonitorWebInterface.jar*), and Applications (*Hospital. war, Ambulance. war* and *Mobile App*) components were implemented based on the following architectural diagrams: PAR layered, decomposition, repositories, and component and connector views, presented in Figures 32, 33, 36, 37, 38, 39, 40, 42, 41, 44, 45, 46, 47 and 48. Moreover, the PAR deployment view, presented in Figure 49, describes the mapping between the PAR components and connectors and the hardware on which the platform executes.

As presented in Procedure 3, the team composed of the developers, and software architects of PAR and RAH spent 358 hours to create the proposed components for PAR. Finally, the evidence obtained in this procedure allows arguing that it is possible to implement software architectures of IoT-based healthcare applications instanced from RAH. However, additional instantiations of RAH for the design and implementation of concrete software architectures of IoT-based healthcare applications must be performed to offer more evidence to support this hypothesis.

6.4 DISCUSSION OF RESULTS

The conduction of this case study allowed the software architecture design and implementation of PAR. PAR, which is an IoT-Based healthcare platform, was designed as an instantiation of RAH reference architecture for IoT-based healthcare applications. This instantiation procedure was detailed in Section 6.2.1, and it was not possible to measure the time and effort required to understand how to use RAH adequately. It is believed that this learning curve of RAH can affect the time associated with this procedure. The collected evidence presented that is possible to affirm that RAH allowed the design of the software architecture of PAR, an IoT-based healthcare application. However, more instantiation of RAH is needed to support the hypothesis that this reference architecture allows designing software architectures for IoT-based healthcare applications.

Moreover, the evidence of this procedure revealed that is necessary to define techniques to assist the instantiation, verification, and validation processes in the use of RAH. An approach for continuous updating RAH must be established, since new elements, stakeholders and requirements of this kind of application can appear. The lack of updates might lead to its misuse since without updates its components cannot contemplate the possible elements in new IoT-based healthcare applications. Additionally, these updates could be used to ensure the sustainability, evolution, and maturity of RAH and its instances over time. Another important fact is that the current version of RAH does not support code generation of its instantiated architectures, and does not provide common

components and services, such as Gateway, IoTDataCollector, Intelligence, Fault Detector, etc. The existence of standard components and services could help in the instantiation and implementation process of IoT-based healthcare applications.

After the design of the software architecture of PAR, it was started a procedure of architectural evaluation, presented in Section 6.2.2. In this procedure, four scenarios were created and used to validate this software architecture regarding quality attributes of interoperability, availability, security, and performance. The hypothesis involved the capabilities of this instanced architecture of RAH to address these quality attributes.

The evidence obtained allows arguing that architectural decisions made in RAH and instantiated in PAR confirm the hypothesis that by using RAH, software architectures of IoT-based healthcare applications can address these attributes.

Finally, following with case study conduction, PAR was implemented. This implementation, presented in Section 6.2.3, was performed to support the hypothesis that by using RAH, it is possible to design and implement software architectures of IoT-based health- care applications. Thus, ten artifacts related to the components and services defined in RAH were implemented based on the architectural diagrams documented for PAR and presented in Section 6.2.1. The collected evidence presented that is possible to affirm that RAH allowed the design and implementation of the software architecture of PAR. It is believed that some of these components and services, such as the Gateway, IoTDataCollector, Intelligence, and Monitoring components could be offered in standard versions to facilitate the instantiation and implementation procedures. Moreover, PAR could be evolved into a middleware for IoT-based healthcare applications. However, is necessary more study in this regard.

6.5 THREATS TO VALIDITY

The validity of a study denotes the trustworthiness of the results, and to what extent the results are true and not biased by the researchers' subjective point of view (RUNESON; HOST, 2009). Thus, to ensure the validity of the results obtained conducting the case study presented in this chapter, the four aspects of validity proposed by Runeson and Host (RUNESON; HOST, 2009) were considered, namely, construct, internal, external, and reliability of the study. For each threat to the validity aspects, one or more approaches to mitigate its impact in results analysis were proposed, and are presented as follows.

Construct validity: This aspect of validity reflects what extent the operational measures that are studied represent what the researcher has in mind and what is investigated according to the research questions (RUNESON; HOST, 2009). To avoid threats to the construct validity, the

guidelines proposed by Runeson and Host (RUNESON; HOST, 2009) were followed to support the planning, conduction, analysis, and reporting of this case study. Moreover, the case study planning was reviewed to ensure the correct execution of the study. Hence, the general objective, research questions, units of analysis, and collected data were reviewed by a software engineering researcher before the case study was conducted.

Internal validity: This aspect of validity is of concern when causal relations are examined. When the researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor (RUNESON; HOST, 2009). The following factors that could prejudice the objective of this case study were identified: (i) the Learning curve of RAH, which had no impact on results since this reference architecture was presented to the architect of PAR before conducting the study; (ii) The comprehension of architectural views of RAH, its components, and services, which was prevented, since this reference architecture was described using views, and the responsibility of each service and component was described; (iii) The experience of the software architect of PAR in the process of documenting software architectures, which was resolved, since this process was presented and reviewed before conducting the study; (iv) The experience of the developers with the technologies used to develop PAR, which was resolved, since all the developers had previous experience using these technologies before the case study conduction. (v) The problems in defining the scope of the IoT- based healthcare application, which was resolved through the involvement of nurses for the definition and validation of PAR requirements and documentation.

External validity: This aspect of validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case (RUNESON; HOST, 2009). The documentation of RAH, and its example of instantiation for the design of the concrete software architecture and implementation of PAR, an IoT-based healthcare application, could be used to instantiate other concrete software architecture and implementations of these applications. To generalize the findings of this case study, more extensive research should be done involving multiple cases with more variations in different scenarios of IoT-based based healthcare applications. It is possible that during the establishment of other IoT-based healthcare applications, modifications in RAH could be required depending on the specificities of systems under design. However, the results reported here are expressive for IoT-based healthcare applications, since PAR requirements of remote intelligent monitoring were responsible for instantiating 13 of the 14 components proposed in RAH.

Reliability of the study: This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers (RUNESON; HOST, 2009). To improve the reliability of

the results presented in this case study, the guidelines proposed by Runeson and Host (RUNESON; HOST, 2009) were followed. Therefore, the study was designed and planned, defining its objective, research questions, hypothesis, units of analysis, and methods to collect data, as presented in Section 6.1. These data were collected following the planned methods, correctly coded to avoid misunderstandings, and documented in the procedures. Collected data are available in Appendix A, Section 6.2, and PAR source code repositories7 to be consulted by other researchers who desire to replicate the results obtained in this study. Finally, a qualitative analysis of these data was performed and reported in Section 6.3.

6.6 FINAL REMARKS

In this chapter, the results of evaluating RAH were presented. This evaluation was made through the conduction of a case study that was designed, planned, conducted, and reported following the guidelines proposed by Runeson and Host (RUNESON; HOST, 2009). The objective of this case study is to validate the suitability of RAH to support the software architecture design of IoT-based healthcare applications. Therefore, the software architecture of PAR, an IoT-based healthcare application for intelligent remote monitoring of patients in a critical situation, was designed as an instance of RAH. Moreover, it was specified and documented for scenarios to support the architectural evaluation of PAR, regarding quality attributes of interoperability, availability, security, and performance. Following, PAR was implemented considering its proposed software architecture.

The procedures performed in this case study conduction, presented in Section 6.2, allowed to collect of evidence for investigating each research question, presented in Section 6.1, offering support or refuting their related hypothesis. Thus, in Section 6.3, all evidence was analyzed to answer these research questions. Section 6.4 presented a discussion about the results of this case study, revealing the necessity to define techniques to assist the instantiation, verification, and validation processes in the use of RAH, and continuous updating of this reference architecture. Finally, threats to the four validity aspects found in case studies (RUNESON; HOST, 2009), presented in Section 6.5, namely, construct, internal, external, and reliability of the study were identified and mitigated, to ensure the trustworthiness of results obtained in the case study presented in this chapter.

[7] https://projetos.imd.ufrn.br/iothealthcareplatform

The IoT-based technologies are allowing the development of applications in many markets, such as healthcare, manufacturing, electricity, agriculture, and others. Particularly in the healthcare market, it is expected to see the development of applications following this trend as part of the future, since it can improve e-health to allow hospitals to operate more efficiently and patients to receive better treatment. This paradigm is reshaping modern healthcare, connecting everything to the Internet, shifting "from anytime, anyplace connectivity for anyone" to "connectivity for anything." IoT can be the main enabler for distributed healthcare applications, thus having a significant potential to contribute to the overall decrease in healthcare costs while increasing health outcomes. Moreover, with the projections of the increase in population aging and chronic diseases that might result in more patients at hospitals, the use of IoT-based healthcare applications is a strategy to minimize the institutionalization process and the effects of the high cost of patient care.

There is a variety of IoT-based applications that do not contemplate interoperation with other existing systems, and research trends in IoT-based healthcare include network architectures and platforms, new services and applications, interoperability, and security among others (ISLAM et al., 2015). There is also a projection of the development of technologies and applications related to IoT infrastructure for healthcare (AL-FUQAHA et al., 2015). In this scenario, there are a lot of challenges in the development and deployment of this kind of application, such as interoperability, availability, usability, security, flexibility, productivity, and others. This complex and heterogeneous nature of IoT-based healthcare applications makes its design and development difficult. It also causes an increase in the development cost, as well as an interoperability problem with the existing systems.

Therefore, a strategy to design a software reference architecture to systematically organize the main elements of IoT-based healthcare applications, their responsibilities, and their interactions, promotes a common understanding of these applications' architecture. Aiming for guidelines to develop IoT-based applications, several reference architectures have been proposed considering the necessity to address these requirements, but they are too abstract, and none of them is focused on supporting the development of IoT-based healthcare applications. The problem addressed in this book is the lack of guidelines to conduct the development of interoperable, secure, efficient, available, and standardized IoT-based healthcare applications. Its main objective is to establish a reference architecture, named Reference Architecture for IoT-based Healthcare Applications (RAH), to improve the understanding and systematization of the IoT-based healthcare applications' architectural design, and offer guidelines for the development of these applications.

The RAH reference architecture was established and evaluated through the conduction of a case study for the design and implementation of an IoT-based healthcare application, named PAR, for intelligent remote monitoring of patients in a critical situation.

ABAWAJY, J. H.; HASSAN, M. M. Federated internet of things and cloud computing pervasive patient health monitoring system. IEEE Communications Magazine, IEEE, v. 55, n. 1, p. 48–53, 2017.

AEHLERT, B. J. ACLS Study Guide-E-Book. [S.l.]: Elsevier Health Sciences, 2012.

AHMADI, H. et al. The application of internet of things in healthcare: a systematic literature review and classification. Universal Access in the Information Society, Springer, p. 1–33, 2018.

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. Communications Surveys & Tutorials, IEEE, IEEE, v. 17, n. 4, p.2347–2376, 2015.

AL-TAEE, M. A. et al. Mobile health platform for diabetes management based on the internet-of-things. In: IEEE. Applied Electrical Engineering and Computing Technologies (AEECT), 2015 IEEE Jordan Conference on. [S.l.], 2015. p. 1–5.

ANGELOV, S.; GREFEN, P. An e-contracting reference architecture. Journal of Systems and Software, Elsevier, v. 81, n. 11, p. 1816–1844, 2008.

ANGELOV, S.; GREFEN, P.; GREEFHORST, D. A framework for analysis and design of software reference architectures. Information and Software Technology, Elsevier, v. 54, n. 4, p. 417–431, 2012.

ANGELOV, S.; TRIENEKENS, J.; GREFEN, P. Towards a method for the evaluation of reference architectures: Experiences from a case. Software architecture, Springer, p. 225–240, 2008.

ARCHIP, A. et al. An iot based system for remote patient monitoring. In: IEEE. Carpathian Control Conference (ICCC), 2016 17th International. [S.l.], 2016. p. 1–6. ARKKO, J. et al. Architectural considerations in smart object networking. 2015.

ASSOCIATION, A. D. Cheking Your Blood Glucose, http://www.diabetes.org/living- with-diabetes/treatment-and-care/blood-glucose-control/checking-your-blood-glucose.html. 2017. Disponível em: <http://www.diabetes.org/living-with-diabetes/treatment-and- care/blood-glucose-control/checking-your-blood-glucose.html>.

ASSOCIATION, A. H. Understanding Blood Pressure Readings, http://www.heart.org. 2017. Disponível em: <http://www.heart.org/>.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. Comput. Netw., Elsevier North-Holland, Inc., New York, NY, USA, v. 54, n. 15, p. 2787–2805, out. 2010. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>.

BACHMANN, F. et al. Documenting Software Architectures: Views and Beyond. [S.l.]: Addison-Wesley Professional, 2011.

BANERJEE, P. et al. Everything as a service: Powering the new information economy. Computer, IEEE, v. 44, n. 3, p. 36–43, 2011.

BARNAGHI, P. et al. Semantics for the internet of things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS), IGI Global, v. 8, n. 1, p. 1–21, 2012.

BARROCA, I.; AQUINO, G.; LIMA, M. A. Promoting better healthcare for patients in critical condition: An iot-based solution to integrate patients, physicians, and ambulance services. In: Next-Generation Mobile and Pervasive Healthcare Solutions. [S.l.]: IGI Global, 2018. p. 1–21.

BARROCA, I.; AQUINO, G. S. Iot-based healthcare applications: A review. In: SPRINGER. International Conference on Computational Science and Its Applications. [S.l.], 2017. p. 47–62.

BARROCA, I.; AQUINO, G. S. Proposing an iot-based healthcare platform to integrate patients, physicians and ambulance services. In: SPRINGER. International Conference on Computational Science and Its Applications. [S.l.], 2017. p. 188–202.

BARROCA, I.; AQUINO, G. S. A software reference architecture for iot-based healthcare applications. In: SPRINGER. International Conference on Computational Science and Its Applications. [S.l.], 2018. p. 173–188.

BASS, L.; CLEMENTS, P.; KAZMA, R. Software Architecture in Practice (3rd Edition). [S.l.]: Addison-Wesley, 2013.

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice. [S.l.]: Addison-Wesley Professional, 2003.

BASSI, A. et al. Enabling things to talk. [S.l.]: Springer, 2016.

BAUER, M. et al. Iot reference model. In: Enabling Things to Talk. [S.l.]: Springer, 2013. p. 113–162.

BAZERBASHI, H. et al. Low tissue oxygen saturation at emergency center triage is predictive of intensive care unit admission. Journal of critical care, Elsevier, v. 29, n. 5, p. 775–779, 2014.

BLACKMAN, S. et al. Ambient assisted living technologies for aging well: a scoping review. Journal of Intelligent Systems, De Gruyter, v. 25, n. 1, p. 55–69, 2016.

BOLINDER, J. et al. Novel glucose-sensing technology and hypoglycaemia in type 1 diabetes: a multicentre, non-masked, randomised controlled trial. The Lancet, Elsevier, v. 388, n. 10057, p. 2254–2263, 2016.

CAHILL, T. J.; PRENDERGAST, B. D. Infective endocarditis. The Lancet, v. 387, n. 10021, p. 882 – 893, 2016. ISSN 0140-6736. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140673615000677>.

CALVARESI, D. et al. Exploring the ambient assisted living domain: a systematic review. Journal of Ambient Intelligence and Humanized Computing, Springer, v. 8, n. 2, p. 239–257, 2017.

CASTILLEJO, P. et al. Integration of wearable devices in a wireless sensor network for an e-health application. Wireless Communications, IEEE, IEEE, v. 20, n. 4, p. 38–49, 2013.

CHEN, M. et al. Wearable 2.0: Enabling human-cloud integration in next generation healthc are systems. IEEE Communications Magazine, IEEE, v. 55, n. 1, p. 54–61, 2017.

CHEN, Y. Analyzing and visual programming internet of things and autonomous decentralized systems. [S.l.]: Elsevier, 2016.

CHIUCHISAN, I.; COSTIN, H.-N.; GEMAN, O. Adopting the internet of things technologies in health care systems. In: IEEE. Electrical and Power Engineering (EPE), 2014 International Conference and Exposition on. [S.l.], 2014. p. 532–535.

CLEMENTS, P. et al. Documenting Software Architectures: Views and Beyond. 2. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321552687.

CLEMENTS, P. et al. Evaluating software architectures. [S.l.]: Tsinghua University Press Beijing, 2003.

COUTURIER, J. et al. How can the internet of things help to overcome current healthcare challenges. 2012.

DATTA, S. K. et al. Applying internet of things for personalized healthcare in smart homes. In: IEEE. Wireless and Optical Communication Conference (WOCC), 2015 24th. [S.l.], 2015. p. 164–169.

DOUKAS, C.; MAGLOGIANNIS, I. Bringing iot and cloud computing towards pervasive healthcare. In: IEEE. Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on. [S.l.], 2012. p. 922–926.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. Inf. Softw. Technol., Butterworth-Heinemann, Newton, MA, USA, v. 50, n. 9-10, p. 833–859, ago. 2008. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2008.01.006>.

EBERT, C. et al. Reference architectures for the internet of things. IEEE Software, p. 112–116, 2016.

EWER, A. K. Pulse oximetry screening: do we have enough evidence now? The Lancet, Elsevier, v. 384, n. 9945, p. 725–726, 2014.

FAN, Y. J. et al. Iot-based smart rehabilitation system. Industrial Informatics, IEEE Transactions on, IEEE, v. 10, n. 2, p. 1568–1577, 2014.

FREMANTLE, P. A reference architecture for the internet of things. WSO2 White paper, 2014.

GAO, R. et al. Web-based motion detection system for health care. In: IEEE. Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on. [S.l.], 2015. p. 65–70.

GASPARRINI, A. et al. Mortality risk attributable to high and low ambient temperature: a multicountry observational study. The Lancet, Elsevier, v. 386, n. 9991, p. 369–375, 2015.

GIA, T. N. et al. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In: IEEE. Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. [S.l.], 2015. p. 356–363.

GIA, T. N. et al. Customizing 6lowpan networks towards internet-of-things based ubiquitous healthcare systems. In: IEEE. NORCHIP, 2014. [S.l.], 2014. p. 1–6.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. Future Gener. Comput. Syst., Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 29, n. 7, p. 1645–1660, set. 2013. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2013.01.010>.

HALL, J. E. Guyton and hall textbook of medical physiology. Philadelphia: Saunders Elisever, 2011.

HASSAN, M. M.; ALBAKR, H. S.; AL-DOSSARI, H. A cloud-assisted internet of things framework for pervasive healthcare in smart city environment. In: ACM. Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities. [S.l.], 2014. p. 9–13.

HIREMATH, S.; YANG, G.; MANKODIYA, K. Wearable internet of things: Concept, architectural components and promises for person-centered healthcare. In: IEEE. Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on. [S.l.], 2014. p. 304–307.

HOCHRON, S.; GOLDBERG, P. Driving physician adoption of mheath solutions. Healthcare Financial Management, Healthcare Financial Management Association, v. 69, n. 2, p. 36–40, 2015.

HOFMEISTER, C.; NORD, R.; SONI, D. Applied software architecture. [S.l.]: Addison-Wesley Professional, 2000.

HOSSAIN, M. S.; MUHAMMAD, G. Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring. Computer Networks, Elsevier, v. 101, p. 192–202, 2016.

HU, F.; XIE, D.; SHEN, S. On the application of the internet of things in the field of medical and health care. In: IEEE. Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing. [S.l.], 2013. p. 2053–2058.

IDC. IDC FutureScape: Worldwide IT Industry 2017 Predictions, https://www.idc.com/getdoc.jsp?containerId=US41883016. 2017. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=US41883016>.

ISLAM, S. R. et al. The internet of things for health care: a comprehensive survey. IEEE Access, IEEE, v. 3, p. 678–708, 2015.

JARA, A. J.; ZAMORA-IZQUIERDO, M. A.; SKARMETA, A. F. Interconnection framework for mhealth and remote monitoring based on the internet of things. Selected Areas in Communications, IEEE Journal on, IEEE, v. 31, n. 9, p. 47–65, 2013.

JARA, A. J.; ZAMORA, M. A.; SKARMETA, A. F. Drug identification and interaction checker based on iot to minimize adverse drug reactions and improve drug compliance. Personal and ubiquitous computing, Springer, v. 18, n. 1, p. 5–17, 2014.

JSON. Introducing JSON, http://www.json.org/. 2016. Disponível em: <http://www.json.org/>.

KEVIN, I. et al. A wearable internet of things mote with bare metal 6lowpan protocol for pervasive healthcare. In: IEEE. Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom). [S.l.], 2014. p. 750–756.

KHATTAK, H. A. et al. Coap-based healthcare sensor networks: A survey. In: IEEE. Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014. [S.l.], 2014. p. 499–503.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering. [S.l.], 2007.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, O. P. Using mapping studies as the basis for further research–a participant-observer case study. Information and Software Technology, Elsevier, v. 53, n. 6, p. 638–651, 2011.

KODALI, R. K.; SWAMY, G.; LAKSHMI, B. An implementation of iot for healthcare. In: IEEE. Intelligent Computational Systems (RAICS), 2015 IEEE Recent Advances in. [S.l.], 2015. p. 411–416.

LAPLANTE, P. A.; LAPLANTE, N. The internet of things in healthcare: Potential applications and challenges. IT Professional, IEEE, v. 18, n. 3, p. 2–4, 2016.

LE-PHUOC, D. et al. Live linked open sensor database. In: ACM. Proceedings of the 6th International Conference on Semantic Systems. [S.l.], 2010. p. 46.

LIN, S.; CRAWFORD, M.; MELLOR, S. The industrial internet of things volume g1: Reference architecture. Industrial Internet Consortium, 2017.

LÓPEZ, P. et al. Survey of internet of things technologies for clinical environments. In: IEEE. Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on. [S.l.], 2013. p. 1349–1354.

MAKSIMOVIĆ, M.; VUJOVIĆ, V.; PERIŠIĆ, B. A custom internet of things healthcare system. In: IEEE. 2015 10th Iberian Conference on Information Systems and Technologies (CISTI). [S.l.], 2015. p. 1–6.

MARTÍNEZ-FERNÁNDEZ, S. et al. Benefits and drawbacks of software reference architectures: A case study. Information and Software Technology, Elsevier, v. 88, p. 37–52, 2017.

MEDICINET. Definition of Heart Rate, http://www.medicinenet.com/script/main/art.asp?articlekey=3674. 2017. Disponível em: <http://www.medicinenet.com/script/main/art.asp?articlekey=3674>.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. Ad Hoc Networks, Elsevier, v. 10, n. 7, p. 1497–1516, 2012.

MOHAMMED, J. et al. Internet of things: Remote patient monitoring using web services and cloud computing. In: IEEE. Internet of Things (iThings), 2014 IEEE International Conference on. [S.l.], 2014. p. 256–263.

MOULLEC, Y. L. et al. A modular 6lowpan-based wireless sensor body area network for health-monitoring applications. In: IEEE. Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA). [S.l.], 2014. p. 1–4.

MULLER, G. A reference architecture primer. Eindhoven Univ. of Techn., Eindhoven, White paper, 2008.

ORGANIZATION, W. H. mHealth New horizons for health through mobile technologies, http://www.who.int/goe/publications/goe_mhealth_web.pdf. 2011. Disponível em: <http://www.who.int/goe/publications/goe_mhealth_web.pdf>.

ORGANIZATION, W. H. E-health, http://www.euro.who.int/en/health- topics/Health-systems/e-health/e-health-readmore. 2018. Disponível em: <http://www.euro.who.int/en/health-topics/Health-systems/e-health/e-health- readmore>.

PERERA, C. et al. Context aware computing for the internet of things: A survey. IEEE Communications Surveys & Tutorials, IEEE, v. 16, n. 1, p. 414–454, 2014.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: EASE. [S.l.: s.n.], 2008. v. 8, p. 68–77.

POENARU, E.; POENARU, C. A structured approach of the internet-of-things ehealth use cases. In: IEEE. E-Health and Bioengineering Conference (EHB), 2013. [S.l.], 2013. p. 1–4.

QI, J. et al. Advanced internet of things for personalised healthcare systems: A survey. Pervasive and Mobile Computing, Elsevier, v. 41, p. 132–149, 2017.

RAAD, M. W.; SHELTAMI, T.; SHAKSHUKI, E. Ubiquitous tele-health system for elderly patients with alzheimer's. Procedia Computer Science, Elsevier, v. 52, p. 685–689, 2015.

RAY, P. P. Internet of things for sports (iotsport): An architectural framework for sports and recreational activity. In: IEEE. Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on. [S.l.], 2015. p. 1–4.

ROZANSKI, N.; WOODS, E. Software systems architecture: working with stakeholders using viewpoints and perspectives. [S.l.]: Addison-Wesley, 2012.

RUNESON, P.; HOST, M. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, Springer US, v. 14, n. 2, p. 131–164, 2009. ISSN 1382-3256. Disponível em: <http://dx.doi.org/10.1007/s10664-008-9102-8>.

SEBESTYEN, G. et al. ehealth solutions in the context of internet of things. In: Proc. IEEE Int. Conf. Automation, Quality and Testing, Robotics (AQTR 2014), Cluj-Napoca, Romania. [S.l.: s.n.], 2014. p. 261–267.

SHARMA, V. et al. Spark: personalized parkinson disease interventions through synergy between a smartphone and a smartwatch. In: SPRINGER. International Conference of Design, User Experience, and Usability. [S.l.], 2014. p. 103–114.

STANKOVIC, J. A. Research directions for the internet of things. IEEE Internet of Things Journal, IEEE, v. 1, n. 1, p. 3–9, 2014.

STRAVOSKOUFOS, K.; SOTIRIADIS, S.; PETRAKIS, E. Iot-a and fiware: bridging the barriers between the cloud and iot systems design and implementation. In: Proc. 6th Int'l Conf. Cloud Computing and Services Science. [S.l.: s.n.], 2016. p. 146–153.

SUNDMAEKER, H. et al. Vision and challenges for realising the internet of things. Cluster of European Research Projects on the Internet of Things, European Commision, v. 3, n. 3, p. 34–36, 2010.

SWIATEK, P.; RUCINSKI, A. Iot as a service system for ehealth. In: IEEE. e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on. [S.l.], 2013. p. 81–84.

TABISH, R. et al. A 3g/wifi-enabled 6lowpan-based u-healthcare system for ubiquitous real-time monitoring and data logging. In: IEEE. 2nd Middle East Conference on Biomedical Engineering. [S.l.], 2014. p. 277–280.

TAN, L.; WANG, N. Future internet: The internet of things. In: IEEE. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. [S.l.], 2010. v. 5, p. V5–376.

THABIT, H.; BALLY, L.; HOVORKA, R. Available at a flash: a new way to check glucose. Elsevier, 2016.

TRCEK, D.; BRODNIK, A. Hard and soft security provisioning for computationally weak pervasive computing systems in e-health. IEEE Wireless Communications, IEEE, v. 20, n. 4, p. 22–29, 2013.

VALK, S. van der et al. Sensor networks in workplaces: Correlating comfort and productivity. In: IEEE. Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on. [S.l.], 2015. p. 1–6.

VIDAL-PETIOT, E. et al. Cardiovascular event rates and mortality according to achieved systolic and diastolic blood pressure in patients with stable coronary artery disease: an international cohort study. The Lancet, Elsevier, v. 388, n. 10056, p. 2142–2152, 2016.

WOHLIN, C. et al. Experimentation in software engineering. [S.l.]: Springer Science & Business Media, 2012.

WU, M. et al. Research on the architecture of internet of things. In: IEEE. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. [S.l.], 2010. v. 5, p. V5–484.

YAAKOB, N. et al. On the effectiveness of congestion control mechanisms for remote healthcare monitoring system in iot environment—a review. In: IEEE. Electronic Design (ICED), 2016 3rd International Conference on. [S.l.], 2016. p. 348–353.

YANG, C.-T. et al. On construction of an intelligent environmental monitoring system for healthcare. In: IEEE. 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies. [S.l.], 2013. p. 246–253.

YANG, G. et al. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. Industrial Informatics, IEEE Transactions on, IEEE, v. 10, n. 4, p. 2180–2191, 2014.

YANG, G. et al. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. IEEE transactions on industrial informatics, IEEE, v. 10, n. 4, p. 2180–2191, 2014.

YANG, Z. et al. Study and application on the architecture and key technologies for iot. In: IEEE. Multimedia Technology (ICMT), 2011 International Conference on. [S.l.], 2011. p. 747–751.

YANG, Z. et al. An iot-cloud based wearable ecg monitoring system for smart healthcare. Journal of medical systems, Springer, v. 40, n. 12, p. 286, 2016.

YAQOOB, I. et al. Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. IEEE wireless communications, IEEE, v. 24, n. 3, p. 10–16, 2017.

ZANCHETTI, A.; THOMOPOULOS, C.; PARATI, G. Randomized controlled trials of blood pressure lowering in hypertension. Circulation research, Am Heart Assoc, v. 116, n. 6, p. 1058–1073, 2015.

ZASLAVSKY, A.; PERERA, C.; GEORGAKOPOULOS, D. Sensing as a service and big data. arXiv preprint arXiv:1301.0159, 2013.

1 HEALTHCARE PLATFORM FOR REMOTE MONITORING OF PATIENTS IN CRITICAL CONDITIONS. VERSION 1.1

This Appendix presents PAR, an IoT-based healthcare platform for intelligent mote monitoring of patients, developed to evaluate RAH software reference architecture, presented in Chapter 5. It describes the actors and uses cases that guided its development.

Table 20: Review History

Date	Version	Description	Authors
13.07.2018	1.0	This document is a specification of use cases of PAR, an IoT-based Health care Platform for intelligent remote monitoring of patients in critical condition.	Cephas Barreto, Itamir Filho, Rafael Queiroz, Lúcio Oliveira, Rubem Kalebe, Katia Maria and Maria Alzete.
06.08.2018	1.1	Improvements in grammar, further clarifications on the use cases, and titles on figures.	Cephas Barreto, Itamir Filho, Rafael Queiroz, Lúcio Oliveira,Rubem Kalebe, Katia Maria and Maria Alzete.

1.1 USE CASES SPECIFICATIONS

The next parts of this document will show the aspects of each use case under the most summarized format as possible. Note that, in general, use cases with the term “management” are related to CRUD - Create, Read, Update, and Delete operations.

1.1.1 Patient’s Data Management

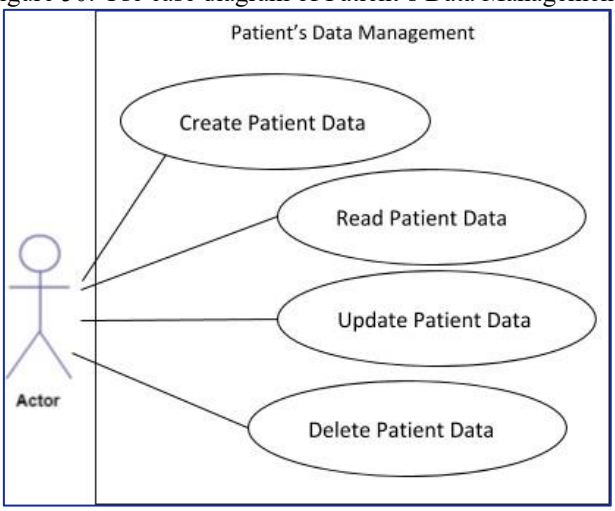
This use case describes tasks for managing patient data. Some context and scenarios are presented for addressing the way actors will use the platform and some observations will also be made, when necessary.

General Preconditions

- ACTOR LOGGED IN. The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.

Use Case Diagram: Figure 50.

Figure 50: Use case diagram of Patient's Data Management.



Actors:

- **Primary Actors:** Hospital Operator; Physicians and Nurses.
- **Secondary Actors:** Hospital Module.

1.1.1.1 Create Patient Data

Basic Flow

1. INSERT PATIENT DATA

a) The platform provides a way to insert data about a patient.

b) The actor inserts the patient data: name, gender, date of birth, contacts, address, family information, physician information (name and contacts), and health insurance information.

2. SAVE PATIENT DATA

a) The actor selects the option Save Patient.

b) The platform validates the inserted patient data.

c) The platform shows a message confirming insertion was done successfully.

d) The use case ends.

Alternative Flows

1. CANNOT SAVE PATIENT

a) PATIENT ALREADY EXISTS. If in **step 2. b** of the basic flow, the platform identifies the patient already exists on the system, then the platform shows a message warning about this. The use case ends.

b) INVALID DATA INSERTED. If in **step 2. b** of the basic flow, the platform identifies invalid data or required data not provided, then the system shows a warning message and does not proceed. The use case resumes at **step 1. b** of the basic flow.

1.1.1.2 Read Patient Data

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) The actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

b) The platform shows the patient data.

3. READ PATIENT DATA

a) The actor reads the patient data.

b) The use case ends.

Alternative Flows

1. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verifies the patient (according to the entered data) does not exist on the system, then the platform shows a message warning about this. The use case ends.

1.1.1.3 Update Patient Data

Specific Preconditions

1. ACTOR READING PATIENT DATA

a) It is necessary that the actor is visualizing the detailed patient data, so he can select the update option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.1.2.**

Basic Flow

1. READ PATIENT DATA

a) Actor is at **step 3. a** of the **Use Case A.1.1.2.**

2. CHOOSE UPDATE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Update*.

3. MODIFY PATIENT DATA

a) The platform shows the previous patient data in an editable way.

b) The actor modifies any patient data as needed.

4. UPDATE PATIENT DATA

a) The actor selects Update Patient.

b) The platform validates the inserted patient data.

c) The platform shows a message confirming the update operation was performed successfully.

d) The use case ends.

Alternative Flows

1. CANNOT UPDATE PATIENT

a) INVALID DATA INSERTED. If in **step 4. b** of the basic flow, the platform verifies that are invalid data or required data not provided, then the system shows a warning message and does not proceed with the use case. Finally, the use case resumes at **step 3. b** of the basic flow.

1.1.1.4 Delete Patient Data-Specific Preconditions

1. ACTOR READING PATIENT DATA

a) It is needed that the actor visualizes the detailed patient data, so he can select the delete option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.1.2.**

Basic Flow

1. READ PATIENT DATA

a) Actor is at **step 3. a** of the **Use Case A.1.1.2.**

2. CHOOSE DELETE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Delete*.

3. CONFIRM DELETE

a) The platform asks the actor to confirm the deletion.

b) The actor confirms the operation inserting his password.

4. DELETE PATIENT DATA

a) The platform marks the patient as inactive.

b) The platform shows a message confirming the delete operation was performed successfully.

c) The use case ends.

Alternative Flows

1. CANNOT DELETE PATIENT

a) WRONG PASSWORD. If in **step 3. b** of the basic flow, the platform asks for confirmation and receives a wrong password, then the system shows a message warning about this. The use case resumes at **step 3.** of the basic flow.

1.1.2 Clinical Staff Data Management

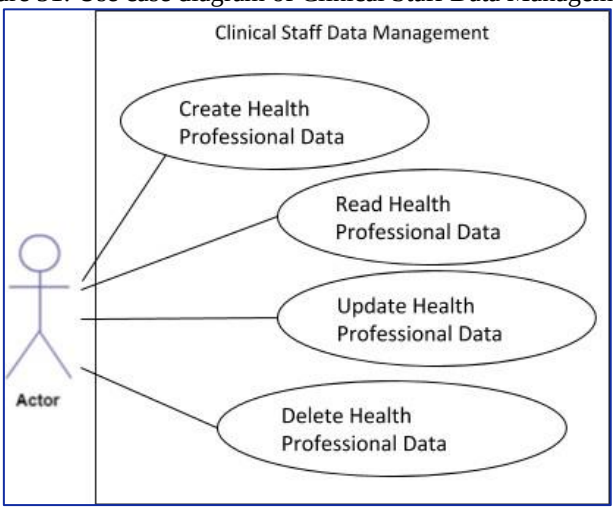
This use case describes tasks for managing clinical staff data. The clinical staff includes data from physicians and nurses. Along with this document, the members of the clinical staff will be called health professionals.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.

Use Case Diagram: Figure 51.

Figure 51: Use case diagram of Clinical Staff Data Management.



Actors:

- **Primary Actors:** Hospital Operator.
- **Secondary Actors:** Hospital Module.

1.1.2.1 Create Health Professional Data

Basic Flow

1. INSERT HEALTH PROFESSIONAL DATA

a) The platform provides a way to insert data of a health professional.

b) The actor inserts the health professional data: name, gender, date of birth, contacts, address, and medical specialty.

c) The actor chooses the professional type: *physician* or *nurse*.

2. SAVE HEALTH PROFESSIONAL DATA

a) The actor selects the option *Save Health Professional.*

b) The platform validates the health professional data.

c) The platform shows a message confirming insertion was done successfully.

d) The use case ends.

Alternative Flows

1. CANNOT SAVE HEALTH PROFESSIONAL

a) HEALTH PROFESSIONAL ALREADY EXISTS. If in **step 2. b** of the basic flow, the platform identifies the health professional already exists on the system, then the platform shows a message warning about this. The use case ends.

b) INVALID DATA INSERTED. If in **step 2. b** of the basic flow, the platform identifies invalid data or required data not provided, then the system shows a warning message and does not proceed. The use case resumes at **step 1. b** of the basic flow.

1.1.2.2 Read Health Professional Data

Basic Flow

1. FIND HEALTH PROFESSIONAL

a) The platform provides a way to find a health professional.

b) Actor inserts only one or a combination of ID; CPF, health professional name, and its professional type.

c) The platform shows a list of health professionals that matches the inserted data.

2. CHOOSE A HEALTH PROFESSIONAL

a) The actor selects a health professional.

b) The platform shows the health professional data.

3. READ HEALTH PROFESSIONAL DATA

a) The actor reads the health professional data.

b) The use case ends.

Alternative Flows

1. CANNOT FIND HEALTH PROFESSIONAL

a) HEALTH PROFESSIONAL DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verifies the health professional (according to the entered data) does not exist on the system, then the platform shows a message warning about this. The use case ends.

1.1.2.3 Update Health Professional Data

Specific Preconditions

1. ACTOR READING HEALTH PROFESSIONAL DATA

a) It is necessary that the actor is visualizing the detailed health professional data, so he can select the update option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.2.2.**

Basic Flow

1. READ HEALTH PROFESSIONAL DATA

a) Actor is at **step 3**. a of the **Use Case A.1.2.2.**

2. CHOOSE UPDATE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Update*.

3. MODIFY HEALTH PROFESSIONAL DATA

a) The platform shows the previous health professional data in an editable way.

b) The actor modifies any health professional data as needed.

4. UPDATE HEALTH PROFESSIONAL DATA

a) The actor selects the option *Update Health Professional.*

b) The platform validates the inserted health professional data.

c) The platform shows a message confirming the update operation was performed successfully.

d) The use case ends.

Alternative Flows

1. CANNOT UPDATE HEALTH PROFESSIONAL

a) INVALID DATA INSERTED. If in **step 4. b** of the basic flow, the platform verifies that are invalid data or required data not provided, then the system shows a warning message and does not proceed with the use case. Finally, the use case resumes at **step 3. b** of the basic flow.

1.1.2.4 Delete Health Professional Data

Specific Preconditions

1. ACTOR READING HEALTH PROFESSIONAL DATA

a) It is needed that the actor is visualizing the detailed health professional data, so he can select the delete option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.2.2.**

Basic Flow

1. READ HEALTH PROFESSIONAL DATA

a) Actor is at **step 3. a** of the **Use Case A.1.2.2.**

2. CHOOSE DELETE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Delete*.

3. CONFIRM DELETE

a) The platform asks the actor to confirm the deletion.

b) The actor confirms the operation inserting his password.

4. DELETE HEALTH PROFESSIONAL DATA

a) The platform marks the health professional as inactive.

b) The platform shows a message confirming the deletion was performed successfully.

c) The use case ends.

Alternative Flows

1. CANNOT DELETE HEALTH PROFESSIONAL
   a) WRONG PASSWORD. If in **step 3. b** of the basic flow, the platform asks for confirmation and receives a wrong password, then the system shows a message warning about this. The use case resumes at **step 3.** of the basic flow.

1.1.3 Health Insurance Data Management

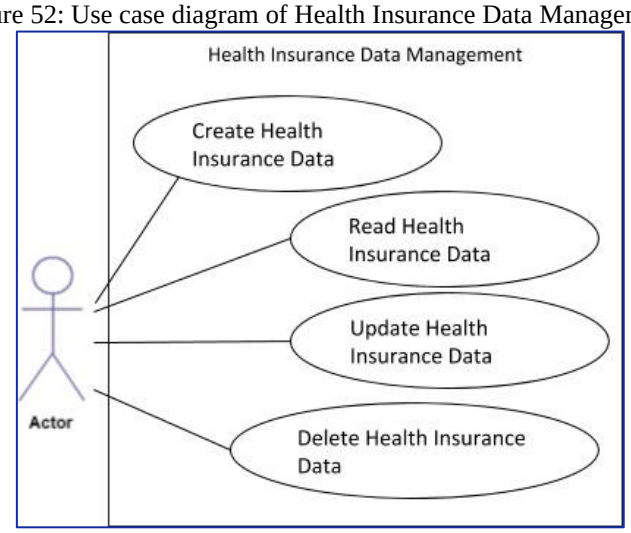
This use case describes tasks for managing health insurance data.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.

Use Case Diagram: Figure 52.

Figure 52: Use case diagram of Health Insurance Data Management.



Actors:

- **Primary Actors:** Hospital Operator.
- **Secondary Actors:** Hospital Module.

1.1.3.1 Create Health Insurance

Basic Flow

1. INSERT HEALTH INSURANCE DATA
   a) The platform provides a way to insert data on health insurance.
   b) The actor inserts the health insurance data: name, code, and initials.

2. SAVE HEALTH INSURANCE DATA
   a) The actor selects the option Save Health Insurance.
   b) The platform validates the health insurance data.
   c) The platform shows a message confirming insertion was done successfully.
   d) The use case ends.

Alternative Flows

1. CANNOT SAVE HEALTH INSURANCE
   a) HEALTH INSURANCE ALREADY EXISTS. If in **step 2. b** of the basic flow, the platform identifies the health insurance already exists on the system, then the platform shows a message warning about this. The use case ends.
   b) INVALID DATA INSERTED. If in **step 2. b** of the basic flow, the platform identifies invalid data or required data not provided, then the system shows a warning message and does not proceed. The use case resumes at **step 1. b** of the basic flow.

1.1.3.2 Read Health Insurance Data

Basic Flow

1. FIND HEALTH INSURANCE
   a) The platform provides a way to find health insurance.
   b) Actor inserts only one or a combination of ID; health insurance name and initials.
   c) The platform shows a list of health insurance that matches the inserted data.

2. CHOOSE A HEALTH INSURANCE
   a) The actor selects health insurance.
   b) The platform shows the health insurance data.

3. READ HEALTH INSURANCE DATA
   a) The actor reads the health insurance data.
   b) The use case ends.

Alternative Flows

1. CANNOT FIND HEALTH INSURANCE
   a) HEALTH INSURANCE DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verifies the health insurance (according to the entered data) does not exist on the system, then the platform shows a message warning about this**. The use case ends.**

1.1.3.3 Update Health Insurance Data

Specific Preconditions

1. ACTOR READING HEALTH INSURANCE DATA
   a) The actor must visualize the detailed health insurance data, so he can select the update option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.3.2.**

Basic Flow

1. READ HEALTH INSURANCE DATA
   a) Actor is at **step 3. a** of the **Use Case A.1.3.2.**

2. CHOOSE UPDATE
   a) The platform provides two options to be performed: *Update* and *Delete*.
   b) The actor chooses *Update*.

3. MODIFY HEALTH INSURANCE DATA
   a) The platform shows the previous health insurance data in an editable way.
   b) The actor modifies any health insurance data as needed.

4. UPDATE HEALTH INSURANCE DATA
   a) The actor selects the option *Update Health Insurance.*
   b) The platform validates the inserted health insurance data.
   c) The platform shows a message confirming the update operation was performed successfully.
   d) The use case ends.

Alternative Flows

1. ANNOT UPDATE HEALTH INSURANCE
   a) INVALID DATA INSERTED. If in **step 4. b** of the basic flow, the platform verifies that are invalid data or required data not provided, then the system shows a warning message and does not proceed with the use case. Finally, the use case resumes at **step 3. b** of the basic flow.

1.1.3.4 Delete Health Insurance Data

Specific Preconditions

1. ACTOR READING HEALTH INSURANCE DATA
   a) It is needed that the actor must visualize the detailed health insurance data, so he can select the delete option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.3.2.**

Basic Flow

1. READ HEALTH INSURANCE DATA
   a) Actor is at **step 3. a** of the **Use Case A.1.3.2.**

2. CHOOSE UPDATE
   a) The platform provides two options to be performed: *Update* and *Delete*.
   b) The actor chooses *Delete*.

3. CONFIRM DELETE
   a) The platform asks the actor to confirm the deletion.
   b) The actor confirms the operation inserting his password.

4. DELETE HEALTH INSURANCE DATA
   a) The platform marks the health insurance as inactive.
   b) The platform shows a message confirming the deletion was performed successfully.
   c) The use case ends.

Alternative Flows

1. CANNOT DELETE HEALTH INSURANCE
   a) WRONG PASSWORD. If in **step 3. b** of the basic flow, the platform asks for confirmation and receives a wrong password, then the system shows a message warning about this. The use case resumes at **step 3.** of the basic flow.

1.1.4 Patient and Health Professional Association

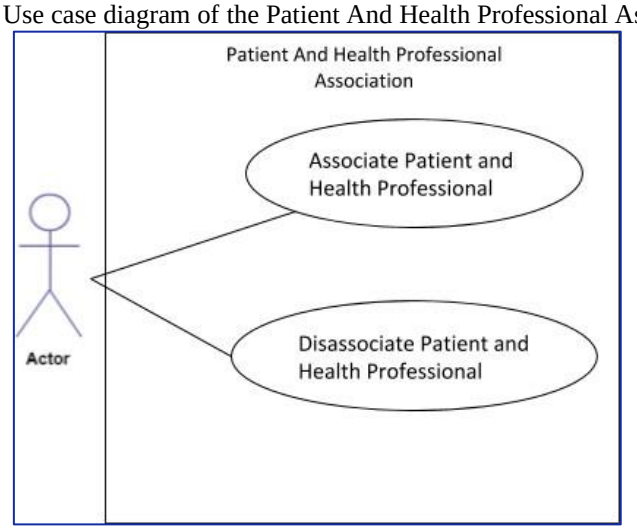
This use case describes tasks for associating and disassociating patients and health professionals.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.
- *PATIENT EXISTS.* It is a precondition that the patient is already registered in the platform.
- *PHYSICIAN EXISTS.* It is a precondition that the physician is already registered in the platform.

Use Case Diagram: Figure 53.

Figure 53: Use case diagram of the Patient And Health Professional Association.



Actors:

- **Primary Actors:** Hospital Operator.
- **Secondary Actors:** Hospital Module.

1.1.4.1 Associate Patient with Health Professional

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. FIND HEALTH PROFESSIONAL

a) The platform provides a way to find health professionals.

b) Actor inserts only one or a combination of ID; medical specialty and health professional name.

4. CHOOSE HEALTH PROFESSIONALS

a) The platform shows a list of health professionals that matches the inserted data.

b) The actor selects one or more health professionals.

5. ASSOCIATE

a) Platform provides a way to associate health professionals and patients.

b) Actor selects *Associate Patients and Health Professionals.*

6. CONFIRM ASSOCIATION

a) The platform asks the actor for confirmation.

b) The actor confirms the association.

c) The use case ends.

Alternative Flows

1. FIND MORE HEALTH PROFESSIONALS

a) In **step 5. b** of the basic flow, the actor can do a new search for other health professionals to be associated with the patient. The platform provides a way to perform a new search with the same parameters. Then, the use case resumes at **step 3.** of the basic flow.

2. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

3. CANNOT FIND HEALTH PROFESSIONAL

a) HEALTH PROFESSIONAL DATA DO NOT EXIST. If in **step 4. a** of the basic flow, the platform verifies the health professional data does not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

1.1.4.2 Disassociate Patient with Health Professional

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. IEW ASSOCIATED HEALTH PROFESSIONALS

a) The platform provides a way to visualize the health professionals associated with the patient.

4. CHOOSE HEALTH PROFESSIONAL

a) The actor chooses one or more health professionals.

5. DISASSOCIATE HEALTH PROFESSIONAL

a) The platform provides a way to choose *disassociate health professionals*.

b) The actor chooses *Disassociate*.

6. CONFIRM DISASSOCIATION

a) The platform asks the actor for confirmation.

b) The actor confirms the disassociation.

c) The use case ends.

Alternative Flows

1. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

1.1.5 Patient's Critical Values Configuration

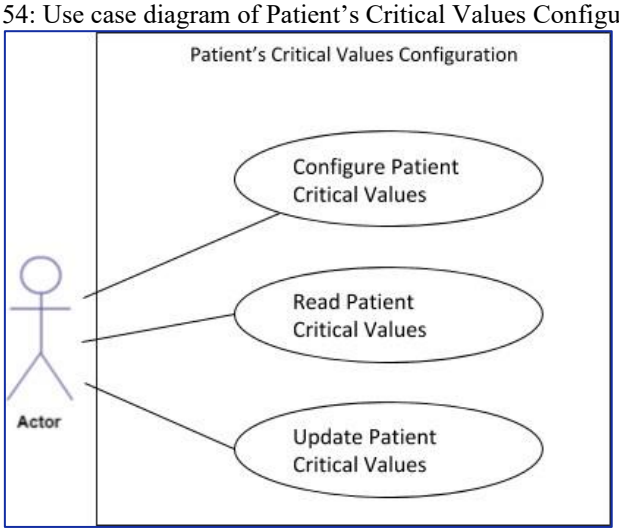
This use case describes tasks for the very first configuration of health-critical values and also provides a means to update these values.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.
- *PATIENT EXISTS.* It is a precondition that the patient is already registered in the platform.

Use Case Diagram: Figure 54.

Figure 54: Use case diagram of Patient's Critical Values Configuration.



Actors

- **Primary Actors:** Nurse, Physician.
- **Secondary Actors:** Intelligence Module.

1.1.5.1 Configure Patient Critical Values

Basic Flow

1. GET DEFAULT CRITICAL VALUES
   a) At the moment of patient creation, the platform must get updated default values for all health parameters of the body and also of the environment.

2. CRITICAL VALUES ATTRIBUTION
   a) The platform attributes values to the created patient.
   b) The use cases end.

Alternative Flows

1. CANNOT CREATE VALUES

a) DEFAULT VALUES DO NOT EXIST. If **in step 1. a** of the basic flow, the platform identifies that default values do not exist, the system asks the actor to manually update the values. The platform provides a way to choose *Update Critical Values Manually.*

1.1.5.2 Read Patient's Critical Values

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. READ PATIENT CRITICAL VALUES

a) The platform provides a way to read the patient's critical values.

b) The actor reads the patient's critical values.

c) The use case ends.

Alternative Flows

1. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

1.1.5.3 Update Patient Critical Values

Specific Preconditions

1. THE ACTOR VISUALIZES THE PATIENT'S CRITICAL VALUES. The actor must visualize the patient's critical values, so he can select the update option.Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.5.2**.

Basic Flow

1. READ PATIENT'S CRITICAL VALUES

a) Actor is at **step 3. a** of the **Use Case A.1.5.2.**

2. CHOOSE UPDATE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Update*.

3. MODIFY PATIENT'S CRITICAL VALUES

a) The platform shows the previous patient's critical values in an editable way.

b) The actor modifies any critical value as needed.

4. UPDATE PATIENT CRITICAL VALUES

a) The actor selects *Update Critical Values.*

b) The platform validates the critical values.

5. CONFIRM UPDATE

a) The platform asks the actor for confirmation.

b) The actor confirms the update with his password.

c) The use case ends.

Alternative Flows

1. CANNOT UPDATE PATIENT'S CRITICAL VALUES

a) INVALID DATA INSERTED. If in **step 4. b** of the basic flow, the platform identifies invalid data or required data not provided, then the system shows a warning message and does not proceed. The use case resumes at **step 3. b.**

b) WRONG PASSWORD. If in **step 5. b** of the basic flow, the platform asks for confirmation and receives a wrong password, and then the system shows a message warning about this. The use case resumes at **step 5.** of the basic flow.

2. SET VALUE TO DEFAULT. In **step 3. b**, the platform must provide a way for the user to choose the default value by the option set to default. The use case continues its flow.

1.1.6 Health Data Management

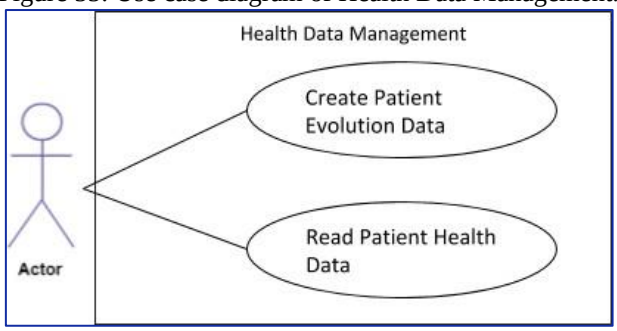
To the platform, health data is composed of body sensors data; environment sensors data, and patient evolution. The next use cases are related to automatic tasks made by sensors and platforms and, also with the interaction and manipulation of this kind of data by health professionals.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.
- *PATIENT EXISTS.* It is a precondition that the patient is already registered in the platform.

Use Case Diagram: Figure 55.

Figure 55: Use case diagram of Health Data Management.



Actors

- **Primary Actors:** Health Professionals.
- **Secondary Actors:** Hospital Module.

1.1.6.1 Create Patient Evolution Data

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. CHOOSE INSERT

a) The platform provides an option for inserting evolution data for a patient: Insert.

b) The actor chooses *Insert.*

4. INSERT PATIENT EVOLUTION DATA

a) The platform provides a way to insert information about the patient's evolution.

b) The actor inserts evolution information, including conscience level and physical exams.

5. CONFIRM INSERTION

a) The platform asks the actor for confirmation.

b) The actor confirms the update with his password.

6. ADD TIME AND SENSORS DATA

a) The platform adds a timestamp to the evolution information.

b) The platform adds data from body and environment sensors to the evolution information.

c) The use case ends.

Alternative Flows

1. INSERT CUSTOM INFORMATION

a) INSERT FREE NOTES. In **step 4. b,** the actor can insert a free annotation about the patient's context and his health that makes sense for future references.

2. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

1.1.6.2 Read Patient Evolution Information Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. LIST EVOLUTION INFORMATION

a) The platform shows a list with evolution information from the last 3 (three) days.

4. CHOOSE AN EVOLUTION INFORMATION

a) The actor chooses an evolution information to see more details.

5. READ PATIENT'S EVOLUTION INFORMATION

a) The platform provides a way to read patient's evolution information.

b) The actor reads the patient's evolution information.

c) The use case ends.

Alternative Flows

1. FIND SPECIFIC DATES. In **step 3.** of the basic flow, the platform provides a way to select evolution information for a specific date range. The actor sets the dates and then the platform shows a list of the patient evolution during that period. The use case ends.

2. CANNOT FIND PATIENT

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

3. CANNOT FIND EVOLUTION INFORMATION

a) EVOLUTION INFORMATION DOES NOT EXIST. If in **step 3. a** of the basic flow, the platform verifies the patient does not have evolution information, the platform shows a message warning about this. The use case ends.

b) DATE RANGE DOES NOT HAVE EVOLUTION INFORMATION. If in **step 1. a** of Alternative Flow 1 (Find Specific Dates), the platform verifies the patient does not have evolution information for that specific date range, the system shows a message warning about this. The use case ends.

1.1.7 Emergency Alert Data Management

To the platform, health data is composed of body sensors data; environment sensors data, and patient evolution. The next use cases are related to automatic tasks made by sensors and platforms and, also with the interaction and manipulation of this kind of data by health professionals.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.
- *PATIENT EXISTS.* It is a precondition that the patient is already registered in the platform.
- *CRITICAL VALUES EXIST.* It is a precondition that the patient's critical values already exist in the platform.

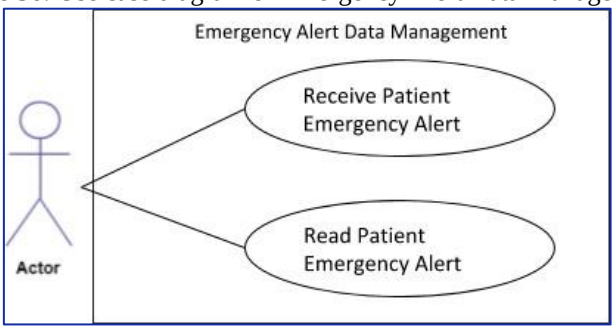
Considerations

It is important to note that alert messages are related to a patient. When an alert is triggered, every situation that results from it is related to the patient as well. In simple words, an alert must work as follows:

1. The platform identifies one or more values out of the limit defined by the patient's critical values;
2. According to the values identified, the platform defines the emergency level from an alert that will be triggered.

Use Case Diagram: Figure 56.

Figure 56: Use case diagram of Emergency Alert Data Management.



Actors

- **Primary Actors:** Health Professionals, Ambulance, Family.
- **Secondary Actors:** Hospital Module.

1.1.7.1 Receive Patient Emergency Alert

Basic Flow

1. RECEIVE PATIENT ALERT
   a) The actor receives an alert from the platform.

2. READ ALERT DATA
   a) The platform provides a way to read data from alerts.
   b) The actor reads data from the alert.
   c) The use case ends.

Alternative Flows

1. READ MORE DETAILS.
   a) SEE REAL TIME DATA. In **step 2. b** of the basic flow, the actor can choose to see in real-time data from sensors in the patient's body and environment. The actor views the data. The use case ends.
   b) SEE PATIENT EVOLUTION INFORMATION. In **step 1. a** of the alternative flow, the actor can choose to see the patient's evolution information. The platform provides a way to choose one or more patients evolution. The actor views the data. The use case ends.

Read Patient Emergency Alert

Basic Flow

1. FIND PATIENT
   a) The platform provides a way to find a patient.
   b) Actor inserts only one or a combination of ID; CPF and patient name.
   c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT
   a) The actor selects a patient.

3. LIST EMERGENCY ALERT
   a) The platform shows a list of alerts from the last 30 days.

4. CHOOSE ONE EMERGENCY ALERT
   a) Actor chooses one alert to see more details.

5. READ EMERGENCY ALERT DETAILS

a) The platform provides a way to read alert details.

b) Actor reads detailed alert information.

c) The use case ends.

Alternative Flows

1. FIND SPECIFIC DATES. In **step 3.** of the basic flow, the platform provides a way to choose emergency alerts of a specific date range. The actor sets the dates and then the platform shows a list of alerts on that period. The use case ends.

2. CANNOT FIND PATIENT.

a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

3. CANNOT FIND EMERGENCY ALERTS

a) EMERGENCY ALERTS DOES NOT EXIST. If in **step 3. a** of the basic flow, the platform verifies the patient does not have emergency alerts, the system shows a message warning about this. The use case ends.

b) DATE RANGE DOES NOT HAVE EMERGENCY ALERTS. If in **step 1. a** of the Alternative Flow 1 (Find Specific Dates), the platform verifies the patient does not have emergency alerts for that specific date range, the system shows a message warning about this. The use case ends.

1.1.8 Ambulance Data Management

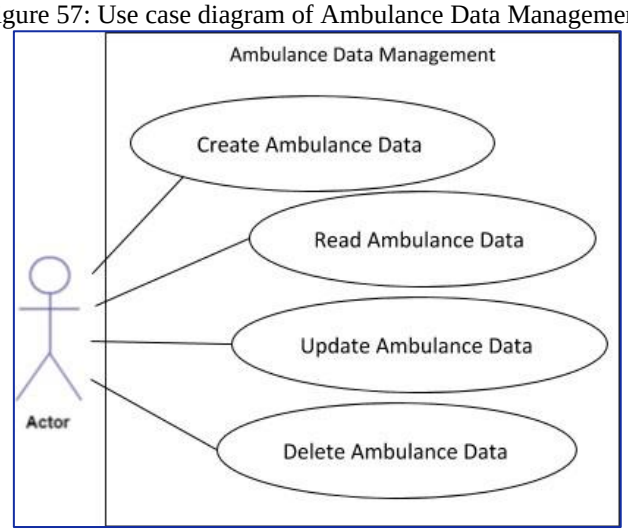
This use case describes tasks for managing ambulance data. Some context and scenarios are presented for addressing the way actors will use the platform for this goal and some observations will also be presented, when needed.

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform.

Use Case Diagram: Figure 57.

Figure 57: Use case diagram of Ambulance Data Management.



Actors

- **Primary Actors:** Ambulance Service Operator.
- **Secondary Actors:** Ambulance Module.

1.1.8.1 Create Ambulance Data

Basic Flow

1. INSERT AMBULANCE DATA
   a) The platform provides a way to insert data on an ambulance.
   b) The actor inserts the ambulance data: plate number, type, and complexity level.

2. SAVE AMBULANCE DATA
   a) The actor selects the option Save Ambulance.
   b) The platform validates the ambulance data.
   c) The platform shows a message confirming insertion was done successfully.
   d) The use case ends.

Alternative Flows

1. CANNOT SAVE AMBULANCE
   a) AMBULANCE ALREADY EXISTS. If in **step 2. b** of the basic flow, the platform identifies the ambulance already exists on the system, then the platform shows a message warning about this. The use case ends.

b) INVALID DATA INSERTED. If in **step 2. b** of the basic flow, the platform identifies invalid data or required data not provided, then the system shows a warning message and does not proceed. The use case resumes at **step 1.** b of the basic flow.

1.1.8.2 Read Ambulance Data

Basic Flow

1. FIND AMBULANCE

a) The platform provides a way to find an ambulance.

b) Actor inserts only one or a combination of ID and Plate Number.

c) The platform shows a list of ambulances that match the inserted data.

2. CHOOSE A AMBULANCE

a) The actor selects an ambulance.

b) The platform shows the ambulance data.

3. READ AMBULANCE DATA

a) Actor reads the ambulance data.

b) The use case ends.

Alternative Flows

1. CANNOT FIND AMBULANCE

a) AMBULANCE DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verifies the ambulance (according to the entered data) does not exist on the system, then the platform shows a message warning about this. The use case ends.

1.1.8.3 Read Ambulance Data

Specific Preconditions

1. ACTOR READING AMBULANCE DATA. The actor must visualize the detailed ambulance data, so he can select the update option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.8.2.**

Basic Flow

1. READ AMBULANCE DATA.

a) Actor is at **step 3. a** of the **Use Case A.1.8.2.**

2. CHOOSE UPDATE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Update.*

3. MODIFY AMBULANCE DATA

a) The platform shows the previous ambulance data in an editable way.

b) The actor modifies any ambulance data as needed.

4. UPDATE AMBULANCE DATA

a) The actor selects the option *Update Ambulance.*

b) The platform validates the inserted ambulance data.

c) The platform shows a message confirming the update operation was performed successfully.

d) The use case ends.

Alternative Flows

1. CANNOT UPDATE AMBULANCE

a) INVALID DATA INSERTED. If in **step 4. b** of the basic flow, the platform verifies that are invalid data or required data not provided, then the system shows a warning message and does not proceed with the use case. Finally, the use case resumes at **step 3. b** of the basic flow.

1.1.8.4 Delete Ambulance Data

Specific Preconditions

1. ACTOR READING AMBULANCE DATA. The actor must visualize the detailed ambulance data, so he can select the update option. Thus, the basic flow continues from **step 3. a** of the **Use Case A.1.8.2.**

Basic Flow

1. READ AMBULANCE DATA.

Actor is at **step 3. a** of the **Use Case A.1.8.2.**

2. CHOOSE DELETE

a) The platform provides two options to be performed: *Update* and *Delete*.

b) The actor chooses *Delete*.

3. CONFIRM DELETE

a) The platform asks the actor to confirm the deletion.

b) The actor confirms the operation inserting his password.

4. DELETE AMBULANCE DATA

a) The platform marks the ambulance as inactive.

b) The platform shows a message confirming the deletion was performed successfully.

c) The use case ends.

Alternative Flows

1. CANNOT DELETE AMBULANCE

a) WRONG PASSWORD. If in **step 3. b** of the basic flow, the platform asks for confirmation and receives a wrong password, then the system shows a message warning about this. The use case resumes at **step 3.** of the basic flow.

1.1.9 Health Data Monitoring and Reporting

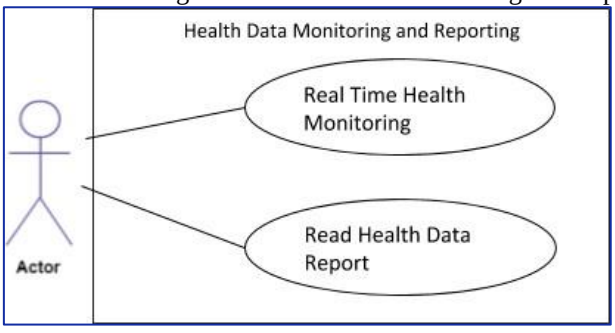
This use case is for real-time monitoring and history of the patient's body and environment (health data).

General Preconditions

- *ACTOR LOGGED IN.* The actor needs to be logged in to the platform. An actor is logged in if his inserted login and password are validated by the platform. Besides that, if the actor is a member of the patient's family, he also can access the platform and it is authorized for visualizing the data.
- *PATIENT EXISTS.* It is a precondition that the patient is already registered in the platform.

Use Case Diagram: Figure 58.

Figure 58: Use case diagram of Health Data Monitoring and Reporting.



Actors

- **Primary Actors:** Health Professionals, Ambulance, Family.
- **Secondary Actors:** Hospital Module.

1.1.9.1 Real-Time Health Monitoring Basic Flow

1. READ REAL-TIME DATA.

a) The platform provides a way to read patient data in real-time.

b) The actor reads real-time data.

c) The use case ends.

Alternative Flows

1. READ MORE DETAILS

a) SEE PATIENT EVOLUTION INFORMATION. In **step 1.** b of the alternative flow, the actor can choose to see the patient's evolution information. The platform provides a way to choose one or more patients evolution. The actor views the data. The use case ends.

b) In **step 1. a** of the alternative flow, the actor can choose to see the patient alert information. The platform provides a way to choose one or more emergency alerts. The actor views the data. The use case ends.

1.1.9.2 Read Health Data Report

Basic Flow

1. FIND PATIENT

a) The platform provides a way to find a patient.

b) Actor inserts only one or a combination of ID; CPF and patient name.

c) The platform shows a list of patients that matches the inserted data.

2. CHOOSE A PATIENT

a) The actor selects a patient.

3. BUILD REPORT SELECT

a) Actor selects the option *Read Report.*

b) The platform builds a report with patient data, health data, and alerts.

4. READ HEALTH DATA REPORT
   a) The platform provides a way to read the report.
   b) The actor reads the report. The use case ends.

Alternative Flows

1. FAMILY MEMBER REPORT
   a) When the actor is a family member or even the patient, the use case must start at **step 3.** of the basic flow. The use case resumes in the same way for the remainder.

2. FIND SPECIFIC DATES
   a) In **step 3. a** of the basic flow, the platform provides a way to generate reports from a specific date range. The actor sets the dates and then the platform produces the report. The use case ends.

3. CANNOT FIND PATIENT
   a) PATIENT DATA DO NOT EXIST. If in **step 1. c** of the basic flow, the platform verify the patient data do not exist on the system (according to the entered data), the platform shows a message warning about this. The use case ends.

4. CANNOT FIND HEALTH DATA
   a) EMERGENCY HEALTH DATA DO NOT EXIST. If in **step 3. a** of the basic flow, the platform verifies the patient does not have any health data, the system shows a message warning about this. The use case ends.
   b) DATE RANGE DOES NOT HAVE HEALTH DATA. If in **step 1. a** of the Alternative Flow 1 (Find Specific Dates), the platform verifies the patient does not have health data for that specific date range, the system shows a message warning about this. The use case ends.

REALIZATION:

SEVEN
publicações acadêmicas

ACCESS OUR CATALOGUE!

WWW.SEVENPUBLI.COM

CONNECTING THE **RESEARCHER** AND **SCIENCE** IN A SINGLE CLICK.