

**Universidade do Minho**

Escola de Engenharia

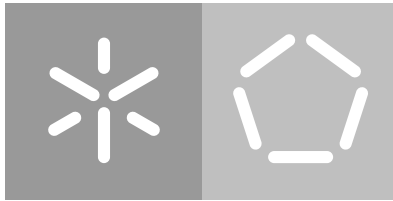
Departamento de Informática

Patrícia Alexandra Soares Loreto

## **Progressive Web Development**

**A case study in Obstetrics**

December 2018



**Universidade do Minho**

Escola de Engenharia

Departamento de Informática

Patrícia Alexandra Soares Loreto

## **Progressive Web Development**

### **A case study in Obstetrics**

Master dissertation

Master Degree in Medical Informatics

Dissertation supervised by

**Professor José Manuel Ferreira Machado**

**Professor Hugo Daniel Abreu Peixoto**

December 2018

---

## ACKNOWLEDGEMENTS

---

I would like to start by thanking both my supervisor and cosupervisor, Professor José Machado and Professor Hugo Peixoto, for all the support and time provided during this past year.

I also want to thank Professor António Abelha and Dr Jorge Braga for the opportunity to work in such a challenging and fulfilling project.

To my project partners and, above all, friends, Francisca Fonseca and Catarina Peixoto, I want to say that without you this journey would not have been so pleasant. Thank you for all your support, it was crucial.

A special thanks to Cláudia Brito for always encouraging me and being such a good listener and to Ana Rita Morais for always being present.

To all the friends I have made during the past five years, thank you for all the patience, support, motivation and moments.

Last but not least, all of this would not have been possible without my family, specially my mother. Thank you for all your unconditional support and encouragement.

*There are places I'll remember  
All my life, though some have changed  
Some forever, not for better  
Some have gone and some remain*

*All these places have their moments  
With lovers and friends I still can recall  
Some are dead and some are living  
In my life, I've loved them all*

*(In my life, Beatles)*

---

## ABSTRACT

---

In the past years, with the evolution of technology and the rise of the Internet, Personal Health Records appeared. These records are maintained by patients and turn them into a more active stakeholder in their own health management. They can be used to record medical parameters or give useful information to the patient, among others.

However, this was not the only result from this evolution. With the Internet, more medical information became available to everyone, most of it not from reliable sources, which brought additional problems.

The obstetrics field was also impacted by these changes. This, combined with fact that a pregnancy is a delicate medical condition, brought the necessity to find solutions for this case.

This dissertation aims to develop a Personal Health Record for pregnant women that provides reliable information, and that has a set of features they find useful.

With this goal in mind, a literature review on the technologies and methodologies that are used nowadays and on the use of technology by pregnant women is made.

Then, all the development process is presented, as well as the final result. This process was supervised by a medical institution, which had the advantage of facilitating the process of having feedback from pregnant women and it also provided all the medical information displayed in the application.

The final result is an application that has a wide range of useful features, provides trustworthy information, is available in all devices and that was developed using modern technologies.

---

## RESUMO

---

Nos últimos anos, com a evolução das tecnologias e o aparecimento da Internet, surgiram *Personal Health Records*. Estes registos são mantidos pelos pacientes e tornam-os uma parte mais ativa na gestão da sua saúde. Eles podem ser usados para registar parâmetros médicos ou dar informações úteis ao paciente, entre outros.

No entanto, este não foi o único resultado desta evolução. Com a Internet, mais informação médica ficou disponível para todas as pessoas, sendo que a maioria não provém de fontes de confiança, o que trouxe problemas adicionais.

A área da obstetrícia também foi influenciada por estas mudanças. Isto, combinado com o facto de uma gravidez ser uma condição médica delicada, trouxe a necessidade de se encontrar novas soluções para este caso.

Esta dissertação pretende desenvolver um *Personal Health Record* para grávidas que dê informação de confiança, e que tenha um conjunto de funcionalidades que elas considerem úteis.

Com este objetivo em mente, é feita uma revisão da literatura sobre as tecnologias e metodologias usadas atualmente e sobre o uso das tecnologias pelas grávidas.

Depois, todo o processo de desenvolvimento é apresentado, assim como o resultado final. Este processo foi supervisionado por uma instituição de saúde, o que teve a vantagem de facilitar o processo de recolha de opiniões de grávidas e também de fornecer toda a informação médica disponibilizada na aplicação.

O resultado final é uma aplicação que apresenta um grande conjunto de funcionalidades úteis, que dá informação de confiança, está disponível em todos os dispositivos e que foi desenvolvida usando tecnologias modernas.

---

## CONTENTS

---

1	INTRODUCTION	1
1.1	Background	1
1.2	Motivation	2
1.3	Aim	3
1.4	Structure	4
2	STATE OF THE ART	6
2.1	Web development	6
2.1.1	Hypertext Transfer Protocol (HTTP)	7
2.1.2	Client side development	8
2.1.3	Server side development	10
2.1.4	Software stacks	11
2.2	Mobile Development	12
2.3	Use of technology by pregnant women	14
2.3.1	Personal Health Records for pregnant women	14
2.3.2	Online searching habits	16
3	METHODOLOGIES AND TECHNOLOGIES	18
3.1	Research methodology	18
3.2	Technologies	19
3.2.1	React	20
3.2.2	Node.js and Express	22
3.2.3	MongoDB	24
3.2.4	KeystoneJS	26
3.3	RESTful web services	28
3.4	Progressive Web Apps	30
3.4.1	Service worker	31
3.4.2	Offline Support	33

3.5	Evaluation methodologies	34
3.5.1	Lighthouse reports	34
3.5.2	SWOT Analysis	35
3.5.3	Usability Testing	36
4	CASE STUDY	38
4.1	Requirements analysis	38
4.2	Architecture	42
4.3	Developed system	43
4.3.1	Database	44
4.3.2	Web server	47
4.3.3	User Interface	49
4.3.4	Progressive Web App	61
4.3.5	Administration panel	63
4.4	Discussion	65
5	PROOF OF CONCEPT	67
5.1	Lighthouse reports	67
5.2	SWOT Analysis	69
5.3	Usability test	71
6	CONCLUSION	73
6.1	Contributions	73
6.2	Future work	75
A	PUBLICATIONS	87
A.1	Improving Maternity Care with Business Intelligence	87
A.2	Step Towards Progressive Web Development in Obstetrics	88
A.3	Predicative Vagueness in Lung Metastases in Soft Tissue Sarcoma Screening	88
A.4	Predicting Low Birth Weight Babies through Data Mining	89

---

## LIST OF FIGURES

---

Figure 1	Evolution of the interest over time for React and AngularJS all over the world. The displayed data comes from the Google Trends tool. The score is calculated according to the amount of times the terms are searched using the Google search engine, being 100 the biggest one and 0 the lowest [49].	20
Figure 2	Hello World example using the JavaScript <i>Application Programming Interface (API)</i> .	21
Figure 3	Hello World example using JSX.	21
Figure 4	Synchronous file reading operation [61].	22
Figure 5	Asynchronous file reading operation [61].	23
Figure 6	Basic Express app configuration [67].	24
Figure 7	MongoDB document example.	25
Figure 8	MongoDB schema example.	26
Figure 9	Administration <i>User Interface (UI)</i> example provided by KeystoneJS [80].	28
Figure 10	<i>Strengths, Weaknesses, Opportunities, Threats (SWOT)</i> diagram structure.	36
Figure 11	Sitemap for the area that does not require authentication.	41
Figure 12	Sitemap for the authenticated area.	42
Figure 13	Architecture of the developed platform.	43
Figure 14	Mongoose schema where the information about the pregnancy is saved.	45
Figure 15	Mongoose schema where the <i>Frequently Asked Questions (FAQs)</i> content is saved.	46
Figure 16	Global Home Page.	50



Figure 17	<i>Perguntas Frequentes</i> page.	50
Figure 18	<i>Perguntas Frequentes</i> page with the tab expanded.	51
Figure 19	<i>Legislação</i> page.	51
Figure 20	<i>Criar conta</i> page.	52
Figure 21	<i>Criar conta</i> page with an error message. In this case, the user did not insert a valid email so a error message shows up when the user tries to create an account.	53
Figure 22	<i>Entrar na conta</i> page.	54
Figure 23	<i>Dados pessoais</i> page.	55
Figure 24	<i>Ficha Pessoal</i> page.	55
Figure 25	<i>Ficha Pessoal</i> page with an open dialog modal where the woman can edit her personal data.	56
Figure 26	<i>Ficha Pessoal</i> page on mobile.	57
Figure 27	<i>Documentos</i> page.	58
Figure 28	Dialog modal to add a document.	58
Figure 29	Message that shows up when the woman clicks the delete button in the <i>Documentos</i> page.	59
Figure 30	<i>Marcações</i> page.	60
Figure 31	Dialog modal to add an appointment.	60
Figure 32	<i>Marcações</i> page when an item in one of the lists is clicked.	61
Figure 33	<i>Documentos</i> page when there is no Internet connection. When the user is offline, all buttons that edit or add data are disabled and a warning is presented. Still, the data that was available when the user was online for the last time is displayed.	63
Figure 34	Initial page of the administration panel.	64
Figure 35	Page where the FAQs content is managed in the administration panel.	64
Figure 36	Modal dialog to add a new FAQ.	65
Figure 37	Lighthouse report main results when the application is not cached, that is, when a user first accesses it.	68

Figure 38	Lighthouse report main results when the application is already cached.	68
-----------	--	----

---

## LIST OF TABLES

---

Table 1	Some actions and their respective source code using Mon- goose [75, 76]	27
Table 2	Examples of companies that started using <i>Progressive Web Apps (PWAs)</i> and the main highlights of this change [86, 87, 88, 89, 90, 91, 92, 93]	32
Table 3	Routes defined in the <i>Representational State Transfer (REST)</i> API	48

---

## ACRONYMS

---

### A

AJAX Asynchronous JavaScript and XML. 9

API Application Programming Interface. vi, ix, 9, 13, 20, 21, 23, 25, 26, 28, 30, 42, 43, 47, 48, 65, 66, 69, 73

### B

BSG Boletim de Saúde da Grávida. 38, 40–42

### C

CHP Centro Hospitalar do Porto. 3

CMIN Centro Materno Infantil do Norte. 3, 4, 38, 39, 74, 75

CRUD Create, Read, Update and Delete. 28

CSS Cascading Style Sheets. 9, 12, 30, 48, 61

### D

DOM Document Object Model. 9, 20

DSR Design Science Research. 18, 19

DSS Decision Support Systems. 1, 15

### E

EHR Electronic Health Record. 1

### F

FAQ Frequently Asked Question. vi, vii, 39, 40, 42, 44, 46–49, 62–65

## H

HIS Health Information Systems. 1, 2, 4, 15

HL7 Health Level Seven. 1

HTML Hypertext Markup Language. 6, 7, 9, 12, 30

HTTP Hypertext Transfer Protocol. 6–8, 11, 23, 24, 28–30, 47

HTTPS Hypertext Transfer Protocol Secure. 31, 68

## I

IS Information Systems. 18, 67

IT Information Technology. 18, 19

## J

JSON JavaScript Object Notation. 9, 11, 24, 28, 29

## M

MVC Model-View-Controller. 10, 12, 20, 43

## P

PHR Personal Health Record. 1–4, 14–16, 38, 66, 73

PWA Progressive Web App. ix, 13, 18, 30–33, 35, 40, 43, 44, 48, 61, 65, 67, 68, 70,  
74

## R

REST Representational State Transfer. ix, 11, 18, 28–30, 42, 43, 47, 48, 66, 69, 73

## S

SDK Software Development Kit. 12

- SEO Search Engine Optimisation. 21, 35
- SNOMED-CT Systematized Nomenclature of Medicine - Clinical Terminology. 1
- SOA Service Oriented Architecture. 11
- SOAP Simple Object Access Protocol. 11, 29
- SQL Structured Query Language. 11
- SWOT Strengths, Weaknesses, Opportunities, Threats. vi, 4, 5, 18, 34–36, 67, 69, 74
- U
- UE Usability Evaluation. 36, 37
- UI User Interface. vi, 10, 26, 28, 36, 49, 66, 69, 73
- URL Uniform Resource Locator. 6, 7, 23, 24, 28–30, 47, 48, 61, 62
- UX User Experience. 9, 12, 13, 33, 66
- W
- WWW World Wide Web. 6
- X
- XML Extensible Markup Language. 9, 11, 28, 29

---

## INTRODUCTION

---

This document describes the development of a *Personal Health Record (PHR)* for the support of pregnant women. This dissertation was carried out during the last year of the Integrated Master in Biomedical Engineering (Medical Informatics) at the University of Minho.

This chapter is divided in four parts. In the first section (1.1), the most important concepts regarding *Health Information Systems (HIS)* are overviewed. Then, both the motivation behind this project (1.2) and the main objectives (1.3) are explained. Finally, the structure this document follows is presented (1.4).

### 1.1 BACKGROUND

The rise of **HIS** changed the way healthcare is provided. When they first appeared, **HIS** were used to store and retrieve data from financial and administrative services in medical institutions. Throughout the years, with the development of new technologies, such as the Internet and smaller and more affordable computers, they have changed drastically and started to record medical data, be distributed and communicate between each other [1]. Another important advance was the adoption of *Electronic Health Records (EHRs)*. These records save all the information related to the patient health, include *Decision Support Systems (DSS)* that help professionals and data are saved according to interoperability standards (e.g. *Health Level Seven (HL7)* or *Systematized Nomenclature of Medicine - Clinical Terminology (SNOMED-CT)*) [1, 2].

Nowadays, another shift is happening in the HIS world with the adoption of PHRs by patients [3]. PHRs empower patients by including them in their own health management. These systems are maintained by the patients and allow them to receive individualised content according to their medical history and to record medical indicators (e.g. glucose levels). With the recorded information, PHRs can alert patients when abnormal values are detected or even health professionals if the record is connected to the hospital system. This is particularly important for patients with chronic diseases that need extensive monitoring. PHRs can have other functionalities as well, such as providing a way of saving medical imaging exams, analysis or scheduled appointments and giving alerts for upcoming appointments [1, 3].

The evolution of technologies also impacted the way people have access to information. Information is spread in the Internet and anyone can publish it, even if it is not correct or accurate. This impacted particularly the medical field, because this field deals with delicate information and people are not usually able to evaluate the quality of the information they find [4, 5]. It can lead to anxiety and worries as well, specially if the patient already has a predisposition for such problems [6]. However, online health information has the potential to increase patients' engagement and decrease health literacy disparities [5].

## 1.2 MOTIVATION

The adoption of PHRs and the increase in the amount of medical information available online also impacted the Obstetrics field.

A pregnancy is a delicate medical condition that lasts about 40 weeks. Therefore, extensive monitoring is necessary and problems need to be detected as soon as possible. As a consequence, it is advisable for pregnant women to monitor certain parameters. With this in mind, there are several PHRs available that track the woman's weight, blood pressure, among other parameters. However, there are systems that provide other functionalities, such as delivering relevant medical information, scheduling appointments or saving ultrasounds [7, 8].



As for searching medical information online, most pregnant women look for information in the Internet and the number increases if it is a first pregnancy. Although they usually find information reliable and useful, they do not speak with their healthcare professionals about the information they encounter, which may lead to misinformation and worries. This could be solved if the care givers provided reliable information to women at the right moment. The topics that women search for the most in the Internet are related to fetal development and nutrition, but women have also shown interest in information related to the use of medicaments during pregnancy [9].

This dissertation will focus on the development of an electronic PHR for pregnant women. This project was developed in partnership with the *Centro Materno Infantil do Norte (CMIN) (Centro Hospitalar do Porto (CHP))*, a Portuguese maternity. This relationship was beneficial for all the parts involved and provided the chance to test the developed solution in a real environment.

### 1.3 AIM

Given what was previously presented, this project aims to develop a PHR to support pregnant women. To achieve this goal, four research questions were proposed:

- *Research Question 1:* Is the developed platform scalable?
- *Research Question 2:* Does the developed system reach as many pregnant women as possible?
- *Research Question 3:* Does it offer a set of features pregnant women find useful?
- *Research Question 4:* Does the proposed solution offer reliable information to the pregnant women?

The main goals necessary to answer these questions and the steps needed to achieve them are the following:

1. Develop an application for pregnant women:
  - Define the typology of the application (e.g. web application or mobile application);
  - Gather the necessary requirements;
  - Decide which technology stack to use;
  - Write the content for the application (this will be done with the support of the [CMIN](#));
  - Build the application;
  - Develop an administration panel to maintain the displayed static content in the main application.
  
2. Evaluate the application:
  - Performance analysis;
  - [SWOT](#) analysis;
  - Usability test.

The technologies used to develop this application were mainly React for the front end, Node.js and Express for the back end, MongoDB for the database and KeystoneJS for the administration panel. These technologies are reviewed in section [3.2](#).

## 1.4 STRUCTURE

This document is organised in six chapters. In this chapter, important concepts and subjects, such as [HIS](#), [PHRs](#) and online health information, were introduced and the motivation behind this dissertation presented. Then, the main objectives that this dissertation aims were exposed. The remaining document is divided in the following chapters:

- [STATE OF THE ART](#) - in this chapter, important concepts necessary to understand the work developed are explained. These concepts are divided in

three main subjects: Web Development, Mobile Development and use of technology by pregnant women;

- **METHODOLOGIES AND TECHNOLOGIES** - this chapter overviews the research and evaluation methodologies that were chosen, as well as the main technologies, web service architecture and application type that were used. It is divided in five sections: **Research methodology**, **Technologies**, **RESTful web services**, **Progressive Web Apps** and **Evaluation methodologies**;
- **CASE STUDY** - in this chapter the developed application is presented. This exposition includes its requirements, the architecture of the system, the components that make up this system and the final result. In the end, the developed system is analysed;
- **PROOF OF CONCEPT** - after presenting the developed application it is evaluated in this chapter. The application is evaluated using the reports from the Lighthouse tool and by performing a **SWOT** analysis. It is also proposed a usability test to evaluate the application;
- **CONCLUSION** - in this chapter, a final balance of the developed work is exposed and prospects for future work are presented.

Before resuming this document, it is important to explain that, since this project was developed with two other students, each one of us was responsible for different parts of the application. This division is explained in **CASE STUDY**, but it has influence in the topics that are overviewed in the next two sections.

---

## STATE OF THE ART

---

In this chapter, it is made a literature review of the state of the art of concepts and technologies that are necessary to understand the work developed. This chapter is divided in three sections: Web development (2.1), Mobile development (2.2) and use of technology by pregnant women (2.3).

In section 2.1, important concepts related to Web development are presented as well as the main technologies in this field. In section 2.2 an overview of the current trends in Mobile development is made. Finally, in section 2.3, concepts and relevant studies related to the the use of technology by pregnant women are presented.

### 2.1 WEB DEVELOPMENT

The *World Wide Web (WWW)* or Web is a hypertext system that is a subset of the Internet. It was proposed in 1990 and its core elements are:

- A unique identifier for resources called *Uniform Resource Locator (URL)*;
- A protocol that describes how requests and responses behave called *Hypertext Transfer Protocol (HTTP)*;
- A web server that responds to **HTTP** requests;
- A language to publish documents called *Hypertext Markup Language (HTML)*;
- A browser that can make **HTTP** requests from **URLs** and that displays the **HTML** it receives in the response [10].

In the beginning of the Web, websites were **static websites**. Static websites are web pages that are published and periodically updated. Users can read them, but cannot provide any type of input, and these websites' pages are the same for all users, there is no type of personalised content. The main technology behind these websites in the early days was [HTML](#) [10].

Through out the years, websites became more complicated and began to use programs in the server side to generate dynamic content. To retrieve dynamic content, these programs would get information from databases, for example. These websites are called **dynamic websites** and [HTML](#) stopped being enough to develop them. Now, developers also need to have programming knowledge [10].

In the 2000s, a new era began: **Web 2.0**. For users, websites are more interactive and people can not only use them, but also contribute (e.g. Wikipedia or YouTube). For developers, this shift meant that the programming logic stopped being exclusive to the server side and started existing as well in the browser. With this change, developers need to know **JavaScript** as well, since this programming language runs in browsers [10].

### 2.1.1 Hypertext Transfer Protocol (HTTP)

In the Web, there are two main stakeholders: the clients and the servers. By **client**, one means client machines (e.g. desktop computers or smart phones) that request a resource to the server. The **server** is what hosts web applications, stores data and answers the requests. The [HTTP](#) defines how the transmission of hypermedia documents (e.g. [HTML](#)) should proceed between the server and the client [10, 11]. The server is waiting for requests. When the client sends a request with the [URL](#) for the resource they need, the server sends a response. The response has a response code, headers and a message (optional) [10].

**Headers** are sent both in the request and in the response and they have information about the transaction [10, 12]. Request headers have information about the client, such as its operating system, browser and the types of media it ac-

cepts, and it specifies the host that we want the response from. Response headers include information about the server (e.g. the operating system) and the data that is being sent. It tells the client the length of the data, when it was last modified (this is important for caching), its type and encoding [10].

This protocol also defines a set of **request methods** to define the action to be performed. The most important ones for this dissertation are:

- **GET** - request a resource;
- **POST** - submit new data;
- **PUT** - replace the resource with the data from the request;
- **DELETE** - delete the specified resource [13].

To specify the state of the request, the **HTTP** has a set of **response codes**. These codes are sent in the response header. There is a wide range of response codes, but the most important ones in this context are:

- **200: OK** - the request was successful;
- **304: Not Modified** - if the cache related headers are sent appropriately, the response might not send any data, because there is not a newer resource;
- **400: Bad Request** - there is something wrong with the request headers or response;
- **401: Unauthorised** - the requested resource is protected and the credentials needed to request it were not sent or were not correct;
- **404: Not Found** - the requested resource was not found;
- **500: Internal Server Error** - the server encountered an error that made it impossible to complete the request [14].

### 2.1.2 Client side development

Client side development means developing scripts that run locally in the client. There are several programming languages for this kind of development, such as

Java, Flash and JavaScript. However, JavaScript is the most popular one, because it works on most browsers. JavaScript combined with [HTML](#) and *Cascading Style Sheets (CSS)* are the core technologies for front end development [10, 15].

Client side scripting reduces the amount of work performed in the server, because some tasks can be done in the client side, and improves the *User Experience (UX)* by answering to events faster than the server and by interacting with the [HTML](#) sent by the server. On the other side, one must be cautious when developing with JavaScript, because it might not be enabled in the client browser, there might be differences between browsers and JavaScript-heavy applications may be difficult to maintain and debug [10].

JavaScript became popular in the 2000s, because of the *Asynchronous JavaScript and XML (AJAX)* websites. In this approach, websites are developed using JavaScript, [HTML](#), [CSS](#), *Extensible Markup Language (XML)*, the *Document Object Model (DOM)*, among other technologies. It is important to refer that back then [XML](#) was the only data format to transport data, but nowadays *JavaScript Object Notation (JSON)* is the most used [10, 16]. JavaScript can be incorporated in [HTML](#) in various ways:

- It can be included directly in some [HTML](#) elements (e.g. in a button to perform an action when the user clicks it);
- The code can be inside a script element in the [HTML](#);
- A script element can have a link to an external JavaScript file [10].

To access elements and attributes in [HTML](#), JavaScript uses the [DOM API](#). In this model the [HTML](#) document is structured in a tree view. Each node in the tree is a part of the [HTML](#) (element, text or attribute). This tree is saved in a object and these elements are accessed as if they were properties of that object. There are methods as well to access and change the object (create a new element, for example). The [DOM](#) also allows the change of elements or the addition of new styles without dealing directly with the [HTML](#) [10, 17]. To perform actions and react to user actions, events are used. Events are triggered by actions (the user clicking a button, for example) and are handled by functions (a function

that submits a form, for example) that perform an action (the sending of form data to the server, for example) [10].

Nowadays, JavaScript frameworks that apply the *Model-View-Controller (MVC)* presentation pattern have become popular, because they give a structured way of developing web applications by separating the data representation (model) from the data presentation (view) [10, 18]. Examples of these frameworks are AngularJS and Backbone [10].

The *MVC* pattern divides the application parts into three categories: model, view and controller. The model represents the data, the view is the visual part of the *UI* and the controller is the bridge between the model and the view. The view notifies the controller for interactions and sends to the controller input from the user. The model notifies the views when changes happen and the views update. The controller listens to events from the user in the views and updates the model [18, 19].

### 2.1.3 *Server side development*

As previously explained, the server side is responsible for hosting the web application, storing data and taking care of the security. Server side scripts can access the resources that are in the server's scope. These resources are databases for data storing, Web Services that communicate with the client and Software Applications that perform additional tasks (e.g. e-mail services). Nowadays, some of the most common scripting technologies for server-side development are PHP, Django, Node.js and Ruby on Rails [10].

**Databases** are crucial for Web Development to store data which is what brings dynamism to websites. The visual look of websites is usually the same and the content provided by databases is what changes. They provide a structured and optimised way of inserting, updating, querying, and getting data. The main advantage of using databases over other alternatives, such as files, is that databases provide a set of rules that ensure data integrity and reduce data duplication [10].



There are two main types of databases: relational databases and noSQL databases. In relational databases, data is stored in tables. Each table represents an entity and is composed by records (rows). In a table all records have the same number of fields (columns). Each record has a primary key which is a field used to uniquely identify records. Tables are connected between each other with foreign keys. Foreign keys are fields that are the primary key of another table. Relationships between tables can be one-to-many, one-to-one and many-to-many and they express the relationships between the entities. To store and manipulate data, *Structured Query Language (SQL)* is used. MySQL, Oracle and PostgreSQL are examples of relational databases [10].

NoSQL databases appeared later has a solution for highly scalable applications. Unlike relational databases, data is not save in a predefined schema which allows continuous evolution over time with the addition of new fields. Since this type of database does not define a type of storage, there are several approaches to store data. It can be column-oriented, document store, key value store or graph, among others. Examples of noSQL databases are MongoDB (document store), Cassandra (column-oriented) and Redis (key value store) [20].

As for **Web Services**, they provide a standardised way of connecting and providing communication between two software applications. Web Services use the **HTTP** protocol to communicate through the Internet and **XML** or **JSON** to encode the data sent. Since they use universal standards, they work on a wide range of platforms. Web Services can implement a *Service Oriented Architecture (SOA)* which allows a better integration of different software applications. The most common architectures used to implement SOAs in Web Services are *Simple Object Access Protocol (SOAP)* and **REST** [10].

#### 2.1.4 Software stacks

A software stack is a set of technologies that together are enough to build a complete application. The oldest and more popular one is the **LAMP stack**.

LAMP originally stood for the Linux operating system, the Apache web server, MySQL for the database and PHP as scripting language [21].

However, in the last years the **MEAN stack** has gained popularity. The MEAN stack consists in MongoDB as the database, Express.js as a back end framework, Angular.js for the front end and Node.js for the web server. All these technologies have in common the fact that they are based in JavaScript. This is an advantage since it enables the development of a complete application with only one scripting language [21, 22].

More recently the **MERN stack** appeared. The difference between the MERN stack and the MEAN stack is in the front end main technology. The MERN stack uses React, a library developed by Facebook, instead of Angular. The fact that React is a library and not a framework as Angular makes the development more flexible. Angular implements the whole **MVC** pattern, but React only implements the view and the remaining parts and how they are connected are implemented by the developer [22]. In this project the MERN stack was the software stack chosen.

## 2.2 MOBILE DEVELOPMENT

Developing for mobile devices has its own specificities. Usually mobile development strategies are divided into two main groups: native development and web-based development [23].

In **native development**, mobile applications are developed with the target platform in mind and the tools that are used are specific of that platform. For example, to develop for Android devices one must use Java and the Android *Software Development Kit (SDK)*, while iOS applications are written in Objective-C through XCode. Since the code is developed for each platform, it leads to heavier development and higher maintenance costs. However, **UX** and performance are usually better [23, 24, 25].

On the other side, **web-based development** approaches use standard web technologies, such as **HTML**, **CSS** and JavaScript, to develop applications that

can be used in several platforms [23, 25]. Web-based technologies can be divided in three categories:

1. **Mobile Web Apps** - these apps are websites optimised for mobile use. They are accessed in browser apps on the user's device. Since standard technologies are used, the UX is the same no matter the platform which leads to less costs and maintenance. Even though these apps can access some resources from the device, such as the camera or the microphone, they fail at accessing other features that native apps can access. They fail as well in delivering complex animations and graphics and are not available in app stores [23, 26];
2. **Hybrid Mobile Apps** - these apps use standard technologies as well, but they use a wrapper that makes them available in any platform. This wrapper is achieved by using a hybrid development framework (e.g. PhoneGap). The framework adds a native wrapper to the application that provides an API to match the requests to their equivalents in the corresponding native platform. However, not all native APIs are supported. This approach allows code reusability which in turn decreases development time and maintenance [23, 27];
3. **Progressive Web Apps** - PWAs are enhanced web apps. They are developed using standard web technologies as well and run in browsers, but they provide other advanced functionalities, such as push notifications. They can also work offline, be added to the user's homescreen and be accessed in fullscreen, just like a native app. On one side they have the low development and maintenance costs of a web-based app, while also being able to have some of the functionalities of a native app [23, 24, 28].

## 2.3 USE OF TECHNOLOGY BY PREGNANT WOMEN

In section 1.2, a listing of the main characteristics that PHRs for pregnant women have is presented, as well as their online searching habits. In this section, these topics are deepened and related studies are presented.

### 2.3.1 *Personal Health Records for pregnant women*

A simple search in the Google Play Store or in the App Store for pregnancy related applications returns hundreds of applications. Tripp et al. (2014) found more than one hundred pregnancy-related applications in these stores. These applications have different features and purposes. For example, some of them focus on providing information, while others focus on providing tools [29].

Bachiri, Idri, Aléman & Toval (2016) carried a study that analysed 33 mobile PHRs for pregnancy monitoring for Android-based and iOS-based systems. Their main goal was to understand which features these PHRs had and if it was possible to improve in the future. All the analysed applications had a calculator to estimate the birth date and 97% included a weight tracking feature and the monitoring of other parameters, such as waist measurements. 94% of the studied applications displayed information for each stage of the pregnancy and 82% presented a progress bar or a birth date countdown. Other features are less common, such as recording the baby's kicks and contractions and tracking symptoms like nausea and vomiting. However, they can potentially be useful to prevent complications. As for the way this features are shown, for the evolution of the pregnancy, the information is presented more visually (videos or drawings) than textually and, for the recording of parameters, tables or graphs are usually used. Some applications also include: a calendar to record important events that has the possibility to enable reminders and notifications, a pregnancy journal, medication management (but is rarely present), tips about health habits and contraction counters, among others. The authors concluded that a good

mobile [PHR](#) needs to cover the maximum number of areas it possibly can and should be adaptive to the woman's needs [7].

[PHRs](#) have the potential to make women more responsible for their pregnancy and more engaged in this process, while also turning them into a more vital stakeholder in their health management. This might prevent complications and improve birth outcomes [8, 30, 31].

Bush et al. (2017) developed a study with two groups of pregnant women: one that used a mobile application provided by them and one that did not use the application. With this study, the authors aimed to understand the impact that a mobile health application may have in birth outcomes and user engagement. Even though the group that used the application was small compared to the other one, not making it possible to generalise the conclusions, the use of the application was associated with a small increase in the attendance of prenatal visits in the 6 months previous to birth and a lower incidence of low-birth weight newborns [31].

If the [PHR](#) is integrated and communicates with the medical institution that provides care to the pregnant woman, other advantages arise [2, 32]. [DSSs](#) have become increasingly more popular in [HISs](#) to help health professionals in the decision-making process [33]. These systems take information regarding the patient and give suggestions or alerts to the caregiver. To get to the decision, they usually use Artificial Intelligence techniques [34, 35]. [PHRs](#) could be a way of providing more information to [DSSs](#) and to improve the quality of information, which is an important requirement to have good results [36]. [DSSs](#) and intelligent systems can be applied in the obstetric practice, for example, to categorise pregnant women and predict preterm deliveries, the incidence of perineal tear or the associated risk [37, 38, 39, 40].

Nevertheless, [PHRs](#) also face challenges. The information presented in most applications does not come from a trusted party, such as a doctor. Spreading information that is not accurate may lead to unnecessary worries and anxiety, specially since most women are not capable to assess the quality of the information provided [29, 41]. The lack of quality of the content displayed in these

applications could be solved by involving health professionals in the development of these systems [42, 43].

### 2.3.2 *Online searching habits*

Pregnant women do not use only PHRs to find the information they want. With the rise of the Internet, large volumes of information, many times not from health professionals or institutions, became available online and women started to look for it to satisfy their needs of information. However, the access to large amounts of information does not mean that the woman understands it entirely, regardless of her literacy skills, and the lack of accuracy raises other issues [44].

Kennedy et al. (2017) conducted a study that had as main goal to find out which features pregnant women look for in a web-based nutritional information system. Even though the main goal was related only to the nutrition area, the writers also assess women's online searching habits and what they value in the mobile applications they use. All the women that were part of this study had Internet access, only 3% did not own a smartphone and 82,2% used online pregnancy fora, websites or mobile applications. When it comes to the way women search information online, Google/Internet search comes in first place (24,7%) which indicates that women do not look for evidenced based information, but rather the most popular websites provided by search engines. This may be an issue, since women may not be able to identify whether the information is credible. As for what the participants value in the mobile applications, websites or fora, the most popular answers were: social features (17,8%), informative (12,9%), information on pregnancy (11,9%), ease of use/convenience (9,9%) and ability to track foetal development (5%). Regarding what women looked for in a web-based nutrition intervention, the most popular answers were recipes (88%) and exercise advice (71%) [45].

Another study was conducted by Bjelke, Martinsson, Lendahls & Oscarsson (2016) to analyse how women use the Internet during their pregnancy and the consequences that it brings. 95% of the women that were part of this study

used the Internet to search for information. Women used the Internet during their pregnancy mainly to find information, read about people in the same situation and find support. The two main sources of information were Information websites (94%) and Forum websites (71.6%). 65,6% of the women said they felt worried after reading information about the pregnancy online and Forum web pages were the web pages that caused more anxiety (43,7%). Women with a high search frequency also contacted healthcare services more often due to worries. As for the topics that women read the most online, they are foetal development, delivery and complications during pregnancy. Women that attended parental education classes still searched for information online. These women find searching for information online a complement to the information they receive from health professionals. To minimise the worries that most women feel, it is advisable that health institutions and professionals inform women about the advantages and disadvantages of looking for information online and recommend trustworthy web pages [46].

---

## METHODOLOGIES AND TECHNOLOGIES

---

In this chapter, the methodologies for research and evaluation are presented, as well as the main technologies used and development strategies followed.

Thus, section 3.1 explains the main research methodology used, *Design Science Research (DSR)*. Section 3.2 focus on React, Node.js, Express and MongoDB, the main technologies behind this project. In section 3.3, the **REST** pattern is presented and in section 3.4 the **PWA** strategy and its principles and characteristics are further explained. Finally, in section 3.5, the evaluation methodologies that were used to evaluate the developed system are presented. These methodologies are Lighthouse reports, **SWOT** analysis and usability testing.

### 3.1 RESEARCH METHODOLOGY

Research in the *Information Systems (IS)* field is different from research in more traditional subjects, such as social or natural sciences, in the sense that it is an applied research field that solves problems from other areas (e.g. economics) using *Information Technology (IT)*. Therefore, the methodologies followed in the **IS** area, should be different. **DSR** is a methodology that aims to solve this situation [47].

**DSR** focus on designing a product that solves a problem, evaluating the result and communicating it to the proper audiences [47, 48]. According to Hevner et al. (2004), **DSR** has seven principles:



1. **Design as an Artifact** - a viable artifact must be produced. This artifact can be a construct, a model, a method or an instantiation;
2. **Problem Relevance** - the developed artifacts must solve important and relevant problems;
3. **Design Evaluation** - the artifact must be well evaluated in order to assess its utility, quality and efficacy;
4. **Research Contributions** - **DSR** must give important contributions in the field of the designed artifact, design principles or design methodologies;
5. **Research Rigor** - the research process should follow rigorous methods during both development and evaluation;
6. **Design as a Search Process** - the design process is a search process to discover an effective artifact. This implies using the existing means (actions or resources) while satisfying the environment's laws (e.g. conditions and restrictions);
7. **Communication of Research** - the artifact must be presented to the proper audiences [48].

**DSR** was the methodology chosen for this project, because this work aims to solve a problem in the obstetrics field that needs the application of **ITs**.

### 3.2 TECHNOLOGIES

In this section the main technologies chosen to accomplish this project are presented. These technologies are React for the front end, Node.js and Express for the server side, MongoDB as the database and KeystoneJS for the administration panel.

The choice of the software stack was made according to what was previously presented in chapter 2. MERN stack was chosen mainly because it is a recent stack and it only needs one scripting language, JavaScript, which turns the development of a complete application easier. MERN stack was chosen over the

MEAN stack, because React has had a significant growth over the last few years and now is becoming even more popular than Angular, which was the most popular JavaScript framework for the front end. This growth is visible in Figure 1.



Figure 1.: Evolution of the interest over time for React and AngularJS all over the world. The displayed data comes from the Google Trends tool. The score is calculated according to the amount of times the terms are searched using the Google search engine, being 100 the biggest one and 0 the lowest [49].

### 3.2.1 React

React is an open-source JavaScript library developed by Facebook and that first appeared in 2013. React implements the view in the MVC model, so it can only be used to implement the visual interface of the application [50, 51].

React does not use the traditional DOM, it uses the virtual DOM. The virtual DOM exists on top of the traditional one and it solves the state manipulation problem by figuring out the best way to propagate data changes to the DOM [52, 53].

There are two ways of handling the virtual DOM, using the JavaScript API or using JSX [52]. They are represented in Figure 2 and in Figure 3, respectively. Comparing the two approaches, one can see that using JSX results in a simpler and easier to read approach, specially in bigger components.

```
const Hello World = () => {
  return React.createElement(
    'div',
    React.createElement('h1',
      'Hello World'),
  );
};
```

Figure 2.: Hello World example using the JavaScript API.

```
const Hello World = () => {
  return React.createElement(
    <div>
      <h1>Hello World</h1>
    </div>
  );
};
```

Figure 3.: Hello World example using JSX.

JSX is an extension of the ECMAScript, the base of JavaScript. It uses a syntax similar to XML and enables the addition of HTML-like code with JavaScript. JSX is not understood by browsers, so it is necessary to use a transpiler to transform JSX into JavaScript. In this project the chosen transpiler was Babel [54, 55].

*React-dom* is the most used renderer and it enables both client and server side rendering. By implementing server-side rendering, the server renders the initial markup and sends it to the client with the initial data as well. This improves performance and is also useful for *Search Engine Optimisation (SEO)* [52].

One concern we had during the development of this application was to make it responsive, that is, to make it work on all devices no matter their screen size. To accomplish this, we used Bootstrap. Bootstrap is a popular open-source toolkit to build responsive and mobile-first websites [56]. To add Bootstrap to our application the React-Bootstrap library was used. This library offers a way of integrating Bootstrap into React. At the moment, it supports the version 3 of Bootstrap [57].

### 3.2.2 Node.js and Express

Node.js is a platform built on top of the JavaScript runtime suitable for building fast and scalable network applications. To achieve this, Node.js uses an event-driven and non-blocking I/O model [58, 59].

Both Node.js and browsers are event-driven, that is, they use an event loop. The event loop passes operations to the system kernel when possible. The system kernel is usually multi-threaded, unlike JavaScript, and, therefore, can perform multiple operations in the background. When the system kernel finishes an operation, it tells Node.js. Then, Node.js adds the operation's callback to the kernel's poll queue to be later executed. It is important to mention that JavaScript is asynchronous, so, unless some synchronous strategy is used, the order that the code is written, may not be the order that it is executed [58, 60].

I/O stands for interactions with the system's disk and the network. Blocking happens when the execution of a Node.js process has to wait for a non-JavaScript operation to complete. In Node.js all the standard I/O methods have asynchronous versions that are non-blocking and accept callbacks. Some of these methods also have synchronous versions. An example from the Node.js documentation that illustrates the difference between synchronous (blocking) and asynchronous (non-blocking) operations is in Figure 4 and Figure 5, respectively [61].

```
const fs = require('fs');
const data = fs.readFileSync('/file.md'); // blocks here until file is read
console.log(data);
// moreWork(); will run after console.log
```

Figure 4.: Synchronous file reading operation [61].

```

const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
  console.log(data);
});
// moreWork(); will run before console.log

```

Figure 5.: Asynchronous file reading operation [61].

The I/O non-blocking model makes the server more responsive and able of handling more requests and clients. It also reduces latency and makes better use of the server's resources. This is why Node.js is good for web applications. Real-time web applications need a server that can respond fast and that can handle many users and Node.js is capable of doing both [58].

Studies have been made in order to compare Node.js performance with other server-side technologies, such as Apache, Nginx, PHP and Python. These studies have concluded that Node.js is more suitable for serving dynamic content by handling more requests per second, making better use of the CPU capabilities and handling large requests better [62, 63, 64].

There are frameworks that simplify the development of web applications. On the server side, these frameworks provide APIs to handle HTTP methods for example. For Node.js one of the most popular frameworks is Express. Express is a routing and middleware framework to build modern web and mobile applications [58, 65, 66].

Express does not have a fixed structure for the application, the developer is the one that defines the directory structure [58]. A basic Express application is shown in Figure 6. The Express framework is loaded and an Express application instance called *app* is created. Then, it is defined that when a GET request to the URL `"/"` is made, the server responds with `"Hello World!"`. *App* listens to requests on port 3000 and will respond to all requests with a 404 response code, except for the request that was defined [67].

```
const express = require('express')
const app = express()

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(3000, () => console.log('Example app listening on port 3000!'))
```

Figure 6.: Basic Express app configuration [67].

This is a simple example, but that touches one of the most important features that Express provides: routing. Route methods define how the application should respond to client requests. In each routing method is specified the [HTTP](#) method, a [URL](#) and a handler function. When the application receives a request that matches the [HTTP](#) method and [URL](#), the handler function is performed [68].

Express also provides other more advanced functionalities, such as middleware functions. These functions can access the request object and the response object, change them and call the next middleware function [69].

### 3.2.3 MongoDB

MongoDB is a document database suitable for web applications. It is highly scalable, flexible and has good performance. Apart from these advantages, MongoDB is also attractive due to its intuitive data model [70, 71].

In MongoDB databases, data is stored in [JSON](#)-like documents, which enables all information about an instance to be stored in only one document, contrasting with relational databases where it would be scattered in several tables [71].

Documents are stored in Collections which are the equivalent to tables in relational databases. By default, documents do not have to have all the same schema, that is, they can have different fields and the data type of a field does not have to be always the same [71]. A simple example of what a document looks like is in Figure 7. This document stores basic information about a user

and his appointments. It has a set of properties and their values. The *\_id* field is the primary key.

```
{
  _id: ObjectID('5aa3c1a707117514c1bb25ff'),
  name: 'Steve Rogers',
  birthDate: 1918-07-04,
  appointments: [
    {
      doctor: 'Jackson Avery',
      date: 2018-09-11,
    },
    {
      doctor: 'Miranda Bailey',
      date: 2018-08-10
    }
  ]
}
```

Figure 7.: MongoDB document example.

Node.js supports MongoDB through the MongoDB driver. It enables the creation, update, delete and querying of collections and documents [72]. However, there are other alternatives to interact with MongoDB. One of these alternatives is Mongoose. Mongoose provides an [API](#) to deal with MongoDB that includes validation and query building [73].

The base unit in Mongoose are Schemas. Schemas define the fields and their data types of a MongoDB collection. This ensures that all documents in a collection have the same fields and that the data type of a field is consistent throughout all documents, while also providing validation [74]. For the example in Figure 7, a possible Schema would be the one presented in Figure 8.

```
const userSchema = new Schema({
  name: String,
  birthDate: Date,
  appointments: [
    {
      doctor: String,
      date: Date
    }
  ]
});
```

Figure 8.: MongoDB schema example.

To use a Schema, it is necessary to convert it into a Model. Then, it is possible to apply methods to change the documents inside a Model, as documents are instances of Models [75]. Some of the basic operations and their respective source code using Mongoose are in Table 1.

#### 3.2.4 KeystoneJS

An administration panel enables the manipulation of data and can also provide a dashboard with relevant business metrics and analytics in real-time. The use of an administration panel can increase productivity, give useful information about the stored data and the users, and help in finding bugs [77].

For the development of the administration panel, the open-source framework KeystoneJS was chosen. KeystoneJS enables the development of database-driven websites, applications or APIs using Node.js, Express and MongoDB [78].

A KeystoneJS project can be connected to a MongoDB database using Mongoose and it provides an administration UI that can be personalised in order to view and manipulate content from the database. It provides a set of out-of-the-box features to do this manipulation, such as editing data, creating documents, searching inside a model (with or without additional filters) and deleting docu-



Table 1.: Some actions and their respective source code using Mongoose [75, 76]

Action	Source code
Create a model	<pre>const User = mongoose.model('User', userSchema);</pre>
Create and save a document	<pre>const newUser = new User(   { name: 'Steve Rogers',     birthDate: new Date('1918-07-04'),     appointments: [ ]   } ); newUser.save();</pre>
Update a document	<pre>User.update(   { _id: ObjectID('5aa3c1a707117514c1bb25ff') },   { \set:     { appointments: [{ doctor: 'Jackson Avery', date:       2018-09-11 } ] }   },   callback );</pre>
Delete a document	<pre>User.deleteOne(   { _id: ObjectID('5aa3c1a707117514c1bb25ff') },   function (err) {     if (err) return handleError(err);   } );</pre>
Query a collection	<pre>User.find({name: 'Steve Rogers'}, function(err, users) {   \Do something with the users });</pre>

ments [79]. In Figure 9 is an example of what an administration UI looks like using KeystoneJS.

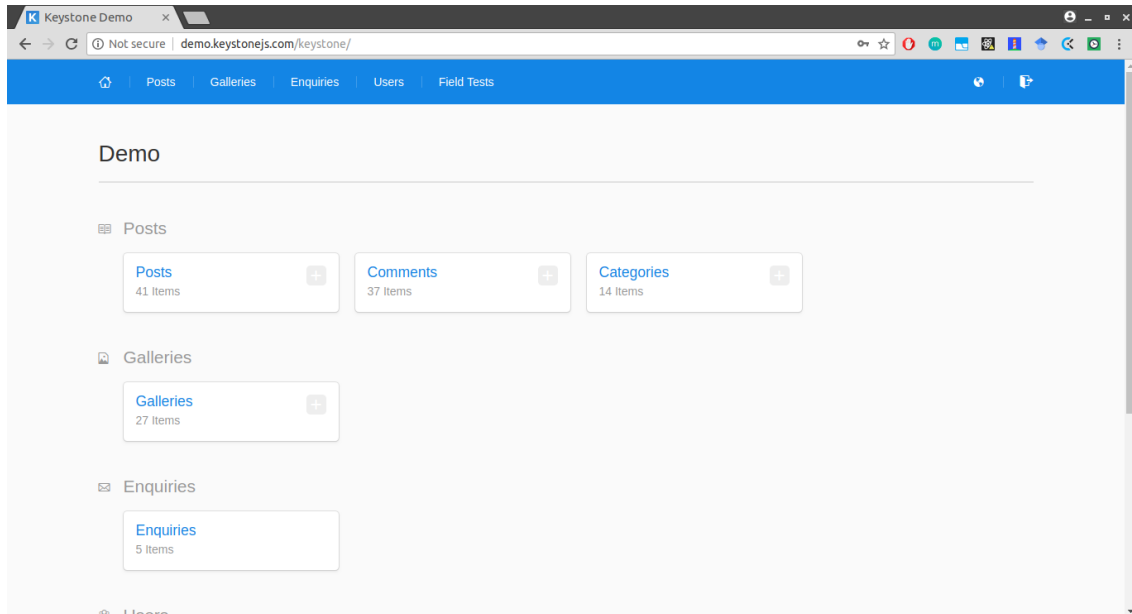


Figure 9.: Administration UI example provided by KeystoneJS [80].

### 3.3 RESTFUL WEB SERVICES

The technologies explained above, namely Node.js and Express, are the ones that make it possible to develop a RESTful web service for the server side.

The **REST** concept was first proposed by Roy Fielding in 2000. A web service that has a **REST API** is a RESTful web service. A RESTful web service provides a way of implementing *Create, Read, Update and Delete (CRUD)* actions. To do so, the **HTTP** request methods GET, POST, PUT and DELETE are used to manipulate or retrieve a resource identified by a **URL**. The server should respond using **JSON** or **XML** [58, 81]. Briefly, some of the principles behind RESTful web services are:

1. *Addressability*: all resources should have a unique identifier;

2. *Uniform Interface*: all resources interact through the same uniform interface. In the context of Web, this interface contains methods in accordance to the [HTTP](#) methods;
3. *Stateless interactions*: these services do not establish permanent sessions between each other. Sessions are solely each interaction, no state remains after the interaction. Once the interaction is complete, the resource is updated and this new version becomes available to all clients;
4. *Self-describing messages*: all the information necessary should be explicit in both the request and the response. Nor the client or the service make assumptions about how the representation should be processed or what it means;
5. *Hypermedia*: hypermedia is a way of adding connections (identifiers) to related resources inside the resource or in its meta-data. The Client can identify these identifiers and decide to follow the link. Hypermedia is useful with decentralised resource discovery and dynamic discovery. Even though it is a useful, most RESTful web services do not use it [82].

According to its maturity, a RESTful web service can be classified into 4 levels:

- *Level 0: HTTP as a tunnel* - only exchanges [XML](#) or [JSON](#) documents using [HTTP](#) POST requests and responses. Even though, these services do not use [SOAP](#) messages, they do not use the [HTTP](#) protocol according to the [REST](#) standards as well. Since all messages, go to the same [URL](#), the web service only knows which operation to perform by processing the message's content;
- *Level 1: Resources* - the difference between this level and the previous one is that Level 1 web services use multiple identifiers to identify the resources. [HTTP](#) request methods are still not present explicitly in the request;
- *Level 2: HTTP Verbs* - the requests sent to the web service specify the [HTTP](#) request that the client wants to apply to the specified the resource and the request is made according to each method's rules. For example, if a PUT

request fails, it can be automatically resent. Web services can also use [HTTP](#) status codes to respond to the client;

- *Level 3: Hypermedia* - this is the most advanced level. These web services make use of the Hypermedia advantages in addition to the characteristics previously presented [82].

To create a [REST API](#), the first step should be to identify the resources that will be involved, their [URLs](#) and the action that should be performed (e.g. create a user). The action may involve interaction with the database [58]. This is what Express does with its Routing system and, therefore, shows how Express is suitable for this type of architecture.

### 3.4 PROGRESSIVE WEB APPS

The technology chosen to develop this application was [PWA](#). In section 2.2, an overview of the state-of-the-art of mobile development strategies is made. From this overview, one can conclude that [PWAs](#) are a viable alternative to Hybrid Mobile Apps and have superior features to those of Mobile Web Apps. However, the one characteristic that influenced the most our decision was the fact that with only one app we could develop for both web and mobile, while still having access to most features available that native apps have, decreasing significantly development time and also targeting a bigger audience.

Google introduced [PWAs](#) in 2015. These apps are websites optimised to work on mobile devices. Their core technologies include [HTML5](#), [CSS](#) and [JavaScript](#) [83]. [PWAs](#) have several characteristics that differentiate them from other web apps, namely:

- They are available for everyone no matter the browser or device;
- Content is displayed properly in any device;
- They don't depend on Internet connectivity, because the service worker will handle the app behaviour when offline or on low-quality networks;

- The app is always updated thanks to the service worker;
- They are served through a *Hypertext Transfer Protocol Secure (HTTPS)* connection;
- The service worker and the app manifest make the app discoverable in search engines;
- The app is installable in the user's home screen like a native app [24, 83, 84].

Even though this is a recent technology, some known companies have already adopted this development strategy. This change has resulted in better performance and more user engagement. In Table 2 are listed some of the companies that did this change and the main highlights that resulted from this change. Windows also has started to allow the publishing of PWAs in the Windows Store recently [85].

When a user first visits a PWA, it is accessed like a regular web app through the browser. After a group of metrics are met, the user is prompted to install the app. The metrics that fire the prompt are :

1. The app is not already installed;
2. The user has interacted with the app for at least 30 seconds;
3. Includes a web app manifest that has the parameters *short\_name*, *name*, *icons*, *start\_url* and *display* (*fullscreen*, *standalone* or *minimal-ui*);
4. Is served over HTTPS;
5. Has a service worker registered [94].

If the user accepts the prompt, the app is added to the home screen and all necessary static files are cached, so that they can be used offline [24].

#### 3.4.1 Service worker

The essence of a PWA is the service worker. It is a JavaScript file that runs separately from the main browser thread. This makes service workers inde-

Table 2.: Examples of companies that started using PWAs and the main highlights of this change [86, 87, 88, 89, 90, 91, 92, 93]

Company	Highlights
Tinder	10% of the size of the native Android app More swipes and messages, longer sessions and more profiles edited than on the native apps
Trivago	More than a half a million have added the trivago site to the home-screen Users that added to the homescreen engage 150% more, thanks to the push notifications Clickouts to hotel offers increased 97%
Forbes	10% increase in impressions per session PWA users spend up to 40% more time per session and view 15% more pages The number of users that read less than a fourth of an article decreased by 20% Updates to the site are made in real-time
OLX	User engagement increased by 250% only by adding push notifications and the "Add to Home screen" feature The time until the page becomes interactive decreased by 23%
Twitter (in Twitter Lite)	Pages per session increased by 65% Tweets sent increased by 75% 250 000 users launch the app from the homescreen daily, 4 times a day on average There are over 10 million push notifications sent through the PWA daily Data consumption reduced by 70% The PWA is 600kB while the native android app is 23,5mB When the user returns to the app, it loads in less than 3 seconds even in slow networks or slow mobile devices
Pinterest	Time to interactive was reduced from 23 seconds to 5,6 seconds The PWA is only 150kB, contrasting with 9,6mB for the Android app is 56mB for the iOS one After one year, weekly active users increased by 103%, session length increased by 296%, number of Pins increased by 401%, logins increased by 370% and new signups increased by 843% Mobile web became the top platform for new signups In less than 6 months since the launch, 800 thousand weekly users accessed the PWA from their homescreen
Lancôme	Over a year, the number of pages per session increased by 50% and transactions increased by 23% Time to be interactive decreased from 11,195 seconds to 1,772 seconds

pendent from the application they are associated, which in turn makes them asynchronous [83]. Service workers make it possible to:

- **Cache resources:** by caching resources, the app will load faster even under bad network conditions;
- **Work offline:** network requests are intercepted and their response may be modified with content that was not the requested. This method can be used to serve assets from the cache when the user is offline, making the app fully capable of working offline;
- **Use advanced features:** service workers are the starting point for more advanced features, such as notifications, push messages or background sync [83, 95].

The service worker's lifecycle has three stages: Registration, Installation and Activation. In Registration, the browser is told where the service worker is and to start installing it in the background. When the service worker is registered, Installation happens. During the Installation event it is possible to perform some tasks, such as precaching resources. Finally, once this step is completed, the service worker is activated. During Activation, it controls all pages and resources within its scope and listens to events from them [95].

#### 3.4.2 *Offline Support*

One of the biggest advantages of **PWAs** is the ability to serve the app even when there is no Internet connection. As previously explained, this is possible due to service workers, because they have the ability to use the Cache Interface. The storage from the Cache Interface is independent from the browser's HTTP cache and, therefore, it is available offline. Aside from the cache, service workers can also use IndexedDB to store data. Using the offline capabilities of **PWAs**, the overall **UX** improves, since it results in faster loading and less network expenses [96].

Offline support can be enabled by caching static data during Installation and have the service worker listening for resource fetches. When a resource is requested, the service worker tries to find it in the cache and only if the resource is not available, it uses the network. When the last option happens, the service worker may copy the resource and cache it for later use [96]. There are other strategies to deal with resources, namely:

- **Cache only:** the service worker only retrieves the cached resource, unless it is expressed otherwise;
- **Network only:** the service worker only retrieves data from the network;
- **Cache, falling back to network:** if the resource is not cached, it will use the network to get it;
- **Network falling back to cache:** the contents retrieved when online are the most updated, but when the user is offline the cached version is retrieved;
- **Stale while revalidate:** if the cached resource is available, it is retrieved and then a network request is made to update the cache [97].

### 3.5 EVALUATION METHODOLOGIES

To perform the proof of concept, three different methods were used: Lighthouse reports to evaluate the technical performance, [SWOT](#) analysis to assess the state of the application and areas to improve, and usability testing to understand if the application is intuitive and easy to use by pregnant women.

#### 3.5.1 *Lighthouse reports*

Lighthouse is an open-source tool developed by Google that can run in any web page. It can be used from the Chrome DevTools, the command line, a Node.js module and as a Chrome Extension [98].



It assesses the page's **performance**, **accessibility** and **SEO**, to which degree the page is a **PWA** and if it is developed according to the **best practices** [98]. The fact that this tool evaluates such a wide range of performance indicators, some of them specific of **PWAs**, was the reason why this tool was chosen.

It evaluates from 0 to 100 each indicator, being 0 the worst score and 100 the best one. A score of 50 represents the 75<sup>th</sup> percentile [99]. Lighthouse also identifies topics to be improved and provides links to web pages that have helpful information regarding those topics [100].

### 3.5.2 SWOT Analysis

**SWOT** analysis is an acronym for *Strengths, Weaknesses, Opportunities* and *Threats*. It is one of the most used tools for strategic planning used by organisations and allows managers to quickly identify both internal and external factors that need to be taken into account and helps in the decision-making process as well [101].

The biggest advantage that the **SWOT** analysis provides is its simplicity of use. Because of that, it does not have to be performed by specialised professionals. Aside from this advantage, it can also be applicable to different levels in an organisation and at different depths, and it is extremely visual, making it easy to communicate the results to stakeholders [101, 102]. However, it is necessary to be careful while using it, in order to include quality data and non-biased data so that good results can be achieved [102].

An **external factor** is an element related to the environment that the organisation cannot control. On the other side, an **internal factor** can be controlled or changed by the organisation. **Strengths** are internal factors that strengthen the organisation's competitive position. **Weaknesses** are internal factors as well, but that weaken the organisation's competitive position. Both **opportunities** and **threats** are external factors, but the first ones have a positive influence in the competitive position of an organisation, while the latest impact negatively the organisation's position. Strengths and opportunities are **positive factors**,

because they are factors that can help succeeding. Weaknesses and threats are **negative factors**, because they compromise the success of the organisation [102].

This division allows the organisation to easily identify the main factors that are influencing both positively and negatively its performance. It also allows organisations to prioritise factors in terms of their expected impact [102]. When performing the analysis, a **SWOT** diagram should be drawn with the structure that is shown in Figure 10.

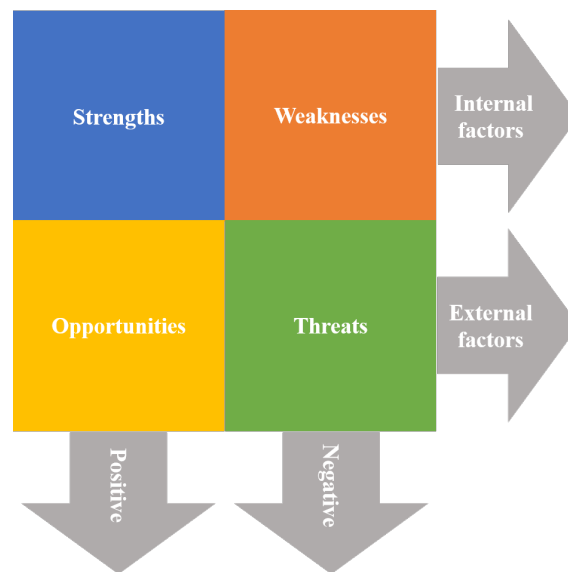


Figure 10.: SWOT diagram structure.

### 3.5.3 Usability Testing

Usability of a software system is the capability that this system presents to enable users to achieve effectively and efficiently the tasks they aim to complete [103].

*Usability Evaluation (UE)* methods measure the usability of a UI and identify specific problems. UE usually has three stages:

1. **Data gathering:** collect usability data, such as completion time of tasks and errors;

2. **Analysis:** analyse the data previously gathered and identify usability problems;
3. **Critique:** provide possible improvements and solutions to the identified problems [103].

User-based methods are a type of **UE** methods in which the main participants are possible users. In this type of testing, users complete a set of tasks previously determined or explore freely the system. The users' behaviours are examined and recorded to be later analysed. Aside from the user's behaviour, other parameters can be recorded, such as which tasks were completed, the necessary time and the number and types of errors. Then, results are analysed and improvements are proposed.

The first step in a usability test, is to identify the test main objectives. Then, one should determine the characteristics that the participants need to present. After these two steps, the tasks that the participants will perform should be selected, as well as the creation of task scenarios. It is also important to define the measures that will be recorded. The organisers need to prepare the test's materials and environment and define the data analysis procedures. Finally, results should be presented.

One of the biggest decisions during this process is to decide how many users should be tested. There is not a consensus regarding this issue. Although most studies have shown that with 4 or 5 five participants most problems are identified, there are also some studies that concluded that with only a larger group more serious issues were found.

Most tests are recorded in order to later evaluate the tasks duration, how many times the user chose the wrong icon, etc. There are some tools that record the user actions, including audio, camera and video. They are specially convenient when testing web sites [104].

---

## CASE STUDY

---

In this chapter, the resulting platform, *As 10 luas: gravidez, parto e pós-parto do CMIN*, is presented, as well as the process behind its development. In the end, a critical assessment of the developed system is performed.

### 4.1 REQUIREMENTS ANALYSIS

The first step in this project was to perform a requirements analysis in order to determine the features that should be available in the platform and to identify technical constraints.

The information that was analysed during this stage came from several sources. One of these sources was the [CMIN](#). Dr Jorge Braga, an obstetrician in the [CMIN](#), provided feedback on which were his perceptions about which are the needs of pregnant women. He realised that most pregnant women had recurring questions regarding their pregnancies and they would reach their doctors or nurses for information. Dr Jorge Braga considered useful to have a digital version of the *Boletim de Saúde da Grávida (BSG)* for the pregnant women as well. In the [BSG](#), medical information is registered during appointments and it has a place where women should register certain parameters between appointments.

The analysis of the existing literature on this matter presented in chapter 2 was another important source to understand the [PHRs](#) available and which are the pregnant women's habits regarding technology.

However, these were not the only sources. Since this project intended to develop a system for pregnant women and that they would want to use, it was important to have their feedback during this process. The process of having feedback from pregnant women was facilitated due to the fact that we already had a connection in the [CMIN](#). This process consisted in informal conversations with a total of 25 pregnant women. With each conversation we wanted to understand:

- If the pregnant woman searches for information online:
  - if yes, which topics she usually searches for;
- If she uses mobile apps:
  - If yes, which one(s) and which feature(s) she enjoys;
  - If no, why not and what would make her use one.

Given all the gathered information, the requirements for this platform were the following:

1. It should be available for as much pregnant women as possible. To do this, the development techniques chosen have to work in both **mobile** and **web** environments;
2. There should be and **administration panel** that allows the management of the content available in the application;
3. There are features content available for everyone:
  - a) **FAQs** - information provided by the [CMIN](#);
  - b) **Calculator** - calculator of the due date and the current gestational age. The algorithm behind these calculations was provided by Doctor Jorge Braga;
  - c) **Maternity and Paternity legislation** - the current Portuguese legislation for the maternity and paternity;
4. If the pregnant woman chooses to create an account, she has access to other features:

- a) A **personal file** that saves information about the woman and her clinical history;
  - b) There has to be a place to save **appointments**;
  - c) There has to be a place to save important **documents**, such as ultrasounds and analysis;
  - d) She will receive **personalised information** related to nutrition and lifestyle, according to her clinical history and the pregnancy;
  - e) She will receive **weekly updates** with information about the fetus and the changes she should be noticing;
5. The information recorded in the **BSG** should also be available in the developed system, this includes information related to the woman's clinical history and some parameters (e.g. weight and blood pressure), among others.

After this analysis, the technologies that would be used, the structure that the application would follow and the features that would be present were defined.

Given the first point, it was decided that this platform would be a **PWA**. Considering the options available this is the one that gives a better compromise between the final result and development cost, while being available to anyone who has access to a computer or a mobile phone, as explained in chapter 3.4.

For the second point, it was decided that KeystoneJS would be appropriate for the administration panel, as presented in chapter 3.2.4;

For the features that would be available and the general structure, it was decided that the platform would have an area that does not required authentication and an area that requires.

The area that does not require authentication is structured as in Figure 11. In this area there is a *Home Page*. From this page the user can access the following features: *Calculadora* (calculator), *Perguntas Frequentes (FAQs)*, *Legislação* (legislation). The *Perguntas Frequentes* area is divided in: *Antes da gravidez* (before the pregnancy), *Durante a gravidez* (during the pregnancy) and *Após o parto* (after delivery). The user can also access the log in and sign up pages. When

the pregnant woman first creates an account, she goes through two pages to insert her personal and clinical data, *Ficha pessoal* and *Ficha clínica* respectively. The information recorded in these two steps is the same that is recorded in the *BSG*. Then, she has access to the *Dashboard* page that is the starting point of the authenticated area.

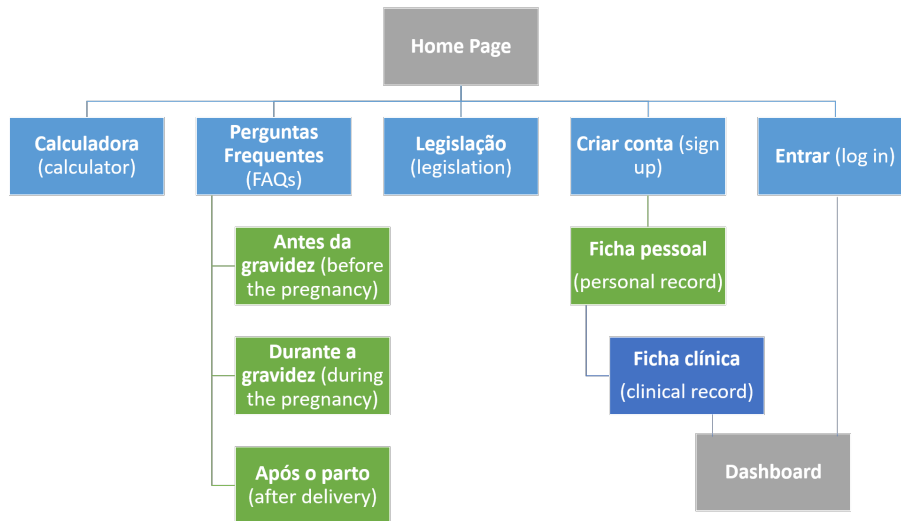


Figure 11.: Sitemap for the area that does not require authentication.

In the authenticated area (Figure 12) the woman has access to the same features that she had in the unauthenticated area under the tab *Outros* (others). Aside from these features, the pregnant woman also has access to:

- *Ficha Pessoal* (personal record) to see and update her personal data (*Dados pessoais*) and her clinical data (*Ficha clínica*);
- *Marcações* (appointments) where she has a calendar and can add, update and delete appointments;
- *Documentos* (documents) to upload files;
- *Saúde e Bem-estar* (health and wellness) where she can not only receive personalised information regarding the fetus development weekly, medical conditions she might present and tips about health and nutrition, but also register important moments and symptoms like nausea and vomits;

- *Registos* (records) to register her weight (*Registar peso*), blood pressure (*Registar pressão arterial*), uterine length (*Registar altura uterina*), baby's kicks (*Registar movimentos fetais*) and sugar levels (*Registar glicemia*). These records are the same that are made in the [BSG](#).

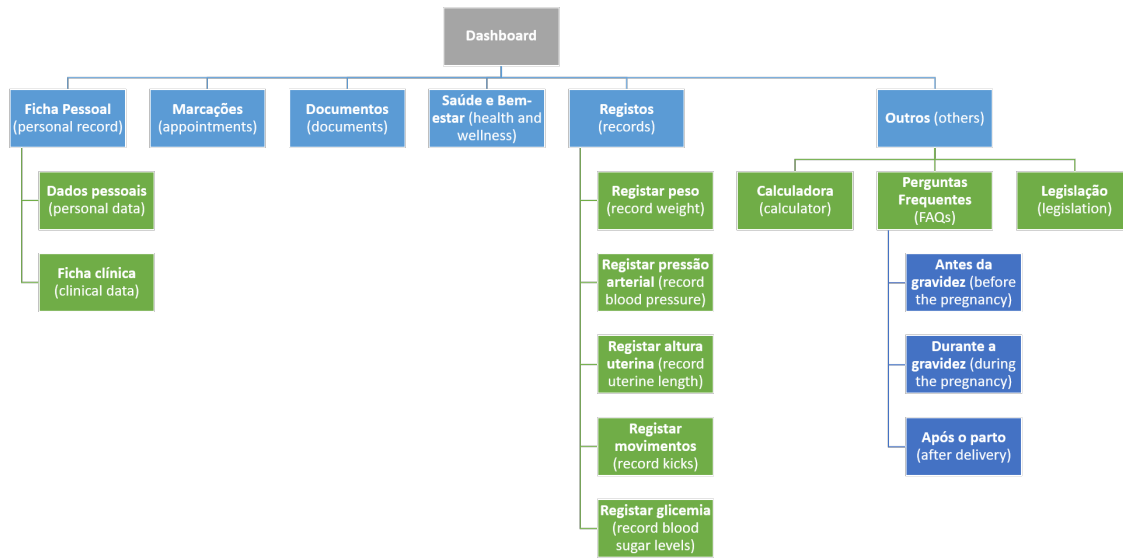


Figure 12.: Sitemap for the authenticated area.

## 4.2 ARCHITECTURE

In chapter 3, an overview of the main technologies and methodologies was made. In this chapter, we understand how they connect between each other to form this platform. The architecture of this platform is presented in Figure 13.

On the server side we have a web server, a database and an administration panel. The database is a noSQL database, more precisely a MongoDB database. It saves all the users' data, but also the content that is shown in the application (e.g. [FAQs](#)). The web server is written in Node.js and has a middleware in Express. These two parts form a [REST API](#) or RESTful web service. The [REST API](#) is coded to answer requests from the client side by getting information from



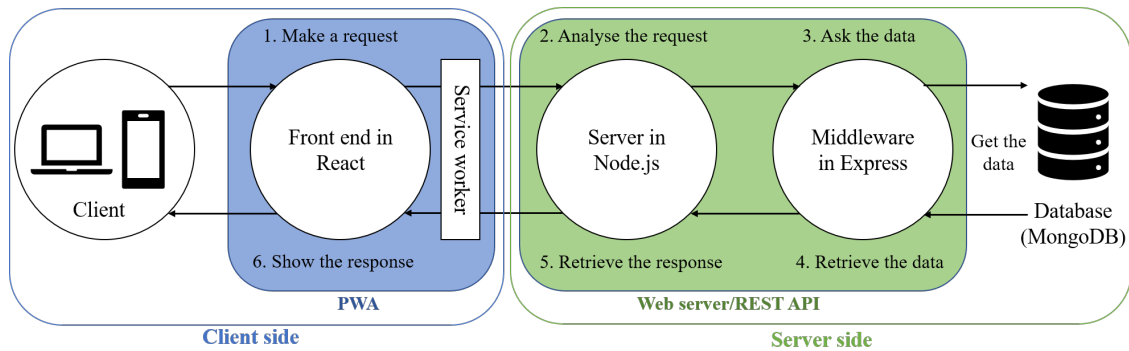


Figure 13.: Architecture of the developed platform.

the database. Even though the web server acts mainly as RESTful web service, it also performs other minor actions (e.g. handles authentication).

The client side is composed by the [PWA](#) and the user's devices. The [PWA](#) is made of a front end in React (view part from the [MVC](#)) and a script (service worker) that intercepts all requests made to the back end. The application receives user inputs and interactions and answers accordingly. If necessary, it requests resources to the server. The service worker enables most features that make this application a [PWA](#), namely offline support.

The administration panel shows the data from the database and can also send requests to the database (e.g. deletion of an entry).

In [1.4](#), it was explained that three students worked in this project. Therefore, each one was responsible for different parts. I was responsible for the [REST API](#), the transformation to a [PWA](#), the administration panel and some parts of the front end (*Criar conta*, *Entrar*, *Dados Pessoais*, *Marcações*, *Documentos*, *Perguntas Frequentes* and *Legislação*). Thus, these will be the parts that will be focused in the upcoming sections.

### 4.3 DEVELOPED SYSTEM

In this chapter a detailed analysis of each element that make up the platform is made. Thus, it starts by explaining the structure of the application. Followed by the implementation of a RESTful web service in Node.js and Express. On

the client side, the user interface for the relevant features for this dissertation will be shown and the transformation to a PWA will be explained. Finally the developed administration panel is presented.

#### 4.3.1 Database

As previously said, the database used in this platform is a MongoDB database. Given the requirements that this project presents, it was necessary two schemas, one to save the data collected from the pregnant women and another to save the FAQs.

The schema that saves the information about the pregnancy is called *gestacaoSchema*. It saves all the personal and clinical data from the pregnant woman, as well as the information necessary for the authentication, documents and appointments, among others. In Figure 14 is part of the complete schema with the relevant properties for this dissertation.

It is important to remember that by using Mongoose it is possible to restrict the data types that each property has, as well as other restrictions, helping the validation process. For example, for the property *email*, it is established that it is a required string and that it should be unique and trimmed before saving to the database.

In order to save the password in a safe way it was important to not save it in its natural form, that is, to save its hash. This was accomplished by having a method that gets the hash of the password and substitutes the password value with it. This method is called before performing a save action in this schema and if the password parameter is being altered.

Given this, the properties that make up this schema are the following:

- *email*: user's email;
- *password*: hash of the user's password;
- *gravida*: object that saves information about the pregnant woman. It has the following properties:

```

const GestacaoSchema = new mongoose.Schema({
  email: {
    type: String,
    unique: true,
    required: true,
    trim: true,
  },
  password: {
    type: String,
    required: true,
  },
  gravida: {
    nome: String,
    dataNascimento: Date,
    profissao: String,
  },
  outroProgenitor: {
    nome: String,
    dataNascimento: Date,
    profissao: String,
  },
  consultas: [{
    dataConsulta: Date,
    nomeConsulta: String,
  }],
  documentos: [{
    dataDocumento: Date,
    nomeDocumento: String,
    categoriaDocumento: String,
    caminhoDocumento: String,
    fileName: String,
  }],
}, {
  collection: 'users',
});

```

Figure 14.: Mongoose schema where the information about the pregnancy is saved.

- *nome*: pregnant woman's name;
- *dataNascimento*: pregnant woman's birth date;
- *profissao*: pregnant woman's job;
- *outroProgenitor*: object that saves information about the other parent. It has the following properties:
  - *nome*: name of the other parent;
  - *dataNascimento*: birth date of the other parent;
  - *profissao*: job of the other parent;

- *consultas*: array with multiple objects that represent appointments. Each item in the array has the following properties:
  - *dataConsulta*: appointment's date;
  - *nomeConsulta*: appointment's name;
- *documentos*: array with multiple objects that represent documents. Each item in the array has the following properties:
  - *dataDocumento*: document's date;
  - *nomeDocumento*: document's name;
  - *categoriaDocumento*: document's category;
  - *caminhoDocumento*: system path where the document is saved;
  - *fileName*: name of the document's file.

As for the [FAQs](#), the schema is smaller (Figure 15) and has fewer properties:

- *tipo*: category where the [FAQ](#) belongs;
- *titulo*: [FAQ](#)'s question;
- *resposta*: [FAQ](#)'s answer. It is an array, so that the answer can be divided in paragraphs.

```
const Faqs = new mongoose.Schema({
  tipo: String,
  titulo: String,
  resposta: [String],
}, {
  collection: 'faqs',
});
```

Figure 15.: Mongoose schema where the [FAQs](#) content is saved.

#### 4.3.2 Web server

As previously explained the developed web server acts mainly as a [REST API](#), but it also performs other actions, such as handling authentication. In this section, we will go through each function that the web server performs.

##### *REST API*

In section [3.3](#) the purpose and main principles behind RESTful web services were introduced. To have a mature RESTful web service it is necessary to identify the involved resources, define unique identifiers for these resources and handle requests made to these resources.

Express is suitable for these type of web services because, as explained in section [3.2.2](#), it makes it possible to define route methods, that is, methods that specify the [HTTP](#) method, the [URL](#) and a handler function for when both match.

In this study case, there will be 4 resources: the information about the pregnancy (*users*), appointments (*consultas*), documents (*documentos*) and [FAQs](#) (*faqs*). The defined routes are presented in table [3](#).

##### *Other actions*

The web server also performs other tasks, namely:

- Authentication;
- Connects to the database;
- Serves the zipped version of static assets;

For the authentication, only local authentication was made available. To enable authentication, Passport, an authentication middleware for Node.js, was used. Two strategies were developed to handle signing in and logging in. The strategy that handles signing up is responsible for creating an entry in the *GestacaoSchema* and creating a token. The strategy for logging in checks if the account exists and if the saved password's hash corresponds to the one that the user is

Table 3.: Routes defined in the REST API

URL	HTTP method	Handler function
/users/:id	GET	Gets the complete object that is in the database entry for the ID specified in the <a href="#">URL</a> and returns it
	PUT	Receives an object with information about the user and, after validating it, saves it in the database entry for the specified ID
/consultas/:id	GET	Gets the <i>consultas</i> object that is in the database entry for the ID specified in the <a href="#">URL</a> and returns it
	PUT	Receives a <i>consultas</i> object and, after validating it, saves it in the database entry for the specified ID
/documentos/:id/:idDocumento	GET	Get the file that corresponds to the ID for the <i>GestacaoSchema</i> and the document's ID and returns it
	PUT	Updates the information related to a specific document. It receives a <i>documentos</i> object and maybe a file. After validating the object and checking if a new document file was sent, it updates the database entry for the specified ID and, if the case, deletes the previous file and saves the new one
	DELETE	Deletes the information about the document in the database and the file
/faqs	GET	Gets the <a href="#">FAQs</a> data from the database

entering. If everything is correct, it creates a token for the session and retrieves it. The token is used to verify if the user is logged in the application.

As for connecting to the database and serving the zipped version of assets, these two tasks are smaller, but important nonetheless. Connecting to the database is crucial to be able to save information and retrieve it. Serving the zipped version of static assets was a task added later to the project, but that improved the overall performance of the developed [PWA](#), because it makes it faster. To achieve this, it was added a route that intercepts requests that ask for JavaScript or [CSS](#) files and changes them in order to ask for the zipped version.

### 4.3.3 User Interface

In this section, each component of the UI will be presented as well as the overall organisation. As previously explained, React combined with Bootstrap were used to develop the view part of the application.

The UI can be divided in 4 different areas: the area that does not require authentication (*Perguntas Frequentes* and *Legislação*), the signing up page (*Criar conta*), the logging in page (*Entrar na conta*) and the area that requires authentication (*Dados pessoais*, *Marcações*, *Documentos*, *Perguntas Frequentes* and *Legislação*). The names used for the pages are the same used in the sitemaps previously presented (Figures 11 and 12).

#### *Global Home Page*

When a pregnant woman first enters the application, she is greeted with the page presented in Figure 16. This page has the name of the application (*As 10 luas: gravidez, parto e pós-parto do CMIN*) and the user can choose to create an account (*Criar conta*), log in (*Entrar*) or continue to the website (*Continuar para o site*) which leads to the area that does not require authentication.

#### *Perguntas Frequentes*

The *Perguntas Frequentes* area has 3 different pages according to the category where the FAQ belongs (before the pregnancy, during the pregnancy and after labour). In Figure 17 is the FAQs for the during the pregnancy category. At the top of the page is a tab that when is clicked expands to show additional information, as shown in Figure 18. Then, there is a search bar for the user to search for keywords. Finally, the FAQs are listed. The pages for the other two categories are similar to this one.



Figure 16.: Global Home Page.



Figure 17.: Perguntas Frequentes page.





Figure 18.: *Perguntas Frequentes* page with the tab expanded.

### *Legislação*

In the *Legislação* page (Figure 19) the woman has access to the latest legislation for the maternity and the paternity.

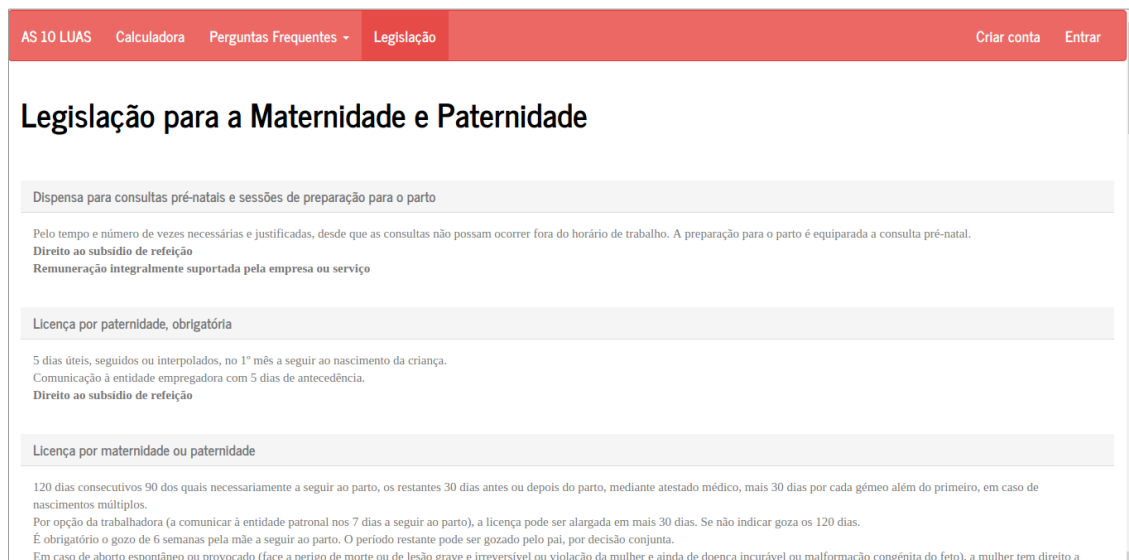


Figure 19.: *Legislação* page.

The *Perguntas Frequentes* area and the *Legislação* page are equal in the authenticated area and in the area that does not require authentication. The only thing that changes is the header, because it is different according to each case. To not

repeat them, these two parts are only shown in the area that does not require authentication, but they also exist in the authenticated area.

### *Criar conta*

The *Criar conta* page (Figure 20) is the page where pregnant women can create an account. It asks for a email and a password. Once the women submits this information, it goes through a validation process to evaluate if what was inserted is a valid email and if the password's length is equal or greater than 8 characters. When something is not correct, an error shows up as presented in Figure 21. If everything goes accordingly, the woman is redirected to a page to add her personal information.

**Criar conta**

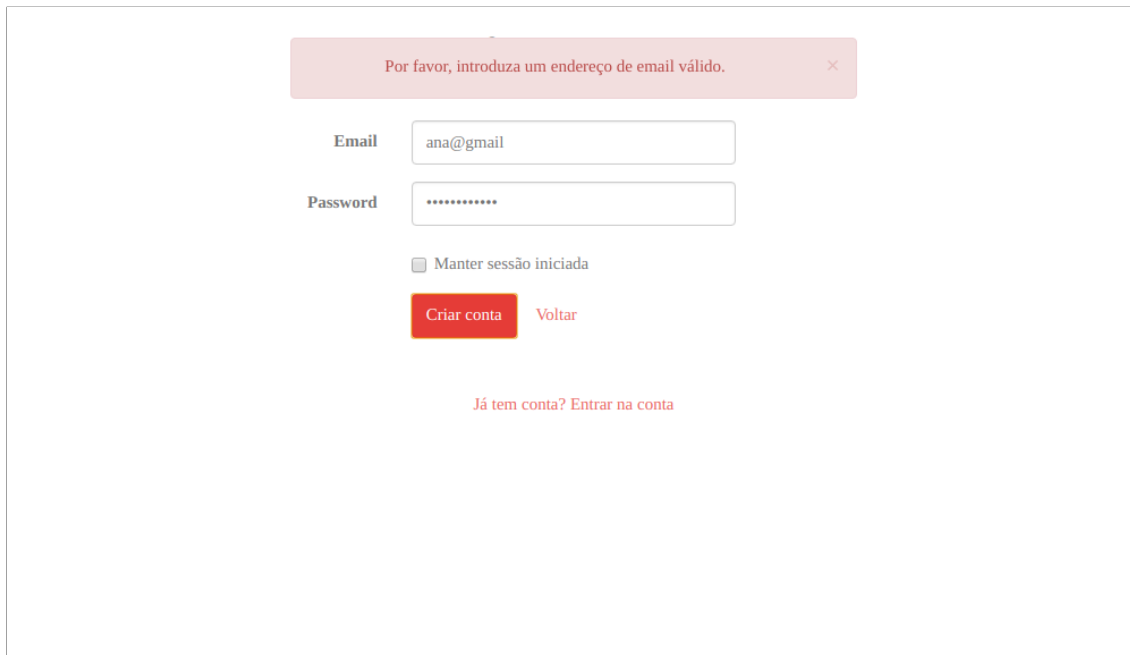
Email

Password

Manter sessão iniciada

[Já tem conta? Entrar na conta](#)

Figure 20.: *Criar conta* page.



The screenshot displays a web form for account creation. At the top, a red error message box contains the text "Por favor, introduza um endereço de email válido." with a close button (X). Below this, the form includes an "Email" field with the text "ana@gmail" and a "Password" field with masked characters "\*\*\*\*\*". A checkbox labeled "Manter sessão iniciada" is present. At the bottom of the form, there are two buttons: "Criar conta" (highlighted in red) and "Voltar". Below the buttons, there is a link that says "Já tem conta? Entrar na conta".

Figure 21.: *Criar conta* page with an error message. In this case, the user did not insert a valid email so a error message shows up when the user tries to create an account.

#### *Entrar na Conta*

The *Entrar na Conta* page (Figure 22) is the page where the pregnant women can enter in her account. It asks for a email and a password as well. The validation process from the previous page is repeated, but with additional validation steps. It is also validated if the account exists and if the password is correct. If something is not alright an error message shows up as in the previous example. If everything goes accordingly, the woman can be redirected to one of two pages. If she has not inserted the required personal information before, she will be redirected to a page to do so, the *Dados pessoais* page. Otherwise, she will be redirected to the *Dashboard*.

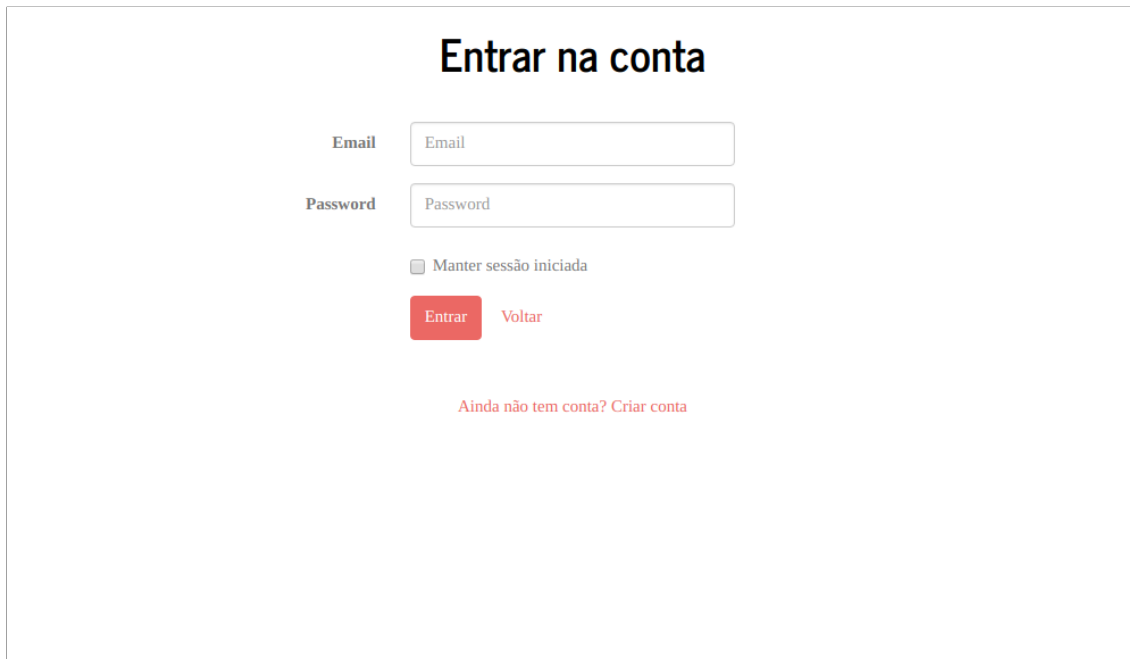


Figure 22.: *Entrar na conta* page.

#### *Dados pessoais*

The *Dados pessoais* page (Figure 23) is the page where the pregnant women enters her and the other parent's personal information. The other parent's information is optional. Once the information is submitted, it goes to the server, where is validated. If everything is correct, it is saved and the user is redirected to the *Ficha clínica* page. The *Ficha clínica* page will not be presented, since it is outside the scope of this dissertation.

#### *Ficha Pessoal*

The *Ficha Pessoal* page (Figure 24) is a place where the woman can access the information she gave during the *Dados pessoais* step after creating an account. In this page, the woman can edit the data (*Editar dados*). When she clicks the button to edit the data, a modal dialog shows up (Figure 25) and the woman has the opportunity to change the recorded data and save it.

Inscrição - Ficha Pessoal

**Dados progenitor**

Nome

Data de nascimento

Profissão

**Dados do outro progenitor**

Nome

Data de nascimento

Profissão

[Seguinte: ficha clínica](#)

Figure 23.: *Dados pessoais* page.

AS 10 LUAS [Ficha Pessoal](#) [Marcações](#) [Documentos](#) [Saúde e Bem-estar](#) [Registos](#) [Outros](#) [Terminar sessão](#)

Os seus dados

Nome	Ana Silva
Data de nascimento	11/11/1989
Profissão	Estudante

[Editar dados](#)

Dados do outro progenitor

Nome	João
Data de nascimento	
Profissão	

[Editar dados](#)

As 10 luas: gravidez, parto e pós-parto do CMIN

Figure 24.: *Ficha Pessoal* page.

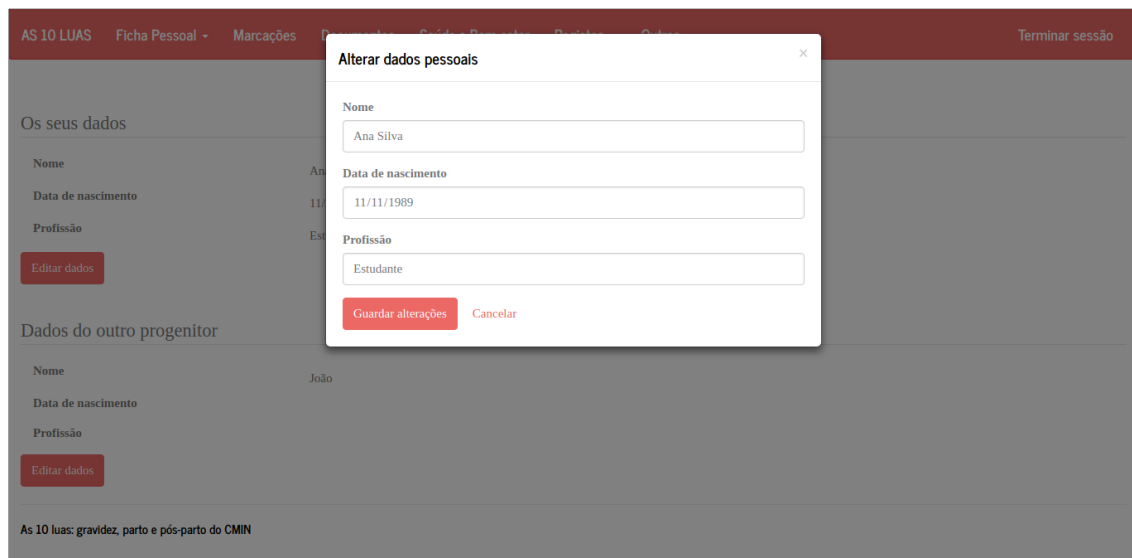


Figure 25.: *Ficha Pessoal* page with an open dialog modal where the woman can edit her personal data.

To understand the impact that Bootstrap had in making the application responsive, in Figure 26 is the *Ficha Pessoal* page in a mobile environment. As one can see, the content is adjusted according to the window's size. The header is also adjusted for mobile use.



Figure 26.: *Ficha Pessoal* page on mobile.

### *Documentos*

In the *Documentos* page (Figure 27) the woman can add documents, download the uploaded documents and delete documents. The uploaded documents entries are organised in a table with the name of the document, the date when the document was uploaded and the category where it belongs.

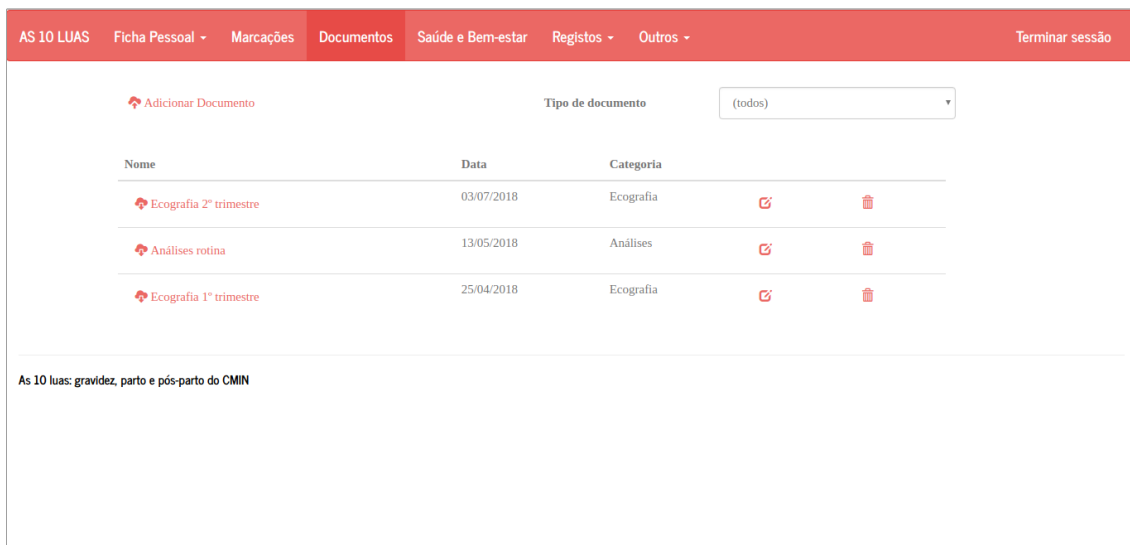


Figure 27.: Documentos page.

Above the table, there is a button to add a document (*Adicionar a Documento*) and a dropdown to filter by category. When the woman clicks the button to add a document, a dialog modal shows up as presented in Figure 28. In this modal, the woman chooses a document to upload, gives it a name and chooses a category. The available categories are: *Ecografia* (ultrasound), *Análises* (analysis) and *Outros* (others).

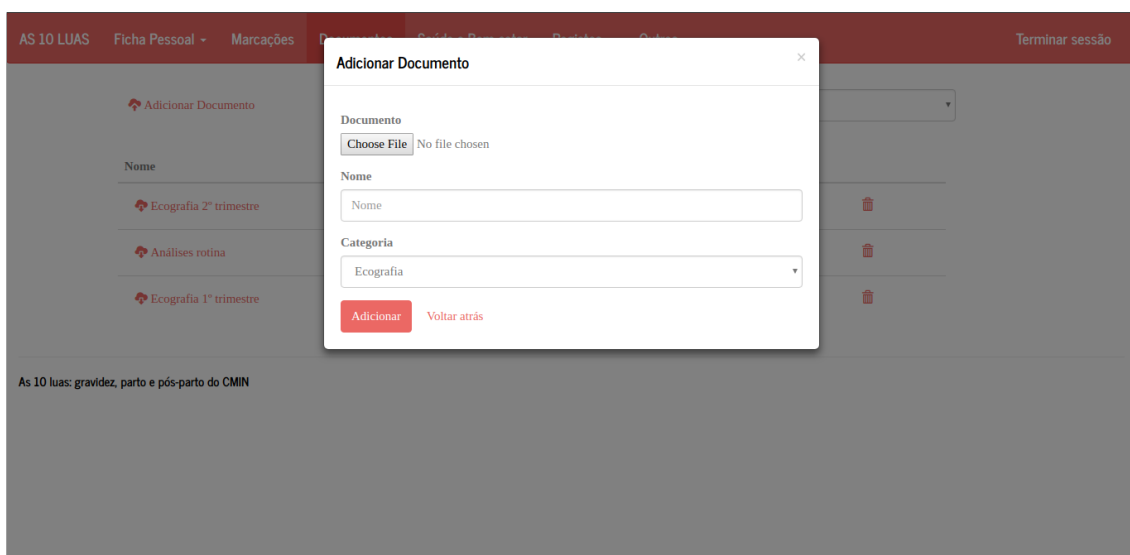


Figure 28.: Dialog modal to add a document.



On the right side of the table with the listed documents, there are two buttons, one to edit the document and another to delete the document. When the woman chooses to edit the document, a similar dialog modal to the one to add a document is presented. When the women decides to delete a document, a message is presented to the woman to ask her if she wants to delete the document. This is shown in Figure 29

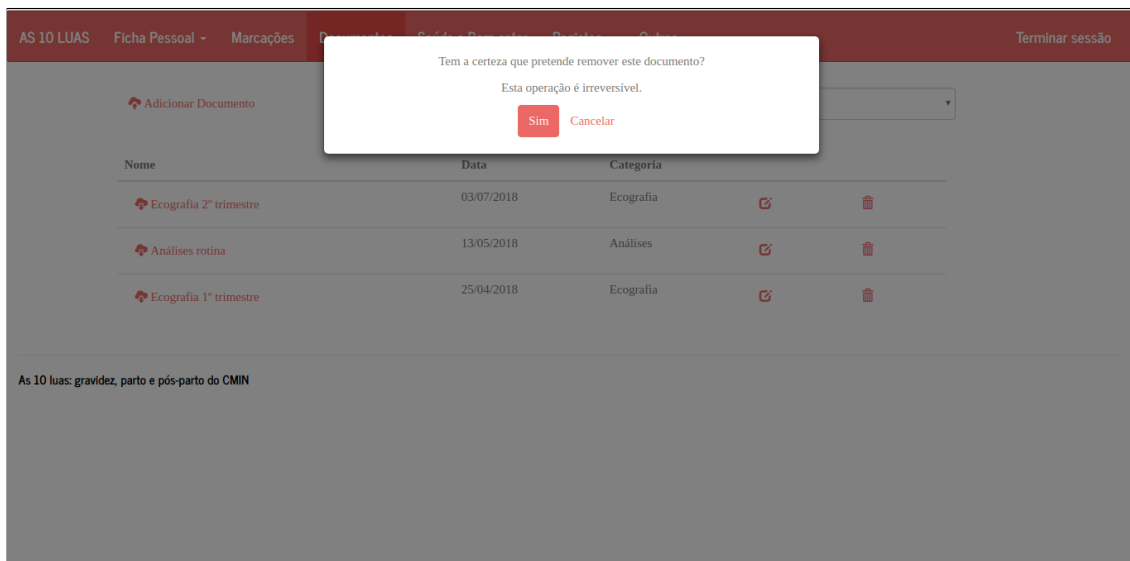


Figure 29.: Message that shows up when the woman clicks the delete button in the *Documentos* page.

### *Marcações*

In the *Marcações* page (Figure 30), on the left side, is a button to add an appointment (*Adicionar Marcação*) and a calendar. On the right side, there is a list with the upcoming appointments (*Próximas marcações*) and the appointments that already happened (*Marcações anteriores*). If the woman clicks the button to add a appointment, a dialog modal is presented as shown in Figure 31. In this modal, the woman specifies the date and hour of the appointment and its name.

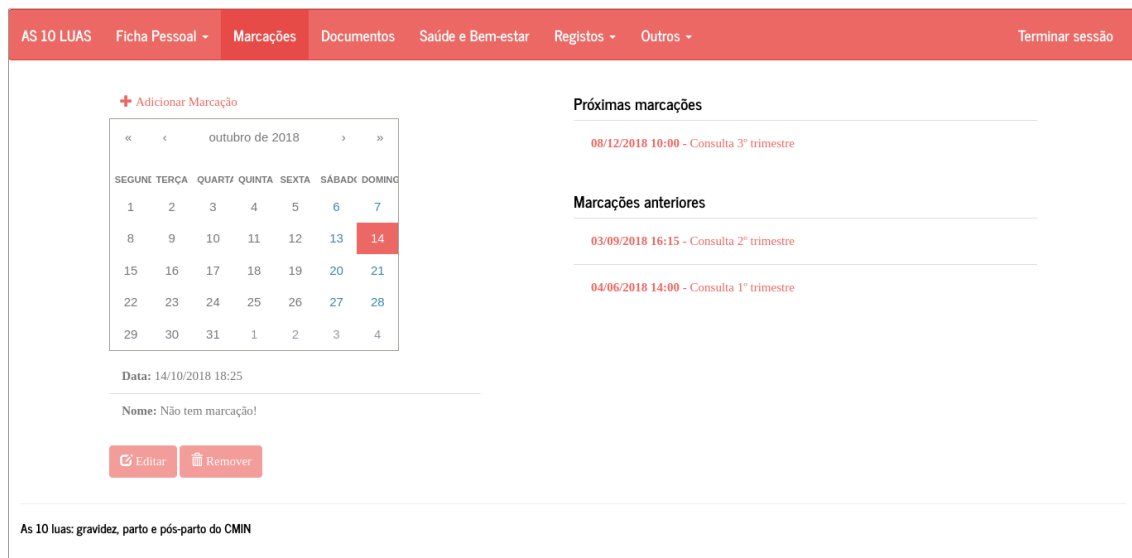
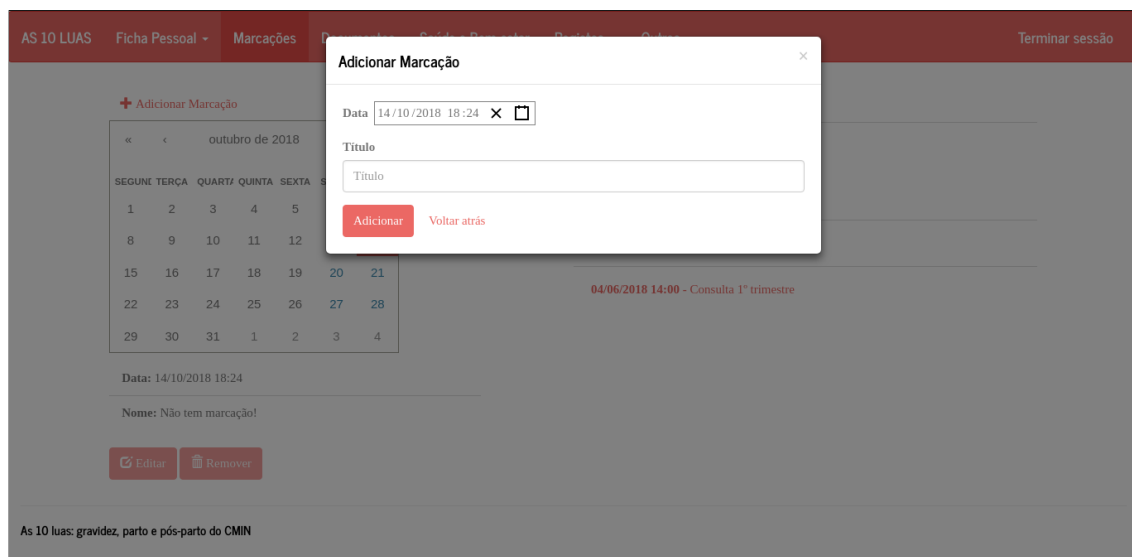
Figure 30.: *Marcações* page.

Figure 31.: Dialog modal to add an appointment.

Each element on the list of the right side is clickable and, when clicked, the day of the appointment is focused on the calendar and the area below is updated. This is shown in Figure 32. Now, the appointment can be edited and deleted, because the buttons are now active. When the edit button (*Editar*) is clicked, a dialog modal shows up to edit the appointment. When the delete button (*Remover*) is clicked, a message shows up to ask the woman if she really wants

to delete the appointment. This message is similar to the one that shows up in the *Documentos* page.

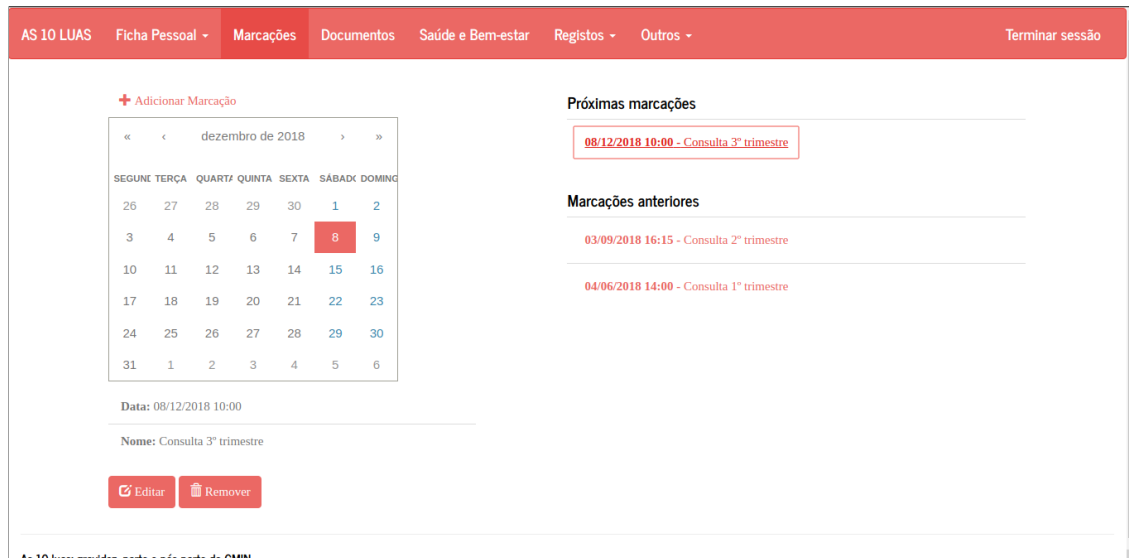


Figure 32.: *Marcações* page when an item in one of the lists is clicked.

#### 4.3.4 Progressive Web App

After developing the base of the application, it was transformed into a **PWA**. According to the concepts presented in 3.4, it was necessary to:

1. Add a service worker;
2. Make it work offline;
3. Add an app manifest.

The developed service worker is responsible for caching assets, such as **CSS** or JavaScript files necessary for the platform to work properly, and the pages. To cache pages, a list of **URLs** is provided to the service worker. When a client makes a request, it is intercepted by the service worker before going to the server. If the request is cached, the service worker responds to the request. If it is not cached, the request is sent to the server. When the client accesses the application after the first utilisation, it will be considerably faster, because it will have everything it needs cached. To make sure that assets are updated, a cache

version number is defined. When the service worker becomes active, it checks if there are old caches. If there are, it deletes the old ones.

The service worker is also responsible for adding some data in IndexedDB. This data consists in the content for the [FAQs](#). This way, this content does not have to be requested each time the user enters the pages and is also available even when there is no Internet connection.

To work offline, it was necessary to make all [URLs](#) work offline and to save the most recent data about the user, so that the user can access this information (e.g. appointments) even when there is no Internet connection. It was also defined that users would only be able to edit data when there is a Internet connection. To make all [URLs](#) work offline, they were precached with the help of the service worker. As for having the most recent user data available, there was some logic inserted that saves the most recent user data and updates it when there is an Internet connection. To make it impossible to perform actions that need to update data, there was some logic responsible for detecting when the user becomes offline and online. If the user is offline, all the buttons that lead to data update are disabled and a warning is presented. Once the user has Internet connection again, the buttons become active again and the warning disappears. An example of the offline capability in action is presented in [Figure 33](#).

Lastly, the app manifest was added, so that the users can add the app to their home screen on mobile, like a native app. For the app manifest, an icon, a name, short name, start url and the display mode (fullscreen) were provided as well as a theme color and a background color.

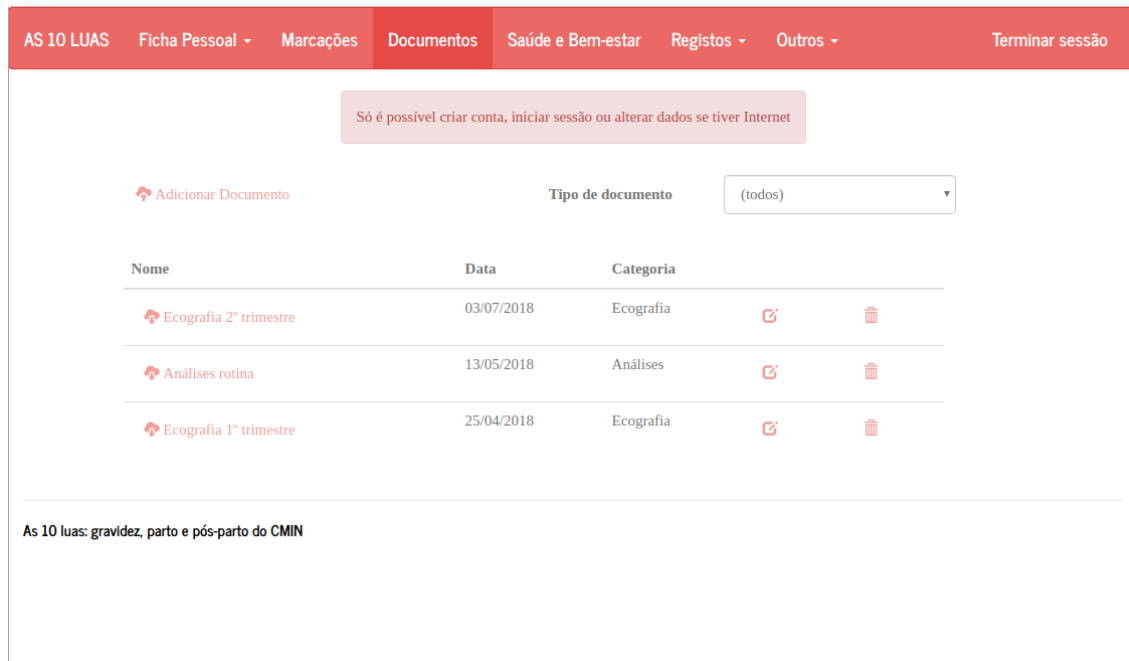


Figure 33.: *Documentos* page when there is no Internet connection. When the user is offline, all buttons that edit or add data are disabled and a warning is presented. Still, the data that was available when the user was online for the last time is displayed.

#### 4.3.5 Administration panel

For the administration panel, KeystoneJS was used. The aim of developing an administration panel was to make the process of updating static content easier. In this dissertation, only the [FAQs](#) content is presented, but there were also other contents in the final product that were available to edit. The information about the users is also available, but cannot be edited.

The first step in the development of the administration panel was to connect it to the database, so that one can view the content saved in the database and edit it. Then, it was necessary to add routes for the [FAQs](#) and the user content. After this step, it was possible to add shortcuts in the initial page and in the navigation bar for this content. The initial page is presented in Figure 34.

Both the user content and the [FAQs](#) content have dedicated pages where it is possible to view the entries that exist in the database and filter them. The page

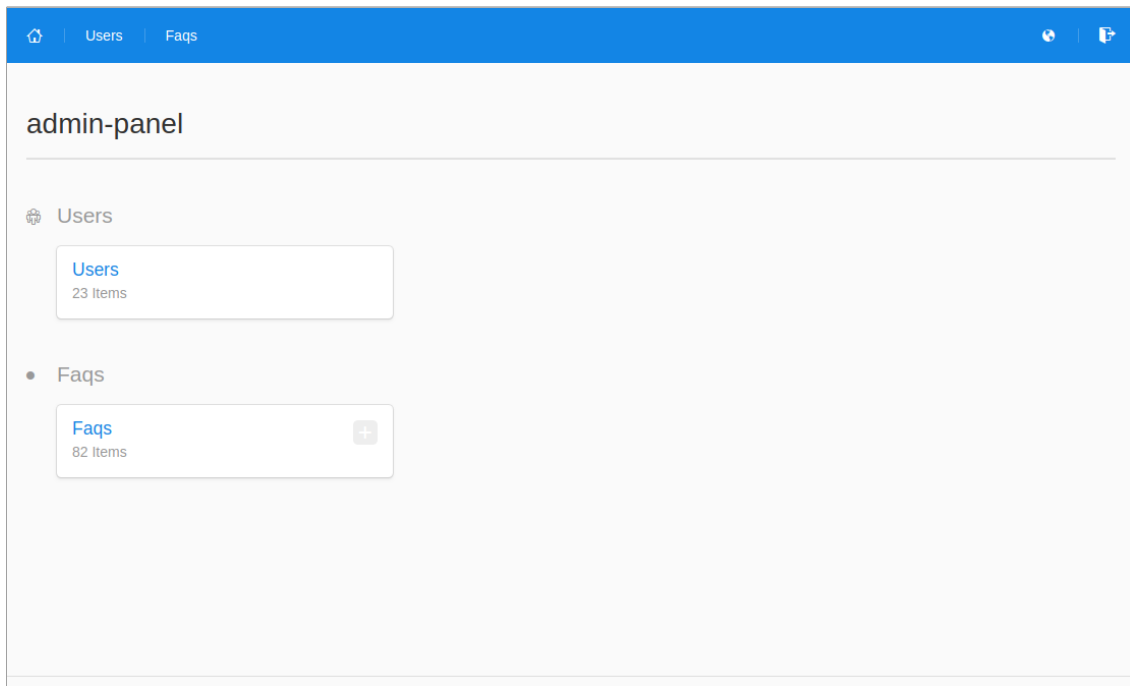


Figure 34.: Initial page of the administration panel.

for the FAQs is presented in Figure 35. In the FAQs case, it is also possible to enter new entries (Figure 36), delete entries and update entries.

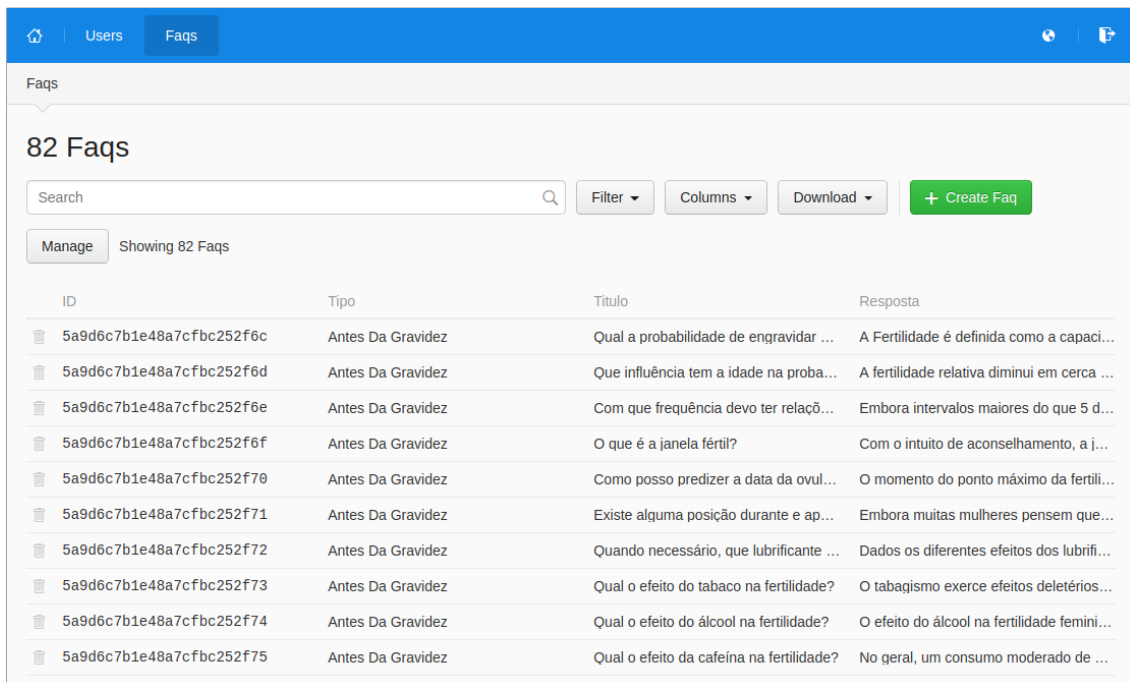


Figure 35.: Page where the FAQs content is managed in the administration panel.

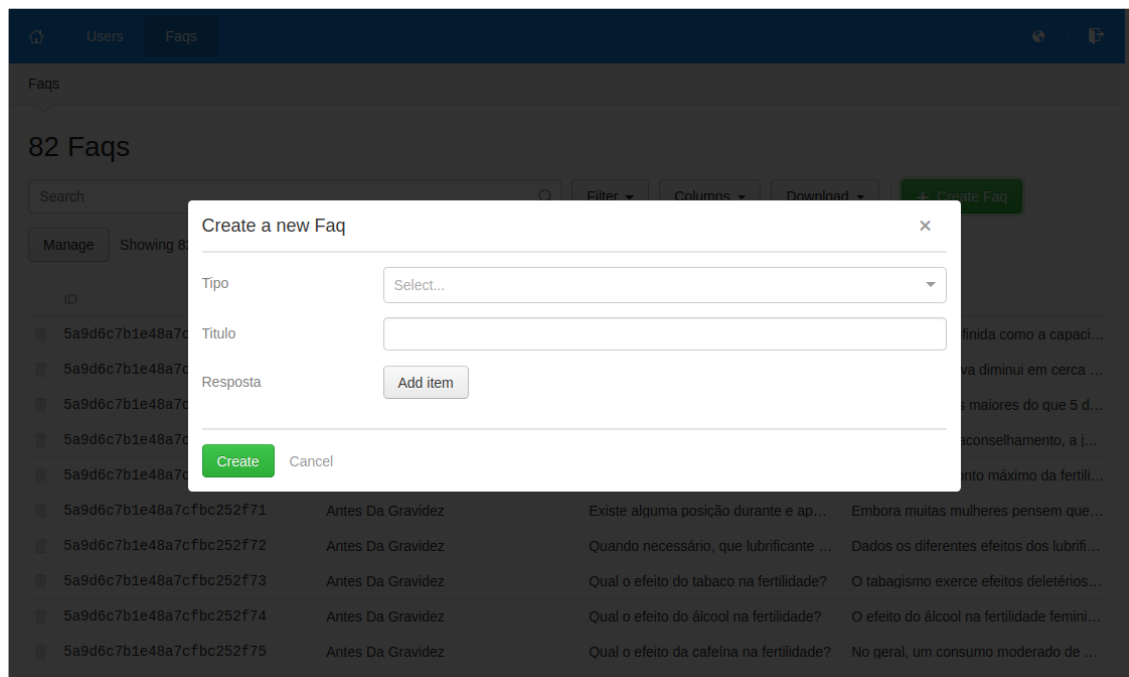


Figure 36.: Modal dialog to add a new [FAQ](#).

#### 4.4 DISCUSSION

The developed application uses modern technologies and methodologies to solve a real necessity that pregnant women have. Even though it is not perfect, it has a set of features that have the potential to engage pregnant women more and be useful in the management of their health.

Browser compatibility is one of the factors that may difficult the adoption of this application. This is not an issue if the user uses Google Chrome, because the main driver behind [PWAs](#) is Google. For other platforms, this issue tends to decrease over time. For example, with the latest release of iOS Safari in last March (version 11.3), service workers are now compatible with Safari.

The application works offline, but there are some features available only with an Internet connection. This could be solved by using the Background Sync [API](#), but it is not available in some browsers (e.g. Safari), so the work that would be applied to enable these functionalities would not pay off at the moment. However, if this [API](#) starts having a more wide spread adoption this may be an interesting area to improve.

This system is available on all devices as long as there is a browser and adapts itself to different screen sizes. On computers, it works as any other website. On mobile, users can either open it in the browser or add the application to the home screen, like a native mobile application from the app stores. In both cases, it works offline and has a wide range of features available. The availability in such a wide range of devices might have a positive impact in the adoption of the developed [PHR](#).

If the user adds the application to the home screen, the access to the application becomes easier and it will look closer to a native one, since it has a splash screen and is displayed in fullscreen. This, combined with the fact that the application is responsive, which makes the application [UI](#) adapt to the screen size, improves the overall [UX](#) and gives a more native look to the application.

The combination of saving content in IndexedDB and using a service worker impacts positively the performance, since it decreases the time spent in retrieving resources.

From a technical point of view, the application is highly scalable, mainly due to the MongoDB database and the [REST API](#) in the server side. This makes the addition of new features an easy and intuitive process.

Since the application was developed taking into account feedback from pregnant women, it was designed with features that they find interesting and useful. This, combined with the fact that a medical institution provided reliable information, has the potential to engage pregnant women more and turn them more active in their pregnancy management, which may impact positively birth outcomes.



---

## PROOF OF CONCEPT

---

All projects in the **IS** field must be evaluated before being openly available. This evaluation process is important to understand if there are any improvements that should be made and if the found solution is suitable for its purpose.

In this dissertation, the proof of concept is made in three different ways. First, a more technical analysis is performed using the Lighthouse tool. Then, a **SWOT** analysis is performed to understand the factors involved and which areas need improvement. Finally, the proposal of a usability test is made.

### 5.1 LIGHTHOUSE REPORTS

As explained in 3.5, the Lighthouse tool evaluates a web site by giving a score in the following parameters: *Performance*, *Progressive Web App*, *Accessibility*, *Best Practices* and *Search Engine Optimisation*.

When performing a Lighthouse audit using the Chrome DevTools, it is possible to choose if the test should be performed in a Desktop environment or in Mobile environment, which throttling mechanisms should be applied during the audit and if the browser's storage should be cleared.

Given the fact that the developed system is a **PWA**, two audits were performed, one that cleared storage and one that did not. This way it is possible to evaluate the performance of the application the first time it is visited (without storage) and in the following times (with storage). For throttling, the *Simulated Fast 3G*,

*4x CPU Slowdown* option was the one chosen and both tests were performed in a Mobile environment. The results are shown in Figure 37 and in Figure 38.



Figure 37.: Lighthouse report main results when the application is not cached, that is, when a user first accesses it.



Figure 38.: Lighthouse report main results when the application is already cached.

From the analysis of both results, it is clear that the two main factors that differentiate one case from the other are the *Performance* and the *Progressive Web App* parameters. The remaining parameters had good scores.

The first time a user enters the application, the service worker caches all the assets it needs to run the application normally without needing to request assets to the web server. This is what makes the performance of the application not so good when the user first uses it. The *Progressive Web App* parameter is also worst, because one of the factors used to evaluate it is how fast the application is and in the first utilisation, as explained before, there are many processes occurring that impact performance, leading to a worse score.

However, when a user visits the application the following times and the cache version has not changed, the browser already has all the necessary assets to run the application. Therefore, it is much faster and performance improves. This is why the *Performance* and *Progressive Web App* parameters are much better in this case. It is also important to explain that the *Progressive Web App* parameter has a score of 92, because the application is not yet using the [HTTPS](#) protocol and this is one of factors evaluated. Otherwise, the application is completely compliant with the rules that define a [PWA](#).

To improve performance in the first utilisation of the application, it would be of great interest to evaluate the impact that some performance techniques, such as lazy-loading, might have and to identify which assets are harming performance the most, so that further studies can be made to understand if they are really necessary or if some performance technique can be applied.

## 5.2 SWOT ANALYSIS

According to the concepts presented in 3.5, a **SWOT** analysis is important to identify the strengths and weaknesses of a product and which are the external opportunities and threats of the developed solution, so that further measures can be defined.

In the study case presented in this dissertation, the main strengths identified are:

- The possibility to use the system in any type of device;
- The offline capability that allows the application to work without an Internet connection after the first utilisation;
- Responsive **UI** that makes the application adaptable to any screen size;
- The fact that it was developed with the support of a medical institution and taking into account the feedback pregnant women provided;
- Wide range of features available;
- Provides trustworthy information;
- Highly scalable due to the combination of a **REST API** and a MongoDB database;
- Easy to use;
- Administration panel to manage content;
- Authentication available, which leads to higher security and privacy.

On the other side, the main weaknesses that the proposed system has are:

- The performance of the application during the first utilisation is poor;
- An Internet connection is necessary to perform actions that lead to changes in the database;
- Documents are for now saved in a rather primitive way;
- Even though security and privacy were a concern during development, they need to be improved;
- When a woman creates an account, she needs to add personal information that she already gave to her obstetrician;
- The documents and appointments areas are not synchronised with the medical institution.

As for now, only internal factors were identified. However, there are external factors that have influence. The main positive ones, the opportunities, they are:

- A more widespread adoption of [PWAs](#) by users;
- [PWAs](#) available in more app stores;
- Increasing interest from patients to monitor their health;
- A pregnancy is a medical condition that needs extensive monitoring;
- Growth of general apprehension in searching for medical information on-line.

The threats, that is, the external factors that have a negative influence are the following:

- There are many alternatives available;
- Some platforms are not fully compatible with [PWAs](#).

### 5.3 USABILITY TEST

A usability test assesses how easy and intuitive it is to use the system under evaluation, as explained in section 3.5. During the scope of this dissertation, there was not enough time to perform a usability test. However, in this section a proposal of a usability test to evaluate the developed features is presented.

The first step in a usability test is to define the purpose of the test. In this case, it is to evaluate if the user can easily use the application and to identify any major problems that may not let the user use the application fully.

Then, it is important to determine which are the characteristics that the participants need to have. In this case, all participants need to be pregnant women, since they are the target public.

After these two steps, the actions that will be performed by users should be defined. Each task will have a maximum time to be completed. If, after that time, the user could not complete the task, the person supervising the test will complete it if it prevents the completion of other tasks, and the user will do the remaining tasks. In this case, the tasks and the maximum times of completion are:

- Create an account (5 minutes);
- Go to one of the pages of the *Perguntas Frequentes* area (3 minutes);
- Go to the *Legislação* page (3 minutes);
- Edit the personal file (5 minutes);
- Add an appointment (5 minutes);
- Edit an appointment (5 minutes);
- Delete an appointment (3 minutes);
- Add a document (5 minutes);
- Edit a document (5 minutes);

- Delete a document (3 minutes).

The user will have access to a physical copy with the aim of the test outlined, the tasks to perform and the time available to accomplish them.

As for the measures that will be recorded, they will be:

- Whether the user completed the task:
  - If yes, the time necessary to accomplish the task;
- How many times the user asked for help.

In the end, the pregnant woman should have the chance to give her opinion about the application. Thus, in the printed test there should be a dedicated area for that use.

After performing the test, all the data recorded needs to be analysed to identify if there are any abnormal behaviours happening and identify pages that need to be improved or reorganised in order to be more intuitive to use.

---

## CONCLUSION

---

After presenting the work developed under the scope of this dissertation, the final result will be overviewed and the guidelines for the next steps of this project are presented.

Thus, in the upcoming section (6.1), a brief summary of the main contributions is made. Followed by section 6.2 were the next steps in this project are explained.

### 6.1 CONTRIBUTIONS

This project had two main goals: to develop a [PHR](#) for pregnant women that would provide a set of features they find useful, while also providing trustworthy information about areas they are interested in, and to evaluate the developed application.

The entire process of developing the application is described, since the requirements analysis to the presentation of the final product, passing through the literature review, choice of the tools and methodologies, development and evaluation. During this project, [105] was published. This paper describes part of the developed work.

To achieve the proposed solution, a web server that implements a [REST API](#) using Node.js and Express, and a [UI](#) using React and Bootstrap, to make it responsive and mobile friendly, were developed. As for the database, MongoDB was chosen due to its flexibility and scalability. The combination of a [REST API](#) and a MongoDB database was what made possible to achieve the *Research Ques-*

*tion 1*. To make the application work in both web and mobile, the [PWA](#) methodology was applied with good results, making the developed system available in a wide range of devices and answering the *Research Question 2*. In addition, an administration panel was also developed, using [KeystoneJS](#), to help in the process of updating the static content displayed in the application.

The *Research Question 3* focused on offering pregnant women a set of features they find useful. Since the developed system has a wide range of features chosen from the analysis of the existing literature on the matter and from the feedback pregnant women provided, one can conclude that this research question was answered. The fact that a medical institution was involved in the development of this system also adds value to the final solution, since the development was supervised by professionals and all the information displayed comes from a trustworthy source that pregnant women recognise as such. This allowed the final solution to offer reliable information to the pregnant women which was the goal of the *Research Question 4*.

Nevertheless, the developed solution is not only important because it gathers a wide set of features, that are usually not found all together, and because it has the support of a medical institution, but also because it was completely developed using modern tools and methodologies. All the technologies used are modern technologies and methodologies, some of them only now starting to show their worth (e.g. [PWAs](#)), and this dissertation successfully shows how they can be applied in the health field.

With this approach, the final solution is a system that works on both web and mobile, works offline, is highly scalable, presents a wide range of useful features and has the support of a trustworthy institution. It also has the potential to increase the engagement of pregnant women and maybe improve birth outcomes.

In the end, the final result was not fully evaluated as aimed. It was not possible to evaluate its usability, but this process will be facilitated, because of the connection to the [CMIN](#). When it comes to performance, given that no special actions were taken to improve it, the final results were still very good. A [SWOT](#)



analysis was also performed to evaluate the designed solution and areas that need attention were identified.

## 6.2 FUTURE WORK

To make this system available for pregnant women as quickly as possible, the next step should be to perform the usability test proposed in 5.3. This test is important to identify areas that need improvement and to adapt the solution to the feedback provided. As previously explained, this step will have the support of the CMIN to contact possible participants and to provide a controlled place to perform the tests.

In the future, after releasing it to the public, it would also be of great interest to study the adoption rates and the impact such a system has in the engagement of pregnant women and in birth outcomes.

Given the way this system was developed, it is possible to increase it and maybe add new features in the future. A possibility that should be taken into account is to add a connection between this system and the medical institution that provides care to the pregnant woman and to create an area for the obstetrician. This would require a great amount of work and research to find the best solutions, but, in the end, it would be advantageous for both women and medical institutions, because information would be shared among each other.

In this case, pregnant women would not have to insert the information they already gave to the medical professionals, and appointments and documents saved in the hospital's system would be displayed in the application to the pregnant woman.

On the other side, medical professionals would have access to the medical parameters recorded in the application and alerts could be set to alert professionals when abnormal values or patterns are detected.

---

## BIBLIOGRAPHY

---

- [1] Karen A Wager, Frances W Lee, and John P Glaser. *Health care information systems: a practical approach for health care management*. John Wiley & Sons, 2017.
- [2] Luciana Cardoso, Fernando Marins, Filipe Portela, Manuel Santos, António Abelha, and José Machado. The next generation of interoperability agents in healthcare. *International journal of environmental research and public health*, 11(5):5349–5371, 2014.
- [3] Alex H Krist and Steven H Woolf. A vision for patient-centered health information systems. *Jama*, 305(3):300–301, 2011.
- [4] Nicola Diviani, Bas van den Putte, Stefano Giani, and Julia CM van Weert. Low health literacy and evaluation of online health information: a systematic review of the literature. *Journal of medical Internet research*, 17(5), 2015.
- [5] Nan Xiao, Raj Sharman, H Raghav Rao, and Shambhu Upadhyaya. Factors influencing online health information search: An empirical analysis of a national cancer-related survey. *Decision Support Systems*, 57:417–427, 2014.
- [6] Susanne E Baumgartner and Tilo Hartmann. The role of health anxiety in online health information search. *Cyberpsychology, behavior, and social networking*, 14(10):613–618, 2011.
- [7] Mariam Bachiri, Ali Idri, José Luis Fernández-Alemán, and Ambrosio Toval. Mobile personal health records for pregnancy monitoring functionalities: Analysis and potential. *Computer methods and programs in biomedicine*, 134:121–135, 2016.

- [8] Chung-Wei Chang, Tien-Yan Ma, Mei-San Choi, Yu-Yun Hsu, Yi-Jing Tsai, and Ting-Wei Hou. Electronic personal maternity records: both web and smartphone services. *Computer methods and programs in biomedicine*, 121(1):49–58, 2015.
- [9] Padaphet Sayakhot and Mary Carolan-Olah. Internet use by pregnant women seeking pregnancy-related information: a systematic review. *BMC pregnancy and childbirth*, 16(1):65, 2016.
- [10] Randy Connolly. *Fundamentals of web development*. Pearson Education, 2015.
- [11] Mozilla. Http, 2018. <https://developer.mozilla.org/en-US/docs/Web/HTTP>, Last accessed on 2018-08-07.
- [12] Mozilla. Http headers, 2017. <https://developer.mozilla.org/pt-PT/docs/Web/HTTP/Headers>, Last accessed on 2018-08-07.
- [13] Mozilla. Http request methods, 2018. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>, Last accessed on 2018-08-07.
- [14] Mozilla. Http response status codes, 2018. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>, Last accessed on 2018-08-07.
- [15] Mozilla. Web technology for developers, 2018. <https://developer.mozilla.org/bm/docs/Web>, Last accessed on 2018-08-07.
- [16] Mozilla. Ajax, 2018. <https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX>, Last accessed on 2018-08-07.
- [17] Mozilla. Introduction to the dom, 2018. [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction), Last accessed on 2018-08-07.
- [18] Mozilla. Mvc architecture, 2018. [https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern\\_web\\_app\\_architecture/MVC\\_architecture](https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture), Last accessed on 2018-08-07.

- [19] Dragos-Paul Pop and Adam Altar. Designing an mvc model for rapid web application development. *Procedia Engineering*, 69:1172–1179, 2014.
- [20] Gaurav Vaish. *Getting started with NoSQL*. Packt Publishing Ltd, 2013.
- [21] Andrew John Poulter, Steven J Johnston, and Simon J Cox. Using the mean stack to implement a restful service for an internet of things application. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 280–285. IEEE, 2015.
- [22] Sanchit Aggarwal and Jyoti Verma. Comparative analysis of mean stack and mern stack. 2018.
- [23] Ivano Malavolta. Beyond native apps: web technologies to the rescue!(keynote). In *Proceedings of the 1st International Workshop on Mobile Development*, pages 1–2. ACM, 2016.
- [24] Tim A Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Progressive web apps: the definite approach to cross-platform development? 2018.
- [25] Samuel Valente, Jorge Braga, José Machado, Manuel Santos, and António Abelha. The impact of mobile platforms in obstetrics. *Procedia Technology*, 9:1201–1208, 2013.
- [26] Ville Ahti, Sami Hyrynsalmi, and Olli Nevalainen. An evaluation framework for cross-platform mobile app development tools: A case analysis of adobe phonegap framework. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, pages 41–48. ACM, 2016.
- [27] Matteo Ciman and Ombretta Gaggi. An empirical analysis of energy consumption of cross-platform frameworks for mobile development. *Pervasive and Mobile Computing*, 39:214–230, 2017.
- [28] Andreas Biørn-Hansen, Tim A Majchrzak, and Tor-Morten Grønli. Progressive web apps: The possible web-native unifier for mobile develop-

- ment. In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST)*, pages 344–351, 2017.
- [29] Nadia Tripp, Kirsten Hainey, Anthony Liu, Alison Poulton, Michael Peek, Jinman Kim, and Ralph Nanan. An emerging model of maternity care: smartphone, midwife, doctor? *Women and Birth*, 27(1):64–67, 2014.
- [30] Ali Idri, Mariam Bachiri, José Luis Fernández-Alemán, and Ambrosio Toval. Experiment design of free pregnancy monitoring mobile personal health records quality evaluation. In *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*, pages 1–6. IEEE, 2016.
- [31] James Bush, Dilek E Barlow, Jennie Echols, Jasmine Wilkerson, and Katherine Bellevin. Impact of a mobile health application on user engagement and pregnancy outcomes among wyoming medicaid members. *Telemedicine and e-Health*, 23(11):891–898, 2017.
- [32] Hugo Peixoto, Manuel Santos, António Abelha, and José Machado. Intelligence in interoperability with aida. In *International Symposium on Methodologies for Intelligent Systems*, pages 264–273. Springer, 2012.
- [33] António Abelha, Eliana Pereira, Andreia Brandão, Filipe Portela, Manuel Filipe Santos, José Machado, and Jorge Braga. Improving quality of services in maternity care triage system. *International Journal of E-Health and Medical Communications (IJEHMC)*, 6(2):10–26, 2015.
- [34] Ana Morais, Hugo Peixoto, Cecilia Coimbra, Antonio Abelha, and Jose Machado. Predicting the need of neonatal resuscitation using data mining. *Procedia Computer Science*, 113:571–576, 2017.
- [35] José Neves, Henrique Vicente, Marisa Esteves, Filipa Ferraz, António Abelha, José Machado, Joana Machado, João Neves, Jorge Ribeiro, and Lúzia Sampaio. A deep-big data approach to health care in the ai age. *Mobile Networks and Applications*, 23(4):1123–1128, 2018.

- [36] Andreia Brandão, Eliana Pereira, Filipe Portela, Manuel Santos, António Abelha, and José Machado. Real-time business intelligence platform to maternity care. In *Biomedical Engineering and Sciences (IECBES), 2014 IEEE Conference on*, pages 379–384. IEEE, 2014.
- [37] Sónia Pereira, Filipe Portela, Manuel F Santos, José Machado, and António Abelha. Clustering-based approach for categorizing pregnant women in obstetrics and maternity care. In *Proceedings of the Eighth International C\* Conference on Computer Science & Software Engineering*, pages 98–101. ACM, 2015.
- [38] Sónia Pereira, Filipe Portela, Manuel F Santos, José Machado, and António Abelha. Predicting preterm birth in maternity care by means of data mining. In *Portuguese Conference on Artificial Intelligence*, pages 116–121. Springer, 2015.
- [39] Francisca Fonseca, Hugo Peixoto, Filipe Miranda, José Machado, and António Abelha. Step towards prediction of perineal tear. *Procedia Computer Science*, 113:565–570, 2017.
- [40] José Manuel Machado, António Abelha, Manuel Santos, Filipe Portela, Eliana Pereira, and Andreia Brandão. Predicting the risk associated to pregnancy using data mining. 2015.
- [41] Deborah Lupton and Sarah Pedersen. An australian survey of women’s use of pregnancy and parenting apps. *Women and birth*, 29(4):368–375, 2016.
- [42] Yeonkyu Lee and Mikyung Moon. Utilization and content evaluation of mobile applications for pregnancy, birth, and child care. *Healthcare informatics research*, 22(2):73–80, 2016.
- [43] Karen M Scott, Gastao A Gome, Deborah Richards, and Patrina HY Caldwell. How trustworthy are apps for maternal and child health? *Health and Technology*, 4(4):329–336, 2015.

- [44] Heather A Grimes, Della A Forster, and Michelle S Newton. Sources of information used by women during pregnancy to meet their information needs. *Midwifery*, 30(1):e26–e33, 2014.
- [45] RAK Kennedy, L Mullaney, CME Reynolds, S Cawley, DMA McCartney, and MJ Turner. Preferences of women for web-based nutritional information in pregnancy. *Public health*, 143:71–77, 2017.
- [46] Maria Bjelke, Anna-Karin Martinsson, Lena Lendahls, and Marie Oscarsson. Using the internet as a source of information during pregnancy—a descriptive cross-sectional study in sweden. *Midwifery*, 40:187–191, 2016.
- [47] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.
- [48] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1), 2004.
- [49] Google Trends, 2018. <https://trends.google.pt/trends/explore?date=today%205-y&q=%2Fm%2F01211vxv,%2Fm%2F0j45p7w>, Last accessed on 2018-10-07.
- [50] Artemij Fedosejev and Adam Boduch. *React 16 Essentials: A fast-paced, hands-on guide to designing and building scalable and maintainable web apps with React 16*. Packt Publishing, 2 edition, 2017.
- [51] AM Vipul and Prathamesh Sonpatki. *ReactJS by Example-Building Modern Web Applications with React*. Packt Publishing Ltd, 2016.
- [52] Juho Vepsäläinen. Introduction to react, 2018. <https://survivejs.com/react/getting-started/introduction-to-react/>, Last accessed on 2018-08-28.
- [53] Facebook. Virtual DOM and internals. <https://reactjs.org/docs/faq-internals.html>, Last accessed on 2018-08-28.

- [54] Facebook. Introducing JSX. <https://reactjs.org/docs/introducing-jsx.html>, Last accessed on 2018-08-28.
- [55] Babel. <https://babeljs.io/>, Last accessed on 2018-08-28.
- [56] Bootstrap. <https://getbootstrap.com/>, Last accessed on 2018-08-28.
- [57] React-Bootstrap. <https://react-bootstrap.github.io/>, Last accessed on 2018-08-28.
- [58] Mike Cantelon, Marc Harter, TJ Holowaychuk, and Nathan Rajlich. *Node.js in Action*. Manning Greenwich, 2014.
- [59] Node.js. About node.js. <https://nodejs.org/en/about/>, Last accessed on 2018-08-30.
- [60] Node.js. The node.js event loop, timers, and process.nextTick(). <https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/>, Last accessed on 2018-08-30.
- [61] Node.js. Overview of blocking vs non-blocking. <https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/>, Last accessed on 2018-08-30.
- [62] Kai Lei, Yining Ma, and Zhi Tan. Performance comparison and evaluation of web development technologies in php, python, and node. js. In *2014 IEEE 17th International Conference on Computational Science and Engineering (CSE)*, pages 661–668. IEEE, 2014.
- [63] Ioannis K Chaniotis, Kyriakos-Ioannis D Kyriakou, and Nikolaos D Tselikas. Is node. js a viable option for building modern web applications? a performance evaluation study. *Computing*, 97(10):1023–1044, 2015.
- [64] Robert Ryan McCune. Node. js paradigms and benchmarks. *STRIEGEL, GRAD OS F*, 11:86, 2011.
- [65] Express. <https://expressjs.com/>, Last accessed on 2018-09-03.
- [66] Pedro Teixeira. *Professional Node. js: Building Javascript based scalable software*. John Wiley & Sons, 2012.



- [67] Express. Hello world example. <https://expressjs.com/en/starter/hello-world.html>, Last accessed on 2018-09-03.
- [68] Express. Routing. <https://expressjs.com/en/guide/routing.html>, Last accessed on 2018-09-06.
- [69] Express. Writing middleware for use in express apps. <https://expressjs.com/en/guide/writing-middleware.html>, Last accessed on 2018-09-06.
- [70] MongoDB. What is mongodb? <https://www.mongodb.com/what-is-mongodb>, Last accessed on 2018-09-11.
- [71] Kyle Banker. *MongoDB in action*. Manning Publications Co., 2011.
- [72] MongoDB. Mongodb node.js driver. <http://mongodb.github.io/node-mongodb-native/>, Last accessed on 2018-09-13.
- [73] Mongoose. <https://mongoosejs.com/>, Last accessed on 2018-09-13.
- [74] Mongoose. Schemas. <https://mongoosejs.com/docs/guide.html>, Last accessed on 2018-09-13.
- [75] Mongoose. Models. <https://mongoosejs.com/docs/models.html>, Last accessed on 2018-09-13.
- [76] Mongoose. Documents. <https://mongoosejs.com/docs/documents.html>, Last accessed on 2018-09-13.
- [77] William Forbes. Why building an admin panel should be in your first sprint, 2017. <https://www.bytelion.com/why-building-an-admin-panel-should-be-in-your-first-sprint/>, Last accessed on 2018-09-19.
- [78] KeystoneJS. <https://keystonejs.com/>, Last accessed on 2018-09-19.
- [79] Manikanta Panati. *Beginning KeystoneJS: A practical introduction to KeystoneJS using a real-world project*. Apress, 2016.
- [80] KeystoneJS. Keystone demo. <http://demo.keystonejs.com/keystone/>, Last accessed on 2018-09-19.

- [81] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.
- [82] Cesare Pautasso. Restful web services: principles, patterns, emerging technologies. In *Web Services Foundations*, pages 31–51. Springer, 2014.
- [83] Ivano Malavolta, Giuseppe Procaccianti, Paul Noorland, and Petar Vukmirovic. Assessing the impact of service workers on the energy efficiency of progressive web apps. In *Mobile Software Engineering and Systems (MOBILESoft), 2017 IEEE/ACM 4th International Conference on*, pages 35–45. IEEE, 2017.
- [84] Google Developers. Your first progressive web app, 2018. <https://codelabs.developers.google.com/codelabs/your-first-pwapp/>, Last accessed on 2018-06-18.
- [85] E. D. Navara, Y. Huang, M. Wojciakowski, Long C., and L. McCormick. Progressive web apps in the microsoft store, 2018. <https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps/microsoft-store>, Last accessed on 2018-09-19.
- [86] Addy Osmani. A tinder progressive web app performance case study, 2017. <https://medium.com/@addyosmani/a-tinder-progressive-web-app-performance-case-study-78919d98ece0>, Last accessed on 2018-09-19.
- [87] Think with Google. The next billion users: trivago embrace progressive web apps as the future of mobile, 2017. <https://www.thinkwithgoogle.com/intl/en-gb/consumer-insights/trivago-embrace-progressive-web-apps-as-the-future-of-mobile/>, Last accessed on 2018-09-19.
- [88] Ross Benes. With new mobile site, forbes boosted impressions per session by 10 percent, 2017. <https://digiday.com/media/new-mobile-site-forbes-boosted-impressions-per-session-10-percent/>, Last accessed on 2018-09-19.

- [89] Google Developers. Olx boosts re-engagement on the mobile web by 250% with a progressive web app, 2017. <https://developers.google.com/web/showcase/2017/olx>, Last accessed on 2018-09-19.
- [90] Google Developers. Twitter lite pwa significantly increases engagement and reduces data usage, 2017. <https://developers.google.com/web/showcase/2017/twitter>, Last accessed on 2018-09-19.
- [91] Addy Osmani. A pinterest progressive web app performance case study, 2017. <https://medium.com/dev-channel/a-pinterest-progressive-web-app-performance-case-study-3bd6ed2e6154>, Last accessed on 2018-09-19.
- [92] Pinterest Engineering. A one year pwa retrospective, 2018. [https://medium.com/@Pinterest\\_Engineering/a-one-year-pwa-retrospective-f4a2f4129e05](https://medium.com/@Pinterest_Engineering/a-one-year-pwa-retrospective-f4a2f4129e05), Last accessed on 2018-09-19.
- [93] Mobify. Lancôme’s engaging progressive web app drives 36% lift in mobile revenue, 2018. <https://www.mobify.com/customers/lancome/>, Last accessed on 2018-09-19.
- [94] Google Developers. Add to home screen, 2018. <https://developers.google.com/web/fundamentals/app-install-banners/>, Last accessed on 2018-06-18.
- [95] Google Developers. Introduction to service worker, 2018. <https://developers.google.com/web/ilt/pwa/introduction-to-service-worker>, Last accessed on 2018-06-18.
- [96] Google Developers. Offline quickstart, 2018. <https://developers.google.com/web/ilt/pwa/offline-quickstart>, Last accessed on 2018-06-18.
- [97] Google Developers. The offline cookbook, 2018. <https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/>, Last accessed on 2018-06-18.

- [98] Google Developers. Lighthouse, 2018. <https://developers.google.com/web/tools/lighthouse/>, Last accessed on 2018-09-22.
- [99] Google Developers. Lighthouse v3 scoring guide, 2018. <https://developers.google.com/web/tools/lighthouse/v3/scoring>, Last accessed on 2018-09-22.
- [100] Eric Bidelman. Building a better web with lighthouse, 2018. [https://developers.google.com/web/updates/2016/12/lighthouse-dbw#new\\_look\\_and\\_feel](https://developers.google.com/web/updates/2016/12/lighthouse-dbw#new_look_and_feel), Last accessed on 2018-09-22.
- [101] Nigel Piercy and William Giles. Making swot analysis work. *Marketing Intelligence & Planning*, 7(5/6):5-7, 1989.
- [102] Alan Sarsby. *SWOT Analysis*. Lulu. com, 2016.
- [103] Melody Y Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys (CSUR)*, 33(4):470-516, 2001.
- [104] J. M. Christian Bastien. Usability testing: some current practices and research questions. *International journal of medical informatics*, 2010.
- [105] Patrícia Loreto, Jorge Braga, Hugo Peixoto, José Machado, and António Abelha. Step towards progressive web development in obstetrics. *Procedia Computer Science*, 141:525-530, 2018.



---

## PUBLICATIONS

---

### A.1 IMPROVING MATERNITY CARE WITH BUSINESS INTELLIGENCE

**Authors:** Patrícia Loreto, Francisca Fonseca, Ana Morais, Hugo Peixoto, António Abelha and José Machado

**Conference:** 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)

**Year:** 2017

**Status:** Published

**Abstract:** The aim of this paper is to develop clinical indicators for obstetrics through the use of Business Intelligence (BI) tools, since valid and reliable clinical indicators can help measuring quality of healthcare services and support decision-making processes. This paper gives an overview of concepts related to Health Information Systems (HIS) and BI, along with some related work to highlight the advantages that BI solutions can bring when applied to healthcare. In this paper is also presented the data warehousing and the ETL process, that was necessary for the development of indicators and which is usually hidden from end users, is described. The indicators were developed using Power BI and were analysed and compared with reference values from both national and international health reports. The discussion of the developed indicators made it possible to measure the quality of the obstetrics service, to identify the problematic areas and to decide whether improvement measures should be taken.

**Keywords:** Health Information Systems (HIS), Interoperability, Business Intelligence (BI), Data Warehousing, Extract-Transform-Load (ETL), Power BI, Indicators, Obstetrics.

#### A.2 STEP TOWARDS PROGRESSIVE WEB DEVELOPMENT IN OBSTETRICS

**Authors:** Patrícia Loreto, Jorge Braga, Hugo Peixoto, José Machado and António Abelha

**Journal:** Procedia Computer Science

**Year:** 2018

**Status:** Published

**Abstract:** The aim of this paper is to develop a Personal Health Record (PHR) for the support of pregnant women. With this goal in mind, concepts such as PHR and their importance in the obstetrics field are overviewed, as well as mobile development strategies. The system was developed with the support of a medical institution and taking into account what pregnant women find useful. The developed app is a Progressive Web App (PWA). This is a recent technology that allows the same app to work on most devices, gives a native feel to it when using on mobile devices and enables offline support. Further testing is necessary to understand the impact that this system may have in the engagement of pregnant women and in birth outcomes.

**Keywords:** Personal Health Records, Obstetrics, Mobile Development, Progressive Web Apps.

#### A.3 PREDICATIVE VAGUENESS IN LUNG METASTASES IN SOFT TISSUE SARCOMA SCREENING

**Authors:** José Neves, Almeida Dias, Ana Morais, Francisca Fonseca, Patrícia Loreto, Victor Alves, Bruno Fernandes, Jorge Ribeiro, Cesar Analide, Filipa Ferraz, João Neves and Henrique Vicente

**Conference:** 6th International Conference on Mining Intelligence and Knowledge Exploration (MIKE 2018)

**Year:** 2019

**Status:** Accepted

**Abstract:** Soft Tissue Sarcomas (STSs) pose a potential risk for the development of lung metastases, which in turn results in a negative prognosis for patients. Presumptions about the occurrence of these abnormalities during STSs treatment would have countless implications for both patients and healthcare professionals as they could increase the efficacy of the treatment and improve overall survival. Prediction is based on a creative Logic Programming, Case Based Reasoning approach to problem solving, that is complemented with an unusual approach to Knowledge Representation and Reasoning, as it takes into consideration not only the data items entropic states but introduces the concept of Vague's Predicate Extension.

**Keywords:** Soft Tissue Sarcoma, Magnetic Resonance Imaging, Logic Programming, Knowledge Representation and Reasoning, Case Based Reasoning, Entropy, Predicative Vagueness.

#### A.4 PREDICTING LOW BIRTH WEIGHT BABIES THROUGH DATA MINING

**Authors:** Patrícia Loreto, Hugo Peixoto, António Abelha and José Machado

**Conference:** 7th World Conference on Information Systems and Technologies (WorldCist'19)

**Year:** 2018

**Status:** Submitted

**Abstract:** Low Birth Weight (LBW) babies have a high risk of developing certain health conditions throughout their lives that affect negatively their quality of life. Therefore, a Decision Support System (DSS) that predicts whether a baby will be born with LBW would be of great interest. In this study, six different Data Mining (DM) algorithms are tested for five different scenarios. The scenarios combine information about the mother's physical characteristics and habits and

the gestation. Results are promising and the best model achieved a sensitivity of 91,4% and a specificity of 99%. Good results were also achieved without considering the gestational age, which showed that the use of DM might be a good alternative to the traditional medical imaging exams in the prediction of LBW early in the pregnancy.

**Keywords:** Knowledge Discovery in Databases, Data Mining, Classification, Decision Support Systems, Low Birth Weight, CRISP-DM.