Universidade do Minho
Escola de Engenharia
Departamento de Informática

Filipe André Martins Nunes

New Generation of Interoperable Artefacts in Medical Informatics

May 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Filipe André Martins Nunes

New Generation of Interoperable Artefacts in Medical Informatics

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
José Machado
Vasco Abelha

May 2021

ACKNOWLEDGEMENTS

Firstly, I would like to express my profound gratitude to my father for his sacrifices for educating and preparing me for my future and for being my unwavering support, without which none of this work or the previous five years would have been possible. Furthermore, I would like to thank my brother, grandmother, cousins and uncles for their encouragement and motivation.

I would like to thank my mentor, José Machado, for providing me with this opportunity, and for his patience and support during the development of this dissertation. I would also like to thank Vasco Abelha and Regina Sousa for their commitment and time in assisting me with the editing of this dissertation.

I want to offer my gratitude to the *Hospital da Santa Casa da Misericórdia de Vila Verde*, especially to Armindo Dias and António Ferreira for their assistance and support, as well as for testing the applications and providing their valuable feedback.

I am deeply grateful to all my friends for always being there for me. The last few years would have been unthinkable without all of the moments we have shared and all the support they have provided.

Finally, I would like to thank TecMinho for providing me with the opportunity to participate in the PRM-Patient Relationship Management project and for giving me all of the indispensable support to complete this work.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

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

ABSTRACT

A healthcare institution produces huge amounts of data on a daily basis. These data come from *n* sources and therefore have *m* different formats. As it is evident, this heterogeneity of information sources can be a huge obstacle, since increases the need for this information to be available and shared among the various information systems. The exchange of clinical information between Health Information System (HIS) is crucial for the effective provision of care, significantly improving the performance of the institutions.

In the healthcare industry, the ability of different information and software systems to communicate and share data, as well as use the data exchanged, is called interoperability. This knowledge can be exchanged around the healthcare ecosystem thanks to the use of standards and data sharing models, regardless of the application being used. However, the lack of interoperability remains a concern. The Agency for the Integration, Diffusion and Archive (AIDA) proposes to achieve levels of interoperability never before implemented. For this purpose, web services are used for the processing, dissemination, and archiving of clinical information.

In the context of this master's dissertation, the aim is to develop and explore new information technology artefacts to help the administrative and accounting teams of the *Hospital da Santa Casa da Misericórdia de Vila Verde*. This solution aims to fill the existing gaps between the hospital and the *Assistência na Doença aos Servidores do Estado* (ADSE). The first gap occurs in the dentistry speciality, where it is not possible to verify the patient's ADSE status to perform some dental medicine act under co-payment. The second gap occurs in the invoicing process through ADSE in real-time, in order to perform some medical appointments and/or acts resulting from them since the hospital's accounting team cannot verify if a patient complies with the ADSE requirements. The gaps identified can make ADSE refuse to reimburse the hospital for the medical acts performed, which makes the administrative and accounting work unpleasant. In this situation, the hospital will have to find a solution that suits the patient and the ADSE. Very often, the hospital has to bear the costs.

In order to overpass these problems with the validation and invoicing process of medical acts through ADSE, this project consists of two web applications that enable the hospital's information systems to interoperate with the ADSE web service.

KEYWORDS Health Information Systems, Interoperability, AIDA, Web Services, Medical Acts, ADSE

RESUMO

Diariamente, numa instituição de cuidados de saúde é produzida uma grande quantidade de dados. Esses dados derivam de *n* fontes e, consequentemente, têm *m* formatos diferentes. Como é evidente, esta heterogeneidade de fontes de informação pode ser um grande obstáculo, uma vez que aumenta a necessidade de essa informação ter de estar disponível e ser partilhada entre vários sistemas de informação. A troca de informação clínica entre sistemas de informação na saúde é crucial para uma prestação eficaz dos cuidados de saúde, aumentando significativamente o desempenho das instituições.

Na área da saúde, a capacidade de diferentes sistemas de informação e de diferentes sistemas informáticos para comunicar e partilhar dados, assim como de utilizar esses dados trocados, é chamada de interoperabilidade. Este conhecimento pode ser trocado em todo o ecossistema de saúde graças ao uso de padrões e de modelos de partilha de dados, independentemente da aplicação que está a ser utilizada. Contudo, a falta de interoperabilidade continua a ser preocupante. A *Agency for the Integration, Diffusion and Archive* (AIDA) propôs atingir níveis de interoperabilidade nunca antes atingidos. Para isso, são utilizados *web services* para processar, disseminar e arquivar informação clínica.

No contexto desta dissertação, o objetivo consiste no desenvolvimento e exploração de novos artefactos da tecnologia da informação para ajudar as equipas de gestão e contabilidade do Hospital da Santa Casa da Misericórdia de Vila Verde. Esta solução visa preencher as lacunas existentes entre o hospital e a Assistência na Doença aos Servidores do Estado (ADSE). A primeira falha ocorre na especialidade de medicina dentária, onde não é possível verificar a situação de um paciente na ADSE para praticar um ato odontológico sob comparticipação. A segunda lacuna ocorre no processo de faturação através da ADSE para consultas médicas e/ou atos resultantes destas, uma vez que a equipa da contabilidade do hospital não consegue verificar se o paciente cumpre com os requisitos da ADSE. Estas lacunas podem fazer com que a ADSE se recuse a reembolsar o hospital pelos atos efetuados, o que torna o trabalho da gestão e contabilidade do hospital desagradável. Quanto isto acontece, o hospital tem de encontrar uma solução que satisfaça o paciente e a ADSE o que, na maioria das vezes, passa por ser o hospital a acarretar com os custos.

De maneira a ultrapassar estes problemas com a validação e faturação dos atos médicos através da ADSE, este projeto abrange duas aplicações *web* que permitem a interoperabilidade entre os sistemas de informação do hospital e o *web service* da ADSE.

PALAVRAS-CHAVE Sistemas de Informação na Saúde, Interoperabilidade, AIDA, Web Services, Atos Médicos, ADSE

CONTENTS

LIST OF FIGURES

ACRONYMS

ADSE *Assistência na Doença aos Servidores do Estado*.
AI Artificial Intelligence.
AIDA Agency for the Integration, Diffusion and Archive.
API Application Programming Interfaces.

CORS Cross-Origin Resource Sharing.

DBMS Database Management System.
DDL Data Definition Language.
DICOM Digital Imaging and Communications in Medicine.
DML Data Manipulation Language.
DOM Document Object Model.
DSR Design Science Research.

EHR Electronic Health Record.

HIS Health Information System.
HTML HyperText Markup Language.
HTTP Hypertext Transfer Protocol.

IS Information System.
IT Information Technology.

JSON JavaScript Object Notation.

MAS Multi-Agent Systems.

NTLM New Technology LAN Manager.

PACS Picture Archive and Communication System.
PCR Patient Clinical Record.
PoC Proof of Concept.

REST Representational State Transfer.

SDK Software Development Kit.
SMTP Simple Mail Transfer Protocol.
SOA Service Oriented Architecture.

SOAP Simple Object Access Protocol.
SPA Single Page Application.
SQL Structured Query Language.
SWOT Strengths Weaknesses Opportunities Threats.

UDDI Universal Description, Discovery, and Integration.
UI User Interface.
URL Uniform Resource Locator.

VDOM Virtual DOM.

WSDL Web Services Description Language.
WSIHIS Web Service-Based Integrated Healthcare Information System.

XML eXtensible Markup Language.

Part I

INTRODUCTORY MATERIAL

1

INTRODUCTION

The present document describes the research and analysis for the development and exploration of two different platforms that interact with the *Assistência na Doença aos Servidores do Estado* (ADSE). The project emerges from the master's dissertation of the Integrated Master's in Informatics Engineering at Minho University in the school year of 2019/2020.

There are four parts to this introductory chapter. There is a brief contextualisation of this project, including references to key topics of discussion (Section 1.1), and the main motivation that led to its realisation (Section 1.2). This chapter also contains the main aims proposed for the development of this dissertation, as well as a summary of the work completed (Section 1.3). Finally, it concludes by presenting the document's structure in order to make it easier to read (Section 1.4).

1.1 CONTEXT

In any industry, a tremendous amount of data is generated on a daily basis. Focusing on the healthcare industry, the diversity of data sources and formats increases as does the need for this information to be available and shared among the various Information System (IS). As it is evident, the heterogeneity of information sources (medical applications, software, medical equipment, and even the clinical knowledge of health professionals) is a huge obstacle that needs research and investment [49].

Healthcare systems have spoken different languages for a very long time, making communication and interaction between them difficult. However, information sharing and knowledge are important for the quality of treatment at all levels of the healthcare delivery system - the patient, the care team, and the healthcare organisation - [42]. As a result, one of the primary goals is for IS to exchange and share clinical knowledge. To achieve the smoothest possible exchange of information, it is essential to adopt standards for the different Health Information System (HIS) to share.

In the healthcare industry, interoperability refers to the ability of various ISs and software systems to communicate and share data, as well as use the data exchanged [39]. The use of standards and data exchange models enables this information to be shared between healthcare providers, professionals, patients, hospitals, pharmacies, laboratories, etc., regardless of the application being used [1]. Both for syntactic interoperability, referring to the structure of communication, and for semantic interoperability, which refers to the meaning of communication, the health sector has developed and appropriate standards for various purposes related to messaging, termi-

nology, documents, conceptual schemes, applications, and architectures [38]. However, in health, there are a significant number of standards that can hinder interoperability decisions between systems, so it is important to correctly define and establish policies, standards and guidelines to be implemented [38].

Interoperability in health information systems is becoming more of a necessity than an option. Standards and innovations, such as multi-agent systems, have proved to be useful methods for resolving interoperability concerns [8]. Then, in order to reach interoperability in healthcare units, the Artificial Intelligence Group, in the Informatics Department at the University of Minho, have worked on building the Agency for the Integration, Diffusion and Archive (AIDA). This agency provides intelligent electronic workers in charge of tasks such as communicating with heterogeneous systems, sending and receiving information, managing and saving the information, and answering requests. This platform also supports the use of web-based services to provide direct access, to knowledge and communication resources [2].

Web services provide a single protocol/format for communication that enables multiple applications to send and receive data from web services regardless of the programming language they use. Thus, they can help in the exchange of data between IS in the healthcare environment. To prove it, a case study conducted by Zhang and Xu was made for system interoperability concerns in the healthcare field. The authors concluded, that the use of web services is very promising and offers more advantages for improving interoperability in healthcare.

In this context, this project has emerged, consisting of the development and exploration of two new platforms to help the administrative and accounting teams of the *Hospital da Santa Casa da Misericórdia de Vila Verde* with the validation and invoicing of medical acts through the ADSE. ADSE works as Portuguese health insurance, to which all public employees have access free of charge. Even so, this right may cease for several reasons, such as death, termination of the legal public employment relationship, breach of ADSE rules, among others [56]. The gaps identified make the hospital's administrative and accountant's work unpleasant, since, if any of the situations where the beneficiaries have lost the right to insurance occur, the ADSE will refuse reimbursement. In this situation, both will have to find a solution that suits the patient and also the ADSE. Very often, the solution is for the hospital to bear the costs.

Therefore, this project is comprised of two web platforms that make the interoperability of AIDA web service, available in the hospital, with that of ADSE. It is then intended to overpass these problems with the validation and invoicing of medical acts through the ADSE. Hence, both web applications were designed, developed, and explored, adopting a set of methodologies and technologies available and viable for the conception of the solutions — also called Information Technology (IT) artefacts.

1.2 MOTIVATION

Accordingly, the main motivation for this dissertation project focuses on developing and exploring a new generation of interoperable artefacts in the field of IT in order to help the administrative and accounting teams of the *Hospital da Santa Casa da Misericórdia de Vila Verde*. Therefore, two web applications were developed, using several methodologies and technologies currently available and viable for the conception of the defined IT solutions.

As a first case study, a new web application was developed to overpass an existing gap in dental medicine speciality, in the *Hospital da Santa Casa da Misericórdia de Vila Verde*, where it is impossible to verify the patient situation about ADSE rights. This creates an unpleasant situation for the administrative and accounting work of the hospital since ADSE will refuse to reimburse and they must find a solution that suits the patient and ADSE. Then, the platform must communicate with the ADSE web service in order to verify if a patient can perform some dental medicine act with ADSE co-payment. The application allows dentists to obtain this information quickly and easily before they perform the acts, and allows the hospital to save funds. This application was developed using React by adopting modern technologies and frameworks and allowing the implementation of additional features and modules.

For the second case study, another web application was developed to solve another problem in the hospital with ADSE insurance. The problem occurs in the invoicing process through ADSE for medical appointments and/or acts resulting from them. Because the hospital's accounting team only invoices all the medical acts on the following day, when patients don't comply with ADSE restrictions, this can create an issue for the hospital, since the ADSE may refuse these outgoing invoices. Then, it is necessary that this application invoice the medical acts at the moment of the patient admission in the hospital's reception. This platform should handle first with the patient admission, manual or via citizen card, and after with the invoicing process through ADSE, complying with all the requirements. Like the previous application, this is also developed using React by adopting modern technologies and frameworks.

1.3 OBJECTIVES

Within the scope of this project, the main goal established was to build two web applications to solve the existing issues with the ADSE insurance in the *Hospital da Santa Casa da Misericórdia de Vila Verde*. The first application will be used in the field of dentistry, where dentists can check whether a patient can perform a dental medicine act with ADSE co-payment. The second application is to be applied in the hospital's reception area, where the receptionist will be able to easily process patient admissions and invoice the medical appointments and/or their related acts via ADSE.

Therefore, in the context of this dissertation project, the following Research Questions have arisen:

- **Research Question 1:** What are the hospital's main challenges and necessities with ADSE insurance in the dentistry speciality, and how can the implementation of IT artefacts help optimise the administrative and accounting workflow, reducing costs?

To answer this question, the following Case Study and its objectives were highlighted:

- Case Study 1: Development of a new web platform to validate the dental medicine acts through ADSE:
  - Quick access to basic patient information drawn from the hospital's database and the AIDA web service;
  - Simplification of the validation process through ADSE making it more user-friendly;

  - Improved day-to-day administrative procedures;
  - Reduction of unwanted costs to the hospital resulting from wrong or not allowed requests to ADSE.

- **Research Question 2:** What are the hospital's main challenges and necessities with the invoicing process of medical appointments and/or their related acts through ADSE insurance, and how can the implementation of IT artefacts help optimise the administrative and accounting workflow, reducing costs?

To answer this question, the following Case Study and its objectives were proposed:

- Case Study 2: Development of a new web platform to invoice the hospital acts through ADSE:
  - Quick access to basic patient and episode information drawn from the hospital's database and the AIDA web service;
  - Simple and secure patient admission into the hospital, manually or via his citizen card;
  - Instantaneous production and submission of the medical appointment and/or their related acts invoice in ADSE;
  - Improved day-to-day administrative and accounting procedures;
  - Reduction of unwanted costs to the hospital resulting from delayed or not allowed requests to ADSE.

1.4 DOCUMENT ORGANISATION

The following document is structured in three parts. The first part is the Introductory material divided into Introduction, State of the art, Research Methodologies and Tools, and The problem and its challenges; the second part is the Core of the dissertation divided into Case Study 1 - Platform to validate the dental medicine acts through ADSE, Case Study 2 - Platform to invoice the hospital acts through ADSE, Proof of Concept, and Conclusions and Future Work; the third part is the Appendices divided into Platform to validade the dental medicine acts through ADSE requirements and Platform to invoice the hospital acts through ADSE requirements.

- **Introductory material**
  - Chapter I: Introduction: The main of this chapter is to provide a background and framing for the work, as well as the main motivation, main objectives, and the document structure.
  - Chapter II: State of the art: In this chapter all the theoretical and scientific concepts of interest for this study in ISs are presented and documented, namely Health Information Systems, Interoperability, AIDA and Web Services.
  - Chapter III: Research Methodologies and Tools: The aim of this chapter is to present the Design Science Research investigation methodology for the development of all study cases. In addition, the tools and technologies planned for usage are documented.
  - Chapter IV: The problem and its challenges: In this chapter, a brief presentation about the problems and a proposed approach to solve them are presented.

- **Core of the dissertation**
  - Chapter V: Case Study 1 - Platform to validate the dental medicine acts through ADSE: In chapter V the platform to validate the dental medicine acts through ADSE is introduced. The background and motivations are described, in addition to the platform's main objectives and requirements elicited during the development. Comprehensive documentation of all steps taken during the platform's design and development process is described. The platform's architecture, database models and technologies adopted/used are documented. This chapter concludes with an analysis and discussion of the results obtained, as well as a brief conclusion and an insight into future work.
  - Chapter VI: Case Study 2 - Platform to invoice the hospital acts through ADSE: In chapter VI the platform to invoice the hospital acts through ADSE is presented. The context and motivations are shown, as well as the platform's key objectives and requirements elicited during the development. Comprehensive documentation of all the steps taken during the platform's design and development process is illustrated. The platform's architecture, database models and technologies adopted/used are documented. This chapter ends with an analysis and discussion of the results, as well as a conclusion and an insight into future work.
  - Chapter VII: Proof of Concept: In this chapter, the Proof of Concept made for the developed project is presented. A Strengths Weaknesses Opportunities Threats (SWOT) analysis was performed in both web applications.
  - Chapter VIII: Conclusions and Future Work: The final chapter of this document seeks to summarise and present the main conclusions and contributions made possible by the development of these platforms. Furthermore, future work proposals to improve them and their implementations are discussed.
- **Appendices**
  - Appendix A: Platform to validate the dental medicine acts through ADSE requirements: The functional and non-functional requirements of the platform to validate the dental medicine acts through ADSE are described.
  - Appendix B: Platform to invoice the hospital acts through ADSE requirements: The functional and non-functional requirements of the platform to invoice the hospital acts through ADSE are described.

2

STATE OF THE ART

2.1 INTRODUCTION

This chapter study and document the different discussion topics of this dissertation. To this end, a careful review of various scientific articles was done to understand what other authors concluded, which were their principal challenges and how they get through them.

An introduction to ISs, in particular HISs, which define their key concepts and characteristics, their information workflow, their goals and their structure. It also addresses how they should work together to achieve interoperability and its advantages. Then, various forms of interoperability are clarified and the key problems for the implementation of interoperable HIS are discussed.

In order to address these key problems, an introduction to the AIDA - Agency for Integration, Diffusion and Archiving - is made, where it is shown how it works, its architecture, how it can achieve interoperability in multiple healthcare systems, and its strengths and weaknesses. In addition, because AIDA supports web-based services, it is important to clarify the main concepts of web services, their design and technologies, how they operate and how they can be applied to HISs in order to ensure data sharing between them and promote interoperability in the healthcare context.

2.2 HEALTH INFORMATION SYSTEMS

In the hospital environment, a lot of data are daily generated in different formats and from different data sources. Furthermore, it is common that each institution adopts a distinct IS to storage data. So, the exchange and sharing of clinical information between IS are becoming one of the key ways of enhancing the quality of patient care. However, it is not so easy to reach this main objective. A large number of heterogeneous sources of information, such as medical applications and software, medical equipment and even clinical staff self-knowledge incorporated as needed in the Patient Clinical Record (PCR), is one of the key obstacles [49].

An IS could be described as a technologically implemented medium for the recording, storage, dissemination and drawing of conclusions from linguistic expressions [29]. More precisely, an IS is an automated or manual system that involves the collection, processing, transmission and distribution of data, by individuals, machines and/or organised methods, that represent knowledge to the user [55]. In the healthcare sector, an IS needs to

"handle aspects of information overload, with adverse clinic activities, which results in constant changes in the IS structure and behaviour" [62].

According to MeSH, a HIS is defined as an "integrated computer-assisted system to store, manipulate, and retrieve healthcare administrative and clinical data". This way, the HIS must be accessible according to the needs of administrative staff, healthcare professionals, managers and/or patients. For this reason, the HISs need to communicate and cooperate in order to enhance their overall performance and effectiveness, to improve HIS quality of diagnosis, but mostly, to improve the quality of patient care [9]. Indeed, the collaboration and sharing of data and knowledge are some of the most relevant features, it is the essence for the optimisation of existing resources and the improvement of the decision-making process through information consolidation, verification and dissemination [37]. This increases complexity to HIS, which depends not only on the number of systems but also on the number of providers.

In this way, figure 1 presents the information workflow, and it is easy to understand that HISs combine and use common elements inside a healthcare facility to produce real resources and ensure information to users when and where they need it the most, making decisions more easily from raw inputs [49].

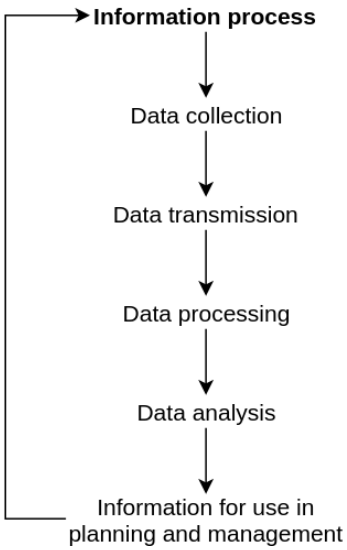


Figure 1: The Information Process in Health Information Systems, adapted from Janovsky and of Strengthening of Health Systems.

As mentioned above, the principal purpose of a HIS is to improve the quality and efficiency of health services. In addition, they can also provide better coordination among medical professionals and facilities, thus reducing the number and incidence of medical errors and, at the same time, they can reduce healthcare costs and may provide a means to improve the management of hospitals [46]. According to Vasconcelos et al., it shall comply with a set of objectives in order to achieve the referred proposes:

- Provide patient information (medical events) to all healthcare units;
- Provide to the patient context medical information with his health profile, as well as information about his clinical state and respective clinical history;

- Develop access, distribution and share mechanisms of health information;
- Increase the performance of administrative processes in healthcare units;
- Standardise patient management services and the management procedures in all healthcare units.

To reach these goals, HISs can use a wide technological variety. According to *Serviço de Bioestatística e Informática Médica da Faculdade de Medicina da Universidade do Porto*, in terms of the user interface, it can be in text mode, graphical or web environment applications. In terms of these systems structure, it depends on the institution size and the system quality. It can be:

- **Standalone**: works only in one computer;
- **Client-server**: the application is installed in various computers, but the database server is located on just one server;
- **in Web environment**: the application is a set of pages that can be accessed from different computers, with the data and the application on the server;
- **Remote access**: using remote-desktops or thin-clients to access environments that are on the server.

The development of healthcare and services is a complex process that involves different professionals with different views, different organisations and physical resources. Therefore emerges the need to create a universal system that attaches all pieces of information shared between services. It is necessary to develop a solid and efficient process of integration and interoperation that must take into consideration scalability, flexibility, portability and security [9]. In this way, one of the needs found by institutions concern in the integration of these ISs with their internal and external partners, in order to guarantee the exchange of information between them [19]. This integration is called system interoperability.

2.3 INTEROPERABILITY

According to Sayão and Marcondes interoperability is "the user possibility to search for heterogeneous informational resources, storage at different network locations, using a single interface and without knowledge need about how the resources are stored". Completing the previous concept, according to Arms interoperability seeks to "develop useful services and solutions to users, stem from informational resources which are technically diverse, and often managed by different institutions".

In the view of Guy and Miller, interoperability is understood as a "continuous process based on ensuring that an organisation's processes, procedures and culture are conducted to optimise the potential for sharing and reuse of knowledge", either internally or externally to organisations. Although the concept of interoperability is broadly comprehensive, according to Techopedia Inc. there are two main types of it:

- **Syntactic Interoperability**: Where two or more systems are able to communicate and exchange data. It allows different software components to cooperate, even if the interface and the programming language are different;

- **Semantic Interoperability**: Where the data exchanged between two or more systems is understandable to each system. The information exchanged should be meaningful, since semantic interoperability requires useful results defined by the users of the systems involved in the exchange.

Healthcare has an intensive activity, as described above, and each department speaks its own language and has its own applications [49]. Each of these applications is responsible for generating data, which through interoperability could result in data knowledge, and for turning healthcare into a science-based on information and reputation [23]. In fact, the increasing number of heterogeneous applications encourages the development of data silos, i.e. a collection of information that is isolated and not accessible by others, which inhibit the easy flow of information within a healthcare organisation.

In addition, patients visit many hospitals during their lives, and each time a new episode is documented when they are accepted. Therefore, patients have several episodes in several health departments and this number grows with time [45]. Data present in past episodes are most frequently lost and fragmentation occurs. Since healthcare is science-based on information, there is an intense need to access previous data. The need to access patient information is growing in parallel with the need to integrate patient information through a healthcare organisation's numerous structures [49].

Due to the current complexity of the medical activity, specified above, it is clear that there is an increasing need to make HISs interoperable. Despite it seems a simple idea, the interoperability is hampered because, briefly, "the medical terminologies and clinical ontologies used in HISs are heterogeneous, (...) leading to the need for information standardisation that is shared" [19]. In other words, healthcare institutions can have different systems that were developed using different languages, different system platforms and Database Management System (DBMS), creating problems with system and language interoperability [70]. This way, according to Zhang and Xu, the interoperability issue can be seen from different perspectives:

- **Database System Interoperability**: Patient records are mostly stored in separate database systems, however, it is not possible to exchange data from different database systems with each other or use it in applications based on different DBMS;

- **Language Interoperability**: Usually, HISs in healthcare institutions can be totally heterogeneous about the programming languages used to build them. This makes the reuse and share of applications between different HISs very hard;

- **System-platform Interoperability**: Different HISs can be developed based on different development platforms, which make those HISs only work on some certain systems platforms;

- **Semantic Interoperability**: Some interoperability problems are caused by semantic differences. Semantic interoperability assumes that the components of the distributed application will have different meanings.

These lacks of integration between the different HISs are not only an obstacle for a more effective clinical practice, but it may lead to insufficient care for the patient [47]. So, it can easily be understood that interoperability in hospital environment among IS brings certain benefits, such as:

- Medical data and functions can be easily shared and exchanged between HISs;
- Systems would be accessible to people using different operating systems;
- New technologies can be plugged into original systems;
- Allows eliminate the use of paper in hospitals;
- Allows increase the quality of services and information;
- Makes patient treatment faster and better;
- Reduces costs;
- Reduces the number of errors;

While interoperability has been studied and its implications for care delivery have been considered, in most healthcare organisations, the degree of interoperability between systems remains frustratingly poor [10]. In healthcare, it is globally recognised that technology can achieve the requisite knowledge sharing across disciplines and venues, but the lack of well-defined and universally accepted standards prevents HIS from being interoperable [49]. The solution is to integrate, diffuse and archive the information under a dynamic framework, in order to share this knowledge with every information system that needs it.

2.4 AIDA - AGENCY FOR INTEGRATION, DIFFUSION AND ARCHIVING

Over the last decade, the Artificial Intelligence Group, in the Informatics Department at the University of Minho, has dedicated their studies to build the Agency for Interoperability, Diffusion and Archive called AIDA. AIDA is an agency that provides intelligent electronic workers, called pro-active agents, in charge of tasks such as communicating with the heterogeneous systems, sending and receiving information, managing and saving the information and answering requests, with the necessary resources to their correct and on-time accomplishment [2].

AIDA includes many different integration capabilities as Service Oriented Architecture (SOA) and Multi-Agent Systems (MAS) to implement interoperability, in accordance with standards, comprising of all service providers within a health institution [36]. This platform also supports web-based services to facilitate direct access to the information and communication facilities set by the humans; the main goals are to integrate, diffuse and archive large sets of information from heterogeneous sources; also provides tools in order to implement the communication with human agents based on web-based services [2].

2.4.1 *AIDA Architecture*

Figure 2 shows the architecture with AIDA as the central element in a healthcare environment, which ensures interoperability and communication between the following systems [9]:

- The **Electronic Medical Record** (EHR), a type of repository with health-related information on an individual, in a format that can be processed by computer, stored and transmitted from a secure and accessible by multiple authorised users;
- The **Administrative Information System** (AIS), which intends to represent, manage and archive the administrative information during the episode (i.e., a collection of all the operation assigned to the patient);
- The **Medical Support Information System** (MIS), which intends to represent, manage and archive the clinical information during the episode;
- The **Nursing Support Information System** (NIS), which intends to represent, manage and archive the nursing information during the episode;
- The **Information Systems** (DIS) of all the **departments** or services, in particular of the laboratories (Labs), **Radiology Information System** (RIS) and Medical Imaging (Picture Archive and Communication System (PACS)), which deals with images in a standard format, the Digital Imaging and Communications in Medicine (DICOM) one.

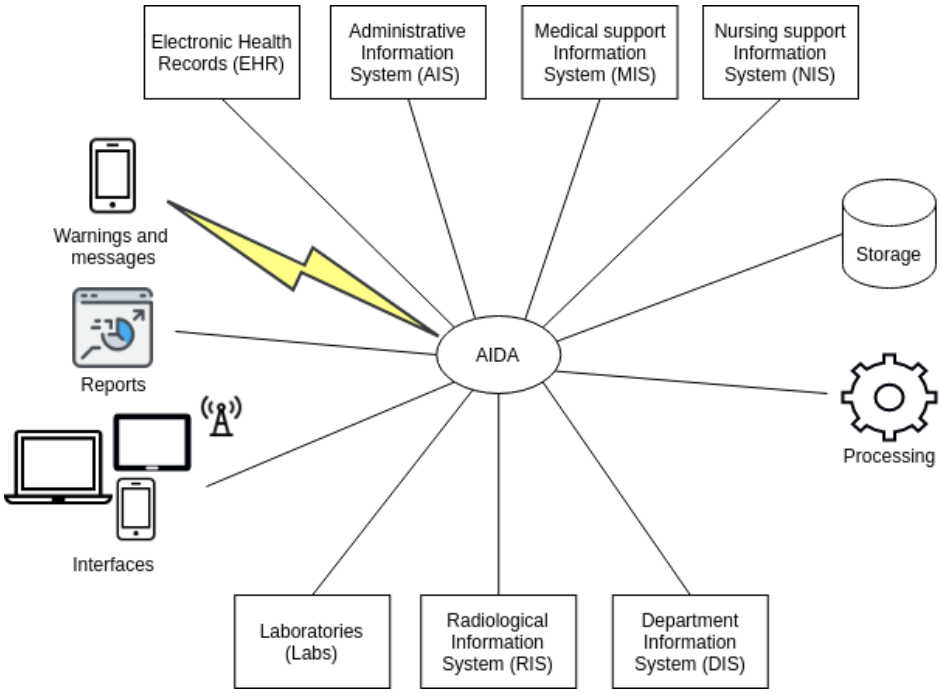


Figure 2: AIDA architecture, adapted from Cardoso et al..

The presented architecture was expected to support medical applications in terms of AIDA and Electronic Health Record (EHR) and has the form of intelligent information processing web systems, its major subsystems, their functional roles, and the flow of information and control among them, with adjustable autonomy.

Healthcare staff acquires this information and its value is automatically stored and distributed to where it is needed. Every document created within specialised service respect these rules, making different and individualised departments closer. The coding and ordering features are very useful to link different data to one specific problem, as coded data is much easier to access and it is recommended for decision support using Artificial Intelligence (AI). The electronic ordering embedded in EHR can be used not only to obtain medical equipment or pharmacological prescriptions but also for acquiring laboratory and imaging studies outside the service where it is used. Furthermore, it may enable the centralisation of exam display, allowing different services to share results concerning the same patient, reducing costs on unnecessary exams, and above all, improving the quality of service being provided [2].

Medical data is sensitive and must be accessed only by authorised personnel, but it must be flexible in order to allow professionals to access it when needed. The messaging system allows creating, sending and receiving messages online and it can be very useful for the treatment of data, images or even to exchange files [48].

2.4.2 *AIDA as a Multi-Agent System*

As mentioned early, the AIDA platform is a pure communication MAS, i.e., there is no external environment influence and the agents only communicate with each other via messages. However, AIDA contains different types of agents [9]:

- The **Proxy Agents** (PAs) that provide the bridges between users and the system in terms of questions that can be explained; decisions may have to be taken and/or visualisation of the results;
- The **Decision Agents** (DAs) provide mediation capacities, acting by accepting a task of PAs. They can break down tasks into sub-tasks, sending them to be processed by the CAs, later integrating the results;
- The **Computing Agents** (CAs) that accept requests of DAs specific tasks, returning the results;
- The **Resource Agents** (RAs) who have all the knowledge needed to access a specific information resource;
- The **Interaction and Explanation Agents** (IEAs), which act on the basis of argumentative processes that are fed with data and/or knowledge from both the PA and the DAs.

2.4.3 *AIDA Database*

In healthcare units, databases have a vital role, since they store very important information about the patients' clinical status, administrative information and other relevant information for the healthcare services. Therefore, it is crucial to ensure the availability, reliability, confidence and safety of the database.

Additionally, it is essential to ensure the integrity and permanent availability of data even in the presence of faults. To achieve these goals, fault tolerance mechanisms based on the data or components redundancy are used. In the AIDA database, there is also another fault tolerance mechanism: a data guard solution. Shortly, this mechanism consists of one or more standby databases (replicas), which should be in different places. In this way, when the master database is unavailable the replica can be used in read-only mode [9].

2.4.4 *AIDA Abilities and Limitations*

Like any platform that aims to achieve interoperability between HISs, AIDA has its strengths and weaknesses. According to Cardoso et al., its main strengths are:

- High availability and full-time support;
- High accessibility;
- Ease of maintenance;
- Ease of use;
- Immediate access to detailed clinical information;
- Customised reports to meet the need required;
- Ability to remotely access the system in a safe way;
- Interoperability.

Despite these features, AIDA can still be improved and has weaknesses like:

- System documentation does not exist;
- Graphical interface slightly confusing;
- Insufficient education and training of health professionals;
- Computers in the healthcare environment are usually old and consequently slow.

AIDA supports web-based services and other features to facilitate direct access to the information and, that's why it has opportunities to grow and become a robust system widely used.

2.5 WEB SERVICES

Web services have emerged as the next generation of integration technology. It allows any piece of software to communicate with each other in a standardised and structured eXtensible Markup Language (XML) messaging framework, based on open standards. Web services, as a new type of software service, are modular self-describing, and self-contained applications that can be published, located and dynamically invoked through the web [70].

2.5.1 *Web Services Architecture*

The architecture for web services, as shown in figure 3, is divided into three areas: communication protocols, service descriptions, and service discovery, each of which is defined by an open standard [13].

Web services are typically comprised of two main technologies (XML and Simple Object Access Protocol (SOAP)) and two assistant technologies (Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI)). **XML** was designed to store and transport data in hierarchically organised documents; **SOAP** is an XML-based protocol for messaging and remote procedure calls, which works on existing transport protocols, such as Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) and MQSeries; **WSDL** is an XML format that describes web services as collections of communication endpoints that can exchange certain messages; **UDDI** offers users a unified and systematic way to find service providers through a centralised registry of services [13].

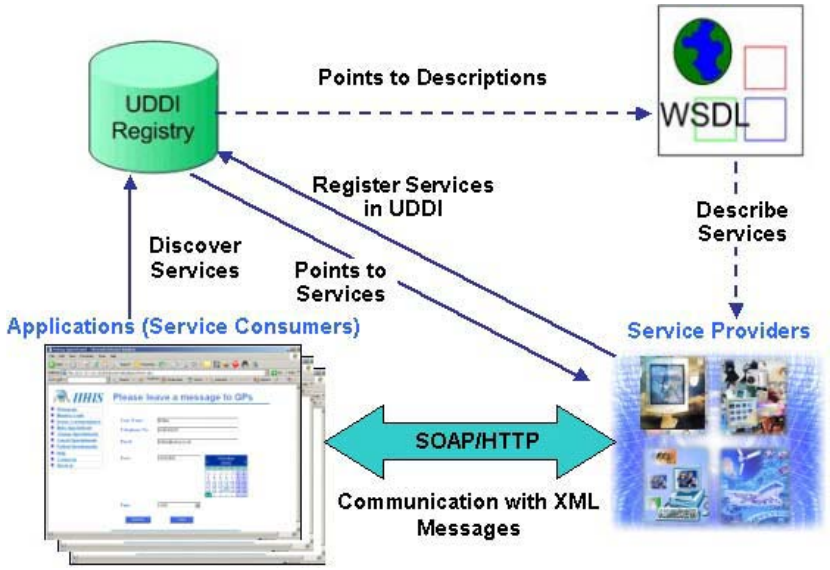


Figure 3: Web service architecture [70].

Figure 3 also indicates how web services work. Firstly, service providers describe their web services using WSDL. Next, service providers will register and publish their services in UDDI. Service consumers or applications find services through UDDI that, according to the description of web services, guide them to the relevant services.

As far as the previous step is concerned, applications or service consumers will invoke the related web services using SOAP, which is transmitted over the Internet using HTTP [70]. More recently, some web services start to use Representational State Transfer (REST) protocol, since it simplifies access to web services. It also uses HTTP protocol and allows the use of several data representation formats, such as JavaScript Object Notation (JSON) (one of the most used), XML, or others.

2.5.2 *How do applications and web services interact?*

SOAP offers a way of interacting between applications that are built in different programming languages and operate on different operating systems. Using open standards and XML encoding, web services directly provide distributed computing technologies to incorporate applications on the Internet [70]. These characteristics are what make web services interoperable.

Figure 4 shows the interaction between service consumers or applications and web services. As mentioned above, applications send requests and responses to and from web services via SOAP. When a program invokes a web service method, the request and all relevant information are bundled in a SOAP message and sent to the proper destination. When this message is received by the web service, the content starts to be processed, which specifies the method the client needs to execute and the arguments the client is transferring to that method. After the web service receives this request and parses it, the proper method is called with the specified arguments, and the response is sent back to the client in another SOAP message. The client parses the response to retrieve the result of the method call [70].

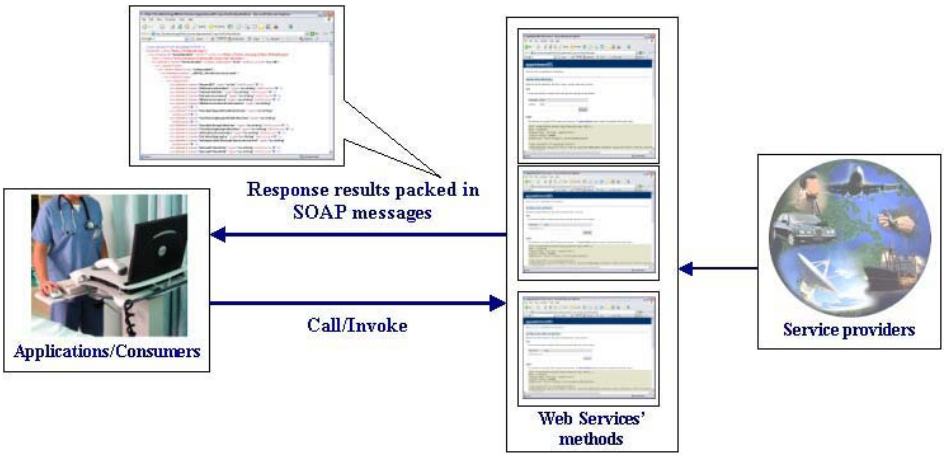


Figure 4: Interaction between applications/consumers and Web services [70].

2.5.3 *Web Services in Healthcare*

As previously mentioned, web services are all about sharing data. In the healthcare environment, a lot of data is daily produced in different HISs and, most of the times, it needs to be exchanged between them. In this way, achieving interoperability between ISs is extremely important in healthcare.

To prove that web services can help in this exchange of data, Zhang and Xu did a case study for system interoperability concern in the healthcare field. In this study, was developed a web service built on Microsoft .Net to give support to existing HISs creating, thus, a new technology called Web Service-Based Integrated Healthcare Information System (WSIHIS). This technology aims to overtake system and language interoperability issue.

Inside WSIHIS, as shown in figure 5, web service plays the function of middleware that hides from users and developers all the differences in system platforms, programming languages, and database systems. Accordingly, users would be able to get access to WSIHIS regardless of their different system platforms. From developers' perspective, they can invoke or reuse applications of WSIHIS in their systems with the support of web services [70].

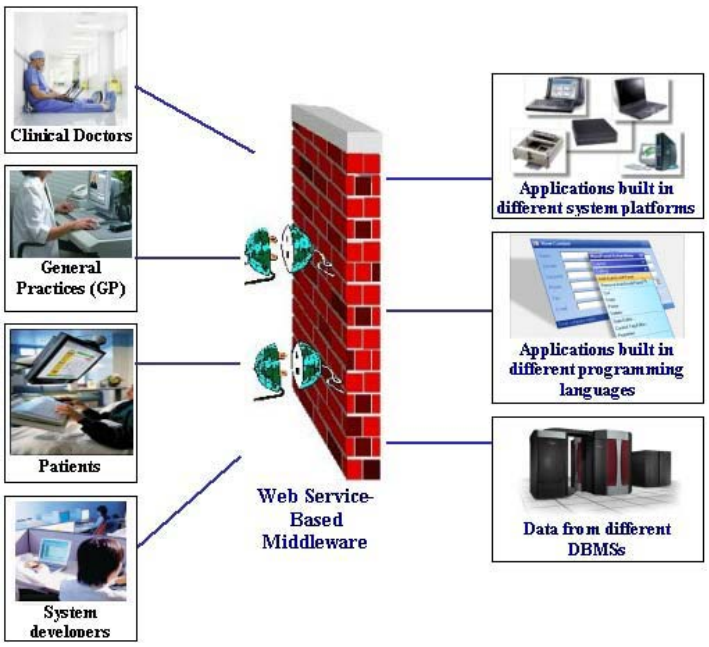


Figure 5: The general concept of Web service-based solution of WSIHIS [70].

Any platform that can format and parse an XML message can expose and consume web services because it uses XML to format requests and responses. This allows XML-based web services to bring together different pieces of functionality internal or external to an institution. When web services accept requests from applications, web services can, according to application specifications, extract data from various DBMSs into data sets. In addition, SOAP would act as an XML envelope to bundle these XML-based data sets into SOAP messages. All the data sets would be written in XML messages. And then these SOAP messages are transmitted via HTTP back to applications. SOAP offers a way to interact between applications built with different programming languages

and running on different operating systems in web service-based middleware. In addition, the Microsoft .Net platform is another significant part of the WSIHIS interoperability solution [70].

The authors concluded, after testing WSIHIS on applications built in different programming languages, system platforms and database systems, that the use of web services and Microsoft .Net technologies in HIS is very promising and offers more advantages for improving the system and language interoperability.

2.6 CONCLUSION

Several topics of interest to the project were presented in this chapter, namely HISs, interoperability, the AIDA ecosystem and web services. A thorough analysis of theoretical concepts and past works is an important process in order to create a new generation of interoperable artefacts.

A theoretical background on HISs was offered, introducing their key concepts and characteristics, their information workflow, their goals and their structure. It was also addressed how they should work together to achieve interoperability and its advantages were presented. Then, various forms of interoperability were clarified and the key problems for the implementation of interoperable HIS were discussed.

In order to address these key problems, an introduction to the Agency for the Integration, Diffusion and Archive (AIDA) was made, where it was shown how it works, its architecture, how interoperability can be reached by it in multiple healthcare systems, and its strengths and weaknesses. In addition, because AIDA supports web-based services, it was important to clarify the main concepts of web services, their design and technologies, how they operate and how they can be applied to HISs in order to ensure data sharing between them and promote interoperability in the healthcare context.

In the next chapter, the Design Science Research (DSR) investigation methodology, as well as the tools and technologies used for developing solutions, are explored and reviewed.

3

RESEARCH METHODOLOGIES AND TOOLS

Any research project requires the adoption of methodologies to define its stages, processes and methods. In this way, it is essential to choose the right methodologies and instruments that are better for creating one solution. Nevertheless, apart from the benefits provided by them, this decision must take into account their drawbacks, as well as ensure project compliance and fulfil security requirements.

With this in mind, the creation of this dissertation project will be based on the methodology of Design Science Research (DSR), which is highly used to build and assess strict and reliable solutions. Thus, several methodologies, technologies and toolkits that are suitable for defining and implementing each portion of the final solution will be introduced and implemented at each stage of the development of this system.

The *Design Science Research* methodology will be presented in the next section, explaining the stages and techniques to making knowledge. After that, the requisite tools and technologies will also be provided for the implementation of the proposed solution. However, during the implementation process, these tools may be modified or new ones can be introduced into the final solution.

3.1 DESIGN SCIENCE RESEARCH

Information design can be defined as "the art and science of preparing information so that it can be used by human beings with efficiency and effectiveness" [26], having as purpose organising, coding and presenting data [57]. This, when framed with Information Science, enables problems that still have no solution to be investigated and solved. In this way, DSR tends to be "a more organised way of constructing artefacts or improvements" [53]. This methodology is used to develop solutions and new approaches in ITs, making knowledge that can be shared and addressed.

Design Science Research can be understood as a form of research that tackles two types of problems: "practical problems" that, in other words, allow a difference in the world based on decision-makers goals and "knowledge problems", that is to say, that demand that our knowledge about the world is changed [64]. This can unify the science capacity of learning "which is" with the design capacity of learning "what can it be" [24]. To understand the relevance of DSR aimed at information systems, it is important to learn first about *Design Science*. This includes an objective paradigm of research that aims to develop innovative artefacts to address many real-world problems [58] and to make new knowledge. This can clearly be understood by figure 6.

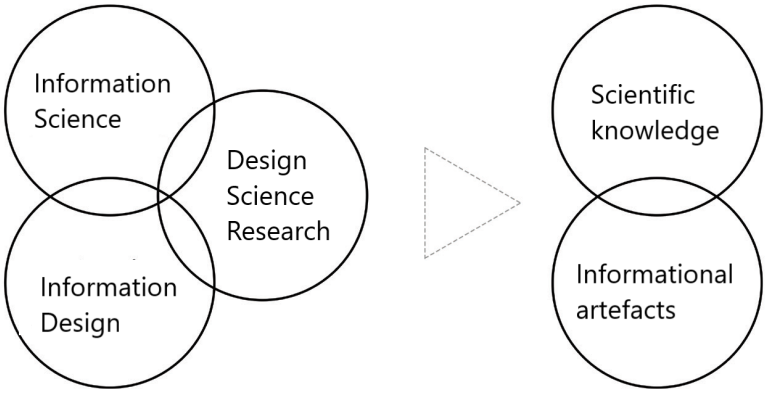


Figure 6: Interdisciplinary interaction model to concept and improvement of informational artefacts with the production of scientific knowledge, adapted from Rodrigues.

In DSR, a practical problem is responsible for guiding research and as of its may arise other practical problems and questions about knowledge. These problems and possible questions cause a real cycle called the "regulative cycle" [64] (figure 7).

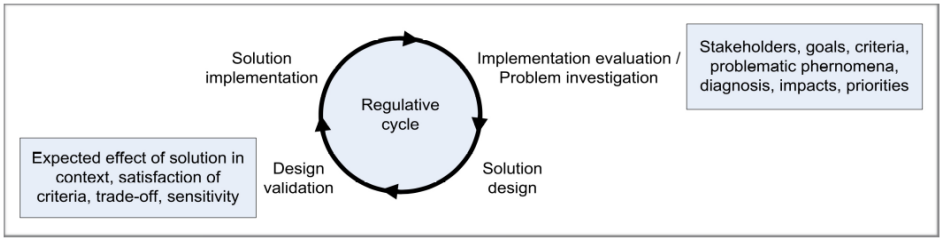


Figure 7: Regulative Cycle [64].

The initial phase of the cycle is the problem investigation, a stage defined as knowledge interrogation. This stage presents a theoretical nature since exists a demand for information to understand the issue, without having the capacity to modify it yet. The next step, the solution design, will analyse a given practical problem. The design validation is the stage where knowledge is built, namely where possible outcomes from a well-heeled project implementation are evaluated by a researcher. The cycle follows with the stage of solution implementation, a stage totally practical. Finally, the stage of implementation evaluation aims to generate scientific knowledge based on the research previously done.

DSR is used for this dissertation project to ensure that the final solutions allow the *Hospital da Santa Casa da Misericórdia de Vila Verde* to fulfil the requirements and needs. In this way, an effective and reasonable approach is reached based on previously explored methodologies and technologies that fit better into the issue at hand, enabling new insights to be explained for the hospital teams as well as for the scientific community.

3.2 PROGRAMMING LANGUAGES AND TOOLS

Various tools and technologies have been used for the implementation of the system. The programming languages and frameworks adopted for the development of the proposed platform will be presented in this section. In overview, the frameworks and programming languages used were: *Node.js*, an open-source JavaScript interpreter used to build a web server; *React*, a JavaScript library used to build user interfaces; *Oracle*, a DBMS, where all patient and hospital environment information is kept; *GraphQL*, a query and manipulation language for web and mobile applications.

For platforms development, Node.js at version 12.16.2 was adopted, along with Express at version 4.17.1. GraphQL version at 15.3.0 was used to define the application data type GraphQL, Apollo Server at version 2.18.2 to processes GraphQL operations from application clients, and Apollo Client at version 2.6.10 to execute these queries. To gather and manipulate the data needed was used the hospital's databases, Oracle databases at version 12.2.0.1.0.

3.2.1 *Node.js and Express*

Node.js is an asynchronous JavaScript interpreter that operates on the server-side and enables real-time and high-scalability network applications to be easily and quickly created [50]. It is not a web server by itself but it has a built-in HTTP server library. This way, compared to conventional web servers, it offers a different paradigm since that it provides to the programmer a framework called Express, used to build an application, which is almost a server [7].

As a single-threaded server, Node.js web servers are fast and use system resources efficiently, serving more clients with a small number of resources [25]. A Node.js app's standard library provides a collection of asynchronous I/O primitives that prevent JavaScript code from blocking. Node.js functions are non-blocking, allowing different commands to run in a non-sequential way. The libraries provided by Node.js are commonly written using non-blocking paradigms [44].

This set of features enables Node.js to handle several concurrent connections with a single server without enforcing the burden of thread concurrency management, which can be a major source of bugs [44]. Since Node.js and the most front-end technologies, p.e. React, are written in JavaScript, front-end developers can write both server-side and client-side code without having to learn a new programming language.

Express.js is defined as a lightweight and flexible web applications framework for Node.js that provides a broad range of features for web and mobile applications [43]. Express builds a web server that receives requests and gives replies offering:

- An interface to link an incoming Uniform Resource Locator (URL) to a certain piece of code (read from the database, for example);
- HyperText Markup Language (HTML) responses for server-side rendering;
- Session support for users identification.

Thus, Express.js abstracts a lot of complexity and the repetition of multiple tasks, allowing web applications to be more rapid, more efficient and maintained [25].

3.2.2 *React library*

React is a declarative, efficient, and flexible JavaScript library for building the User Interface (UI) or UI components, released by Facebook in 2013. This library, through the rendering of components, i.e. small pieces of isolated code as browser elements, allows the development of complex and interactive single-page applications [15].

There are two types of components in React, stateful components and stateless components. The state represents data in which the component's property values are stored, that can be changed, making the component re-rendering. These changes could occur for a variety of reasons, including database upgrades, user modifications, and system-generated events, among others [15].

A stateful component, defined as a *class*, can keep track of changing data by maintaining internal state data. A separate render method for displaying JSX on the screen is also included. The stateless component, defined as a *function*, doesn't have its own state. It simply prints what is passed to it through "props" (similar to function parameters), or makes the same thing every time. A React-based web application typically contains components that need to receive data to work properly. "Props", a React built-in object that stores data are used to transfer data from one component to another.

As shown in figure 8, every component can be found in one of three states: mounting, updating or unmounting, which can be thought of as the component's birth, growth and death, respectively [17].

- **render():** Handles the rendering of a component to the UI and it is the only method needed in a class component. The *render()* function should be pure with no side-effects, i.e., it should not change the component state and should always return the same output when the same input is passed.
- **componentDidMount():** This method is called as soon as the component is mounted and ready, and is commonly used to trigger data loading from a remote endpoint.
- **componentDidUpdate():** It is called immediately after an update has taken place. The most common use case is updating Document Object Model (DOM) in response to prop or state changes.
- **componentWillUnmount():** Before a component is unmounted and destroyed, this method is called. A component instance can never be mounted again once it has been unmounted. If any cleanup actions are needed, such as cancelling Application Programming Interfaces (API) calls or clearing any storage caches, they should be performed here.

DOM is a programming interface that treats an XML or HTML document as a tree structure, with each node is an object representing a part of the document. Web pages objects are defined in a DOM document, which is manipulated whenever the object's state changes. React makes the DOM tree-like structure of all the components

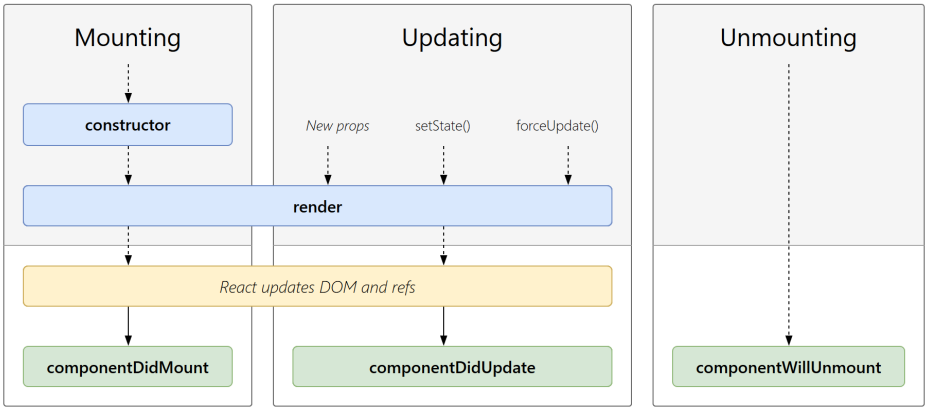


Figure 8: React Lifecycle Methods [31].

internally called the Virtual DOM (VDOM) [11]. It is a copy of the actual DOM and is kept in the browser memory in the form of a JavaScript object. When the state of an object changes, the VDOM only changes that object in the real DOM instead of updating all objects [18].

Therefore, for React, the view layer is like a hierarchy of components. In order to construct applications more consistently, this division into integrated parts is therefore necessary, offering two advantages: [65]:

- Allows individually learning of each piece, namely, there is no need to grasp everything at once;
- All parts are replaceable.

React owns a collection of packages that could be useful in applications development. The React Router, which provides a collection of navigation components that allows declarative routing, is an example. With this package, it's possible to route a single-page application to browser URLs, acting similarly to standard web pages.

Although it only manages the view layer, a complete, versatile and interchangeable structure is created by the React ecosystem [67].

3.2.3 *GraphQL*

GraphQL is a data query language for APIs and an execution environment to handle these queries with the available data [61]. This allows query request to get specific data, allowing the client to have more control about which information will be sent. In addition, it is possible to collect data from various sources with a single request.

This query language allows reading (queries), writing (mutations) and continuous reading (subscriptions) operations. Each operation is just a *string* that must comply with the GraphQL specification [66].

Since a GraphQL operation reaches the back-end of the application, it can be interpreted by the GraphQL schema and resolved by the resolver with data that will be sent to the front-end [66]. In brief, a schema defines

all the available data for clients querying, while the resolver is responsible for generating an answer for each query [52].

GraphQL is thus a feasible alternative to RESTful architectures and offers some benefits [66]:

- Data selection is done through a declarative approach;
- Over-fetching doesn't occur;
- Schema is the only source of truth in GraphQL applications;
- Strongly typed language;
- No versioning needed.

Although these features provide flexibility and performance benefits relative to REST architectures, both are effective in defining how the API will operate and how applications will access data from it [14].

3.2.4 *Apollo platform*

As mentioned in the previous section, GraphQL is a data query language implemented in JavaScript, and, for this reason, it is important to use the Apollo platform to grow its ecosystem and make GraphQL accessible to a larger audience. It comprehends the client-side as the server-side because it offers a huge library ecosystem for both of them [66].

A service that processes GraphQL operations from application clients is required by GraphQL's data graph. This service communicates with back-end data sources to fetch and modify data as needed. Apollo Server can be used to create this service. In this way, Apollo helps to construct, execute queries and manage a data graph, which is a single data layer that enables applications to communicate with data from any combination of linked data stores and external APIs [33].

Apollo Server is an extensible, open-source JavaScript GraphQL server. It can define:

- A GraphQL schema that specifies all of the types and fields available in a graph;
- A set of resolvers that determine how data from back-end data sources should populate each field of a schema.

To execute queries from application clients to the data graph can be used Apollo Client. Apollo Client is a "customisable, open-source JavaScript GraphQL client with powerful caching and state management features" [33]. It allows to define queries directly within the UI components that use them, and updates those components automatically when query results arrive or change. The cache of the Apollo Client replicates the sections of the data graph that the client requires locally. This allows the client to query itself for data that is already available, resulting in significant performance gains while avoiding unnecessary network requests [33].

By analysing figure 9, it is possible to check that there is an Apollo data graph, which uses GraphQL, between clients and back-end providers, facilitating the data flow between them [33].

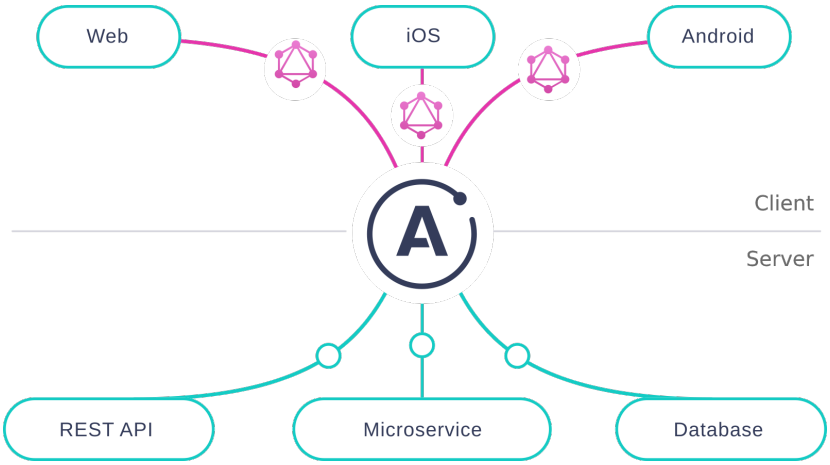


Figure 9: Example of data flow in an Apollo platform [33].

3.2.5 *Oracle Database*

Oracle is a relational DBMS typically used to process online transactions, data warehousing and mixed. Oracle databases are available through several service providers on-prem, on-cloud or as hybrid cloud installation [68].

A relational Database Management System (DBMS) is based on the relational model for data proposed by E. F. Codd, where all knowledge is logically structured inside relationships (tables). There are a name and a set of data attributes (columns) in any relationship. Each tuple (row) has a value as well as a unique identifier per attribute. Relationships allow tuples of two or more relationships to be connected [12].

Through Structured Query Language (SQL) programs and users can access existing data in Oracle databases. This one allows to create databases, perform data management tasks and execute queries. Despite SQL be a single language, it can be divided in two principal types: Data Definition Language (DDL) and Data Manipulation Language (DML). DDL defines a database structure and control the data access to it (CREATE, ALTER, DROP) and DML enables direct data interaction (SELECT, INSERT, UPDATE, DELETE) [12, 59].

3.3 PROOF OF CONCEPT METHODOLOGY

A Proof of Concept (PoC) is a theoretical demonstration of a product, process or concept to determine whether an idea can be turned into a reality, i.e. test whether an idea is viable and explore the idea's potential to be developed or built [51]. A methodology is made up of practical models that can prove or validate a concept through analysis and development. Proof of concept research "presents a discovery about our knowledge of the world and the structure of existence" [28].

A PoC is considered a crucial method in the process of planning, creating, implementing and proposing ITs software solutions. They verify whether a project meets its key characteristics and established goals, as well as identify possible defects or errors in the solution being created. Furthermore, the notion of evidence in science seeks to address a question whose answer is generally applicable in areas beyond those tested for [28].

In this dissertation project, the viability and functionality discussion of the developed tools was corroborated by the application of a PoC methodology. As such, two SWOT studies were performed, one for each developed platform.

The Proof of Concept (PoC) of this work, and all steps taken into account towards its creation, is described in chapter 7 of this document.

3.4 SWOT ANALYSIS

The Strengths Weaknesses Opportunities Threats (SWOT) analysis planning technique is used to develop strong strategic plans by analysing the strengths and weaknesses, in addition to opportunities and threats, of a business or project plan [35]. A SWOT analysis brings a fact-based and data-driven of positive or negative internal and external influences, helping to reveal new ways to leverage and recognise potential vulnerabilities [20, 69].

SWOT analysis groups key data into two main categories: internal and external. Internal factors are an organisation's current strengths and weaknesses, while external factors are the opportunities and threats provided by the environment, outside the organisation's control [69]. SWOTs can be used as inputs for the creation of strong, well-calculated strategies [22].

In this context of applying a SWOT analysis to the developed platforms, each study can thus be based on four main components:

- Strengths: internal attributes of the platforms that are helpful for their successful development and integration;
- Weaknesses: internal attributes of the platforms that are harmful for their successful development and integration;
- Opportunities: external attributes of the platforms that are helpful for their successful development and integration;
- Threats: external attributes of the platforms that are harmful for their successful development and integration.

A carefully considered SWOT analysis generate useful information that helps in decision-making. As the possible effects of this information on the organisation are thoroughly analysed, concrete analysis occurs [22].

The SWOT matrix was represented by Grant in figure 10, where each characteristic is made up in one quadrant, providing a quick overview of a project's position.

The SWOT analysis technique will be performed on the developed platforms in subsection 7.2.1 and subsection 7.2.2.

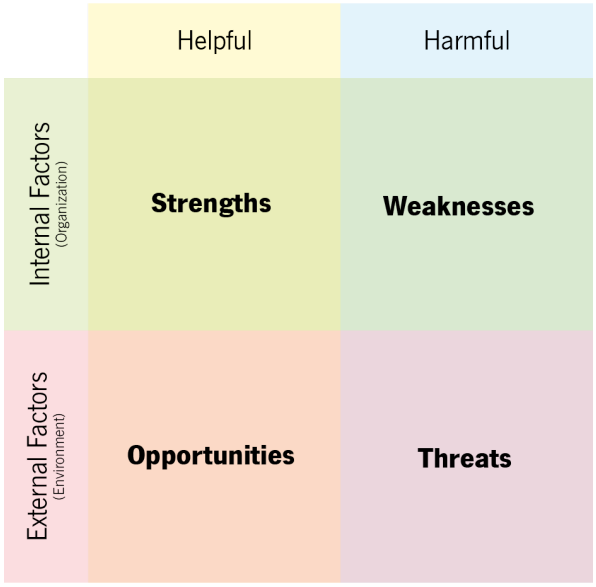


Figure 10: SWOT Matrix [20].

3.5 CONCLUSION

This chapter has presented the Design Science Research (DSR) methodology, all tools and technologies that were adopted, as well as the Proof of Concept (PoC) methodology and the SWOT analysis technique, ensuring that the solution to the problem accomplishes all needs and requirements of the professionals, while elucidating new knowledge both for the institution and the scientific community.

For the developed platforms, *React* was used to implement the platforms' front-end, while *Node.js*, *Oracle*, *GraphQL* and *Apollo* were used to store and query all relevant data.

In the next chapter, the issues of each case study in this dissertation project will be seen, as well as their respective challenges and solutions.

4

THE PROBLEM AND ITS CHALLENGES

As mentioned above, in the hospital environment, a lot of data are daily generated in different formats and from different data sources. So, the exchange and sharing of clinical information are becoming one of the key ways of enhancing the quality of patient care. However, it is not so easy to reach this objective. A large number of heterogeneous sources of information, such as medical applications and software, medical equipment and even clinical staff self-knowledge, is one of the key obstacles. In this way, the issues and challenges of each case study, as well as the possible solutions, will be discussed in this chapter.

4.1 CASE STUDY 1 - PLATFORM TO VALIDATE THE DENTAL MEDICINE ACTS THROUGH ADSE

4.1.1 *The problem*

In the *Hospital da Santa Casa da Misericórdia de Vila Verde* there is a gap in dental medicine speciality, where it is impossible to verify the patient situation about ADSE. ADSE is Portuguese health insurance for public servants. The dentist has no way of knowing whether a patient is covered by ADSE or whether he has ADSE rights to perform any acts of dental medicine. This can bring some problems to the hospital management and the patients since the dentist can do some acts even if the patient has no ADSE rights. This creates an unpleasant situation for the administrative and accounting work of the hospital since ADSE will refuse to reimburse and they must find a solution that suits the patient and ADSE. Quite often, the solution is the hospital bears the burden.

4.1.2 *Proposed approach*

In order to overpass this challenge, it is necessary to build a platform to validate the dental medicine acts through ADSE. It is important that this platform has a user-friendly interface to facilitate the adaptation of the dentists.

The first information that the dentist should see is the relevant hospital data about the patient and if the patient has ADSE rights. Thus, the platform must communicate with the ADSE web service. If the patient has them, the platform must provide a form with the necessary information for invoicing acts through ADSE. After the dentist fills in the details of all the acts that the patient wants or needs to perform, the platform will verify if they can be

performed by the patient via ADSE. If he can't, the platform should present the errors sent by the ADSE web service.

4.2 CASE STUDY 2 - PLATFORM TO INVOICE THE HOSPITAL ACTS THROUGH ADSE

4.2.1 *The problem*

The second case of study of this dissertation aims to solve another problem in the hospital with ADSE insurance. As occurred previously in the dentistry speciality, the reception of patients with ADSE benefits for appointments causes some difficulties to the hospital accounting, due to ADSE restrictions and the accounting workflow. Because the hospital's accounting team only invoices all the medical acts on the following day, when patients don't comply with these restrictions, this can create a real problem for the hospital, since the ADSE services may refuse these outgoing invoices and so to reimburse this acts. Yet again, this can imply a lot of costs for the hospital.

4.2.2 *Proposed approach*

In order to solve this problem, it is necessary to build a platform to invoice the hospital acts through ADSE. This platform would be implemented in the hospital's reception area, so it must have a user-friendly interface.

At first, this application must handle the patient admission to the hospital. To this end, if it is possible, the platform should reed the patient's citizen card to verify its data and whether it is valid. After that, it must verify whether the patient has ADSE rights. To do this, it will be necessary that the platform communicates with the ADSE web service, to check the ADSE rights, and a card reader to collect and verify the patient information. If all data is correct, then the patient admission is complete.

After that, the reception team can invoice the respective acts. For that, the application should produce the medical appointments invoices, stores them and all the relevant information in the hospital's database, and sends them to the ADSE web service for invoicing. In this way, it will be necessary that the platform communicates with AIDA and ADSE web services in order to obtain and to invoice, respectively, some essential data about the patient and its acts. Finally, all of this information needs to be saved in the hospital's database. If something goes wrong in the invoicing process, the errors must be shown to the reception team.

Part II

CORE OF THE DISSERTATION

5

CASE STUDY 1 - PLATFORM TO VALIDATE THE DENTAL MEDICINE ACTS THROUGH ADSE

5.1 INTRODUCTION

In this chapter, the platform to validate the dental medicine acts through ADSE is introduced and discussed. This platform is composed of a web application supporting the administrative and accounting work of *Hospital da Santa Casa da Misericórdia de Vila Verde*. Key features include a patient profile with all important information for dentistry acts, an efficient menu where dentists may select the acts, teeth and other important information that they need to test and a confirmation window with the ADSE response to that test.

In the following sections, the problem and its motivations are described, as well as the platform's main objectives and requirements elicited during its development. Also, all steps taken during the platform's design and development process are described. Thus, the platform's architecture, database models used, and the usage of Node.js, React and GraphQL are analysed and documented. This chapter ends with a discussion of the results obtained, as well as a brief conclusion and an insight into future work.

5.2 CONTEXT AND MOTIVATION

The *Hospital da Santa Casa da Misericórdia de Vila Verde* was born from the need to provide healthcare access to the population of Vila Verde and its surroundings. The hospital currently provides more than thirty specialities, including paediatrics, pulmonology, cardiology, dental medicine, general surgeon, and others. As a result, due to its high evolution and services provided, the hospital has formed a growing number of agreements with public and private entities, insurance providers, health systems and others.

To make all of this work, the hospital has an integrated system that leads with patient records, hospital information and with administrative work. However, it is possible to find some issues in certain areas. It is the case of patients with ADSE, Portuguese health insurance for public servants, that has some restrictions on the number of medical appointments, and on the number of acts that a patient could do on the same day.

In the *Hospital da Santa Casa da Misericórdia de Vila Verde* there is a gap in dental medicine speciality, where the dentists can't verify if an ADSE patient that pretends to perform some dental medicine acts has the rights to

do it or not. This could bring some problems to the hospital management and to the patient since ADSE could refuse to reimburse for not allowed acts performed by the patient.

This master's dissertation was thus proposed, consisting of the design, development, and exploration of a new platform for validating dentist's acts through ADSE. This way, it is possible to dentists verify if some patient has ADSE rights to do some procedure before actually do it. Consequently, this will decrease administrative problems to the hospital that may arise from a non-validation with ADSE.

Thus, in the first case study of the development of this dissertation project, a new platform for the dentist's acts through ADSE was designed and developed, comprised of a web application that contacts with ADSE and AIDA web services. The decisions of all components and features developed in the platform were based on meetings with the administrative and accountancy teams of the *Hospital da Santa Casa da Misericórdia de Vila Verde* in order to bring about the main objectives to fulfil the development of the solution, specifically the necessary business and technical requirements.

5.3 CASE STUDY OBJECTIVES

The platform to validate the dental medicine acts through ADSE was designed to help the management of the dentist's acts of the *Hospital da Santa Casa da Misericórdia de Vila Verde*. In particular, the design aids to be simple to make easily to the dentists to test the patient ADSE rights.

This case study aims to design, develop, and implement a new web platform that interacts with ADSE and AIDA web services, in order to get the answer for the validation of the dental medicine acts of some patient. Then, it is possible to have quick access to basic patient information, simplify and optimise the workflow validation process through ADSE, improve day-to-day administrative procedures and reduce unwanted costs to the hospital.

Thus, the main objectives to fulfil with the development of the platform are the following:

- Researching requirements of the design and development of the platform thought meetings with administrative and accountancy teams of the hospital;
- Usage and development of database models that store all relevant information such as teeth, acts, users and patient's information;
- Usage of APIs from the hospital and web services from AIDA and ADSE to give support to the web application;
- Development of a web app using React library, including multiple components and features;
- Implementation and deployment of this platform in the *Hospital da Santa Casa da Misericórdia de Vila Verde*;
- Ensuring the proposed platform is called by the existing hospital's software and can be used.

5.4 PLATFORM REQUIREMENTS

In this section, the platform requirements will be presented. In order to make the scope of this application more perceptible, a use case diagram was drawn (figure 11). This diagram explains which actors in the system, as well as which functionalities the system should have.

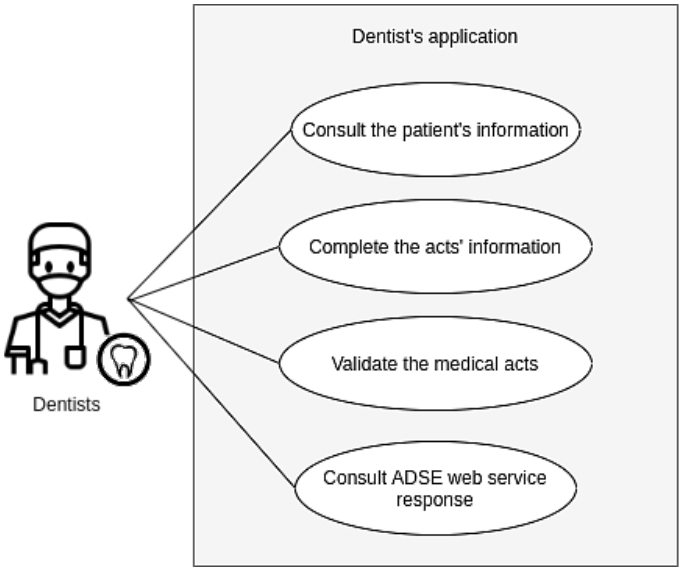


Figure 11: Dentist's application use case diagram.

Use cases

- **Consult the patient's information:** A dentist can consult the patient's hospital data useful to ADSE validation.
- **Complete the acts' information:** A dentist can complete the required information about the medical acts to make a validation request through the ADSE web service.
- **Validate the medical acts:** A dentist can send the acts' information to the ADSE web service in order to get a validation response.
- **Consult ADSE web service response:** A dentist should be able to consult the response from the ADSE web service.

Based on these use cases, the various functional and non-functional requirements of the web application were identified. These are presented in the Appendix section A.

5.5 BUSINESS REQUIREMENTS

Multiple meetings with the administrative and accountancy teams of the hospital were scheduled, to discuss the main objectives and requirements of the platform. The hospital's main problems and challenges for dental medicine speciality were identified, along with how the proposed platform can help them to save funds and solve problems caused by a non-verification of the dentist acts. The platform's key components, features, and constraints, including their technical feasibility and goals, were analysed and discussed.

Several main requirements for the system components include:

- The application should only be accessed by the dentists at the *Hospital da Santa Casa da Misericórdia de Vila Verde*;
- The application must be compatible and responsive with all commonly used web browsers;
- The application main's components are the patient's information with its important data, the acts and teeth selection for validation, and the confirmation with the ADSE response;
- The application should have an intuitive user interface, adopting a simple and clear design;
- The application must allow for future integration and expansions.

In particular:

- **Patient's information scene**
  - This component must show clearly and simply the patient's health and personal information;
  - Verifies if a patient has ADSE rights, through a request to ADSE web service.
- **Acts selection scene**
  - Allows to fill all fields with need information, teeth and acts to perform an ADSE verification;
  - Allows to choose the number of acts;
  - Perform a request to the ADSE web service to verify if a patient can do these acts.
- **Confirmation scene**
  - This component must show intuitively the ADSE response or potential errors.

5.6 PLATFORM'S ARCHITECTURE AND DEVELOPMENT

Multiple tools and technologies were researched to successfully build the system and the most suitable ones were selected to create the solution. The chosen technologies are:

- **Node.js**, an open-source, cross-application JavaScript run-time environment used to create the web server along with the **Express** framework;
- **Oracle** is the DBMS, storing all the information about the hospital, users and patients;
- **React.js** is a JavaScript library used to build the web application's interface;
- **Apollo** is a fully-featured caching GraphQL client with integration for React and others to build UI components that fetch data via GraphQL;
- **GraphQL** is a query language for APIs, and a server-side runtime for executing queries and mutations.

In the following subsections, the platform's architecture is presented, as well as the databases tables created and manipulated, how the web services are called, and the application interface is shown.

5.6.1 *Platform's Architecture*

There is only one sort of user in the application, dentists, so they can access all the functionality of it. Dentists can see all the needed patient's information and whether he has ADSE rights. If the patient has them, dentists may test if they can do some acts by filling all the needed fields. After that, they can check the ADSE validation for that patient.

This platform was developed with the React library. This library allows for the construction of components that manage their own state, then compose them to create complex interactive user interfaces [15]. To communicate to the back-end, which uses GraphQL, both application and server have Apollo, a platform that helps to build, query, and manage a data graph [33]. The back-end was developed using Node.js together with the Express framework.

There is no authentication needed for this application since it is called by the existing hospital's software where dentists are already logged in. All the information regarding all the users, the patients and the dental medicine acts are stored on the Oracle databases of the hospital. The application uses **XMLHttpRequest** object, **Fetch API** and **node-libcurl** library in order to connect with AIDA and ADSE web services.

Figure 12 represents the platform's architecture and its interactions.

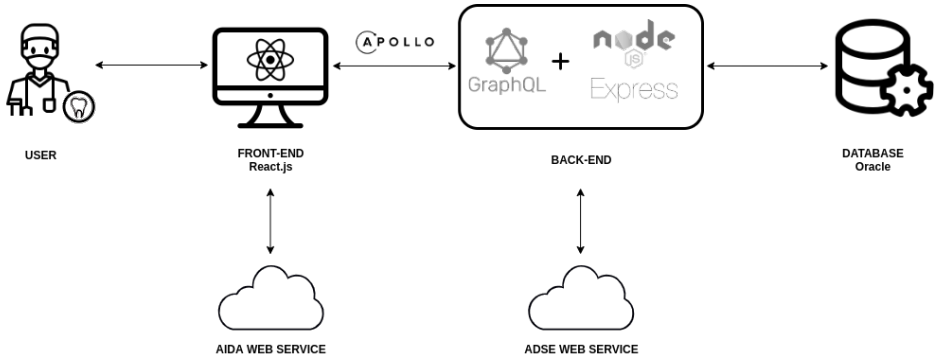


Figure 12: Architecture and tools of the dentist's platform.

The developed platform is currently deployed and installed on a Windows machine (Windows 10 Pro 64-bit) at the *Hospital da Santa Casa da Misericórdia de Vila Verde*.

5.6.2 *Databases Models*

This platform requires access to the hospital's databases to gather information about users, patients, acts and teeth. The Oracle Database Management System (DBMS) stores all tables and relationships, while GraphQL and **oracledb** library are used to connect, to specify the data and to interact with the databases.

Figure 13 shows what tables have been used from the hospital's databases or created to give support to the platform:

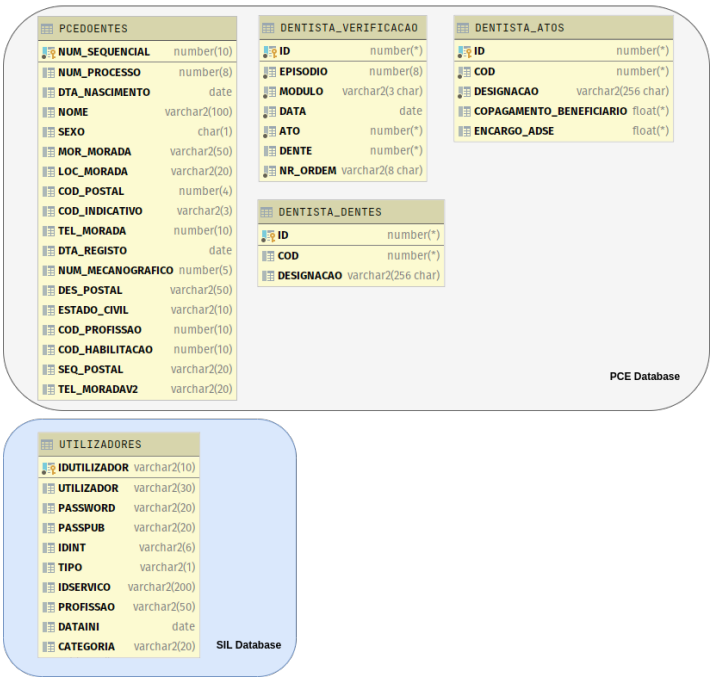


Figure 13: Databases tables used on the dentist's platform.

5.6.3 *Web Services*

To contact the used platform's web services multiple libraries were used, providing a set of features.

For ADSE web service, where a user can send XML requests with patient and hospital information, is used **node-libcurl**. It is described as "a free and easy-to-use client-side URL transfer" [30]. This library allows making asynchronous Curl requests for web services with New Technology LAN Manager (NTLM) authentication without having problems with Cross-Origin Resource Sharing (CORS), as needed for ADSE web service. To do that, it was necessary to build a server route to receive patient and doctor details from the front-end. After that, the server creates the request with all data and headers and sends a Curl POST to the URL of the ADSE web service.

The server finally receives the response and sends it to the client. This interaction can be easily understood by analysing figure 14, an example of the validation process through ADSE.

For the AIDA web service, the application makes the request directly from the front-end. To do that, the **XMLHTTPRequest** object and the **Fetch API** were used. XMLHTTPRequest interacts with servers and "retrieve data from a URL without having to do a full page refresh" [40]. To do so, the constructor's open method is used to create a synchronous GET request to the AIDA web service. A synchronous request was chosen since a request from the React function reader is needed, and it does not make render a promise within it. AIDA will then submit the answer, which will be saved in the front-end. For all other requests, the Fetch API was chosen because it "provides a more powerful and flexible feature set" [41].

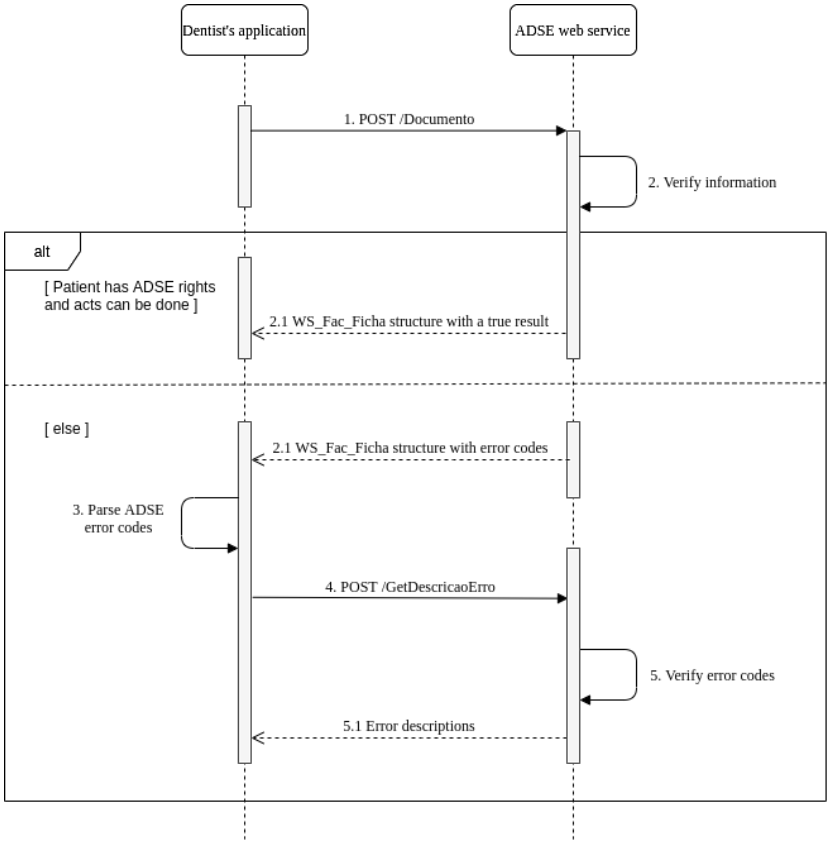


Figure 14: Sequence diagram of the interaction between the platform and the ADSE web service.

5.6.4 *Web Application*

The web application will support the administrative work of the hospital for dental medicine acts through ADSE. As previously mentioned, this application helps dentists to find out if a patient has ADSE rights and whether he should do any dental medicine act.

The React JavaScript library was used to build a Single Page Application (SPA) that loads a single HTML page and dynamically updates the page as the user interacts with the app [63]. By adopting modern technologies and

frameworks, additional features can easily be implemented. Create React App, which is an officially supported to create React SPA and provides a modern configuration-free build setup, was used to build these application [16]. Because this application is a SPA, it was appropriate to use React Router for declarative routing, XMLHttpRequest and Fetch API for web service and server requests, and Apollo Client and other libraries to easily build its UI components and to fetching and parsing data via GraphQL.

This platform is comprised of three scenes, namely the patient's information scene, the acts selection scene and the confirmation scene, which have their own components.

Patient's Information Scene

The Patient's Information Scene presents the patient's hospital data useful to ADSE validation, as shown in figure 15.

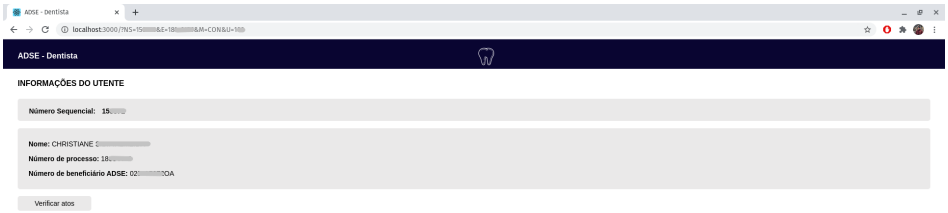


Figure 15: Interface of the Patient's Information Scene of the dentist's application.

The application receives, to load the patient's information, a GET request with its sequential number, the episode and the module and the dentist's number on the Portuguese Medical Association from the existing hospital's program. The first one is then loaded into the search bar component, and the application automatically gets the information about the patient through the hospital's database. To obtain the patient's ADSE beneficiary number the application does a request to the AIDA web service with the patient's process number. If the patient has ADSE rights, then the dentist can move forward to the Acts Selection Scene. However, if this not happen, the application shows a no ADSE rights error, as in figure 16.

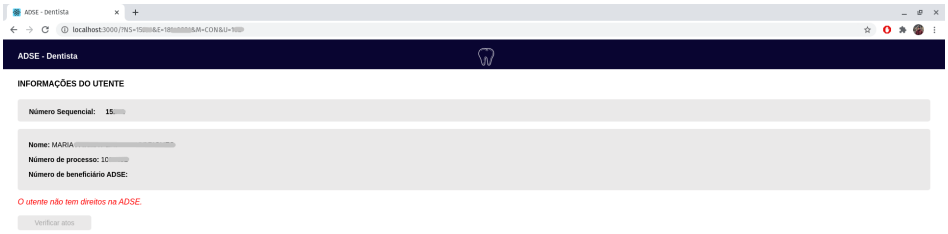


Figure 16: Interface of the Patient's Information Scene of the dentist's application for patients without ADSE rights.

Acts Selection Scene

In this scene, the dentist can fill the needed fields to make a validation through the ADSE web service, as shown in figure 17. Some fields are already filled by the application with the default values for the major of procedures, but the dentist can change them if he needs them.

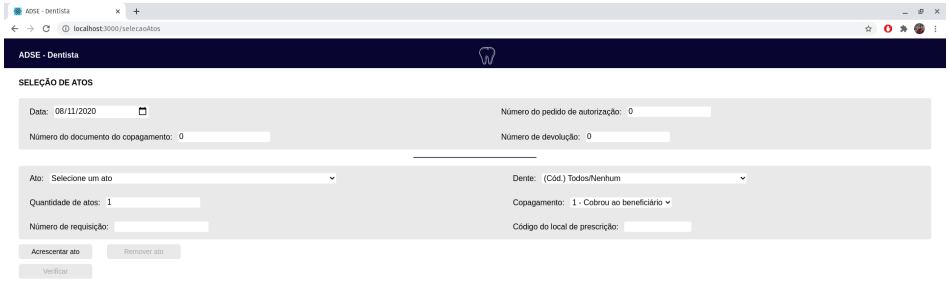


Figure 17: Interface of the Acts Selection Scene.

Initially, the doctor has to fill the general information about the set of acts as the date (*data*), the co-payment document number (*número do documento do copagamento*), the authorisation request number (*número do pedido de autorização*) and the return number (*número de devolução*). The dentist may also add more acts or remove them.

Thereafter, the dentist needs to complete the information about every single procedure, as the act's code (*ato*), the tooth's code (*dente*), the acts quantity (*quantidade de atos*), the way of co-payment (*copagamento*), the requisition number (*número de requisição*) and the prescription place's code (*código do local de prescrição*). These last two fields may not be needed to fill. Figure 18 is an example of Acts Selection Scene filled with two different dental medical procedures.

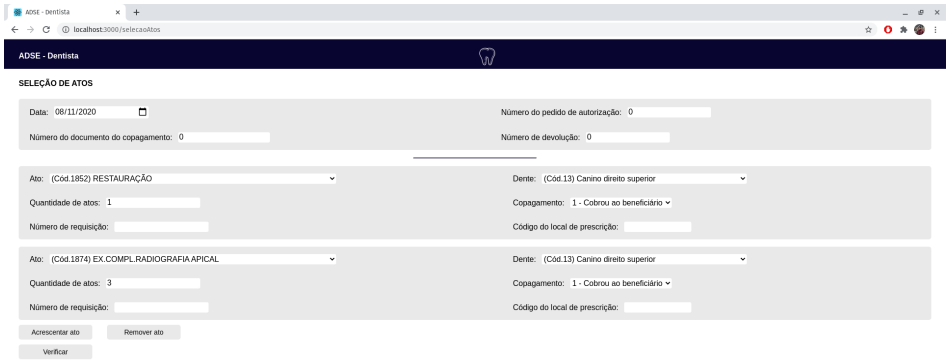


Figure 18: Interface of an example of Acts Selection Scene filled.

Confirmation Scene

After the doctor presses the "*Verificar*" button, all of the acts' information is sent to the ADSE web service, where it is processed and a response is returned. If everything is correct and the acts can be done with the

ADSE co-payment, then this information is saved in the hospital's database. Figure 19 shows, for the previous example, that all the acts can be done by the patient with ADSE co-payment.

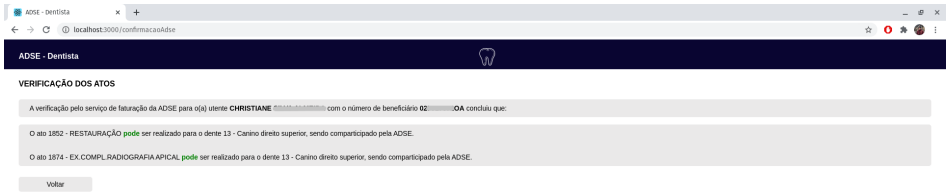


Figure 19: Interface of the Confirmation Scene without ADSE errors.

If the dentist fills the fields incorrectly for some reason or if he chooses incompatible dental medicine acts, the ADSE response will come with errors, as in figure 20. The dentist will then examine them and attempt to correct them, by backing to the Acts Selection Scene.

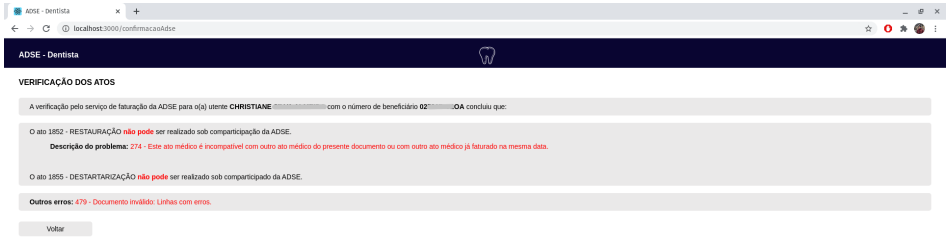


Figure 20: Interface of the Confirmation Scene with ADSE errors.

5.7 DISCUSSION

This first case study consisted of the development of a new web platform for validating if a patient has the rights to do some dental medicine acts through ADSE. To make the application convenient for dentists, it needs to be straightforward and the most user-friendly possible. The development uses a combination of thoughtful design, the latest technologies and frameworks. Therefore, the application can be easily extended to suit the future needs of the hospital.

This application helps the administrative and accounting teams of the hospital by solving issues with patients who perform some acts without having ADSE privileges or that have exceeded the ADSE limit. This can happen because the hospital only bills the medical procedures on the following day of their realisation. For this reason, ADSE does not have to bear the costs of these acts, and patients must be approached by the hospital accountancy to afford all expenditures or the hospital has to pay it by itself, causing needless expenditure. Then, the developed platform solves the *Hospital da Santa Casa da Misericórdia de Vila Verde* problems previously stated by allowing dentists to check whether they can do any dental medicine acts via ADSE before they actually do it.

To allow the application to solve the current issues, some characteristics are relevant to mention. It was designed to make the user interface intuitive and to automate the request to the ADSE web service, without the need for the dentist to fill in all the information about the patient or the hospital, avoiding potential errors

or security failures. This makes the platform inter-operable since it communicates with AIDA and ADSE web services.

Besides that, the web application adopts technologies, namely Node.js and React, that can use the npm package manager to install, update and remove packages. By using these technologies, new features can be easily added by installing packages. Also, allow for installing this application in a new machine and extending its functionalities, both without big complications. The platform is also responsive so it will work, without big issues, in different browsers or different screens.

5.8 CONCLUSION AND FUTURE WORK

The platform to validate the dental medicine acts through ADSE was introduced and discussed in this chapter, as well as the solution's key objectives and the requirements elicited during its development. This first case study carried out within the scope of this dissertation project is composed of a web application, which communicates with external web services, that has been presented and documented during this chapter. Additionally, the benefits gained and how the developed solution can assist the administrative and accounting work at the *Hospital da Santa Casa da Misericórdia de Vila Verde* are also explored.

The main reasons that led to the development of this case study were the need for a platform that verifies if a patient in dental medicine speciality has ADSE rights to perform some act. Previously, if a patient, for any reason, did not have ADSE rights to do a dental medical act, the hospital didn't have how to know. This way, without ADSE verification, the dentists would perform the acts and the hospital would charge them, which would later create issues with ADSE because it would refuse to co-pay for these acts. To get over this issue, the requirements elicitation process began by identifying main points that would justify the design, development and implementation of this new application for validating dental medicine acts through ADSE.

As future work, the application could be extended to support other medical specialities or other procedures available in the *Hospital da Santa Casa da Misericórdia de Vila Verde* and can be co-payed by ADSE. This way, this application could be an ADSE verification platform for all medical acts available in the hospital.

6

CASE STUDY 2 - PLATFORM TO INVOICE THE HOSPITAL ACTS THROUGH ADSE

6.1 INTRODUCTION

The platform for invoicing hospital acts via ADSE is introduced and discussed in this chapter. Like the previous platform, this application is composed of a web application supporting the hospital's administrative and accounting work. Key features include a patient and medical speciality page with important information where the patient admission can be done, manually or invoking the citizen card reader and an invoicing page where the invoice PDF can be seen and downloaded or where the errors are shown.

The problem and its motivations, as well as the platform's key objectives and requirements elicited during development, are defined in the following sections. In addition, all steps involved in the platform's design and development are detailed. The architecture of the platform, the database models used, and the use of Node.js, React, GraphQL, and the citizen card reader middleware are all examined and documented. This chapter ends with a discussion of the results obtained, as well as a brief conclusion and insight into future work.

6.2 CONTEXT AND MOTIVATION

As mentioned in the previous chapter, the *Hospital da Santa Casa da Misericórdia de Vila Verde* provides diverse healthcare access to the population with more than thirty specialities. This way, this hospital has wide affluence for its services with people coming at it from different places and backgrounds so, the hospital must be prepared to deal with this diversity of data soon in the reception of patients.

As occurred previously in the speciality of dental medicine, the reception of patients with ADSE benefits for appointments causes some difficulties to the hospital accounting, due to ADSE restrictions and the accounting workflow. An example is that a patient can only have one speciality appointment per day through ADSE. Because the hospital's accounting team only invoices all the medical acts on the following day, when patients don't comply with these restrictions, this can create a real problem for the hospital, since the ADSE services may refuse these outgoing invoices and so to reimburse this acts.

In this way, this master's dissertation also includes the design, implementation and exploration of a new platform to invoice the hospital acts through ADSE. This way, it is possible to check whether a patient has

ADSE rights to do some appointment in the hospital, verify its identity and data by reading its citizen card, and invoice the appointment immediately.

Therefore, a new framework for the invoicing hospital's entries via ADSE was built and developed, being the second case study of this dissertation project. This consisting of a web application that contacts ADSE and AIDA web services, uses the patient's citizen card to do some validations, and builds and sends invoices. The choices of all the components and features built-in the platform were focused on meetings with the *Hospital da Santa Casa da Misericórdia de Vila Verde* administrative and accounting teams in order to achieve the key objectives for the creation of the solution, in particular, the required business and technical requirements.

6.3 CASE STUDY OBJECTIVES

This web application for invoicing hospital's entries through ADSE was built to eliminate an old issue on appointments invoicing at the *Hospital da Santa Casa da Misericórdia de Vila Verde*.

This case study aims to build, develop, and implement a new platform that interacts with ADSE and AIDA web services, and can interact with an external card reader if needed, in order to verify whether the patient has ADSE rights and if he has its citizen card valid. It also allows the production of medical appointments and acts invoices and send them to ADSE. Then, it is possible to disseminate this problem and, thus, reduce costs to the hospital and potential drawbacks to the accounting team.

The main objectives to fulfil with the development of the platform are the following:

- Researching requirements of the design and development of the platform thought meetings with administrative and accountancy teams of the hospital;
- Usage and development of database models that store all relevant data such as medical speciality and patient's information;
- Usage of APIs from the hospital and web services from AIDA and ADSE to give support to the web application;
- Usage of an external card reader to validate patient's information;
- Production and submission of the medical appointment and acts invoice with all the data necessary in ADSE.
- Development of a web app using React library, including a number of components and features;
- Implementation and deployment of this web application in the *Hospital da Santa Casa da Misericórdia de Vila Verde*;
- Ensuring the proposed platform is called by the existing hospital's software and can be used.

6.4 PLATFORM REQUIREMENTS

In this section, the platform requirements will be presented. In order to make the scope of this application more perceptible, a use case diagram was drawn (figure 21). This diagram explains which actors in the system, as well as which functionalities the system should have.

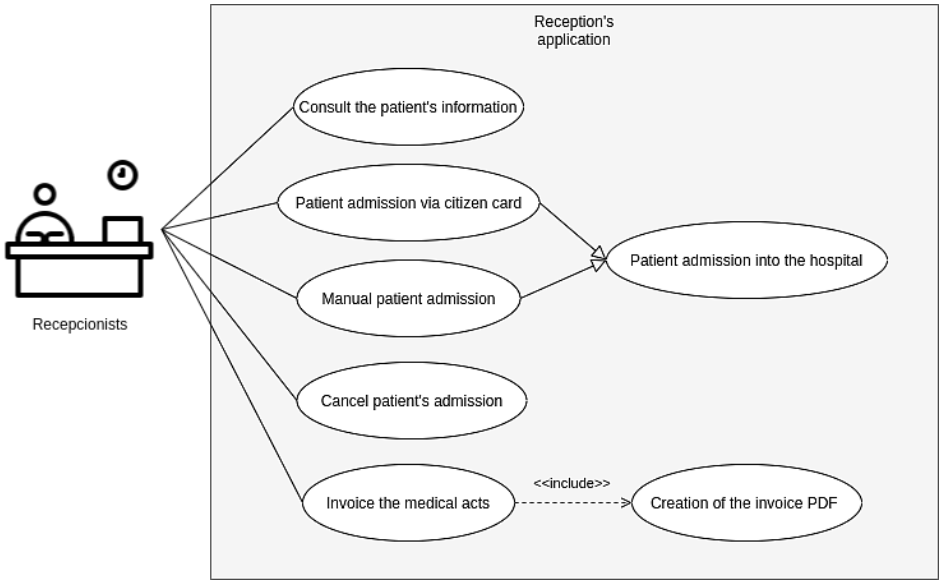


Figure 21: Reception's application use case diagram.

Use cases

- **Consult the patient's information:** A receptionist can consult the patient's hospital data useful to ADSE invoicing.
- **Patient admission via citizen card:** A receptionist can admit the patient into the hospital via his citizen card.
- **Manual patient admission:** A receptionist can manually admit the patient into the hospital.
- **Cancel patient's admission:** A receptionist should be able to cancel the patient's admission.
- **Invoice the medical acts:** A receptionist can send the information of the patient and of the episode to the ADSE web service in order to invoice the medical acts.
- **Consult ADSE web service response:** A receptionist should be able to consult the response from the ADSE web service.
- **Print the invoice PDF:** A receptionist should be able to print the invoice PDF generated.

Based on these use cases, the various functional and non-functional requirements of the web application were identified. These are presented in the Appendix section B.

6.5 BUSINESS REQUIREMENTS

In order to identify and discuss the main objectives and requirements of this platform, multiple meetings with the financial management and accounting teams of the hospital were scheduled. The key issues and concerns with invoicing via ADSE were described, as well as how the proposed platform would help the hospital's administration in saving money and resolving issues created by billing on the day after the medical appointments are performed. The platform's main components, features, and constraints were analysed and discussed, as well as their technical feasibility and goals.

The following are some of the most important system requirements:

- The application should be accessed by the reception team at the *Hospital da Santa Casa da Misericórdia de Vila Verde*;
- The application must be compatible and responsive with all commonly used web browsers;
- The application key's components are the patient's information with its important data and where its admission can be done, and the invoicing page where the billing results from ADSE are shown;
- The application should have an intuitive user interface, adopting a simple and clear design;
- The application must allow for future integration and expansions.

In particular:

- **Patient's information scene**
  - This component must show the acts' medical speciality;
  - It must show clearly and simply the patient's health and personal information;
  - Verifies if a patient has ADSE rights, through a request to the ADSE web service;
  - Allows to do the patient admission at the hospital, manually or by his citizen card;
  - Allows to cancel a previously made admission;
  - Perform a request to the ADSE web service to invoice the medical appointment or acts.
- **Invoicing scene**
  - This component must show and allow to manipulate of the invoice PDF or show potential errors.

6.6 PLATFORM'S ARCHITECTURE AND DEVELOPMENT

To successfully build the application multiple tools and technologies were researched and the most suitable ones were selected to create the solution. The chosen technologies are:

- **Node.js** used to create the web server along with the **Express** framework;
- **Oracle**, where is stored all the information of the hospital and the patients;
- **React.js** used to build the web application's interface;
- **Apollo**, to build UI components that fetch data via GraphQL;
- **GraphQL**, for executing queries and mutations;
- **Citizen Card Middleware** is a set of libraries used to access and gather information from the Portuguese citizen card.

The architecture of the application, the databases tables generated and manipulated, how the citizen card middleware works and how web services are called, as well as the application interface, are all covered in the following subsections.

6.6.1 *Platform's Architecture*

There is only one sort of user in the application, the reception team of the hospital, so they can access all the functionality of it. The reception team can see the acts' speciality and the needed patient's information, and whether he has ADSE rights. If the patient has them, the user may do the patient's admission, manually or via its citizen card, and subsequently invoice the medical appointment or the acts, if he fulfils the ADSE requirements.

As the previous one, this platform was also developed with the React library to create the user interface, GraphQL to execute queries and mutations, and Apollo to make the integration between the front-end and the back-end. The back-end was developed using Node.js together with the Express framework.

To make the patient's admission via his citizen card was used the Portuguese citizen card middleware. This Software Development Kit (SDK) needs the application ***Autenticação.gov***, which allows the management of the Portuguese citizen card. It allows visualise the citizen's information, modifies its notes, changes its PINs and signs files digitally [4].

There is no authentication needed for this application since it is called by the existing hospital's software where the reception team are already logged in. All the information regarding the patients and their admission, the medical acts, all the needed information to invoice them through ADSE and the invoice PDF are stored on the Oracle databases of the hospital. The application uses **XMLHttpRequest** object, **Fetch API** and **node-libcurl** library in order to connect with AIDA and ADSE web services.

Figure 22 represents the platform's architecture and its interactions with the external components.

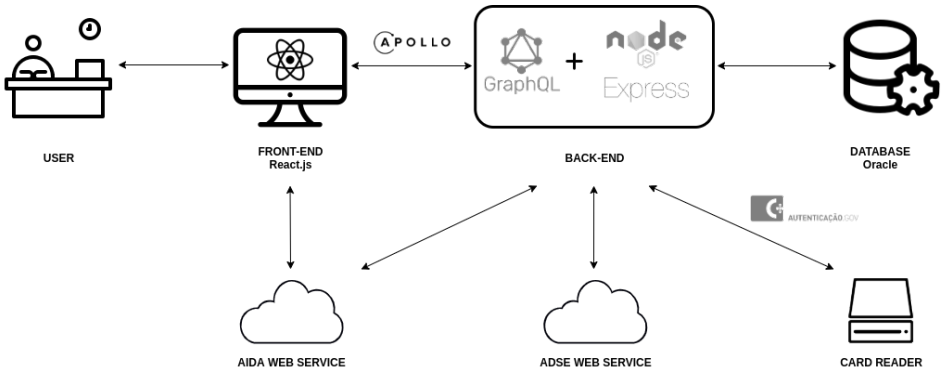


Figure 22: Architecture and tools of the reception's platform.

The developed platform is deployed and installed on a Windows machine (Windows 10 Pro 64-bit) at the *Hospital da Santa Casa da Misericórdia de Vila Verde*, but real-world testing and evaluations are yet to be completed.

6.6.2 *Databases Models*

This platform requires access to the hospital's database for information about patients, episodes, admissions, doctors, teeth, and the needed invoicing information. As in the previous application, the Oracle DBMS stores all tables and relationships, while GraphQL and **oracledb** library are used to connect, to specify the data and to interact with the databases.

Figure 23 shows what tables have been used from the hospital's databases or created to give support to the platform.

6.6.3 *Web Services*

To contact the AIDA and ADSE web services different libraries were used, providing a set of features.

As in the platform to validate the dental medicine acts through ADSE, this platform uses **node-libcurl** to contact the ADSE web service. This web service was contacted only by the server and is used to verify if the patient has ADSE rights, to get his ADSE beneficiary number by his number tax, to invoice the medical acts and to get the description of potential errors. All of these requests are made by a Curl POST and they are sent to the client when they are needed. These requests are similar to the request presented in figure 14 .

To make requests to the AIDA web service the application's server uses only the **Fetch API**. On the other hand, the front-end uses **XMLHttpRequest** for synchronous requests and Fetch API for all the other requests. This web service is contacted by the application to obtain the patient's information and to update the needed data in the hospital's database.

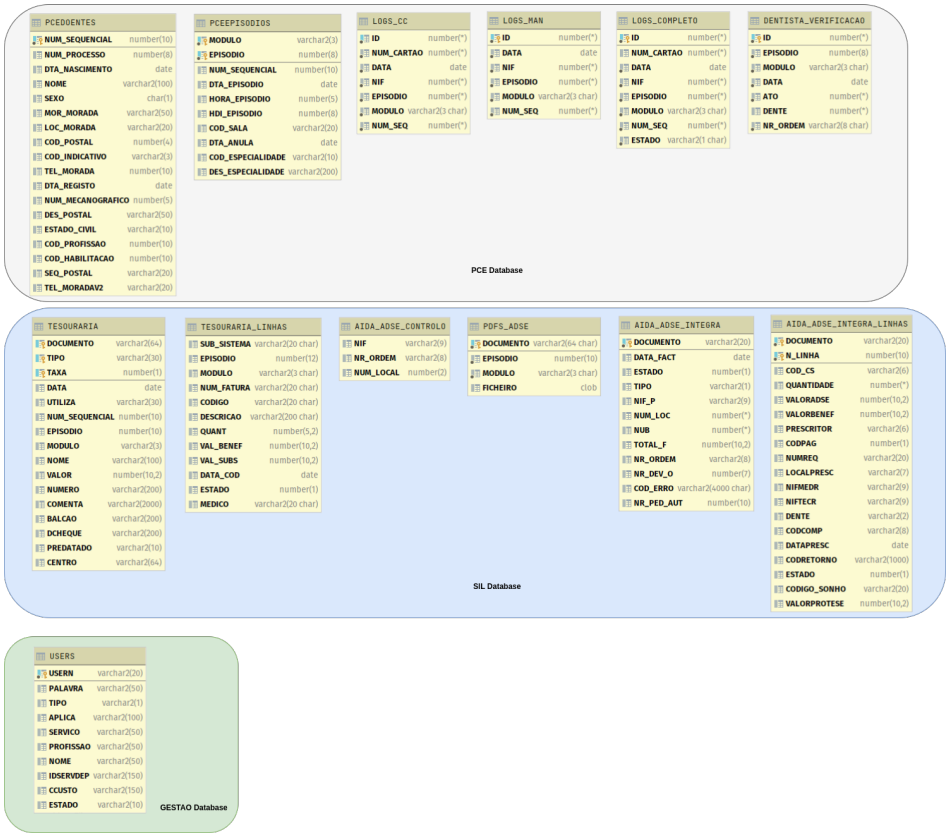


Figure 23: Databases tables used on the reception's platform.

6.6.4 *Citizen Card Middleware*

An important feature of this platform is the patient admission by his citizen card, so this platform must have some middleware that is able to connect to the card reader and extract the needed details from his citizen card.

A preexisting Java SDK, developed by the Portuguese government, was used to accomplish this. This SDK allows to get events for insertions and removals of cards, get identity information, get a photo of a cardholder, get address information and signing documents from the Portuguese citizen card [5]. To use it, it is necessary to have the **pteidlibj** library installed on the computer, which is added automatically when the ***Autenticação.gov*** program is installed. This program provides an interface by which a person can access the electronic features of his citizen card [3].

The Java SDK was updated to only collect the information about the patient that is needed. After that, a jar file was built from the Java SDK to be used in the platform to invoice the hospital acts through ADSE. To accomplish this, a server end-point was generated that executes a child process that runs the jar file as required.

As mentioned above, in this application, the admission via the citizen card needs to connect to the card reader and get the patient's information existing in his card. A check of the card's validity is also done. After that, the data from the citizen card is cross-checked and verified with the data present in the database. This whole process can be easily understood by analysing the activity diagram presented in figure 24.

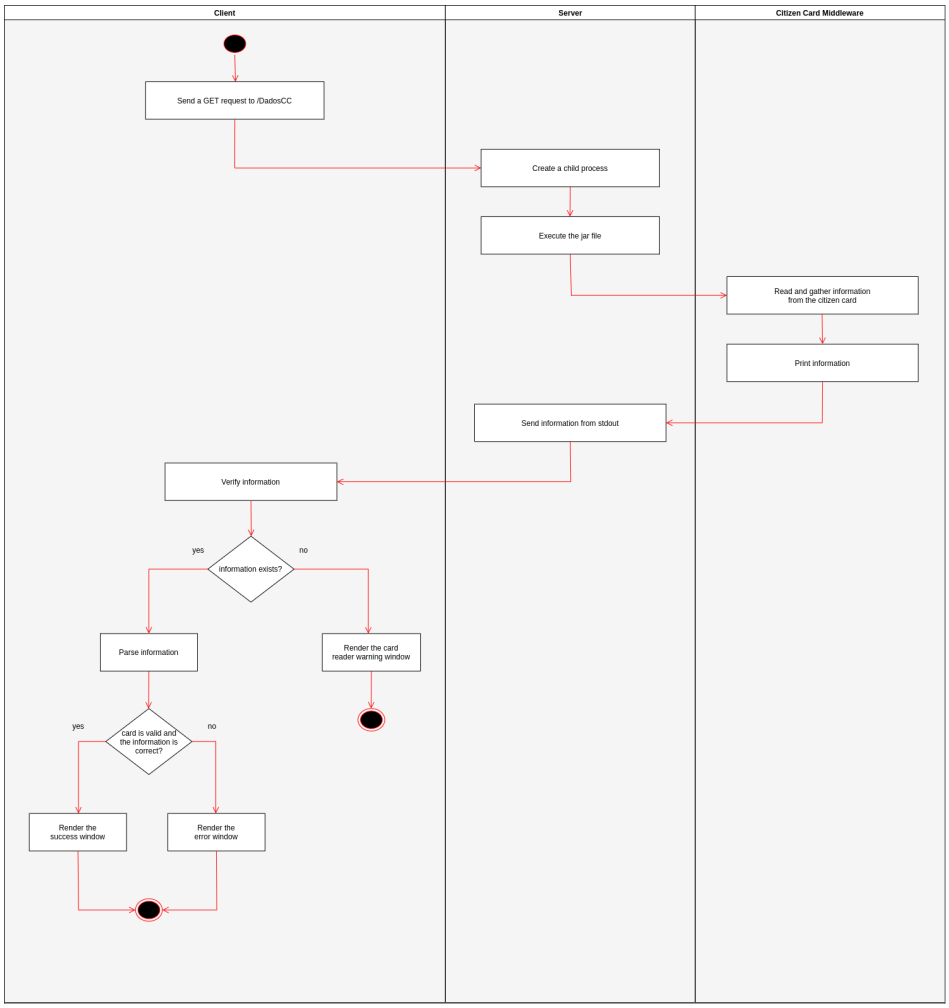


Figure 24: Activity diagram showing the patient admission via citizen card process.

6.6.5 *Web Application*

The web application will support the administrative work of the hospital for the medical appointments and the acts resulting by them for patients with ADSE rights. As previously mentioned, this application helps the reception team to find out if a patient has ADSE rights and whether he fulfils the ADSE requirements to do a medical appointment.

As in the previously developed application, the React.js library was used to build a SPA. This library was chosen because it allows the implementation of additional features if it is needed, by adopting modern technologies and frameworks. Create React App is also used since is officially supported to create React SPA. As a SPA application, it was appropriate to use React Router for declarative routing, XMLHttpRequest and Fetch API for web service and server requests, and Apollo Client and other libraries to easily build its UI components and to fetching and parsing data via GraphQL.

This web platform is comprised of two scenes, namely the patient's information scene and the invoicing scene, which have their own components.

Patient's Information Scene

The Patient's Information Scene presents the information about the episode and the patient, useful to ADSE invoicing, as shown in figure 25.

Figure 25: Interface of the Patient's Information Scene of the reception's application.

The application receives, to load the information about the episode and the patient, a GET request from the existing hospital's program with the number of the episode, the hospital's module, the speciality of the medical appointment and if it is an emergency medical appointment also receives the doctor's number on the Portuguese Medical Association. The platform then performs a series of checks. By these parameters, the application retrieves information about the episode and verifies whether or not the medical acts have been saved in the hospital's database and whether or not the patient has been admitted previously. These checks are useful for situations when the patient has to pay more than one medical act separately.

When the application finishes these checks, it gets the patient's information from the hospital's database and loads it and the information about the episode to the scene's components. Lastly, the application requests the AIDA web service with the patient's process number to obtain the patient's ADSE beneficiary number. If the patient has ADSE rights, then the reception team can make his admission to the hospital. However, if that not happen, the application shows an error, as in figure 26.

As mentioned earlier, this application bears two different types of patient admission to the hospital: manually or via his citizen card. The manual admission nothing more is than a simple database insertion with important information that needs to be saved in order to make a register of the patient (figure 27). The admission via the citizen card needs to gather the patient's information from his citizen card using an external card reader. If all the gathered data is correct and the card is valid, then the admission is completed, and the patient admitted, as shown in figure 28a. If data from the citizen card differs from the hospital's database data or if the card isn't valid, the patient admission is not possible, and an error window appears. (figure 28b). For some reason, if the

Figure 26: Interface of the Patient's Information Scene of the reception's application for patients without ADSE rights.

application cannot read the data from the citizen card through the card reader or if no card reader is connected, a warning window appears (figure 28c).

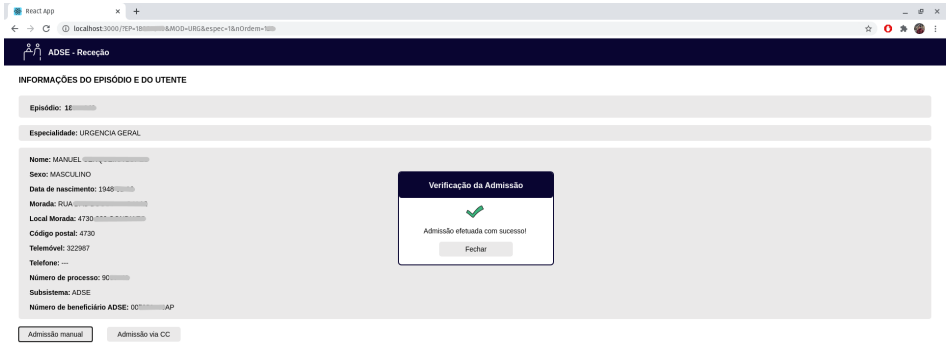
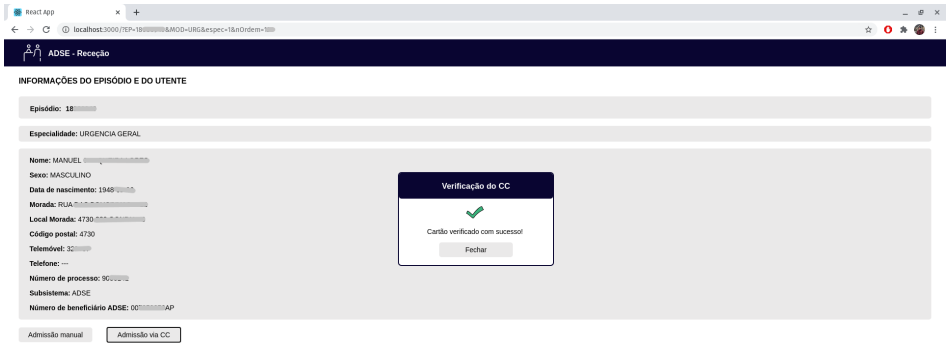


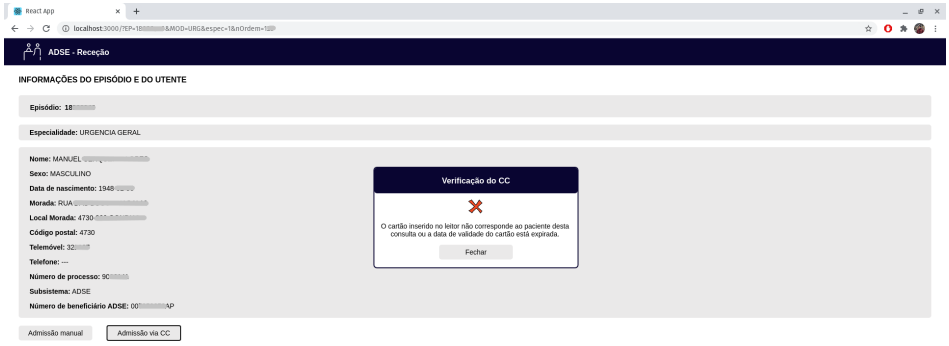
Figure 27: Interface of a manual patient admission on Patient's Information Scene.

When the patient is admitted to the hospital, the reception team can cancel his admission or invoice the medical appointment or the medical acts resulting from it. (figure 29). If the user intends to cancel the patient's admission, the data will be removed from the related table with the log (*LOGCC* or *LOGMAN*), but not from the database (table *LOGS_COMPLETO*). This information is preserved since the hospital needs to keep this record. Then, a window with a successful message will appear after this, as shown in figure 30.

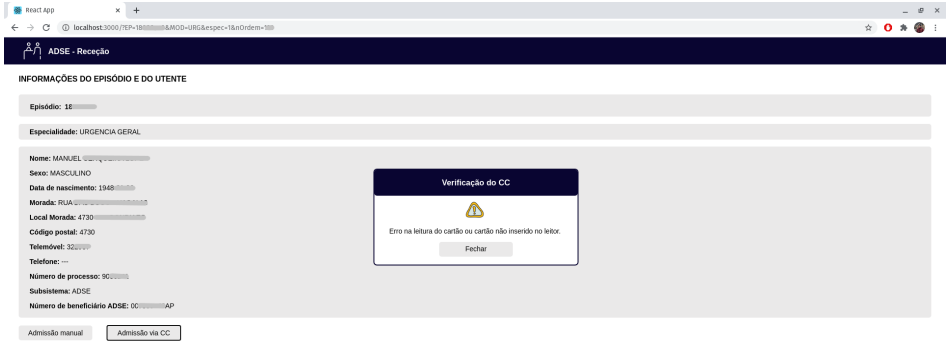
In turn, the user can invoice the medical appointment or the medical acts by pressing the "*Faturar*" button. Depending on the hospital's module of the episode, the application will behave differently. The explanation for this is that the episode is not related to the doctor's number on the Portuguese Medical Association, which is needed to invoice via ADSE. In the case of the module is 'CON' or 'COU', the application gets the doctor's number from the episode information from the AIDA web service. In the case of the module is 'URG', the application gets the doctor's number from the parameters of the initial GET request. For all other modules, the user needs to select the doctor that will perform the medical acts from a list, as shown in figure 31.



(a) Successful patient's admission



(b) Unsuccessful patient's admission



(c) Warning in the patient's admission

Figure 28: Interface of patient's admission via citizen card on Patient's Information Scene.

Figure 29: Interface of an admitted patient on Patient's Information Scene.

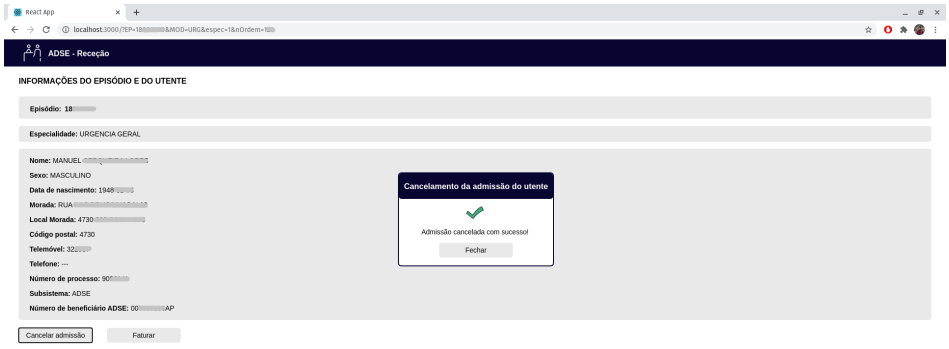


Figure 30: Interface of cancel the patient's admission on Patient's Information Scene.

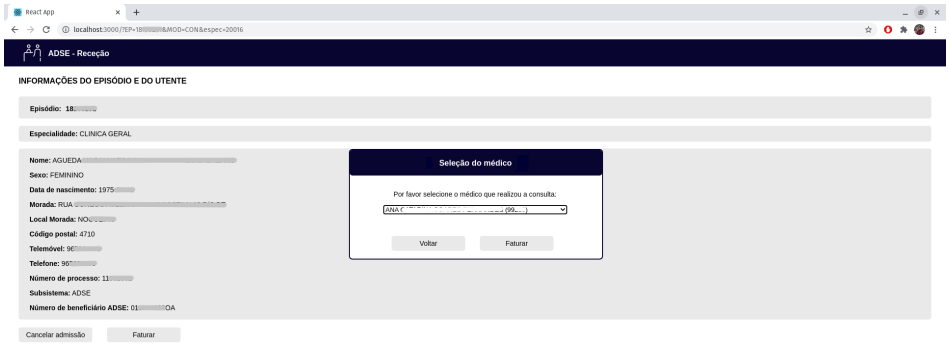
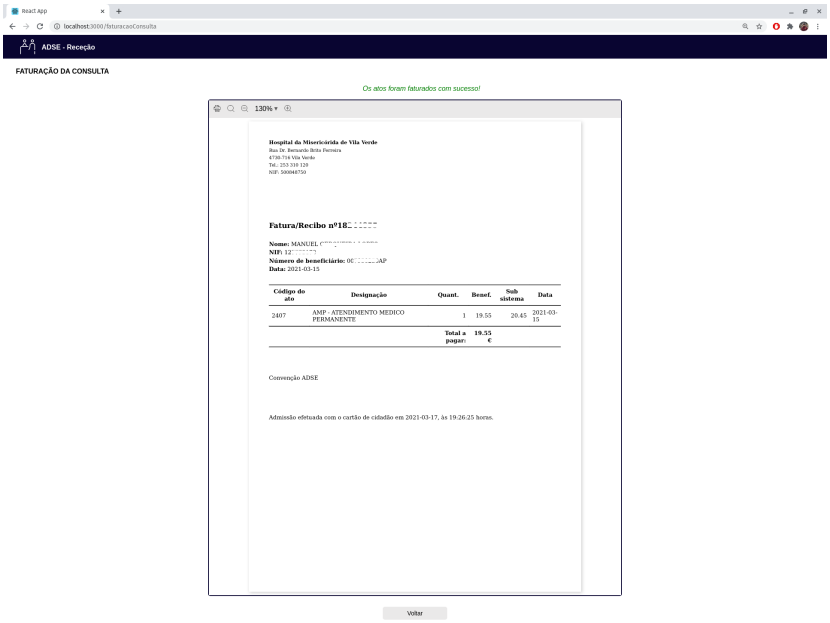


Figure 31: Interface of doctors selection on Patient's Information Scene.

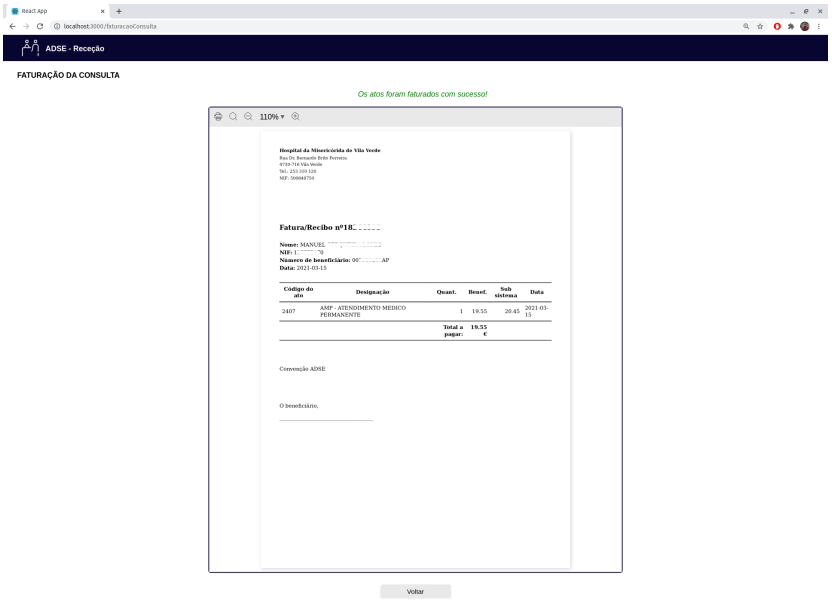
Invoicing Scene

After that, the doctor's number and all the information about the episode are sent to the server in order to invoice the medical acts. Then, the server needs to gather some additional information from the hospital's database and the AIDA web service and do some checks about the medical acts. If everything is correct, the

server generates the invoice PDF with the hospital's and patient's information and with medical acts that will be performed. Then, the PDF invoice is saved in the hospital's database, and the invoice request is sent to the ADSE web service with all the needed information. If everything goes well, this information is saved in the hospital's database, and the PDF invoice is sent to the user's interface.



(a) Invoiced PDF for admissions via CC.



(b) Invoiced PDF for manual admissions.

Figure 32: Interface of the Invoice Scene with the invoice PDF generated.

As shown in figure 32a, if the patient did make admission via his citizen card, the invoice process is completed since the generated PDF has that indication and do not need to be signed by the patient. Otherwise, the generated PDF must be signed by the patient and replaced in the ADSE web service, as shown in figure 32b. Then, this platform allows printing this document from it, being the hospital's accounting team responsible for sending it to the ADSE later.

If some problem occurs during the invoicing process, the ADSE response will come with the errors and they will be parsed by the application to be described, as demonstrated in figure 33.

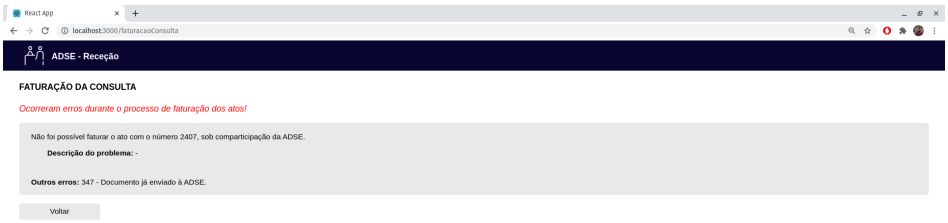


Figure 33: Interface of the Invoice Scene with ADSE errors.

Finally, the website displays a warning after all of the patient's medical acts have been invoiced, or if the patient has no medical acts to invoice (figure 34).

Figure 34: Interface of the Patient's Information Scene when the acts are invoiced.

6.7 DISCUSSION

This second case study consisted of the development of a new platform for invoicing medical appointments and the possible acts resulting from them through ADSE. This application needs to be straightforward and the most user-friendly possible to make its usage easy to the hospital's reception team. The development uses a combination of the latest technologies, frameworks and a thoughtful design. Therefore, the application can be easily extended to suit the future needs of the hospital.

This platform helps the hospital's administrative and accounting teams solving issues with patients who perform some acts without complying with the ADSE requirements or that have exceeded the ADSE limits. As in

the previous application, this can happen because the hospital only bills the medical procedures on the following day of their realisation. For this reason, ADSE does not have to bear the costs of these acts and patients must be contacted by the hospital accountancy to afford all expenditures, or the hospital has to pay it by itself, causing needless expense. The built platform then tackles the *Hospital da Santa Casa da Misericórdia de Vila Verde* issues by checking whether some patient can do a medical appointment or a procedure and if he can invoicing it at that time.

In order for the application to solve the current issues, some characteristics are relevant to mention. It was designed to make the user interface simple, intuitive and to automate and streamline the invoice process through ADSE, without the need for the reception team to fill all the information about the patient, the medical acts or the hospital, avoiding potential errors or security failures. Besides, when a patient is admitted using his citizen card, the invoicing process is shorter and simpler because he does not need to sign the bill, and the hospital's accounting team does not need to re-enter it into the ADSE later. This application also enables the receptionists to quickly identify possible errors that can occur during the billing process. These characteristics make the platform inter-operable since it communicates with AIDA and ADSE web services to complete the invoicing process.

Aside from that, the web application uses Node.js and React.js, which can install, update, and uninstall packages using the npm package manager. New features can be quickly implemented by downloading new packages using these technologies. Also, they enable the installation of this application on new computers as well as the extension of its functionalities without major complications. The platform is also responsive, meaning it can run properly in a variety of browsers and on a variety of screens.

6.8 CONCLUSION AND FUTURE WORK

This chapter presents and discusses the platform to invoice the hospital acts through ADSE, as well as the solution's main objectives and requirements elicited during its development. In addition, presents and documents the second case study completed as part of this dissertation project, which consists of a web application that interacts with external web services and has the ability to read information from the Portuguese citizen card. The advantages gained, as well as how the established solution will help the administrative and accounting teams at the *Hospital da Santa Casa da Misericórdia de Vila Verde*, are also discussed.

The development of this case study was motivated by the need for an application that verifies the patient's condition in the ADSE and invoices medical acts via ADSE at the time of the patient admission to the hospital. Previously, if a patient, for any reason, did not have ADSE rights or have exceeded ADSE limits to do some medical appointment or act, the hospital didn't have how to know. In this way, without ADSE verification, the hospital would allow the patient to perform the medical acts, which would create issues with the ADSE insurance because it would refuse to co-pay for these acts. To get over this issue, the requirements elicitation began by identifying main points that would justify the design, development and implementation of this new platform for invoice hospital acts through ADSE.

As future work, new functionalities could be added to the platform in order to cover all the characteristics of the invoicing process through the ADSE web service as, for example, the replacement of the invoice PDF.

7

PROOF OF CONCEPT

7.1 INTRODUCTION

A PoC is considered a crucial method in the process of planning, creating, implementing and proposing ITs software solutions. They verify whether a project meets its key characteristics and established goals, as well as identify possible defects or errors in the solution being created. Furthermore, the notion of evidence in science seeks to address a question whose answer is generally applicable in areas beyond those tested for [28].

Hence, PoC is a theoretical demonstration of a product, process or concept to determine whether an idea can be turned into a reality, i.e. test whether an idea is viable and explore the idea's potential to be developed or built [51]. A methodology is made up of practical models that can prove or validate a concept through analysis and development. Proof of concept research "presents a discovery about our knowledge of the world and the structure of existence [28].

In this chapter, two Strengths Weaknesses Opportunities Threats (SWOT) studies were performed on both platforms. As discussed on section 3.4, a SWOT analysis facilitates a fact-based and data-driven analysis of the internal and external variables that are advantageous or detrimental, helping reveal prospective opportunities to exploit and recognise potential vulnerabilities [20, 69].

7.2 SWOT ANALYSIS

In the following subsections, the Strengths Weaknesses Opportunities Threats (SWOT) analysis of the platforms is presented. The SWOT analysis planning technique is used to develop strong strategic plans by analysing the strengths and weaknesses (internal factors), in addition to opportunities and threats (external factors), of a business or project plan [35]. A SWOT analysis can help reveal opportunities to exploit and understand potential weaknesses, and also develop strong strategies plans [35, 69].

7.2.1 *Platform to validate the dental medicine acts through ADSE*

The first case study of this dissertation project - namely design, development, and implementation of a platform to validate the dental medicine acts through ADSE - was subject to a SWOT analysis. Thus, the solution's strengths, weaknesses, opportunities, and threats were extracted and identified.

The following strengths of the platform were identified:

- Development of high quality, intuitive to use platform;
- Clear and simple user interface;
- Support for the most common web browsers;
- A new way of interaction between the *Hospital da Santa Casa da Misericórdia de Vila Verde* (dentists and their patients) and ADSE;
- Sophisticated and real-time information about patients' ADSE rights and about the dentist's acts validation through ADSE;
- All validations are stored in the hospital's database;
- Easy to maintain architecture, where new features can be quickly added.

However, internal attributes of the platform that are harmful towards its successful development and integration, the weaknesses, were also described:

- Constant internet connection is required;
- Dependency of external web services to work.

In addition, the external attributes to the platform that are helpful towards its successful development and integration, the platform's opportunities, were extracted:

- The platform can be expanded to other hospital's specialities;
- Possible adoption of this platform by other health institutions;

On the other hand, possible external threats to the system were identified:

- Maintenance of the platform and the needed web services is required;
- Competition from external companies and competitiveness of other platforms.

7.2.2 *Platform to invoice the hospital acts through ADSE*

A SWOT analysis was conducted on the second case study of this dissertation project, which included the design, creation, and implementation of a platform to invoice the hospital acts through ADSE. The strengths, weaknesses, opportunities, and threats of the solution were thus extracted and established.

The following strengths of the platform were identified:

- Development of high quality, user-friendly platform;
- Simple and clear user interface;
- Support for the most common web browsers;
- A new way of interaction between the administrative and accounting teams of the *Hospital da Santa Casa da Misericórdia de Vila Verde* and ADSE in the invoicing process;
- Sophisticated and real-time information about patients' ADSE rights and about the invoicing process through ADSE;
- Patients admission to the hospital and the medical acts invoicing process can be done by their citizen card;
- Production of the invoice PDF;
- All the information about the invoicing process are stored in the hospital's database;
- Easy to maintain architecture, where new features can be quickly added.

However, the weaknesses of the platform were also described:

- Require constant internet connection;
- Dependency of external web services to work.

In addition, the platform's opportunities were extracted:

- New functionalities can be added to the platform in order to cover all the possibilities on the ADSE invoice process;
- Possible adoption of this platform by other health institutions.

On the other hand, possible external threats to the application were identified:

- Maintenance of the platform and the needed web services is required;
- Competition from external companies and competitiveness of other platforms.

7.3 CONCLUSION

This chapter introduced and defended the proof of concept of the planned and created case studies during this dissertation project. Both platforms were subjected to the SWOT analyses, which identified their key strengths, weaknesses, opportunities, and threats.

As a result, the feasibility, utility, and usability of the platforms have been presented in this chapter. The key points gathered during the SWOT analyses allow their easy maintainability and expansion for the health institution. Additionally, all the developed features are user-friendly and simple to use, thus meeting all requirements of the *Hospital da Santa Casa da Misericórdia de Vila Verde*. This is due to the fact that a thought requirements elicitation process and a carefully thought-out design. These applications will greatly help the hospital's administrative and accounting teams with the ADSE insurance processes.

The next and last chapter aims to summarise and present the main conclusions and contributions obtained through the development of these platforms. In addition, proposals for future work in order to improve the platforms and their applications are presented.

8

CONCLUSIONS AND FUTURE WORK

8.1 INTRODUCTION

After the scope of the project and the problems it aims to solve have been presented, and after reporting the development, implementation and evaluation of the proposed solution, it is then possible to summarise the dissertation and draw the main conclusions about the work carried out.

In section 1.3 of this document, two Research Questions were presented as the platforms' objectives, serving as a guide for exploring the case studies presented. Hence, in the following sections, the answers to the questions proposed are presented, summarising the main contributions of this dissertation project, as well as including some suggestions for future work.

8.2 MAIN CONTRIBUTIONS

Information technology systems are changing the healthcare sector, from the identification of disease cures and the development of new therapeutic methods to improving the diagnosis of patients and improving the efficient delivery of healthcare. From small health facilities to large-scale hospitals, the increased use of IT methods in clinical practices increases the quality of care of the patient, as well as optimises the resources of the health institution [8]. Therefore, the ISs in a healthcare institution must interoperate with each other by adopting standards in the exchange of information. Some platforms are trying to reach this interoperability in the healthcare units, as the AIDA, by supporting the use of web-based services to provide direct access to knowledge regardless of the difference between the ISs.

The present dissertation aims to analyse the specific needs of a healthcare institution in order to understand what problems they have in the fields mentioned above and what functionalities and requirements are needed in new IT artefacts to overtake this issue. This project results from the collaboration of the Department of Informatics of the Minho University, in Braga, with a Portuguese hospital, the *Hospital da Santa Casa da Misericórdia de Vila Verde*. The project counts on the collaboration of a work team made up of engineers, a financial administrator and an accountant, with the ability to intervene and participate in the multiple phases of the project.

It was possible to delineate a valid strategy starting from topics and main ideas that, with the revision of the literature, became more stable and justified, thanks to the methodologies and investigative strategies chosen.

Accordingly, the result of this project is the creation of two new web applications that allows the hospital's ISs to interact with the ADSE health insurance in a simple way. These applications gather the needed information about the patient and/or the episode from the hospital's database and AIDA web service and send it to the ADSE web service, in a new interoperable digital ecosystem.

In order to answer the first research question, a case study was proposed, consisting of the development of a new web platform to validate the dental medicine acts through ADSE. This platform, developed using the React library, will support the hospital's administrative and accounting teams by solving problems with patients who performed some acts without having ADSE privileges or that have exceeded the ADSE limit. This way, the dentists only perform the medical acts after having the ADSE validation, reducing possible issues with the ADSE co-payment and helping the hospital to save funds.

Afterwards, the second case study was introduced, consisting of the development of another web application to invoice the hospital acts through ADSE. This platform is also developed using the React library and will be implemented in the hospital's reception area and will support the hospital by solving issues with the invoicing process through ADSE. As in this hospital the medical acts performed over the ADSE co-payment are only invoiced on the following day some patients don't comply with the ADSE restrictions and perform the acts anyway. With this application, this is not possible anymore, since it allows to perform the patient admission into the hospital and invoice, at that moment, the medical appointment and/or their related acts through ADSE. Then, this platform allows the hospital to check the patients' ADSE situation and save funds.

To give support and validation to both case studies, a Proof of Concept (PoC) methodology was carried out, consisting of Strengths Weaknesses Opportunities Threats (SWOT) analyses. This study has confirmed the platforms' viability and effectiveness based on their feature set, their architecture, and their user-friendly interfaces.

Additionally, this Master's work also has scientific outcomes, namely a research paper with the following reference:

- F. Nunes, R. Sousa, A. Abelha, J. Machado. New Generation of Interoperable Artifacts in Medical Informatics. *Transactions on Computational Science & Computational Intelligence*. 2021, In press.

8.3 PROSPECT FOR FUTURE WORK

Multiple solutions, also known as Information Technology (IT) artefacts, were developed as part of this dissertation project. These artefacts will not only make the hospital's administrative and accounting work easier, but will also support the communication between the hospital and the ADSE insurance. It is expected that both web applications help the hospital optimising the verification/invoice process through ADSE allowing it to save funds.

The first case study, namely the design and development of the platform to validate the dental medicine acts through ADSE, has been implemented in the hospital and it has already been checked by the administrative and accounting teams, making it ready to be used by the dentists. Later, if necessary, the application can be extended to support other medical specialities or procedures available in the *Hospital da Santa Casa da Misericórdia de Vila Verde*.

The second case study, namely the design and implementation of the platform to invoice the hospital acts through ADSE, has been deployed in the hospital, with the administrative and accounting departments conducting tests to ensure that everything is working properly. Thereafter, new functionalities can be added to the platform in order to cover all the existing possibilities of the invoicing process through the ADSE web service.

This way, all of the previously defined Research Questions in section 1.3 and its objectives were successfully accomplished.

BIBLIOGRAPHY

[1] *HIMSS Dictionary of Healthcare Information Technology Terms, Acronyms and Organizations, Third Edition*. HIMSS Book Series. Healthcare Information & Management Systems Society, 2013. ISBN 9781938904295. Appendix B, p190.

[2] A. Abelha, J. Machado, M. Santos, S. Allegro, F. Rua, M. Paiva, and J. Neves. Agency for integration, diffusion and archive of medical information. In *Proceedings of the Third IASTED International Conference - Artificial Intelligence and Applications*, Benalmadena, Spain, 2002. ISBN 0-88986-390-3.

[3] Agência para a Modernização Administrativa, IP. Aplicação autenticação.gov para computador, 2021. URL https://www.autenticacao.gov.pt/cc-aplicacao. Last accessed in 18/03/2021.

[4] Agência para a Modernização Administrativa, IP and Instituto dos Registos e do Notariado, IP. Manual do sdk – middleware do cartão de cidadão, 2019. URL https://amagovpt.github.io/autenticacao.gov/user_manual.html. Last accessed in 18/03/2021.

[5] Agência para a Modernização Administrativa, IP and Instituto dos Registos e do Notariado, IP. Java sdk examples, 2021. URL https://amagovpt.github.io/docs.autenticacao.gov/sdk/java/. Last accessed in 18/03/2021.

[6] W. Y. Arms. A spectrum of interoperability: the site of science for prototype for the nsdl. 2002. URL http://www.dlib.org/dlib/january02/arms/01arms.html. Accessed in 23/11/2019.

[7] E. Brown. *Web Development with Node and Express: Leveraging the JavaScript Stack*. O'Reilly Media, 2014. ISBN 9781491902295. URL https://books.google.pt/books?id=HsLvAwAAQBAJ.

[8] L. Cardoso, F. Marins, F. Portela, M. Santos, A. Abelha, and J. Machado. The next generation of interoperability agents in healthcare. *International Journal of Environmental Research and Public Health*, 11:5349 – 5371, 2014.

[9] L. Cardoso, F. Marins, C. Quintas, F. Portela, M. Santos, A. Abelha, and J. Machado. Interoperability in healthcare. In *Health Care Delivery and Clinical Science*, pages 689–714, 2017. doi: 10.4018/978-1-5225-3926-1.ch036.

[10] C. Carr and S. Moore. Ihe: a model for driving adoption of standards. *Computerized Medical Imaging and Graphics*, 27(2):137 – 146, 2003. ISSN 0895-6111. doi: https://doi.org/10.1016/S0895-6111(02)00087-3. URL http://www.sciencedirect.com/science/

article/pii/S0895611102000873. Picture Archiving and Communication Systems 20 Years Later.

[11] Ajinkya Chanshetty. How does the virtual dom work in react!, 2020. URL https://dev.to/_don_2/how-does-the-virtual-dom-work-in-react-1hin. Last accessed in 29/03/2021.

[12] T. Connolly and C. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Addison-Wesley, England, 6th edition, 2015. ISBN 9780132943260. URL https://www.pearson.com/us/higher-education/program/Connolly-Database-Systems-A-Practical-Approach-to-Design-Implementation-a PGM116956.html?tab=overview.

[13] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *Internet Computing, IEEE*, 6:86 – 93, 04 2002. doi: 10.1109/4236.991449.

[14] Devathon Team. Graphql vs rest in 2020: A detailed comparison, 2019. URL https://devathon.com/blog/graphql-vs-or-rest/. Accessed in 08/09/2020.

[15] Facebook Inc. React - a javascript library for building user interfaces, 2019. URL https://reactjs.org/. Accessed in 07/09/2020.

[16] Facebook Inc. Getting started, 2020. URL https://create-react-app.dev/docs/getting-started. Accessed in 25/10/2020.

[17] Facebook Inc. React.component, 2021. URL https://reactjs.org/docs/react-component.html. Last accessed in 29/03/2021.

[18] Facebook Inc. Virtual dom and internals, 2021. URL https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom. Last accessed in 29/03/2021.

[19] F. Farinelli and M. B. Almeida. Interoperabilidade semântica em sistemas de informação de saúde por meio de ontologias formais e informais: um estudo da norma openehr. In *Conferência Biredial Istec*, 2014.

[20] M. Grant. Swot analysis definition, 2019. URL https://www.investopedia.com/terms/s/swot.asp. Last accessed in 10/02/2021.

[21] M. Guy. Interoperability focus: looking at interoperability. 2005. URL http://www.ukoln.ac.uk/interop-focus/about/leaflet.html. Accessed in 23/11/2019.

[22] G. Hay and G. Castilla. Object-based image analysis: Strengths, weaknesses, opportunities and threats (swot). volume 36, 05 2006.

[23] W. R. Hersh. Medical InformaticsImproving Health Care Through Information. *JAMA*, 288(16):1955–1958, 10 2002. ISSN 0098-7484. doi: 10.1001/jama.288.16.1955. URL https://doi.org/10.1001/jama.288.16.1955.

[24] A. Hevner and S. Chatterjee. *Design science research in information systems*. Springer, 2010. ISBN 1441956522.

[25] S. Holmes and C. Herber. *Getting MEAN with Mongo, Express, Angular, and Node*. Manning Publications, 2019. ISBN 9781617294754. URL https://books.google.pt/books?id=rfgpswEACAAJ.

[26] R. Jacobson. *Information Design*. MIT Press, Massachusetts, 1999. ISBN 9780262600354.

[27] K. Janovsky and World Health Organization. Division of Strengthening of Health Systems. The challenge of implementation : district health systems for primary health care / [prepared by] katja janovsky, 1988.

[28] C. Kendig. What is proof of concept research and how does it generate epistemic and ethical categories for future scientific practice? *Science and engineering ethics*, 22, 05 2015. doi: 10.1007/s11948-015-9654-0.

[29] B. Langefors and B. Sundgren. *Information Systems Architecture*. Petrocelli/Charter, 1975. ISBN 9780884053002. URL https://books.google.pt/books?id=i4KCwJxHxfYC.

[30] J. Machado. node-libcurl, 2020. URL https://www.npmjs.com/package/node-libcurl. Accessed in 24/10/2020.

[31] W. Maj. React lifecycle methods diagram, 2021. URL https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/. Last accessed in 29/03/2021.

[32] MeSH. Medical subject headings - health information systems, 1987. URL https://www.ncbi.nlm.nih.gov/mesh/68006751. Accessed in 18/10/2020.

[33] Meteor Development Group Inc. The apollo platform, 2020. URL https://www.apollographql.com/docs/intro/platform/. Accessed in 08/09/2020.

[34] P. Miller. Interoperability. what is it and why should i want it? 2000. URL http://www.ariadne.ac.uk/issue24/interoperability. Accessed in 23/11/2019.

[35] Mind Tools Content Team. Swot analysis, 2019. URL https://www.mindtools.com/pages/article/newTMC_05.htm. Last accessed in 10/02/2021.

[36] M. Miranda, J. Duarte, A. Abelha, J. Machado, and J. Neves. Interoperability in healthcare. In *EUROPEAN SIMULATION AND MODELLING CONFERENCE (ESM 2010), Hasselt, Belgium, 2010 – "Proceedings of the European Simulation and Modelling Conference (ESM 2010) ". [S.l.].*

[37] M. Miranda, M. Salazar, F. Portela, M. Santos, A. Abelha, J. Neves, and J. Machado. Multi-agent systems for hl7 interoperability services. *Procedia Technology*, 5:725 – 733, 2012. ISSN 2212-0173. doi: https://doi.org/10.1016/j.protcy.2012.09.080. URL http://www.sciencedirect.com/science/article/pii/S2212017312005117. 4th Conference of ENTERprise Information Systems – aligning technology, organizations and people (CENTERIS 2012).

[38] J. M. G. Montóm. Interoperabilidad: La torre de babel de los sistemas de salud, 2016. URL http://laesalud.com/hackathonsalud/2016/interoperabilidad/interoperabilidad-informacion-echcos-sistema-salud/. Last accessed in 27/03/2021.

[39] J. M. G. Montóm. Aplicação autenticação.gov para computador, 2017. URL https://www.ehcos.com/en/interoperability-in-healthcare-systems-successes-and-new-challenges-to-va Last accessed in 27/03/2021.

[40] Mozilla and individual contributors. Xmlhttprequest, 2020. URL https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest. Accessed in 24/10/2020.

[41] Mozilla and individual contributors. Fetch api, 2021. URL https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API. Accessed in 18/03/2021.

[42] National Academy of Engineering and Institute of Medicine. *Building a Better Delivery System: A New Engineering/Health Care Partnership*. The National Academies Press, Washington, DC, 2005. ISBN 978-0-309-09643-0. doi: 10.17226/11378. URL https://www.nap.edu/catalog/11378/building-a-better-delivery-system-a-new-engineeringhealth-care-partnershi

[43] OpenJS Foundation. Express - fast, unopinionated, minimalist web framework for node.js, 2019. URL https://expressjs.com/. Accessed in 16/05/2020.

[44] OpenJS Foundation. Introduction to node.js, 2021. URL https://nodejs.dev/learn/introduction-to-nodejs. Last accessed in 30/03/2021.

[45] B. Orgun and J. Vu. Hl7 ontology and mobile agents for interoperability in heterogeneous medical information systems. *Computers in Biology and Medicine*, 36(7):817 – 836, 2006. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2005.04.010. URL http://www.sciencedirect.com/science/article/pii/S0010482505000806. Special Issue on Medical Ontologies.

[46] L. Palazzo, M. Rossi, A. F. Dragoni, A. Claudi, G. Dolcini, and P. Sernani. A multi-agent architecture for health information systems. volume 252, pages 375–384, 05 2013. doi: 10.3233/978-1-61499-254-7-375.

[47] H. Peixoto, J. Machado, A. Abelha, J. Neves, A. Correia, and M. Santos. Aasys : Appointment alert system : an open-source- based software to improve show rates in a health care unity. In *Proceedings of the European Simulation and Modelling Conference.*, 2011.

[48] H. Peixoto, M. Santos, A. Abelha, and J. Machado. Intelligence in interoperability with aida. In *Proceedings of the 20th International Conference on Foundations of Intelligent Systems*, ISMIS'12, page 264–273, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642346231. doi: 10.1007/978-3-642-34624-8_31. URL https://doi.org/10.1007/978-3-642-34624-8_31.

[49] H. Peixoto, A. Domingues, and B. Fernandes. *Steps towards Interoperability in Healthcare Environment*. 02 2016. doi: 10.4018/978-1-4666-9882-6.ch001.

[50] P. Pinto. Tutorial node.js: Saiba como usar a framework express, 2019. URL https://pplware.sapo.pt/tutoriais/tutorial-node-js-como-usar-o-express/. Accessed in 16/05/2020.

[51] M. Pratt. proof of concept (poc), 2020. URL https://searchcio.techtarget.com/definition/proof-of-concept-POC. Last accessed in 10/02/2021.

[52] Red Hat, Inc. O que é graphql?, 2020. URL https://www.redhat.com/pt-br/topics/api/what-is-graphql. Accessed in 07/09/2020.

[53] D. Rodrigues. Design science research como caminho metodológico para disciplinas e projetos de design da informação. *Revista Brasileira de Design da Informação*, 15(1):111–124, 2018. ISSN 1808-5377.

[54] L. F. Sayão and C. H. Marcondes. O desafio da interoperabilidade e as novas perspectivas para as bibliotecas digitais. *TransInformação*, 20(2):133–148, 2008. doi: 10.1590/S0103-37862008000200002.

[55] Serviço de Bioestatística e Informática Médica da Faculdade de Medicina da Universidade do Porto. Sistemas de informação em saúde, 2006. URL http://im.med.up.pt/si_saude/si_saude.html. Accessed in 01/02/2021.

[56] Serviços Partilhados do Ministério da Saúde. Adse: Direitos e deveres., 2020. URL https://www2.adse.pt/beneficiarios/direitos-e-deveres/. Last accessed in 27/03/2021.

[57] N. Shedroff et al. Information interaction design: A unified field theory of design. *Information design*, pages 267–292, 1999.

[58] H. Simon. *The sciences of artificial*. MIT Press, 1996. ISBN 9780585360102.

[59] A. G. Taylor. *SQL For Dummies*. For Dummies, 7th edition, 2010. ISBN 9780470557419. URL https://www.oreilly.com/library/view/sql-for-dummies/9780470557419/.

[60] Techopedia Inc. Interoperability, 2021. URL https://www.techopedia.com/definition/631/interoperability. Accessed in 02/01/2021.

[61] The GraphQL Foundation. A query language for your api, 2020. URL https://graphql.org/. Accessed in 07/09/2020.

[62] J. B. Vasconcelos, R. Henriques, and A. Rocha. Modelo para o desenvolvimento de sistemas de apoio à decisão clínica para a prática da medicina baseada na evidência. In *Anais do X Congresso Brasileiro de Informática em Saúde (CBIS 2006)*, pages 1162–1167, 2006.

[63] M. Wasson. Single-page applications: Build modern, responsive web apps with asp.net, 2013. URL https://docs.microsoft.com/en-us/archive/msdn-magazine/2013/november/asp-net-single-page-applications-build-modern-responsive-web-apps-with-as Accessed in 25/10/2020.

[64] R. Wieringa. Design science as nested problem solving. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, 8. ACM, 2009. ISBN: 978-1-60558-408-9.

[65] R. Wieruch. *The Road to React: Your journey to master plain yet pragmatic React.js*. Independently Published, 2017. ISBN 9781720043997. URL https://books.google.pt/books?id=RRLmDwAAQBAJ.

[66] R. Wieruch. *The Road to GraphQL: Your journey to master pragmatic GraphQL in JavaScript with React.js and Node.js*. INDEPENDENTLY PUBLISHED, 2018. ISBN 9781730853937. URL https://books.google.pt/books?id=axLmDwAAQBAJ.

[67] R. Wieruch. React libraries in 2020, 2020. URL https://www.robinwieruch.de/react-libraries. Accessed in 07/09/2020.

[68] Wikipedia contributors. Oracle database, 2020. URL https://en.wikipedia.org/wiki/Oracle_Database#cite_note-5. Accessed in 09/09/2020.

[69] Wikipedia contributors. Swot analysis, 2021. URL https://en.wikipedia.org/w/index.php?title=SWOT_analysis&oldid=885318472. Last accessed in 10/02/2021.

[70] J. K. Zhang and W. Xu. Web service-based healthcare information system (wshis): A case study for system interoperability concern in healthcare field. In *2006 International Conference on Biomedical and Pharmaceutical Engineering*, pages 588–594, 2006.

Part III

APPENDICES

A

PLATFORM TO VALIDATE THE DENTAL MEDICINE ACTS THROUGH ADSE REQUIREMENTS

In this appendix the functional and non-functional requirements of the web platform to validate the dental medicine acts through ADSE are presented.

A.1 FUNCTIONAL REQUIREMENTS

Requirement no.: 1
Requirement type: Functional.
Use case: Consult patient's information.
Description: The dentist should be able to consult some patient's hospital data.
Reason: This knowledge will help the dentist validate the specifics of the episode, if he need.

Requirement no.: 2
Requirement type: Functional.
Use case: Consult patient's information.
Description: The dentist should be able to consult the patient's ADSE beneficiary number.
Reason: Since the patient's ADSE beneficiary number is necessary to the validation process, the dentist must be able to consult it.

Requirement no.: 3
Requirement type: Functional.
Use case: Consult patient's information.
Description: The dentist should be able to consult if the patient has ADSE rights on that day.
Reason: Since the application should only work for patients with ADSE rights, this verification is needed.

Requirement no.: 4
Requirement type: Functional.
Use case: Complete the acts' information.

Description: The dentist should be able to complete the general information about the set of acts.
Reason: This information is needed to validate the set of acts through the ADSE web service.

Requirement no.: 5
Requirement type: Functional.
Use case: Complete the acts' information.
Description: The dentist should be able to complete the specific information about each act.
Reason: This information is needed to validate each act through the ADSE web service.

Requirement no.: 6
Requirement type: Functional.
Use case: Validate the medical acts.
Description: The dentist should be able to send all the acts' information to the ADSE web service.
Reason: Without this stage it is impossible to validate the acts through the ADSE.

Requirement no.: 7
Requirement type: Functional.
Use case: Consult ADSE web service response.
Description: The dentist should be able to see the positive returned ADSE web service response.
Reason: This information is important to the dentist since he needs to know if the patient is able to do the medical acts or not.

Requirement no.: 8
Requirement type: Functional.
Use case: Consult ADSE web service response.
Description: The dentist should be able to see potential errors from the returned ADSE web service response.
Reason: This information is important to the dentist since he can know why the patient is not able to do some medical acts.

Requirement no.: 9
Requirement type: Functional.
Use case: Consult ADSE web service response.
Description: The dentist should be able to repeat the validation process through the ADSE web service.
Reason: The dentist can fill the acts' information badly or may want to test other acts.

A.2 NON-FUNCTIONAL REQUIREMENTS

In this section, the non-functional web platform requirements collected are presented, each describing a restriction imposed on the system.

Usability Requirements

Requirement no.: 10
Requirement type: Usability requirement.
Description: The application shall use terms appropriate to the context in which the system operates.
Reason: The application must use terms and vocabulary appropriate to the context in which it is inserted to facilitate its understanding by dentists.

Requirement no.: 11
Requirement type: Usability requirement.
Description: The application should be easy to use.
Reason: The application should be user-friendly for all dentists.

Performance Requirements

Requirement no.: 12
Requirement type: Performance requirement.
Description: The application should provide answers to the dentist requests in a fast way.
Reason: The interaction of the dentist with the application should be smooth so that the use of the platform is pleasant.

Requirement no.: 13
Requirement type: Performance requirement.
Description: The application should contact web services instantaneously.
Reason: The interaction between the application and the different web services should be fast and immediate.

Requirement no.: 14
Requirement type: Performance requirement.
Description: The application should always be available.
Reason: The application should always be available so as not to impede its use by dentists.

Portability Requirements

Requirement no.: 15
Requirement type: Portability requirement.
Description: The application should work in different browsers.
Reason: The application should work in different browsers, mainly in the hospital's computers with older browsers.

B

PLATFORM TO INVOICE THE HOSPITAL ACTS THROUGH ADSE REQUIREMENTS

In this appendix the functional and non-functional requirements of the web platform to validate the dental medicine acts through ADSE are presented.

B.1 FUNCTIONAL REQUIREMENTS

Requirement no.: 1
Requirement type: Functional.
Use case: Consult patient's information.
Description: The receptionist should be able to consult some patient's hospital data.
Reason: This knowledge will help the receptionist validate the patient's information if he need.

Requirement no.: 2
Requirement type: Functional.
Use case: Consult patient's information.
Description: The receptionist should be able to consult the episode information.
Reason: This knowledge will help the receptionist validate the specifics of the episode, if he need.

Requirement no.: 3
Requirement type: Functional.
Use case: Consult patient's information.
Description: The receptionist should be able to consult the patient's ADSE beneficiary number.
Reason: Since the patient's ADSE beneficiary number is necessary to the validation process, the receptionist must be able to consult it.

Requirement no.: 4
Requirement type: Functional.
Use case: Consult patient's information.

Description: The receptionist should be able to consult if the patient has ADSE rights on that day.
Reason: Since the application should only work for patients with ADSE rights, this verification is needed.

Requirement no.: 5
Requirement type: Functional.
Use case: Patient admission via citizen card.
Description: The receptionist should be able to admit the patient into the hospital via his citizen card.
Reason: This process simplify the hospital's administrative and accounting teams work since they won't need to send the invoice PDF to the ADSE.

Requirement no.: 6
Requirement type: Functional.
Use case: Patient admission via citizen card.
Description: The receptionist should be able to know if the patient admission via citizen card went well or if there were any errors.
Reason: Since the patient admission into the hospital is a required process, the receptionist must know if everything goes well.

Requirement no.: 7
Requirement type: Functional.
Use case: Manual patient admission.
Description: The receptionist should be able to admit the patient into the hospital when they don't have a citizen card.
Reason: If some patient doesn't have a citizen card, it is important to do his admission into the hospital and save his information on the hospital's database.

Requirement no.: 8
Requirement type: Functional.
Use case: Cancel patient's admission.
Description: The receptionist should be able to cancel the patient admission.
Reason: If some patient, for some reason, does not perform the medical acts or wants to leave the hospital, this information needs to be saved.

Requirement no.: 9
Requirement type: Functional.
Use case: Invoice the medical acts.
Description: The receptionist should be able to choose the doctor that will perform the medical acts.
Reason: There is a gap in the hospital's databases and for some episodes, it is impossible to know the doctor's

number on the Portuguese Medical Association. Since it is essential to invoice through ADSE, the receptionist should be able to insert it.

Requirement no.: 10
Requirement type: Functional.
Use case: Invoice the medical acts.
Description: The receptionist should be able to send all the acts' information to the ADSE web service.
Reason: Without this stage it is impossible to invoice the acts through the ADSE.

Requirement no.: 11
Requirement type: Functional.
Use case: Consult ADSE web service response.
Description: The receptionist should be able to see the positive returned ADSE web service response.
Reason: This information is important to the receptionist since he needs to know if the invoice process goes well or not.

Requirement no.: 12
Requirement type: Functional.
Use case: Consult ADSE web service response.
Description: The receptionist should be able to see the invoice PDF.
Reason: The receptionist should be able to see the invoicing information about the patient and the medical acts, such as the price that the patient must pay.

Requirement no.: 13
Requirement type: Functional.
Use case: Print the invoice PDF.
Description: The receptionist should be able to print the invoice PDF.
Reason: Since the patient who has been admitted manually must sign the PDF invoice, the receptionist should be able to print it.

B.2 NON-FUNCTIONAL REQUIREMENTS

In this section, the non-functional web platform requirements collected are presented, each describing a restriction imposed on the system.

Usability Requirements

Requirement no.: 14
Requirement type: Usability requirement.
Description: The application shall use terms appropriate to the context in which the system operates.
Reason: The application must use terms and vocabulary appropriate to the context in which it is inserted to facilitate its understanding by the reception team.

Requirement no.: 15
Requirement type: Usability requirement.
Description: The application should be easy to use.
Reason: The application should be user-friendly for all receptionists.

Performance Requirements

Requirement no.: 16
Requirement type: Performance requirement.
Description: The application should provide answers to the receptionist requests in a fast way.
Reason: The interaction of the receptionist with the application should be smooth so that the use of the platform is pleasant.

Requirement no.: 17
Requirement type: Performance requirement.
Description: The application should contact web services instantaneously.
Reason: The interaction between the application and the different web services should be fast and immediate.

Requirement no.: 18
Requirement type: Performance requirement.
Description: The application should read the information from the patient's citizen card quickly.
Reason: The interaction between the application and the card reader should be fast and immediate.

Requirement no.: 19
Requirement type: Performance requirement.
Description: The application should always be available.
Reason: The application should always be available so as not to impede its use by the receptionist.

This work is financed by the Associação Universidade-Empresa para o Desenvolvimento - TECMINHO, within project PRM-Patient Relationship Management.