UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Electronic health record exchange tools directed toward patients with chronic diseases

Dissertation presented by
Julie CHATELAIN

for obtaining the master's degree in
Computer science

Supervisor(s)
Benoit MACQ

Readers(s)
Axel Legay, Nicolas Vincent

Academic year 2016-2017

Table of content

List of Abbreviations and Symbols

ACL	Access Control List
EHR	Electronic Health Record
FHIR	Fast Healthcare Interoperability Resources
IETF	Internet Engineering Task Force
JWT	JSON Web Token
MEAN.js	MongoDB, Express, AngularJS and Node.js
REST	REpresentational State Transfer
RFC	Request For Comments
SNOMED CT	Systematized Nomenclature of Medicine Clinical Terms
WHO	World Health Organization

1. Introduction

The goal of this project is to promote exchanges between different health practitioners and between patients and their practitioners.

The project aims to create an application that will allow health practitioners to create a small electronic health record (EHR, cf. 2.2.) and share it with other practitioners. It will also allow them to send essential data to their patients with chronic diseases. The patients will be able to contribute to their EHR and add their own data.

It will only target the diabetic patients and those suffering from heart failure but offer the possibility to extend it to other chronic diseases.

1.1. Context

Chronic diseases account for 63% of all deaths. They are the first cause of mortality in the world [1].

The number of people suffering from diabetes keeps rising. From 108 million in 1980 to 422 million in 2014. The World Health Organization (WHO) predicts that diabetes will be the 7th leading cause of death in 2030. Diabetes can cause blindness, heart attacks, kidney failures, strokes and lower limb amputation. Those consequences can be avoided or delayed if the condition is correctly managed [2]. The patient needs to monitor, his diet, his glucose level, his physical activity, his medication….

There is a real need for tools to help patients manage their conditions. Among others, the Eglé application (cf. 3.1.) try to answer this need and offer a platform of tools designed to increase patient empowerment and improve medical care.

On the practitioners' side, they deplore the difficulty of gathering all the medical data of a new patient. The sharing of patient records between hospitals and all the different practitioners is not an easy thing. Sometimes a patient ends up being subjected several times to the same analysis and tests because the practitioner ordering it wasn't aware that it had already been done.

A centralized EHR seems to be a solution, but this idea needs to take into account all the strict laws on privacy, medical data protection, patient's rights and doctor-patient confidentiality (cf. 2.3.).

1.2. Objectives

1.2.a. Data Storage

The first goal is to set up a server REST that will enable the storing and sharing of the medical data of the patients. The storage should be focused on data pertaining to diabetes and heart failure but offers the possibility of storing any kind of medical data.

1.2.b. Small Medical Record Application

The second goal is to set up a small local application for the practitioners that allow them to create a mini health record and share it with other practitioners as well as transfer essential information to their patients (ex: analysis results). It will also display the information received from their patients.

1.2.c. Link with Eglé

The third goal is to link the REST server with Eglé (cf. 3.1.) so that patients receive the information shared by their practitioner directly in the application. It should also allow the patient to share information with their practitioner and add data to their health record.

1.2.d. Security

The fourth goal is to respect the privacy law and patient rights and ensure that all the operations are safe.

Data need to be encrypted. Access must be restricted and controlled.

Those questions have to be answered:

- Who can see what?
- Who can add or modify what?
- How to get authorisation to see / add / modify something and how long does the authorisation last? Who can give the authorisation and how?
- How do we keep traces of who has seen what and who has modified or added what and when?
- What's the best (and most secure) way to set up the authentication?
- Which technology is needed to ensure the security?
- What are the possible security risks?
- What are the legal restrictions?

1.2.e. User Experience Design

One of my goal is to follow the user experience design (UX) guide lines to enhance usability and accessibility.

Ideally, I would regularly ask several users (doctors and patients) to test the application and give me feedback on their interaction. I could then adapt the application to their needs.

2. Background

2.1. Chronic diseases

2.1.a. Diabetes Mellitus

Diabetes is a disease that affects the body capacity to produce and / or use the insulin. Insulin is a hormone made by the pancreas, it is needed for the cells to take in the glucose in the blood and use it as fuel.

There's three main types of diabetes:

a. Type I (also called insulin-dependent): it often begins during childhood and it results in the body not producing enough insulin. The cause is unknown and we cannot prevent it. People with this condition need to regularly take insulin. [2]
b. Type II: it happens when the body develops a resistance to insulin and becomes unable to use it efficiently. It can be treated with medication with or without insulin. However the disease can progress until the patient needs injections of insulin. This represents the majority of persons with diabetes in the world. [2]
c. Gestational diabetes appears first during pregnancy. It usually resolves itself after birth.

There is no cure for diabetes. The treatment is lifelong (except for gestational diabetes).

The treatment involves regularly checking their blood sugar level, controlling what they eat, how much they exercise and be on the lookout for complications.

When it is not treated, diabetes causes hyperglycaemia, an excess of glucose in the blood. It can damage the small vessels in the eyes, kidneys, heart, feet… and on the long term, it can lead to heart attacks, blindness, strokes, kidney failure, foot ulcer and infections and need for amputation. [2]

(i) HbA1c

HbA1c refers to glycated haemoglobin. It appears when the haemoglobin in the blood combines with glucose [3]. The amount of glucose that joins with the haemoglobin is proportional to the quantity of sugar in the system. Red blood cells survive for approximatively twelve weeks. Therefore, the quantity of HbA1c reflects the average blood sugar's level over the past three months.

It is a really useful information for diabetic patients since the higher the HbA1c is, the higher the risk of developing complications. They often have a level of HbA1c as target that they need to reach or maintain.

There exists two standard to express HbA1c values. The new standard uses a measurement in mmols/mol and the old one reported it with a percentage. Lot of people still uses the percentage unit.

2.1.b. Heart Failure

Heart failure is a cardiovascular disease that is characterized by the heart inability to pump blood efficiently. The heart is unable to maintain the blood flow needed by the organs such as the brain, liver and kidneys [4].

Symptoms of heart failure include:

a. Shortness of breath (dyspnea)
b. Fatigue
c. Swellings in the feet (oedema)
d. Sleep apnea
e. Dizziness
f. Palpitations

Treatment for heart failure involves controlling how much they exercise, keeping track and limiting the amount of salt and liquid they ingest, taking medication and sometimes implanting devices or even having a heart transplant. If lifestyle and diet changes are not made, medication not taken, the disease can progress causing irreversible damages and eventually leading to death [5].

(i) BNP

BNP refers to Brain natriuretic peptide. This is a protein secreted by the heart and released when it stretches. It is especially elevated in patients with heart failure. This is why it is used as a biomarker to determine the severity of heart failure.

2.2. EHR

"An Electronic Health Record (EHR) means a comprehensive medical record or similar documentation of the past and present physical and mental state of health of an individual in electronic form, and providing for ready availability of these data for medical treatment and other closely related purposes."

– Commission Recommendation of 2 July 2008 on cross-border interoperability of electronic health record systems [6]

2.2.a. Advantages

The advantages of EHR are numerous.

It removes the need to search for the patient's previous paper medical records. It also helps ensuring that data is legible. Handwritten reports are not always easily decipherable. The cliché of doctors with terrible handwriting does come from somewhere.

Another good point is that it decreases the risk of lost paperwork. The record is also more likely to be up to date since there's only one "copy". It is more accessible and easily shared. Any processing (like making statistics) is easier with EHR than with paper records.

2.2.b. Drawbacks

There's also some drawbacks with EHR. The biggest one are the privacy and security concerns. Medical data is really sensitive and the law demands that is well protected. Access, authentication, storage, transmission… every operation involves risks.

There's also the three main threats of information security we encounter with any program:

a. Human threats (employees' misuse, hackers…)
b. Natural threats (earthquake, fires…)
c. Technology failures (system crash, bugs…)

Centralized EHR:

A centralized EHR would regroup all the EHR the caregivers and hospitals have on their own computer or network in one point and combine them in one big and complete EHR.

This would multiply the advantages gained with EHR…but also the drawbacks.

"Réseau de santé wallon": it is a network that enable the exchange of EHR between caregivers for a same patient. It usually does not store the EHRs. They stay on their original servers [7].

2.3. Legal Context

There is a lot of laws in Belgium pertaining to the protection of medical data.
Here are some of the most relevant ones:

Loi vie privée du 8 décembre 1992 définit les droits et devoirs de la personne dont les données sont traitées mais aussi ceux du responsable d'un tel traitement:

1. **Définitions** :
    a. **Données médicales** : « *données relatives à une personne physique identifiée ou identifiable et dont on peut déduire une information sur l'état antérieur, actuel ou futur de la santé physique ou psychique.* »
    b. **Traitement de données** : « *toute opération éventuelle appliquée à des données à caractère personnel, telle que la collecte, l'utilisation, la gestion ou la communication.* »

Art. 4. *§ 1er. Les données à caractère personnel doivent être :*
5° conservées sous une forme permettant l'identification des personnes concernées pendant une durée n'excédant pas celle nécessaire à la réalisation des finalités pour lesquelles elles sont obtenues ou pour lesquelles elles sont traitées ultérieurement.

Art. 7. *§ 1er. Le traitement de données à caractère personnel relatives à la santé est interdit.*
§ 2. Exceptions :
a) *lorsque la personne concernée a donné son consentement par écrit à un tel traitement, pour autant que ce consentement puisse à tout moment être retiré par celle-ci;*
h) *lorsque le traitement porte sur des données manifestement rendues publiques par la personne concernée;*
§ 4. Le traitement des données à caractère personnel relatives à la santé peut, sauf dans le cas d'un consentement écrit de la personne concernée ou lorsque le traitement est nécessaire pour la prévention d'un danger concret ou la répression d'une infraction pénale déterminée, uniquement être effectué sous la responsabilité d'un professionnel des soins de santé.

Art. 9. *§ 1er. Le responsable du traitement ou son représentant doit fournir à la personne concernée auprès de laquelle il obtient les données la concernant et au plus tard au moment où ces données sont obtenues.*

Art. 12. *§ 1er. Toute personne a le droit d'obtenir sans frais la rectification de toute donnée à caractère personnel inexacte qui la concerne.*
Toute personne a en outre le droit de s'opposer, pour des raisons sérieuses et légitimes tenant à une situation particulière, à ce que des données la concernant fassent l'objet d'un traitement, sauf lorsque la licéité du traitement est basée sur les motifs visés à l'article 5, b) et c).

Loi sur les droits du patient de 2002 :
Art. 7*. § 1er. Le patient a droit, de la part du praticien professionnel, à toutes les informations qui le concernent et peuvent lui être nécessaires pour comprendre son état de santé et son évolution probable.*

Art. 9*. § 1er. Le patient a droit, de la part de son praticien professionnel, à un dossier de patient soigneusement tenu à jour et conservé en lieu sûr.*
A la demande du patient, le praticien professionnel ajoute les documents fournis par le patient dans le dossier le concernant.
§ 2. Le patient a droit à la consultation du dossier le concernant.
Il est donné suite dans les meilleurs délais et au plus tard dans les 15 jours de sa réception, à la demande du patient visant à consulter le dossier le concernant.

Art. 10*. § 1er. Le patient a droit à la protection de sa vie privée lors de toute intervention du praticien professionnel, notamment en ce qui concerne les informations liées à sa santé.*

Those law put a lot of restrictions on all medically related softwares. The requirements for security are especially high.

3. Related work

3.1. Eglé

The Eglé project aims to improve medical care, increase patient empowerment and foster communication between the patient, his family and health practitioners [8]. To do so it provides a scalable platform of multimedia tools that helps the patient monitor his health on a daily basis. The interface was designed to be intuitive and easy to use.

It targets patients with chronic diseases. For now it is restricted to patients suffering from diabetes or heart failure.

3.1.a. Architecture

Eglé is a web application build using the MEAN.js solution (cf. 4.1.). This choice comes from the desire to offer a platform immediately accessible without the need to download and install a software.

The interface has been implemented using AngularJs, HTML5 and Bootstrap (cf. 4.1. (ix) . It follows the responsive web design approach. It can adapt to any screen (desktop, smartphone…).

An ACL (Access Control List) was implemented with the roles “administrator”, “caregiver” and “patient” to manage the access permissions.

Authentication in Eglé uses JSON web token instead of a session system.

3.1.b. Dashboard

The home page of Eglé is a dashboard made of widgets. A widget is a small panel containing data. The dashboard is customizable. In the parameters, the patient can choose which widget he wants to see on his dashboard and hide those that don’t interest him.

Some widgets appears at fixed intervals to ask the user what’s their weight, how much they exercised today and all the relevant variables needed for the application.

For diabetic patients, there is a widget containing a to-do list (Figure 1: Eglé's todo list) to remind the patient to take his glycaemia levels, check what he ate and to take his dose of insulin. It facilitates the daily encoding of the value and if the patient missed one entry it is immediately visible.

The other widgets display the data collected, mostly in graphs. There’s several different type of graphs and each type was chosen in order to best represent the data collected. For some of them, it is possible to see the average value on the graph or to set objectives with a minimum and a maximum value.
Here is the list of all the implemented widgets:

- *Estimation of glycated haemoglobin*: takes all the value of glycaemia for the past three months and computes an estimation of the HbA1c value. Only for diabetic patients.
- *Number of steps*: monitors the number of steps taken by a patient with heart failure. The value to aim for is generally 10 000 steps per day.
- *Blood pressure*: monitors the blood pressure (diastolic and systolic) of patients with heart failure. The graph indicate the normal range for the blood pressure so that the patient can immediately see when his value is abnormal.
- *Heartrate:* monitors the heartrate of the patient. There's two values: the value after effort and the value at rest.
- *Symptoms:* monitors the evolution of the symptoms of a heart failure patient.
- *Glycaemia:* monitors the glucose level. Only for diabetic patients.
- *Insulin:* monitors the insulin doses. Only for diabetic patients.
- *Weight:* monitors the patient's weight. For both diabetic and heart failure patients.
- *Sport:* monitors the physical activity of the patient.
- *Salt:* monitors the quantity of salt ingested each day by a patient with heart failure.
- *Liquid:* monitors the quantity of liquid ingested each day by a patient with heart failure.

Figure 1: Eglé's todo list

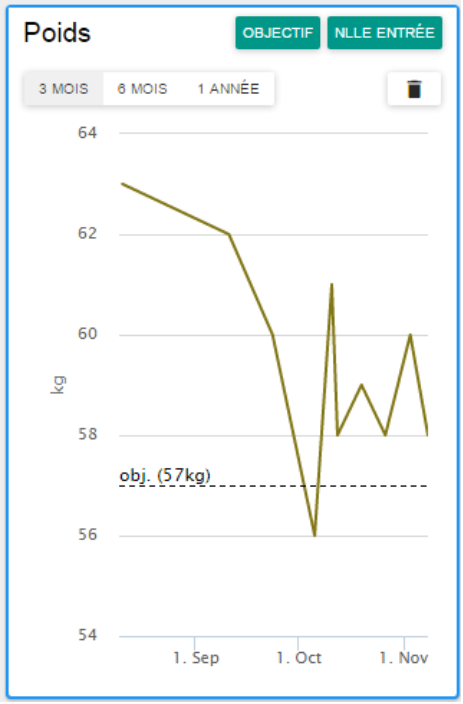

Figure 3: Eglé's weight widget

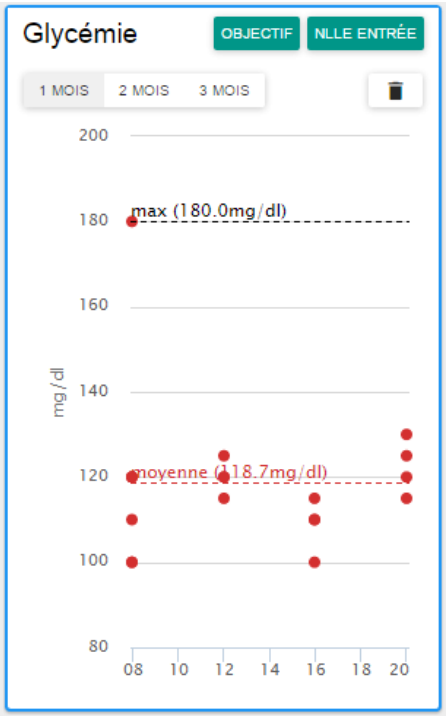

Figure 2 : Eglé's glycaemia widget

3.1.c. Agenda

Eglé propose an agenda for organizing meetings, appointment with the doctors… or any other event for the user.

3.1.d. Chat & video calls

Eglé offers a discussion tool (deferred or real-time). It includes all the common features found in other popular instant messaging applications: contact management, call notifications, call duration, call history, missed calls…

3.1.e. Tips & tricks

This page gathers all the useful information the patient needs to know about his disease. The user also has the option to bookmark some tips and to filter the tips and tricks list according to the bookmarking.
These tips are grouped into 12 categories: diabetes monitoring, glucagon, glucagon injection, glycaemia measurement, insulin: estimation of the dose, insulin: injection and absorption, insulin: different types, needles, glycated haemoglobin, complications, nutrition and travel.

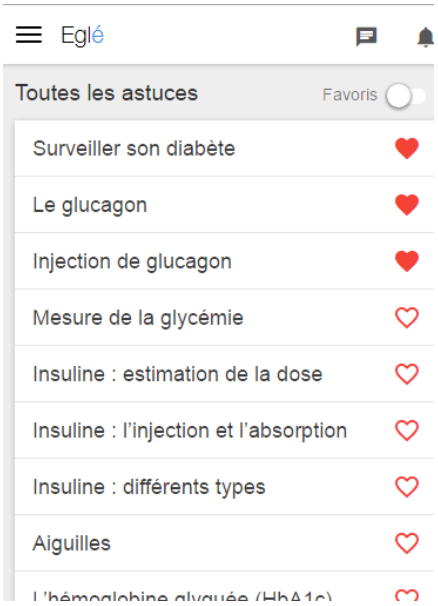

Figure 4: Eglé's tips & tricks

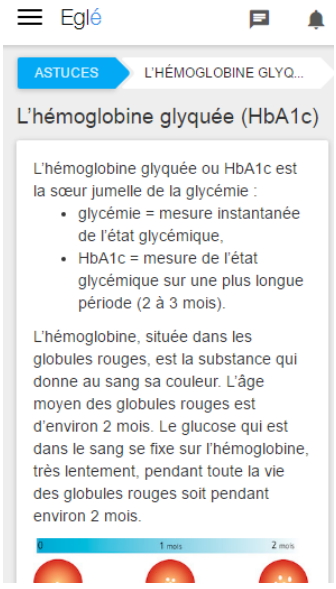

Figure 5: Eglé's "glycated haemogobin" tips

4. Technical choices and analysis

4.1. Resources

Here is the list of the technologies and sources of information I chose to make this project:

(i) FHIR

The acronym means Fasth Healthcare Interoperability Resources. It is a standard describing data formats for clinical concepts and a REST api for exchanging EHR. It is a new specification created by the Health Level-7 (HL7) standards organization [9].

HL7 is a non-profit organization that creates standards and frameworks for the exchange, integration, sharing, and retrieval of electronic health information. The goal of HL7 is to provide standards that empower global health data interoperability so that everyone can safely access and use the right health data when they need it. HL7 has more than 1600 members in over 50 countries around the world. Members include healthcare providers, government stakeholders, pharmaceutical companies… [10].

I chose it because it is designed for the web and the resources are based on JSON / XML structures. It can be easily adapted to a lot of environments and it was conceived to be suitable for use in a (really) wide variety of contexts. It is especially flexible. My choice was also motivated by the fact that it is built to be used with an http-based RESTful service where each resource has predictable URL. That was the solution I planned to use for data storage and sharing.

The basic building block of FHIR is a “resource”. Resources can be thought of as paper “forms” that contains different types of medical information that can be shared. A FHIR resource gives a generic form template for clinical or administrative information such as conditions, symptoms, prescriptions, practitioner or patient information…

Each resource contains a small amount of highly specific data. A single resource doesn’t say a lot, but several resources linked together can make up a full medical record.

Resources are rather generic since they were built to be usable in a lot of different contexts. FHIR provides the option to adjust the resources to handle specific needs but it wants to keep the base form that everyone uses from being too complex. And in order to keep the number of different resources reasonable, some resource defines really broad concepts such as the “Observation” resource that can be used to describe lab results, vital signs, risk factors… [11]

All resources share characteristics:

- A definition and representation built using “data types” (common reusable patterns).
- A set of metadata containing: version id, last updated, date of creation, created by, date of publication, updated by and a human readable part that contains a summary of the resource.
- A human readable part.

Example of a "meta" attribute of a patient resource:

```
{
      "versionId" : "3",
      "created" : "2014-08-18T15:43:30Z",
      "lastUpdated" : "2015-08-18T15:43:30Z",
      "createdBy" : "Patient/1238df568fds5",
      "published" : "2017-01-23T15:43:30Z ",
      "updatedBy" : "Patient/1238df568fds5",
      "text": {
                      "status": "generated",
                     "div": "<p><strong>Patient</strong><br>
                            <strong>Name : </strong> Jean Dupont<br></p>"
                }
}
```

One of the main data type used by FHIR is the "Coding" one, included into the "Codeable Concept" data type.

```
{
  "system" : "<uri>",         // Identity of the terminology system
  "version" : "<string>",     // Version of the system - if relevant
  "code" : "<code>",          // Symbol in syntax defined by the system
  "display" : "<string>",     // Representation defined by the system
  "userSelected" : <boolean> // If this coding was chosen directly by
the user
}
```

(This data type definition was taken directly for the FHIR website [12]*)*

It is used to describe a resource with a code system. For example, here is what we could find in a resource "Observation" made for a patient suffering from headaches:

```
{
  "system" : http://snomed.info/sct",
  "code" : "25064002",
  "display" : "Headache"
}
```

Since the information contained in a resource is so highly specific, it needs to make a lot of reference to other resources to make sense in the larger context. There is two type of references: reference to a resource in the system and external reference to resource found elsewhere.

Reference appears under the form of a URL, either absolute or relative rather than by id key as is usually done. That way, the client application can directly ask the referenced resource by making a request to the specified URL.

There is a specific “reference” data type used to encode it. It includes a text description that briefly summarizes the resource linked by the reference. This text description is highly useful when we just need to know what the reference is about without needing the full resource. It saves superfluous requests to the server.

```
{
  "reference" : "<string>",
  "display" : "<string>"
}
```

On top of the id generated by the database, each resource also has an id that is the URL of the resource on the server. This id is the element found in the reference of other resources.

I used the JavaScript implementation available on the FHIR website (licensed under Creative Commons "No Rights Reserved [13]) that generates Mongoose models for FHIR resources as base for my project.

(ii) SNOMED CT

SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) is a really precise clinical health terminology. It is composed of a code system for a collection of clinical concepts organized into class hierarchies, descriptions that link the concepts to human readable terms and relationships that link the different concepts to other related concepts [14].

Here are some of the top level hierarchies concepts found in SNOMED:

- “*Body structure*” : this branch represents normal and abnormal anatomical structures
- “*Clinical finding*”: represents the results of a clinical observation. It includes the concepts used to define a diagnoses.
- “*Physical object*”: represents any object, natural or man-made.
- “*Procedure*”: represents any activities occurring during healthcare. It includes anything from surgical operations to administration of medicines.
- “*Substance*”: represents any substance, chemical constituent of product as well as body fluids.
- “*Social context*”: represents social circumstances that could be relevant to healthcare, such as a person’s job or economic status or life style…
- “*Observable entity*”: represents an observation that can be useful to produce an answer to a question, such as the colour of or iris or the diastolic blood pressure.

Within a hierarchy, concepts are organized from the more generic to the more detailed. Concepts in a hierarchy are linked using the “is a” relationship. For example, far under the “body structure” concept the “Nucleated red blood cell” concept can be found. Each concept can have several supertype parent concepts. This forms a “polyhierarchy” structure.

“Is a” is not the only relationship possible between concept. SNOMED defines a whole set of relationships such as “finding site” or “causative agent”.

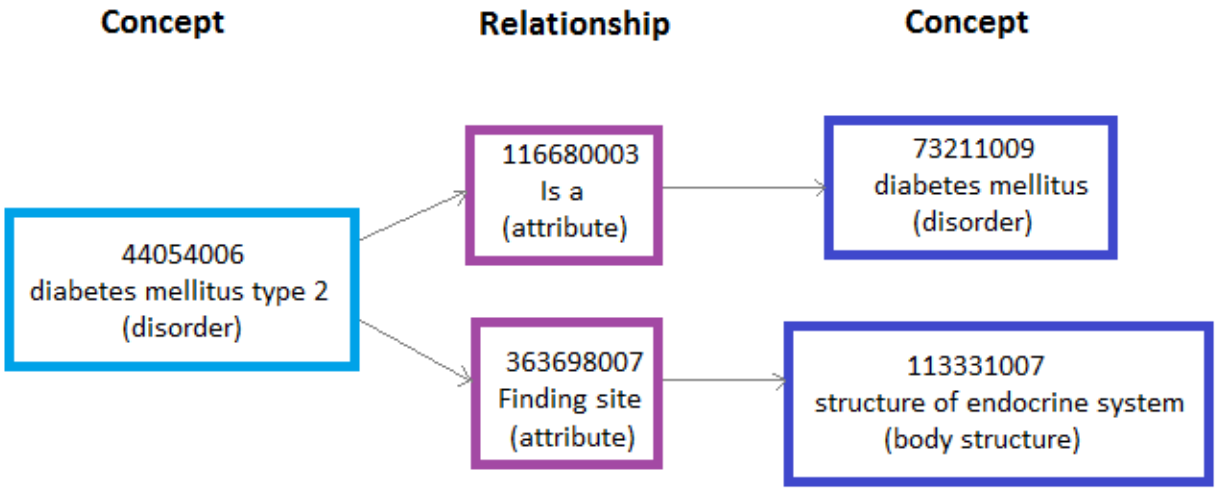

Figure 6: Snomed's relationships

Every concept has a set textual descriptions. This set contains one official description and can have several synonyms. This allows the users to select the term they prefer to represent the concept. The official term is unique, but several concepts can have the same synonym term. Interpretation of the synonym therefore depends on the official term.
Here are some examples of synonyms for "myocardial infarction (disorder)" (official term): heart attack, infarction of heart, cardiac infarction, myocardial infarct…

Each clinical term is associated with a numeric identifier, a code, to make it easily referenceable in any language and processable by computer. It aims to help the recording of medical data and improve patient care.
It can be mapped to other international terminologies and code systems such as ICD 9 (International Classification of Diseases) or LOINC (Logical Observation Identifiers Names and Codes). It is already used in 50 countries around the world including Belgium.

SNOMED contains an extensive list of codes and make it possible to encode really detailed information. Accurate concepts can be represented by a combination of codes.

Example: "*cut on the right forefinger caused by a knife*"

64572001 | disease |:
246075003 | causative agent | = 17472008 | Knife |
, 363698007 | finding site | = (83738005 | index finger structure | :
272741003 | laterality | = 24028007 | right |)
, { 116676008 | associated morphology | = 134352000 | Incised wound |
, 363698007 | finding site | = 39937001 | skin structure | }

I chose to use it rather than the ICD specification because of this possibility to describe almost everything with it. ICD is a lot more restricted.

(iii) KMEHR

It is an implementation of the recommendation of the Belgian Healthcare Telematics Commission, enabling the exchange of structured clinical information [15]. It is a Belgian standard for medical data exchange.

KMEHR consists of an XML message format that defines a grammar, a set of medical transactions and a set of reference tables of values that can be used within KMEHR message [15].

A KMEHR message is composed of a header and a non-empty set of folders. The header contains information about the sender and receiver of the message. A folder corresponds to a patient and contains a series of transactions. The transaction author must be a healthcare professional. A transaction must have an id, a type, a date and indicate whether it is completed and validated. They can also contains items that represents atomic medical information.

Example:

```
<transaction>
   <id S="ID-KMEHR" SV="1.0">1</id>
   <cd S="CD-TRANSACTION" SV="1.0"> treatmentsuspension</cd>
   <date>20016-04-21</date>
   <time>01:30:10</time>
   <author>
      <hcparty>
         <id S="ID-HCPARTY" SV="1.0">1235489213289</id>
         <cd S="CD-HCPARTY" SV="1.0"> persdietician</cd>
         <firstname>Jean</firstname>
         <familyname>Dupont</familyname>
      </hcparty>
   </author>
   <iscomplete>false</iscomplete>
   <isvalidated>true</isvalidated>
   <item></item>
</transaction>
```

(iv) Git

Git is a version control system almost unavoidable in software development. It manages the changes in the documents and helps us safely backup our work.

(v) MEAN.js

MEAN.js is a solution stack primarily used to build web applications. MEAN stands for “MongoDB, Express, AngularJs, Node.js” [16]. Each word in the acronym is a JavaScript software component.

- MongoDB is an open-source document database. A document is a data structure

composed of attribute and value pairs, similar to JSON objects [17].

- Express is a free and open-source web framework for Node.js [18].
- AngularJs is a JavaScript client-side framework that allows the use of HTML as template language [19].
- Node.js is an open-source JavaScript runtime system mostly used to build server-side applications [20].

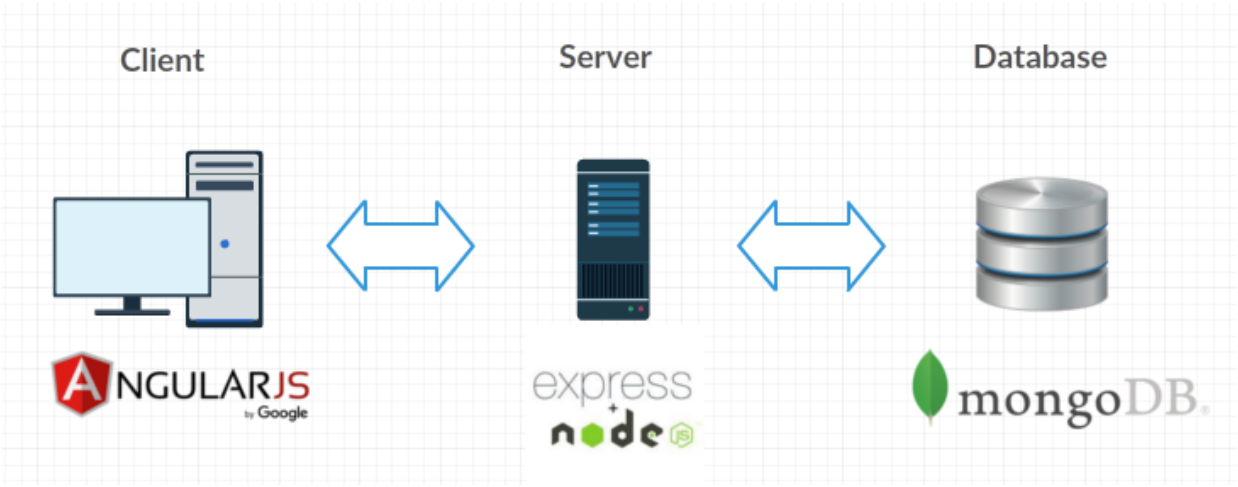

Figure 7: MEAN.js architecture

I chose this solution because my project needed to be included into the Eglé project and Eglé already used MEAN.js. So for easy maintenance and collaboration I chose not to introduce a new technology.

(vi) Node Webkit or NW.js

Node.js being a server-side system, I had to find something more if I wanted to keep it for the desktop application part of my project. The solution I found was Node Webkit (now called NW.js).

Node Webkit is an app runtime based on Chromium and node.js [21]. It makes it possible to build desktop applications in HTML and JavaScript, using all the Node.js modules.

(vii) Nedb

Nedb is javascript embedded persistent or in memory database for nw.js. The API is a subset of mongoDB's API [22]. It allows all the most common operations.

I choose to use it for the desktop application because I needed something lighter than the full mongoDB but I didn't want to use something entirely different that would require to learn and maintain another type of database.

(viii) JWT

The authentication is made using JSON Web Token (JWT). JWT is an IETF standard described in the RFC 7519 [23].

JWT is method for securely transmitting information between parties. The data transferred is in a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted [23].

(ix) Bootstrap

Bootstrap is a free and open-source front-end HTML, CSS and JavaScript framework [24]. It follows the responsive web design approach that allows web pages to adapt to the size of the screen one is viewing it with.

It is classed second in the list of projects with the most stars on github [25].

(x) Eclipse

Eclipse is the IDE (Integrated Development Environment) I prefer. It is primarily a Java IDE, but there's a plug-in for JavaScript and Node.js in particular: Nodeclipse. There's also a plug-in for git.
I find it really comfortable to use. And it is free and open-source.

(xi) Source code of Eglé

Available on the UCLouvain github (https://github.com/uclouvain/egle), this is the code I used to set up my small EHR network.

(xii) Consultation with a general practitioner and a diabetologist

I have had the occasion to discuss with a general practitioner and a diabetologist about how they do their consultation, what software they use to record their EHR, what are the advantages and drawback and what other functionalities they would like their application to have.

Drawbacks:

- Hard to share the EHR with other practitioners or hospitals.
- Complex to use.

Other functionalities:

- An easy way to share at least the essential information.

- A way for the patient to share his personal and family history.
- A way for the patient to share his risk factors.
- An alert system that let the practitioner know when one of his patient variables are critical.

4.2. Architecture

The architecture had to allow three kinds of data transfer:

- practitioner to practitioner
- practitioner to patient
- patient to practitioner

It also had to take into account that the desktop application could be used offline. And each user would like to control his own data:

- who can see it
- gives or revokes access to it
- who can edit it and which part can they edit

To follow all those requirements, I chose to let each user make his own HER(s). Sharing would allow them to see other users' versions of their EHR and copy the data relevant for them into their own EHR. That way, only the original user can actually edit his EHR and that removes the risk of non-wanted edition or deletion.
I lose the centralized EHR advantage of "only one version always up to date" but I get more security in exchange and the "Who is responsible for what part?" question that is quite important legally has a clear answer. Each person is responsible for the EHR they create.

4.2.a. Global view

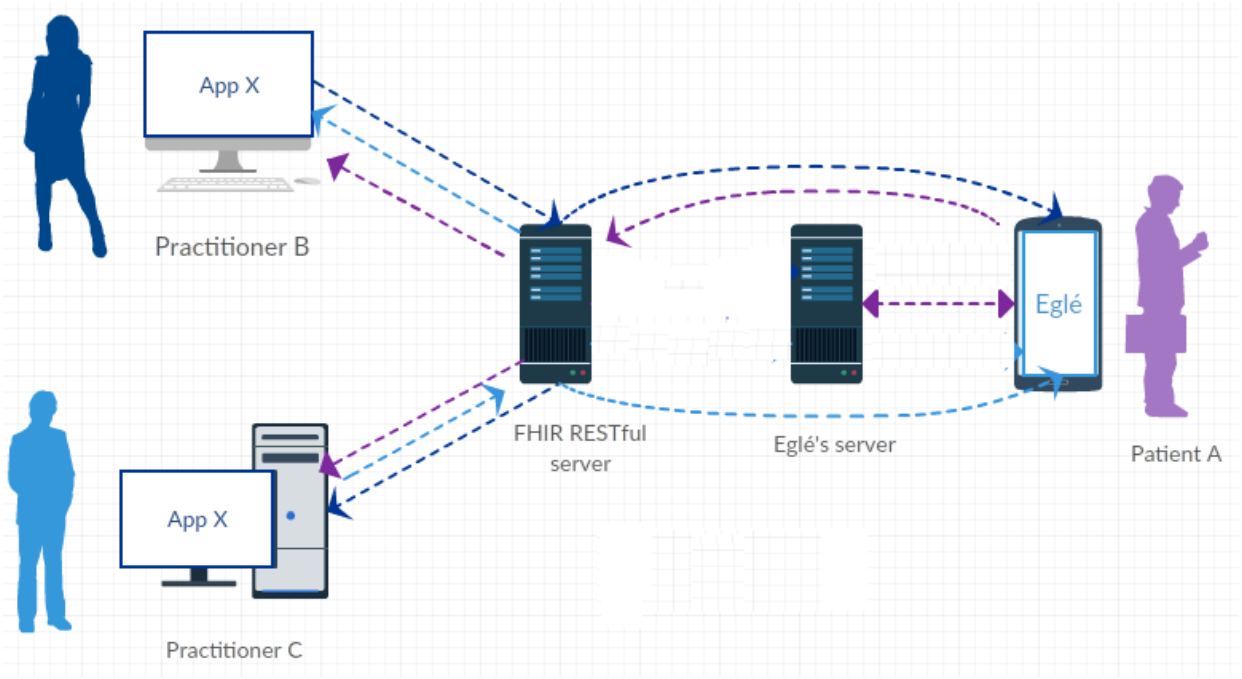

Figure 8: Global view of the data exchanges

The desktop application (named "App X" for now) can be used by itself and offline. To share data

with other people, the practitioner will have to first register on the server.

The patient will also need to register on the server via the Eglé application in order to receive data from his practitioner and to send him the information he asked for (see Figure 9: Sequence Diagram "Register to the server").

Users can choose to share a health record; all the record or just a piece of it. The application (Eglé or App x) sends the record to the server. Then the user can share it with someone by giving them verbally the code associated with this record.

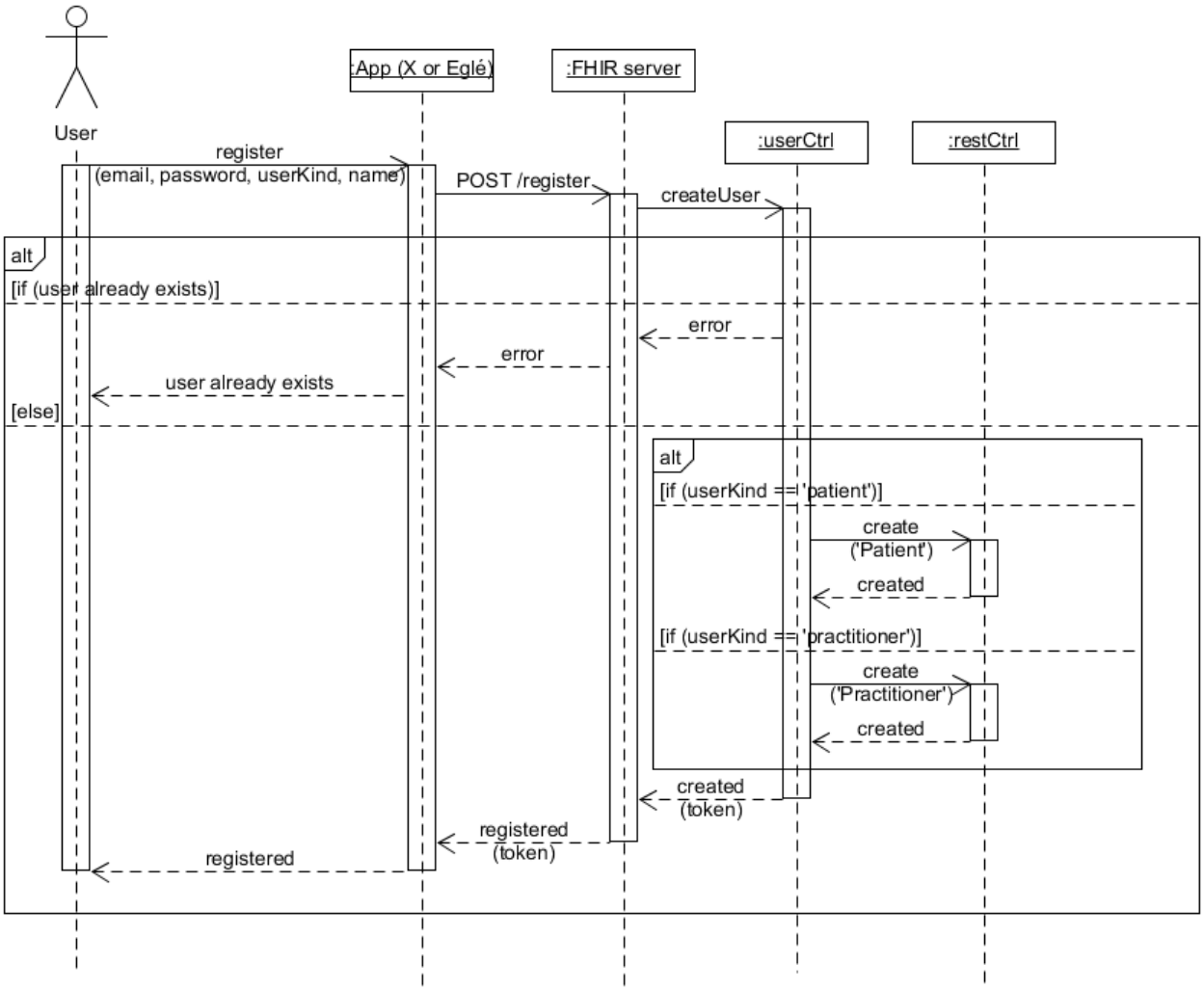

Figure 9: Sequence Diagram "Register to the server"

4.2.b. Practitioner side

With the desktop application, the practitioner should be able to:

- Create, read, update and delete a patient record (see Figure 12).
- Add (and delete) an analysis result (HbA1c or BNP) to the record.
- Write the personal history of the patient.
- Write the conditions the patient is suffering from.
- Create an account and login to the FHIR server (see Figure 9).
- See the code needed to share the record.
- Mark and unmark every piece of information of a record as “to be shared” (see Figure 11).
- Revoke the right of someone to see his record.
- Enter the name of a patient and a code (and the name of the practitioner if the record was not made by the patient) to see another record.
- See the list of records he received access to.
- Copy data from another related record into his record.
- Renounce the right to see a record.

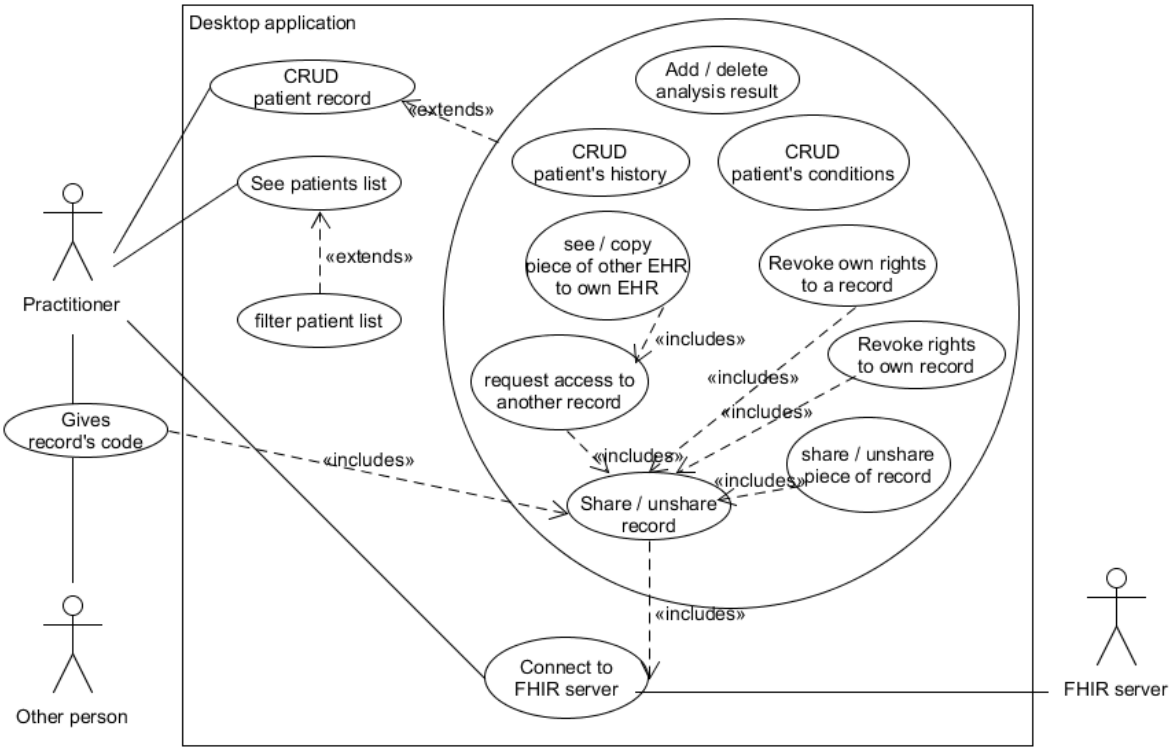

Figure 10: Practitioner Use Case Diagram

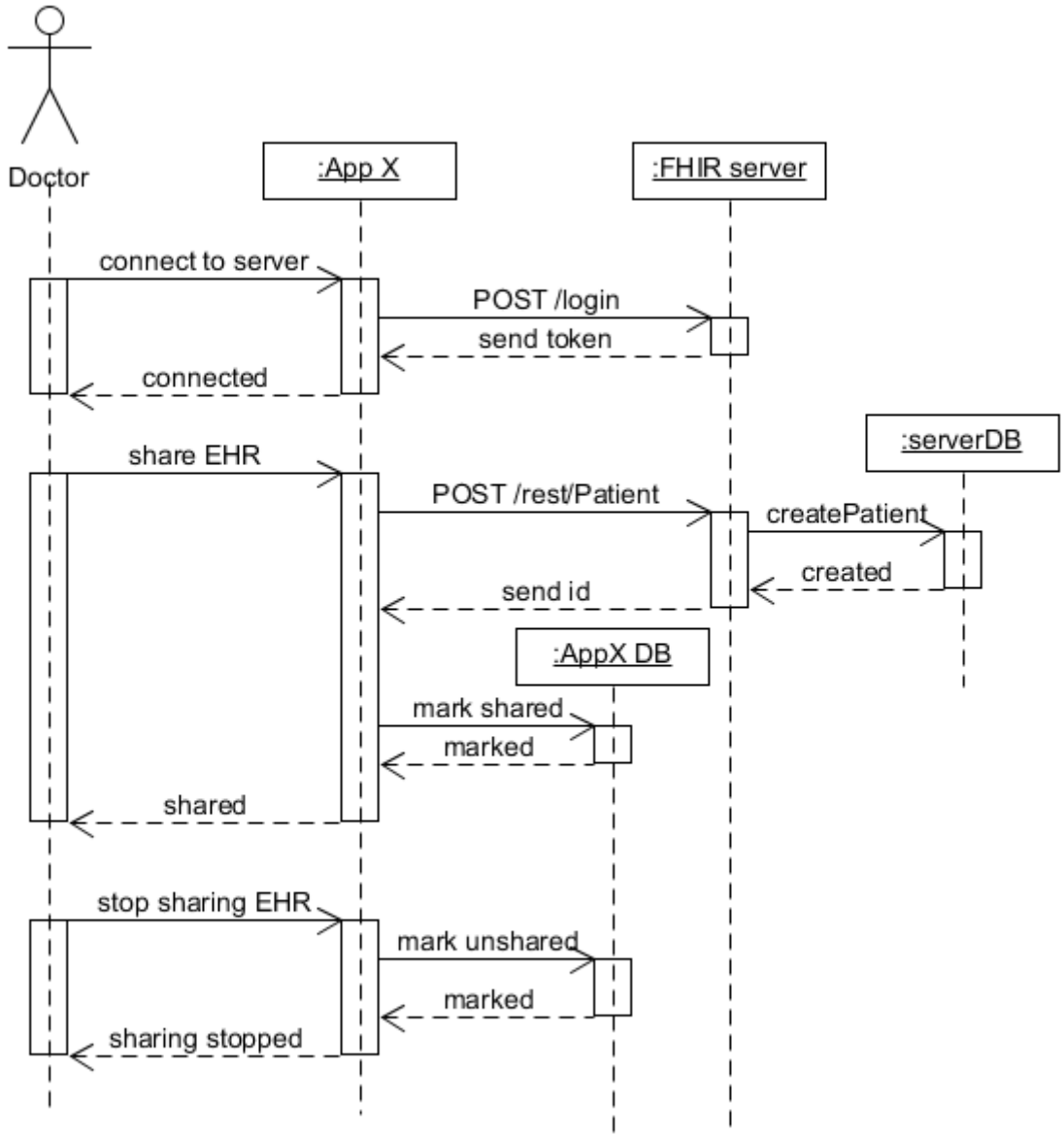

Figure 11: Sequence diagram "doctor shares and unshares a record'"

When the practitioner unshares a record, the record is not deleted from the FHIR server. Unsharing only stop the application from sharing future modifications to the record.

If the practitioner wants to make the record totally unavailable, he will have to revoke the rights of all the people he shared it with in the dedicated section of the application or delete the record from the server.

It is to be noted however that if the people he shared the record with chose to copy some of the data. They will still be able to see the copied data.

I made that choice to emulate the rights and control practitioners already have with their

paper records. Paper records can be exchanged with another practitioner. The other practitioner can make a copy. The original practitioner can chose to stop sharing the evolution of his patient with him when it is no longer relevant to his speciality. But the other practitioner still keeps his copy and would not be happy if data suddenly goes missing in one of his record, even if it was data provided by another practitioner.

My solution would keep the same rights and control but would facilitate the exchange of data. I only replace the physical stack of paper by a code to share verbally with someone else. The practitioner has the responsibility to choose wisely to whom he shares the code with.

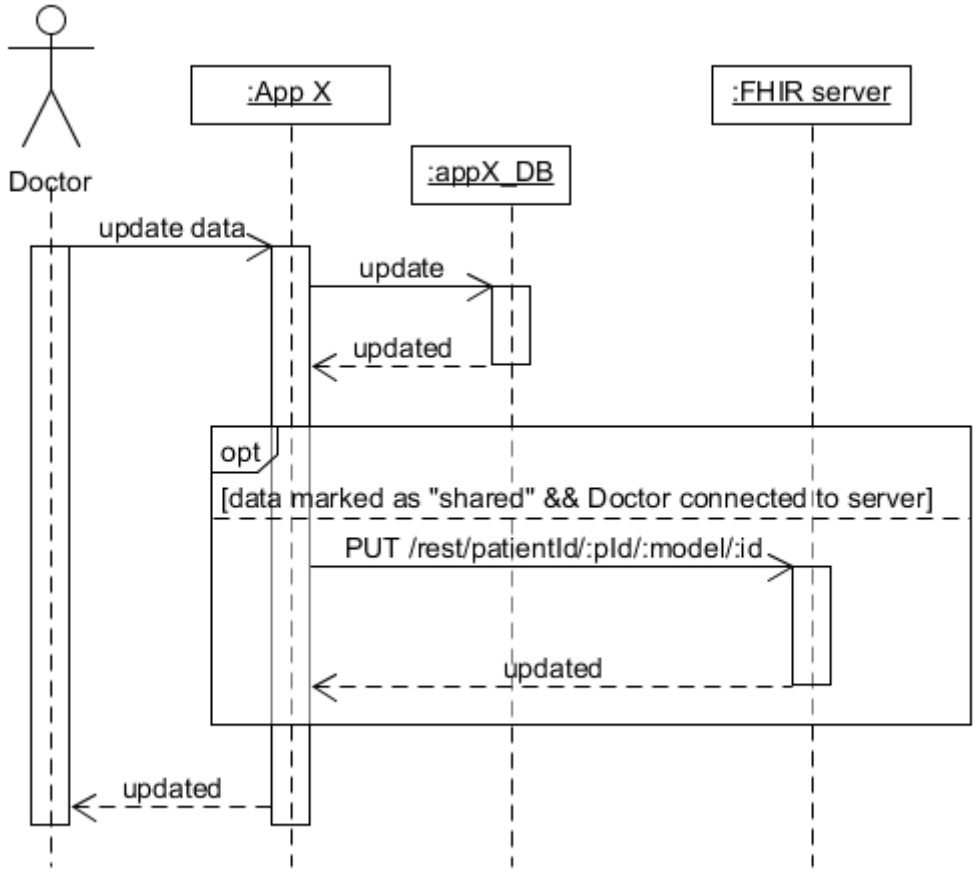

Figure 12: Sequence diagram "doctor updates record"

When a practitioner updates data in a record, it is only sent to the server if the practitioner is online and connected to the server and has chosen to share this particular piece of data.

If the practitioner is offline, the updates are not sent to the server, but the application keeps track of it by indicating when each piece of data was last shared and when it was last updated. When the system sees that a piece of the record has been updated more recently than it has been shared, it will show a “synchronize” button next to id that indicates to the

user that the version on the server is not up to date and offers the option to update it.

4.2.c. Practitioner –> practitioner exchange

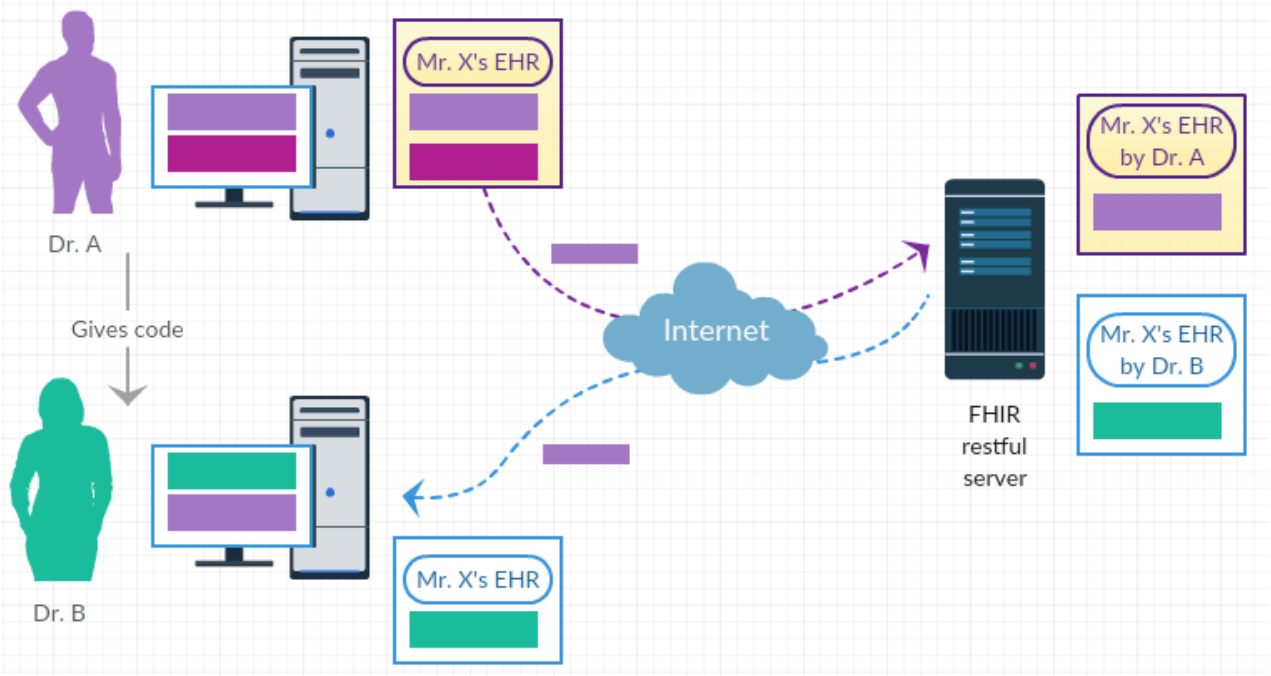

Figure 13: Data exchange between practitioners

The important questions to answer here were: “Where goes the data, how many copies of it are there and who has what rights on those copies?”

This is my answer:

When Dr A wants to share a piece of a record he made with Dr B, he first has to mark that piece has “to be shared” in the application. The application will sent the data to the server. Then Dr A will give Dr B the code for the record. Dr B will connect himself to the server via the application then enter the code, the name of the patient and the name of Dr A. The server will sent to Dr B all the data for this record that Dr A has marked as “to be shared”.
Dr B will be able to see the data as long as he is connected to the server. If he wants to see the data offline, he will have to copy it into his own corresponding record. Once the data’s copy is in Dr B’s record, he is responsible for it. He can edit it, he can share it with someone else. He can also still see the original data made by Dr A. He cannot edit in anyway the original data but he can edit the copy he made.

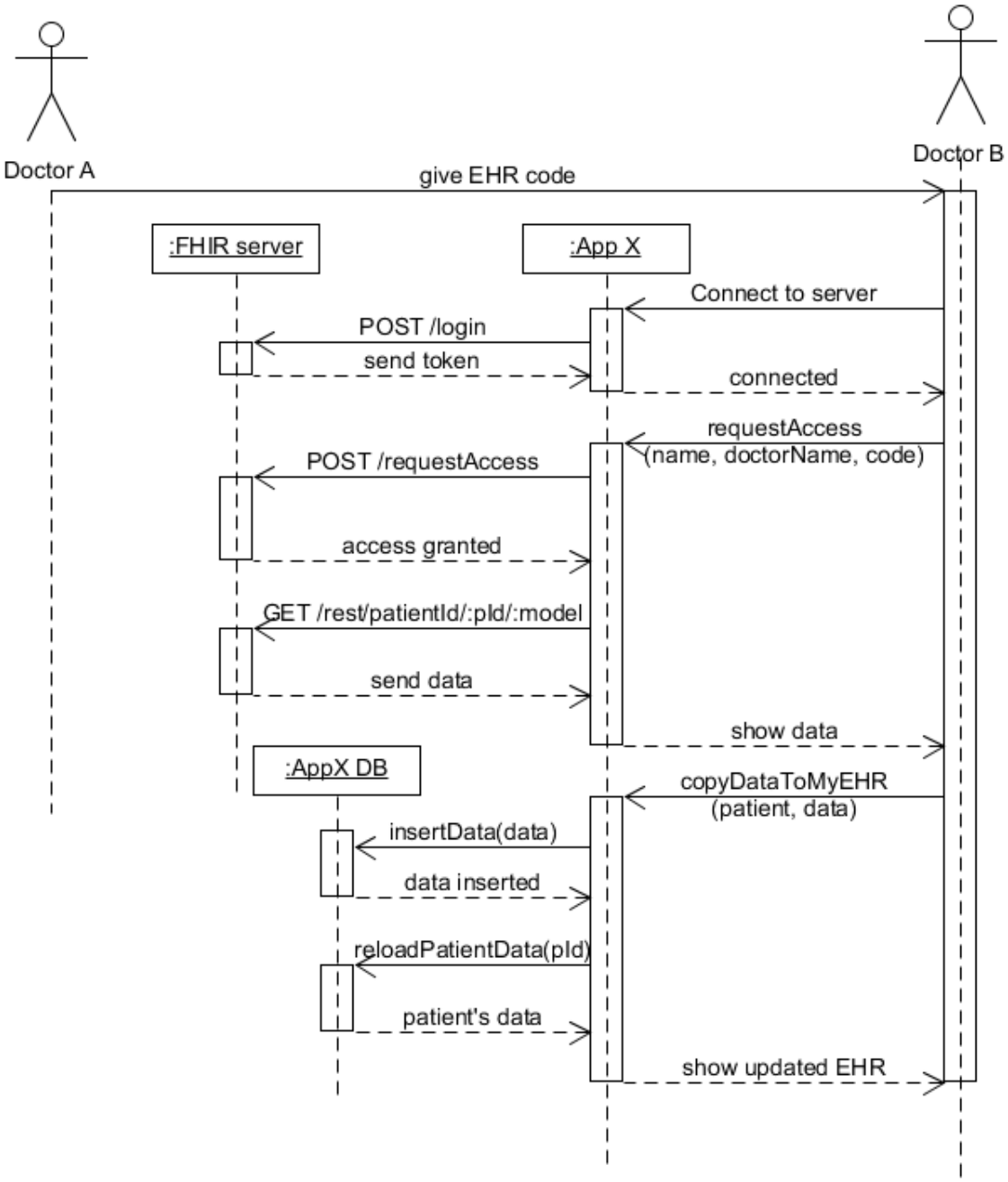

Figure 14: Sequence diagram "copy data from other EHR"

4.2.d. Patient side

Via the Eglé application, the patient will be able to register on the server and start making his own small EHR with the data often asked by practitioners such as personal history and current conditions. He will also be able to make a list of the analysis results he received. Either by entering himself the results or by copying them from the EHR his doctor shared with him.

Since Eglé is a website, the EHR will be directly saved on the server and there's no need to manage an offline mode. The record will only be visible and editable when the patient is connected to the FHIR server.
As a result, there's not part of the record that can be kept only on the patient's machine like with the practitioner application. Everything the patient put into his record is considered as "to be shared" and can be seen by all the people the patient gives his record's code to. There's no granularity provided for the patient side.

The patient as the option to see the record his doctor chose to share with him by giving him the code. The patient also has the option to revoke the right to see his record he previously gave to his doctor as well as revoking his own right to see one of his doctor's record.

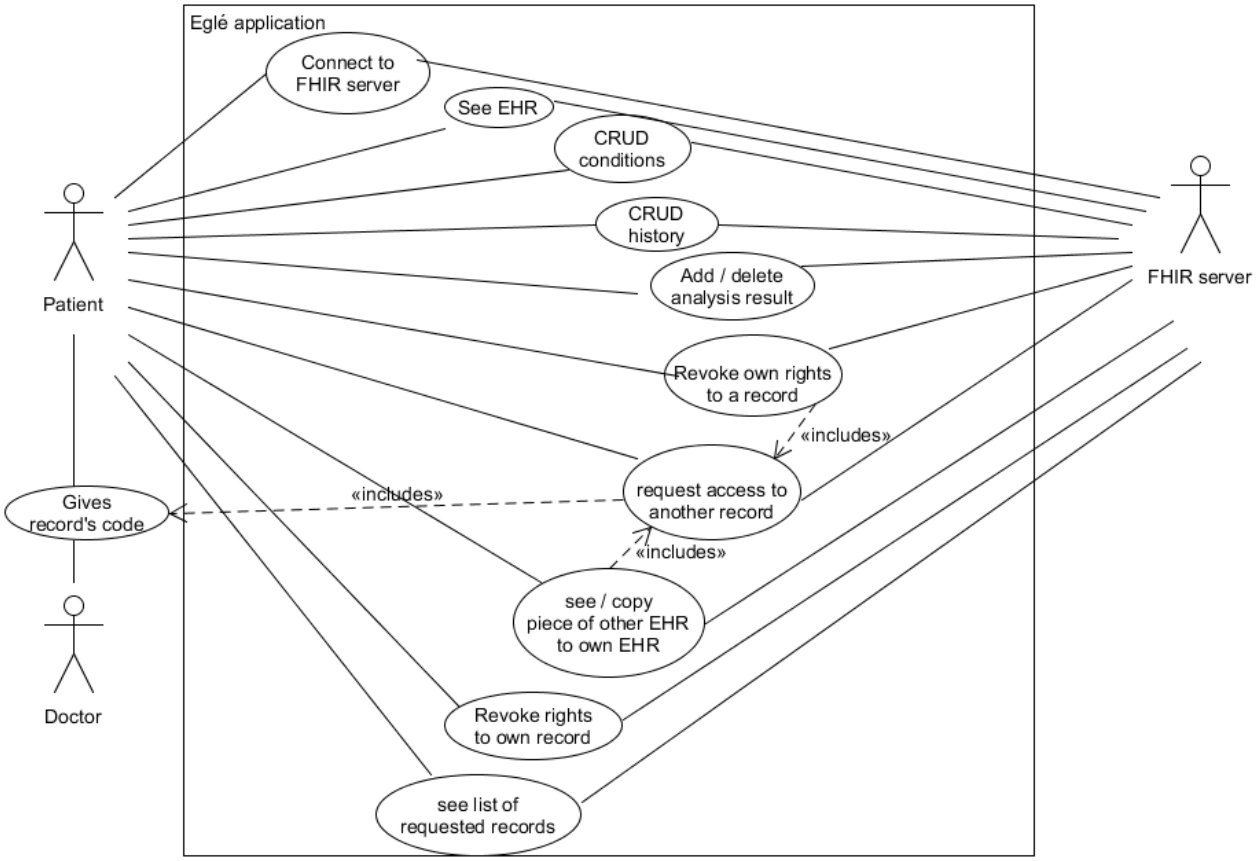

Figure 15: Use case diagram "Patient's side"

4.2.e. Practitioner -> patient exchange

To share his EHR, the practitioner verbally gives the patient the code for his record. Then the patient can connect to the server via Eglé and request access to the EHR by giving the code, his name and the name of the practitioner that made the EHR.
Eglé will then ask the FHIR server to send it the record and will display it for the patient.

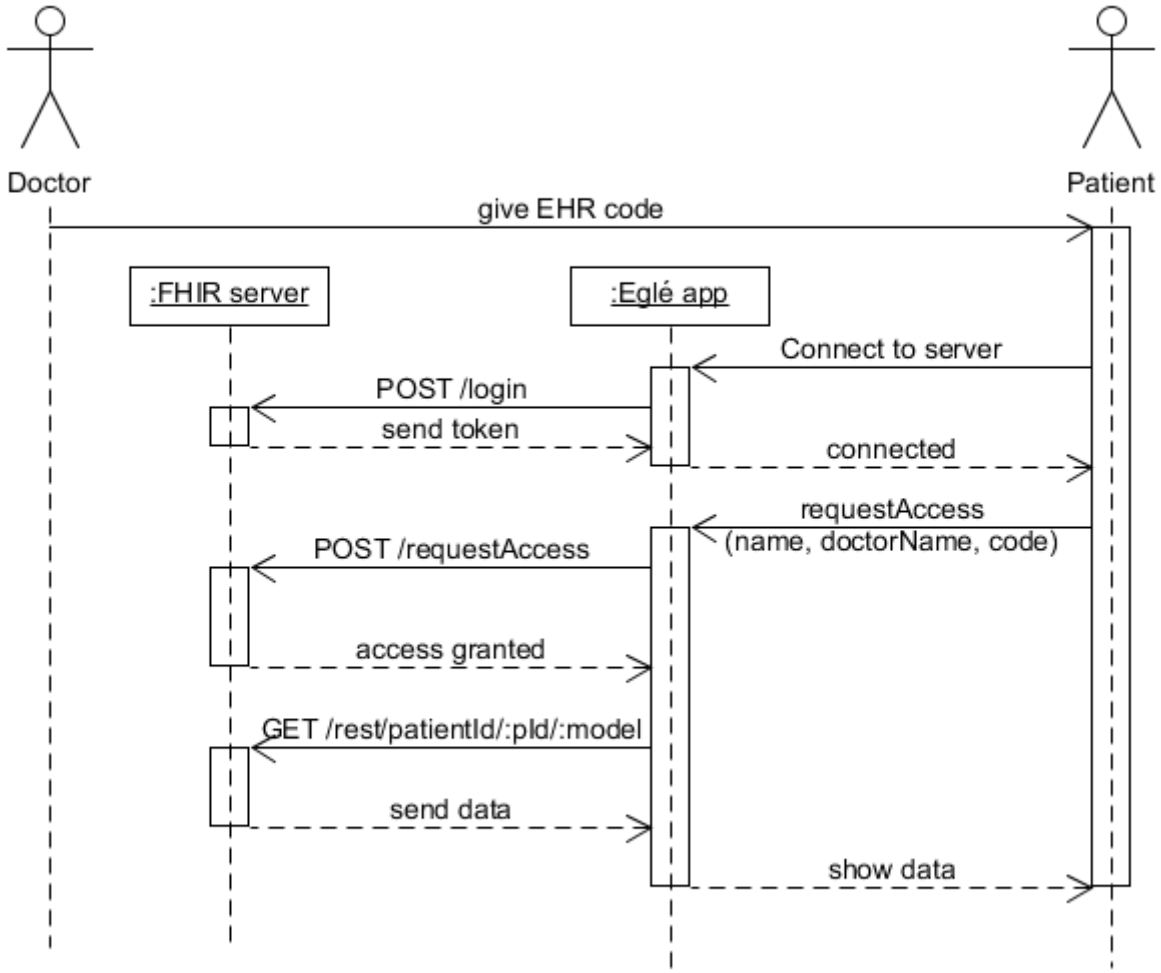

Figure 16: Sequence diagram "Practitioner -> patient exchange"

The patient has the option to copy the data from his doctor into his personal record. Then he will see both the original and the copy. The copy can be edited or shared with another practitioner.

I chose not to automatically copy the data sent by the practitioner in order to help the patient realise that he is responsible for the copy he made and that he is the one that controls it. Once again I chose to sacrifice a bit of usability for clearer rights and duties.

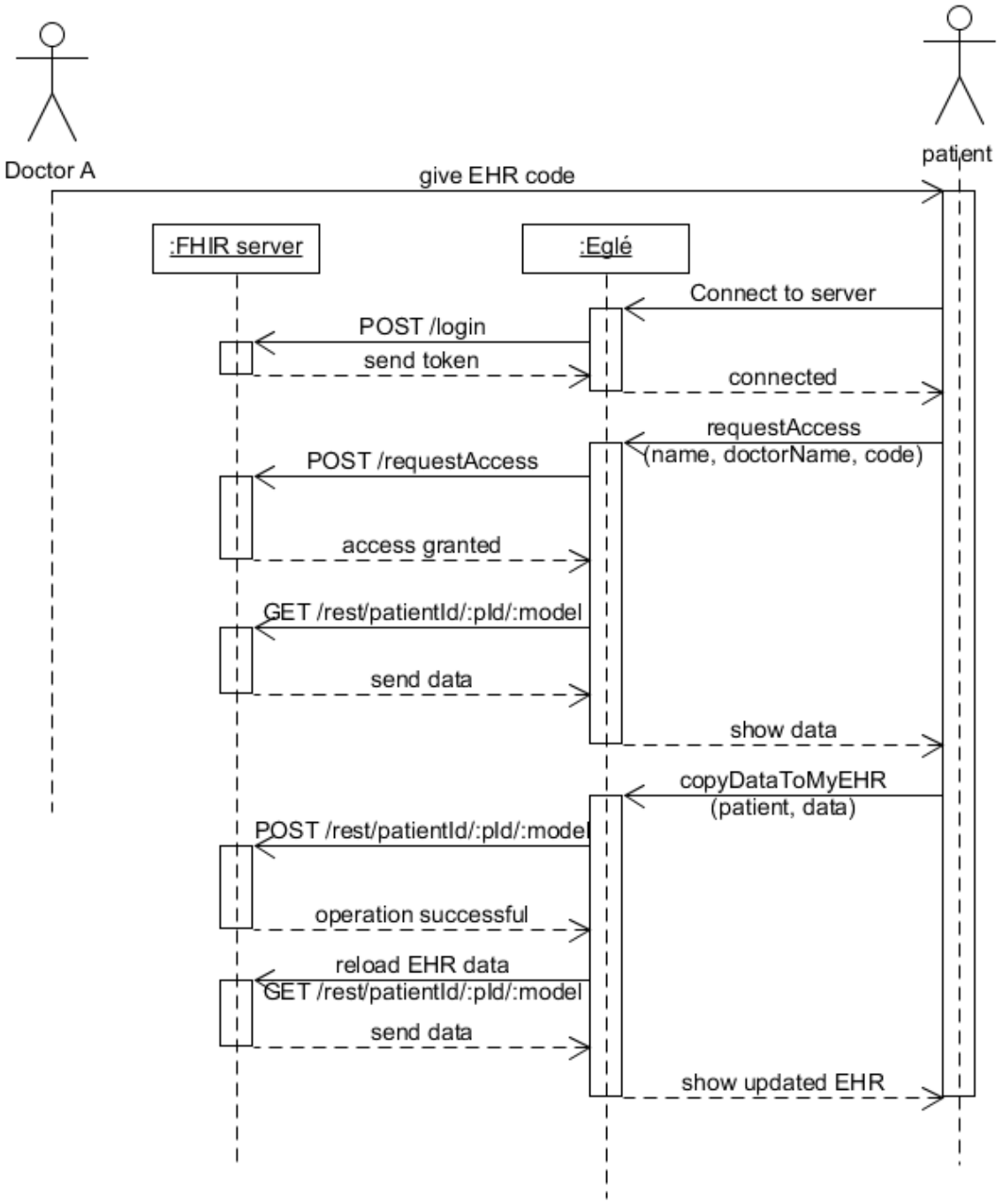

Figure 17: Sequence diagram "patient copy data to own EHR"

4.2.f. Patient -> practitioner exchange

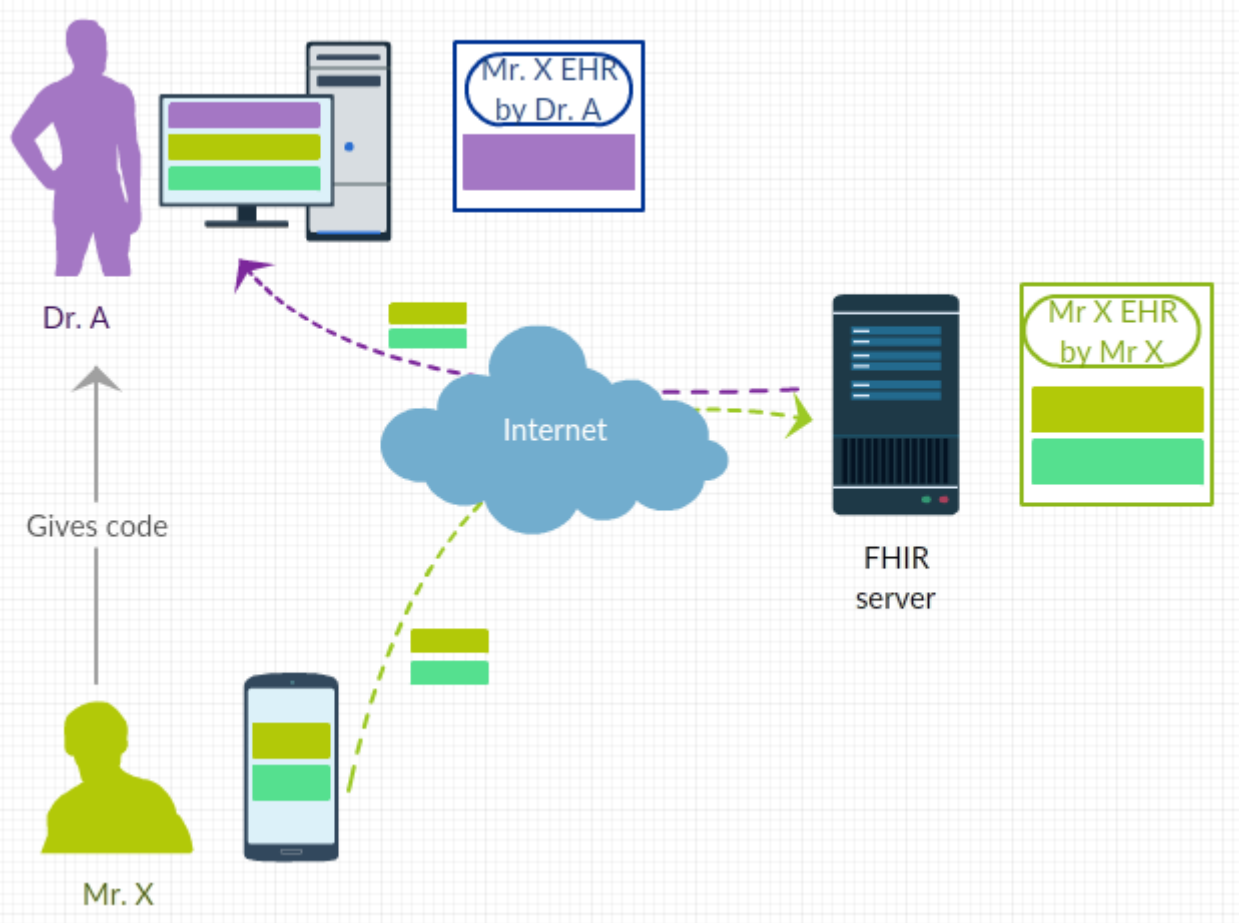

Figure 18: Data exchange from patient to practitioner

This is the same process for the exchange from patient to practitioner. The patient has created his own record with data he entered (conditions, history…) and / or data he copied from the record of one of his other doctors. The record has a code assigned to it. The patient gives his new doctor the code for his EHR. His doctor enters the code and the patient's name into the application and if he is connected to the server, the server sends him the data.
The doctor then has the option to copy the data the patient sent him into his own record.

5. Results

5.1. RESTful server

5.1.a. Interactions

Here are the requirement of the FHIR standard:

- **Create**: POST https://server/path/[resourceType]
- **Read**: GET https://server/path/[resourceType]/[id]
- **Update**: PUT https://server/path/{resourceType}/[id]
- **Delete**: DELETE https://server/path/[resourceType]/[id]
- **Search**: GET https://server/path/[resourceType]?[search parameters]
- **History**: GET https://server/path/[resourceType]/[id]/_history
- **History**: GET https://server/path/[resourceType]/[id]/_history/[vid]

I deviated a bit from the standard in order to facilitate the use of access control lists and more easily manage the permissions (see 5.4.b. for more details). For each resource, the id of the "patient" resource it refers to must be indicated in the URL of the request.

I use the id of the "patient" resource as id of the whole record. Permissions are granted to the record as a whole and not for each resources. So each time we want to edit the record by adding a resource to it or deleting or updating one of the resources it is composed of, we need to refer to it and the server will check if we have the right permission.

Here is my implementation:

- **Create a "patient" resource**: POST https://server/path /rest/Patient
  I had to make a route specific for the creation of "patient" resources since the id of the "patient" resource obviously cannot be known before the resource is created.
- **Create :** POST https://server/path/rest/patientId/{patientId}/{resourceType}
- **Read**: GET https://server/path/rest/patientId/{patientId}/{resourceType}/{id}
- **Update**: PUT https://server/path /rest/patientId/{patientId}/{resourceType}/{id}
  Updates need to be made one at a time. I did not implement a "batch updates" option.
- **Delete**: DELETE https://server/path rest/patientId/{patientId}/{resourceType}/{id}
  Same as for updates, no "batch deletes" option.
- **Search**: GET https://server/path/rest/patientId/{patientId}/{resourceType}?[search parameters]
  If no search parameters are indicated, the server will return all the resources that match the resource type indicated and that refer to the correct patient.
- **History**: GET https://server/path/rest/patientId/{pId}/{model}/{id}/_history

Each time a resource is modified, the server actually create a new resource with the modification and keep the previous unmodified one in the database. It is therefore possible to see all the previous versions of a particular resource.

- **History**: GET https://server/path/rest/patientId/{pId}/{resourceType}/{id}/_history/{vid}
  This command return one version (indicated by {vid}) of one resource (indicated by {id}).

5.1.b. Models used

The FHIR resources I used to build the EHR are:

- **Patient:** refers to any individual receiving the care of a practitioner or registered via Eglé.
- **Practitioner:** refers to a person providing healthcare.
- **Condition**: can contain information about an illness, problem, concern, issue or symptom.
- **Observation:** measurements or assertion about a patient (ex: blood pressure, weight, tobacco use…). It is used to record risk factors, blood tests results...
- **Encounter**: records an interaction between a patient and a practitioner.
- **DiagnosticReport**: contains the results and interpretations of diagnostic tests performed on the patient (ex: HbA1c blood test result performed on a diabetic patient).
- **DiagnosticOrder**: records the request for a diagnostic test to be performed. I use it to record the tests the practitioner ask for his patient (ex: HbA1c test, BNP test…).
- **MedicationOrder**: records an order for supply and instructions for the administration of a medication to a patient.

All these resources contains common reusable patterns: the FHIR data types. I chose to implement it using the subdocument option of mongoose rather than nested schemas. The reason is simple: FHIR is still an evolving standard. And it evolves fast. The definition of those subdocuments already changed while I was working on this project. With subdocuments, only one file need to be modified to make the schemas up to date with the FHIR standard.

All those schema where made from the specification found on the FHIR website. The comments explaining the use of each attribute where also found on the website [26].

Other (not from FHIR):

- **Resource_history**: used to keep track of all the different versions of a resource.
- **User**: contains the login, password, language… of the persons that register on the server. A user can be a patient or a practitioner.

(i) Constraints

FHIR contains a lot of models that can provide for every situation you can imagine. It's therefore possible to encode the same information in a lot of different ways.

To build a medical record, it's necessary to restrict those different ways in order to know where to look to get the information.

Here are the restrictions imposed:

- Each of the patient's illnesses need to be recorded in a “Condition” object.
- The code systems used are restricted to SNOMED CT.
- The “identifier” attribute of each resource must contains the code of the record it is associated with and the name and id of the record’s owner. If there’s no code nor id, the application won’t be able to see who has permission to access the resource and will simply reject all requests.
- If the resource has been copied from another record, the identifier list must also contains the identifier from the previous owner. The identifier of the current owner must always be the first in the list. This is so we can track the origin of each resource and help users decide how much credibility they can accord to the information contained in the resource.

Example:

```
Identifier : [{
	value: “x5y6z7”,
	assigner: {
		reference: “Practitioner/0123fs545sf845sf89d”, // id of the “patient” or “practitioner” resource referring to the owner on the FHIR server
		display: “Jean Dupont”
	}
}]
```

(ii) Additions

Thought the FHIR specification planned for a wide range of uses, it stays a bit generic and needs to be customized for a specific use.

Here are the small things I added:

- A “risk-factor” category for the “Observation” resource. FHIR put the risk factors into the “social-history” category, but for this project, I needed to differentiate the risk factors from the rest of the history.
- A “profession” attribute for the “Patient” resource. The diabetologist I consulted explained that he liked to know what his patients did for a living since it could influence how much exercise they did.

5.2. Small desktop EHR application

5.2.a. Overview

The desktop application main panel consists of the patient list with a filter option. Clicking on a patient in the list will make the details of his record appear. The parameters option in the menu lead to a panel where there is the form to connect to the EHR server.

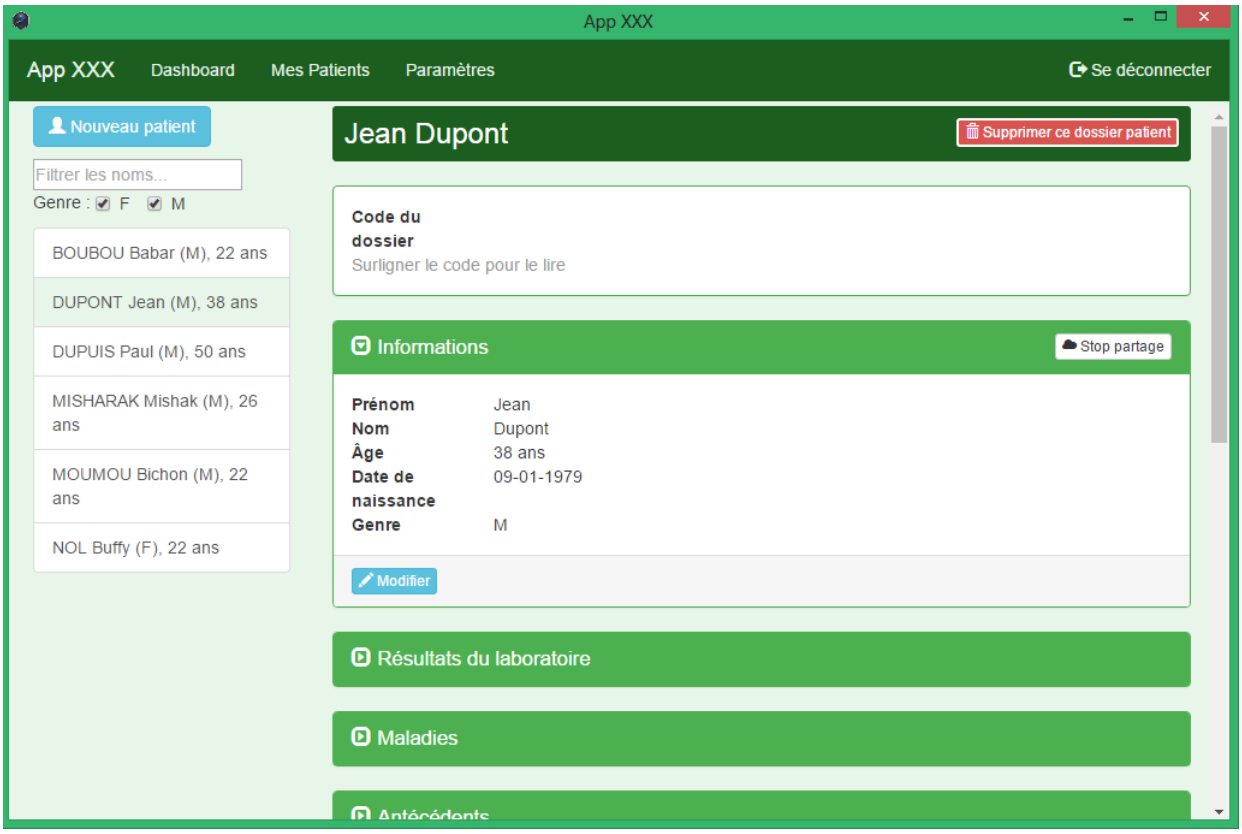

Figure 19: Desktop application overview

A patient record is created by entering the family name, given name, gender and age or birthdate of the patient. Entering an age automatically adjust the birthdate and entering the birthdate automatically adjust the age.

The record consists of four information panel. The first one contains the basic information of the patient: his name, age, gender. The second one is for the lab results. Only three types of lab results are possible at this time: HbA1c, BNP and NT-proBNP. The first concerns diabetic patients and the last two concern patients with heart failure.

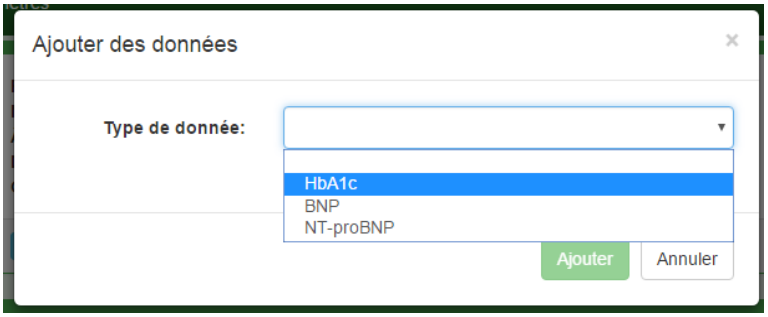

Figure 20: Desktop application "adding lab result"

The third panel is for the current conditions of the patient and the last one for his history.

I chose to put the form for adding or updating data into modals that pops up when we click on the “add’ button. I found it to be clearer than navigating between different screens.

5.2.b. Sharing

The code to share the EHR is the first panel in the record. It appears after the user chose to click on the “share” button of the record for the first time.

Figure 21: Desktop share button

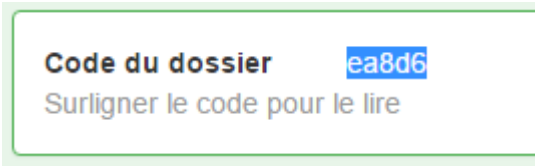

Figure 22: Desktop record's code

Once the sharing has started, the user get the option to separately share each piece of the record. Sharing can be cancelled with the “unshare” button that appears after we have clicked on the “share” button. And it can be restarted with the “restart” button. There also a “synchronize” button that appears when some data marked as “shared” has been updated but the update could not be sent to the server.
All these different buttons are there to give the user a clear idea of what state his record is currently in and to help him correctly manage the sharing of his records.

Figure 24: Desktop "restart sharing" button

Figure 23: Desktop "synchronize" & "unshare" buttons

Below the record, there’s a panel containing the form to fill to request access to another record for the current patient.

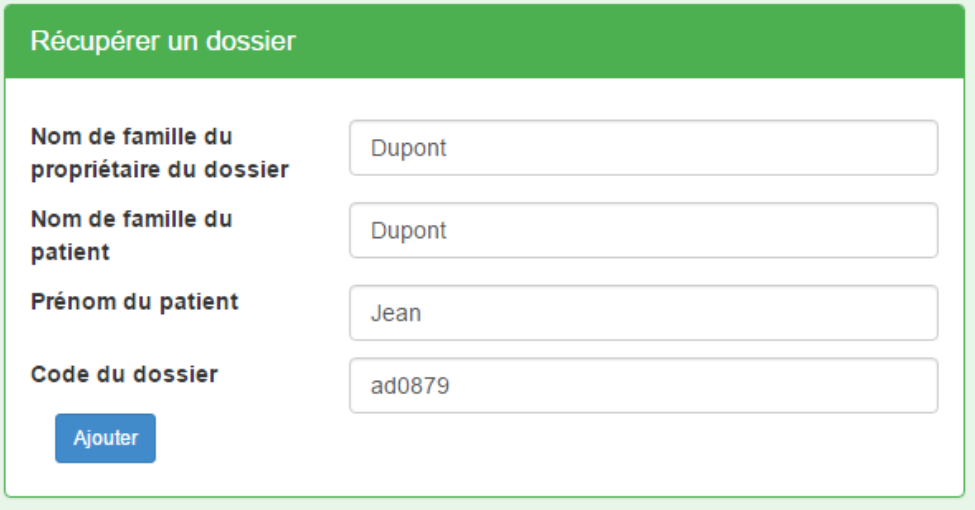

Figure 25: Desktop "add record" panel

If the request is valid, the new record will be visible in the list of records linked to this patient just below the panel. Via this list, the user can choose to revoke his right to see the record when he doesn't need it anymore.
And just below this panel, there's the option to revoke the right we gave someone else to see the record of the patient.

Once another record has been linked to the patient main record, we can see all the information it contains (or at least the information the owner chose to share) in the corresponding tabs of each information panel. In those tabs, the user also has the option to copy the date into his own EHR. And if the user also shared his record with the person, he may be able to see in the other record some information he wrote if the other person chose to copy it into it.

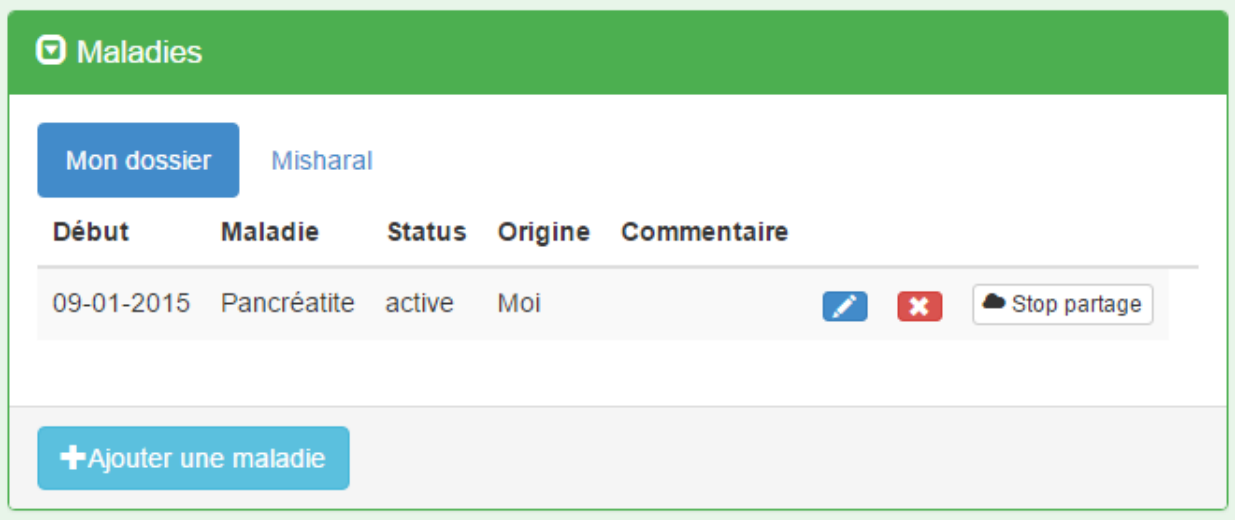

Figure 26: Desktop conditions panel, own EHR tab

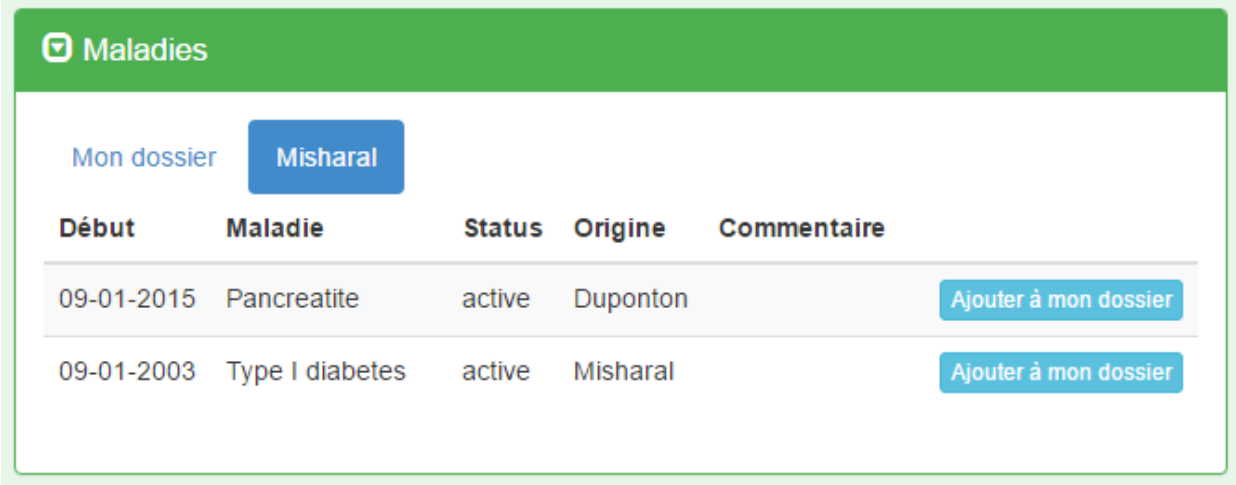

Figure 27: Desktop conditions panel, another EHR tab

5.3. Eglé widgets

The form to register and login to the EHR server is accessible via the parameters of the application. Once the user is logged in, he can see his own record's code.

Figure 28: Eglé's connection to EHR server widget

The medical record is dissociated from all the widgets on the dashboard and can be accessed via a link in the menu that is not visible if the patient is not connected to the server. This was done in order to raise the patient awareness on exactly which data is official (as in "coming from his doctor") and which was produced by himself or the Eglé application.

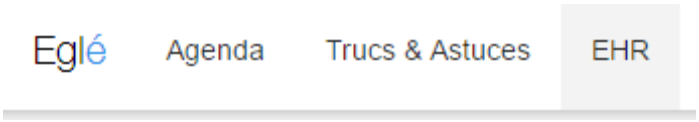

Figure 29: Eglé's EHR menu

5.3.a. Information received from the practitioner

On the EHR, the user can see and manage the list off all the record his doctors shared with him. He can add a new one or choose to revoke his rights to see one.

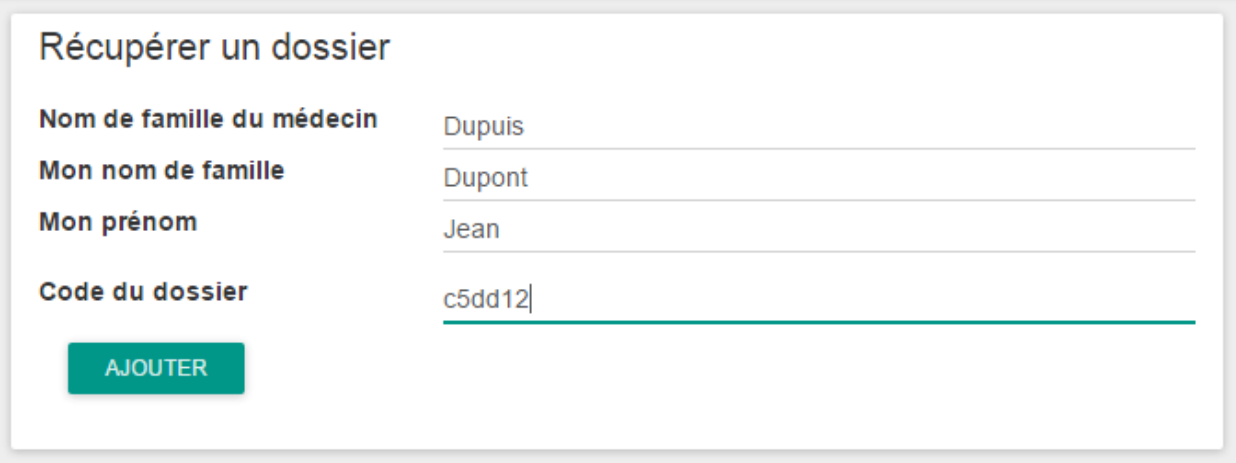

Figure 30: Eglé's "add a record" widget

Figure 31: Eglé's "My EHRs list" widget

Once the patient is connected to the EHR server and has entered the code his doctor gave him, he can see what the doctor chose to share.

Below the list of available records, comes the widgets containing medical data: history, analysis result and conditions. Each widget is fractioned according to the sources of the data. The first tab contains the data inside the EHR made by the patient. He can create a new element or edit a previous one. In the other tabs, there is the versions made by his doctors. The user can compare the different versions and update his own EHR accordingly. Beside each element of the data provided by the doctors, there's a button allowing the patient to copy and add this piece of data to his self-made EHR.

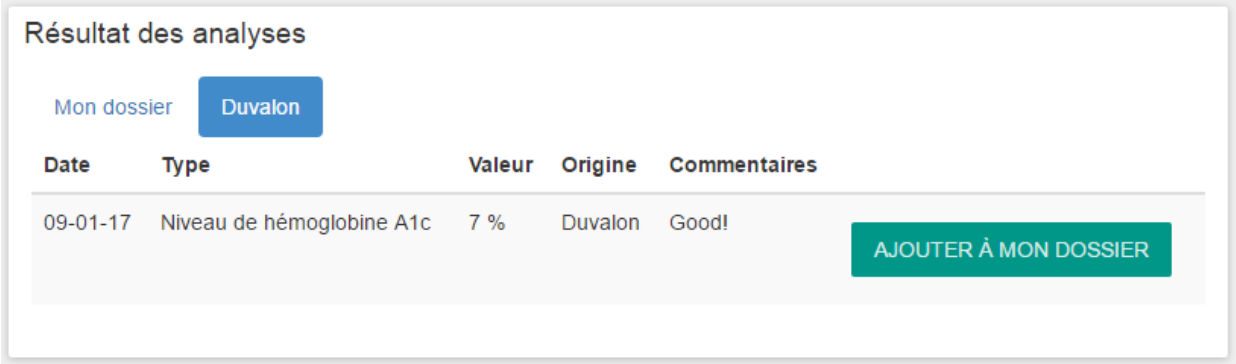

Figure 32: Eglé's "Lab results" widget

The server will keep in memory the origin of the copy. This is done in order for the doctor with whom the patient may choose to share is EHR to know how to interpret each piece of data and the level of credibility he can accord it. Patients don't always know the correct medical term for their condition or symptom and sometimes use a wrong one. For example, some people don't like to use the word "diarrhoea" and prefer to say they have the dysentery. This is really not the same thing. So if a doctor reads the EHR and see "Dysentery" in the patient history, he can have some doubts if the information originated from the patient but if he sees it comes from one of his colleague he will be able to trust it without further inquiry.

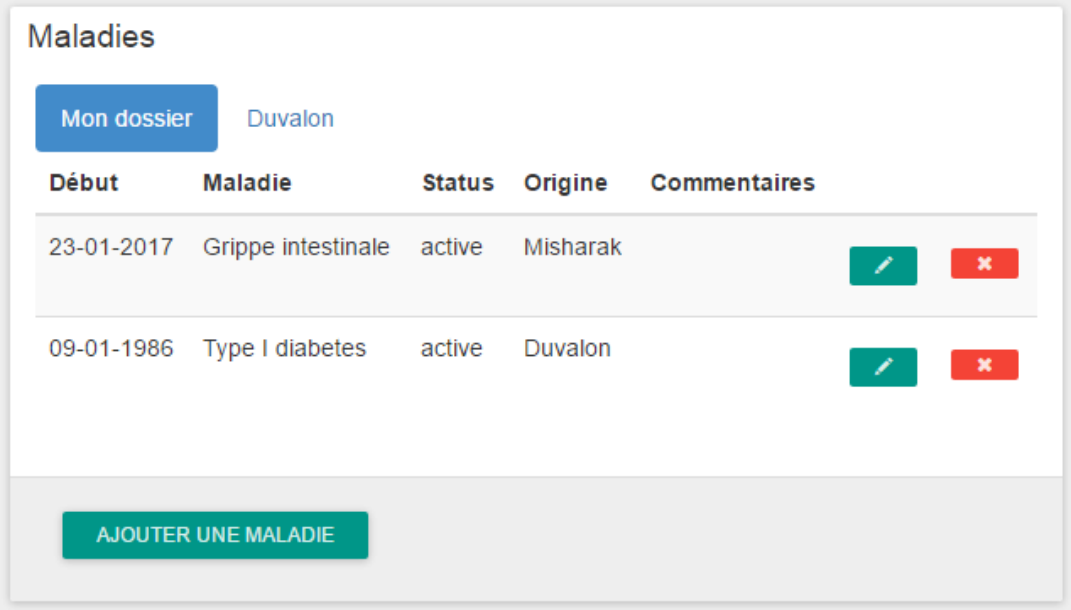

Figure 33: Eglé's "Conditions" widget

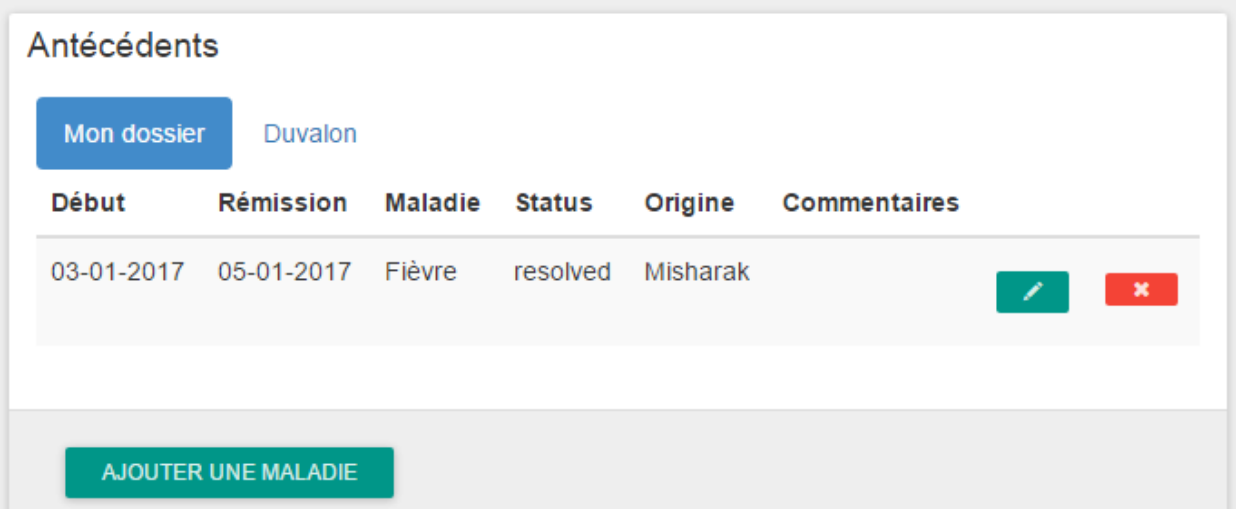

Figure 34: Eglé's "History" widget

5.3.b. Information shared by the patient

Any information the patient enters in the application is directly transmitted to the EHR server and will be shared with all those the patient gave the code of his record to. Once the code is given, all the data will be viewed.

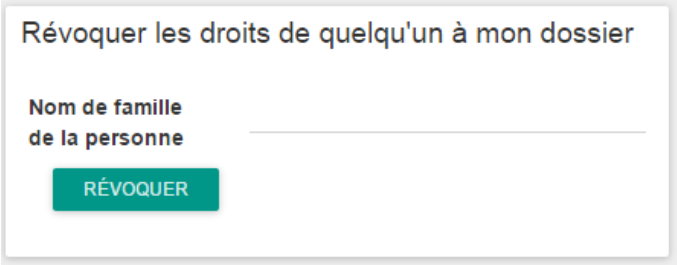

Figure 35: Eglé's "revoke rights" widget

On the EHR page, there's a widget that allows the patient to revoke the right of someone to see his record, but he can control the potential copies the person made. Those will still be accessible. It will be the responsibility of the patient to choose wisely with whom he shares his record.

5.4. Security

5.4.a. Authentication

The authentication is made using JSON Web Token (JWT) (see 4.1. (viii) . This is a stateless authentication mechanism.

A token is created each time the user logs in. The client saves it locally and sends it with each request by placing it in the "authorization" header. It replaces the usual session and cookies. It can be used across different domains and it can securely transmits information because they can

be signed and encrypted using public and private keys. It means the tokens can contains all the information we want and reduce the need to access the database.

The advantages are multiple. JSON is a compact language less verbose than XML, the size of the data transmitted is less which is always appreciated with HTTP. There's parsers that can easily map JSON to objects.

For this application, the token contains all the user basic information. We can directly check if he has the right to make a particular request to the server without first needing to extract his data from the database. The verification is done using the middleware option of node.js. Before acceding to a request, we can do a series of operations to check whether or not we should reject the request.

```
app.get('/ehr/rest/Patient', auth.ensureAuthorized, rest.search);
```

Here, it is the "auth.ensureAuthorized" function that does the necessary verifications. It extracts the token from the header, checks if it is valid, decodes it, saves the decoded information and does a really basic access control before allowing the request to continue:

```
jwt.verify(token, process.env.JWT_SECRET, function(err, decoded) {
	if (err) {
		console.log("CheckToken error: " + err);
		return res.sendStatus(403);
	} else {
		req.user = decoded;
if(!req.params.pId && checkBasicAccess(req.params.model, req.user, req.method))
	next();
else if (req.params.pId){
	// check if the resource we want to edit belongs to the record
	// whose id is in the parameters
	if(req.method == 'POST' || req.method == 'PUT'){
		if(req.params.model.toLowerCase() != 'patient'){
			var ref = 'Patient/' + req.params.pId;
			if(req.body.subject && req.body.subject.reference != ref){
				return res.sendStatus(403);
			}
			if(req.body.patient && req.body.patient.reference != ref){
				return res.sendStatus(403);
			}
		} else{
			var ref = 'Patient/' + req.params.pId;
			if(req.body.id && req.body.id != ref){
				return res.sendStatus(403);
			}
		}
	}
	next();
}
else{
	console.log("Access refused for " + req.params.model);
	return res.sendStatus(403);
}
}
});
```

On the client side, it is the module http provider's interceptor that takes care of inserting the token into each request made to the server:

```
$httpProvider.interceptors.push(['$q', '$window', function($q, $log, $window) {
     return {
          'request': function (config) {
             config.headers = config.headers || {};
             if ($window.localStorage.serverToken) {
              config.headers['x-access-token'] = $window.localStorage.serverToken;
             }
             return config;
          },
          'responseError': function(response) {
             return $q.reject(response);
          }
     };
 }]);
```

5.4.b. Access Control List

To manage the permissions to the EHRs, I made use of an access control list (ACL). An ACL is a list of entries specifying the roles of each user and the permissions accorded to each role for some resources. I choose not to implement the ACL system myself and made used of the npm module "acl".

In my case, the resources are the URLs each user is allowed to access and there is one role per user. When someone request access to a record, if the data and code provided are correct, a role will be created for this user (if he doesn't already have one) and this role will have the permission to view the record. The creator of a record has all the permissions. He can edit and delete it and he can revoke someone else access to it.

```
var role = "user/" + user._id;
acl.allow(role, urlEHR + idRecord, 'view');
acl.addUserRoles(user._id, role);
```

For example: when the user makes an http GET request to the URL:
`https://server/ehr/rest/patientId/d5d5d46dfs845fds685/Condition`
The user is asking for all the conditions related to the record with the id "d5d5d46dfs845fds685". The URL "`ehr/rest/patientId/d5d5d46dfs845fds685`" represents the record and is the resource that will need to be protected. So each time the server will receive this request, he will check if the user has a role that is allowed access to this resource. The verification is done using the middleware provided by the "acl" module.

```
app.get('/ehr/rest/patientId/:pId/:model', auth.ensureAuthorized,
auth.acl.middleware(4, auth.getUserId, 'view'), rest.search);
```

A role represents a user and a URL represents a record. After several exchanges of record between users, the users' roles will have the permissions to view several records.

5.4.c. Encryption

Medical data being so sensitive, it was a given that all the exchanges as well as the data stored needed to be encrypted.

(i) HTTPS

HTTPS is a widely used protocol for secure communication over the internet. It uses the public / private key encryption system with the Diffie-Hellman key exchange. The public key is a certificate signed by a trusted Certificate Authority (CA). CA are companies charged to ensure that the entity it gives the certificate to exists and is in charge of the domain they claim.

Since the testing phase of this project was done on localhost, I didn't pay for a certificate and used a dummy instead.

(ii) Crypto-js

For the storage part of the project, I used the "crypto-js" npm module to encrypt the information in the database. For the desktop application, the database "Nedb" provides an option to transform data after it was serialized and before it is written to disk and another to undo the transformation before it is deserialized. We only need to provide it with the function that does the transformation.

Example:

```
db.users = new Datastore({
            filename : path.join(appDataPath, 'users.db'),
            afterSerialization : EncryptionService.encrypt,
            beforeDeserialization : EncryptionService.decrypt,
            corruptAlertThreshold : 0,
            autoload : true
      });
```

The function I made uses crypto-js "`AES.encrypt`" to encrypt the data and the reverse "`AES.decrypt`" afterward to decrypt it.

AES or Advanced Encryption Standard is a standard for the encryption of electronic data defined by the U.S. National Institute of Standards and Technology. The AES algorithm is a symmetric block cipher [27]. This means it uses the same key for both encrypting and decrypting. It is used worldwide and while there has been successful attacks against it, it is considered secure enough.

6. Possible improvements

6.1. Link to a SNOMED CD database

The project uses a minuscule part of all that SNOMED has to offer. Just a few codes that describe the diseases or lab test results are used and their corresponding descriptions are hard coded in the application.

If the project is ever expanded, using the full SNOMED CT database with all the specifications would be necessary. For example, it could help the user type his disease's name in the form with an auto-complete functionality. It would ensure that the name written does not have any typo. And since we would have all the codes' corresponding clinical terms readily available, the applications would be able to interpret every code without needing to check the "display" attribute. Language wouldn't be a barrier anymore since the codes are the same in every language and have a corresponding description.

6.2. Connection via identity card

That is a functionality I was planning to implement but lacked the time to do. As it is, the connection to the EHR server is done with an email and a password. This is seriously lacking. There is no guaranty of the user real identity. Thankfully, it does not impact the security of record sharing since the code of the record needs to be given orally to the person and his identity can be confirmed at the time. But the application would benefit greatly from a more accurate way of authentication.

6.3. Medication and prescription functionality

When a person has several health issues, she can end up with a lot of medications prescribed by several doctors. Keeping track of all of them to be able to give the complete list when asked by the doctors is not an easy thing. And forgetting to mention one can potentially have severe consequences if it has a bad interaction with another medication.

Being able to create an electronic list of all the medications the patient takes (and all he has taken in the past too) whether it is prescribed or self-medication, add it the EHR and enable the sharing of the list would, I believe, be incredibly helpful in some cases. And even more with a functionality that automatically detects interactions between medications.

That would require access to a medication database that is kept up to date with all the new medications. It would also require the medication database to have all the active ingredients contains in the drugs and we would need another database that contains all the possible interactions with their severity level.

The resources needed could be difficult to acquire but it definitely would be worth it.

6.4. More information in the patient record

EHR can contains lots of information. Just about anything in the patient life could be useful to know medically one day. Functionalities could be added to collect all this information.

Even without going deep into the details of the patient life, there is still some essential information asked by almost every practitioner that is missing from the project. To cite only those that seems the most important to me: the risk factors, allergies and intolerances, biometrics variables and family history.

6.5. More granularity for the permissions

As it is, once the user gives the code of one of his EHR to someone else, the person automatically has access to everything the user chose to put on the server. There's no differentiation between users. Everyone that has access to the EHR sees the same thing.

That could one day pose problem if we end up including some really sensitive information (such as gynaecology or psychiatry data for example) that the patient (or practitioner) wants to share with only one person (for example only her gynaecologist or her psychiatrist).

To resolve this problem, we would need to implement more granularity in the permissions. We could follow the Facebook model. Each user could create groups and put the persons he chooses to share his EHR with in a group. And the user could then determine which part of his EHR each group is allowed to see.

6.6. Sharing of all the data collected by Eglé

The project does not fully exploit all the possibilities offered by the link with Eglé. The only data sharable with the practitioners is the one collected in the "EHR" part of the application. But it is not the only data available on Eglé. All the previous widgets contains information that could be useful to the practitioner.

Enabling the sharing of all the data in Eglé and making it appears on the desktop application would be a nice improvement of the project.

6.7. Alert system

One of the functionalities of interest to the practitioners interviewed was the possibility to have a system that alerts them when the data collected for one of their patient showed an abnormality or a danger.

The alert would be shown on the dashboard of the desktop application so that the practitioner sees them as soon as he launches it.

There would be the need to determine when to generate an alert. The documentation already provides some answers at this question. We know the theoretical maximum and minimum thresholds for variables such the glycated haemoglobin, the blood pressure, the weight gain or loss, the heart rate or the glycaemia level. But this is all theoretical knowledge that can vary a bit from person to person. Ideally we would need to ask the patient or his practitioner to set themselves the alert thresholds.

We should also let the practitioner choose for which patient he wants to receive alerts.

7. Conclusion

Looking back at the objectives at the start of the project, I can say that they have been mostly fulfilled. There is room for improvement with some of them as indicated in the previous section, but only the last “user experience design” objective has been left out. I was not able to get a team of practitioners and patients to test the application and give me feedback, due to a lack of time and organisation mostly. I could only glean a few advices, remarks and warnings from one or two doctors and no patient tested the new Eglé widgets. I relied on online documentation on software design and my own opinion to try to make an application that is intuitive and easy to use.

The first goal “data storage” has been completed. The REST server is implemented and operational, the FHIR format allows the encoding of any kind of medical data.

The second goal “building a small desktop application” has been mostly completed. The desktop is implemented and operational and it indeed allows the users to create a mini health record and share it with other practitioners and his patients. It is also able to receive and integrate data from other sources. But there is still so much that could be done, the health record created is so minimalist compared to all the possibilities that it can only be considered “mostly” completed.

The third objective “link with Eglé” has also been mostly completed. Desktop application and website can communicate and exchange data, but once again the data exchanged is just a fraction of all that could be done.

All the questions in the fourth objective “security” have been answered. The data is encrypted, access is restricted and controlled with ACL. Authentication is done with JSON web tokens. Rights and responsibilities have been made as clear as possible. Each user is responsible for the record he creates. No one but him is authorized to modify the record. Only the right to see and copy the record can be granted. It does mean there exists several records referring to the same person on the server, but the redundancy allows the users to have more control over their EHR and their responsibilities and rights are clearer.

I learned a lot during this project. I learned as much from all the problems, difficulties and questions I encountered during its realisation than from the people I met and interacted with. It definitely was an interesting project and working on something that could potentially help people with chronic diseases and be useful in a medical context was extremely motivating.

8. References

[1] "OMS - Maladies chroniques," 18 Octobre 2016. [Online]. Available: http://www.who.int/topics/chronic_diseases/fr/.

[2] "WHO - Diabetes," 10 2016. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs312/en/.

[3] "What is HbA1c?," [Online]. Available: http://www.diabetes.co.uk/what-is-hba1c.html. [Accessed December 2016].

[4] "Definition of Heart failure," 28 08 2013. [Online]. Available: http://www.medicinenet.com/script/main/art.asp?articlekey=3672. [Accessed 24 10 2016].

[5] "Heart Failure," 22 September 2016. [Online]. Available: http://www.medicinenet.com/heart_failure/article.htm. [Accessed 24 October 2016].

[6] "Commission Recommendation of 2 July 2008 on cross-border interoperability of electronic health record systems (notified under document number C(2008) 3282)," 2008.

[7] "Réseau de santé wallon - Fonctionnement," 2016. [Online]. Available: https://www.reseausantewallon.be/FR/patients/Information/general-informations/Pages/Operating.aspx. [Accessed 24 October 2016].

[8] "Healthcompass - Eglé," [Online]. Available: http://lifetechbrussels.com/members/healthcompass/. [Accessed 07 November 2016].

[9] "FHIR - Summary," [Online]. Available: https://www.hl7.org/fhir/summary.html. [Accessed 16 November 2016].

[10] "About HL7," [Online]. Available: http://www.hl7.org/about/index.cfm?ref=common. [Accessed December 2016].

[11] "FHIR Overview," 24 October 2015. [Online]. Available: https://www.hl7.org/fhir/overview-clinical.html. [Accessed December 2016].

[12] "FHIR - Data type "Coding"," [Online]. Available: https://www.hl7.org/fhir/datatypes.html#Coding. [Accessed 16 Novembre 2016].

[13] "License and legal terms," [Online]. Available: https://www.hl7.org/fhir/license.html. [Accessed 07 November 2016].

[14] "SNOMED CT Starter Guide," [Online]. Available: https://confluence.ihtsdotools.org/display/DOCSTART. [Accessed December 2016].

[15] "Introduction - KMEHR," [Online]. Available: https://www.ehealth.fgov.be/standards/kmehr/.

[16] "MEAN.js," [Online]. Available: https://meanjs.org/. [Accessed 16 November 2016].

[17] "Introduction to MongoDB," [Online]. Available: https://docs.mongodb.com/manual/introduction/. [Accessed 16 November 2016].

[18] "Express," [Online]. Available: http://expressjs.com/. [Accessed 16 November 2016].

[19] "What is Angular?," [Online]. Available: https://docs.angularjs.org/guide/introduction. [Accessed 16 November 2016].

[20] B. Wen, "6 things you should know about Node.js," 12 December 2013. [Online]. Available: http://www.javaworld.com/article/2079190/scripting-jvm-languages/6-things-you-should-know-about-node-js.html. [Accessed 16 November 2016].

[21] "Node Webkit - Introduction," [Online]. Available: https://github.com/nwjs/nw.js.

[Accessed 16 November 2016].

[22] “GitHub - louischatriot/nedb,” [Online]. Available: https://github.com/louischatriot/nedb. [Accessed December 2016].

[23] “JSON Web Token,” May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519. [Accessed December 2016].

[24] “Bootstrap,” [Online]. Available: http://getbootstrap.com/. [Accessed 16 November 2016].

[25] github, “Search - stars:>1,” [Online]. Available: https://github.com/search?l=&o=desc&q=stars%3A%3E1&ref=advsearch&s=stars&type=Repositories. [Accessed 16 November 2016].

[26] “FHIR - Resource index,” 24 October 2015. [Online]. Available: https://www.hl7.org/fhir/resourcelist.html. [Accessed September 2016].

[27] “Announcing the ADVANCED ENCRYPTION STANDARD (AES),” 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. [Accessed 2016].

[28] “Github - Eglé,” [Online]. Available: https://github.com/uclouvain/egle. [Accessed 07 November 2016].

Table of figures

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgium **www.uclouvain.be/epl**