Universidade de Aveiro Departamento de
2018 Eletrónica, Telecomunicações e Informática

**João Rafael
Duarte de Almeida**

Ferramenta de gestão de protocolos clínicos

Software solution for clinical protocol management

Universidade de Aveiro Departamento de
2018 Eletrónica, Telecomunicações e Informática

**João Rafael
Duarte de Almeida**

Ferramenta de gestão de protocolos clínicos

Software solution for clinical protocol management

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Mestrado Integrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica de Prof. Doutor José Luís Oliveira, Professor Associado com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e co-orientação da Doutora Joana Guimarães Costa, Endocrinologista no Centro Hospitalar Baixo Vouga.

agradecimentos / acknowledgements

Obrigado a todos por tudo... Talvez não seja o suficiente. Certas pessoas merecem muito mais do que um simples obrigado, e certamente este espaço não é suficiente para exprimir toda a minha gratidão.

Ao meu orientador, o professor José Luís Oliveira, gostaria de agradecer esta oportunidade, e por todo o apoio que me deu durante este percurso. Sem a sua ajuda, nada teria sido possível. Também gostaria de agradecer à doutora Joana Guimarães, pela orientação, disponibilidade e simpatia.

Agradeço também a todos os membros do grupo de Bioinformática do Instituto de Engenharia Electrónica e Informática de Aveiro (IEETA), pelo apoio e bom ambiente de trabalho.

Gostaria de agradecer à Olga por ter-me apoiado e incentivado a perseguir este objetivo. Ela sem dúvida foi uma amiga que contribuiu imenso para o sucesso deste trabalho.

Um agradecimento especial ao Bastião, Pratas, João, Malta, Cajús, Jorge e ao Arnaldo pois foram os amigos que mais me acompanharam durante esta jornada.

Por último, mas não menos importante, gostaria de deixar um especial agradecimento à minha família, que me apoiaram da melhor forma que conseguiram.

Aos restantes, que não mencionei... Obrigado a todos por tudo.

Thanks to all for everything.

Palavras-chave

Registo electrónico de pacientes, Sistema de apoio à decisão, protocolos clínicos, endrocrinologia

Resumo

Nos últimos anos, têm sido estudadas diversas metodologias para aumentar a qualidade da execução dos tratamentos oferecidos aos doentes hospitalizados. Foram igualmente desenvolvidos sistemas computacionais para auxiliar a tomada de decisões clínicas.

O objetivo deste trabalho consistiu no desenvolvimento de uma aplicação web para apoiar a execução de tratamentos clínicos, seguindo regras previamente estabelecidas. Estas regras constituem as premissas base que definem o procedimento a aplicar, ou seja, a estrutura do protocolo clínico. Este trabalho teve como principal motivação o tratamento de pacientes com diabetes que são internados ou atendidos em serviços hospitalares não especializados nesta doença. Contudo, para não limitar a sua aplicação a um cenário específico, a solução foi pensada para ser flexível e ser aplicável em qualquer cenário clínico. Esta aplicação segue o modelo cliente-servidor. com base numa arquiteture de microserviços, fornecendo uma interface de utilizador web moderna.

O projeto decorreu em estreita colaboração com o Centro Hospitalar do Baixo do Vouga, tendo como resultado uma solução que auxilia os profissionais de saúde no tratamento de doentes internados, reduzindo o risco de erros e aumentando o controlo e monitorização dos cuidados de saúde.

Keywords

Electronic health records, clinical decision support system, clinical protocols, endocrinology

Abstract

Decision support systems are currently important tools to guide the clinician's decisions and to help on the patient's treatments. These systems have been studied over the last decades, leading to some well-defined best practices for building new solutions.

This project had the objective of building a clinical decision system with a core engine based on predefined rules, which can be customized by end-users. This work had as main motivation the treatment of diabetic inpatients and outpatients, in hospital services others than endocrinology. To keep the solution generic, the system does not depend on any specific patient data, neither on the protocols. This application follows the client-server model. based on a microservice architecture, providing a modern web user interface.

The project was carried out in a close collaboration with the Hospital Center of Baixo do Vouga, resulting in a solution that can assists health professionals in the treatment of patients, reducing errors and providing a better monitoring of health care services.

List of contents

1 Introduction **1**
1.1 Objectives . . . 2
1.2 Outline . . . 3

2 The Motivation: Diabetes Management **5**
2.1 Diabetic inpatients . . . 6
2.2 Hypoglycemia . . . 11
2.3 Surgical diabetic inpatient . . . 12
2.4 Continuous intravenous infusion . . . 16
2.5 Summary . . . 17

3 The Current Methodology **19**
3.1 Clinical Decision Support Systems . . . 20
3.2 CDSS Tools: General Solutions . . . 21
3.3 CDSS Tools: Diabetes Solutions . . . 24
3.4 Systems Comparison . . . 25
3.5 Summary . . . 25

4 System Requirement Analysis **27**
4.1 Challenges . . . 27
4.2 Functional Requirements . . . 28
4.3 Non-Functional Requirements . . . 30
4.4 Use Case Analysis . . . 31
4.5 Summary . . . 34

5 Solution Architecture **35**
5.1 Client-Server model . . . 36
5.2 Server Side Modules . . . 36
5.3 Client Side Modules . . . 40
5.4 Summary . . . 42

6 The System Implementation **43**
6.1 Server Side Implementation . . . 44
6.1.1 Technologies used . . . 45
6.1.2 Web Services . . . 46
6.1.3 Tests . . . 47
6.2 Client Side Implementation . . . 47
6.2.1 Technologies used . . . 48
6.2.2 Web Services Client . . . 50
6.3 Deployment . . . 50
6.4 Interface Components . . . 54
6.5 Summary . . . 60

7 Conclusion **61**

A Web Services **69**

B Application Links **71**

List of figures

2.1 Flowchart of the hypoglycemia protocol 13
2.2 Flowchart of the surgical diabetic inpatient protocol 15

4.1 Use case diagram 32

5.1 System architecture, which follows a client-server model 36
5.2 Architecture overview (including the Server Side on the left, and the Client Side on the right). 37
5.3 Backend architecture. 38
5.4 History model logic. 38
5.5 Database structure for storing dynamically the patient's clinical variables. . . . 39
5.6 Frontend architecture. 41

6.1 System technologies. 44
6.2 Client component architecture. 48
6.3 Comparison between containers and virtual machines. 51
6.4 System deployment logic. 53
6.5 Home page. 54
6.6 Initial dashboard. 55
6.7 Patient information visible to the health professionals. 56
6.8 Backend dashboard. 56
6.9 Backend database management (Configuring new clinical variables). 57
6.10 Protocol creation page (Visualization mode). 57
6.11 Protocol assignments. 58
6.12 Protocol execution (Complex type). a) Starting the protocol execution; b) Third step of the protocol execution. 59
6.13 Protocol execution (Simple type). 60

List of tables

2.1 Example of different sliding scales with the necessary adjust per patient response. 8
2.2 Percentage of long-acting and short-acting insulin, and units for the sliding scale insulin, to adjust the current dosage at the reevaluation moment. 9
2.3 Percentage of long and short acting insulin from the most common insulin products. 10
2.4 Unit of short-acting insulin to administrate according to the blood glucose. . . 14
2.5 Procedures to take when blood glucose is below the target range. 16
2.6 Changes in the infusion rate. 17
2.7 Rate change considering the current clinical state of the patient. 18

3.1 Clinical decision support systems features comparison. 26

List of acronyms

API	Application Programming Interface
BG	Blood Glucose
CASNET	Causal ASsociational NETworks
CAT	Clinical Decision Support Administration Tool
CBMS	Computer-Based Medical Systems
CDSS	Clinical Decision Support Systems
CHBV	Centro Hospitalar do Baixo Vouga
CMS	Centers for Medicare and Medicaid Services
CPOE	Computer-based Physician Order Entry
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
DSS	Decision Support Service
EHR	Electronic Health Records
GESDOR	Guideline Execution by Semantic Decomposition of Representation
GLEE	Guideline Execution Engine
GLIF	Guideline Interchange Format
HbA1c	Glycated Haemoglobin
HL7	Health Level Seven International
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICE	Immunization Calculation Engine
IT	Information Technology
IV	Intravenous
JSON	JavaScript Object Notation
NoSQL	Not Only SQL

NPM	Node Package Manager
ORM	Object-Relational Mapping
PIP	Pip Installs Packages
RBAC	Role-Based Access Control Policies
RDBMS	Relational DataBase Management System
RESTful	REpresentational State Transfer
SOA	Service Oriented Architecture
SQL	Structured Query Language
TCP	Transmission Control Protocol
TDD	Total Daily Dosage
URL	Uniform Resource Locator
vMR	Virtual Medical Record

CHAPTER 1

Introduction

"The beginning is the most important part of the work."
— Plato, The Republic

In the past sixty years, many efforts have been made to automate and improve healthcare processes through technological systems [1]. However, its impact in medical practice remains low, in comparison to other areas.

Therefore, developers and physicians are committed in building better solutions by improving guidance in clinical decisions and preventing medical errors. This commitment has already led to the creation of Computer-based Physician Order Entry (CPOE) systems, which were developed to better manage the orders and preventing human errors [2, 3]. These approaches improved the quality and safety of patients' care [4–6]. However, these solutions, *per si*, are not enough to guide all treatments protocols. This lack led to the creation of Clinical

Decision Support Systems (CDSS), which are computer systems designed to aid clinical decisions of individual patients at the time these decisions are being made [7]. Currently, the CDSS is a key requirement in Electronic Health Records (EHR), as it was established by the Centers for Medicare and Medicaid Services (CMS) [8].

Clinical decision support can be provided by textbooks, teaching, or through other ways. However, these methodologies could lead to misunderstandings or human errors, due to the complexity of some clinical protocols, which contain complex calculations or a large set of iterations. Thus, if used properly, CDSS has the potential to change the way medicine has been practiced. This raises the question: how can a computerized system can help assisting treatments? Within the current state of healthcare systems, it is possible to identify technological advances in some health areas. However, frequently, there are gaps in the treatment management. For instance, when a physician is not available to supervise a specialized patient treatments.So, this monitoring could be performed using a system, which would support the clinical decisions and help guide nurses along the treatments. Indeed, some attempts to develop such systems have already been made. There are some clinical decision support systems designed to perform simple but repetitive tasks, such as recognizing that a laboratory test result is out of a normal range, or that a medication ordered has a dangerous interaction with another one that a patient is already taking [1]. There are, also, some solutions for specific diseases, prepared to guide the treatments [9, 10].

The problem with the above mentioned systems is the lack of adaptability, i.e., they were developed for a specific medical purpose, and its reuse in different contexts is quite difficult. Furthermore, there is no open source solution, generic enough, to be adapted to any situation, i. e., a generic clinical decision support system.

This dissertation proposes a generic solution to support and guide treatments independently of the disease. Additionally, as a proof of concept, a demo installation was made to guide diabetes treatments.

1.1 Objectives

The purpose of this dissertation was to investigate a new software solution to guide and manage clinical protocols, independently of the medical context.

In a first phase, a detailed analysis was made to understand how a clinical decision support system is defined. Then, all the necessary protocols for the application scenario have been described. This step helped in the identification of the requirements for our solution. After this analysis, the solution architecture was studied and designed, followed by its implementation.

All these steps resulted in a set of goals, which can be expressed in the following order:

1. develop a solution capable to fulfill the application scenario - diabetes treatment management;

2. assure that the solution is adaptable to different medical scenarios;
3. and implement the protocols used in the Centro Hospitalar do Baixo Vouga (CHBV).

1.2 Outline

This dissertation is organized into six more chapters, which are briefly described below.

The Chapter 2 describes in detail the protocols that led to this research. This chapter contains detailed medical and technical information needed to implement the protocols used in the proof-of-concept scenario.

Chapter 3 aims to provide a state-of-the-art description related to this work. With the intention to solve this dissertation's goal, some solutions and different approaches were studied and evaluated.

The following chapter, chapter 4, analyses the major challenges of the system's conception. It details all the essential requirements, which results in functional and non-functional requirements.

Chapter 5 describes the solution's architecture as an answer to the requirements mentioned before. This detailed solution follows a client-server model, with a micro-kernel architecture.

In chapter 6 the final solution is presented the system's model and the specifications from the server and client sides, are discussed. Finally, a brief description of the solution's deployment and the automated tests, to assure the quality of the system is also contemplated.

The last chapter presents the conclusions pointing out some future developments that can improve the system.

CHAPTER 2

The Motivation: Diabetes Management[1]

"The greatest wealth is health."
— Virgil

Hyperglycemia is a health condition characterized by abnormally high blood glucose, typically caused by a deficient usage, or lack, of insulin. Due to the metabolic derangements of this clinical condition, its regular monitoring, as well as the administration of the most effective treatment, are major concerns for healthcare institutions.

The control of inpatient hyperglycemia is a special concern in medical institution systems. Inpatient hyperglycemia is an event that occurs frequently, with a rate of approximately 40%

[1]This chapter is mainly based on the publication *Simplifying the digitization of clinical protocols for diabetes management, 2018 IEEE 31th International Symposium on Computer-Based Medical Systems (CBMS) [11]*. All the protocols presented have been discussed with Dra Joana Guimarães, an endocrinologist at the CHBV

of all hospitalizations, a quality metric that deserves a special attention from health care institutions and public health services [12].

To manage the hyperglycemia of hospitalized diabetic patients, a basal-bolus insulin therapy is generally recommended [13]. However, this therapy is related to the high rates of hypoglycemia, reaching values up to 32%. The main reason for this occurrence is that the meal insulin and food intake do not match [14]. As a result, it is possible to recognize that most of the adverse medication occasions and blunders happen when insulin is prescribed or administered. These cases of hypoglycemia in non-intensive care unit settings are a concern because they have been associated with increased length of stay, hospital complications, and mortality [15]. To reduce these high rates, several protocols were proposed for glycemic administration [16]. However, these procedures are normally available on paper and difficult to follow, which hinders its regular use by non-trained professionals.

The goal of this research is to reduce this numbers using a system to support the execution of the protocols. With a proper system, some calculation errors, data analysis to perform these calculations and misunderstood instructions can be avoided. The main objective is to implement some protocols, already in use in the hospital and to enable the implementation of new protocols, which provide more accurate dosages of insulin, in order to hide the protocols' complexity from health professionals. Thus, in the subsequent sections, the following protocols are described in detail:

1. Diabetic inpatients [17]
2. Hypoglycemia
3. Surgical diabetic inpatient
4. Continuous intravenous infusion [18]

2.1 Diabetic inpatients

The protocol *Diabetic inpatients* should be applied to inpatients with type two diabetes. However, this protocol has two different executions, depending on the patient clinical state. The first execution will be addressed to the patient if, in the admission moment, s/he is on oral hypoglycemics. The execution details in case the patient is on insulin prior to admission, will be described posteriorly.

However, the measurements schedule for this two different types of patients is similar. The factor that can influence this schedule is the patient's diet. For instance, if the patient is eating, the measurements should be done before meals and at bedtime. On the other hand, if the patient is on an zero diet, i.e. the patient does not eat, measurements should be done every 4 hours. In both cases, some target values have been defined, such as the blood glucose, which should be between 140 and 180 mg/dl and the HbA1c, that should not be higher than 8%.

This protocol has two main stages: 1) the starting phase, where the patient starts the treatment and is only done at the beginning of the protocol execution; and 2) the reevaluation phase, which happens time to time, normally in intervals of 48 until 72 hours. In the study carried out within this dissertation, the reevaluation phase happened every 72 hours of observations.

In the starting phase, which duration is 72 hours, the HbA1c and the blood glucose are measured. Regarding these values, if the HbA1c value of the patient is greater than 8% or if his blood glucose average is greater than 180 mg/dl, the basal-bolus insulin must be started. The insulin dosage is calculated following some rules. Firstly, the insulin's Total Daily Dosage (TDD) for a patient is calculated, considering his weight. However, in some situations, the weight is not available, so there are some default values. The weight information is key, so that the insulin dosages can be more precise.

The insulin's TDD for patients with the right weight information in the system, are 0,4 units/kg when the fasting blood glucose value is less than 200 mg/dl, and 0,5 units/kg otherwise. The resulting value will be used as a reference for the basal or long-acting insulin dosages, which in this hospital will be INN-insulin glargine.

This long-acting insulin is administered at bedtime or before breakfast, and the dosage is half of the insulin's TDD. The moment at which this insulin is administrated depends on the bedtime and the breakfast measurement values. If the blood glucose is higher at the bedtime comparing to the breakfast value, this administration is done before breakfast. Otherwise, it will be done at bedtime.

The prandial insulin, which is $\frac{1}{6}$ of the insulin's TDD, is administrated with the three daily meals. Each dosage is one-third of the total prandial insulin and it should be administrated before the meals. This insulin is a short-acting type, and this hospital uses the Lispro.

When the patient's weight information is missing, the units of long-acting insulin should be 10 units, if the blood glucose value is greater than 180 mg/dl. If the blood glucose value is grater than 280 mg/dl an additional 10 units should be added, establishing a total of 20 units of long-acting insulin.

Additionally, supplemental or "sliding scale" insulin should be administrated to the patient. This insulin should be the same type as the prandial insulin, i.e., it should be insulin Lispro. This insulin dosage should be administrated only before the meals, since at bedtime it can cause hypoglycemia in patients on long-acting insulin. This kind of insulin should, also, be adjusted considering the patient's degree of insulin resistance. Usually, lean patients are more insulin-sensitive and obese patients are more likely to be insulin resistant. However, this does not necessarily apply in all situations. Table 2.1 presents the units to be administrated to patients considering the plasma glucose and the patient resistance.

Table 2.1: Example of different sliding scales with the necessary adjust per patient response.

Plasma glucose	**Sensitive**	**Usual**	**Resistant**
<150 mg/dl	0 units	0 units	0 units
151 - 200 mg/dl	1	2	4
201 - 250 mg/dl	2	4	7
251 - 300 mg/dl	3	6	10
301 - 350 mg/dl	4	8	13
>350 mg/dl	5	10	16

In the reevaluation phase, it is necessary to adjust the long-acting insulin, as well as the pre-meal and sliding scale insulin. For the long-acting insulin, the adjustments are made according to average values of the blood glucose observed before breakfast. Table 2.2 presents the percentage of insulin that should be added (or subtracted) to the current dosage, considering the resulting average referenced above.

The same principle is applied in the sliding scale insulins. However, in this case, units are added or subtracted, accordingly to the average blood glucose observed before breakfast.

Finally, additional conditions should be considered. For instance, if the blood glucose is high (greater than 180 mg/dl) at lunch and normal at dinner, the pre-breakfast short-acting insulin should be increased, according to the table's percentages. In addition, if the blood glucose is high at bedtime, the pre-dinner short-acting insulin must be increased.

All the described steps and conditions only applied to patients, that in the admission moment, are on oral hypoglycemics. In what follows, the additional conditions of this protocol for patients on insulin prior to admission, is analyzed.

Firstly, the insulin used by the patient at home needs to be converted to the ones used in the hospital. These conversions will be analyzed later in this section. Then, adjustments to the insulin dosages taken by the patient at home must be made when s/he is admitted to the hospital. These patients may take only long-acting insulin dosages at home, or may taken both, short and long-acting insulin dosages.

For patients which only take long-acting insulin, the dosage should be decreased in the hospital by 25%, if s/he is on oral diet, and 50% in zero diet situations. If a patient takes both, long and short-acting insulin, without being mixed, the long-acting insulin will follow the same pattern described above. The short-acting insulin will be reduced to 25% on oral

Table 2.2: Percentage of long-acting and short-acting insulin, and units for the sliding scale insulin, to adjust the current dosage at the reevaluation moment.

Plasma glucose	**Percentage of insulin**	**Units for sliding scale insulin**
<70 mg/dl	Call endocrinologist	
71 - 120 mg/dl	- 10%	- 2 units
180 - 230 mg/dl	10 %	2
231 - 280 mg/dl	15 %	4
281 - 330 mg/dl	20 %	6
>330 mg/dl	25 %	8

diet and removed if the patient is in zero diet.

However, some patients may have been taking biphasic insulin, which is a premixed insulin, with prandial and basal aspects of glucose regulation [19], normally with different percentages of long and short-acting insulins. In addition, the percentage of each can change depending on the brand. For these situations, it is necessary to interpret and calculate the insulin's TDD for each (long and short-acting insulin) that the patient is taking at home, adding all the daily administration dosages. Then, the dosage to be administrated in the hospital will be $\frac{2}{3}$ of the resulting values. In case of decimal values, the dosage is rounded down.

The administration schedule for this situations, for the long-acting insulins, is at the same moment the patient takes the biggest dosage at home, i.e., if the patient takes more insulin units ate breakfast, in the hospital s/he receives his/her dosage at that moment too. In the case of short-acting insulin, the resulting value will be divided into the three daily meals in the hospital. The conditions described above are exemplified next.

Example: Consider a patient taking Mixtard 30 Penfill dosages. This insulin contains 70% of long-acting and 30% of short-acting insulin [20], and consider that this patient is taking 30 units before breakfast and 15 units before dinner. So, his daily dosage insulin is 45 units, and because only 70% of this units are long-acting insulin, the patient will need $\frac{2}{3}$ of 31,5 units of long-acting insulin at the breakfast. The short-acting insulin will be the $\frac{2}{3}$ of the remaining daily dosage, which is 13,5 units, divided by the three daily meals into equal portions. Additionally, the sliding scale insulin needs to be administrated, but this will no be detailed. To easily understand the detailed conversions, table 2.3 describes the amount of each insulin type for the most popular biphasic insulin.

Table 2.3: Percentage of long and short acting insulin from the most common insulin products.

Insulin products	**Long-acting insulin**	**Short-acting insulin**
Mixtard 30 Penfill	70%	30%
Insuman Comb 25	75%	25%
Humulin M3	70%	30%
NovoMix 30	70%	30%
Humalog Mix 25	75%	25%
Humalog MIx 50	50%	50%

There are two more different insulin types that need to be converted to the long or short-acting insulin. In the case of short/neutral insulin, this is converted to the short-acting insulin used in the Hospital, and the dosage will be $\frac{2}{3}$ of the dosage the patient takes at home. These dosages will be administrated at the same moment the patient takes them at home. The following insulin products are used in this situations:

- Actrapid Penfill
- Insuman Rapid
- Humulin Regular

Finally, the isophane insulins, which are intermediate-acting insulins, will be replaced by long-acting insulin. The procedure is similar to some previously described. The amount of insulin taken by the patient at home will be added, then $\frac{2}{3}$ of the resulting value is administrated at the same time the patient takes the biggest dosage at home.

Finishing the protocol description, from a medical point of view, some more information is necessary to perform its execution in the system. A particular aspect is the clinical variables for each patient. Therefore, for this particular protocol, the following variables are necessary:

- **BG:** The blood glucose variable is normally collected in each interaction with the patient. Its value is used to define the insulin dosage for the patient.

- **HbA1c:** This widely used variable may have the same amount of records as BG. Similarly to the blood glucose, the value of this variable is used to decide if the patient needs insulin.
- **Diet:** The variable diet, in this case, only defines if the patient is eating or not.
- **Weight:** The variable weight is optional; However, as already mentioned, this information will provide more precise insulin dosages.
- **Insulin resistiveness:** The sliding scale to have more precise dosages, can provide different values according to the patient insulin resistiveness. Therefore, the value of this variable defines if the patient is sensitive, usual or resistant to the insulin.
- **State prior to admission:** This state defines if the patient was on oral hypoglycemics or on insulin when admitted to the hospital.
- **Insulin used at home:** This variable is optional, and indicates the insulin product that the patient uses at home. Using table 2.3 and this information, the right dosage can be defined based on the insulins in use at the hospital.
- **Insulin dosage used at home:** This variable depends on the previous one and is only used if the patient is taking insulin at home.
- **Percentage of long-acting insulin taking at home:** Although the system is already configured with some insulin products and their percentages, this variable, which is optional, can be useful in special situations, or when the patient is only taking long-acting insulin at home.
- **Percentage of short-acting insulin taking at home:** As the previous clinical variable, this variable has the same role, but in this case for the short-acting insulin.

However, this does not mean that these variables are confined only to this protocol. These variables can be reused in other protocols, but, it is worth noting, that without these variables, the protocol cannot be executed properly.

2.2 Hypoglycemia

This protocol for hypoglycemia applies to patients with hypoglycemia. This is a fast-acting protocol and the measurement's periodizations are quite short, i.e., measurements should be made every 15 minutes if the blood glucose patient is less than 80 mg/dl.

The protocol methodology only requires the knowledge of the patient's diet and his blood glucose values. When the patient is able to eat, he receives one sugar packets (which is about 6 grams of sugar) if the blood glucose value is between 50 and 70 mg/dl. However, if this value is less than 50 mg/dl, the patient receives two sugar packets. A patient on a zero diet receives half of a 30% IV glucose ampoule and glucose serum 5%, when the blood glucose value is between 50 and 70 mg/dl, and a 30% IV glucose ampoule and glucose serum 5% when it is less than 50 mg/dl.

Moreover, the application of this protocol suggests that there is something wrong with the patient. Therefore, the endocrinologist should be informed because some adjustments in the current therapeutic scheme, may be necessary.

Concerning the protocol description from a medical point of view, its description lies in what has been described previously. From a technical perspective, the following required clinical variables were defined for this protocol:

- **BG:** In this protocol, this variable indicates if the protocol should proceed or not. When it's value is less than 80 mg/dl, the application of this protocol is required.
- **Diet:** The diet type is a condition that decides what the nurses should do in the treatment. However, this information is only introduced into the system a few times during the protocol execution, since this value does not change as frequently as the blood glucose values.

In Figure 2.1 this protocol is represented, and all the steps followed by the system in its execution. All these steps are not perceptible for the final user. S/he only introduces the necessary data and receives a set of instructions. However, this representation helps the non-medical people to understand the protocol's logic.

2.3 Surgical diabetic inpatient

The next protocol applies to the surgical diabetic inpatient. This protocol influences the surgery realization. Initially, it is verified if the patient's HbA1c is higher than 8.5%, in which case the possibility of postponing the surgery should be considered. The blood glucose measurements before and after the surgery must be made before meals and before bed. Furthermore, the oral hypoglycemics must be interrupted in the day of the surgery. In case of metformin, the oral hypoglycemics is always suspended 48 hours after the IV contrast administration.

There are three different scenarios for this protocol. When the surgery is minor and the patient is diabetic type 2, with a good metabolic control, the following recommendations should be applied:

- suspend the morning antidiabetic;
- avoid glycosylated serum;

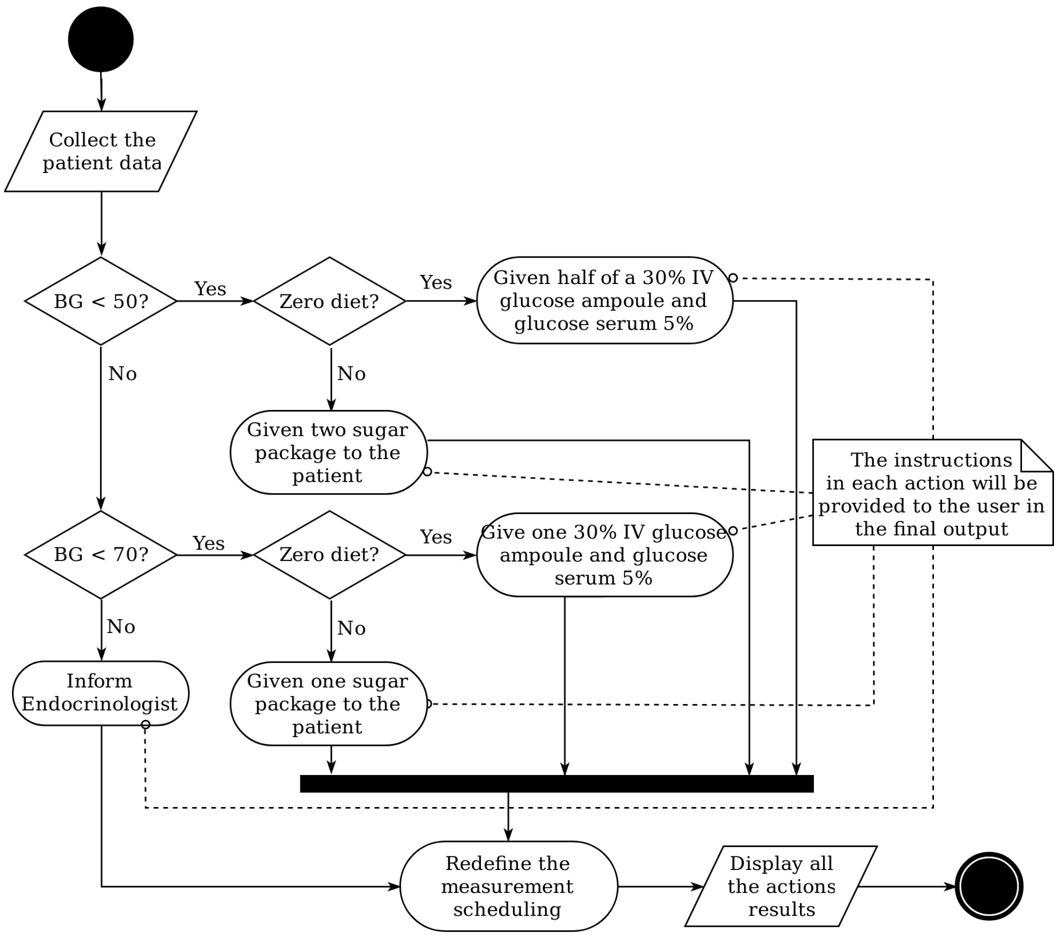

Collect the patient data
BG < 50?
Yes
No
Zero diet?
Yes
No
Given half of a 30% IV glucose ampoule and glucose serum 5%
Given two sugar package to the patient
The instructions in each action will be provided to the user in the final output
BG < 70?
Yes
No
Zero diet?
Yes
No
Give one 30% IV glucose ampoule and glucose serum 5%
Inform Endocrinologist
Given one sugar package to the patient
Redefine the measurement scheduling
Display all the actions results

Figure 2.1: Flowchart of the hypoglycemia protocol

- perform blood glucose measurements every 4 hours and administrate sliding scale insulin if needed, the dosages are represented in table 2.4;
- and restart the antidiabetic when the patient starts eating.

Concerning minor surgeries with a type 1 diabetic inpatient with good metabolic control, the precautions that should be taken are a little different. Below is a description of all the actions needed:

- administrate half of the normal isophane/intermediate/short-acting insulin dosages in the surgery morning. However, if it was administrated the night before, the dosage is kept;
- do not administrate short-acting insulin in the morning of the surgery;
- use glucose serum at 5%;
- administrate the sliding scale insulin every 4 hours, as represented in table 2.4;

Table 2.4: Unit of short-acting insulin to administrate according to the blood glucose.

Plasma glucose	**Short-acting insulin units**
140 - 180 mg/dl	2 units
181 - 220 mg/dl	4
221 - 260 mg/dl	5
261 - 300 mg/dl	6
301 - 340 mg/dl	7
341 - 380 mg/dl	8
>380 mg/dl	10

- and perform glucose measurements every 2 hours.

Finally, when the surgery is major, the patient is diabetic type 1 with bad metabolic control, or did not eat for more than 12 hours, the protocol applied is the continuous intravenous infusion, detailed in section 2.4.

Analyzing the technical perspective of this protocol, the required clinical variables for this implementation and execution, were defined. The following list details these variables, explaining their usefulness:

- **BG:** As can be noticed, this variable is present in all protocols. After all, the insulin dosages depends on these values.
- **HbA1c:** This variable helps to understand the patient's clinical state. In some situations, the value of this variable could influence a surgery delay.
- **Diet:** The diet, like before, only to defines if the patient is eating or not.
- **Metformin:** This variable only indicates if the patient is taking this kind of oral hypoglycemics.
- **Surgery type:** The protocol depends on the surgery type, as previously described. So, this variable indicates if the surgery is major or minor.
- **Diabetic type:** The diabetic patient's type is a non-optional variable. There are several procedures that are taken depending on this value.

Figure 2.2 describes, in a flow chart, the defined protocol. By following all the steps in this workflow, the system should provide a result to the user. However, there are several conditions that lead to another protocol execution. In this case, the system should not execute the new protocol automatically, but it should inform the user about the situation.

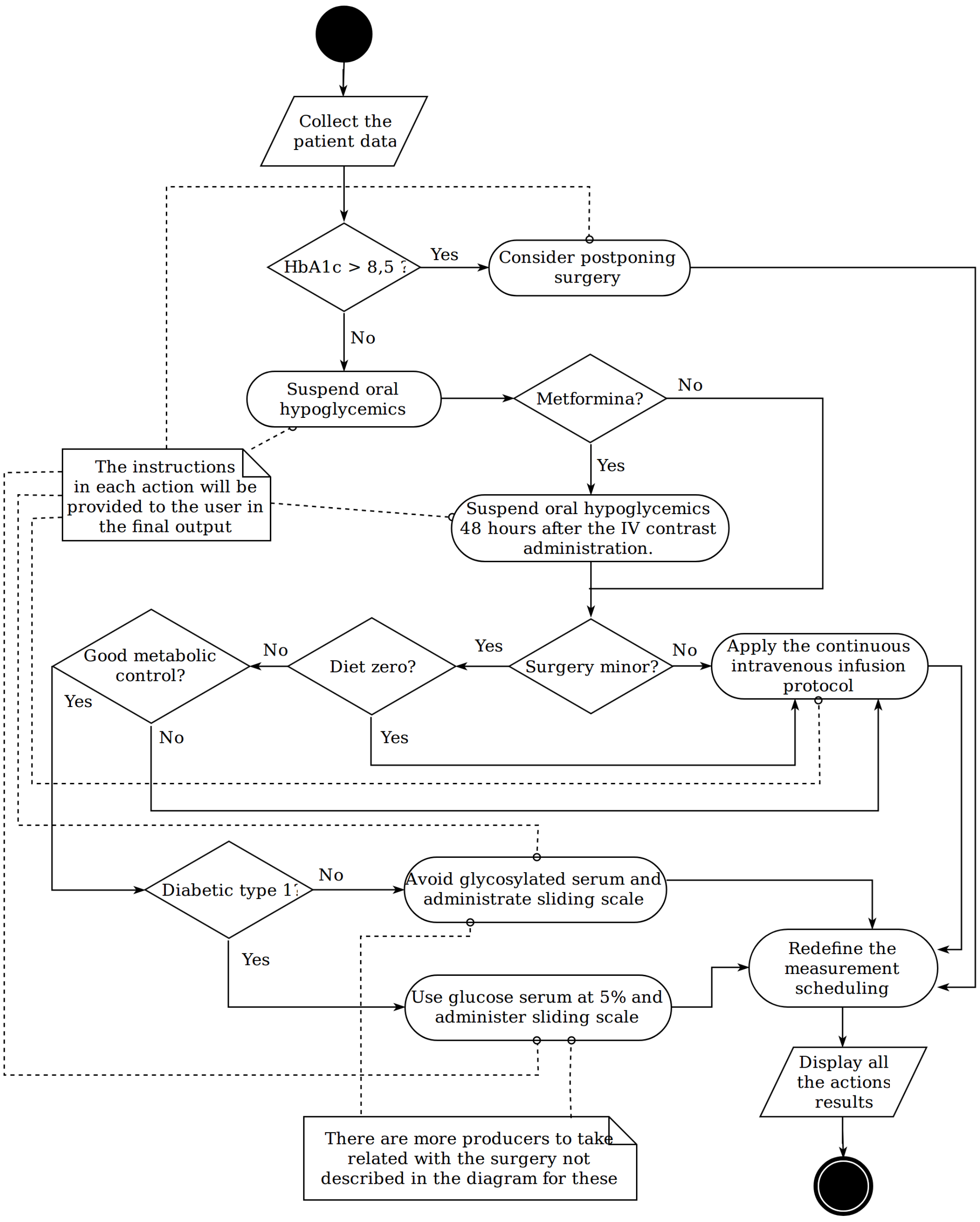

Collect the patient data
HbA1c > 8,5 ?
Yes
Consider postponing surgery
No
Suspend oral hypoglycemics
Metformina?
No
Yes
The instructions in each action will be provided to the user in the final output
Suspend oral hypoglycemics 48 hours after the IV contrast administration.
Good metabolic control?
No
Diet zero?
Yes
Surgery minor?
No
Apply the continuous intravenous infusion protocol
Yes
No
Yes
Diabetic type 1?
No
Avoid glycosylated serum and administrate sliding scale
Yes
Use glucose serum at 5% and administer sliding scale
Redefine the measurement scheduling
Display all the actions results
There are more producers to take related with the surgery not described in the diagram for these

Figure 2.2: Flowchart of the surgical diabetic inpatient protocol

2.4 Continuous intravenous infusion

This protocol is a bit different from the previous ones. From a medical perspective, it works differently. The continuous intravenous infusion is an interrupt procedure with regular adjustments that will lead to a more precise insulin dosages for the patient. This is one of the scenarios where it would be good to have a machine learning model to suggest more precise dosages [21]. However, this work was not made thinking in this kind of implementations.

This protocol is intended to be used in hyperglycemic adult patients in the intensive care unit. It should not be used in diabetic ketoacidosis or hyperosmolar hyperglycemic state, as these patients may require higher initial insulin doses. In any patient with blood glucose greater then 500 mg/dl, the initial orders should also be carefully reviewed with the endocrinologist [18]. The blood glucose target is between 140 and 180 mg/dl.

The infusion begins considering the current blood glucose value. One unit of short-acting insulin is added to the result of the following formula: Divide initial blood glucose level by 100, then round down to the nearest 0.5 unit and add this result to the initial infusion rate. However, if the enteral/parenteral nutrition abruptly stopped, the infusion rate must be reduced by 50%. In the following example, exemplifies how the previous calculation are made.

Example: Initial BG = 274 mg/dl: $274 \div 100 = 2.74$, round to 2.5:

These initial steps are made only to start the infusion rate. Afterwards the blood glucose must be monitored every hour, until 3 consecutive measurement values are in the target range. Then, the measurements can be done every 2 hours. If needed, some adjustments should be made to these measurements.

In the case blood glucose values are below the target range the dosage described in the table 2.5 should be administrated. Afterwards, the blood glucose values need to be rechecked every 15 minutes until they reach 90 mg/dl. Measure every hour until the value obtained lies in the target range, then wait for 30 minutes and restart the insulin infusion at the percentage described in the table 2.5, of the most recent rate.

Table 2.5: Procedures to take when blood glucose is below the target range.

Plasma glucose	**Dosage to administer**	**Infusion rate percentage**
<50 mg/dl	1 amp (25g) D50 IV	50%
50 - 79 mg/dl	1/2 amp (12.5g) D50 IV	50%
80 - 119 mg/dl	-	75%

Finally, when the blood glucose values are on target or above, the adjustments should follow the table 2.7. The first 4 columns represent the blood glucose ranges and each cell contains the hourly blood glucose interval change. Then, after identifying the right cell in which the patient fits, the instruction in the last column should be followed, supported by the infusion rates represented in the table 2.6.

Table 2.6: Changes in the infusion rate.

Current rate (Units/hr)	Δ = **Rate change (Units/hr)**	2Δ = **2x Rate change (Units/hr)**
<3.0	0.5	1
3.0 - 6.0	1	2
6.5 - 9.5	1.5	3
10.0 - 14.5	2	4
15 - 19.5	3	6
>20 (unusual)	4	8

This protocol only requires the current and the last blood glucose values. So, the clinical variable to support this protocol are the blood glucose and its history. All the actions follow the previous table and conditions, providing the necessary adjustments in the infusion.

2.5 Summary

This chapter details the protocols that should be implemented in the required system. All the protocols are described in detail since they were the main reason that led to this research. As a health professional, it is hard to memorize and follow all the steps described previously. Furthermore, as detailed, some protocols required periodic measurements to decide which is the best treatment.

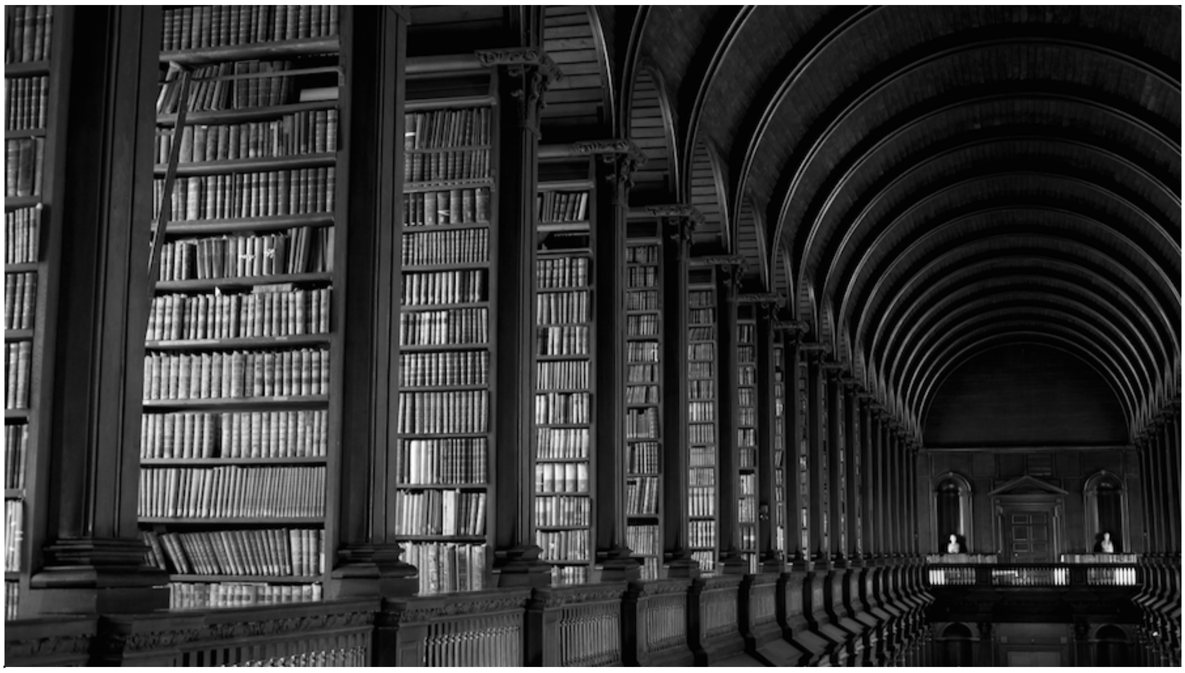
Concerning these needs, in the next chapter we will evaluate the traditional methodology. This simplified the search for a solution, because the comprehension of these protocols helped to feel the struggle that every health professional faces when treatments are updated. Therefore, now, there is a well-defined starting point to search for existing solutions, that, at least, fulfill these requirements.

Table 2.7: Rate change considering the current clinical state of the patient.

BG: 120 - 139 mg/dl	**BG: 140 - 178 mg/dl**	**BG: 180 - 220 mg/dl**	**BG: >220 mg/dl**	**Instructions**
-	-	BG ↑ by $>$ 60 mg/dl/hr	BG ↑	↑ infusion by 2Δ
-	BG ↑ by $>$ 40 mg/dl/hr	BG ↑ by 1 - 60 mg/dl/hr or BG Unchanged	BG Unchanged or BG ↓ by 1-20 mg/dl/hr	↑ infusion by Δ
BG ↑	BG ↑ by 1-40 mg/dl/hr, BG Unchanged, or BG ↓ by 1-20 mg/dl/hr	BG ↓ by 1-40 mg/dl/hr	BG ↓ by 21-60 mg/dl/hr	No infusion change
BG Unchanged or BG ↓ by 1-20 mg/dl/hr	BG ↓ by 21-40 mg/dl/hr	BG ↓ by 41-60 mg/dl/hr	BG ↓ by 61-80 mg/dl/hr	↓ infusion by Δ
BG ↓ by $>$ 20 mg/dl/hr	BG ↓ by $>$ 40 mg/dl/hr	BG ↓ by $>$ 60 mg/dl/hr	BG ↓ by $>$ 80 mg/dl/hr	hold 30 min, then ↓ infusion by 2Δ

CHAPTER 3

The Current Methodology



"Research is creating new knowledge."
— Neil Armstrong

Clinical decisions can be made without a computational system, following well-defined guidelines. However, medicine knowledge is always growing and new and better treatments are often found. These treatments usually follow some rules, that can quickly become very complex. It is challenging for a physician to follow these new protocols and it could lead to mistakes due to the numerous of treatment conditions.

Currently, there are some systems that can potentially be used to assist clinical decisions [7]. Moreover, in the past years, the healthcare information systems have been incorporated with decision support features. However, the common use of these features is to support retrospective analyses of financial and administrative data [22, 23]. The problem is that when vendors advertise systems with decision support capabilities, they are not normally designed for treatments guidance. Concerning this issue, the interest of having a Clinical Decision Support Systems (CDSS) directed to guide treatments has increased. Thus, more EHR vendors have begun to include these CDSS as a system feature, or at least provide a way to include them from external sources [24, 25].

This chapter is divided into four sections. The first section briefly describes CDSS types and how they can be useful. However, for the issue investigated in this dissertation, the nonknowledge-based approach is not the best solution, once health professionals need a different type of system. Thus, these type of solutions are not analyzed. In the next section, which concerns the knowledge-based CDSS tools, or, as some authors usually call it, the rule-based CDSS, describes the existent solutions and why these can not solve the presented problem. In the third section, some of the existent ad-hoc solutions, specifically designed for diabetes management, are analyzed. The main goal of this research is a generic solution, however, the problem could be solved with an ad-hoc solution, this research would have followed a different direction with different goals. Finally, all the CDSS solutions will be compared, in order to try to understand what can be done that was not already done.

This analysis, followed some specifications to guide the evaluation of the existent solutions. These specifications are constituted by the essential needs of the health professionals in the CHBV. However, some technical requisites must, also, be followed, keeping always in mind that these ones can be a somewhat flexible. Therefore, the functional requirements used are represented in the following list:

- User roles and activities must be controlled by an administrator, using role-based access control policies;
- The physician should be able to define protocols;
- The defined protocol most support adjustments;
- The system needs to notify health professionals about the patients monitoring schedule and treatments;
- Assigned protocols should be open for adjustments over time.

3.1 Clinical Decision Support Systems

There are two main types of CDSS [25]:

- Nonknowledge-based, which are based on artificial intelligence, and use support vector machines and neural networks;

- and Knowledge-based, which are mainly based on defined rules that most often take the form of IF-THEN rules.

The nonknowledge-based CDSS are mainly computers trying to simulate the human's thinking [26, 27]. These systems use artificial intelligence to learn from past experiences or recognize patterns in the clinical data [28].

This type of systems has the goal of inferring from a collection of data the facilitation of the decision-making processes [29, 30]. However, this implies the use of statistical approaches to model the data [31] and which implies the necessity of selecting features to classify the data in one or more groups of classes. This could be an interesting approach, but without an expert to help the system in the learning phase, this could produce wrong decisions.

The system developed in this dissertation follows a rule-based approach, since the requirements established for this phase lead to this type of methodology.

A rule-based system is a system where the specialist can configure the protocol conditions following a rule-based approach. Normally, such CDSS is divided into three parts: 1) the knowledge base; 2) the inference engine; and 3) the mechanism to communicate with the user [32].

The knowledge base consists of a compilation of information in a format legible for the system, which is often, but not always, in a if-then format.

The second part of the CDSS, the inference engine, contains all the formulations to combine the rules in the knowledge base, using the patient clinical data.

Finally, the last part is the bridge between the health professionals and the system, i.e., this part is the way of getting the patient's clinical data into the system and getting the system's outputs that will support the user's decision.

Most of the CDSS solutions are incorporated in EHR systems. This way, the data is already managed by the core system, and the CDSS only need to access that data [33]. However, the objective of this work is to create a autonomous solution to work without having to be attached to a EHR system.

3.2 CDSS Tools: General Solutions

Decision support systems can be used in different clinical scenarios, namely, as disease diagnosis, laboratory procedures, as treatment orientation, which is the objective. The following solutions are some of the most relevant for this work.

The Open Clinical Organization[1], is an organization created to promote the use of decision-making systems, clinical workflows, and knowledge management technologies. This organization presents some methodologies to represent the medical algorithms, and some standard formats for modeling and execute clinical guidelines.

[1]http://www.openclinical.org/

The Causal ASsociational NETworks (CASNET) was the first system developed for diagnosis and treatments. This system was a prototype system created for the diagnosis and therapy of glaucoma [34]. It follows a causal reasoning approach, which is a process of identifying causality, defining the relationship between a cause and its effect. Currently, there is not much information available about the system, and it is more a historical mark in the progression of this research.

GLIF3 is an object-oriented query language for clinical decision support and has been designed to support computer-based guideline executions [35]. This language defines the logical criteria, the patient's clinical data, the clinical actions and the guideline workflow. This is an update of the GLIF2 language, with new specifications and inclusion of new features [36]. The GLIF3 Guideline Execution Engine (GLEE), was developed to execute guidelines in the GLIF3 format. Its goal is the execute the guideline workflow, recording the state of each guideline step and their transitions [37, 38]. This system follows a hybrid approach, where it is not directly connected to any EHR system, but it allows the integration with external sources. It also traces the patient's records, when a guideline is applied to the patient. As a first look, it seems to be a good solution. However, no installation or way to test/use the system is available, and the lack of more details and information about it was discouraging. Furthermore, the few system screenshots available have an outdated design. This conveys the idea that the system does not allow its executions in mobile devices, which is one of the health professional requirements.

The Guideline Execution by Semantic Decomposition of Representation (GESDOR) follows the approach used by GLEE. However, in contrast to GLEE, which executes guidelines in the GLIF3 format, the GESDOR uses generalized guidelines [39]. This way, the guidelines can be encoded in different formats and then stored in the repository. The problem with this approach is that no system is available. It is only a model to execute guidelines that are encoded in different formats.

The OpenEMR[2] is an open-source EHR system, designed for medical records and patient management [40]. It is a complete system with many features that are not related to this dissertation's subject. However, this system supplies a feature designed for clinical decision rules. This module has some features, that were identified in the initial requirements, and which could almost fulfill the health professional needs. The system has a reminder feature for the physicians and for the patients and allows a fully customization of the protocols. However, despite being an excellent system, it is an EHR system, which brings a lot of complexity to install and maintain. Furthermore, the integration of two different EHR systems is not an easy task [41]. Finally, after studying the clinical decision support module, the conclusion is that it is not very user-friendly.

The OpenCDS[3] is an open-source system that provides a reference implementation of the

[2]https://www.open-emr.org/
[3]http://www.opencds.org/

Decision Support Service (DSS). This system is considered a service-oriented framework and it is compliant with the HL7 Virtual Medical Record (vMR) and HL7 Decision Support Service (DSS) standards [42]. In theory, the system almost fulfills the evaluation requirements, but after a local installation, some gaps were found. The system does not have a web client, it only provides some guidelines for the ones that want to develop its own client. The decision to use this system as an engine was discarded, due to the lack of documentation.

The Immunization Calculation Engine (ICE) is an open-source system designed for the forecast immunization, and it has two major components. The core component that evaluates the patient's history and generates a recommendation, which is designated as ICE Web Service. This component runs using the OpenCDS middleware. The second component is the Clinical Decision Support Administration Tool (CAT), which is a GUI tool [43]. This project is a composition of different systems, new in the market, and therefore the use of it was discouraging. Additionally, this system was designed for the forecast immunization and to use it in another scenario, some effort would be required. This effort is hard to estimate since the system is still in an early stage.

The IndiGO is an individualized clinical guideline system based on the Archimedes model, which takes into account more than 30 different variables of the patient's data [44, 45]. This could lead to a problem in the future, since patient's clinical data should be flexible over time. The system can also be integrated with EHR systems, and it can create individual protocols for each patient. However, this system was not consider as a solution due to a lack of documentation.

The MedCPU[4] is a platform created to guide in real time the clinicians and notify them when necessary. This platform can be integrated with EHR systems using HL7 connectivity. This system can also promote protocols for patients, individually. However, this tool follows a text processing paradigm, which is not quite the goal of this work. This system is not open-source, so the access to more documentation is restricted.

EGADSS[5] is an open source tool created for clinical guideline alerts and recalls for primary care. It supports clinical workflows and it was designed to be used as part of a service-oriented architecture. Therefore, this tool can be integrated with external systems, such as EHR systems [46]. However, this project ended in 2006, so currently, not much documentation and IT support exist.

Odyssey[6] is a commercial service that provides advice and care pathways. This system already is fulfilled with protocols, and it is hard for a health specialist in a specific disease to introduce new protocols. The fact that the system is not an open-source solution and that it does not allowed the introduction of a protocol by external specialist, led to the exclusion of this system as a solution for the dissertation's problem.

[4]http://medcpu.com/
[5]http://egadss.sourceforge.net/
[6]https://www.oneadvanced.com/solutions/odyssey/

The LogicNets[7] is another commercial tool that aims the decision-making processes. This tool is a web-based decision support platform that provides an interactive design environment for the users to easily create clinical process diagrams. It has a modern design and it can be used on different devices, desktop or mobile. However, this solution, as indicated, is a commercial solution. Furthermore, the decisions made by this system follow a form approach, i.e., the user needs to answer forms, and in the end, the system provides a decision. It is a close approach for what is intended in this work, but due to the commercial license, the solution cannot be customize programmatically for some more specific requirements.

3.3 CDSS Tools: Diabetes Solutions

As an attempt to only solve the present issue, without using or creating a generic system, some more specific solutions were searched for. A ready-to-use solution, if some exists, should be the path to follow. Afterwards, with this issue solved, more things could be done, to improve the current procedures. However, as described below, there is no solution that fits in the health professionals needs.

The Glucommander [47] is a system intended to control the blood glucose, which provides the expected measure of insulin required by the patient's organism. This amount of insulin is calculated using an algorithm, that processes some patient parameters, e.g. glucose, and informs the health professionals about the amount of insulin to be administrated, and when the next blood glucose measurement must be done. This system fails in some crucial aspects. The used algorithms in the system are not easy to change or update by the institutional experts. Additionally, the carbohydrates quantity in the estimation is not include and the system does not respond quickly to unforeseen changes in glucose.

The EndoTool[8] is another system designed to manage inpatient insulin therapies. This glucose management system uses several more algorithms to predict the insulin dosages, and it also provides insulin dose recommendations for intravenous and subcutaneous administration. This system seems to have more advantages when contrasted with the previous one, however, the possibility to easily configure new protocols to calculate the insulin doses is a crucial missing feature.

GlucoStabilizer[9] is a software application with the same aim as the previous ones. In the same way, this application uses insulin factors and blood glucose to calculate the insulin doses to be administrated. Additionally, the carbohydrate ratios are contemplated, which helps to predict an adequate insulin dosage [48].

Since these systems are ad-hoc solutions, the main focus is the health professionals needs. The technical aspects were not analyzed due to the lack of functional requirements

[7]https://medical.logicnets.com/
[8]https://monarchmedtech.com/
[9]http://glucostabilizer.net/

accomplished in each solution. Furthermore, the discussed systems are commercial solutions which allow guiding insulin therapies, being especially useful in hospitals without endocrinologists or diabetes experts.

3.4 Systems Comparison

Table 3.1 presents, a comparison considering only the solution ready to use. In the previous sections, some approaches have been described, but, since these do not lead to a final solution, they were not included in this table.

Solutions integrated with modules in several open source EHR systems, can also be found. However, despite being a good and integrated approach, these solutions can not be used autonomously with others EHR systems, discouraging its installation and use [49].

3.5 Summary

This chapter reviews the existing systems that could potentially fulfill the goal of this dissertation. Without finding a solution that completely fulfills the health professional needs, a different approach was tried. Ad-hoc solutions for diabetes were studied, since the treatments of this diseases are the reason for this work.

In conclusion, a new solution must be created and the initial requirements used in the evaluation are not sufficient to guide the development of a solution intended by health professionals. Therefore, the next step is a deep analysis of the solution's requirements and this is discussed in the next chapter.

Table 3.1: Clinical decision support systems features comparison.

	Platforms							
Feature	**GLEE**	**OpenEMR**	**OpenCDS**	**ICE/CAT**	**IndiGO**	**MedCPU**	**Odyssey**	**LogicNets**
Base Language	Java	PHP	Java	Java	Unknown	Unknown	Unknown	Unknown
License	Unknown	GNU	Apache 2	GNU	Commercial	Commercial	Commercial	Commercial
Requirements	Needs a EHR system	Needs the incorporated EHR system	Needs a front-end client	Unknown	Unknown	None	None	None
Category	Middleware	Full System	Engine	Full System	Full System	Full System	Full System	Full System
Documentation	Very Sparse	Extensive	Sparse	Sparse	Very Sparse	Very Sparse	Very Sparse	Very Sparse
Web Services	No	Yes	Yes	Yes	Unknown	Unknown	Unknown	Unknown
Web Based	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RBAC	Unknown	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
Protocol Customization	Yes	Yes	Yes	Unknown	Yes	Yes	No	Yes
Notifications	Yes	Yes	Yes	Unknown	Yes	Yes	Yes	Unknown

CHAPTER 4

System Requirement Analysis

"Faith is a necessary prerequisite for a courageous act."
— Sunday Adelaja

In the previous chapter, an evaluation of the existent solutions following a small set of requirements, was made. However, from this analyzes the conclusion was to build a new solution, the requirements used in the evaluation are not enough to proceed the development. Thus, this chapter details the final solution's requirements.

4.1 Challenges

Following the literature reviews, the project analyses, and the physician's needs, some well-defined specifications were established. Since there are already some attempts to solve the issue discussed in this dissertation, there are some starting points to help in the problem's formulation. However, there are some challenges which need to be very well specified to solve the problem in the best way possible.

Creating, customizing and allowing the future edition of the protocol, i.e., all the protocol setup and management, could be one of the hardest challenges in this project. At a first look, it seems an easy issue. However, the complications begin when a physician needs to interpret some protocols already existing in paper, and these protocols must be inserted in the system. This operation needs to be as simple as possible. Additionally, the system needs to be flexible enough to be able to deal with the patient's data and all the protocol rules. All of this gives rise to a small set of requirements.

The system needs to work in different health fields, therefore a dynamic and flexible solution must be build. This specification carries a particular issue since the protocol requires some patient's data to perform its execution. However, this data cannot be defined in the database model during the development stage. In other words, all the patient's data needs to be configurable and adjustable in the system's installation, and at any moment. That is especially important because a new protocol could require some undefined patient's variables. Thus, this definition needs to be done pos-installation, at runtime.

Finally, but not least, the system needs to hide all the complexity involved in the protocols management, and patient's data acquisition and storage. That may seem obvious, but some of the end users do not have experience using new technologies. Consequently, some of their old habits could not motivate them to endorse, with a positive outlook, this new methodology. Maybe build a solution that *influence* the users to adopt this new system and left the old paper notes, will be the biggest challenge.

4.2 Functional Requirements

In collaboration with the health professionals from Centro Hospitalar do Baixo Vouga (CHBV), some functional requirements have been established, to ensure that the final solution fits the users' needs. The following topics describe these requirements:

1. **Multi-User**

   The system only supports registered users, which registration needs to be controlled by an administrator. After a user does a registration, the administrator must approve it. Without the approval, the user cannot perform any action in the system. Furthermore, the user's profile must be defined by the administrator during the approval process, to assure that there is no leak in the privileges.

2. **Create and Manage Protocols**

   The protocol is composed of rules that will interact with the patient's data. This protocols are exclusively created and managed by the physicians. Moreover, the physician should be able to replicate them.

3. **Adjust and Customize Protocols**

   The protocol must be able to be customized for each patient, if necessary. Additionally, the protocol must support adjustments over time, when needed, even if it is already assigned to a patient.

4. **Patient Management**

   The patients must be managed by the physician. Additionally, the system needs to keep all the patient's data, interactions, and protocol history, over time. Clinical variables will store the patient's data, and its configuration is the responsibility of the administrator. In this configuration, the administrator must be able to add or hid the current variables, without losing the data history. Furthermore, after leaving the hospital, these data must be kept in the system for futures admissions.

5. **Protocol Assignment**

   The relation between patients and protocols must be a relation of many-to-many, i.e., distinct protocols can be prescribed to each patient, such as more than one patient could have the same treatment. Therefore, assigning a protocol to a patient must be possible at any moment, as long s/he is admitted to the hospital.

6. **Private Workspace**

   Nurses will have a private list of patients under their responsibility. However, patients may be not assigned to a specific nurse. The physicians should have a private space to have their own protocols. That is important for ongoing protocols, in which the physician is working on, and are not ready to be public in the system. The physician may, also, not want to share a protocol because it is a customized protocol.

7. **Notifications**

   The system needs to notify nurses about the patients' monitoring schedule and treatments, due to the existence of some protocols that require a patient check, in time intervals, which need to be strictly respected.

8. **Platform Administrators**

   The system needs to have a workspace for administrators, where the entire system can be managed without accessing it from the client side. In this area, administrators must be able to manage the patient's variables, visualize the history of all interactions on the system, and manage all the database data.

4.3 Non-Functional Requirements

Concerning the non-functional requirements, namely portability, modularity and cross-platform deployment, some mandatory aspects are detailed in this section. Below, the essential requirements for the system are described:

1. **Service-Oriented Architecture**

   The system must be easily used in multiple platforms, as a service provider. This can be simplified by following a service-oriented architecture, based on REST web services [50]. This architecture, also simplifies the integration with other tools, if in the future this need occurs.

2. **Micro-Kernel Core Architecture**

   The system core must follow a micro-kernel approach to simplify the software extension through plugins [51]. These plugins, which constitute the system core, are detailed in chapter 6.

3. **Role-Based Access Control Policies (RBAC)**

   All the user's roles and activities must be controlled by an administrator, using role-based access control policies [52]. Thus, each user will have a role which has different privileges in the system.

4. **Easy deployment**

   The system must be easy to set up, from the installation to the stage where it is ready to use. This installation process should be simplified through easy-to-deploy technologies [53].

5. **Support mobile and modern web**

   The client's application will be used at different points of the hospital, i.e., some users will use the application close to the patient beds. Concerning that, the application must fully work in smartphones and tablets. There is no need to create different client applications, one to mobile use and another one to desktops. However, the application needs to be responsive and work well in both scenarios. Regarding this requirement, and keeping in mind the creation of only one client, this one should focus on modern web practices. It must work on browsers such as Chrome and Firefox, and adopt techniques such as HTML5 and CSS3 features [54].

6. **Software Quality Assurance**

   Regardless the tests automation is essential to increase the project efficiency [55]. Regarding this, all the modules in the system should pass through software quality tests (unit, integration and feature tests).

7. **HL7 Integration**

   The system must include a communication module for data exchange with external systems, through HL7[1] [56].

4.4 Use Case Analysis

During the requirements analysis, the actors that will use the system were defined:

1. Physicians, who have the most important role in the application, because they create and manage the protocols and the patients;
2. Nurses, the entity that executes the protocols;
3. Administrators, who do not work with the system daily, but need to keep it operational.

These entities have different roles in the system, with different privileges. Therefore, it was mandatory to define the responsibilities to each one. In Figure 4.1, which is a typical use-case diagram, these entities and their actions on the system are represented. It is possible to identify three main action areas in the system: 1) Protocol Management; 2) Patient Management; and 3) System Administration.

The actions associated to these areas are the following:

1. **Manage Protocol**

   Physicians should manage all the protocols. However, it is necessary to allow these permissions to the administrator, just in particular situations. For instance, when the protocol has some complex calculations, the physician could ask the administrator for help in the implementation stage.

2. **Search Protocols**

   Searching for protocols also should be done by physicians. This activity is important in situations where the system has a high number of protocols, or to check if other physicians have introduced the desired protocol. Nurses should not be allowed to search for protocols, to avoid misinterpretations.

[1]http://www.hl7.org/

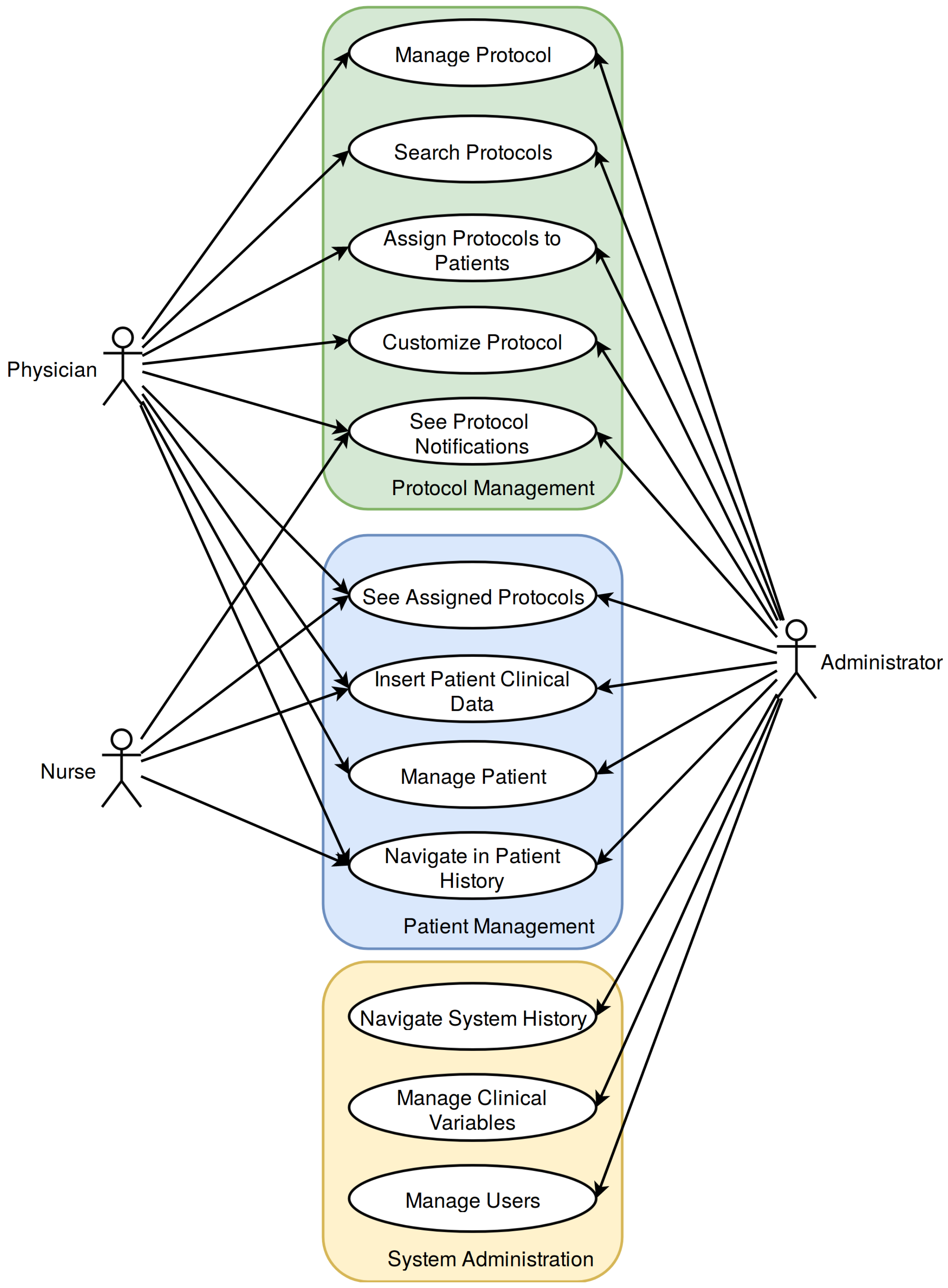


Figure 4.1: Use case diagram

3. **Assign Protocols to Patients**

   When the patient is admitted to the hospital, or even later, the protocol assignment should only be made by the physician. Nurses do not have permissions to take this kind of decisions.

4. **Customize Protocol**

   The protocol customization is another activity which needs to be highlighted. Despite being associated with the protocol management, this action is very singular, because changes are only for the protocol assigned to one patient, i.e., for the system, it is a new protocol. Additionally, only specialists in the disease have the knowledge necessary to perform customizations. This action belong to the physician responsibility.

5. **See Protocol Notifications**

   This requirement is very useful namely for patients follow-up (nurses). The notification system increases the measurements accuracy. Although it is not a task for physicians, they have access to the notifications.

6. **See Assigned Protocols**

   The assigned protocols should be available to all entities. Nurses need to know what to do and how to perform the treatment. Physicians need to know what has been done by the nurses. This track helps them to understand if a protocol change is necessary.

7. **Patient Clinical Data**

   Patients' information needs to be added by both entities as well. Some of this information is given by the physician when s/he checks the patient state. However, nurses also need to introduce the measurements for the protocols' calculations.

8. **Manage Patient**

   The patient management should only be done by the physician, because nurses are not allowed to discharge the patients, neither to admit them in the hospital.

9. **Patient History**

   Allowing nurses to navigate through the patient history could be a tricky decision. A patient's history may contain sensitive data, but it could help a nurse to know if s/he already performed the treatment or if s/he tipped the right data in the system. Regarding this, a patient's history access by nurses should be controlled. However, physicians will not need these restrictions.

10. **System History**

    This activity is more useful for system administration, so, only the administrators should be able to navigate through the system's history. That will allow to see if the system is stable and in correct use.

11. **Manage Clinical Variables**

    After the system's installation, the patient's clinical variables must be defined. Since it is back office activity, and to assure the system's stability, this activity should only be performed by administrators. However, if a physician needs more clinical variables, s/he should request it to the system's administrators.

12. **Manage Users**

    All users in the system, physicians and nurses, should be managed by a registered administrator. Nurses and physicians should not be able to manage other users in any situation.

All the above requirements can also be executed by administrators, just to provide a simple way s/he can test the system's stability when necessary. However, this should be made carefully, because this role implies a lot of responsibilities.

4.5 Summary

This chapter discussed the specifications that supports the system's development. Some of these requirements are functional, which describes what the system must be able to do. Additionally, it analyzed some non-functional requirements, which specifies how the system should work technically.

CHAPTER 5

Solution Architecture

"Whatever good things we build, end up building us."

— Jim Rohn

After discussing and describing the main requirements, the solution's architecture will be analyzed. This architecture needs to fulfill the technical requirements described before, as well as provide a good foundation for the current functional requirements,. allowing the system's scalability and adaptability to new features.

This chapter describes the system's architecture, which follows a client-server model.

5.1 Client-Server model

The proposed architecture follows a Client-Server model, in which each side is sub-divided in several layers (Figure 5.1).

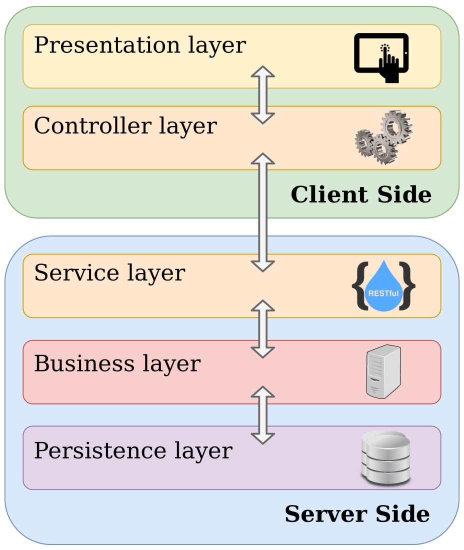


Figure 5.1: System architecture, which follows a client-server model

The Client side encapsulate mostly of the presentation part of the system. It is divided into 2 layers, the presentation, and the controller layer. The presentation layer is responsible for the user interfaces. The controller layer consumes the backend webservices and provides the data to the presentation layer.

The Server, which is mainly the backend core, is subdivided into 3 sub-layers: 1) business; 2) persistence; and 3) service provider. The persistence layer is responsible for storing and maintaining the system's data. The business layer contains most of the application's logic. Finally, the service layer provides a RESTful API with services prepared to interact with all the system's functions with or without the client. This layer will be used by the client to access all the core features.

Figure 5.2 presents the system's architecture, where this decoupled approach is emphasized, namely the core modules and the extensions elements, without having to interact with the rest of the system. All these elements will be detailed in the following sections.

5.2 Server Side Modules

One of the goals for our application was that it should be modular and extensible. To address this, the core was reduced to the minimum, allowing its extensibility through the introduction of new plugins.

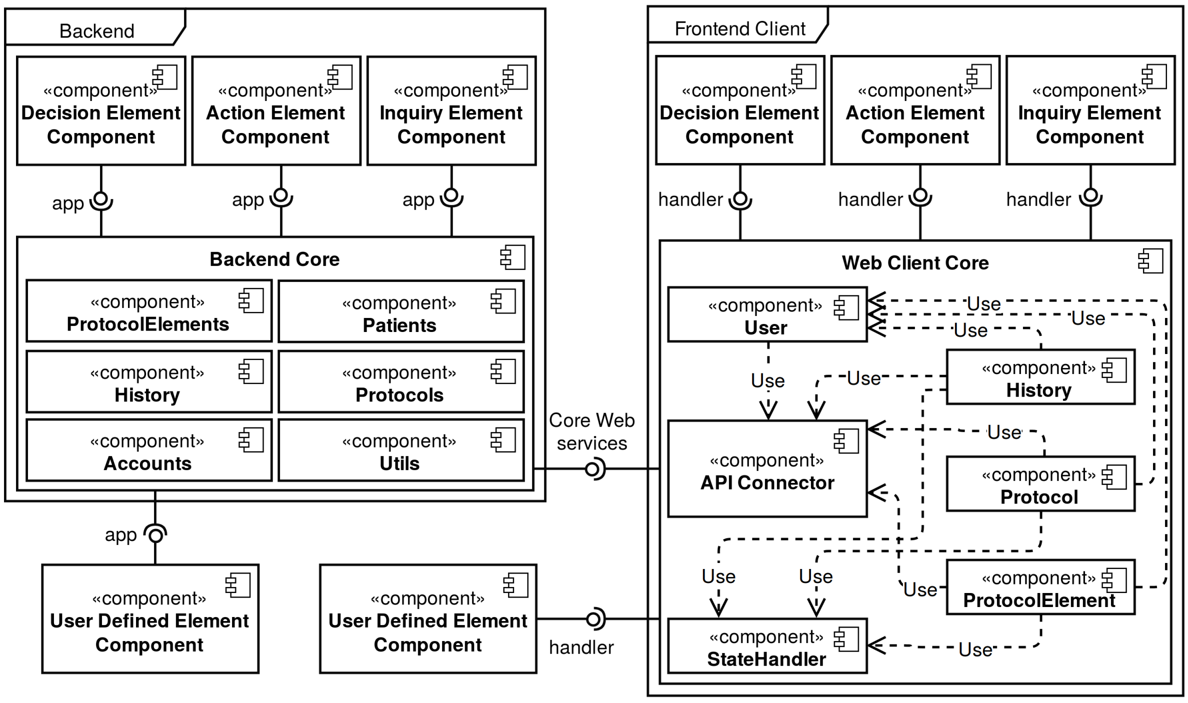

Backend
«component» Decision Element Component
«component» Action Element Component
«component» Inquiry Element Component
app
app
app
Backend Core
«component» ProtocolElements
«component» Patients
«component» History
«component» Protocols
«component» Accounts
«component» Utils
app
«component» User Defined Element Component
«component» User Defined Element Component
Core Web services
handler
Frontend Client
«component» Decision Element Component
«component» Action Element Component
«component» Inquiry Element Component
handler
handler
handler
Web Client Core
«component» User
«component» History
«component» API Connector
«component» Protocol
«component» ProtocolElement
«component» StateHandler
Use

Figure 5.2: Architecture overview (including the Server Side on the left, and the Client Side on the right).

Figure 5.3 represents the backend architecture with all the constituent components. Except for the FlapPages, which is an application to store HTML pages in the database, the remained components will be detailed below.

History

One of the requirements was to keep all the patient's log. However, it was decided that recording all actions performed in the system, would be better, which implies keeping track of who performs which protocols in which patients, i.e., record the history for everything.

For this, we took advantage of previous work in a similar project [57, 58].

The objective of this module is, mainly, recording each action performed on the system in a database table. For each entry, there are some entities associated: 1) an Actor, which is the user that performs the action; 2) an Event, which defines the operations performed by the user in the system; 3) an Object, that can be every object in the system and is the object associated with the actions; and 4) the Viewers, which is a group of users, not directly related with the action, but which should be associated and, in some cases, informed about this.

The Events are actions that can include the Create, Read, Update and Delete (CRUD) operations, as well as some other more specific operations. These operations are always associated with a system's object, for instance, the insertion of a new patient, or a protocol execution.

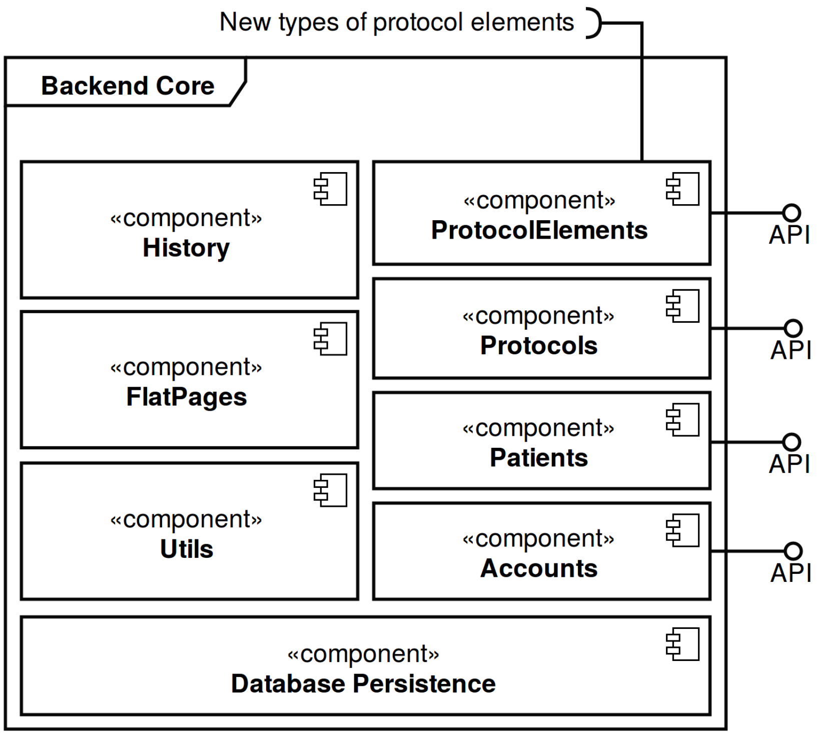


Figure 5.3: Backend architecture.

The Viewers are users that do not perform the action, buy they are attached to it for consistency system log purposes. This methodology is resumed in Figure 5.4.

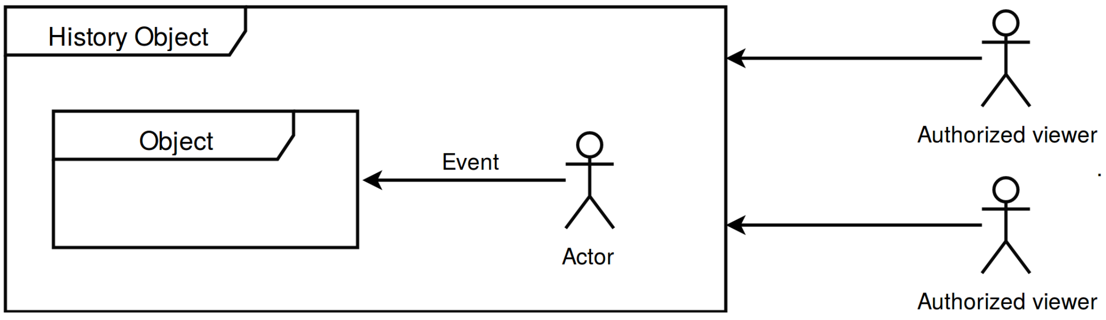


Figure 5.4: History model logic.

Accounts

Another requirement is the creation of a multi-user platform, where only registered and approved user can access the system. Additionally, different users could have different privileges, following a role-based access control policies, controlled by an administrator.

This module is responsible for managing those users, in this case the health professionals and the IT administrators. This module also manages the recovery password requests, the user's roles and all action that allows the users to customize their profiles.

Patients are not considered users of the system and they have been left outside this module

Patients

The patient management is a key element of this system, since almost every action is related to patients or, with protocols that are assigned to patients.

Briefly, this module manages the patient's clinical data. Since these are dynamic, the administrator needs to add or hide patient's clinical variables at runtime, i.e, without changing the source code or the database design.

The adopted approach is the creation of a database structure that is able to deal with these changes. (Figure 5.5). The table CVPatient stores the patient and clinical variable primary keys. The clinical variable column is the variable identifier, not the variable content. The content is stored in the value column. Briefly, this table stores all the measurements and other data from all patients registered in the system.

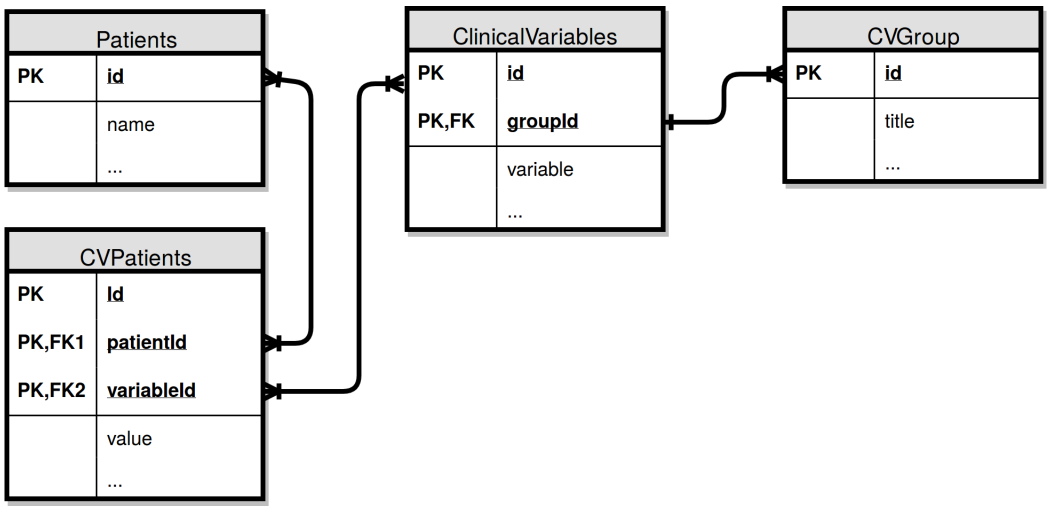


Figure 5.5: Database structure for storing dynamically the patient's clinical variables.

To set up a new clinical variable, a new entry is added in table ClinicalVariable. Finally, the CVGroup table was created to allow grouping variables in the user interfaces, in reporting or in other data management operations.

Protocols

Protocols are the other biggest element of this system. In fact, the system was designed to manage clinical protocols, independently of the medical context.

There are two types of protocols: 1) Simple type, that is based on data insertion, providing a response in the end; and 2) Complex type, which is based on a workflow of tasks. Both protocols are similar, but the first is mainly a group of rules to interact with some patient's clinical variables, providing an output after the protocol's execution. The user only inserts the clinical variables, needed by the protocol. All the condition's complexity and the protocol flow is hidden to the user.

The Complex type is more a step by step execution protocol, where the user interacts with the protocol following a pipeline. For each step, the output defines which is the next step, and each action needed by the user. This step could be data insertion, analyzing data, or chose the protocol behavior considering the defined conditions, or it could only be instructions to help the user.

Regardless of the protocol type, also both of them need a schedule to be executed, and patients to be assigned. This module is responsible for managing these actions. Additionally, the protocol customization is another responsibility of this module.

Protocol Elements

Protocol elements are the basic units of the protocols. Each protocol is at least composed by one element, and this composition follows a well-defined structure that must be respected in the execution. Therefore, each element has an output that specifies which is the next protocol element to be executed.

This module is responsible to manage all protocol elements, by providing an API to easily integrate new protocol elements. By default, the system includes three types of elements: 1) Decision elements, in which the next action depends on some condition; 2) Inquiry elements, where the user needs to supply some information to the system; and 3) Action elements, that only provide information about what users need to do. However, more elements can be created, including hardware integration, for instance, to simplify the recorded patient's clinical data using medical devices.

Utils

This module is mostly responsible for the flat pages management, the multilingual of the system, and some system variables that could be defined in the installation, or post-installation.

5.3 Client Side Modules

The Web Client follows the same micro-kernel pattern as the server. This layer interacts with the backend web services, displaying the information through an easy-to-use interface, aiming the simplification of the protocol management.

Figure 5.6 resumes the web client core architecture with all the components and the way that they communicate and use each other. Almost all the components finish in an API connector as the information source. Additionally, they end up connecting to the State

Handler for keeping their status update in all the application. Furthermore, all the components are connected to the User component due to the authentication control. In the following subsections, these components are detailed.

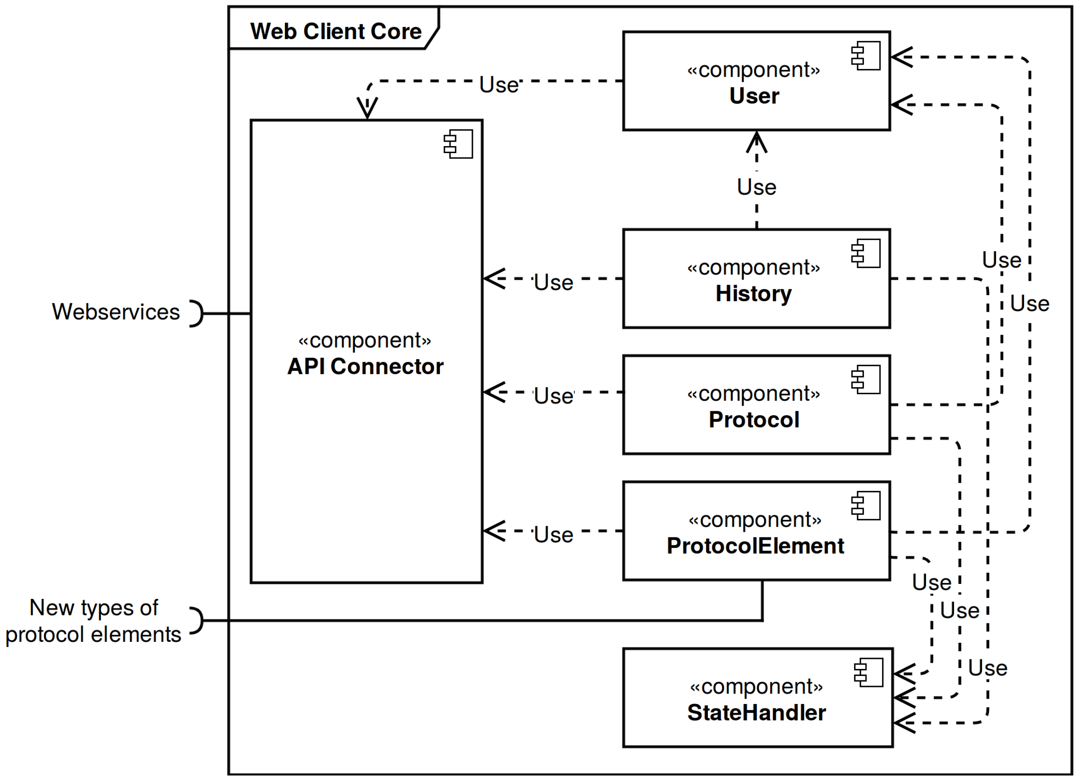


Figure 5.6: Frontend architecture.

Patient

The Patient module provides a visual interface for patient management. This interface represents all the patient's clinical data, but also all the treatments prescribed and a history log with all the interactions between the patient and the health professionals. Since this module manages all the patient's information, it is also responsible for admitting and dismissing patients in the system.

The data used by these interfaces, as well as the related actions are provided by the server side. To access these data and methods, the module ends up using the API Connector. The logic layer of this module is only for presentation proposes, such as the components rendering and some visual actions.

Protocol and Protocol Elements

These modules are mostly responsible for the protocol management and protocol assignment.

As it was explained in the subsection 5.2, there are two different types of protocols. The main difference between these two types is the execution, where one of these is visually more complex.

The execution of the workflow-based protocol type, which is mainly a step-by-step execution, requires more logic in the control layer. For each step, a new element must be presented. Since there are different elements, for each of them, different visual components also exists.

The Protocol Elements is the module responsible for rendering the content of these different elements. Additionally, it also provides an API to add external elements.

User Account Manager

The module Account Manager allows the managing of user profiles. This allows a user to login, configure his profile or request for password recoveries. However, This module was not designed for the administrators who manage the users directly in the backend.

The users components is connected to all the other modules, to keep the track by who performed what action in the system. This way, the system can keep a consistent history log.

API Connector

The API Connector is the source of data for the client side. This module has the responsibility to do the bridge between the interface components to the server side.

Since almost all the webservices in the backend needs an authentication, this module has a strong relationship with the User component. Without login, only the new user's registration is allowed.

5.4 Summary

This chapter details the system architecture and its main components, without considering the technologies used for the implementation.

The next chapter will focus on the system's implementation, describing the technologies used to develop each component.

CHAPTER 6

The System Implementation

"Everything is theoretically impossible, until it is done."
— Robert A. Heinlein

The problem was studied, different solutions were analyzed, following the definition of a set of requirements, which led to a system planning, i. e., the software architecture. All these topics have been described in the previous chapters. This chapter will discuss the implementation details.

Following the architecture model defined in the section 5.1, several technologies were selected to fulfill the architectural requirements. Figure 6.1 represents an overview of the main technologies used in the system.

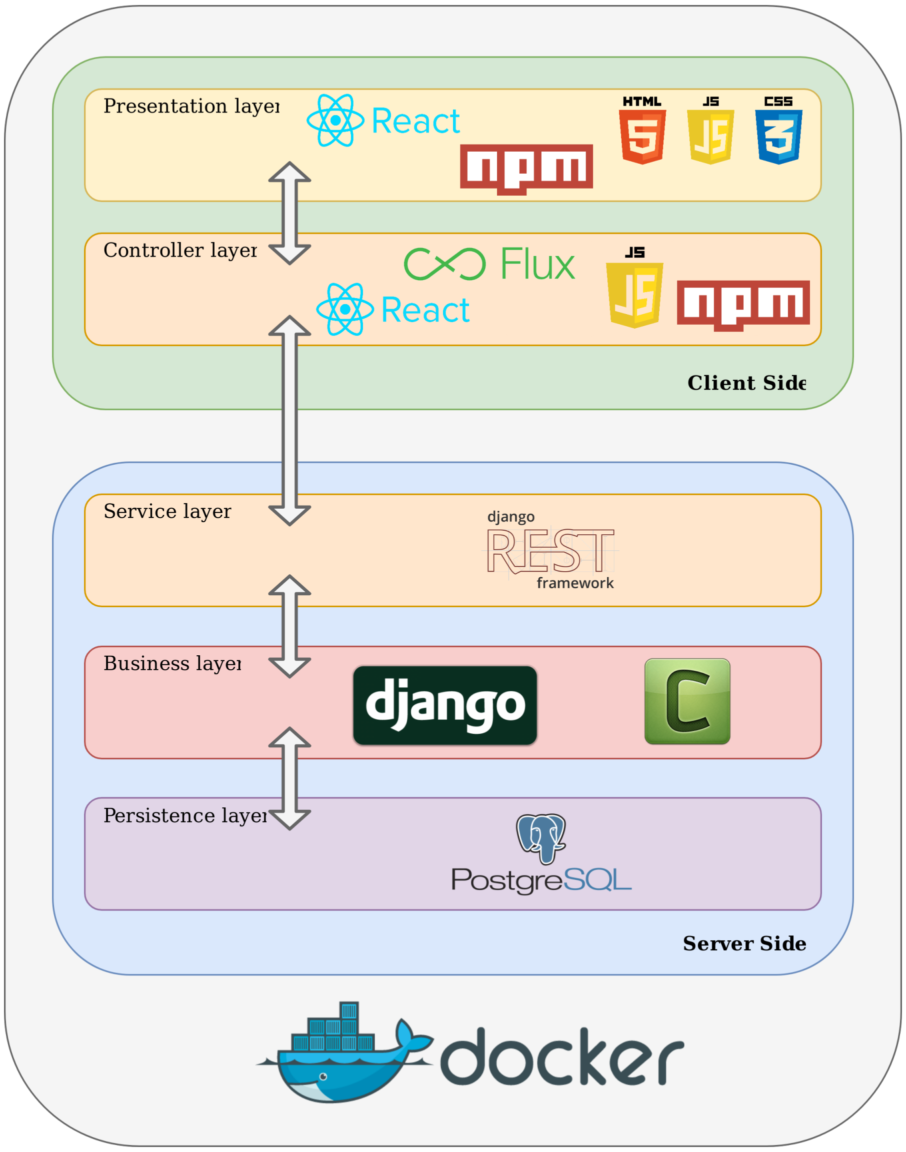


Figure 6.1: System technologies.

6.1 Server Side Implementation

The server side, which is the application core, was developed following the architecture described in section 5.2. This section is divided in three parts:

- Technologies used, where the main technologies used to develop the system backend are described;
- Persistence, a brief description of the database structure;
- Web Services, are the indispensable piece of the system to provide an API for the client;

- Tests, since the system was developed following a testing methodology, it is important to describe this implementation phase.

6.1.1 Technologies used

The technologies used for the backend core are mainly python based. This was influenced by the fact that the core was developed using the Django framework, which is detailed below. For the database management, PostgreSQL was used, for several reasons that are also explained below.

Django

Django[1] is an easy-to-use programming framework to build web or mobile applications [59]. This framework is known by its simplicity, allowing the creation of pieces without unnecessary complexity [60].

This framework is the basis of the whole system, creating a python based CDSS. The main reason for choosing this framework is the excellent documentation and the support that it provides. It also allows the creation of a stable system with a good core. In other projects [57, 58, 61] currently in a production environment, which use this framework as backend core, the resulting systems are stable and with a robust core.

Another positive factor of using Django is the ability to have isolated and decoupled components, which greatly simplifies the integration of a micro-kernel architecture. This is a requirement for this system, making it modular and expandable. Additionally, the framework uses a package manager, the Pip Installs Packages (PIP) to install new packages. This package manager is very useful to replicate the development environment, keeping track of which packages are installed.

Finally, Django has a big community of contributors. This community maintains and develops new third-party applications, which can be integrated into this system. One example is the configuration of the Django console layout. Using the Django JET[2] application, which is a modern template for Django admin interface with improved functionalities, the database management can be simplified. Another example is the Django Constance[3], which allows easy configuration of dynamic settings in Django applications.

Django REST Framework

Django REST Framework[4] is the recommended solution for building RESTful API over Django. The framework has a detailed and well-written documentation, with a great

[1]https://www.djangoproject.com/
[2]http://jet.geex-arts.com/
[3]https://django-constance.readthedocs.io/en/latest/
[4]http://www.django-rest-framework.org/

community support, similar to Django. It is also supported by authentication policies including the packages OAuth1a and OAuth2, and serialization mechanisms [62].

The Service Oriented Architecture (SOA) defined for this system implies the use of some technology for communications between the server and client sides. Django REST Framework takes an important role in this point. It is used in all the system modules, providing a communication API to the outside.

Celery

Celery[5] is an asynchronous task queue system, with support for real-time or scheduled executions. It supports integration with Django applications, being useful for handling background tasks in this system, allowing, for instance, the action scheduling, or the execution of long-running events, such as email sending.

This framework schedules the protocol executions. It is a backend task running for checking if there are protocols to be executed. Currently, it only notifies the user of the main dashboard. However, in the future, if users want to receive an email notification, this is the module responsible for it.

PostgreSQL

Django includes Object-Relational Mapping (ORM) to automatically handling the database integration. The used database was PostgreSQL[6], a popular open-source Relational DataBase Management System (RDBMS). It uses in the database markups the free SQL, which is currently a standard language for relational database management. Additionally, Django works very well with this database driver.

There are some expectations of storing large amounts of data in some tables, such as the patient's clinical variables records. However, the authors [63] used this framework to deal with simple datasets and then used a NoSQL framework that makes it easier to work with larger volumes and more heterogeneous data sources. The use of a NoSQL framework was considered for handling the patient's clinical data, using HBase[7], MongoDB[8] or Cassandra[9], which are NoSQL database engines to deal with big data. However, in this first stage of the system, only the relational model was kept.

6.1.2 Web Services

Concerning the web service implementation, which are JSON based, and detailing a bit deeper into the technical aspects, each system module has several data models and several

[5]http://www.celeryproject.org/
[6]https://www.postgresql.org/
[7]https://hbase.apache.org/
[8]https://www.mongodb.com/
[9]http://cassandra.apache.org/

views. For each view, there is a serializer, which allows complex data such as query sets and model instances to be converted to native Python data types [64]. Additionally, each view has filters defined, used for data sorting, filtering, and which model fields can be filtered.

For each view, an URL is defined, allowing the mapping of each method. With the goal of homogenizing the web services definition, an URL structure was defined. This will help developers to understand from which module, model, and view, the method has been called. This way, all web service starts with the path *'/api*, followed by the module's name and view. Then, the next element in the path can be the method to call or the id, depending on the web service's definition.

There are different views, with different purposes. Thus, some of those do not have all the RESTful standard methods defined. Appendix A presents the list of all web services.

Regarding the listing web services, which are the ones that accept parameters on the requests, they follow the GET parameter '?page=<x>' and '?page_size=<x>', to retrieve the defined page and the number of records of a query set. Additionally, for each one, the possibility exists to filter or sort the results, using the GET parameters '?<field>=<value>' and '?ordering=<field>', respectively.

6.1.3 Tests

Testing is not only about making sure that the system works, but also to ensuring the quality of the system, preventing that the application results in unexpected behaviors.

During the development phase, every module built in the backend was tested. To do this, the testing features provided by Django was used. This feature requires a stack of testing instruction, indicating the desired behavior of the implemented methods. Therefore, a stack of tests for each data model, view or serializer was created and every defined method was tested.

These tests were helpful during the development stage, once the system suffer some changes during this process. Furthermore, these tests will be an advantage and save time in the future when new features are created.

To tests the backend, more precisely the actions related to the data models, it was necessary insert testing data in the database. Also, this data must be the same in every test. Thus, to do this, Django allows the creation of temporary databases, that are created and deleted every time the tests are running.

6.2 Client Side Implementation

Similar to the server side, the client was also built keeping the modularity and extensibility in mind. A good way to achieve these goals is using the ReactJS and RefluxJS combo, with their unidirectional data flow, leading to a well-known pattern composed by actions and data stores. The actions initiate new data to pass through stores before coming back to the view components. When a view component has an event that requires some changes in the data stores, it sends a signal to the data store through the actions available. Figure 6.2 represents this three-tier structure and flows.

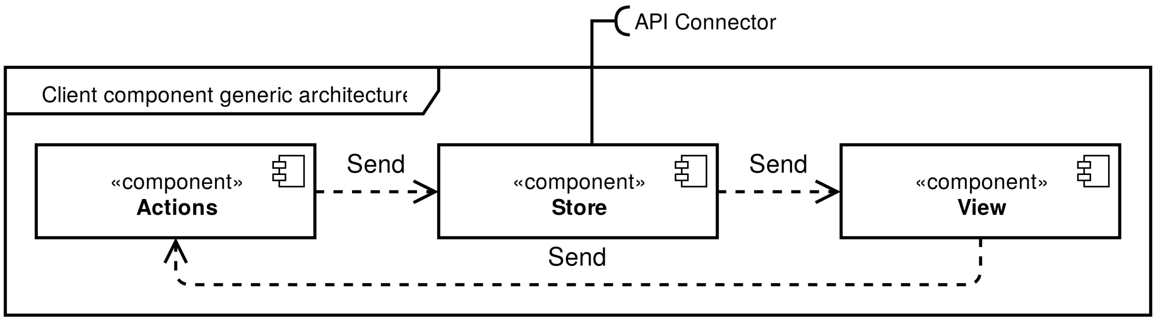


Figure 6.2: Client component architecture.

6.2.1 Technologies used

As already indicated, ReactJS was used as a frontend developing framework. However, by itself, this technology is not enough to achieve the end result. This subsection describes the most important technologies used in the client development.

ReactJS

ReactJS[10] is a JavaScript library to build web components. It is used for handling a view layer for web and mobile applications. Currently, it is one of the most popular technologies used for this purpose.

This framework allows developers to create reusable components, many of them are public and open-source. Beside that, this framework has a huge community of developers and third-party components.

Another positive aspect of this framework is the possibility to create large web applications, that can change data without reloading all the page, only the component. Consequently, this increases the application speed, providing fast, scalable and simple solutions.

RefluxJS

RefluxJS[11] could be consider a pattern instead of a technology. As briefly explained in the begin of this section, this follows a unidirectional dataflow, which is represented in the Figure 6.2.

[10]https://reactjs.org/

[11]https://github.com/reflux/refluxjs

In this system, for each module, an action and a data store components was created. These actions are called by the React components, which then are executed in the data stores. Additionally, the data stores are developed following the singleton pattern, having an internal state to keep the application's data.

Bootstrap

Bootstrap[12] is the most popular open-source toolkit for development with HTML, CSS, and JS. Responsive and mobile-first client applications can be build with it. Furthermore, it has a great community of developers and offers a huge variety of interface components useful in frontend development, such as buttons, navigation bars, modals, panels and much more.

Font Awesome

To increase the visual appearance of the components, some icons were used. It helps the user to recognize behavior patterns, because similar objects, should indicate that the buttons will perform similar actions. For instance, the icon of a saving button is, almost always, a floppy drive, which is easily perceptible that the button is used to save something.

Currently, one of the most popular free open-source toolkits for web icons is Font Awesome[13]. This offers scalable font-based vector icons, that do not require JavaScript or images.

NPM e NodeJS

Similar to the PIP used in the backend as a package manager, the frontend applications also have package managers. Currently, some options exists and this project used the Node Package Manager (NPM)[14]. This is a well-known package manager for JavaScript, which came from NodeJS[15], which is a JavaScript server application. However, it is independent and can be used without needing to write Node applications.

Most well-known JavaScript applications support NPM package manager. Besides this, the ReactJS community uses this package manager to share their components.

Selenium

Beside testing the backend features, it is also important to test the frontend behavior. Create or change a feature could change the system's stability in several points that are not used with frequency, turning the discovery of a bug very difficult. ReactJS has, by itself, no

[12]https://getbootstrap.com/
[13]https://fontawesome.com/
[14]https://www.npmjs.com/
[15]https://nodejs.org/en/

way to create automatized tests and therefore, it was necessary another framework to perform this.

Selenium[16] is an open-source tool that automates browser actions. Basically, it works defining a list of steps actions, and the expected result. If some of the actions do not have the expected behavior the tests fail.

In this project, the desired system behavior is recorded and then, each time a new feature is stable, the stack of tests is executed, checking for inconsistencies.

6.2.2 Web Services Client

The client side has two layers, the presenting layer, and the controller layer. The previous section, already described the principal technologies used in the presenting layer. One of the responsibilities of the controller layers is to consume web services.

For the web service consuming the API Connector was created, making the bridge between the server side and the data stores. To consume the web services in the data stores, it was used the Axios[17], which is a HTTP client for the browser.

The communication between both sides is mainly the following:

1. The web component needs some data stored in the server;
2. It calls for an action defined for the module that is responsible for providing the desired data;
3. The actions trigger the data store.
4. The data store request to the API Connector to consume the web service, providing the necessary information about it;
5. The API Connector communicates with the server side and receives the response;
6. The data store receives the data and updates its state;
7. Finally, with the state change, the component that requested for the data, is also updated.

6.3 Deployment

The solution's deployment is not the biggest concern for development teams, but for a system administrator, it could be tricky. Optimizing this procedure increases the service quality, preventing the system from being down for long periods of time. However, the maintenance periods are not the only concern.

[16]https://www.seleniumhq.org/
[17]https://github.com/axios/axios

The users do not want to know if the installation is complex or if it takes huge efforts to be made. They only want a stable and fast system that fulfills their needs. Therefore, a way to simplify the installation without compromising the quality of the service, must be found.

In the recent years, with the advent of cloud computing, there is a tendency to move the applications towards virtualization, using virtual machines instead of physical ones. A virtual machine can support a complete system, making a flexible use of hardware and isolating the software from the hosting system [65]. Therefore, there were different approaches for different virtualization technologies.

Docker[18] is a container virtualization technology, which is considered a very lightweight virtual machine [66]. One of the main advantages is the computational resource saving, because this technology reuses the same OS for different containers [67]. The use of a container was considered a good tool for deploying microservice-based architectures, which is mainly the architecture that this system follows [66].

Figure 6.3 presents a comparison between containers and virtual machines. As can be noticed in this figure, three different application using machine virtualization will need three different operating system installations (Guest OS). In contrast, using containers, only one operating system with a container engine is necessary. This container is the layer between the OS and the applications.

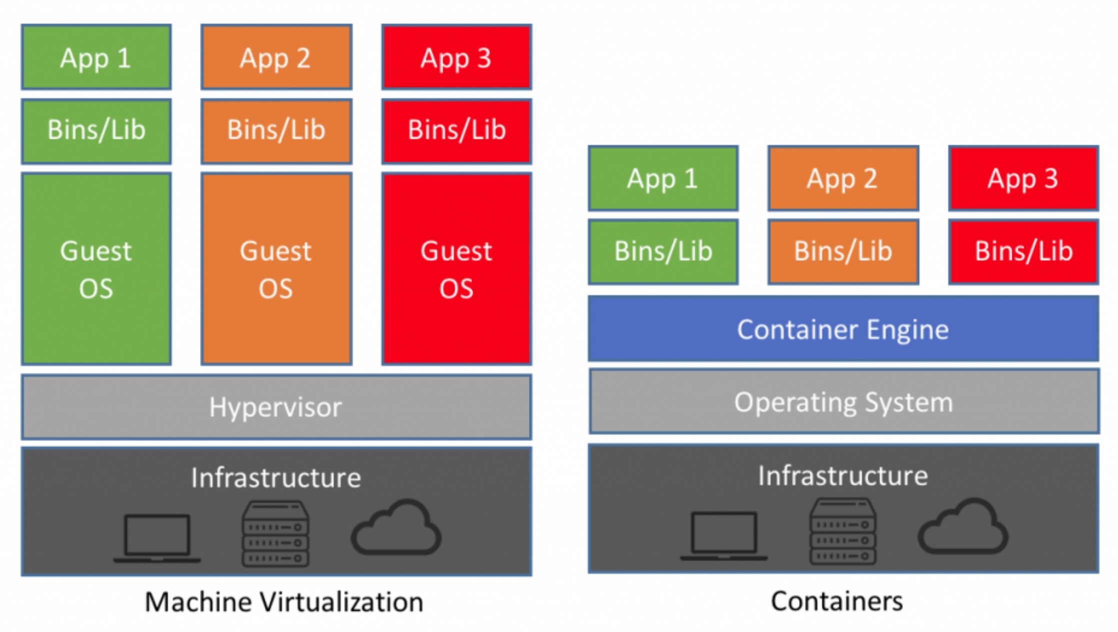


Figure 6.3: Comparison between containers and virtual machines.

[18]https://www.docker.com/

Docker Images

This system deployment uses two docker images. For the database management a PostgreSQL image is used, retrieved by the official repository [68]. This way, the database installation and management are easily processed, being only necessary to establish the connection with the containers which will access the database.

The other image is the system image. This one was created to set up the system with all the necessary dependencies. These steps have been defined in a Dockerfile, which is a guide for the image building [69]. Then, using that image, new containers with this solution can be instantiated.

This container has three main components: 1) the Nginx[19], an efficient web server; 2) the uWSGI[20], an application container for Python code; and 3) the Celery, a component to deal with synchronous tasks.

Django has an internal development server, but, by itself, it has no way to be used on a production scale. Regarding this, Nginx was used, which is robust and provides efficiently a server for web content. This server is designed to serve the page's content, not to behave like an application server [70].

The uWSGI is an application container for Python code, that will generate the static content from the Django code [71]. Additionally, this application is compatible with the Ngnix. So, these two frameworks are used as the mechanism that gets the project into the web.

The Celery is used to deal with a different issue, as it was explained before. This framework is separate from the uWSGI and the communication between them is made through a TCP channel [72].

Docker Compose

After describing the system deployment composition, it is perceptible that these containers need to communicate and exchange data between them. Usually, there is a right order to run these containers and some initial configurations for each one. However, Docker by itself does not allow the orchestration of the containers. There are however some different approaches to support this issue, being one of these the use of a Docker Compose[21].

With the Docker Compose it is possible to define a set of containers to boot up, and their runtime properties. The container that interacts with others in some way follows the specifications in the runtime properties [67].

Considering the advantages of using this orchestrator, it was decided to use it to simplify the system deployment, since this project has multiple containers.

[19]https://www.nginx.com/
[20]https://uwsgi-docs.readthedocs.io/
[21]https://docs.docker.com/compose/

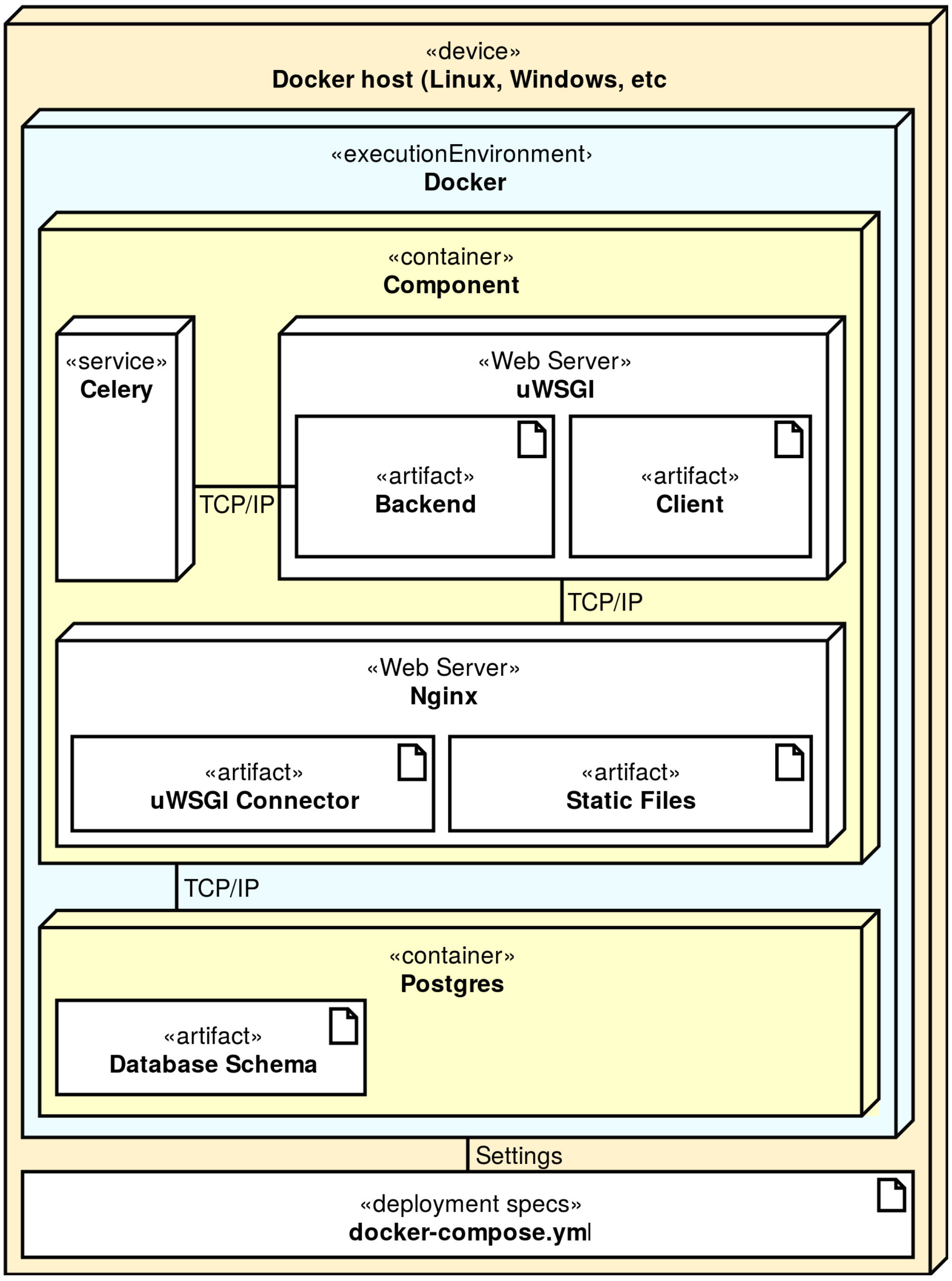


Figure 6.4: System deployment logic.

Overview

After analyzing and describing the approach used for the solution deployment, Figure 6.4 presents all the previously explained components and the way that they interconnect with each other.

The settings used for the Docker Compose are defined in the *docker-compose.yml* file. The system settings, such as the system environment variables, or the access ports are defined only in this file, preventing inconsistency errors [73].

To easily start or stop all the system using the proper commands, the Makefile was used, which adapts the Docker Compose syntax into an easier format. The Makefile is a file that contains some directives used to automatically execute a set of instructions to accomplish a specific goal [74]. Therefore, to start the project, only two commands are necessary in the command line: 1) **make** for creating the system docker image; and 2) **make run** to initialize and run the system containers.

6.4 Interface Components

Different workspaces have been created to cope with the distinct users' roles. In the profile nurse/caregiver, for instance, the first view is the list of patients under supervision. When selecting a patient entry, one can follow the protocol assigned to that patient and what measurements or actions need to be performed in each instance. After executing a protocol, the system presents the prescription that better fits the patient, under the collected conditions. This automatic procedure avoids performing manual calculations or consulting hospital guidelines to identify the right dosages. The management of insulin dosage is provided by the system and this makes the whole process much more simpler.

This section shows and briefly describes the main interface components of this solution. The first page of the system is shown in the Figure 6.5. It is an entry page without much information, allowing the user to sign up or login because the system is protected through a login-based mechanism.

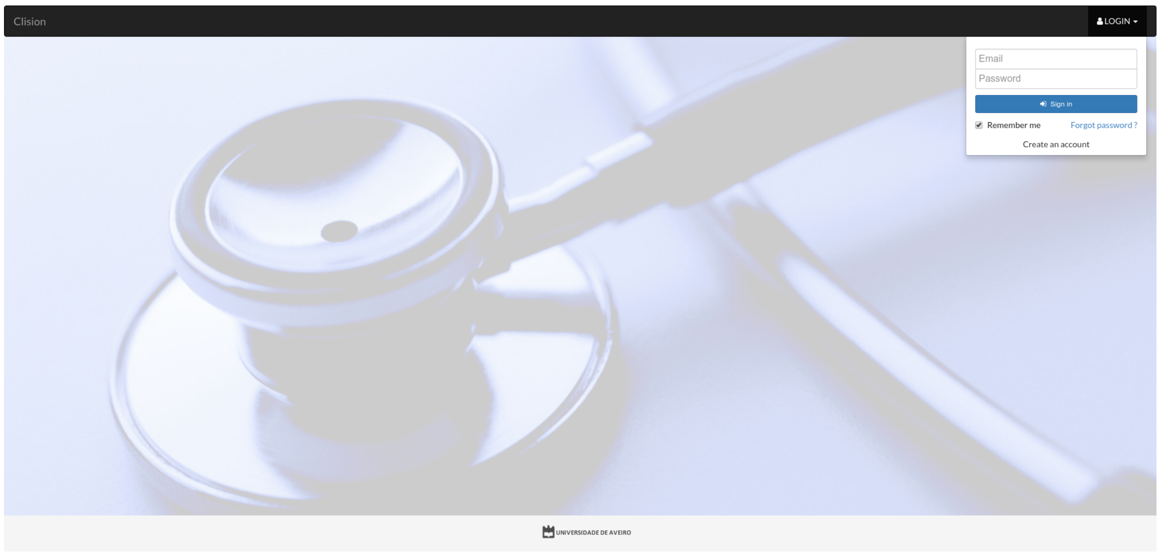
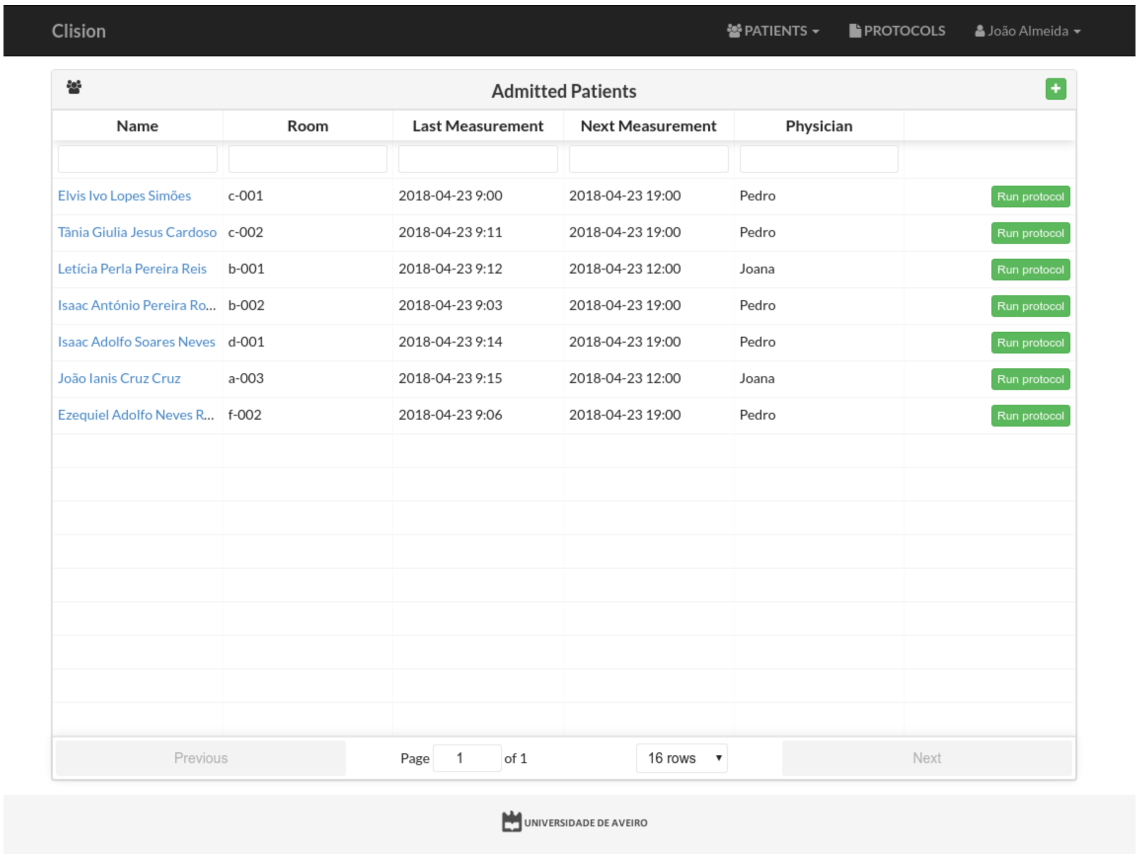


Figure 6.5: Home page.

After the login, the system displays a list of all the admitted patients in the health-care institution (Figure 6.6). This dashboard, shows where the patients are, when the last protocol or data measurements were performed, and when will be the next interaction with the patient. Additionally, it displays the name of the physician responsible for each patient.



Admitted Patients

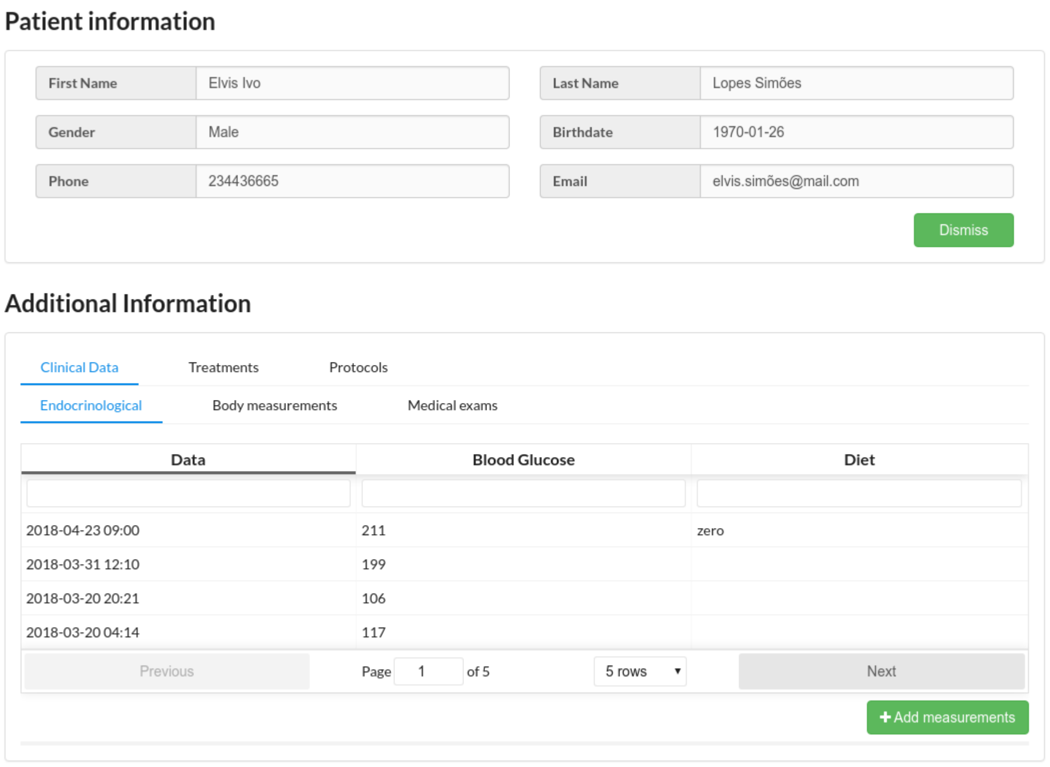
Name	Room	Last Measurement	Next Measurement	Physician	
Elvis Ivo Lopes Simões	c-001	2018-04-23 9:00	2018-04-23 19:00	Pedro	Run protocol
Tânia Giulia Jesus Cardoso	c-002	2018-04-23 9:11	2018-04-23 19:00	Pedro	Run protocol
Letícia Perla Pereira Reis	b-001	2018-04-23 9:12	2018-04-23 12:00	Joana	Run protocol
Isaac António Pereira Ro...	b-002	2018-04-23 9:03	2018-04-23 19:00	Pedro	Run protocol
Isaac Adolfo Soares Neves	d-001	2018-04-23 9:14	2018-04-23 19:00	Pedro	Run protocol
João Ianis Cruz Cruz	a-003	2018-04-23 9:15	2018-04-23 12:00	Joana	Run protocol
Ezequiel Adolfo Neves R...	f-002	2018-04-23 9:06	2018-04-23 19:00	Pedro	Run protocol

Figure 6.6: Initial dashboard.

This layout was used in similar pages, with similar proposes, such as presenting the protocols and all the users in the system. The table in this page provides searching and sorting features for each column. Additionally, in the top right corner of the table, there is a button to open a new page, where new entries in the system can be added. Depending on the table, this entry can be a new user or a new protocol.

Following the navigation pipeline, if the user clicks in the patient's name, a page with all the patient's information is open. In this page (Figure 6.7), there are two groups of patient's data. The first group shows information about the patient, allowing the patient's identification. The other group, named as *Additional Information*, manages all the patient's clinical data, treatments' history and protocols assigned at the moment.

The clinical variables are configurable at runtime, as it was explained before. Additionally, only the administrator has permissions to do this task. Thus, these functionalities are made only in the backend console.



Patient information

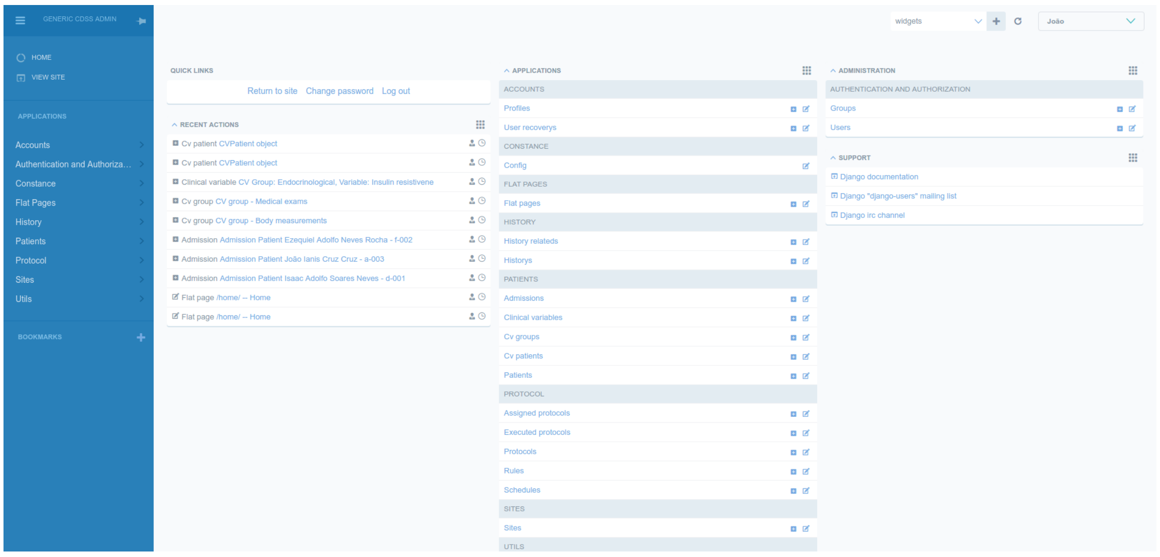
First Name	Elvis Ivo	Last Name	Lopes Simões
Gender	Male	Birthdate	1970-01-26
Phone	234436665	Email	elvis.simões@mail.com

Additional Information

Data	Blood Glucose	Diet
2018-04-23 09:00	211	zero
2018-03-31 12:10	199	
2018-03-20 20:21	106	
2018-03-20 04:14	117	

Figure 6.7: Patient information visible to the health professionals.

Figure 6.8 shows the backend initial dashboard. This dashboard is the administrator's home page and can be customized, with some shortcuts or data analytics.



RECENT ACTIONS

- Cv patient CVPatient object
- Cv patient CVPatient object
- Clinical variable CV Group: Endocrinological, Variable: Insulin resistivene
- Cv group CV group - Medical exams
- Cv group CV group - Body measurements
- Admission Admission Patient Ezequiel Adolfo Neves Rocha - f-002
- Admission Admission Patient João Ianis Cruz Cruz - a-003
- Admission Admission Patient Isaac Adolfo Soares Neves - d-001
- Flat page /home/ -- Home
- Flat page /home/ -- Home

APPLICATIONS

ACCOUNTS
- Profiles
- User recoverys

CONSTANCE
- Config

FLAT PAGES
- Flat pages

HISTORY
- History relateds
- Historys

PATIENTS
- Admissions
- Clinical variables
- Cv groups
- Cv patients
- Patients

PROTOCOL
- Assigned protocols
- Executed protocols
- Protocols
- Rules
- Schedules

SITES
- Sites

UTILS

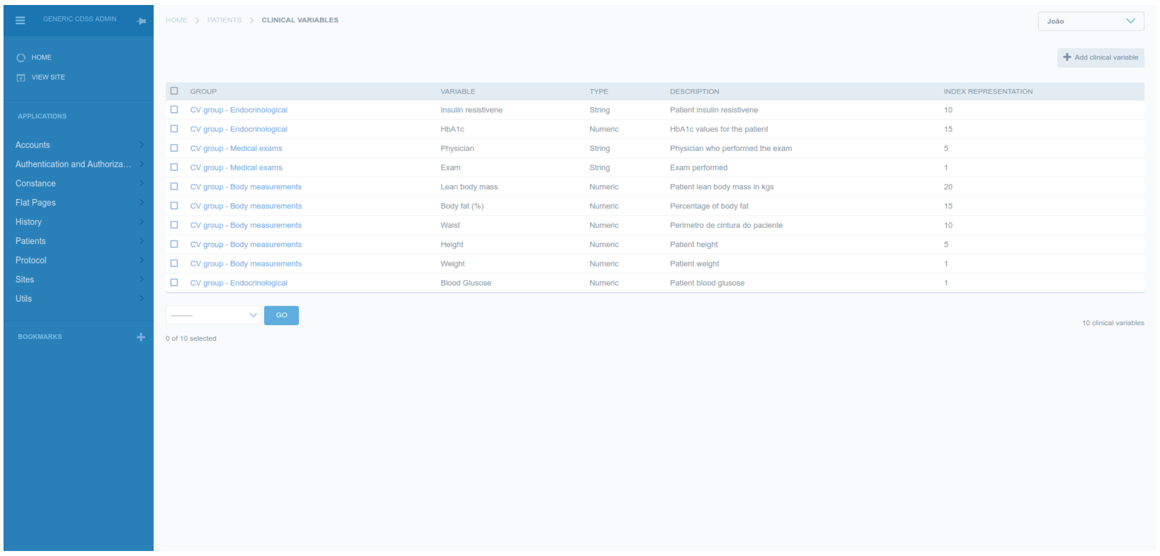
ADMINISTRATION

AUTHENTICATION AND AUTHORIZATION
- Groups
- Users

SUPPORT
- Django documentation
- Django "django-users" mailing list
- Django irc channel

Figure 6.8: Backend dashboard.

Still in the administrator console analysis, in Figure 6.9 the clinical variables table is shown. In this table, new clinical variables can be set up to be presented in the patient information page.



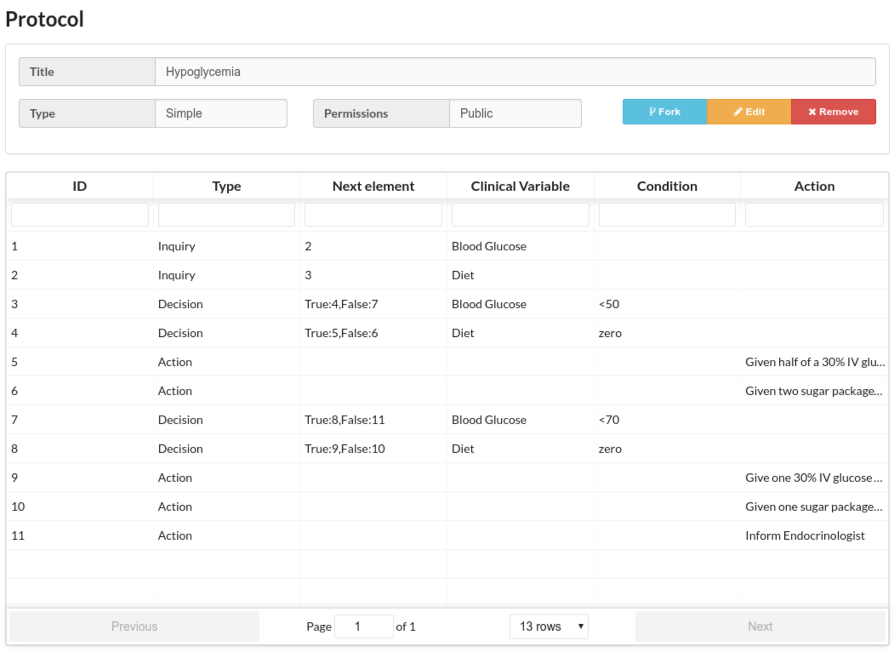
GROUP	VARIABLE	TYPE	DESCRIPTION	INDEX REPRESENTATION
CV group - Endocrinological	Insulin resistivene	String	Patient insulin resistivene	10
CV group - Endocrinological	HbA1c	Numeric	HbA1c values for the patient	15
CV group - Medical exams	Physician	String	Physician who performed the exam	5
CV group - Medical exams	Exam	String	Exam performed	1
CV group - Body measurements	Lean body mass	Numeric	Patient lean body mass in kgs	20
CV group - Body measurements	Body fat (%)	Numeric	Percentage of body fat	15
CV group - Body measurements	Waist	Numeric	Perimetro de cintura do paciente	10
CV group - Body measurements	Height	Numeric	Patient height	5
CV group - Body measurements	Weight	Numeric	Patient weight	1
CV group - Endocrinological	Blood Glusose	Numeric	Patient blood glusose	1

0 of 10 selected

10 clinical variables

Figure 6.9: Backend database management (Configuring new clinical variables).

In this case, the administrator needs to choose the variable group, the variable name, and its type. Additionally, it should be defined as the index representation, which allows the administrator to better organize the position of the menus.



Title	Hypoglycemia
Type	Simple
Permissions	Public

ID	Type	Next element	Clinical Variable	Condition	Action
1	Inquiry	2	Blood Glucose		
2	Inquiry	3	Diet		
3	Decision	True:4,False:7	Blood Glucose	<50	
4	Decision	True:5,False:6	Diet	zero	
5	Action				Given half of a 30% IV glu...
6	Action				Given two sugar package...
7	Decision	True:8,False:11	Blood Glucose	<70	
8	Decision	True:9,False:10	Diet	zero	
9	Action				Give one 30% IV glucose...
10	Action				Given one sugar package...
11	Action				Inform Endocrinologist

Page 1 of 1 13 rows

Figure 6.10: Protocol creation page (Visualization mode).

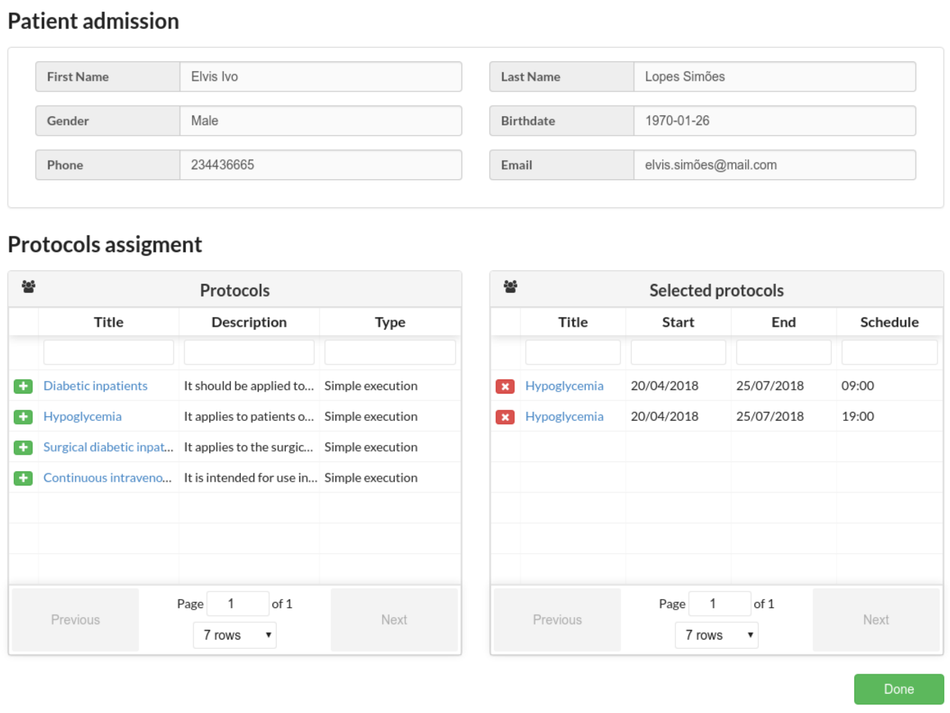
Besides managing the patient's data, the system was also designed to guide clinical protocols. Figure 6.10 shows the implementation of the hypoglycemia protocol (described in the section 2.2). Each protocol is implemented in a workflow structure, following some rules. For each element, the id, the type, and the next element, when it exists, are defined. However, each element type has more individual parameters that are not common between each other.

In Figure 6.10, there are six columns to present the protocol elements and structure. The first column, the ID parameter, identifies the rules, and this is the value used for the next protocol element.

The Clinical Variable identifies the patient's clinical variable and only one of the existent ones in the database can be chosen. However, not all the rules will need a clinical variable. For instance, the protocol element from the Action type does not need any clinical variable, since it is the result of the previous conditions.

The Condition will be a boolean expression that will redirect to one of two elements, this column is only required for protocol elements of type Decision. However, the Decision type has another way to perform the decision. If no condition has been defined, it will behave like a *switch-case*.

Finally, the Action column has a suggestion to the user at the end of the protocol's execution. However, some protocols can have more than one action, and, depending on the execution type, the actions will be shown in the end, or during the execution.


Patient admission

First Name	Elvis Ivo	Last Name	Lopes Simões
Gender	Male	Birthdate	1970-01-26
Phone	234436665	Email	elvis.simões@mail.com

Protocols assigment

Protocols

Title	Description	Type
Diabetic inpatients	It should be applied to...	Simple execution
Hypoglycemia	It applies to patients o...	Simple execution
Surgical diabetic inpat...	It applies to the surgic...	Simple execution
Continuous intraveno...	It is intended for use in...	Simple execution

Selected protocols

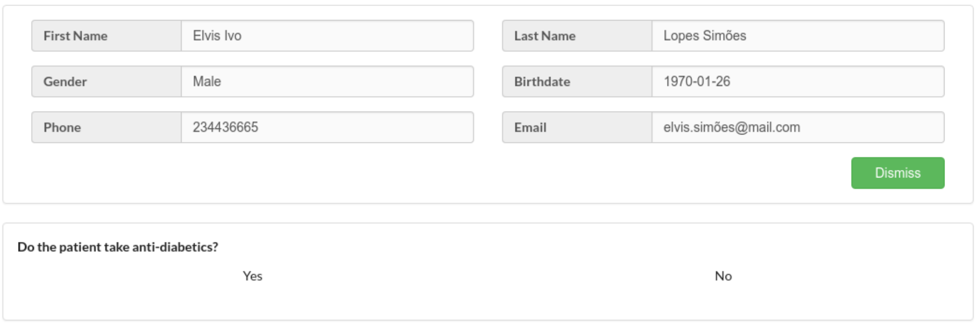
Title	Start	End	Schedule
Hypoglycemia	20/04/2018	25/07/2018	09:00
Hypoglycemia	20/04/2018	25/07/2018	19:00

Figure 6.11: Protocol assignments.

Through the protocol assignment, represented in Figure 6.11, the user can schedule the execution of several protocols to a patient. However, for the diabetes scenario, for each patient, it will only be assigned one protocol with different schedules.

After all these steps, the protocols can be executed. The scheduling only remembers the user about the execution time. Therefore, the system has no way to force the user to execute a protocol at a given moment, or forbid the user from doing it, if it is not the right time.

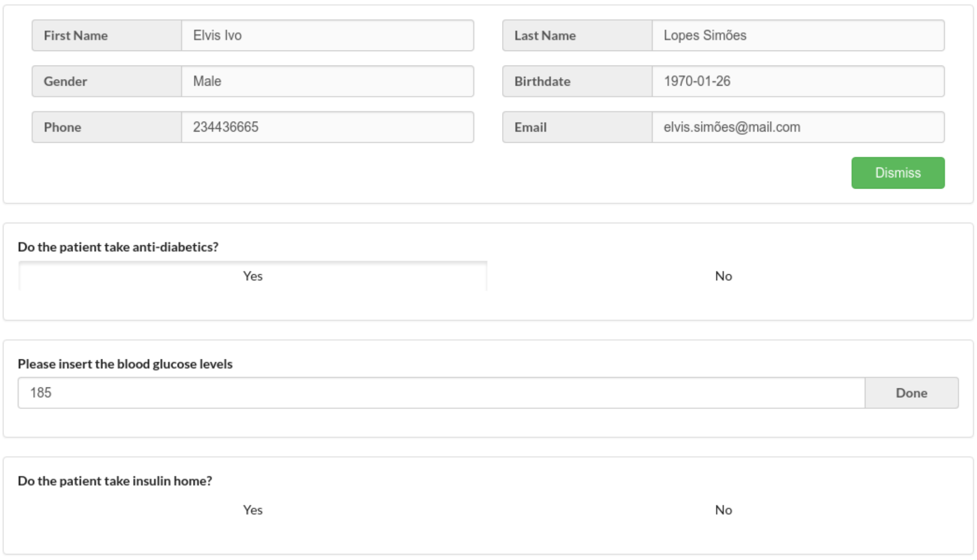
Figure 6.12 shows the execution of a complex protocol, where, for each step, the user needs to interact with the system. The protocol used in the CHBV does not need this type of execution. Thus, it was used as a protocol for testing purposes, since the health professionals found this feature very useful in different scenarios.



First Name	Elvis Ivo	Last Name	Lopes Simões
Gender	Male	Birthdate	1970-01-26
Phone	234436665	Email	elvis.simões@mail.com

Do the patient take anti-diabetics?

(a)



First Name	Elvis Ivo	Last Name	Lopes Simões
Gender	Male	Birthdate	1970-01-26
Phone	234436665	Email	elvis.simões@mail.com

Do the patient take anti-diabetics?

Please insert the blood glucose levels

185

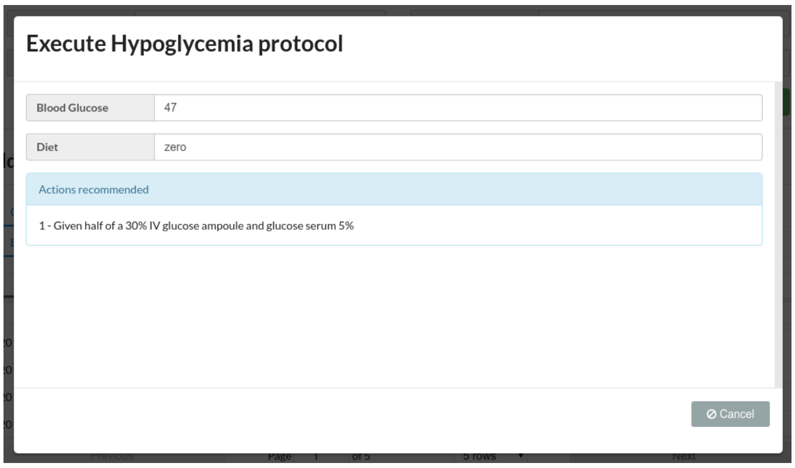
Do the patient take insulin home?

(b)

Figure 6.12: Protocol execution (Complex type). a) Starting the protocol execution; b) Third step of the protocol execution.

The first element of this protocol is the type Inquiry, where a question is defined in the Condition column. The yes or no answer is related with the True or False in the Next Element column for redirecting proposes.

Finally, Figure 6.13 describes the protocol execution, as it was required by the health professionals from the CHBV. In this case, the Inquiry protocol elements are aggregated and presented to the end-user, which inserts the patient's clinical data required in these rules. Then, it provides a list of recommendations based on the defined conditions and variables values.


Execute Hypoglycemia protocol

Blood Glucose	47
Diet	zero

Actions recommended

1 - Given half of a 30% IV glucose ampoule and glucose serum 5%

Figure 6.13: Protocol execution (Simple type).

6.5 Summary

This chapter described the principal implementation steps, following how to set up the developed solution. Additionally, it analyses and details the main system features, presenting a pipeline from the login stage until the execution of an assigned protocol.

CHAPTER 7

Conclusion

"The end is never the end. It's always the beginning of something."
— Kate Lord Brown, The Perfume Garden

In the previous chapters, several phases of the system conception were described, from the requirement analyses until the implementation. After a detailed analysis of the problem and the current state of the art, the decision was made to build a new system.

The system developed was well detailed in the chapters 4, 5 and 6, given an explanation of its conception, concerning the architectural aspects, focusing on the principal concerns, followed by the implementation features.

The project was made in collaboration with the health professionals of the CHBV. Also, they accompanied the solution development process. This cooperation led to new ideas, different visions of the problem and the use of methodologies to analyze the question with the help of professionals from different knowledge areas.

The biggest challenge of this project was the problem analyses and the system design. Defining all the technical needs for a problem, which has a lot of medical specifications demanded a huge effort to understanding all the issues and the requirements needed for its conception. The system implementation has also some interesting challenges, which should also be considered.

The project also had a scientific contribution, a publication in the IEEE 31th International Symposium on Computer-Based Medical Systems (CBMS), entitled as *Simplifying the digitization of clinical protocols for diabetes management* [11].

Additionally, the conceived system is, hopefully, the biggest scientific contribution. One of the main goals was creating a tool to manage clinical treatment protocols. However, the biggest intention was the creation of a tool able to increase the inpatient's health quality. Therefore, the resulting system fulfills the health professional needs and it also fulfills the non-functional requirements. However, this does not mean that the system comes to an end.

Even though this solution fulfills the specified requirements, there are still some points that can be improved. Some of these emerged in the requirements phase, but due to the complexity, some of them could not be developed; others emerged in the development. Concerning this, above is described what can be done and how it should be done in the current system.

It is important connect this system to others systems that already exist in the health institutions. This feature was considered as secondary for the development plan. The goal was to have a flexible system, but if possible to integrate this generic solution with other systems.

The best way is to create a module prepared to communicate with others systems via HL7 [56]. In the system back-end, another component can be created, only responsible for this task. Afterwards, it could be configured like the patient's clinical variables, without compromising the system's flexibility. The idea is to allow the system's administrator to identify the right variables and services in the external system and map them, considering the internal variables or the patient's clinical variables, which are also configurable as described previously.

During the final meetings with the health professionals from CHBV, a new interesting requirement was found. A way to test the protocol and check automatically if it is well implemented in the system. This testing feature should guide the specialist in the protocol implementation and detect some errors during the protocol's digitization.

References

[1] R. A. Greenes, *Clinical decision support: the road ahead.* Academic Press, 2011.

[2] R. Shulman, M. Singer, J. Goldstone, and G. Bellingan, “Medication errors: A prospective cohort study of hand-written and computerised physician order entry in the intensive care unit”, *Critical Care*, vol. 9, no. 5, R516, 2005.

[3] G. Schiff, M. Amato, T. Eguale, J. Boehne, A. Wright, *et al.*, “Computerised physician order entry-related medication errors: Analysis of reported errors and vulnerability testing of current systems”, *BMJ Qual Saf*, vol. 24, no. 4, pp. 264–271, 2015.

[4] M. S. Donaldson, J. M. Corrigan, L. T. Kohn, *et al.*, *To err is human: building a safer health system.* National Academies Press, 2000, vol. 6.

[5] A. Baker, “Crossing the quality chasm: A new health system for the 21st century”, *BMJ: British Medical Journal*, vol. 323, no. 7322, p. 1192, 2001.

[6] D. W. Bates, L. L. Leape, D. J. Cullen, N. Laird, L. A. Petersen, *et al.*, “Effect of computerized physician order entry and a team intervention on prevention of serious medication errors”, *Jama*, vol. 280, no. 15, pp. 1311–1316, 1998.

[7] E. S. Berner and T. J. La Lande, “Overview of clinical decision support systems”, in *Clinical decision support systems*, Springer, 2016, pp. 1–17.

[8] D. Blumenthal and M. Tavenner, “The “meaningful use” regulation for electronic health records”, *N Engl J Med*, vol. 2010, no. 363, pp. 501–504, 2010.

[9] S. Anakal and P. Sandhya, “Clinical decision support system for chronic obstructive pulmonary disease using machine learning techniques”, in *Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2017 International Conference on*, IEEE, 2017, pp. 1–5.

[10] B. Cánovas-Segura, F. Zerbato, B. Oliboni, C. Combi, M. Campos, *et al.*, “A process-oriented approach for supporting clinical decisions for infection management”, in *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, IEEE, 2017, pp. 91–100.

[11] J. R. Almeida, J. Guimarães, and J. L. Oliveira, "Simplifying the digitization of clinical protocols for diabetes management", in *Computer-Based Medical Systems (CBMS), 2018 IEEE 31th International Symposium on*, IEEE, 2018.

[12] S. E. Inzucchi, "Management of hyperglycemia in the hospital setting", *New England journal of medicine*, vol. 355, no. 18, pp. 1903–1911, 2006.

[13] G. E. Umpierrez, R. Hellman, M. T. Korytkowski, M. Kosiborod, G. A. Maynard, *et al.*, "Management of hyperglycemia in hospitalized patients in non-critical care setting: An endocrine society clinical practice guideline", *The Journal of Clinical Endocrinology & Metabolism*, vol. 97, no. 1, pp. 16–38, 2012.

[14] G. E. Umpierrez, R. Gianchandani, D. Smiley, S. Jacobs, D. H. Wesorick, *et al.*, "Safety and efficacy of sitagliptin therapy for the inpatient management of general medicine and surgery patients with type 2 diabetes: A pilot, randomized, controlled study", *Diabetes Care*, vol. 36, no. 11, pp. 3430–3435, 2013.

[15] Y. Kim, K. B. Rajan, S. A. Sims, K. E. Wroblewski, and S. Reutrakul, "Impact of glycemic variability and hypoglycemia on adverse hospital outcomes in non-critically ill patients", *Diabetes research and clinical practice*, vol. 103, no. 3, pp. 437–443, 2014.

[16] A. Neinstein, H. W. MacMaster, M. M. Sullivan, and R. Rushakoff, "A detailed description of the implementation of inpatient insulin orders with a commercial electronic health record system", *Journal of diabetes science and technology*, vol. 8, no. 4, pp. 641–651, 2014.

[17] D. J. Wexler, P. Shrader, S. M. Burns, and E. Cagliero, "Effectiveness of a computerized insulin order template in general medical inpatients with type 2 diabetes: A cluster randomized trial", *Diabetes Care*, vol. 33, no. 10, pp. 2181–2183, 2010.

[18] S. Shetty, S. Inzucchi, P. Goldberg, D. Cooper, M. Siegel, *et al.*, "Adapting to the new consensus guidelines for managing hyperglycemia during critical illness: The updated yale insulin infusion protocol", *Endocrine Practice*, vol. 18, no. 3, pp. 363–370, 2011.

[19] J. Davidson, P. Vexiau, D. Cucinotta, J. Vaz, and R. Kawamori, "Biphasic insulin aspart 30: Literature review of adverse events associated with treatment", *Clinical therapeutics*, vol. 27, S75–S88, 2005.

[20] R. G. Bretzel, S. Arnolds, J. Medding, and T. Linn, "A direct efficacy and safety comparison of insulin aspart, human soluble insulin, and human premix insulin (70/30) in patients with type 2 diabetes", *Diabetes Care*, vol. 27, no. 5, pp. 1023–1027, 2004.

[21] H. Rathore, A. Al-Ali, A. Mohamed, X. Du, and M. Guizani, "Dlrt: Deep learning approach for reliable diabetic treatment", in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–6.

[22] J. Oliveira, "A shotgun wedding: Business decision support meets clinical decision support.", *Journal of healthcare information management: JHIM*, vol. 16, no. 4, pp. 28–33, 2002.

[23] K. B. DeGruy, "Healthcare applications of knowledge discovery in databases", *Journal of healthcare information management*, vol. 14, no. 2, pp. 59–70, 2000.

[24] D. F. Sittig, A. Wright, S. Meltzer, L. Simonaitis, R. S. Evans, *et al.*, "Comparison of clinical knowledge management capabilities of commercially-available and leading internally-developed electronic health records", *BMC medical informatics and decision making*, vol. 11, no. 1, p. 13, 2011.

[25] E. S. Berner, *Clinical decision support systems.* Springer, 2007, vol. 233.

[26] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, "An artificial intelligence program to advise physicians regarding antimicrobial therapy", *Computers and Biomedical Research*, vol. 6, no. 6, pp. 544–560, 1973.

[27] R. A. Miller, H. E. Pople Jr, and J. D. Myers, "Internist-i, an experimental computer-based diagnostic consultant for general internal medicine", *New England Journal of Medicine*, vol. 307, no. 8, pp. 468–476, 1982.

[28] G. M. Marakas, *Decision support systems in the 21st century.* Prentice Hall Upper Saddle River, NJ, 2003, vol. 134.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[30] K. Fukunaga, *Introduction to statistical pattern recognition.* Academic press, 2013.

[31] B. Ozaydin, J. M. Hardin, and D. C. Chhieng, "Data mining and clinical decision support systems", in *Clinical Decision Support Systems*, Springer, 2016, pp. 45–68.

[32] J. K. Tan and S. B. Sheps, *Health decision support systems.* Jones & Bartlett Learning, 1998.

[33] E. S. Berner, *Clinical Decision Support Systems: Theory and Practice.* Springer, 2016.

[34] S. Weiss, C. A. Kulikowski, and A. Safir, "Glaucoma consultation by computer", *Computers in Biology and Medicine*, vol. 8, no. 1, pp. 25–40, 1978.

[35] A. A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q. T. Zeng, *et al.*, "Glif3: A representation format for sharable computer-interpretable clinical practice guidelines", *Journal of biomedical informatics*, vol. 37, no. 3, pp. 147–161, 2004.

[36] M. Peleg, A. A. Boxwala, O. Ogunyemi, Q. Zeng, S. Tu, *et al.*, "Glif3: The evolution of a guideline representation format.", in *Proceedings of the AMIA Symposium*, American Medical Informatics Association, 2000, p. 645.

[37] D. Wang and E. H. Shortliffe, "Glee–a model-driven execution system for computer-based implementation of clinical practice guidelines.", in *Proceedings of the AMIA Symposium*, American Medical Informatics Association, 2002, p. 855.

[38] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, O. Ogunyemi, *et al.*, "Design and implementation of the glif3 guideline execution engine", *Journal of biomedical informatics*, vol. 37, no. 5, pp. 305–318, 2004.

[39] D. Wang, M. Peleg, D. Bu, M. Cantor, G. Landesberg, *et al.*, "Gesdor–a generic execution model for sharing of computer-interpretable clinical practice guidelines", in *AMIA Annual Symposium Proceedings*, American Medical Informatics Association, vol. 2003, 2003, p. 694.

[40] I. Maglogiannis, "Towards the adoption of open source and open access electronic health record systems", *Journal of Healthcare Engineering*, vol. 3, no. 1, pp. 141–161, 2012.

[41] A. Syzdykova, A. Malta, M. Zolfo, E. Diro, and J. L. Oliveira, "Open-source electronic health record systems for low-resource settings: Systematic review", *JMIR medical informatics*, vol. 5, no. 4, 2017.

[42] P. V. Kukhareva, K. Kawamoto, D. E. Shields, D. T. Barfuss, A. M. Halley, *et al.*, "Clinical decision support-based quality measurement (cds-qm) framework: Prototype implementation, evaluation, and future directions", in *AMIA Annual Symposium Proceedings*, American Medical Informatics Association, vol. 2014, 2014, p. 825.

[43] *ICE - CDS Framework Wiki.* [Online]. Available: `https://cdsframework.atlassian.net/wiki/spaces/ICE/overview` (visited on 06/26/2018).

[44] H. Khan and L. Hederman, "A universal clinical decision support system using semantic web services", ESWC, 2012.

[45] J. Bellows, S. Patel, and S. S. Young, "Use of indigo individualized clinical guidelines in primary care", *Journal of the American Medical Informatics Association*, vol. 21, no. 3, pp. 432–437, 2013.

[46] G. McCallum, "Egadss: A clinical decision support system for use in a service-oriented architecture", PhD thesis, 2006.

[47] P. C. Davidson, R. D. Steed, and B. W. Bode, "Glucommander: A computer-directed intravenous insulin system shown to be safe, simple, and effective in 120,618 h of operation", *Diabetes care*, vol. 28, no. 10, pp. 2418–2423, 2005.

[48] S. A. Nisly, S. Harris, L. Aykroyd, J. Carrol, B. Ulmer, *et al.*, "Use of a subcutaneous insulin computerized glucostabilizer$^{\mathrm{TM}}$ program on glycemic control in the intensive care setting: A retrospective data analysis.", *Advances in Diabetes and Metabolism*, vol. 1, no. 1, p. 29, 2013.

[49] R. J. Baron, E. L. Fabens, M. Schiffman, and E. Wolf, "Electronic health records: Just around the corner? or over the cliff?", *Annals of internal medicine*, vol. 143, no. 3, pp. 222–226, 2005.

[50] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[51] M. Richards, *Software architecture patterns*. O'Reilly Media, Incorporated, 2015.

[52] D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (rbac): Features and motivations", in *Proceedings of 11th annual computer security application conference*, 1995, pp. 241–48.

[53] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment", *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[54] M. Carver, *The responsive web*. Manning Publications Co., 2014.

[55] K. Naik and P. Tripathy, *Software testing and quality assurance: theory and practice*. John Wiley & Sons, 2011.

[56] T. Benson, *Principles of health interoperability HL7 and SNOMED*. Springer London: 2010.

[57] J. Almeida, R. Ribeiro, and J. L. Oliveira, "A modular workflow management framework", in *Proceedings of the 11th International Conference on Health Informatics (HealthInf 2018)*, 2018.

[58] J. R. Almeida, T. M. Godinho, L. Bastião Silva, C. Costa, and J. L. Oliveira, "Services orchestration and workflow management in distributed medical imaging environments", in *Computer-Based Medical Systems (CBMS), 2018 IEEE 31th International Symposium on*, IEEE, 2018.

[59] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: A survey of techniques and tools", *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 33, 2016.

[60] D. Greenfeld and A. Roy, *Two Scoops of Django: Best Practices for Django 1.6*. Two Scoops Press, 2014.

[61] L. B. Silva, A. Trifan, and J. L. Oliveira, "Montra: An agile architecture for data publishing and discovery", *Computer methods and programs in biomedicine*, vol. 160, pp. 33–42, 2018.

[62] *Home - Django REST framework*. [Online]. Available: `http://www.django-rest-framework.org/` (visited on 07/07/2018).

[63] C. Vitolo, Y. Elkhatib, D. Reusser, C. J. Macleod, and W. Buytaert, "Web technologies for environmental big data", *Environmental Modelling & Software*, vol. 63, pp. 185–198, 2015.

[64] *Home - Django REST framework.* [Online]. Available: http://www.django-rest-framework.org/api-guide/serializers/ (visited on 07/07/2018).

[65] J. E. Smith and R. Nair, "The architecture of virtual machines", *Computer*, vol. 38, no. 5, pp. 32–38, 2005.

[66] C. Anderson, "Docker [software engineering]", *IEEE Software*, vol. 32, no. 3, pp. 102–c3, 2015.

[67] J. Turnbull, *The Docker Book: Containerization is the new virtualization.* James Turnbull, 2014.

[68] *Library/postgres - Docker Hub.* [Online]. Available: https://hub.docker.com/_/postgres/ (visited on 07/04/2018).

[69] *| Docker Documentation.* [Online]. Available: https://docs.docker.com/engine/reference/builder/ (visited on 07/04/2018).

[70] *NGINX Docs | Welcome to NGINX documentation*, en. [Online]. Available: https://docs.nginx.com/index/ (visited on 07/04/2018).

[71] *The uWSGI project — uWSGI 2.0 documentation.* [Online]. Available: https://uwsgi-docs.readthedocs.io/en/latest/ (visited on 07/04/2018).

[72] *Celery - Distributed Task Queue — Celery 4.2.0 documentation.* [Online]. Available: http://docs.celeryproject.org/en/latest/index.html (visited on 07/04/2018).

[73] *Overview of Docker Compose*, en, Jun. 2018. [Online]. Available: https://docs.docker.com/compose/overview/ (visited on 07/04/2018).

[74] *GNU make.* [Online]. Available: https://www.gnu.org/software/make/manual/make.html (visited on 07/04/2018).

APPENDIX A

Web Services

- **Accounts**

  <baseurl>/api/accounts

- **Protocols**

  <baseurl>/api/protocols/protocol
  <baseurl>/api/protocols/schedule
  <baseurl>/api/protocols/assignedprotocols
  <baseurl>/api/protocols/executedprotocols

- **Patients**

  <baseurl>/api/patients/patient
  <baseurl>/api/patients/clinicalvariables
  <baseurl>/api/patients/admission

- **Utils**

  <baseurl>/api/utils/footer
  <baseurl>/api/utils/header
  <baseurl>/api/utils/language
  <baseurl>/api/utils/about
  <baseurl>/api/utils/help
  <baseurl>/api/utils/home

APPENDIX B

Application Links

- **Public links**

  <baseurl>/
  <baseurl>/signinup
  <baseurl>/profile
  <baseurl>/forgotten
  <baseurl>/help
  <baseurl>/about

- **Authenticated links**

  <baseurl>/admittedpatients
  <baseurl>/allpatients
  <baseurl>/run/protocol/:object/:patient
  <baseurl>/protocols
  <baseurl>/patient/:object
  <baseurl>/show/protocol/:object
  <baseurl>/add/patient
  <baseurl>/assignprotocol/:object

- **Authenticated links**

  <baseurl>/0
  <baseurl>/500
  <baseurl>/404