INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Infraestrutura baseada em HL/'s FHIR para interoperabilidade de temrinologias clinicas

ANA RITA AZEVEDO LOPES
Julho de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Infrastructure based on HL7's FHIR for clinical terminologies interoperability

Ana Rita Azevedo Lopes

Dissertation to obtain the Master Degree in Informatics Engineering, Area of Expertise in Graphical Systems and Multimedia

Advisor: Luiz de Faria

Co-advisor: Antonio Constantino Martins

Supervisor: Humberto Cardoso

Jury:

President:

Other members:

Porto, July 2020

Dedication

I dedicate this thesis to my family, specially my beloved aunt Margarida Azevedo for all the care provided to every family member and friend, and whose strength and positive attitude during harsh moments is an inspiration in everything I do. Although she is no longer with us, her memories continue to leave a smile in every face she encountered during her life, and I can be sure that it would be a pride for her to be present in this moment of my life.

Resumo

A ALERT® é uma empresa de software clinico com produtos espalhados pelo mundo. Estes produtos funcionam em hospitais e clinicas onde outro software e produtos de IoT estão em utilização. Posto isto, os produtos da ALERT® precisam de comunicar com produtos de outros fornecedores, e esta comunicação nem sempre é fácil, pois cada software tem o seu próprio conhecimento interno. Para alcançar interoperabilidade em informação sensível, tal como é qualquer dado clínico, *Health Level Seven* (HL7) criou o FHIR, um protocolo de comunicação e modelação de dados que faz uso de serviços REST. Este é o protocolo mais popular a nível global, e qualquer software que seja capaz de o interpretar está preparado para comunicar com muitos outros. Assim, é benéfico para a ALERT® utilizar este protocolo nos seus produtos.

Para além deste protocolo, os dados em si são codificados usando terminologias clinicas, que são sistemas para classificar conceitos usados em contextos de cuidados de saúde. Estas terminologias garantem que o significado de um determinado conceito é o mesmo em todo o lado, independentemente da linguagem do contexto ou do conhecimento da pessoa.

Assim, este documento foca-se no estudo e desenvolvimento de um conjunto de serviços REST que expõem terminologias clínicas e que são compatíveis com a especificação do FHIR. Estes serviços compõem um servidor de terminologias, que deve ser capaz de responder a todas as necessidades da ALERT® neste campo. Para que o sucesso seja alcançado, o desenvolvimento segue os requisitos definidos após um trabalho de estudo na matéria. O produto será depois avaliado consoante um conjunto de testes que avaliam os requisitos, e o seu sucesso discutido numa apreciação final.

Palavras-chave: Cuidados de Saúde, HL7 FHIR, Interoperabilidade, Java, Terminologias Clinicas

Abstract

ALERT® is a clinical software company with products spread throughout the world. These products perform in hospitals and clinics, in which other software and IoT products are in use. This said, ALERT® products need to communicate with products from other vendors, and this communication is not always easy, since each software as its own internal knowledge. In order to achieve interoperability in sensitive information, as is any clinical data, HL7 created FHIR, a communication and data modeling protocol, making use of REST services. This is the most popular protocol on a global scale and any software that can interpret it is ready to communicate with many others. Therefore, it is beneficial for ALERT® to use this protocol within its products.

Beyond this protocol, the data itself is often codified using clinical terminologies, which are systems for classifying concepts used in a healthcare context. These terminologies ensure that the meaning of a certain concept is the same everywhere, despite the language of the context or knowledge of the person.

Hence, this document focus on the study and development of a set of REST services that expose clinical terminologies and are conformant with the FHIR specification. These services compose a Terminology server, which should be able to respond to every ALERT® need. In order to achieve success, the development follows ALERT® requirements created after a study on the field. The product will later on be evaluated by a set of tests that evaluate the requirements, and its success is discussed in a final appreciation.

Keywords: Clinical Terminologies, Healthcare, HL7 FHIR, Interoperability, Java

Acknowledgements

I owe my greatest gratitude to my advisors Engineering Doctorate Luiz de Faria and Engineering Doctorate Antonio Constantino, as well as to all the teachers in both ISEP and Aveiro University, the institution that received me in the beginning of my higher education. A special thanks to my supervisor nurse Humberto Cardoso, and Aveiro University teachers Engineering Master Carlos Santos and Engineering Master Telmo Silva, the major boosters of my interest in the field.

In addition, I must thank ALERT® for this opportunity and for the support offered throughout the development of this thesis, along with its collaborators, which welcomed and helped me to fit in.

Finally, I owe my greatest gratitude to my family and friends for all the belief, encouragement provided during my education. These people are the dearest to me, being one of the foundations for my motivation and effort.

To all of the people who supported my academic path, and specially this project, I pay my best regards.

Table of Contents

List of Figures

List of Tables

Acronyms and Glossary

Acronyms List

ADW	ALERT® Data Warehouse
AHP	Analytic Hierarchy Process
ALERT	The enterprise ALERT Life Sciences Computing, S.A.
ALERT®	Software developed by ALERT
API	Application Programming Interface
CDA	Clinical Document Architecture
CDW	Clinical Data Warehouse
CORS	Cross-Origin Resource Sharing
CR	Consistency Ratio
CSIRO	Commonwealth Scientific and Industrial Research Organisation
DDD	Domain Driven Design
EHR	Electronic Health Record
FFE	Fuzzy Front End
FHIR	Fast Healthcare Interoperability Resources
FURPS+ and more	Functionality, Usability, Reliability, Performance, Supportability
GRASP	General Responsibility Assignment Software Patterns
GoF	Gang of Four
HIE	Health Information Exchange
HIMSS	Healthcare Information and Management System Society
HIT	Healthcare Information Technology
HL7	Health Level Seven
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICD	International Classification of Diseases

IoC	Inversion of Control
IoT	Internet of Things
IHTSDO	International Health Terminology Standards Development Organization
ISEP	*Instituto Superior de Engenharia do Porto*
ISO	International Organization for Standardization
IT	Information Technology
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JSE	Java Standard Edition
JSON	JavaScript Object Notation
LCIM	Levels of Conceptual Interoperability Model
LOINC	Logical Observation Identifiers Names and Codes
NCD	New Concept Development
NPD	New Product Development
OID	Object Identifier
OOD	Object Oriented Design
PaaS	Platform as a Service
QEF	Quantitative Evaluation Framework
QFD	Quality Function Deployment
REST	Representational State Transfer
RIM	Reference Information Model
SMART	Substitutable Medical Applications and Reusable Technologies
Smile CDR	Smile Clinical Data Repository
SNOMED CT	Systematized Nomenclature for Medicine Clinical Terms
SQL	Structured Query Language
STU	Standard for Trial Use
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VA	Value Analysis

VE	Value Engineering
XML	Extensible Markup Language
XHTML	Extensible Hypertext Markup Language
WHO	World Health Organization

1 Introduction

The current chapter introduces the problem, contextualizing it and describing its main objectives. It also presents the hypothesis of solution, the procedures that will be adopted throughout the process of design and develop of the solution alongside with some tools. Finally, it describes the structure of this document.

1.1 Context

This project arises in the context of completing a Master's Degree in Informatics Engineering from *Instituto Superior de Engenharia do Porto* (ISEP), being the chosen work to develop under the Thesis/Dissertation/Internship Course, whose main goal is to challenge students to design, implement and test an information technology (IT) solution to an existing problem.

The company ALERT Life Sciences Compunting, from now own mentioned as ALERT, proposed the chosen project, "Infrastructure based on HL7's FHIR for clinical terminologies interoperability", and its focal point is on terminology interoperability, a popular study object in electronic Health (eHealth) since the beginning of the implementation of digital systems in health care. ALERT provides a suite of software for healthcare named ALERT®, with several implementations around the globe.

Interoperability in data exchange is one of the most discussed implications of Healthcare Information Technology (HIT) still far away from being solved, being the theme of a plurality of studies and implementations throughout the entire world. It has proved to be the way to globalize health care information, making it coherent everywhere, allowing conduction of larger studies, and motivating an exponential growth in medicine discoveries. In order to achieve interoperability in data exchange is necessary to define standards as multiple organizations spread all around the globe

have been developing over time. The most technologically advanced health corporations turned out to choose and adopt some of these standards on their HITs, demonstrating a general inclination towards Health Level Seven International (HL7) standards.

The high level of rigor denoted in the medical informatics market comes from its importance for humanity, dealing directly with people's health, it is a market where there should be no margin for mistakes. As such, it becomes extremely important for ALERT to be on the edge of technology standards producing better quality software that fits the current state of eHealth.

1.2 Problem

Information Technology (IT) is an area in constant evolution, as medicine is always improving itself, with the creation of new standards, protocols and procedures, and with the discovery of new diseases and ways to diagnose them. With this, IT companies that focus on healthcare software need to keep up with these improvements so that they can deliver effective and useful features to their software.

In order to describe unambiguously health care information, healthcare software makes use of clinical terminologies and classification systems, which identify concepts and assign to them terms and codes in a structured way. Although these represent standards to code information like diagnoses, diseases, procedures, drugs and more, a lot of the clinical software has a need to exchange information with other software for electronic health record (HER) reporting, insurance and other administrative purposes, and not all of the software is compliant with the same communication protocol. This is mostly due to each company storing information in their own data model, as well as the amount of coding standards that exist and can became conflicting.

Yet, this communication is crucial to digitalize and automate health care workflows, and it depends nowadays a lot on interfaces for transforming data models collected by a system that need to be interpreted by another system. All of this contributes to a lack of interoperability of clinical terminologies that has proven to be an important asset for medicine's evolution far from resolution. "They're success will be fully achieved once software is able to use and re-use, and when multiple independently developed medical records, decision support, and clinical information retrieval systems share the same information using the same terminology in everyday use." (Rector, 1999).

HL7 developed a new standard for healthcare software communication, the Fast Healthcare Interoperability Resources (FHIR), defining a single structured data model to exchange every type of healthcare information using Representational State Transfer (REST) Web Services. This represents a new implication for ALERT since its software is

not fully compliant with this standard, so there is a need to create mechanism of transforming ALERT® internal data model to the FHIR model and communicate information using the FHIR protocol.

1.3 Hypothesis

When analyzing the problem it becomes important to formulate a hypothesis for a solution, so that, later on, this hypothesis can be tested in order to conclude about the project achievement level. The hypothesis is related with the achievement of interoperability by developing an infrastructure capable of exposing clinical terminologies and classification systems in a single data model, documenting every necessary attribute of each terminology, using HL7's FHIR. This mechanism should be independent from ALERT® and interact with a database that contains the terminologies that are currently in use by the company within its working markets. It is also important to enhance the need of the system to be efficient and secure. Finally, the preferred technology for the development is Java, however this is not a restriction since any other technology is acceptable as long as its use is justified. In sum, the hypothesis is that the developed system fulfills the established requirements.

1.4 Goals

ALERT® operates in a diversity of health markets spread throughout the entire globe, making use of a large number of terminologies and classifications systems and communicating with a diversity of external systems.

The previous section already portraits the main goal of this project albeit it is denoted that the solution's accomplishment of functionality, performance and safety must be proven, so the Web Services should be tested in these and other necessary domains specified in the requirements analysis phase. These tests should aim for requirements' fulfillment in order to measure the level of accommodation of the solution to the problem.

Besides this main goal, other objectives become important to the scope of the project. Firstly, it is necessary to analyze FHIR technical specification in order to make the best design decisions for the product, following the standard's restrictions. Then a study on the several terminologies must be performed in order to know exactly what attributes each one documents, resulting in a single unified data model compliant with FHIR® able to accommodate all terminologies maintaining the information's integrity. After this, the services can be developed according to the defined data model and making use of

good design practices, followed by the development of the security features and constant testing, for later tweaking of the system to achieve better performance. The documentation associated with the solution must be written regarding all the design and implementation decisions and their proper clearings.

1.5 Procedures

1.5.1 Methodology

During the development of the project a waterfall methodology was adapted, specifically the Rational Unified Process (RUP), a process developed and maintained by Rational® Software (Rational Software, 2011). The methodology was adapted to the project context and needs, as well as the developer. In waterfall-like methodologies is adopted a sequential approach where each stage can only continue when each of the previous stages are completed and the focus is on fixed dates, requirements and outcomes (Gallagher, Dunleavy and Reeves, 2019).

RUP motivates practices like iterative software development, requirements management and visually modeled software, which are considered adequate to the project. Since the solution should be developed by a single developer, supervised by a single more experienced one, an iterative approach is benefic since the developer can define what to do and when, having the main goal in mind. RUP defines an initial stage for identification of actors, use cases, scenarios, business model and requirements, followed by an elaboration phase where the software is designed through complement of the use cases and requirements, and architectural design. Then comes a construction phase for software development and a testing phase. Finally it describes a maintenance phase that won't apply to this project (Rational Software, 2011).

1.5.2 Version Control and Task Management

The version control system used in ALERT is Apache® Subversion® software project (SVN) which is open source and was developed by Apache Software Foundation (Foundation, 2018). TortoiseSVN is used has the client application, a free and easy to use Windows shell extension (TEAM, 2019). These are the tools used in the product for version control.

In association with these tools, a task management application - Jira – is used in order to track and manage task completeness, report problems and map tasks to the source code of the product. Jira is an Atlassian product and allows managing anything product related within the same platform, making use of tasks called Issues.

1.5.3 Planning

Following the chosen methodology – RUP – on an initial stage of the project is defined a plan with fixed dates, although an agile-like approach was merged into this methodology, adjusting this dates when necessary. Another change to the RUP was that is added a stage at the beginning of the process, dedicated to acquiring theoretical knowledge about the project's context and the concepts it involves. Additionally in this phase, is conducted a study of the HL7's FHIR standard and existing implementations of this standard. The forthcoming phases follow the description of RUP aforementioned, with the exception of the last phase – the maintenance phase – that will befall this project. For the, now second, stage, in terms of requirements, the FURPS+ model was embraced.

1.6 Document Structure

With except of the current chapter, the document adopts the following structure:

- **Theoretical knowledge:** is responsible for explaining a set of theoretical concepts indispensable for understanding the project. These concepts relate to eHealth and interoperability. Furthermore, the HL7's FHIR are introduced;
- **State of the Art**: the section presents a study of some existing implementations of FHIR resources in a terminology context;
- **Analysis:** the section presents an analysis of the project's value. Identification and analysis of use cases and requirements. Beyond this, it defines metrics to evaluate requirements fulfillment and describes the decisions made about the technologies to be used in the development phase;
- **Design:** describes the design of the business and data model, as well as the architectural design of the product, based on current standards and good practices for web and software development, as well as the FHIR protocol specification;
- **Development**: the section documents the development stages, detailing the implementation of the design previously made;
- **Experiments**: this section introduces a set of tests developed to evaluate the software, countering the defined metrics for the requirements' fulfillment. The results of these tests are shown and judgements are formed. This is where the hypothesis is evaluated;
- **Conclusions:** as implicit by the title, this section features conclusions about the project, room for it to grow and improve and, finally, it summarizes and qualifies the reached outcomes in a general appreciation.

2 Theoretical knowledge

In order to understand this project it is vital to acquire some theoretical knowledge for a better understanding of some business concepts and maintain formal distinction. The sequence of concepts here explained follows a bottleneck direction, starting in the most generic to the most specific concept.

2.1 Interoperability

As the name says, FHIR (Fast Healthcare Interoperability Resources) are a tool for interoperability in healthcare. To understand the value contributions of this standard to the HIT community it becomes necessary to understand what is interoperability, what are the levels of interoperability and its relevance within this community.

Hufnagel (2009) defines interoperability as the ability of an IT system component to work with other IT systems components without special effort on the part of the user. In harmony with this definition, Healthcare Information and Management Systems Society (HIMSS) (no date) defines it as the extent to which systems and devices can exchange data, and interpret that shared data, using standards that enable data to be shared across disparate healthcare settings regardless of the application or vendor. Interoperability is an important aspect of Health Information Exchange (HIE) that enables the addition of meaningful use of information to simple information exchange, by ensuring, at the higher level, that the data being shared is interpretable and operable by HIT.

“The benefits of joined-up health care, to provide the right information at the right time and place, are predicated on deploying and using standards that enable computer systems to exchange information in a way that is safe, secure, and reliable. (..) The

benefits increase exponentially and more parties are involved, because the number of interfaces needed to connect N systems increases using the formula (N^2-N)/2" (Benson and Grieve, 2016, p. 26). So with the creation of standards for HIE the need for vendors to achieve information compliant among each other is simplified, allowing them to discard the creation of an unsustainable number of interfaces, as well as negotiations for who should be responsible for creating each interface.

Although interoperability brings these and more cost effective benefits, full interoperability is still a large challenge in HIE as it can be seen by analyzing Figure 1 retrieved from Benson and Grieve (2016, p. 34).

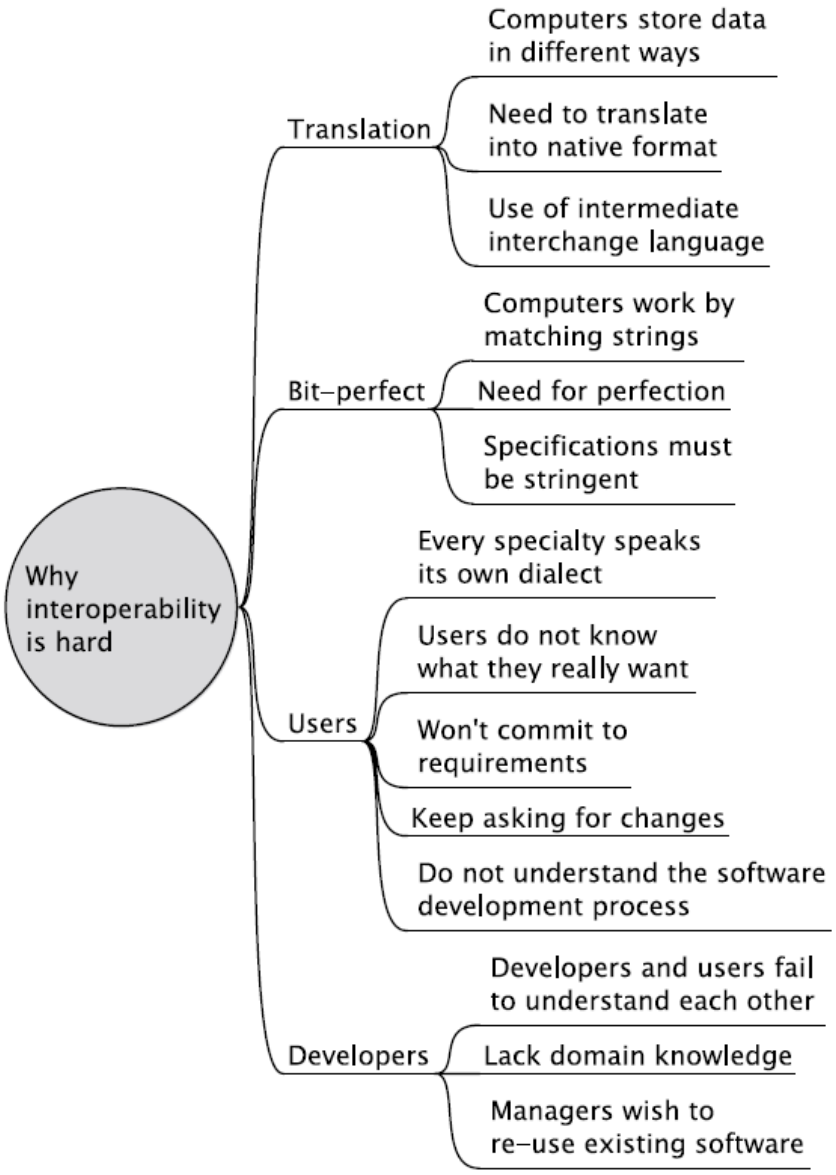


Figure 1 – Why Interoperability is so hard

2.1.1 Levels of Interoperability

Interoperability is achievable at several levels defined by a model named Levels of Conceptual Interoperability Model (LCIM) created by the Virginia Modeling Analysis & Simulation Center and presented in Figure 2. Albeit this model was created for modeling and simulation, other IT domains that attribute importance to modeling can apply it to their products. Following Tolk, Diallo and Turnitsa (2007) explanation's, the levels are characterized as follows:

- **Level 0:** Stand-alone systems;
- **Level 1:** A communication protocol is unambiguously established;
- **Level 2:** A common structure to exchange information exists;
- **Level 3:** A common information exchange reference model is used so the meaning of data is shared;
- **Level 4:** Interoperating systems are aware of the methods and procedures that each other are employing;
- **Level 5:** Interoperating systems are able to comprehend the state changes that occur in the assumptions and constrains that each other is making over time, and are able to take advantage of those changes;
- **Level 6:** Interoperating systems share conceptual models documented based on engineering methods enabling their interpretation and evaluation by other engineers.

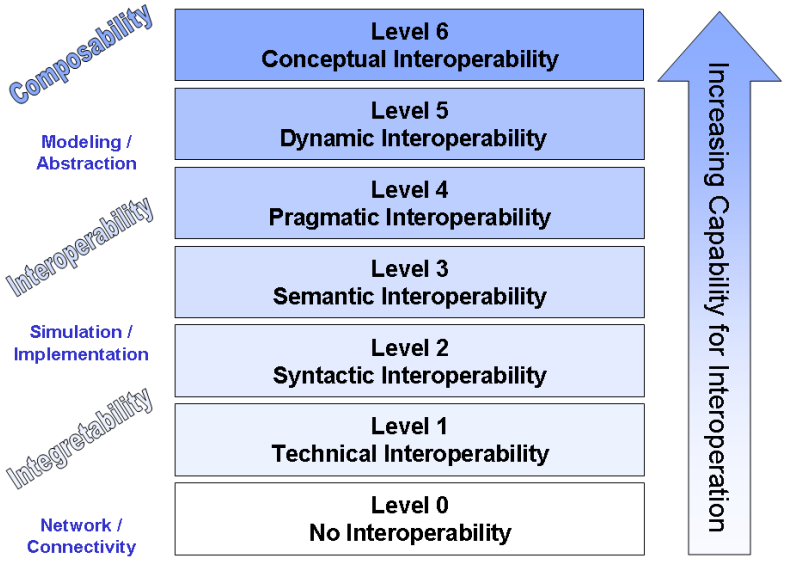


Figure 2 – Levels of Conceptual Interoperability Model

Achieving interoperability at service level in eHealth is the logical answer to improve healthcare, being the most cost-effective approach. Available clinical data across facilities and professionals undeniably enhances patient safety by providing decision

support to professionals who will be more informed, accelerate care by reducing paperwork and time, and even reduce redundant procedures (Hufnagel, 2009).

2.2 Clinical Terminologies

Clinical terminologies and classification system are the base of all interoperability in eHealth, falling under the third level of interoperability because they are the base to share meaningful health and administrative information by its codification. According to Chute (2000) “Aristotle introduced the notion that abstract concepts represent description, or more properly definitions, of things that have been classified by describing their attributes”.

A clinical terminology codifies information relative to diagnosis, procedures, medicine, and more, and can assume many forms of controlled vocabularies. “A controlled vocabulary is a restricted list of words or terms typically used for descriptive cataloging, tagging or indexing” (Hedden, 2010). Hedden further explains that they are controlled not only because of users only applying terms for the vocabulary’s scoped area but also because changes are only made by the editor or taxonomist only under specific conditions. She presents four types of controlled vocabularies described as follows:

- **Term List:** flat term list;
- **Authority File:** terms list that provide synonyms, cross-references and alternatively notes on each term;
- **Taxonomy:** controlled vocabulary in which all the terms belong to a single hierarchical structure and have parent/child or broader/narrower relationships to other terms. It may have synonyms;
- **Thesaurus:** a list like an authority file but synonyms can be used too due to it being designed context independent.

It is worth noting that although “taxonomy” is formally described as above, in the corporate world it is generally used to refer to controlled vocabularies in general due to the simplicity of the single word. Furthermore, she introduces preferred and non-preferred terms has classifications for the main term and its synonyms and alternative terms respectively. However, clinical terminologies do not all fall under one of these types of vocabulary, usually they combine characteristics from more than one of them and even add other meta-data fields not described in any of the vocabularies.

Chute (2000) presents the following definition for clinical terminology: “Standardized terms and their synonyms which record patient findings, circumstances, events, and interventions with sufficient detail to support clinical care, decision support, outcomes research, and quality improvement; and can be efficiently mapped to broader classifications for administrative, regulatory, oversight, and fiscal requirements”.

However, if clinical terminologies are to be used worldwide, a problem associated to language arises, and with it a basic solution: codification.

A label or a term and a code are the basic mandatory fields in a clinical terminology. The term and code are associated with a concept, which is the abstract knowledge that is being represented by the term. That is, the concept is the abstract knowledge holding the meaning, the code is the global representation with implicit knowledge and the term is what turns implicit knowledge into explicit by the use of words. With the identification of a concept using a code and not a term, global unambiguity is guaranteed, even if the characters used in the code are not interpretable in every language because the knowledge it represents is the same. Hereupon, by the codification of health and administrative concepts it becomes possible to have information passible for machine's interpretation, thereby achieving semantic interoperability with the terminologies acting as the reference model that holds the information's meaning.

There are a large number of clinical terminologies used internationally, usually released in English, being that each country or collective of countries has an entity responsible for publishing translations. Beyond translations, each terminology usually has several versions resulting from reviews and updates published by the creator entity. These versions usually take a while to translate and implement, so it is common to have multiple versions in use around the world at the same moment of time. Thus, in order to guarantee interoperability using terminologies has reference models, it becomes critical to keep track of the changes between versions as well as to map concepts inter-versions when possible, without losing the integrity of the meaning. In addition to this, since each terminology has its own data structure, maximum interoperability is still a complex problem. Usually the terminologies do not compete with, but rather complement each other, each one codifying at a certain level of coverage and reclining on a set of health purposes.

3 State of the Art

The development of any IT solution should start with the acquisition of knowledge on similar solutions, this is, their capabilities and, when possible, how they were designed and implemented. The current chapter intends to showcase conceptual standards which define a pattern for mapping concepts, including HL7's FHIR and others, justifying why FHIR is the most appropriate within the context of this project and why it gained so much popularity in the HIT industry. It is also here where some existing implementations of FHIR in several programming languages, and terminology servers are highlighted, some of them making use of the FHIR specification and some of them not, and some of them gathering several terminologies and other dedicated to only one.

3.1 Clinical Terminology Codification Standards

3.1.1 OpenGALEN Project

GALEN is a technology designed to represent clinical information in a new way, producing a computer-based multilingual coding system for medicine, aiming to mitigate interoperability problems in medicine. The OpenGALEN foundation was born in 1990 inspired by a program named Practitioners Entering Notes & Practitioners Accessing Data (PEN&PAD). According to their first technical report by Nowlan, 1993, this program aimed to research, design and develop prototype HIT for daily use by clinicians, in order to accomplish successful and cost-effective use of IT in healthcare. This report introduced a solution for interoperability problems in eHealth with the use of Structured Meta Knowledge (SMK) for the representation of medical concepts and demonstrated its utility for representing medical terminologies. "SMK is implemented as a large and complex computer program. It interprets the terminological facts in a particular model, and on that basis it provides answers to questions. (..) SMK is a

compositional system for the representation of medical concepts based on structures descriptions" (Nowlan, 1993). This first solution led to the creation of GALEN Representation and Integration Language (GRAIL), a concept modelling language originally designed for medical applications.

Paired with GRAIL, the GALEN Common Reference Model emerged and it persists to this day, although being no longer actively maintained. This "is a formal model within which concepts may be defined, described and automatically classified according to formal criteria for equivalence and subsumption" (Rogers *et al.*, 2001). To represent this reference model, GALEN created the GALEN CORE Model, with its ontology being designed to be re-usable and application independent, that is, interoperable. One of the main goals was to organize and index information efficiently and flexibly in order to share distinct compositional code systems using a single model, achieving a new level of interoperability not yet achieved by that time.

According to OpenGALEN (1999b), the GALEN Common Reference Model consists of four parts:

1. The High Level Ontology describes the sorts of concepts and the broad patterns by which they can be composed to produce more detailed concepts;
2. The Common Reference Model itself;
3. Detailed extensions required for specific applications;
4. The model of surgical procedures and other similar models, which define composite concepts made up of the parts from the Common Reference Model.

The model is a set of building blocks and constrains from which concepts can be composed. It separates the concept model from the module of use in clinical practice. The formal logical criteria of one does not correspond to the other, so it is necessary to add an additional pragmatic separately to transform the conceptual model to fit the practitioners' requirements (OpenGALEN, 1999a).

3.1.2 UMLS

The Unified Medical Language System (UMLS) is a repository of biomedical vocabularies and tools developed over more than 15 years by U.S. National Library of Medicine and integrates over 60 families of biomedical vocabularies to enable interoperability between computer systems. It is distributed on CD-ROM and by FTP using their browsers.

"The major component of the UMLS is the Metathesaurus, a repository of inter-related biomedical concepts. The two other knowledge sources in the UMLS are the Semantic Network, providing high-level categories used to categorize every Metathesaurus concept, and lexical resources including the SPECIALIST lexicon and programs for generating the lexical variants of biomedical terms." (Bodenreider, 2004). The organization is by concept, with synonymous terms being clustered together in a

concept, which is linked to other concepts. The symbolic relationships can be hierarchical or associative and there is also statistical relationships. Each concept is categorized by means of semantic types. Finally, UMLS also includes programs for distribution of the data as web services with a web interface (Bodenreider, 2004).

3.1.3 HL7 FHIR

HL7 is a not-for-profit international organization founded in 1987 that focusses on providing a comprehensive framework and American National Standards Institute accredited standards for exchange, integration, sharing and retrieval of eHealth information that supports clinical practice and the management, delivery and evaluation of health services. More than 1,600 members from over 50 countries support it and their vision is to create a world in which everyone can securely access and use the right health data when and where they need it by providing standards that empower global health data interoperability. 7 or "Level Seven" refers to the seventh level of the International Organization for Standardization (ISO) seven-layer communications model for Open Systems Interconnection (OSI) – the application level (Health Level Seven International, 2019b).

HL7 is the organization behind FHIR, an interoperability standard intended to facilitate the exchange of healthcare information between health professionals, researchers and patients. It consists of a content model and a specification for the exchange of the content in the form of real-time REST communications (Health Level Seven International, 2019f). HL7 produced a set of standards that evolved over the year, namely, HL7 Version 2 (HL7 V2) and its Reference Information Model (HL7 RIM), Version 3 (HL7 V3), which included the RIM from V2. Latter it produced the Clinical Document Architecture (HL7 CDA), and most recently, Fast Healthcare Interoperability Resources (HL7 FHIR), explained in the next section.

3.1.3.1 Fast Healthcare Interoperability Resources

HL7's FHIR, proposed in September 2011 as Resources for Healthcare (RFH), with a first release on 9 September of 2012 with a draft version, and a standard version for trial use in September 2013, is currently on version 4.0.1 (R4), released on October 10 2019, with mixed normative and STU specification (Health Level Seven International, 2019e). It combines major characteristics from HL7 V2, HL7 V3 and HL7 CDA while foreseeing the latest web standards and reclining heavily on implementability (Health Level Seven International, 2019g).

"FHIR solutions are built from a set of modular components called 'Resources'. These resources can easily be assembled into working systems that solve real world clinical

and administrative problems at a fraction of the price of existing alternatives." (Health Level Seven International, 2019g). These entities represent the business objects and define its data elements, constrains and relationships, characterizing the format in which data is exchanged. The modules are identified with a URL and can be represented in several formats, namely XML, JSON and RDF, and the most recent one Newline Delimited JSON (ND-JSON), albeit only XML and JSON are in a normative maturity level.

Besides the identifier, which is unique within the space of all resources of the same type on the same server, all the resources have common metadata, a human-readable XHTML summary, the set of defined data elements and an extensibility framework as Figure 3 shows with an example of a Patient resource. Optionally, resources can have more identities, a metadata field, a base language and a reference to some implicit rules. The identities are other business identifiers that allow identifying the same resource as it moves from server to server so it should be globally unique, like an OID, UUID or URI.

The resources usually have a security category and a maturity level associated. The maturity level is hereafter explained in the 3.1.3.4 Maturity Model section. Knowing the FHIR defined security categories is irrelevant since they are related to the sensibility of the data a certain resource holds and this project does not include any type of sensitive data, like a patient's personal information or his medical history. None of the resources used throughout the development of this project assume a security category other than anonymous. This category defines that resources with this level of security will more often be available for anonymous access, even though they hold important information and as such, their access should be submitted to an authentication process in order to protect their integrity in operations that allow changes to its hold information. Finally, the resources can be paginated, retrieving only a number of results per page.


```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

Figure 3 – Example of a Patient Resource

HL7 defines the following principles for appropriate use of the resources:

- Resources should have a clear boundary that matches one or more logical transaction scopes;
- Resources should differ from each other in meaning, not just in usage (different ways to use a resource should not result in different resources);
- Resources need to have a natural identity;
- Resources should be very common and used in many different business transactions;
- Resources should not be specific or detailed enough to preclude support for a wide range of business transactions;
- Resources should be mutually exclusive;
- Resources should use other resources, but they should be more than just compositions of other resources, with each one introducing a novel content;
- Resources should be organized into a logical framework based on the commonalty of the resource and what it links to;
- Resources should be large enough to provide meaningful context.

3.1.3.2 Extensibility

It is common for specific implementations to have valid requirements that are not part of the basic FHIR specification. If all of these requirements where considered in the structure definition it would make it very extensive and difficult to implement, so HL7 introduced extensibility as a fundamental part of the design of this specification. As such, it is expected that any additional valid requirement will be implemented as an extension. Extensions are child elements that represent additional information that is not part of the basic definition of a resource. Although any implementer can define and use extensions, to make them safe and manageable, following there is a set of requirements for their use and definition:

- The URL is mandatory and identifies a retrievable extension definition that defines the content and meaning;
- The URL shall be a canonical URL (absolute web address) of a structure definitions, with the exception of child extensions that can use a relative URL;
- The structure definition should be available to consumers of an instance, meaning that the URL should be resolvable or the extension should be published on an available registry;
- The extension shall have a value with a content or child extensions but not both;
- If it is not safe for an application to process the content of the resource ignoring the extension then a modifier extension should be used;
- Some resources do not allow extension on the root element, although extensions can be present elsewhere through the resource.

When an extension or set of extensions is used in a resource, we say we are profiling the resource.

3.1.3.3 Profiling

FHIR creates a common foundation on which several solutions are implemented with different purposes. Extensions exist in order to create margin for adaptation of resources to different contexts. In addition to this, other adaptations to specify resources used by an application, API features and which terminologies are used in specific elements can be implemented. What defines all of this adaptions is a profile and it is specified using a Capability Statement resource, which is a key part of the overall conformance framework, defining the set of features and rules for interacting with an application.

Profiling a resource is extending or restricting it using a Structure Definition resource. As such, a profile is a set of constrains on a resource represented as a Structure Definition of the kind “constrain”. The Capability Statement describes both Resource Profiles and Supported Profiles. While the first one describes the general features supported by the system for each kind of resource, the second describes information

handled or produced by the system on a per use case basis. The Capability Statement even describes which REST interactions are supported for each type of resource.

3.1.3.4 Maturity Model

Being a standard, FHIR is in constant change due to needful improvement overtime. This evolution needs to be predictable and manageable for implementers. HL7 delineates stages for its definitions so that the implementation community knows what to expect. These are descriptive terms that outline the level of stability and implementation readiness of the specification. The levels for the content are:

- **Draft:** Not considered to be complete enough or sufficiently reviewed to be safe for implementation;
- **Trial Use:** Well reviewed and is considered ready for use in production systems, albeit it has not yet seen widespread use in production across the full spectrum of environments it is intended to be used in;
- **Normative:** Reviewed and implemented in a wide variety of environments, being considered stable, reaching its final state;
- **Informative:** Provided for implementer assistance and does not restrains implementers;
- **Deprecated:** Outdated and may withdraw in a future version.

What it comes to trial use, there are maturity levels strongly related to stability, so the higher the maturity level, and the more restricted the artifact (resource or profile) is. HL7 defines its FHIR Maturity Model (FMM) as described below.

- **FMM 1:** Artifact published and producing no warnings on the current build process. The responsible work group has considered the artifact substantially complete and ready for implementation;
- **FMM 2:** Artifact has been on the previous level and has been tested and successfully support interoperability among at least three independently developed systems leveraging most of the scope using semi-realistic data and scenarios based on at least one of the declared scopes of the artifact;
- **FMM 3:** Artifact has been on the previous level and the work group who confirms its conformant with the Conformance Resource Quality Guidelines defined by HL7 has verified it. It has at least ten implementer comments from three distinct organizations resulting in at least one substantive change;
- **FMM 4:** Artifact has been on the previous level and it has been tested across its scope, published in a formal publication, and implemented in multiple prototype projects;
- **FMM 5:** Artifact has been on the previous level and it has been published in two formal publication release cycles and has been implemented in at least five independent production systems in more than one country;
- **Normative:** Artifact is now considered stable.

Where “tested across scope” means the FHIR Management Group has signed off on the list of “example contexts” defined for the artifact and that for each example context. The artifact has either been reviewed and approved by a domain expert for that scope area, mapped to an existing implemented scope-area-specific standard or tested in an implementation (Health Level Seven International, 2019k).

3.1.3.5 Terminology Module

The HL7’s FHIR Terminology Module is composed of a set of structures for representing and communicating coded, structured data in the FHIR core specification. These structures and their relationships are represented in Figure 4 and are used to provide functionality required for supporting the use of coded data in FHIR resources throughout the specification (Health Level Seven International, 2019h).

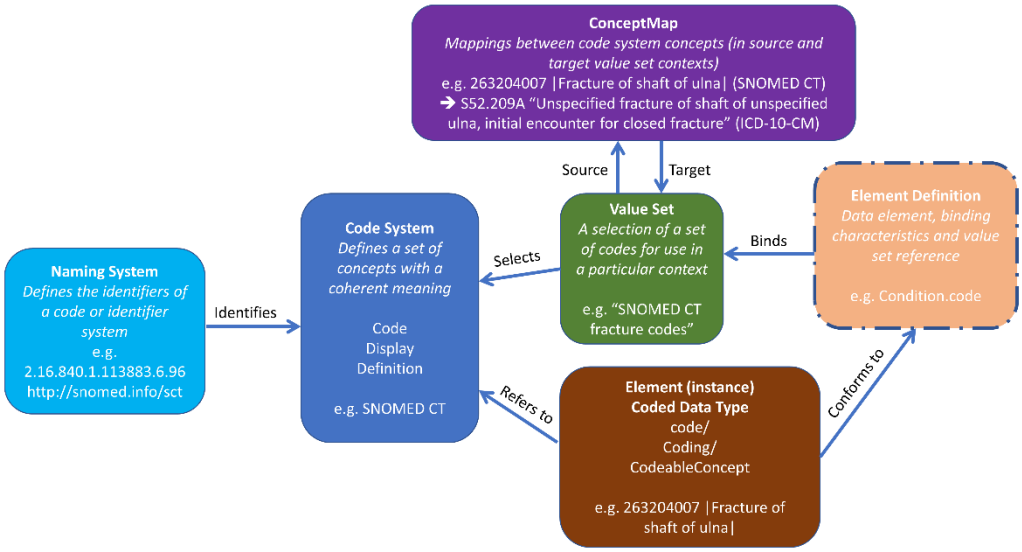


Figure 4 – Primary terminology-related structures and their relationships

Naming System, Code System, Concept Map and Value Set are the resources present in this module and, collectively, have the capabilities of a terminology server, although the first two are very alike in the type of information they can hold. While the Code System resource is on the normative level of the FMM, the Naming System is still on FMM1 so it has not yet been tested enough. The Value Set resource is also on a normative state, while the Concept Map is on the third level and the Terminology Capabilities is still on a draft state. The module is composed by one more resource – Terminology Capabilities – and some FHIR defined data types – code, coding and codeable concept (Health Level Seven International, 2019h).

“A terminology service is a service that lets healthcare applications make use of codes and value sets without having to become experts in the fine details of the Code System, Value Set and Concept Map resources, and the underlying code systems and

terminological principles" (Health Level Seven International, 2019i). In order to fully understand what the resources are capable of holding and which operations they support, each of them will be analyzed hereafter. Only the Code System, Value Set and Concept Map resources will have an analysis made since they are the ones standing on a more mature state and are the base resources necessary to implement a terminology service.

Code System

The Code System resource reached the normative level on the 4.0.0 version of the FHIR. It is designed to declare the existence of and describe a terminology or a terminology supplement and its properties. Optionally it can even fully or partially define its content, which is its concepts. The resource is not intended to support the maintaining of terminologies but to share its properties since it is not an efficient way to distribute its content. It allows to declare parenting relationships between concepts. These relationships can either be a "is-a", a "contains" or a "categorizes" relationship (Health Level Seven International, 2019c). The terminology it is not necessarily a clinical terminology, since there are other ones useful in eHealth and HER, like ISO 3166 that defines internationally recognized codes to be used when refereeing to countries and subdivisions. Hereupon, from now on terminologies will be referred as code systems.

The operations that the Code System resource supports are:

- **Lookup:** with a normative state on the maturity scale, it suits to get additional details about a concept;
- **Validate code:** it is intended to validate if a code belongs to a code system and it has reached the normative level;
- **Subsumes:** intended to validate a relationship between two given codes from the same code system and it has also reached the normative level;
- **Find matches:** the only operation that is still not normative, with a draft state, it works as a search within the content of a code system. Given a set of properties, it returns one or more possible matching codes.

Value Set

A Value Set resource specifies a set of codes drawn by one or more code systems, usually intended for use in a specific context. It may have a compose element or an expansion element, both or none of them. While the compose element is a definition of which codes are intended to be in the value set, the expansion is the list of codes that actually are in the value set under a given set of conditions (Health Level Seven International, 2019j). This is the most appropriate resource to share a code system

entire or partial content. Like the Code System resource, it has the capability to declare relationships between concepts.

The operations that the Value Set resource supports are:

- **Expand:** given or not a set of rules, it returns the concepts within the value set that fit the rules, or, in case of no given rules, returns all the concepts present in a value set. It has reached the normative state on the maturity scale;
- **Validate code:** like in the Code System resource, it is intended to validate if a code belongs to a value set and it has also reached the normative level;

Concept Map

The Concept Map resource has the capability to define mappings from a set of concepts from a code system to one or more concepts defined in other code systems. These mappings are one way, meaning that reverse mappings cannot be assumed unless they are defined. They are intended to be defined in the context of a particular business usage, which is usually defined by the specification of a source and target value set. A typical usage context is with mappings from a code system used in the documentation of an EHR to a code system used for data analysis or billing purposes (Health Level Seven International, 2019d).

The operations that the Concept Map resource supports are:

- **Translate:** given a set of parameters, it returns information about possible mappings to a given code, value set and/or code system. It is still on the FMM3 level;
- **Closure:** also still on the FMM3 level, this operation supports ongoing maintenance of client-side closure tables;

3.1.4 Overview

Although the OpenGALEN project had a good start and was highly supported by the community, the level of complexity it denoted since an early stage and its inadequacy to non-composite code systems was a huge barrier to its worldwide implementation.

As for UMLS, even though is a standard for combining multiple code systems in the same repository, using the same data structure and is capable of maintaining local terminologies, some known problems prevented it from gaining popularity worldwide. A study made by Geller *et al.* (2009) shows high levels of inconsistency in the hierarchical relationships. Its lack of implementations also resides on it supporting only terminology needs, while HL7 supports documentation and exchange of the documented data in every task associated with clinical practice and even administrative and billing tasks.

After understanding interoperability and its importance for health practice, research and management it becomes obvious the concern around the adoption of data codification and exchange standards on a global scale. Without a consensus endorsement from all healthcare entities it will never be possible to achieve international semantic interoperability. Hence, any healthcare organization has an interest in evolving towards a common collection of standards, and even more, HIT developer companies. What it comes to data exchange, reports show that every version of HL7's standards has shown to be a popular choice amongst institutions and software vendors.

According to Google trends, as it is visible in Figure 5, the global popularity on searches of the term "HL7" over the terms "OpenGALEN" and "UMLS" is vast, at least since there is data in 2004 and until the end of 2018. The numbers represent the search interest relative to the highest point on the chart for the given region and time. OpenGALEN shows on the bottom due to not existing enough data for the term.

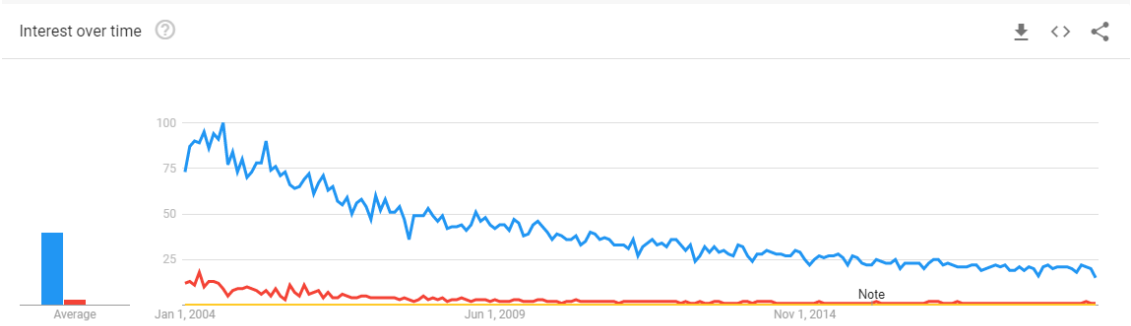


Figure 5 – Worldwide interest in search terms "HL7" (blue), "UMLS" (red) and "OpenGALEN" (yellow) from 2004 to 2018

According to Posnack and Barker, 2018, using U.S. Centers for Medicare & Medicaid Services (CMS) and U.S. Office of the National Coordinator for Health Information Technology (ONC) data, of the hospitals and clinicians that use certified products from the ten certified health IT developers with the largest market share in the U.S., 82% of the hospitals' and 69% of the clinicians' software is certified to any FHIR version. Figure 6 also demonstrates the popularity of certified-API's enabled with FHIR amongst U.S. hospitals.

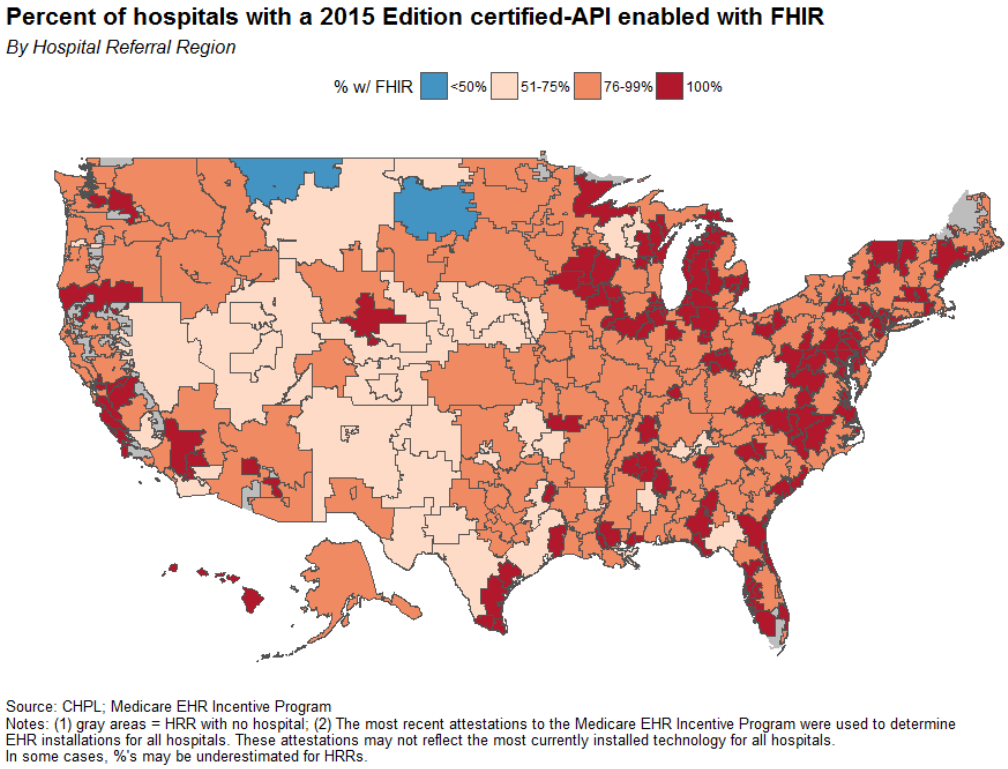


Figure 6 – Percent of U.S. hospitals with a 2015 Edition certified-API enabled with FHIR

By 2018, HL7 accounted with 771 HL7 CDA certified specialists, over 3 500 HL7 V2 certified specialists and already 80 HL7 FHIR certified specialists, and it had 35 countries with HL7 affiliates (Health Level Seven International, 2019a). According to their application registry, by the time of the writing of this document, there are 50 documented products that already support FHIR (Health Level Seven International, no date). It is also notable that there are syndicated applications of the standard for nationwide use, like the CSIRO Ontoserver. All of this shows that HL7 has been and will most likely continue to be the main choice of developers and governments.

3.2 FHIR implementations

Following the discussion of the previous section and since FHIR is the standard considered in the proposal of the project, a deeper study on its implementation is important. To develop any IT solution it is essential to know what already exists and can be used, avoiding the creation of unnecessarily duplicated implementations. Here are identified some existent products that are FHIR implementations, some of them in the form of a library and others of an actual software.

3.2.1 HAPI FHIR

HAPI FHIR is an open source library with a FHIR implementation in Java, known as the HAPI Project, with an Apache license v2, primarily supported by the University Health Network in Toronto, Ontario, Canada (Hussain, Langer and Kohli, 2018). It has features for implementing RESTful servers and clients while adopting FHIR objects. Although HAPI is an implementation of FHIR, it was first presented to the community for HL7 V2. At the time of the writing of this section, it has its last release on November 13 2019, with version 4.1.0 with codename Jitterbug that introduced features like support for ElasticSearch (James Agnew, 2019). The 4.2.0 version was already pre-released.

3.2.1.1 Applicability

According to its documentation (University Health Network, 2019), the library is designed to provide a flexible and composable way of adding FHIR capability to applications. Its main components are:

1. A base module with FHIR Java objects for multiple FHIR versions;
2. A built in parser and serializer that can be used to convert FHIR Java objects into serialized form (both XML and JSON), as well as the opposite operation, as Figure 7 demonstrates;
3. A validator containing FHIR profiles for multiple FHIR versions that validates resource instances;
4. Two alternatives of RESTful clients designed for easy setup, both represented in Figure 8. One of them allows building up FHIR REST invocations using an API. The other is more difficult to use albeit it can achieve faster compile-time;
5. Three alternatives for building RESTful servers. The first is a plain server featuring HTTP processing, the parser/serializer and FHIR REST semantics. This uses a standard Java Enterprise Edition (JEE) HTTP servlet for handling RESTful requests. The second is the same has the first one but using JAX-RS technology. The other alternative is a JPA server with a built in persistence layer, hence being a complete implementation of a FHIR server against a relational database. Unlike the first, it provides its own database schema and handles all storage and retrieval logic.

Thusly, the applications of the library are broad, from fully functional FHIR RESTful servers to develop products against (Figure 10), to more simple ones that can be implemented in an application to allow external clients to access it (Figure 9), and even

FHIR RESTful clients (Figure 8), more or less easy to implement. Finally, HAPI also provides a module for Android development.

Figure 7 – HAPI parser

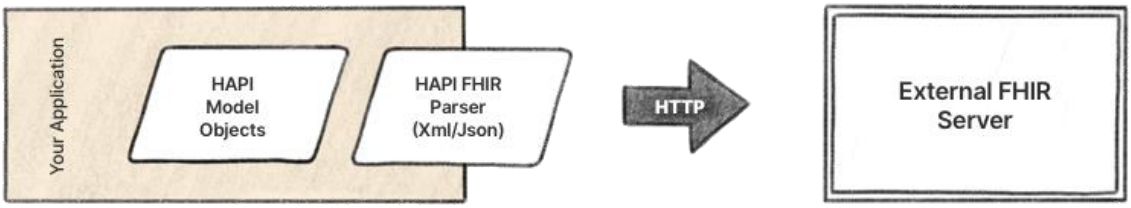


Figure 8 – HAPI FHIR RESTful Client

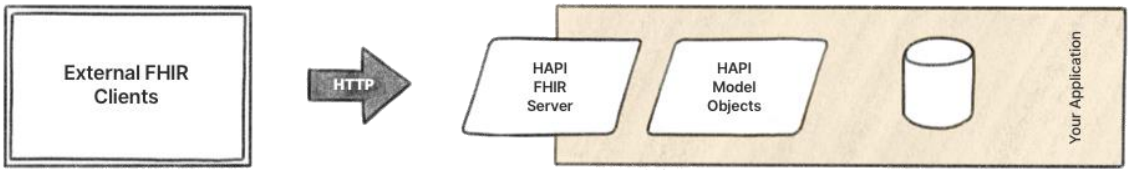


Figure 9 - HAPI Generic FHIR RESTful Server

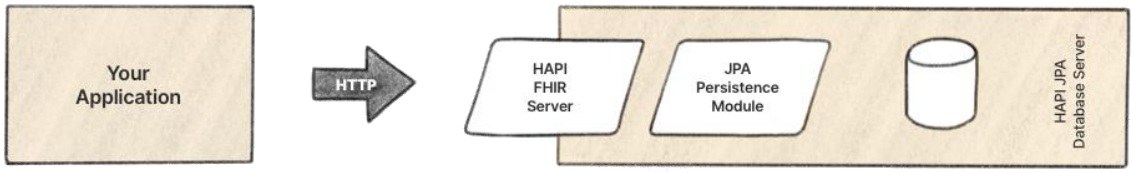


Figure 10 - HAPI JPA RESTful Server and Database

3.2.1.2 Supportability

HAPI supports all REST operations (Create, Read, Update, Delete), as well as every FHIR R4 resources and operations on every resource (for example Look up, Validate Code, Expand and Translate). It also provides support for automatic generation of a Capability Statement resource for FHIR servers. Using the generic server, it supports connection to any database technology, whether they are relational or not. In addition to this, it provides a set of interceptors for handling HTTP requests and responses, which include logging capabilities, response customizations, validation of requests. Is has support and documentation for integration of Elasticsearch and Apache Lucene search libraries, and finally, it also supports versioned resources (University Health Network, 2019).

3.2.1.3 Security

When it comes to security, there are a set of interceptors designated to this issue, encompassing support for Cross-Origin Resource Sharing (CORS), authorization, consent, and has it was mentioned previously, validation of requests including rejection of unsupported HTTP verbs. The consent interceptor allows validation on client requests to apply consent directives and create audit trail events (University Health Network, 2019). Finally, HAPI is prepared to work with HTTPS.

3.2.1.4 Implementation

When it comes to implementation, HAPI's effort to make it as simple as possible reflects on the library modularity, supportability, documentation, forum and code repository. The modularity allows developers to use a variety of components supporting a large variety of scenarios, ranging in easiness of implementation and amount of added code, while the forum and code repository allow developers and interested people to clarify doubts, including implementation related ones, and to report problems.

In scenarios in which it is crucial the addition of extensive code for full implementations not contemplated by the library, the easiness is also comprised since its extensively descriptive documentation and open-source characteristic facilitate the developer's job significantly. For example, the interceptor catalogue it provides comprises most scenarios, but the ones who are not covered should be created with no difficulty since the documentation explains in detail how to develop new interceptors. Most RESTful operations are also implemented with ease by the use of annotations that are well documented with useful examples. It even provides documentation for building the projects in their GitHub repository and for initiate the development of new applications using HAPI modules as dependencies. Lastly, it provides features for automatic upgrade when a new version is released.

3.2.2 Smile CDR

While HAPI is a library and API, Smile CDR, where CDR stands for Clinical Data Repository, is a clinical data repository commercially supported and powered by HAPI itself. It is a platform as a service (PaaS) and targets enterprises, research organizations, applications developers, consultants and integrators, offering them the option to seamlessly transition from community supported open source FHIR server to a commercially supported, enterprise-class FHIR repository (Simpatico Intelligent Systems Inc., 2020a). It versions go along with the HAPI ones, so it is currently on a version also named Jitterbug, launched on the same moment has the HAPI one.

3.2.2.1 Applicability

According to the description of their services (Simpatico Intelligent Systems Inc., 2020d), Smile CDR promises professional support at several levels.

- Assistance with solution's design and coding;
- Assistance with the business architecture creation;
- Training in FHIR, Smile CDR configuration, management and mapping.

Although it is ideally suited to support messaging-based input from patients, remote monitoring tools and healthcare providers, it does not take a perspective approach on FHIR, supporting the entire specification. It is designed to scale rapidly and can be a direct recipient of data from automated systems and a storage node for user applications (Simpatico Intelligent Systems Inc., 2020b). It provides a management interface and a statistics and system health console to help developers to manage their solution.

The prices combine fixed-cost tiers, additional volume discounts and pricing caps (Simpatico Intelligent Systems Inc., 2020c). They will vary, at an initial point, depending on the amount of modules and support necessary, since it is a customizable and modular software. With all of this taken into account, the applications of the platform are not necessarily broader than HAPI since it is built on top of it.

3.2.2.2 Supportability

Smile CDR integrates Apache Lucene search library supporting full text indexing with type-ahead searching and suggestions. As it was stated previously, it is configurable as a multi-node solution, with nodes being plugged and unplugged at any point without loss of data, and provides management services and interfaces. The platform also advertises fast configurability and installation.

Just like HAPI, supports all REST operations, FHIR R4 resources and specified operations, versioned resources and it is prepared to use a variety of relational and non-relational databases. It provides all of the HAPI functionality. Its modules are listed in Figure 11.

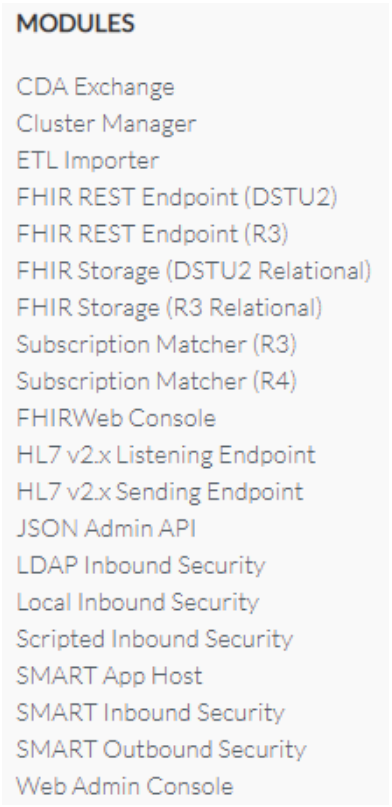


Figure 11 – Smile CDR Modules

3.2.2.3 Security

In terms of security, the service offers multiple solutions. It has alternatives for authentication, consent and auditing. For authentication, it supports user roles and permissions, and two-factor authentication using the Time-based One-time Password algorithm, which has popular implementations like Google Authenticator. It even has feature for a troubleshooting security log to help diagnose issues and outbound security modules to provide security to external applications (Simpatico Intelligent Systems Inc., 2020f).

3.2.2.4 Implementation

Although it provides extra features, the functionally these add is inexistent since most of them are related to the maintenance of Smile CDR itself. This does not mean that they do not provide any advantage, but that they are focused on relieving the developers' work, as the implementation is mainly based on configurations and not coding itself. It provides an extensive documentation for configuration, installation and general usage of the services, as well as commercial support, which largely enhances the ease of implementation but also de dependency on the platform support.

3.2.3 Firely

A group of software engineers, support engineers and FHIR consultants, based in Amsterdam, the Netherlands, maintain Firely (Firely, no date a). It is the open source FHIR implementation for Microsoft .NET, offering a library and several products developed on top of it, with both community and enterprise options (Firely, no date b). Just like Smile CDR, the company offers FHIR and Firely training and enterprise support including product implementation, FHIR profiling and Implementation Guide development, software co-development, and strategical and technical consulting (Firely, no date c).

3.2.3.1 Applicability

Firely product suit gathers solutions like the Firely API, an FHIR Mapper and Vonk, a professional FHIR server, all of them being open source. Just like Smile CDR, it offers a clinical data repository, which is cloud-based and served by Vonk. It can also work as a PaaS, offering modular tools that, with little coding and configuration, can easily serve a full workflow, making use of its commercial support.

Like HAPI FHIR, its library can be used to develop any web based FHIR solution, whether they are servers or clients. It also provides features for resource validation.

When it comes to servers, they offer a ready solution – Vonk FHIR Server – but the developers can chose to create their own, integrable with any database technology and data model.

3.2.3.2 Supportability

The library includes features for serialization and deserialization to XML and JSON, REST operations and FHIR R4 resources and operations. It can be integrated with any database technology, even though their Vonk server is only prepared to work with SQL Server, MongoDB and CosmosDB. This integration can be done using FHIR Facade, its FHIR mapper. It can be hooked up to analytics platforms like Apache Spark and Google's Big Query (Firely, no date b).

3.2.3.3 Security

In terms of security, the library can be extended with OAuth2 and OpenID Connect and supports integration with SMART on FHIR, analyzed next. It also supports HTTPS and it is extensible to any other authentication system.

3.2.3.4 Implementation

When it comes to implementations, both commercial support and documentation with helpful examples is provided. Its repository allows for bug reporting and discussion and includes some implementation examples, albeit not has much has HAPI FHIR.

The implementation can also depend mostly on the Firely team, just like Smile CDR, albeit the process is not mainly based on configurations and actual developers with expertise are needed.

3.2.4 SMART on FHIR

The SMART Health IT project, where SMART stands for Substitutable Medical Applications and Reusable Technologies, gathers a set of tools and solution for HIT including libraries, API's and test environments. It was initially proposed in 2009 (Mandl and Kohane, 2009) with an article on New England Journal of Medicine as an API to transform EHRs into platforms for substitutable iPhone-like apps. In the same year, the project received federal investment to be developed. The first SMART API was launched in 2010 has a draft and open source, the year when the first app adopted it, and only two years after, the first SMART app was put in production environment at Boston Children's Hospital (Computational Health Informatics Program, 2019a). SMART project's most popular invention is the SMART on FHIR API, used in Apple Health to connect the iPhone and Apple Watch app to healthcare systems and other products from Microsoft and Google (Substitutable Medical Applications and Reusable Technologies, 2019) SMART on FHIR is a PaaS that provides a set of libraries in many programming languages, with many instances.

3.2.4.1 Applicability

SMART on FHIR used to maintain a project of an open-source FHIR server in Groovy, a Java alternative that compiles in Java and integrates with Java libraries. Although the server is no longer maintained, it proves SMART on FHIR applicability when it comes to FHIR servers. Thus this is not a popularized application of SMART, since its focus has always been on mobile and web applications (or clients) and not on servers. Nevertheless, the data exchange between applications and servers is a main aspect of SMART on FHIR API, dealing with FHIR resources, authentication and consent.

Although it is not applicable to the entire development of web services, SMART provides a set of FHIR services for data exchange, and by providing a consistent way of authorization, it can be used in backend processing applications. Usually SMART is used in addition to any of the other FHIR implementations when web or mobile applications

need to be developed and integrated on larger systems that already contain FHIR servers.

3.2.4.2 Supportability

SMART tools include JavaScript, Python, R, Ruby, Swift, Java and .NET libraries, API's, application launchers for the web, sample data and sample apps (Computational Health Informatics Program, 2019b). Their tools support data export in JSON and XML in any type of FHIR based resource but do not handle REST or FHIR operations. The only feature server related is the data export from an existing FHIR server and data import onto native FHIR servers, using their FHIR services.

3.2.4.3 Security

SMART on FHIR has solutions integrated in the libraries provided to support powerful authentication using OAuth2 and OpenID Connect (Simpatico Intelligent Systems Inc., 2020e), being one of the most popular feature of SMART tools.

3.2.4.4 Implementation

When it comes to implementation, SMART on FHIR provides a set of examples in many applicability scenarios, including web, mobile and back-office apps. In addition to this, it provides documentation and usage guides on every library, facilitating the developers' job.

3.2.5 Comparison

When comparing all of the previously identified FHIR implementations, the factors considered are the support for REST web services, XML and JSON, FHIR R4 specification, the provision of security features and support for Oracle technologies, namely Java and Oracle Database. This is due to the project's scope is to develop REST web services based on the FHIR specification, and the preferred language for development is Java, albeit not being a restriction. Table 1 demonstrates this comparison.

Table 1 – Comparison between FHIR Implementations

Factors	HAPI FHIR	Smile CDR	Firely	SMART on FHIR
Support for REST web services	✓	✓	✓	✗
Specification for FHIR R4	✓	✓	✓	✗
Supports XML and JSON	✓	✓	✓	✓
Provides security features	✓	✓	✓	✓
Supports Oracle Technologies	✓	✓	✗	✓

While HAPI FHIR, Smile CDR and Firely fulfill all of the factors, SMART on FHIR does not provide support for REST web services and FHIR operations. Therefore, SMART on FHIR

is not applicable to the scope of the project so it will not be considered an option for the design and implementation phases of the solution.

In order to obtain the best value possible from the final solution, reducing the costs when feasible, the next section will focus on identifying and analyzing other solutions to develop REST services in Java, the preferred technology.

3.3 Solutions for REST services

Since Oracle technologies are the most explored skill in ALERT, with Java being the preferred programming language, this section analyses one more tool that can be used for the development of the FHIR RESTful services hereon proposed, so that when a decision needs to be made, the set of options provides viable options.

3.3.1 JAX-RS

JAX-RS, which stands for Java API for RESTful Web Services, is an API for RESTful Web Services development in Java SE (Standard Edition) and Java EE (Enterprise Edition). It utilizes annotations to create web services according to the REST architectural patterns, making the developer job easier. An Expert Group in 2007 and 2008 developed it, with its first draft released on June 2007 and the final release on March 2008. This expert group was led by two Oracle collaborators and composed of companies including Apache and Google (Oracle and/or its affiliates, no date).

3.3.1.1 Applicability

JAX-RS has many applications and counts with a large number of frameworks that implement the API. It can be used to build any application using a REST architecture, helping the developer creating HTTP requests and responses, which can translate on a web client as well as a server. It is a wide popular tool, and HAPI FHIR even provides a module to create servers using this tool.

3.3.1.2 Supportability

The tool provides support for handling HTTP requests and responses but does not support the FHIR specification.

3.3.1.3 Security

When it comes to security, the API provides support for authentication, authorization and encryption. The methods for using these features translate in configuration files,

interfaces, annotations or external libraries like OAuth (Oracle and/or its affiliates, 2020).

3.3.1.4 Implementation

When it comes to implementation, the tools provides extensive documentation, help groups and forums, and since is a highly popular tool, there are plenty experts within the community, spread throughout developer forums and prompt to aid and advice any beginner or not so experiment developer. Furthermore, a basic implementation of RESTful transactions is simple to develop since it is based on annotations, with few configurations or knowledge required. Figure 12 demonstrates simple example of the implementation of a response to a GET request.

```
package com.sun.jersey.samples.helloworld.resources;

import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;

// The Java class will be hosted at the URI path "/helloworld"
@Path("/helloworld")
public class HelloWorldResource {

    // The Java method will process HTTP GET requests
    @GET
    // The Java method will produce content identified by the MIME Media
    // type "text/plain"
    @Produces("text/plain")
    public String getClichedMessage() {
        // Return some cliched textual content
        return "Hello World";
    }
}
```

Figure 12 – Response to a GET request example with JAX-RS

3.4 Known FHIR Terminology Servers

In order to understand FHIR capabilities to the full and to have FHIR certified examples to look up to, here are analyzed some known Terminology Servers that respect the HL7's specification for the standard. The two first servers are dedicated to a specific code system and highlight the resources' power to the fullest with the specific terminology, while the third demonstrates a closer example to this project since it gathers many code systems.

3.4.1 LOINC FHIR Terminology Server

LOINC, which stands for Logical Observation Identifiers Names and Codes, is a code system for laboratory and clinical observations widely used on a global scale that developed his own terminology server with services defined by HL7's FHIR standard. The server is closed source and utilizes HAPI FHIR, an open source implementation of FHIR in Java. Its instance is to the date of the writing of this document live on a beta stage, built using software provided by Smile CDR, the company behind HAPI project (Regenstrief Institute, 2019a).

LOINC FHIR Server supports the FHIR *expand* and *validate code* operations in value sets, *translate* operation in concept maps and *lookup* operation in code systems. When it comes to mappings to other code systems, the server has a set of concept maps, which map LOINC terms to other terminologies (Regenstrief Institute, 2019b). LOINC is a compositional terminology that documents several properties about each concept. The server is capable of exposing almost all of these properties for each concept.

When testing the server was observed that there is some data repetition in the same resource. The following example (Figure 13) demonstrates the problem detected in November, 2019.

```
{
  "name": "property",
  "part": [
    {
      "name": "code",
      "valueCode": "rad-anatomic-location-imaging-focus"
    },
    {
      "name": "value",
      "valueCoding": {
        "system": "http://loinc.org",
        "code": "LP200011-7",
        "display": "Shoulder"
      }
    }
  ]
},
{
  "name": "property",
  "part": [
    {
      "name": "code",
      "valueCode": "rad-anatomic-location-imaging-focus"
    },
    {
      "name": "value",
      "valueCoding": {
        "system": "http://loinc.org",
        "code": "LP200011-7",
        "display": "Shoulder"
      }
    }
  ]
},
```


Figure 13 – Repeated data in LOINC FHIR server (November 2019)

3.4.2 Snowstorm Terminology Server

Snowstorm is an open source SNOMED CT Terminology Server developed by SNOMED International, which determines global standards for health terms and is owned by the International Health Terminology Standards Development Organization (IHTSDO). SNOMED CT stands for Systemized Nomenclature for Medicine Clinical Terms.

SNOMED International maintains a browser that provides a way to browse SNOMED CT and its currently on version 2.1. It has been implemented as part of development within the SNOMED International Open Tooling Framework, which is an open source framework not compliant with the FHIR specification. SNOMED International clarifies that even though the browser makes use of its own REST APIs, these are not to be used as part of production systems in health care settings (SNOMED International, 2019d).

Even though their browser does not make use of FHIR compliant REST web services, SNOMED International developed a set of open source web based services that partially respect the HL7's latest standard specification. According to its GitHub repository by SNOMED International, 2019b, the server is built on top of the HAPI project and Elasticsearch, an open source tool for search and analytics features, and focus on performance and scalability. It caters features like hosting multiple extensions alongside the International Edition, multi-lingual search and content retrieval, Expression Constraint Language (ECL) v1.3 compliance and a read-only FHIR API. "The Expression Constraint Language is a formal syntax for representing SNOMED CT expression constraints. Expression constraints are computable rules used to define a bounded sets of clinical meanings represented by either precoordinated or postcoordinated expressions." (SNOMED International, 2019a).

The repository where the source code for the browser is counts with more than 1 000 commits from 11 contributors, 4 of them belonging to SNOMED International and the other being external contributors. The server's instances have a set of intentionally defined value sets (explicit and implicit SNOMED CT support) and concept maps that document mappings to other code systems (SNOMED International, 2019c). It supports:

- The latest version of FHIR – R4;
- Solely read operations;
- FHIR *lookup* operation on code systems;
- FHIR *expand* operation on value sets;
- FHIR *translate* operation on concept maps.

It does not support any other FHIR defined operation or other code systems. SNOMED CT is a complex modular terminology with an intrinsic hierarchy and categorized relationships between concepts. The browser returns concepts using a Parameter resource and displays metadata, information about the category of the concept, its child concepts, its module and the language of the term displayed.

3.4.3 CSIRO Ontoserver

Commonwealth Scientific and Industrial Research Organisation (CSIRO) is an Australian Government corporate entity with the main purpose of carrying out scientific research and encourage or facilitate the application of the results of such research (CSIRO, 2019). It is responsible for the production and maintenance of Ontoserver, a syndicated FHIR Terminology Server for free use in Australia. It features the above mentioned code systems plus other known ones and even FHIR-based code systems. Similarly to Snowstorm, and according to SNOMED International (2019b), it has a set of implicit and

explicit SNOMED CT value sets, supporting multiple concurrent editions and versions. It also includes several implicit concept maps. Other than this, it supports:

- FHIR STU3 and R4;
- Create/Read/Update/Delete on every endpoint;
- All of the FHIR specified operations with the exception of *find matches* on code systems.

It does not support SNOMED CT post-coordinated expressions. The server was also built on top of HAPI FHIR.

3.4.4 Overview

It is simple to notice that the first two servers were developed by entities responsible for a specific terminology, which explains why they support only their own terminology. Both of them use the HAPI project, a free open source Java implementation of HL7's FHIR analyzed early in this document on section 3.2.1 HAPI FHIR. The two servers are extremely capable in their own terminology but do not support any other ones. Even so, they demonstrate that it is possible to use FHIR with the most complex terminologies in a machine interpretable way, making use of value sets to aggregate concepts within the same category or section of the terminology.

Finally, it can be stated that the Ontoserver is the more capable one of the three, with support for multiple terminologies including LOINC and SNOMED CT, even though not at the same level of detail and metadata, as each of the terminologies own serves. Just like the previous, it is built resorting to HAPI FHIR.

4 Analysis

In this chapter a detailed analysis of the problem, requirements and value is performed, with an engineering point of view. This chapter explains and justifies important decisions that influence later stages of the development of the solutions, mainly what it comes to implementation but also important for the design.

4.1 Problem

Following the considerations and guidelines found in the book The Thinker's Guide to Engineering Reasoning (Paul *et al.*, 2007), the current section demonstrates the problem which the project aims to solve in detail. The understanding of this problem is crucial in order to perceive the value of the proposed solution, as well as it is the foundation for its design and development, bringing to light the project's requirements.

4.1.1 Engineering purpose

At this point, it is known that the main purpose of the project is to develop an interoperability infrastructure for clinical terminologies based on the latest HL7's standard, to play the part of a FHIR terminology server. This server main function is to expose clinical terminologies used in the distinct ALERT contexts, whether that is to the ALERT® suite itself or to external products that interact with the ALERT® ones. Even though the project arises in the context of ALERT® suite, the customer is not only the enterprise itself, but any entity who aims to expose clinical terminologies using FHIR, like a governmental regulatory entity who defines which standards are to be used for medical practice within the governed community, or even other HIT vendors. In this

case, the solution would need previous adaptations for other sources of information besides the ALERT® data storage.

Although at this phase only a set of web based services are on the objectives of the project, they belong to a larger project, which is a terminology browser to be fed by the terminology server. An even larger project in which these services are aggregated to is a set of FHIR interfaces that expose all of the content present in the ALERT® suite, that is on ALERT® database and needs to be consumed by the ALERT® suite.

There market opportunity for the project is broad, since FHIR brings high value to interoperability, improving overall access to medical information with relevant meaning and making use of prevailing technologies. Since it aims to expose a large amount of distinct terminologies, its opportunity is increased by its capability to transmit not just the actual terminologies content, but also meaningful metadata, which allows clients to interpret the information correctly without the need to know the terminology structure or guidelines for usage.

4.1.2 Question at hand

The services value comes from the simplification of the implementation phase when installing any of the ALERT products, making use of purposely-designed value sets that aggregate data in a way that is consistent with the context in which the product is being implemented. That is, the state or country of the context, its language or dialect, its legal standard terminologies, the client's choice of other terminologies, the facility's module and market industry. The crossing of this aspects will designate which terminologies should be imported to the product on the moment of its implementation, as well as when updates are made to the terminologies in use.

To develop the product it is possible to adapt it from an existing solution since there are some open source products that partially answer the problem, some of them analyzed previously on the 3.2 - FHIR implementations section of the 3rd chapter of the present document. There are also a set of implementations of the FHIR standard in several programming languages that can be used to develop the services.

The solution's time-to-market is of high value since the FHIR is broadly adopted, being responsible for most of the existent interoperability in HIT. It also represents a necessary step to level up the interoperability in less evolved context in which ALERT® operates, creating pressure on governments and other vendors to adopt this communication standard as soon as possible. Only with an overall consensual usage by all practitioners of FHIR, regional and global semantic interoperability exists.

4.1.3 Points of view

Here are described some points of view worth noting prior to the design of the solution to the here discussed problem.

4.1.3.1 Engineering point of view

The infrastructure exposes regulated clinical terminologies, dealing with sensitive information that must retain its integrity. Therefore, in terms of design, it must consider the following components:

1. Data access;
2. Web Services;
3. Security;
4. Performance.

While the first one is responsible for accessing data storage for retrieval of clinical terminologies without losing or modifying it, the second exposes it in a way that it can be interpreted. The third one addresses security measures regarding data transactions in web services, ensuring reliable and authenticated requests, as well as other security concerns when accessing the database, and the last one relates to response time.

4.1.3.2 Other relevant points of view

There are many stakeholders and other interested parts involved or not in the project, who would benefit from the solution and whose points of view are worth noticing. These are:

1. ALERT;
2. ALERT Customers;
3. Regulatory entities;
4. HL7;
5. Other vendors.

It is on ALERT interests to adopt internationally approved standards, especially in communications with external software due to the minimization of interfaces.

ALERT Customers benefit from the use of efficient products that consider configurations according to their legal and organizational needs.

Regulatory entities aim to encourage the use of clinical terminologies and data exchange standards.

HL7 is the entity behind FHIR and it on its interest to monitor the standard's implementations in order to improve its specification.

Just like ALERT, any vendor's software that communicates with ALERT® or other software has on its interests the adoption of FHIR in data exchange, avoiding the development of new interfaces every time a new point of communication arises.

4.1.4 Assumptions

The following assumptions are made when designing the solution. These regard the associated risk, security and maturity issues, data availability and integrity, among others.

- The end user has internet access;
- The end user has access to a web client;
- The end user is already a registered user in ALERT® suite;
- The end user that logs in with some credentials is the legitimate user who detains these credentials;
- The end user has licenses, when necessary, for usage of the code systems he has access to;
- The database is always available with no downtime;
- The database contains information on which users have licenses for each code systems;
- The database contains all the code systems required by the users. This denoted the code system's general information, it's content and other meta-information about its structure and usage;
- The database contains complete, updated and righteous code systems;
- FHIR specification with FMM lower than Normative can suffer from updates in the future.

There are even a set of assumptions regarding ALERT's existent knowhow and the developer's capacities.

- The product represents the starting point of a larger product, so its development considers existent ALERT knowhow in order for it to be easily scalable within the company;
- Preferably, the developer is able to integrate ALERT's choices of technologies in every module of the product – Java and Structured Query Language (SQL).

Any change to these assumptions can affect the design of the solution, as well as the development strategy of the product.

4.1.5 Engineering information

In terms of information, most of the information relative to the data exchange is provided by HL7 and the chosen implementation of FHIR. Regarding the code systems information, their regulation entities are the ones who publish their versions. These publications should be the source of the information, guaranteeing the code system's integrity. As it is stated in the assumptions made before, albeit the source of information for the terminologies is an internally managed database, the stored information in there is always be compliant with the terminology's source. All terminology regulatory entities provide licenses for usage of the terminology content, which free or not are the most important guarantee of the information's source. This group of sources of information, in addition to relevant literature regarding the scientific area, should be enough for the development, therefore, all of these sources should be respected.

When it comes to the relevant literature, the acknowledgment and adoption of guidelines for designing web based applications, as well as security, performance and metadata communication guidelines is important to guarantee product's quality and requirements fulfilment, so information relative to this should also be studied.

4.1.6 Concepts and Implications

Whenever the term "ALERT context" is used throughout this document, it refers to a country/state/region representation in ALERT® system. All necessary theoretical concepts for the development of the solution are explained previously on chapter 2 Theoretical knowledge. Still, it is worth noting that code systems have different data models, with these differences mostly regarding their structure, which often competes with others. As such, a detailed understanding of this competing data models becomes crucial to achieve a transversal design without losing any important information. Not all of the code systems provide enough information about its structure or usage, so inferences on data models need to be as austere as possible. In addition to this, there is implications in new releases, translations or adoption of new code systems.

Every time a code system suffers an update resulting in a new version, the integrity of the previous versions must be retained, and mappings between both versions should be added every time they are provided by official entities, guaranteeing some after-

market sustainability. The design of the solution should take this in account in order to minimize the solution's susceptibility to change.

Translations also need to be dealt with extreme care, since most of them take significant time to get released, sometimes even with a difference of one version or more. This is, official translators can release a translation of a specific code system's version only after the next version in the main language is released. Along these lines, the date of release of the same version can despair a lot in different languages or dialects.

There are even implications related to the technology since HL7 can update its FHIR specification anytime, resulting in a change in the implementation of choice. Whichever implementation is selected to be used in the development of the web services, none of them fit in ALERT collaborators' knowhow, neither do the terminology module of the FHIR specification, so this can represent another implication to the maintenance and scalability of the project. Although the preferred technology is Java, if another programming language is considered the best option to develop the solution it can be adopted, as long has its use is justified with substantial advantages that could bring. Therefore, proper documentation of the solution is compelling to the success of the broader project in which this infrastructure will integrate. Due to these implications from now on, a code systems refers to a specific version of a code system, in a specific language.

4.2 Value Analysis

This section focusses on the process of Value Analysis (VA) of the product being developed, an organized approach to improve profitability of product applications that utilizes many different techniques to achieve this goal. The main principle, which is never compromised, is the one of offering value at the lowest optimal cost of production, allowing improvement ideas to be translated into commercial gains. Since it is a new product, a Value Engineering (VE) approach it is adopted, which applies the same principles and many of the VA techniques to pre-manufacturing stages (Rich and Holweg, 2000).

A product's value depends on its use and esteem values, in which the first is related to the product's functionality, while the second one relates to how to consumer views the product and how much subjective value he attributes to the product (Rich and Holweg, 2000).

In brief, VA (and VE) is a systematic, formal and organized process of analysis and evaluation, concerning the function of a product to meet the demands of a customer. Therefore, it is necessary to understand the purpose of the product, as well as to understand and formally manage the functional specification that assesses the level of

fit between the product and the value derived by the customer. The formal review process must result in design improvements that lowers production costs while maintaining its value (Rich and Holweg, 2000). The process consists of the five main stages, each being described in the beginning of its designated segment of this section.

4.2.1 Orientation

The orientation phase encompasses the creation of the VA team and the selection, preparation and presentation of the product (Rich and Holweg, 2000). For this phase, the New Concept Development (NCD) model was adopted. This model provides a common language and definition of the key components of the Fuzzy Front End (FFE) model (Koen *et al.*, 2002).

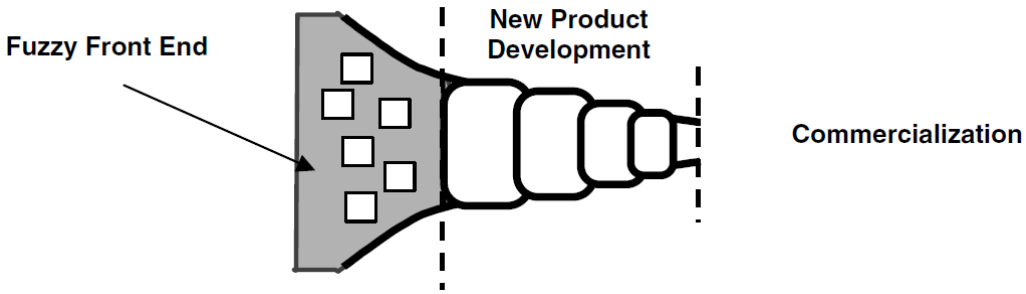


Figure 14 – The innovation process stages - Koen *et al.*, 2002

As Figure 14 (Koen *et al.*, 2002) shows, the innovation process consists of the following three stages:

- The **FFE**, which is an experimental phase of uncertainty when it comes to funding, commercialization dates and revenue expectations. Its main activity is research to optimize product's potential (Koen *et al.*, 2002).
- The **New Product Development** (NPD) that is the process and product development stage where the level of certainty rises with time (Koen *et al.*, 2002).
- The **Commercialization** stage, which comes after the product being developed and, like the name suggests, refers to its commercialization (Koen *et al.*, 2002).

The NCD model is composed of the engine, which represents senior and executive-level support, the elements that are powered by the engine, and the influencing factors, which are the base of the model on which the engine and the elements are placed on top of. The model is a relationship model, not a linear process (Koen *et al.*, 2002). As Figure 15 (Koen *et al.*, 2002) illustrates the model, describing the five elements it encompasses.

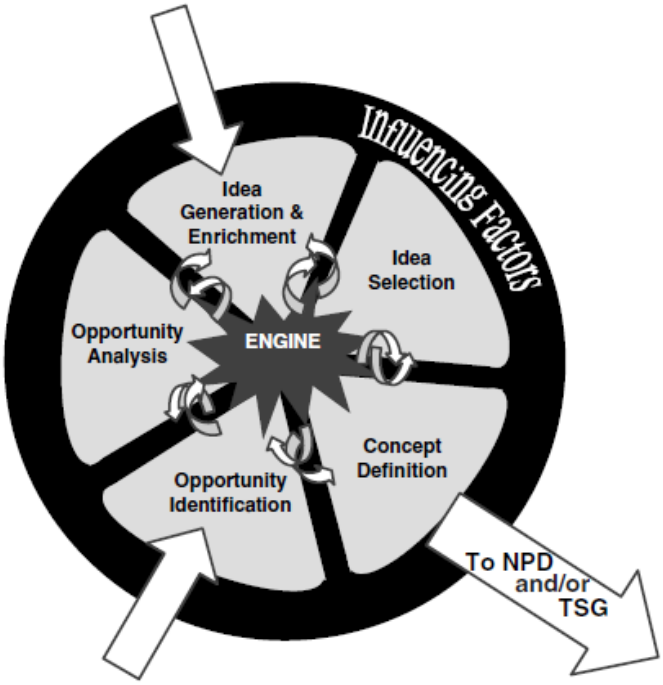


Figure 15 – New Concept Development model - Koen *et al.*, 2002

4.2.1.1 Influencing factors

"The factors are the corporation's organizational capabilities, customer and competitor influences, the outside world's influences, and the depth and strength of enabling sciences and technology" (Koen *et al.*, 2002). As such, the influencing factors of this product recline on one of the following categories:

1. ALERT's organizational capabilities;
2. Strength and maturity of the technology and concepts;
3. Regulatory Entities;
4. Legal influences;
5. Customer influences;
6. Other HIT vendors influences;
7. Other external health related influences.

"Organizational capabilities determine whether and how opportunities are identified and analyzed, how ideas are selected and generated, and how concepts and technologies are developed" (Koen *et al.*, 2002). Referent to ALERT's organizational capabilities, as it was stated in section 4.1.6 Concepts and Implications, ALERT knowhow is an important influencing factor to the product development, since the product is one

component and the starting point of a larger product, which will continue its development in the future. Therefore, the technologies within ALERT's capabilities, as well as its maturity levels, represent influencing factors. Besides these technologies, other technologies are needed so their maturity level is also taken into account. Furthermore, the code system's and FHIR specification's maturity can also produce influencing factors.

ALERT is a worldwide company operating in many distinct contexts. Within each ALERT context there are regulatory entities that define standards for clinical terminology codification for mandatory or advised use, within several medical specialties. Besides the mandatory ones, each ALERT customer can even select other code systems for usage within its operating software. As such, these ALERT contexts, their native language or dialect, their regulatory agencies and the actual ALERT customer private choice of code systems within their medical specialties all represent influencing factors too. The resulting set of code systems gathered from all ALERT customers, and each code system in specific, also consist of influencing factors.

Other factors relate to ALERT competitors, since HIT is an extremely competitive market. The project related to the adoption of a data exchange standard in clinical terminologies, a process that facilitates integration of ALERT® with other software that already makes use of the standard. The standard is highly adopted worldwide, as subsection 3.1.4 Overview of section 3.1 Clinical Terminology Codification Standards demonstrates, with many governments dictating its usage. As a result, if the trend moves towards another standard, the product's value is affected, so this trend amongst competitors is another influencing factor.

Finally, any legislation changes what it comes to clinical data exchange in health can also affect the solution's value, so legislation and governments are not less important influencing factors.

4.2.1.2 Engine

The NCD engine relates to leadership, culture and business strategy. To boost value through creation of new products can only occur if the pipeline for development of that product is consistent with the company's culture and business strategy, and if there is continuous senior management (Koen *et al.*, 2002).

The motivating engine of the project is the company's desire to be ahead of its competitors, in order to attract customers and improve profits. ALERT's mission is "to improve health and prolong life, achieve profitability to benefit society, and inspire others to excel like we do" (ALERT Life Sciences Computing, 2010). In this manner, it is within the company's ambition to adopt excellence procedures and protocols, to

pursue its other providers and inspire other to follow this step, so that a global interpretability of data can be achieved somewhere in the future.

4.2.1.3 Opportunity Identification

While on this stages, business and technological opportunities are considered so resources can be allocated and value can be obtained. These opportunities can be linked to marketplace needs that were not previously known. Later on, the opportunity is analyzed in order to verify if its worthiness (Koen *et al.*, 2002). This is the case of the here present scenario, where marketplace needs are changing, having already affected some focal areas outside the ALERT scope.

As it was stated previously, many governments have imposed the usage of FHIR in HIT. The standard is the worldwide-preferred standard for data exchange and top HIT companies are already using it. With ALERT® being in production in many different countries, even if none of these countries governments proposed or dictated the adoption of FHIR, it is more than evident that any software that adopt this specification prior to this instruction is highly ahead of its competitors who do not. It is also important to notice that FHIR popularity and interoperability needs in healthcare denote that this instruction will eventually come from every government/regulatory agency.

Beyond the growth that can come from the harmonization with top competitors, the pressure put on other HIT vendors to shape their products to the FHIR standard is also beneficial for ALERT. Rather than having to negotiate and delegate the creation of an interface of communication to one of the vendors involved in a point-to-point communication, ALERT can put to the side this responsibility since its software is already prepared to communicate using the worldwide-preferred standard. This brings not just value to ALERT® suite, but adds to the enterprise's competitive advantage.

This stated, it is noticeable that communications between distinct software in healthcare have been increasing over time, with the constant creation of new tools and persistent need to share this data between all of these new tools and the main operating software in a healthcare related entity. These tools are computer devices and mechanical and digital machine, which all communicate with each other creating Internet of Things (IoT) systems. A study (S. O'Dea, 2019) shows that in 2016 there were 0.87 million active IoT connections in healthcare in the European Union. The same study forecasts 6.05 million active IoT connection for the year 2022 and 10.34 million for 2025, all of this still within the European Union. This, alongside with the statistics shown in subsection 3.1.4 Overview of section 3.1 Clinical Terminology Codification Standards, emphasizes even more the market opportunity adjacent to the project, as well as the company need to innovate when it comes to data exchange.

The opportunity presented relates to inter-software communications. Aside from this, there are other scenarios in which a business opportunity can be seized. The first one is the deploy and installation process of ALERT®. Each ALERT® product is composed of the software, the content and a set of configurations. Nowadays, both the software and content are versioned, therefore, the content is dependent on the product version. This happens because every time a content is added or updated, changes may need to occur in the database model. So in order to avoid further problems, the content is only updated on the customer when a new version of the software is released, since ALERT® is clinical software that deals with sensitive data that cannot be lost in any case. Thereby, the new updated database model is prepared to receive new contents. Any opportunity to separate the content from the software, losing the dependency on the version, would be beneficial, since content upgrades could consist in a distinct process, which could be done anytime it shows necessary.

Furthermore, as it has already been mentioned, the project discussed here is the starting point of a larger product – a complete FHIR compliant layer between the software and the database, for handling the entire terminology content in ALERT®. Within this layer there is a terminology app, served by the services here documented. This app is an isolated product for terminology maintenance and can be sold on its own, whether that is to other HIT vendors, regulatory entities and governments. It can even grow to a fully isolated content management system.

4.2.1.4 Opportunities analysis

Now that the opportunities are identified from a business perspective, it is necessary to analyze them in order to conclude if by seizing them we can bring value to the end user. First, it is necessary to understand what value means, so later we can conclude on the solution's value.

According to (Nicola, Ferreira and Ferreira, 2012) value includes need, desire, interest, criteria, beliefs, attitudes and preferences, and its creation is key to any business. "Perceived value is the consumer's overall assessment of the utility of a product based on perception of what is received and what is given" (Zeithaml, 1988) so, as reported by (Ulaga and Eggert, 2006), the same product can represent different perceived value for distinct customers. Therefore, it is decisive to adopt a customer perspective so there is a better understanding of what can be the perceived value of the solution.

On this matter, (Woodall, 2003) suggests that value might be perceived as having both "use" and "exchange" meanings. Based on this notion, he presents a table of benefits and sacrifices for the customer (Figure 16), to be weighted from a customer point of view. This way we are able to verify the balance between value and cost, meaning what is received and what is given, and conclude about the solution worthiness.

BENEFITS		SACRIFICES
Attributes	**Outcomes**	
Perceived quality	Functional benefits	Price
Product quality	Utility	Market price
Quality	Use function	Monetary costs
Service quality	Aesthetic function	Financial
Technical quality	Operational benefits	Costs
Functional quality	Economy	Costs of use
Performance quality	Logistical benefits	Perceived costs
Service performance	Product benefits	Search costs
Service	Strategic benefits	Acquisition costs
Service support	Financial benefits	Opportunity costs
Special service aspects	Results for the customer	Delivery and installation costs
Additional services	Social benefits	Costs of repair
Core solution	Security	Training and maintenance costs
Customisation	Convenience	Non-monetary costs
Reliability	Enjoyment	Non-financial costs
Product characteristics	Appreciation from users	Relationship costs
Product attributes	Knowledge, humour	Psychological costs
Features	Self-expression	Time
Performance	Personal benefits	Human energy
	Association with social groups	Effort
	Affective arousal	

Figure 16 – Benefits and Sacrifices (Woodall, 2003)

In order to make a judgement about the solution's development worthiness, these benefits and sacrifices are put into account. All of the identified opportunities relate to different end users, some being the ALERT customers, others being ALERT itself as an enterprise and even other being possible customers for the terminology maintenance app, including developers, and deploy, support, consulting and financial teams, since all of these will benefit from the solution.

Table 2 demonstrates benefits and sacrifices for the company in three scopes: product, service and relationship between the company and the solution. Likewise, Table 2 shows that, when it comes to product, the company can benefit in terms of performance improvement and attained conformance with FHIR, a data exchange standard highly popular on HIT on a global level. Since with the introduction of the solution detaches the content from the software version, installation and update process become faster since they no longer are coupled with a content update process. For this, it needs to spend human resources and time on the development of both the web services here described, alongside with an FHIR client to access the server and an interface layer to integrate this content in ALERT® products. This interface, would need to me maintained through time, suffering changes anytime a new version of the

software requires it. This represents a sacrifice but not a new one, since the software is already changed at every version, what it comes to content related data consumption.

From a service perspective, ALERT benefits majorly in this area by saving its teams' time in negotiations relative to the interface development, and in the development itself. These negotiations, and sometimes development, happens anytime a new software is implemented in an ALERT customer, and the customer intends to have it communication with one or more ALERT® products. These convenience, logistical and strategical benefits are balanced by a company's sacrifice to maintain the solution so it can be up to date with FHIR.

Finally, the relationship between the enterprise and the solution is benefited by the financial savings resultant from the savings in human resources and time for developing new interfaces of communication with external vendors' software. The adoption of FHIR will also largely increase the company's value and improve its image within the market, due to its product being ready for cooperate with other vendors' solutions.

Table 3 does the same, however it is relative to ALERT customers and not the company itself. The last identified opportunity, relative to the content management system, will not be analyzed on this phase, since it involves a larger product that should be thought independently from ALERT®, and it deviates largely from the scope of this project. New clients like other HIT vendors, regulatory entities and governments will also not be considered since they do not represent the main focus of the project and are a matter to be analyzed after the conclusion of the here discussed solution.

Table 2 – Perceived Value for ALERT

Scope / Domain	Product	Service	Relationship
Benefits	• Performance improvement; • Conformance with popular international standards; • Reduce installation and update times;	• Time and resources savings with interface development discarding;	• Financial savings; • Increase company's value and improve its image in the market;
Sacrifices	• Human resources and time spent on solution's development; • Interface maintenance;	• Maintenance costs;	• Education of the deploy team;

Table 2 shows that, when it comes to product, the company can benefit in terms of performance improvement and attained conformance with FHIR, a data exchange standard highly popular on HIT on a global level. Since with the introduction of the

solution detaches the content from the software version, installation and update process become faster since they no longer are coupled with a content update process. For this, it needs to spend human resources and time on the development of both the web services here described, alongside with an FHIR client to access the server and an interface layer to integrate this content in ALERT® products. This interface, would need to me maintained through time, suffering changes anytime a new version of the software requires it. This represents a sacrifice but not a new one, since the software is already changed at every version, what it comes to content related data consumption.

From a service perspective, ALERT benefits majorly in this area by saving its teams' time in negotiations relative to the interface development, and in the development itself. These negotiations, and sometimes development, happens anytime a new software is implemented in an ALERT customer, and the customer intends to have it communication with one or more ALERT® products. These convenience, logistical and strategical benefits are balanced by a company's sacrifice to maintain the solution so it can be up to date with FHIR.

Finally, the relationship between the enterprise and the solution is benefited by the financial savings resultant from the savings in human resources and time for developing new interfaces of communication with external vendors' software. The adoption of FHIR will also largely increase the company's value and improve its image within the market, due to its product being ready for cooperate with other vendors' solutions.

Table 3 – Perceived Value for ALERT's customers

Scope / Domain	Product	Service	Relationship
Benefits	• Performance improvement; • Security features;	• Easy and independent import and update of clinical terminologies; • Content independent from version; • Ready compliance with other software;	• Faster installation and update; • Support; • Increase in client satisfaction;
Sacrifices	• Acquisition costs;	• Initial setup;	• Education of the IT team;

As seen in Table 3, ALERT's customers can benefit from the product because it is able to bring performance improvement to ALERT® products and their installation process. It can also bring new security features. Associated with these benefits is the initial sacrifice of acquiring access to the product, independently from ALERT® acquisition.

Service-wise, the solution brings to the customer the benefit of having their ALERT® terminology related content independent from the software, accessing terminology

updates without the need to acquire a new software version. Furthermore, this update process would be simple, easily done independently from the company's assistance, and this content would be ready for compatibility with external systems, as long as these are prepared to interpret FHIR. Since most top HIT vendors provide solutions FHIR ready, this could mean full interoperability within every department of an institution. The only sacrifice necessary would be the initial set up of the layer. When it comes to the relationship between ALERT and the customer, the previously mentioned benefits would result in an increase of the client satisfaction. Another benefit is that the installation and update process would become faster, since the content update would no longer be attached, and the costumer would still be getting access to a support team. Opposed to this, the client would need to send some time on education of the IT team, so it could be able to perform terminology updates on its own. This education process would be simple and quick, once a full FHIR terminology layer is complete.

4.2.1.5 Idea

The idea selection is an important element of the NCD since this decision should pursue achievement of the most business value possible. This decision will influence the future health and success of the business (Koen *et al.*, 2002). Thus, the idea behind this project is the adoption of an approved international standard – FHIR – for displaying clinical terminologies to external systems and ALERT® itself, so both ALERT and its clients can beneficiate from the outcomes the solution allows.

4.2.1.6 Concept Definition

Concept definition is the final element of the NCD and it provides the exit to NPD (Koen *et al.*, 2002). Hence, the concept can be defined has the development of RESTful web services to expose the code systems necessary for different scenarios, making use of FHIR specification. These web services should be safe and efficient.

4.2.2 Functional Identification and Analysis

Before proceeding to the functional identification, it is useful to identify the customer value chain in order to obtain a full picture of who the customer actually is. This representation is advantageous since the product aims to serve a set of distinct customers (The McGraw-Hill Companies, 2013), some internal for being company's collaborators, and some external ones, which are the company's customers. Analysis of the customer value chain is the first sept in Quality Function Deployment (QFD) technique to translate the customer requirements into a product design. QFD, being a VA protocol, aims to get higher quality products at a lower cost, achieving customer

design driven design and providing a tracking systems for future improvements (The McGraw-Hill Companies, 2013).

Figure 17 – Customer Value Chain

By analyzing the customer value chain in Figure 17, and when facing it with the opportunity analysis made previously, the target customers are identified. On one hand, there is ALERT development and operations teams, while on the other ALERT customers, which are hospital, clinics or other health institutions.

Next in the VA process comes the functional identification, where the most important functions of the product are systematically identified. These functions are demanded parts of a product or a product, and their esteem value, contributing to the ability of the product to be sold. The system's most important function is not always immediately clear to the team and can lead to a solution far from the initial scope. Therefore, the functions can be grouped and recorded using a Tree Diagram, which allows the structuration of ideas in a logical manner (Rich and Holweg, 2000). Following, the Tree Diagram in Figure 18 was constructed.

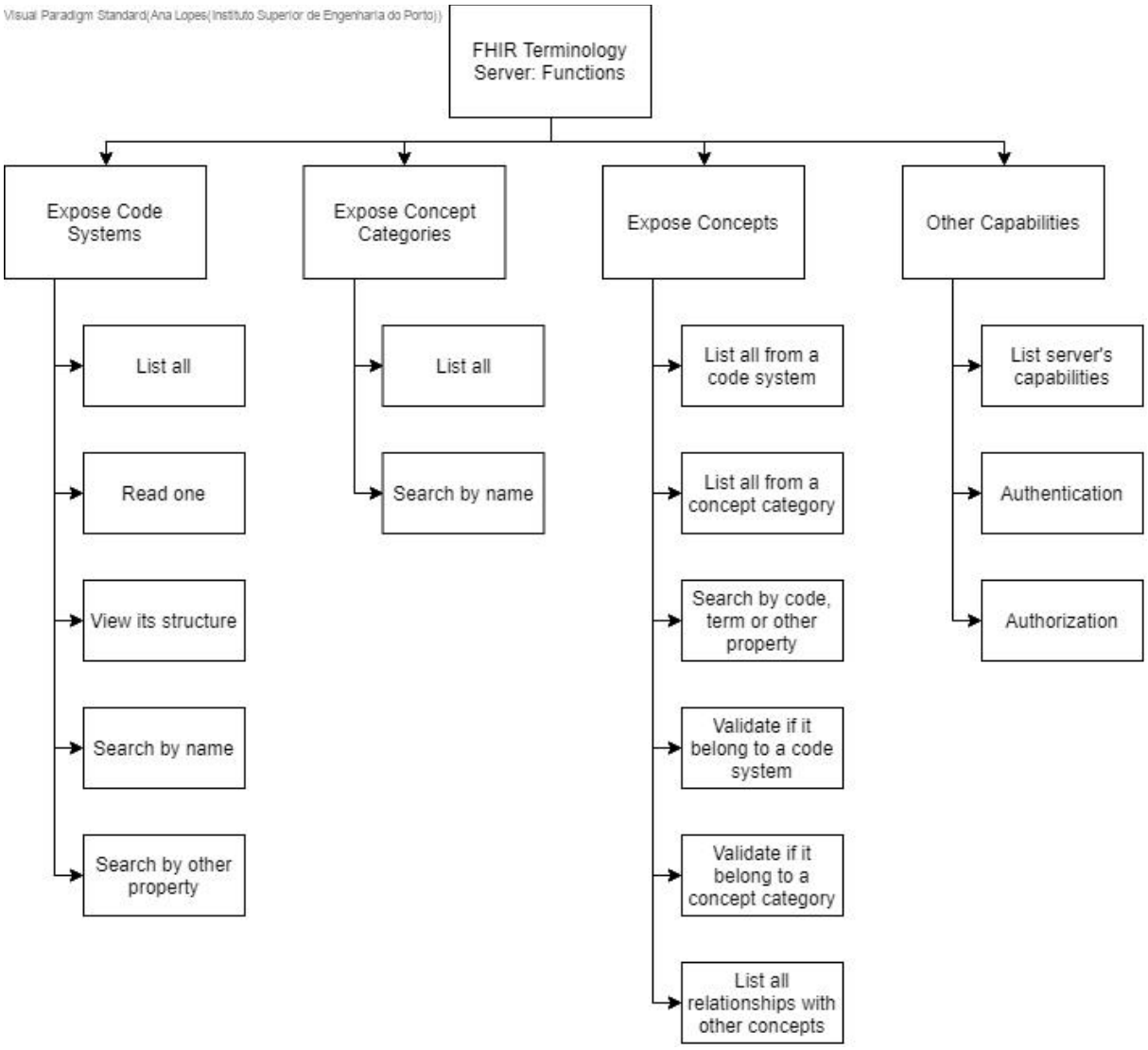


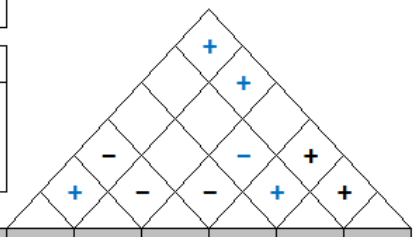
Figure 18 – Functionality Tree Diagram

The House of Quality technique is the most important result of QFD. It gathers desired attributes from customers and translates to engineering characteristics. The starting point is to capture the customer requirements, also known as quality requirements. (The McGraw-Hill Companies, 2003). For this stage, only ALERT collaborators where considered, encompassing the development and operations team. The collaborators were asked to answer a survey in which a set of quality requirements were present. These requirements derive from the initial proposal of the solution when compared with the requirements gathered in the functionality tree diagram, with the help of the project supervisor. Within this survey, the inquired had to attribute a weight to each requirement, with the total of the weights being 100. These requirements are displayed in the House of Quality (Figure 19), and described right underneath. Then are described the technology characteristics to be optimized so that customer satisfaction is assured.

Correlations	
Strong Positive	+
Positive	+
Negative	–
Strong Negative	–
No Correlation	

Relationships		
Strong	●	9
Moderate	○	3
Weak	▽	1

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



Row #	Relative Weight	Customer Importance	Maximum Relationship	Customer Requirements (Explicit and Implicit) / Functional Requirements	Authentication	Authorization	Response time	Data Integrity	Code System Adaptability	FHIR specification
				Column #	1	2	3	4	5	6
				Direction of Improvement	◇	◇	▼	◇	▲	◇
1	15%	15	9	Good performance			●			○
2	30%	30	9	Reliable information				●	●	○
3	15%	15	9	Safe information exchange	●	●		○	▽	
4	10%	10	9	Authorized information access	●	●				
5	30%	30	9	Integration of all necessary code systems				○	●	●
				Target	Supports	Supports	3 seconds	Supports	Supports	Supports
				Max Relationship	9	9	9	9	9	9
				Technical Importance Rating	225	225	135	405	555	405
				Relative Weight	12%	12%	7%	21%	28%	21%

Figure 19 – House of Quality

Quality requirements:

1. Good performance - 15;
2. Reliable information - 30;
3. Safe information exchange - 15;
4. Authorized information access - 10;
5. Integration of all necessary code systems - 30.

Functional characteristics:

1. Authentication;
2. Authorization;
3. Response time;
4. Data integrity;
5. Code system adaptability;
6. FHIR specification.

From the House of Quality (Figure 19) analysis it is possible to conclude that the most important technical requirement for the customer is 5. Code system adaptability (28%), which is the ability the system has of expose all necessary code systems, including their concepts with the fundamental properties, using FHIR specification. This does not focus on generic information on the code system but on the concepts themselves. Furthermore, it is visible that the response time is a technical characteristic that present a negative correlation with many other characteristics, meaning that it can represent a difficulty to the future, even though it demonstrated to be the less important requirement of the analyzed. Both data integrity and FHIR specification requirements present the same significance, coming in second on the importance rating.

Finally, a pairwise comparison is applied to analyze and rank the functions, where each function is compared for importance with each other and a score is given. After all pairs have been compared, the scores are added up for each function, and the one with the highest score is the most significant to the project (Rich and Holweg, 2000). Figure 20 is the pairwise comparison for the functions identified in the functionality tree diagram.

Pairwise Comparison: FHIR Terminology Server

Functions:	
	A: Expose Code Systems
	B: Expose Concept Categories
	C: Expose Concepts
	D: List Server's Capabilities
	E: Authentication and Authorization

		Score:	
A vs B	**B: 3**	A:	3
A vs C	**C: 3**	B:	8
A vs D	**A: 2**	C:	9
A vs E	**A: 1**	D:	0
B vs C	**C: 1**	E:	1
B vs D	**B: 2**		
B vs E	**B: 3**		
C vs D	**C: 2**		
C vs E	**C: 3**		
D vs E	**E: 1**		

Key to Points:		
	1	Little more important/better
	2	Significantly more important/better
	3	A lot more important

Figure 20 – Pairwise Comparison

The scores in the pairwise comparison demonstrate that the most important function of the server is to expose the concepts (9 points), which are the clinical terminologies themselves and their respective code. Next function on the importance hierarchy is to expose concept categories (8 points), this is, the categories by which the concepts can be grouped. Following is the exposal of the code systems (3 points), including their structure. Afterwards comes some security features, including authentication and authorization (1 point) and, finally, the listing of the server's capabilities (0 points).

Comparing these with the House of Quality results, it is conclusive that the most important function of the system is the ability to expose concepts from multiple code systems, using FHIR specification while maintaining the integrity of the information.

4.2.3 Alternatives

Now comes the VA stage where the team identifies a set of alternatives of possible tools that provide support for implementing the functionalities analyzed previously. On the case of the ongoing project, it is useful to choose technologies and tools that can aid the developer on constructing the solution, using already tested and worthy proven tools, avoiding him to loose time on features that are already implemented and for free use. Thus, the chosen technology will help bring value to the costumer and the developer.

The tools selected were already studied when it comes to their technical specification under chapter 3 - State of the Art on the sections 3.2 - FHIR implementations and 3.3 - Solutions for REST services. Since the use of Java as programming language is not a mandatory requirement, but an advice, the options include alternatives in this language and others. Therefore, the chosen tool will determine the programming language.

The alternatives are:

- HAPI FHIR;
- Smile CDR;
- Firely;
- JAX-RS.

4.2.4 Analysis and Evaluation

Coming up in the VA process is the analysis and choice of the tool that can bring the most value to the costumer, at the lowest cost. Since there are a set of meaningful requirements, the decision should have these taken into account when considering the alternatives. As such, different criteria will be used to make the decision. The Analytic Hierarchy Process (AHP) will be adopted, since is a multiple-criteria decision support method. These kind of methods allow to prioritize alternatives relatively to others, on a situation with multiple and conflicting criteria, aiming for a compromise solution (Buchanan and Gardiner, 2003).

AHP is a method that uses paired comparisons to derive priorities for criteria with respect to the goal. The first step is the structuring of the problem as a hierarchy. The first level related to the overall goal, the second one is where the criteria that contribute to the goal fall, and the third contains the candidate solutions (Saaty, 1990).

The criteria for this case is:

- Support for FHIR specification;

- Implementation – easiness of implementation of the tool and its fit to the project;
- Maintainability – related to the company's knowhow on either the programming language and the tool, which will influence easiness of maintenance and scalability of the project;
- Security – security of the REST transactions and authentication and authorization features;

The criteria and alternatives selected translate in the Hierarchical Decision Tree contained in Figure 21.

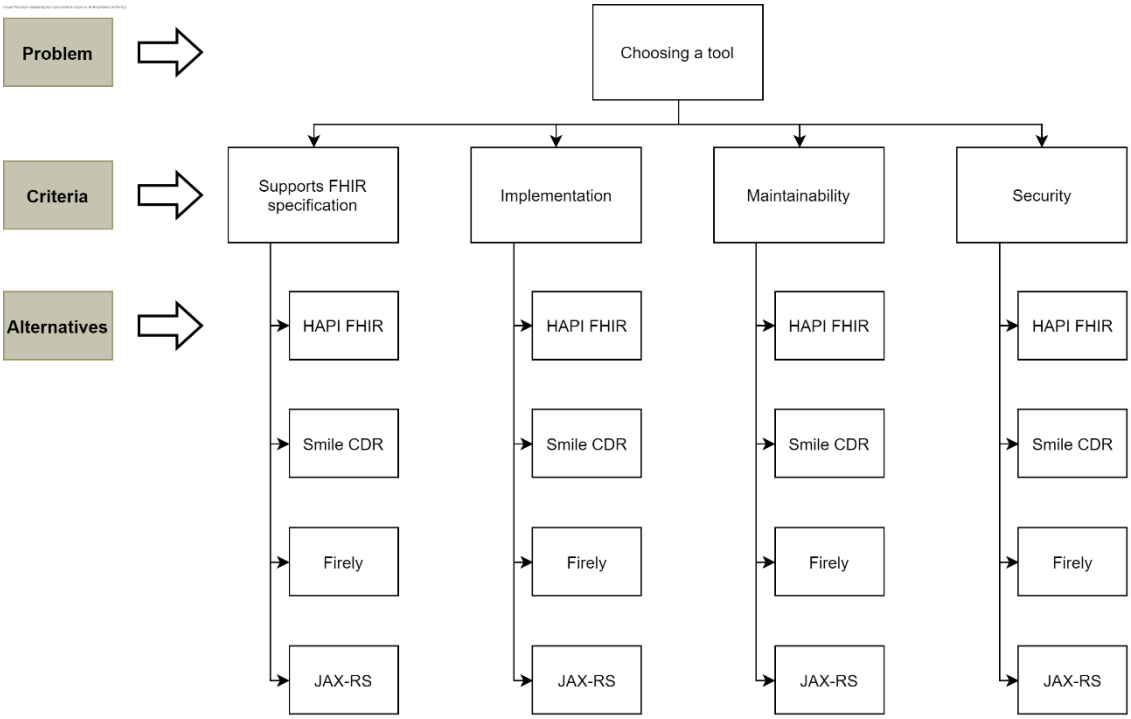


Figure 21 – Hierarchical Decision Tree

The second step in AHP is the elicitation of pairwise comparison judgments, arranging the elements in the second level of the hierarchical decision tree into a matrix. The scale to use making the judgments is represented in Table 4.

Table 4 – AHP fundamental scale (Saaty, 1990)

Importance on an absolute scale	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
3	Moderate importance of one over another	Experience and judgment lightly favor one activity over another
5	Essential or strong importance	Experience and judgment strongly favor one activity over another
7	Very strong importance	An activity is strongly favored and its dominance demonstrated in practice
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between the two adjacent judgments	When compromise is needed

Based on these values, the following comparisons and judgments on the importance of the criteria where performed.

Table 5 – Criteria matrix with column sum

	Supports FHIR specification	**Implementation**	**Maintainability**	**Security**
Supports FHIR specification	1	7	2	5
Implementation	$^1/_7$	1	$^1/_5$	$^1/_2$
Maintainability	$^1/_2$	5	1	3
Security	$^1/_5$	2	$^1/_3$	1
Sum	$\mathbf{^{129}/_{70}}$	**15**	$\mathbf{^{53}/_{15}}$	$\mathbf{^{19}/_2}$

In order to normalize the matrix, the sum is calculated for each column. This way all the criteria are in the same unit so that, later a mean is calculated, allowing for the criteria to be ordered by importance level. Next off is the normalized matrix of criteria, with the approximate level of relative priority.

Table 6 – Normalized criteria matrix with approximate relative priority

	Supports FHIR specification	Implementation	Maintainability	Security	Priority
Supports FHIR specification	$^{70}/_{129}$	$^{7}/_{15}$	$^{30}/_{53}$	$^{10}/_{19}$	**0.53**
Implementation	$^{10}/_{129}$	$^{1}/_{15}$	$^{3}/_{53}$	$^{1}/_{19}$	**0.06**
Maintainability	$^{35}/_{129}$	$^{1}/_{3}$	$^{15}/_{53}$	$^{6}/_{19}$	**0.30**
Security	$^{14}/_{129}$	$^{2}/_{15}$	$^{5}/_{53}$	$^{2}/_{19}$	**0.11**

According to the results, the main prioritized criterion is the support for FHIR specification with approximately 53% of priority, followed by the maintainability, registering an approximate level of importance of 30%. Next off comes the security with approximately 11% of the priority, and lastly, the implementation, which has an approximate level of priority of 6%. The eigenvector resultant is:

$$\begin{bmatrix} 0.53 \\ 0.06 \\ 0.30 \\ 0.11 \end{bmatrix}$$

Now in order to evaluate the consistency of the results, a consistency ratio (CR) of the priorities is calculated. This is calculated by comparison of a random index (RI) with the consistency index (CI), which is the negative average of the other roots of the characteristics polynomial of the matrix. If the result is significantly small (less than 10%), the estimative is accepted (Saaty, 1990). The IC (to the left) and CR (to the right) formula are:

$$IC = \frac{\lambda \max - n}{n - 1} \qquad CR = \frac{IC}{IR}$$

, where the λ_{max} is the eigenvalue and n is the number of criteria .The RI is obtained from the Random Consistency Index table presented below, based on the number of criteria (Nicola, 2019).

Table 7 – Random Consistency Index table (Saaty, 1987)

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

The eigenvalue is the mean of the values of the eigenvector resultant from the initial criteria matrix, divided by the correspondent values of the eigenvector resultant from the normalized matrix (above). The calculations are the following.

$$Criteria\ Matrix \times Eigenvector \cong \lambda max \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 7 & 2 & 5 \\ 1/7 & 1 & 1/5 & 1/2 \\ 1/2 & 5 & 1 & 3 \\ 1/5 & 2 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0.53 \\ 0.06 \\ 0.30 \\ 0.11 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.53 \\ 0.06 \\ 0.30 \\ 0.11 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 2.10 \\ 0.25 \\ 1.20 \\ 0.44 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.53 \\ 0.06 \\ 0.30 \\ 0.11 \end{bmatrix}$$

$$\Leftrightarrow \lambda max \cong 4.03$$

With the λ_{max} value is now possible to calculate the IC, and then the RC. Since there is 4 criterion, n = 4 and RI = 0.90.

$$IC = \frac{\lambda \max - n}{n - 1} \qquad CR = \frac{IC}{RC}$$

$$\Leftrightarrow IC = \frac{4.03 - 4}{4 - 1} \qquad \Leftrightarrow CR = \frac{0.01}{0.90}$$

$$\Leftrightarrow IC = \frac{0.03}{3} \qquad \Leftrightarrow CR = 0.01$$

$$\Leftrightarrow IC = 0.01$$

Since RC = 0.01 (1%) which is smaller than 0.1 (10%) so the priority values are consistent. Next step is the pairwise comparisons of the elements in the lowest level, with respect to how much better one is than another in satisfying each criterion in level 2 (Saaty, 1990).

4.2.4.1 Supports FHIR specification

When it comes to FHIR specification support, since all terminology related specification required for the scope of the solution is supported by all alternatives, the values from the fundamental scale that will be used are only 1 and 3. This way, either a tool supports FHIR specification, and will receive a level 3 of relative importance, or the tool does not support FHIR specification and will receive 1/3 in comparisons with the ones that do support.

Table 8 – Alternatives matrix for the FHIR Support criterion with column sum

Supports FHIR specification	HAPI FHIR	Smile CDR	Firely	JAX-RS
HAPI FHIR	1	1	1	3
Smile CDR	1	1	1	3
Firely	1	1	1	3
JAX-RS	$^1/_3$	$^1/_3$	$^1/_3$	1
Sum	$\mathbf{^{10}/_3}$	$\mathbf{^{10}/_3}$	$\mathbf{^{10}/_3}$	**10**

Following the steps made previously for the criteria, next, the normalized matrix is presented, with the approximate priority for each tool.

Table 9 – Normalized alternative matrix for the FHIR Support criterion with priority

Supports FHIR specification	HAPI FHIR	Smile CDR	Firely	JAX-RS	Priority
HAPI FHIR	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	**0.3**
Smile CDR	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	**0.3**
Firely	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	$^3/_{10}$	**0.3**
JAX-RS	$^1/_{10}$	$^1/_{10}$	$^1/_{10}$	$^1/_{10}$	**0.1**

In order to validate these priorities, the RC needs to be calculated. These calculations are demonstrated in Appendix A, as well as every RC calculations for each criterion. These values are presented later on, along with the priority vectors for the criteria.

4.2.4.2 Implementation

The implementation criterion respects to the easiness of implementation and fit to the project. The easiness of implementation is mostly related to the developer's knowhow on the tool's programming languages, albeit also including the easiness of implementation of the tool itself, independently from the programming language it was developed in. The fit to the projects means the suitableness of the tool and relates to the work involved in implementing the solution. Since this project is a thesis, developed

within the context of a master's degree, it would not be appropriate to adopt a solution based solely on configurations and enterprise support, since is not the scope of the project. These kind of solutions would also increase costs so they are not considered suitable to the project.

Table 10 - Alternatives matrix for the Implementation criterion with column sum

Implementation	HAPI FHIR	Smile CDR	Firely	JAX-RS
HAPI FHIR	1	5	3	1
Smile CDR	$^1/_5$	1	$^1/_3$	$^1/_5$
Firely	$^1/_3$	3	1	$^1/_3$
JAX-RS	1	5	3	1
Sum	$\mathbf{^{38}/_{15}}$	**14**	$\mathbf{^{22}/_3}$	$\mathbf{^{38}/_{15}}$

Table 11 - Normalized alternative matrix for the Implementation criterion with approximate priorities

Implementation	HAPI FHIR	Smile CDR	Firely	JAX-RS	Priority
HAPI FHIR	$^{15}/_{38}$	$^5/_{14}$	$^9/_{22}$	$^{15}/_{38}$	**0.39**
Smile CDR	$^3/_{38}$	$^1/_{14}$	$^1/_{22}$	$^3/_{38}$	**0.07**
Firely	$^5/_{38}$	$^3/_{14}$	$^3/_{22}$	$^5/_{38}$	**0.15**
JAX-RS	$^{15}/_{38}$	$^5/_{14}$	$^9/_{22}$	$^{15}/_{38}$	**0.39**

4.2.4.3 Maintainability

This criterion is respective to the company knowledge of programming languages. Since Java is the preferred one, the tools supporting Java will receive a better score. The knowledge on the tool is also considered, since HAPI FHIR was already used in other projects within the enterprise context. This criterion affects both maintainability and scalability of the solution, since the team can grow or another team can assume the responsibility.

Table 12 - Alternatives matrix for the Maintainability criterion with column sum

Maintainability	HAPI FHIR	Smile CDR	Firely	JAX-RS
HAPI FHIR	1	5	9	2
Smile CDR	$^1/_5$	1	4	$^1/_3$
Firely	$^1/_9$	$^1/_4$	1	$^1/_7$
JAX-RS	$^1/_2$	3	7	1
Sum	$\mathbf{^{163}/_{90}}$	$\mathbf{^{37}/_4}$	**21**	$\mathbf{^{73}/_{21}}$

Table 13 - Normalized alternative matrix for the Maintainability criterion with approximate priorities

Maintainability	HAPI FHIR	Smile CDR	Firely	JAX-RS	Priority
HAPI FHIR	$^{90}/_{163}$	$^{20}/_{37}$	$^3/_7$	$^{42}/_{73}$	**0.52**
Smile CDR	$^{18}/_{163}$	$^4/_{37}$	$^4/_{21}$	$^7/_{73}$	**0.13**
Firely	$^{10}/_{163}$	$^1/_{37}$	$^1/_{21}$	$^3/_{73}$	**0.04**
JAX-RS	$^{45}/_{163}$	$^{12}/_{37}$	$^1/_3$	$^{21}/_{73}$	**0.30**

4.2.4.4 Security

In term of security, both HAPI FHIR and Firely provide several alternatives for authentication and authorization, but Firely offers a more diverse spectrum. Smile CDR also supports a wide spectrum of authentication and authorization tools based on configurations and more powerful than the previous. JAX-RS does provide support for authentication but not as powerful as all of the other alternatives. This said, all of them are easily integrable with other external security libraries, and all of them provide support for safely handling HTTP requests.

Table 14 - Alternatives matrix for the Security criterion with column sum

Security	HAPI FHIR	Smile CDR	Firely	JAX-RS
HAPI FHIR	1	$^1/_5$	$^1/_2$	3
Smile CDR	5	1	3	7
Firely	2	$^1/_3$	1	5
JAX-RS	$^1/_3$	$^1/_7$	$^1/_5$	1
Sum	$\mathbf{^{25}/_3}$	$\mathbf{^{176}/_{105}}$	$\mathbf{^{47}/_{10}}$	**16**

Table 15 - Normalized alternative matrix for the Security criterion with approximate priorities

Security	HAPI FHIR	Smile CDR	Firely	JAX-RS	Priority
HAPI FHIR	$^3/_{25}$	$^{21}/_{176}$	$^5/_{47}$	$^3/_{16}$	**0.13**
Smile CDR	$^3/_5$	$^{105}/_{176}$	$^{30}/_{47}$	$^7/_{16}$	**0.57**
Firely	$^6/_{25}$	$^{35}/_{176}$	$^{10}/_{47}$	$^5/_{16}$	**0.24**
JAX-RS	$^1/_{25}$	$^{15}/_{176}$	$^2/_{47}$	$^1/_{16}$	**0.06**

4.2.4.5 Overview

The calculated values in Appendix A are summarized in Table 16, with the eigenvectors appearing transposed for readability purposes. All of the CRs are less than 0.1 so the priorities are considered consistent. These values translate into the decision tree in Figure 22.

Table 16 – Overview of the criteria eigenvectors and CR

Criterion	Transposed eigenvector	CR
Supports FHIR specification	$[0.30 \quad 0.30 \quad 0.30 \quad 0.10]$	0.00
Implementation	$[0.39 \quad 0.07 \quad 0.15 \quad 0.39]$	0.02
Maintainability	$[0.52 \quad 0.13 \quad 0.04 \quad 0.30]$	0.03
Security	$[0.13 \quad 0.57 \quad 0.24 \quad 0.06]$	0.02

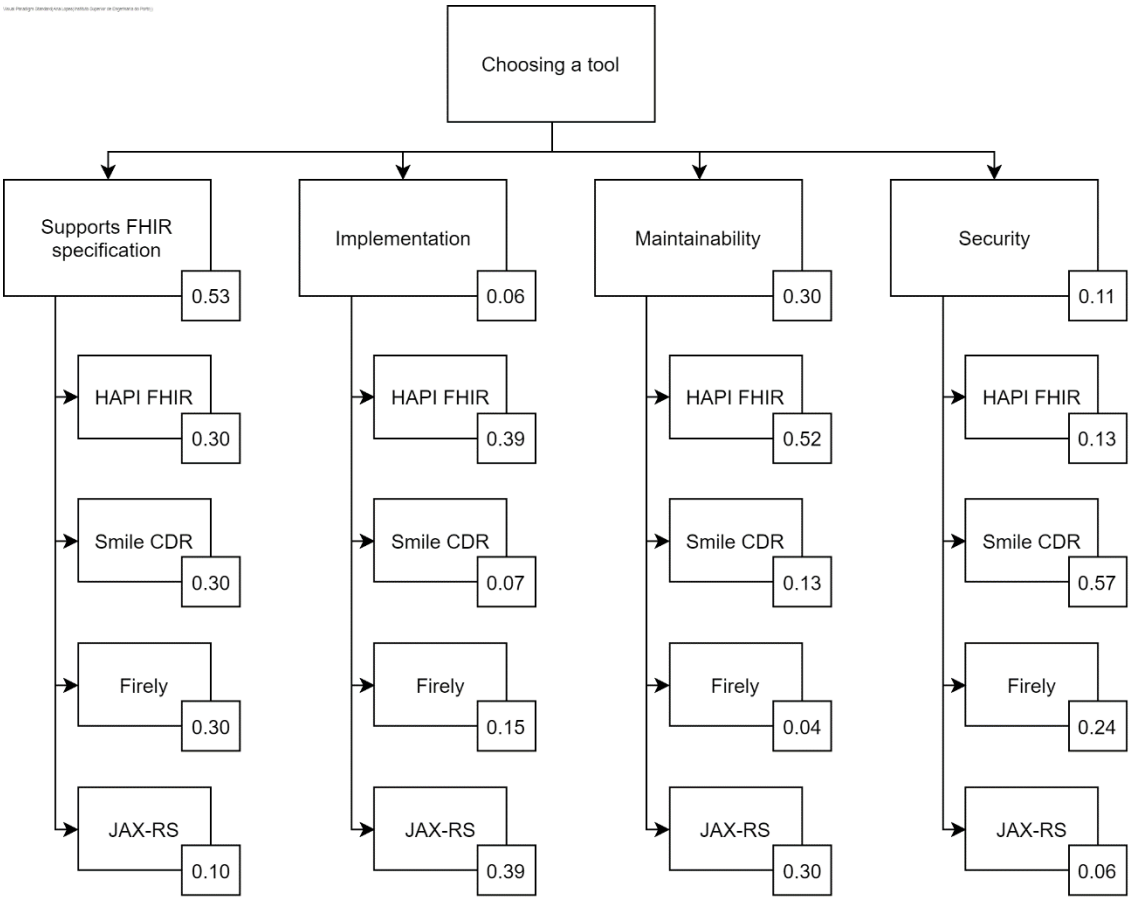


Figure 22 – Hierarchical decision tree with priority values

Now to calculate the most efficient tool is, a matrix with the eigenvalues for each tool in each criterion is created, and it is multiplied by the eigenvector that represents the criteria priorities. This is:

$$\begin{bmatrix} 0.30 & 0.39 & 0.52 & 0.13 \\ 0.30 & 0.07 & 0.13 & 0.57 \\ 0.30 & 0.15 & 0.04 & 0.24 \\ 0.10 & 0.39 & 0.30 & 0.06 \end{bmatrix} \times \begin{bmatrix} 0.53 \\ 0.06 \\ 0.30 \\ 0.11 \end{bmatrix} = \begin{bmatrix} \mathbf{0.35} \\ 0.26 \\ 0.21 \\ 0.17 \end{bmatrix}$$

In conclusion, the top performing tool on the AHP method was HAPI FHIR with 35% of priority, considering the criteria and relative importance level.

4.2.5 Implementation

The AHP allowed to choose a tool for implementing the FHIR Services for clinical terminologies based on multiple criteria. The conclusion of the process resulted in a qualification of HAPI FHIR as the most efficient alternative for developing the solution, so this is the tool on which the design is going to be based on, alongside with the FHIR specification itself which the tool implements.

5 Design

This chapter aims to document the design phase of the project, including the initial and elaboration stage of the RUP (Rational Software, 2011). Before starting to model the solution, it is important to analyze the business domain along with the FHIR specification, requirements and use cases, which will all condition the design. This chapter describes the use cases and requirements, a business model, a data model and, finally, some alternatives for the architectural design of the solution.

5.1 Requirements

This section reclines on the process of requirements gather, based on the previous choice and analysis, and follows the FURPS+ system. The system was first introduced in the book Software Metrics: Establishing a Company-wide Program (Grady and Caswell, 1987). According to the authors, the model involves establishing priorities and making quality attributes measurable. As the methodology for development of the solution proposes, a use-case driven development is adopted so, before any requirements it is important to recognize and analyze use cases.

5.1.1 Use Cases

According to the Rational Unified Process report (Rational Software, 2011), in the process of managing software requirements, the use of use cases and scenarios has proven to be an exemplary way to capture functional requirements, making the design of the solution more likely to fulfill the end user needs. Use case analysis was firstly

introduced in the book Object-Oriented Software Engineering - A Use Case Driven Approach (Jacobson, 1992) is a technique widely used in software engineering. Later on the authors released a new book entitled USE-CASE 2.0 The Guide to Succeeding with Use Cases (Jacobson, Spence and Bittner, 2011) where they describe a use-case model as a model of all the useful ways to use a system. Moreover, they state that the model is primarily made from actors and use cases, with diagrams illustrating the relationships between these two.

The use cases captured are represented in Figure 23. All of them require authentication and authorization, hence they being all connected to these use cases to represent this dependency.

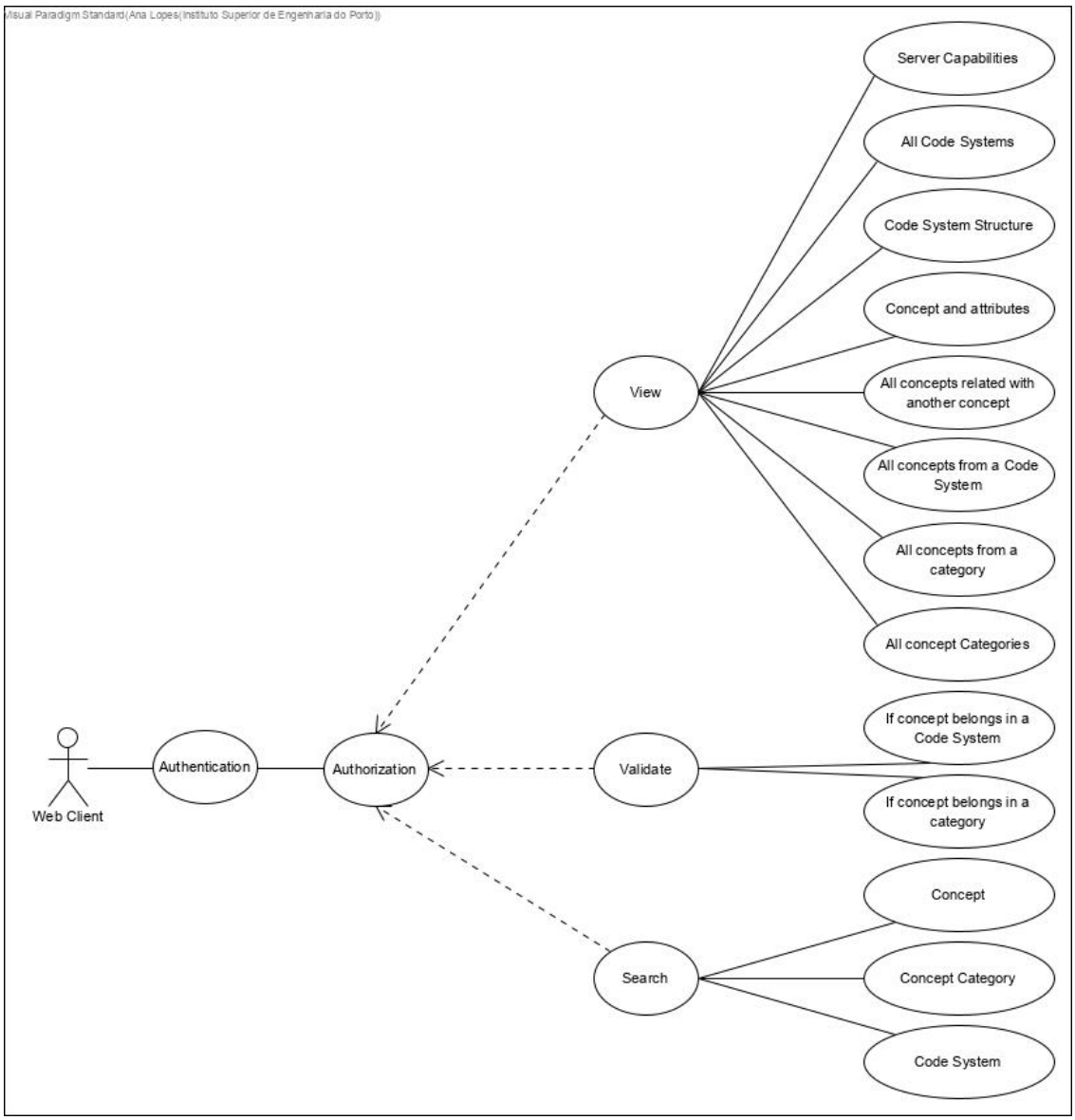


Figure 23 – Use Case Model

There is a single actor performing every use case since the infrastructure is a terminology server with no interface, so a web client is the mandatory agent to access it. This can be a client application like Postman or a web browser, although a web browser does not have the capability to display JSON without the addition of a plug-in.

5.1.2 Functional Requirements

According to the authors of the book previously mentioned (Grady and Caswell, 1987), under the functionality component falls requirements related to the feature set, the capabilities, the generality and security of the system.

Based on the previously demonstrated use case model, the following functional requirements for the infrastructure were captured:

1. Connects to a database that contains the terminologies basic information, their content, concept categories, languages and ALERT contexts;
2. List the server's capabilities;
3. List all code systems;
4. Display a code system's structure;
5. Validate if a concept belongs to a code system;
6. Search a code system by name;
7. Search a code system by other properties;
8. List all concepts from a code system;
9. List all concepts of the same category and code system;
10. Validate if a concept belongs to a category;
11. List all categories;
12. Search categories by name;
13. List relationships between concepts;
14. Search a concept by its code, term or other properties;
15. Pagination;
16. Authentication;
17. Authorization.

Not all of these requirements apply to every scenario due to code systems complexity and extensiveness. In code systems that are split into parts and whose entire content is

too extensive to be retrieved, or its retrieval in full is not useful in any scenario, the eighth requirement will not be applied. When it comes to the authentication, this process should be adapted, if not used entirely, from the already existing authentication process in ALERT®.

5.1.3 Non-functional Requirements

Here are the gathered requirements that do not relate with the functionality of the system. In each section, the segment of FURPS+ that section reclines on is explained.

5.1.3.1 Usability

In the usability component belong requirements relative to human factors, aesthetics, consistency and documentation of the system, in conformity with the authors Grady and Caswell (1987, p. 159). The following requirements were captured:

- Warns against unexpected and unresolved requests;
- Provides information for usage.

5.1.3.2 Reliability

Under the reliability factor fall requirements analogous to the frequency and severity of failure, recoverability, predictability, accuracy and mean time to failure of the system , in accordance with the authors Grady and Caswell (1987, p. 159). The requirements gathered in the factor are:

- Works continuously without failures;
- Ensures data integrity;
- Prevents against data repetition;
- Is accessed through a Web Client by HTTPS.

5.1.3.3 Performance

In conformity with the authors Grady and Caswell (1987, p. 159) under the performance segment fall requirements corresponding to the system's speed, efficiency, resource consumption, throughput and response time. In this segment, only a requirement was gather:

- Responds in 5s or less for listing operations;
- Responds in 2s or less for searching operations;
- Responds in 2s or less for validation operations;

- Responds in 2s or less for subsumes operation;
- Responds in 4s or less for expand operation.

The listing of a Value Set by I's id and the Lookup and find-matches operations are all included in the search requirement.

5.1.3.4 Supportability

According to the authors Grady and Caswell (1987, p. 159) in the supportability aspect are the requirements that correspond to testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, installability and localizability. The subsequent requirements were the ones identified within the supportability component:

- Provides a set of tests;
- Provides support for XML and JSON responses;
- The format of the responses is configurable;
- Is scalable.

5.1.3.5 Other Requirements

Relative to the + in FURPS+, in this section other requirements that can be relative to implementation, interface, physical needs and design constrains are listed. In this case, only the following implementation requirement was set.

- The development of the solution gives preference to technologies within ALERT's knowhow. If not, proper justification must be presented.

5.2 Business Model

By the analysis of the context of the problem and question at hand, the domain model in Figure 24 was constructed, which displays the business concepts and logic.

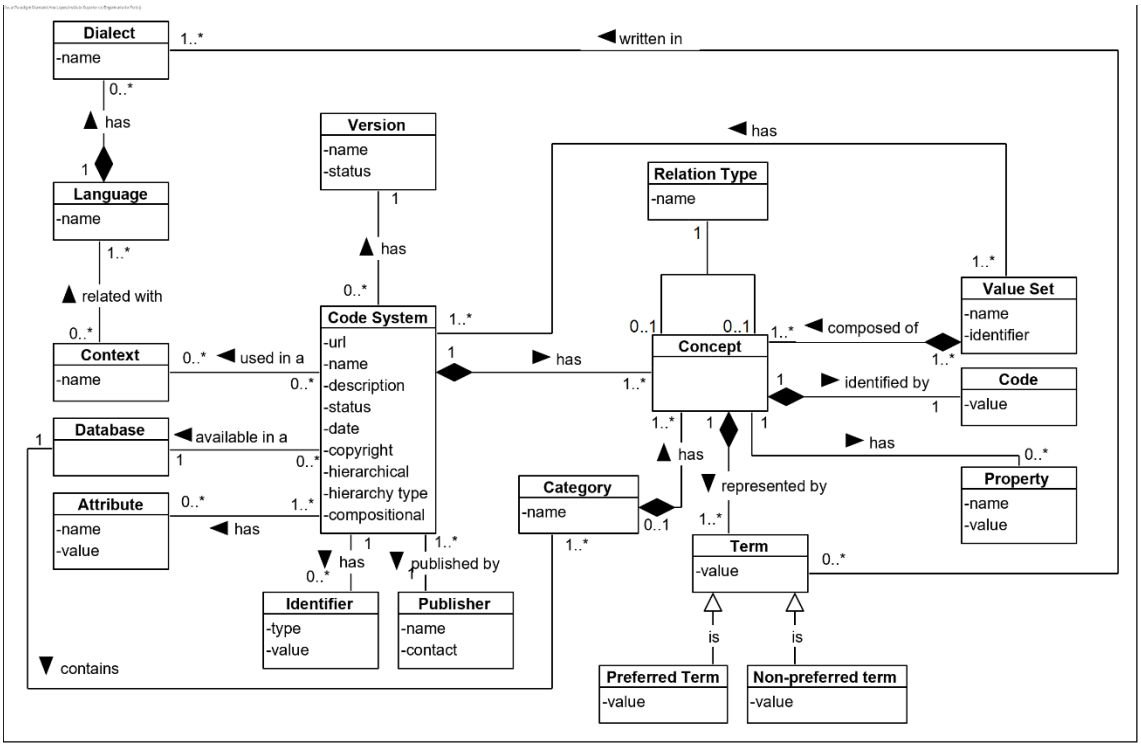


Figure 24 – Domain Model

There are several code systems that the system needs to handle that are published by a certain publisher under a specific license, displayed in the copyright attribute. They can be compositional and/or hierarchical, being that when they are hierarchical, the hierarchy sometimes can have an overall meaning that is applied in every concept relationship. These, along with some other attributes can be used to represent the code system's inner structure. They have more attributes like a name, a release date, a status, which describes whether the code system is in use or not, a version, and can even have more attributes. Their version has a name that identifies it and a status, just like the code system itself. These code systems are put to use within a context, being translated to the language/dialect of that context. Finally, they can have several identifiers using several identifying systems, and are stored in a database. This database contains a set of categories used to group concepts.

All of the code systems codify a set of concepts, each one encompassing a code and a preferred term. Depending on the code system, the concepts can have multiple distinct properties and alternative terms (non-preferred terms) for the same concept. The concepts can even have relationships between each other, and this relationship is described in the relation type whenever this is not a general relationship type applied to the entire code system. Lastly, the value sets display a group of terms, from one or more code systems, and are based on a category, a code system segment or en bloc.

5.3 Architectural Design

The design of the clinical terminology infrastructure aims to follow software and web development good practices, making use of existing patterns and principles for object-oriented design. The architectural design represented in Figure 25 displays three essential components of the infrastructure: the web client, the terminology server and the database.

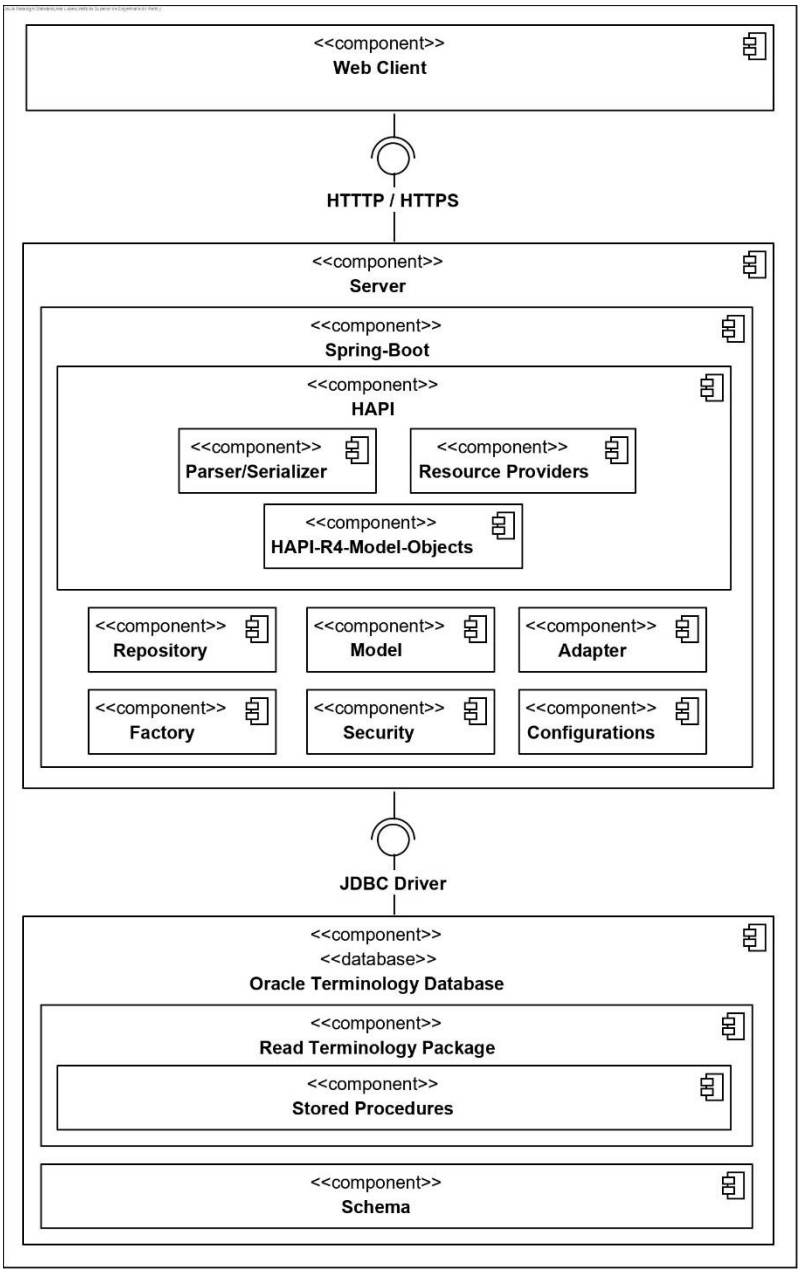


Figure 25 - FHIR Terminology Infrastructure Component diagram

The server is built using Spring Boot, a framework to facilitate the creation of stand-alone applications, handling all the necessary configurations from the process of developing and deploying an application with no code generation. This choice follows indications by ALERT. Although it can handle configurations, the developer can still personalize whatever he has the need. It even provides production-ready features such as metrics (Pivotal Software, 2020).

From HAPI, there is a component to handle the HTTP requests for each endpoint (resource provider), one that contains the Java objects with a resource compliant structure, and a parser/serializer to generate the XML and JSON responses. Other components that constitute the HAPI base and server, required parts for the project, are omitted in the diagram for readability purposes. These include a component responsible for generating an automatic capability statement, based on the existing resource providers and the requests they are ready to handle.

Beyond HAPI, the server contains a factory component responsible for creating all necessary object instances besides from the HAPI ones. This factory is a singleton, as well as the repository, since only one instance of them is necessary. The requests are mapped into a provider, which creates an adapter to perform the operations. The adapter is responsible for knowing the process of transforming the information, and accesses the repository in order to retrieve that data. There are also the model components, which store specific data for the adapter. The repository component calls the JDBC Driver to create a database connection based on configurations from the configurations component, and therefore, consume data. This data then reaches the adapter, which populates the HAPI models so that a response is generated. The data can arrive to the adapter in two alternative formats. Data Transfer Objects (DTOs) can be used, or lists of generic objects containing only the attributes returned by the stored procedure. While the first benefits from the use of well-proven software patterns, the second alternative can be more cost efficient in terms of performance and memory allocation. One of these alternatives will be adopted in the final design, however it is necessary to test the premise before making a decision. Before the retrieval of the data the security component verifies if the user is authenticated and has authorization to access the required data.

On top of the database schema there is a package with a set of stored procedures to get the terminology data. This database is copy from the terminology schema in an ALERT® database, which was built throughout the years and contains numerous code systems that are stored in the database's own data model. There is one or more procedures to handle each HTTP GET request. These are called using the JDBC Driver by the repository component.

5.3.1 Design Patterns

Several design patterns are mentioned above so this section goal is to clarify their description and purpose. These are included in Gang of Four (GoF) patterns or Domain Driven Design (DDD) patterns, which both are based in General Responsibility Assignment Software Patterns (GRASP). GRASP are a set of principles and guidelines for assigning responsibility to objects in Object Oriented Design (OOD), while GoF implements these principles in actual design solutions. DDD is a design approach with the focus on reflecting the domain model in a very literal way. It usually requires software development tools that support a modeling paradigm, such as object-oriented (Evans, 2015, p. 6). From these set of patterns and principles the following are or can be included in the solution's design:

- **Factory** – not a GoF pattern but often considered a simplification of the GoF Abstract Factory pattern, being a DDD pattern. It intends to separate the responsibility of complex creation into cohesive helper objects, hiding the logic creation. Factory objects are often accessed via the Singleton pattern (Larman, 2004).
- **Singleton** – a creational GoF pattern that solves the problem of factory creation and access (Larman, 2004). Its purpose is to ensure a class only has one instance, while allowing other classes to get access to it, providing a global point of access (Gamma *et al.*, 1994, p. 144; Stelting and Maassen, 2002, p. 31).
- **Repository** – a DDD pattern which function is to set up access to a persistence layer providing methods for the CRUD operations, returning fully instantiated objects or collections of objects (Evans, 2015, p. 17).
- **Adapter** – also a structural GoF pattern which function is to act as an intermediary between two classes, converting the interface of one class into another. It creates compatibility between classes that otherwise could not work together (Gamma *et al.*, 1994, p. 157; Stelting and Maassen, 2002, p. 98).
- **Data Transfer Object** – not a GoF or DDD pattern. It is used to define an object that carries data between processes in order to reduce the number of method calls (Fowler, 2004, p. 401).

It is important to notice that, although explained in the current section, the adapter component present in the design is not a real implementation of the adapter design pattern if DTO's are not used. If this is the chosen alternative, the adapter will transform data in order for it to be compatible to be populated onto the HAPI models, but from data coming in a vector and not an instance of an object.

5.3.2 Requests

Following the explanation of the architectural design elements and their interactions, both Figure 26 and Figure 27 display a generic GET Request to some endpoint of the server, with user authentication and other parameters, depending on the operation. Figure 26 demonstrates the workflow for an architectural design without the use of DTOs, while Figure 27 includes this component. The authentication and authorization process are not exhibited with extensive granularity since these features are discussed in the adjacent section. The process of handling the requests is also omitted, as this is a process mediated by HAPI components.

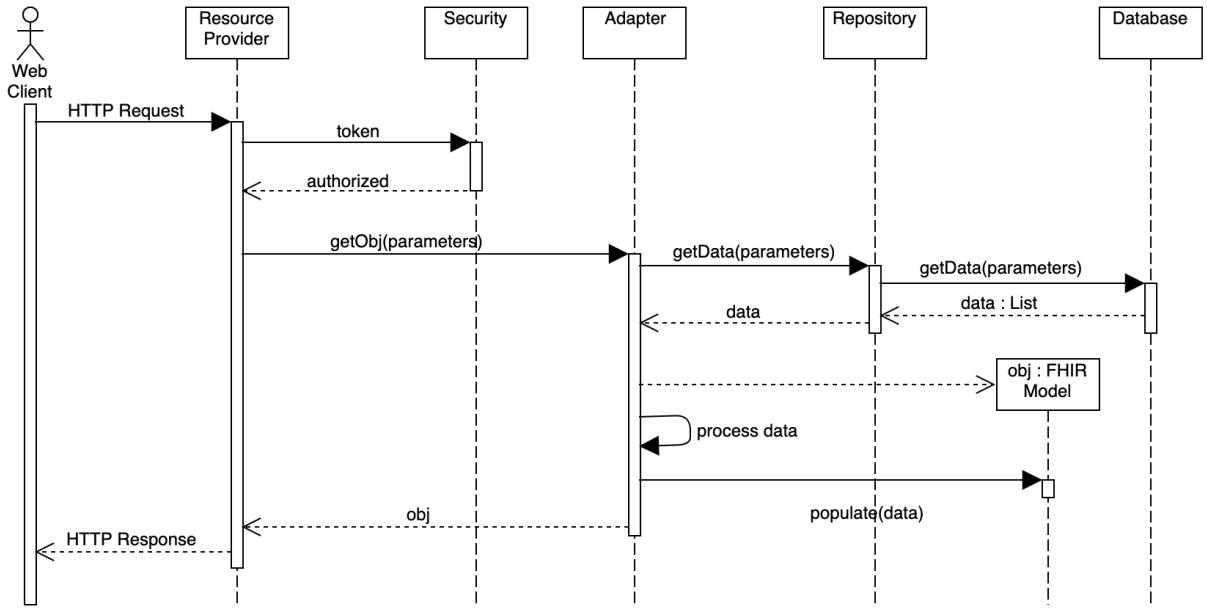


Figure 26 – Generic GET Request without DTO

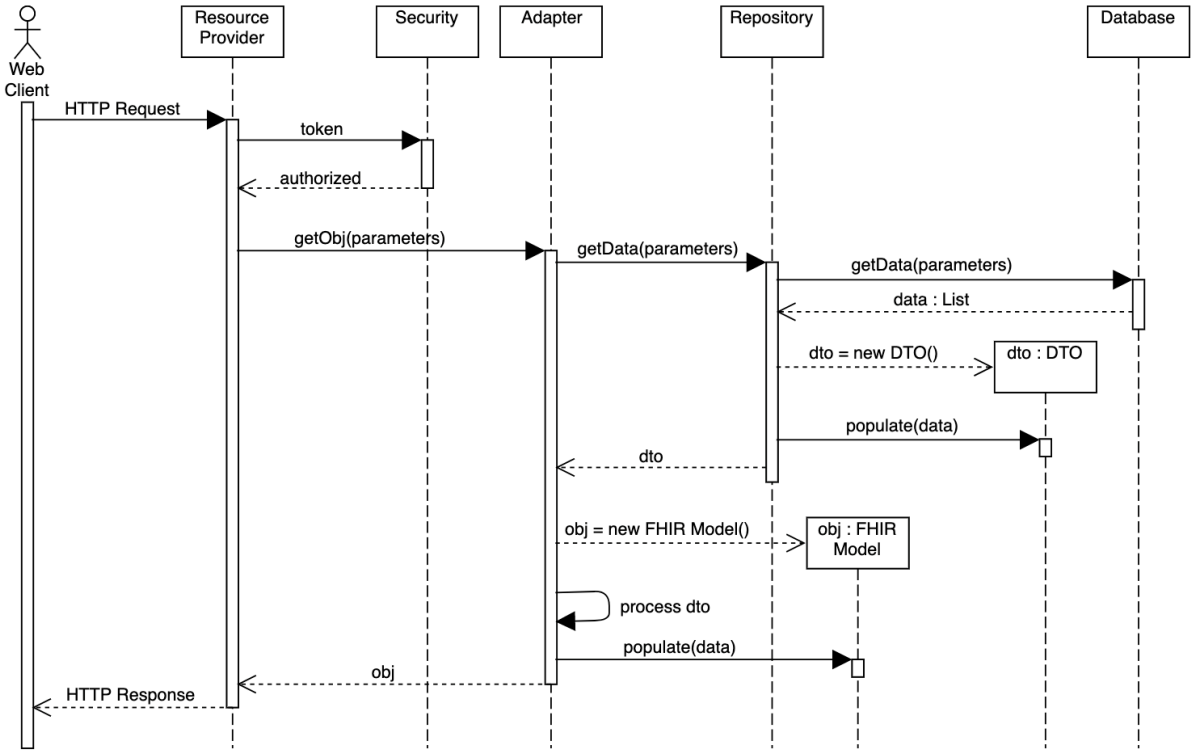


Figure 27 – Generic REST Request with DTO

5.4 Data Model

The domain model explained in section 5.2 Business Model translates into a data model. Since FHIR is a data modeling and exchange protocol, the data model is dependent on its specification, and is represented in Figure 28, without much detail for readability purposes. Appendix B contains a more detailed representation. The required resources for the scope of the project are part of the terminology module and were analyzed in section 3.1.3.5 - Terminology Module of the present document. In the figure there are attributes that start with a prefix “ext”, separated by a hyphen (“-“) from the attribute name, are not part of the FHIR main specification but are extensions. FHIR extensibility is explained in section 3.1.3.2 - Extensibility.

The Concept Map will be the resource used to expose a code system’s structure, so that a FHIR client can interpret any code system without the need of a human terminology expert. Since concept maps allow to exchange information about concepts relationships, it can display hierarchical structures, as well as compositional ones.

It is important to explain that, as it was stated previously, the package built on top of the database schema contains stored procedures created for that specific schema. Since this schema is not FHIR compliant, the stored procedures return the data in a

model that approximates to the FHIR model. As many attributes vary from one code system to another, an adapter is still necessary to adapt the received data to the HAPI Model Objects, which are implementations of the HL7's resources.

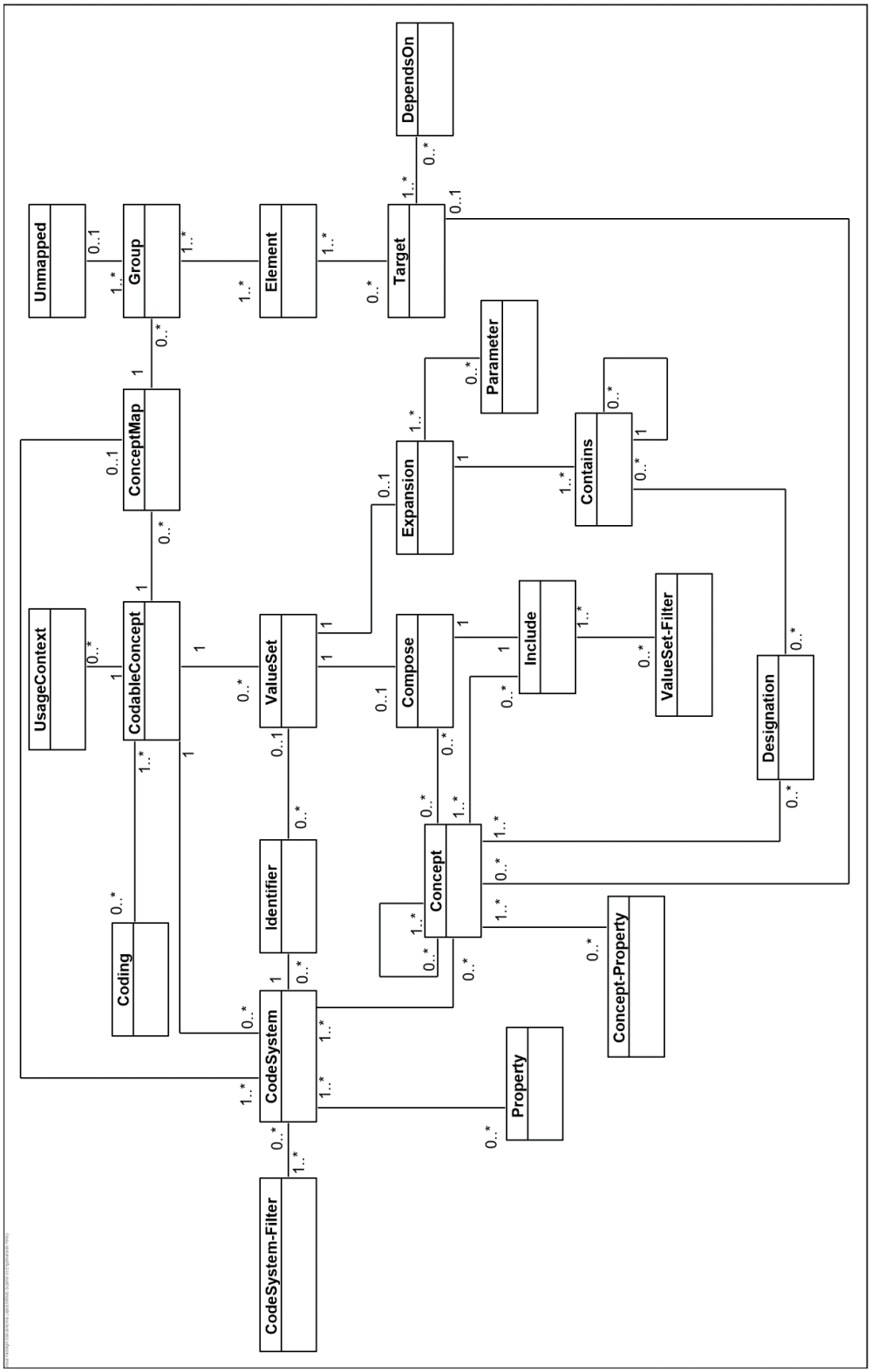


Figure 28 – FHIR R4 Data Model

5.4.1 Operations

From the operations that each resource can support analyzed in section 3.1.3.5 - Terminology Module and the requirements analysis, a list of operations was created. To explain this list is Table 17, where it is visible which FHIR and CRUD Operations should be supported by each resource in the final system. The table also demonstrates which of these operations returns concept related data.

Table 17 - FHIR Related Functionality

Functionality	FHIR Operations	Code System	Value Set	Relates to concepts
Read one	GET/resourceID	✓	✓	×
	Lookup	✓	×	✓
List all	GET	✓	✓	×
	Expand	×	✓	✓
Search	Find matches	✓	×	✓
Validate	Validate code	✓	✓	✓
	Subsumes	✓	×	✓
	Translate	×	×	✓

5.5 Code System Exposure Strategy

At this point, it is worth noticing that early on in the project, it was decided that each code system would consist of a specific terminology, in a specific version and language, due to differences in the information of the code system varying, like date of release/last update, publisher and other properties.

Another important decision was made on the exposure of the concepts. Since a lot of the code systems are very extensive, the listing of a code system by requesting it to the endpoint */CodeSystem* does not retrieve any concepts. When possible and agreed, the value set associated with a code system contains reference to other value sets which together compose the entire code system. Performing an expand operation on this value set allows to retrieve the references, while performing the same operation on the referenced value sets retrieves part of a code system. This away, an entire code system can be retrieved in parts, whenever adequate. In order for this to happen similarly across all code systems, a strategy needs to be defined what it comes to how they will be split.

By analyzing the code systems in use within the company software suite, there is not a pattern in the way they are organized, although most presents a structure that falls under one of the following options:

- The code system has different types of concept which are aggregated into categories (like diagnosis, exams, procedures, etc.);
- The code system has a hierarchy with a maximum of 25 concepts being the on top of the hierarchy, with the other ones being their descendants. These top level concepts are called main ancestors from now on.

Therefore, these conditions should be the decision factors to split a code system when necessary, and should not be intersect. Preference should be given to the categories. The information on where a code system should or should not be split must be stored and accessible when exposing code systems in value sets. This means that when a code system is split by its categories, the categories will not be present in the concepts, while the main ancestors are a concept themselves, therefore appearing in the listed concepts of their own value set.

In order to identify the complete and the partial value sets, as well as to which part the partial ones are relative to and the code system itself, since all of the value sets are fabricated on the go, the following strategy was defined:

- Each code system has an internal unique id;
- The id of each value set contains the exact internal id of the code system;
- If the value set refers to an entire code system, the id of the value set is equal to the id of the code system;
- If the value set refers to a partial code system, the id of the value set is equal to the id of the code system with a suffix referencing the main ancestor or category;
- The suffix is composed of an "_" character followed by the name of the main ancestor or category;
- The main ancestor name is the code itself;
- The category name is the code type name defined in the code system.

This way it can be possible to easily find out if a value set is partial or complete from the identifier provided. It also becomes relevant to notice that the character chosen to initiate the suffix should never be present in any id of the code systems present in the database, and this information should be disclaimed in the server documentation.

5.6 Concept Properties Exposure Strategy

Another important question in the project was how to expose attributes associated with concepts but are very distinct between the terminologies. By performing a Lookup operation on a concept the server should provide all the adequate information about the concept, including alternative designations, concepts with direct relationships and the category in which they fall under, if that is the case. Since this is the operation that allows to obtain the most information on a code provided and it should be standardized, FHIR defines that alternative designations are presented in a vector called designation and with their name has defined by the publisher.

All this data should come in a vector named "*property*", according to FHIR specification, as shown on the following code example, with the code "*C00.1*" from ICD-O-3, which belongs to the "*topography*" category, has an inclusion term and a parent.


```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "code",
      "valueString": "C00.1"
    },
    {
      "name": "display",
      "valueString": "External lower lip"
    },
    {
      "name": "property",
      "part": [
        {
          "name": "code",
          "valueString": "category"
        },
        {
          "name": "value",
          "valueString": "topography"
        }
      ]
    },
    {
      "part": [
        {
          "name": "designation",
          "part": [
            {
              "name": "code",
              "valueString": "inclusion term"
            },
            {
              "name": "value",
              "valueString": "Lower lip; NOS"
            }
          ]
```

```
        }
      ]
    },
    {
      "name": "property",
      "valueString": "parents",
      "part": [
        {
          "part": [
            {
              "name": "code",
              "valueString": "parent"
            },
            {
              "name": "value",
              "valueCodeableConcept": {
                "coding": [
                  {
                    "code": "C00",
                    "display": "Lip"
                  }
                ]
              }
            }
          ]
        }
      ]
    }
  ]
}
```


Code Extract 1 – Partial Lookup response on code C00.1 in JSON

6 Development

The current chapter describes the implementation process of the project, its prerequisites and architecture, the database connectivity system, the algorithm which translates the strategy explained in section 5.5 - Code System Exposure Strategy, showcases other relevant parts of the server and each responsibility. Finally, the implementation regarding security is illustrated.

6.1 Requirements

The program language in which the solution is implemented is Java, since HAPI was the chosen library, as section 4.2.5 - Implementation demonstrates. The project was built using Spring Boot, a framework that simplifies configuration in Spring applications. Spring is a framework that provides infrastructural support at the application level dealing with inversion of control (IoC) and dependency injection. Spring works with Beans that are objects instantiated and assembled by a Spring IoC container, which contain dependencies that are reflected in the configuration of the application (Spring Team, 2016).

HAPI, Spring Boot and Oracle dependencies were needed and to manage the dependencies on these external libraries, maven was the selected tool.

6.2 Implemented operations

The operations and parameter combination implemented in each endpoint of the server is summarized in Table 18. This information is also accessible in the /metadata

endpoint, which is not included in the table. The parameter "_format" is also not represented but can be sent on any request with the value "application/fhir+xml" to obtain the response in XML.

Table 18 - FHIR Implemented operations and parameter combinations on Code Systems

Endpoint	Operation	Parameter	Modifier	Description
CodeSystem	GET	-	-	List all Code Systems
		identifier	-	Search a Code System by the identifier
		name	-	Search a Code System by the exact name
		name	:exact	
		name	:contains	Search a Code System that contains in name
		title	-	Search a Code System by the exact publisher
		title	:exact	
		title	:contains	Search a Code System that contains in publisher
	Lookup	system, code	-	Get additional details about a concept
		system, code, category	_	Get additional details about a concept form a category
	Validate-code	identifier, code	-	Validate if a code is in a Code System
		identifier, code, display	-	Validate if a code is in a Code System with a certain display
		system, code	-	Validate if a code is in a Code System
		system, code, display	_	Validate if a code is in a Code System with a certain display
	Subsumes	identifier, codeA, codeB	_	Check relation between codeA and codeB
		system, codeA, codeB	_	
	Find-matches	identifier/system, code, display	_	Search a concept with a display and a code in a code system
		identifier/system, code, display, category	-	Search a concept with a display and a code in a code system category
		identifier/system, code/display, category	-	Search a concept with a display or a code in a code system category
		identifier/system, code/display	_	Search a concept with a display or a code in a code system

Table 19 - FHIR Implemented operations and parameter combinations on Value Sets

Endpoint	Operation	Parameter	Modifier	Description
ValueSet	GET	-	-	List all Value Sets
		identifier	-	Search a Value Set by identifier
		url	-	Search a Value Set by url
		publisher	:contains/exact	Search a Value Set by publisher
	Expand	identifier, includeDesignations	-	Expands a Value Set exposing its concepts, with or without designations
		url	-	
		identifier, parent	-	Expands a Value Set to only concepts whose parent is the provided
		url, parent	-	
		identifier, child	-	Expands a Value Set to only concepts whose child is the provided
		url, child	-	
		dentifier, ancestor	-	Expands a Value Set to only concepts whose ancestor is the provided
		url, ancestor	-	
		identifier, descentant	-	Expands a Value Set to only concepts whose descendant is the provided
		url, descentant	-	
	Validate-code	identifier, code	-	Validate if a code is in a Value Set
		identifier, code, display	-	Validate if a code is in a Value Set with a certain display
		system, code	-	Validate if a code is in a Value Set
		system, code, display	-	Validate if a code is in a Value Set with a certain display

6.2.1 Contains and Exact Modifier

The modifier seen in the Code System search was implemented in order to provide more flexibility to the search of code systems. They are modifiers specified by HAPI and their default value is “exact”. The difference between the two in the actual implementation is created by a method which appends the character “%” in the beginning, end and in between words of the value provided for the parameter, resulting in a more flexible search. This code is provided in Code Extract 1. The query for searching the Code System uses the expression LIKE instead of an equal sign (=) in the search condition designated to the parameter being handled and compared to an uppercase version of the data in the database.

```
public String appendForFlexibleMatch(String string) {
        String result = "%";

        Pattern pattern = Pattern.compile("\\s");
        Matcher matcher = pattern.matcher(string);
        boolean found = matcher.find();

        if(found) {
               String[] bits = string.split(" ");
               for(String bit : bits) {
                      bit = bit + "%";
                      result = result + bit.toUpperCase();
               }
        } else {
               result = result + string.toUpperCase() + "%";
        }

        return result;
 }
```

Code Extract 2 – Flexible Search

6.3 Package Architecture

The component diagram on presented on section 5.3 - Architectural Design translates into the package diagram below. The configurations component consists of the *config* and *operations* package, while the others have a single designated package with the same name.

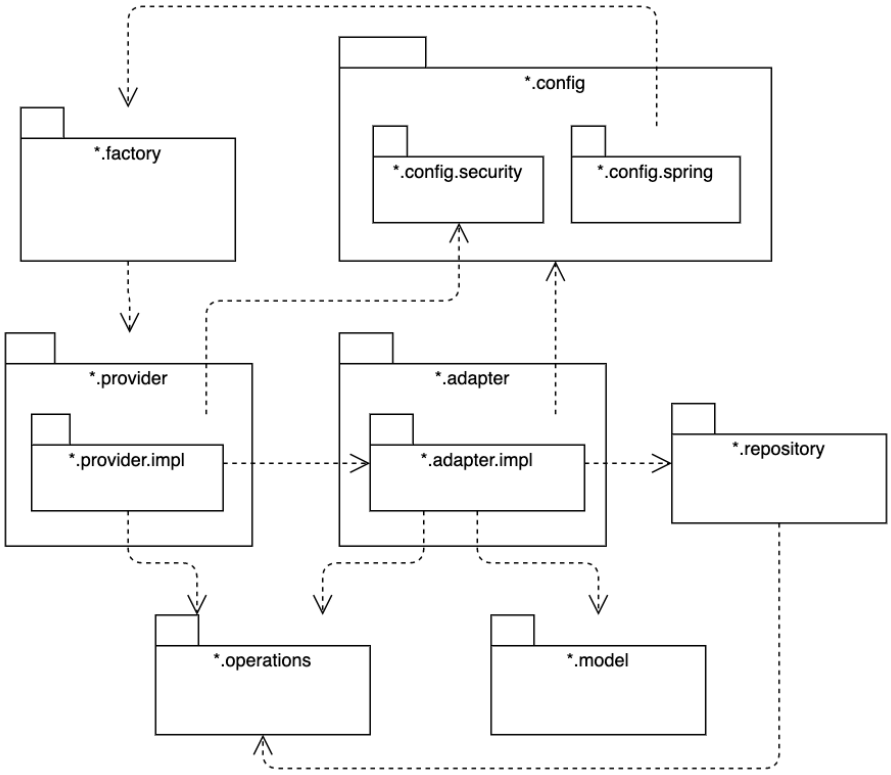


Figure 29 – Package diagram

The use of DTOs ended up not being the preferred solution for the case, although it was necessary to create models that stored information when performing complex tasks. Throughout the experience of the implementation of this project, several changes were made and this ended up being the solution that made sense ate the time. These created objects are referenced as models from now on but it is still important to notice that they're responsibility is very similar to the DTO's

6.4 Model, Constants and Operations

As stated in the explanation of the architectural design in section 5.3, the adapter is the brain of the system, being responsible for knowing which information to provide and how to structure it according to the strategic design decisions made and FHIR. The process of adapting the data format while considering the approach to expose code systems consists of several steps, so model entities were created to help the adapter storing information during these steps. These models and their attributes are presented in the class diagram bellow.

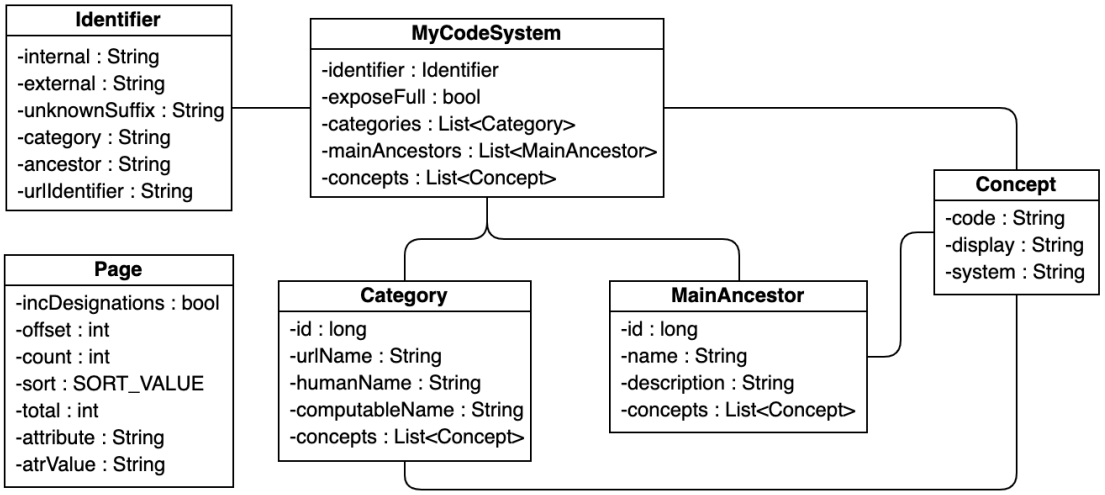


Figure 30 – Models class diagram

The *Identifier* class stores the identifier in several formats and has the responsibility of configuring itself when the new data demands it. This means that it is responsible to store a suffix, even when it is still an unknown suffix. The *MyCodeSystem* class stores important data about a code system necessary for the adapter to access it and decide what to return to the provider. It either has the list of concepts populated or one of the other two, categories and main ancestors, which also can store concepts. Since the HAPI models are the ones to be returned by the provider, no models were created to map the value set data entity, and the happy one was used instead, being populated with more and more data throughout the process.

The *Page* class stores relevant data about the exposure of the concepts, like order and amount. It also stores an attribute and its value used when retrieving a page of concepts given a certain condition on an expand operation. Some of these attributes have a default value or optional values stored in the class *ServerProperties* package *Config* in constants. There is also another class called *Constants* that stores other values like the date format compliant with FHIR.

Finally, there are some operations that are constantly performed on the data, like replacing empty space with a specific character, therefore there is a *ParamOperations* class containing methods to operate HAPI parameters and a *StringOperations* class to operate and transform strings, like the method showcased in Code Extract 2.

6.5 Database and Connectivity

When it comes to the database, it is based on oracle technologies and is an instance of ALERT's internal database schema from terminologies. The package designated to

operate for the server, represented on Figure 32 - FHIR Terminology Infrastructure Component in section 5.3 - Architectural Design, was developed in PL/SQL. This package contains internal functions and procedures. The procedures are the ones to be called by the server when requesting information, while the functions perform specific operations that help the method retrieve data.

To connect the server to the database, the Java DataBase Connectivity (JDBC) Driver was the option, using the SpringBoot Starter JDBC library. The main class used to create the connectivity is he *JdbcTemplate*, which is the central class in the JDBC core package that simplifies the use of the driver (Spring Team). To create the connectivity, the driver requires a *DataSource* element, created through a Spring Bean on moment of deployment (Code Extract 3) and initiated in another, which is then used to create the *JdbcTemplate* by the *Factory* component and attribute it to the *Repository* instance.

```
@Bean
public static DataSource getDataSource() {
	DataSourceBuilder<?> dataSourceBuilder =DataSourceBuilder.create();
	MyDataSource ds = properties.getDataSource();
	dataSourceBuilder.driverClassName(ds.getDriver());
	dataSourceBuilder.url(ds.getUrl());
	dataSourceBuilder.username(ds.getUser());
	dataSourceBuilder.password(ds.getPassword());
	return dataSourceBuilder.build();
}
```

Code Extract 3 – DataSource Bean

The *Repository* contains methods that call stored procedures sending the parameters they require, making use of the following method (Code Extract 4) to instantiate the call. After instantiated, the parameters are declared and attributed and then the call is executed. The output parameters of a procedure also need to be declared before the executing.

```
public SimpleJdbcCall getNewCall(String procedureName) {
	SimpleJdbcCall simpleJdbcCall = new SimpleJdbcCall(jdbcTemplate)
			.withSchemaName(constants.schemaName.toUpperCase())
			.withCatalogName(constants.packageName.toUpperCase())
			.withProcedureName(procedureName.toUpperCase());
	return simpleJdbcCall;
}
```

Code Extract 4 – Calling a procedure stored in the package

Since the use of DTOs was not the chosen path for the project, the data arrives to the *Repository* trough the output parameter in the format of strings for simple data and Ref Cursors for more complex one. Ref Cursors are defined in the package and when returned, are materialized as *ResultSet* objects in Java. The *Repository* extracts the data into *Lists* before returning it to the *Adapter*.

6.6 Exposure Strategy Implementation

As stated previously, the strategy on exposing the code system is to split them using one of two decision factors. It was also explained that listing concepts would be accessible by performing an expand operation to several value sets that together compose an entire Code System. This list of value sets is accessible by performing an expand operation on the Value Set referencing the entire Code System. This creates some difficulties for tasks involving suffixes. In the following sections of this document, some of this difficulties are addressed.

6.6.1 Returning a Value Set with a suffix

When getting a value set by its id, since the id of the value set contains the id of the code system and can or not contain the name of a category or main ancestor, it is important to check this situation. Not much changes in the information returned, and there are no concepts involved but it is still required to validate if there is a suffix and if that suffix represents an existing main ancestor or category of the code system. After this validation, the resource can be populated and returned. This validation is made on the same procedure that returns the data about the code system so less calls are made to the database. Code Extract 5 showcases a partial Value Set where the only filed affected are the id, url and description ones.


```
{
    "resourceType": "ValueSet",
    "id": "ICD-O-3-EN-v2000_topography",
    "url": "http://localhost:8080/Fhir-Server/ValueSet/ICD-O-3-EN-
v2000_topography",
    "version": "2000",
    "name": "ICD-O-3-EN-v2000_topography",
    "title": "International Classification of Diseases for Oncology 3rd Edi
tion 2000 - topography part",
    "status": "active",
    "experimental": true,
    "publisher": "World Health Organization (WHO)",
    "contact": [
        {
            "telecom": [
                {
                    "system": "url",
                    "value": "https://www.who.int/"
                }
            ]
        }
    ],
    "immutable": true,
    "copyright": "license”
}
```


Code Extract 5 – Response to request ValueSet/ICD-O-3-EN-v2000_topography

Categories and main chapters must be distinguishable by the system, since they are not the same structures in the database and distinct queries needed to be executed when performing certain operations that involve the structures. While the main ancestors are a code from a certain code system, the categories represent a code type specified by the Code System itself, usually present in compositional Code Systems. Therefore, when necessary, there are procedures dedicated to working with each one of them. In the get operation mentioned above, if the id provided contains a suffix, the only information that changes in the returned resource relatively to a return resource is the id of the resource and its description.

6.6.2 Expanding a Value Set with a suffix

It becomes crucial for the server to make this distinction when listing, searching and validating concepts within Value Sets that only contain part of a code system, since different procedures exist for different suffix type. When listing concepts in a resource like this the server must ensure interpretability of the suffix in order to expose the correct data. Figure 31 presents a decision tree that demonstrates the decision process that needs to be made on a request of an expand operation.

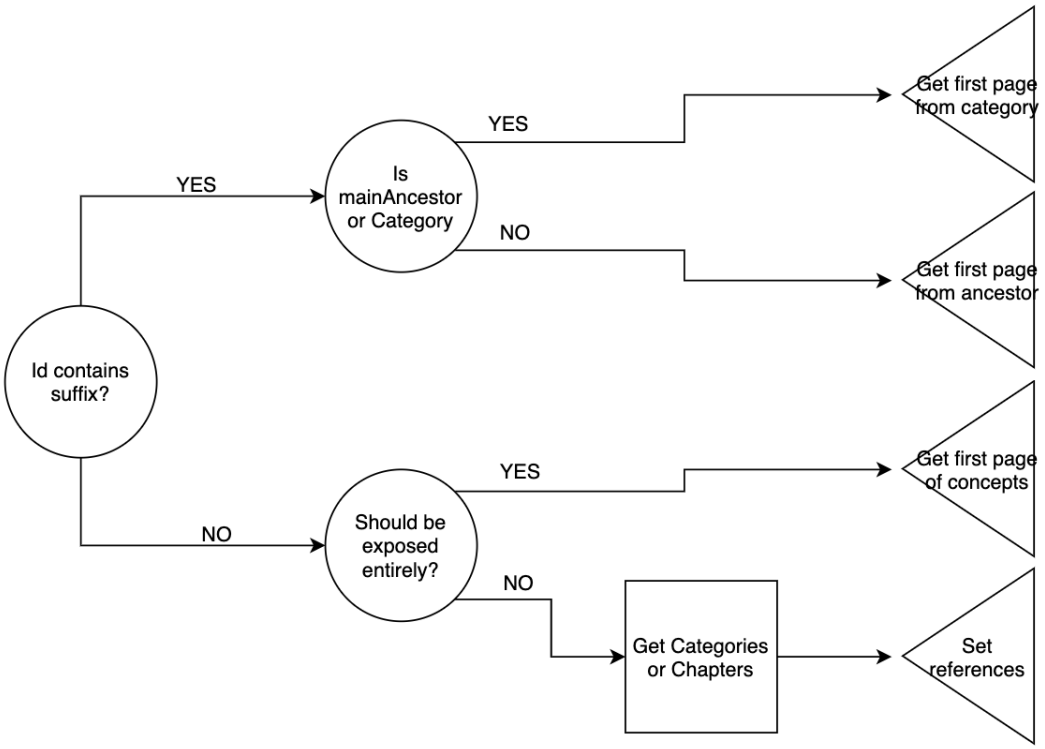


Figure 31 - Decision Tree Expand operation on /ValueSet with id

The first decision knot in the tree questions if que identifier received contains a suffix. Code Extract 6 showcases the creation of an identifier that is done. Since the identifier

is unknown on either it contains a suffix or not, the identifier is asked to sort this in the method *setUnknownIdentifier()*. Then, if the identifier contains an unknown suffix, the method *setMainAncestorOrCategory* is called to identify. This method then asks the repository for the information and stores it adequately as shown in Code Extract 7, part of the method *setMainAncestorOrCateogry* that is seen being called on Code Extract 6.

```
this.cs = new MyCodeSystem();
Identifier csIdentifier = new Identifier();
csIdentifier.setUnknownIdentifier(identifier);
if (csIdentifier.getUnknownSuffix() != null) {
        csIdentifier = setMainAncestorOrCategory(csIdentifier);
}
```

Code Extract 6 – Creation of Identifier and request to identify suffix

```
String result = genericRepository.getAncestorOrCategory(csIdentifier);
if (result != null) {
        String name = csIdentifier.getUnknownSuffix();
        if (result.equals("category")) {
               category = new Category(name);
               this.cs.addCategory(category);
               csIdentifier.setCategory(name);
        } else if (result.equals("ancestor")) {
               mainAncestor = new MainAncestor(name);
               this.cs.addAncestor(mainAncestor);
               csIdentifier.setAncestor(name);
        }
 }
```

Code Extract 7 – SetMainAncestorOrCategory calling the Repository

After this portion of code is executed, this *Identifier* instance is now ready to be used in the following process. Another method can be called to gather a concept, a list of concepts or a validation result. In the example that is being here explained, that method has the name *setExpandedValueSet* and it deals with the decisions presented in the second level on the decision tree, calling the appropriate methods to request data to the repository, receive and adapt it.

The decision to split the code systems also makes listing all value sets at once a very time consuming operation, since for each Code System it needs to be created a value set for each category/main ancestor, plus a value set for the entire Code System. The workflow here is to retrieve all the Code Systems from the database with all the information needed in a Value Set resource. Then, for each Code System, get the categories and check if there are any. If there are, get the main ancestor and create a Value Set for each main ancestor. If there are categories, this is done to the category list and a Value Set is added for each category, this way giving preference to the categories, like determined before.

6.7 Concept Relationship Validation

The Subsumes FHIR operation retrieves the relationship between two concepts of the same code system, allowing to identify if a concept subsumes or is subsumed by another, or if there is no relationship between the two. These operation is simple when taking in consideration Code Systems with simple hierarchy structures. The longer the hierarchy, the more complex this validation becomes.

So when we have a simple two level hierarchy Code Systems, a simple select to the type of relationship will do the job, but when the Code System's internal structure is composed of a multilevel hierarchy, a recursive search needs to be performed. Furthermore, this recursive search need to be performed twice since we need to validate the relationship in both directions, being code A subsuming B or being subsumed by B. This means the search type depends on the inherit structure of the Code System itself, so it is wise to create an association with the type of search to the code system. This way, we could perform different searches based on real time needs and save extra complicated queries for when it is really necessary, improving the overall performance.

Since the type resumes to it being recursive or not which depends on the Code System, and who uploads this content to the server are specialists that know Code Systems better than anyone, a column in the database named recursiveSearch with a Boolean was the decided solution. The column is true whenever a Code System is composed of a multilevel hierarchy and should be filled when adding a new Code System to the database. This information is also disclosed on the server documentation for usage and maintenance. Moreover, by keeping this information in the database, the server can simply perform a request to the procedure that takes care of the subsumes operation and the procedure will quickly be able to know what query to perform according to the result of a simple select using the identifier of the code system, without the server needing to perform an extra request to the database and processing.

6.8 Security

When it comes to security, Spring Security was used to perform JDBC authentication of the users. Each user is associated with a role and is authenticated and authorized to access the server in each request. The authentication is made using an e-mail and a password.

7 Experiments and Evaluation

In order to ensure that the infrastructure fulfills the customer's needs it is necessary to perform tests that can measure the customer satisfaction accurately. This chapter focusses on describing the evaluation methodology and its intrinsic indicators used to verify the hypothesis proposed on section 1.3 Hypothesis of the prevailing document. Later on, results will be discussed, as well as possible improvements.

Since there is a set of requirements gathered in accordance with the costumer's needs, the fulfillment of these dictates the success of the solution. These requirements follow the FURPS+ model, which involves establishing priorities and making quality attributes measurable. The VA is also a contributing factor to establishing priorities since it supports the understanding of importance of functionalities. The VA process concluded that the most important function of the server is to expose concepts from multiple code systems, using FHIR specification while maintaining the integrity of the information.

7.1 Incremental Integration Testing

Throughout the development of the FHIR terminology server, the developer will perform integration tests continuously, as new functionalities are implemented. Their intention is only to test functionalities using an informal approach. Later on, these can be refined to test functionality requirements. These tests consist on a set of HTTPS requests to the several endpoints of the server, testing each operation, using Postman as the web client, a popular tool in API testing which supports creating automated tests.

7.2 Quantitative Evaluation Framework

The main evaluation methodology established for testing the generic hypothesis is the Quantitative Evaluation Framework (QEF). This framework allows the definition of an ideal solution, to evaluate the real solution at a certain moment in time and to calculate the distance between these two, in order to conclude on requirements fulfillment and overall customer satisfaction. Making use of the gathered requirements, their level of importance and metrics to measure each requirement realization, it measures the quality of the system creating a linear relationship between the quality and the percentage of requirement attainment. Requirements are grouped by dimensions, which are spilt into factors (Nuno Escudeiro, 2019). QEF is a form of system testing, since system testing is, according to Myers and Sandler (2004, p. 150) "a form of higher-order testing that compares the system or program to the original objectives" and to complete this form of testing "you must have a written set of measurable objectives", which in this case are the requirements.

In the QEF elaborated to evaluate the solution (Appendix C), the dimensions are constructed based on the FURPS+ categories. Since the dimensions are equality important, a bad balance between the amount of requirements in each one will affect the final result. As such, the FURPS+ categories, with the exception of functionality, where gathered into two larger groups of requirements. For each one is defined a metric to measure its success. These requirements are empirically tested making use of several testing techniques, depending on the requirement itself. There will be two testing phases, one before the system is in the production environment and another after, which is the one that will dictate the solution quality. This allows for improvements in the system before going to production. Before proceeding to the explanation on the tests, it is useful to analyze Appendix C, where the QEF is presented with its requirements and metrics.

7.2.1 Functional/End-to-End Testing

Functional testing is a type of testing that focusses on requirements. It represents a black-box type of testing, which means there is no concern about the internal behavior and structure of the program. Instead, the goal is to find circumstances in which the program does not behave according to its specifications (Myers and Sandler, 2004, p. 95). This is similar to end-to-end testing that is based on the same premise, with the difference that functional testing can be done in an environment distinct from the

production one. Both approaches will occur, each one in their appropriate testing phase. The methodology use encompass a set of requests in Postman, like the incremental integration tests, but with more rigor since they will include data that is on the database to be compared with the responses.

7.2.1.1 Methodology and Indicators

Table 20 describes the requirements, which represent the indicators that will dictate the hypothesis veracity, as well as the metrics used to test them. These requirements come from the Functionality, Usability, Reliability and Supportability (configurability factor) dimensions of the QEF. For more detail on these indicators, reference Appendix C for comparison.

Table 20 – Functional indicators and metrics

Id	Indicator	Metric
FF01	The server connects to a database that contains the terminologies basic information, their content, concept categories, languages and ALERT contexts	Yes/No
FF02	The server lists its capabilities	Yes/No
FF03	The server lists all code systems and their attributes contained in the database	Yes/No
FF04	The server displays a code system's structure in a way interpretable by a FHIR client	Yes/No
FF05	The server allows validation of concepts within code systems	Yes/No
FF06	The server allows to search for a code system by its name	Yes/No
FF07	The server allows to search for a code system by its properties	Amount of properties
FF08	The server is able to list all concepts and their properties from a code system, in a single or multiple value sets	Amount of concepts
FF09	The server is able to list all concepts and their properties from a single concept category within a code system in a single value set	Amount of concepts
FF10	The server allows validation of concepts within concept categories	Yes/No
FF11	The server lists all concept categories contained in the database	Yes/No
FF12	The server allows to search for a concept category by its name	Yes/No
FF13	The server is capable of listing relationships between concepts	Yes/No
FF14	The server allows to search for a concept by its properties	Amount of properties
FF15	The server is capable of displaying paginated resources	Yes/No
US01	The server warns the client against unexpected and unresolved requests	Yes/No
RI01	The server displays accurate and consistent data according to the database	Yes/No
RI02	The server does not display repeated data in the same resource	Yes/No
RS01	A generic web client can access the server by HTTPS	Type of access
SC01	The server is able to represent resources in XML and JSON	Amount of formats
SC02	The server is able to represent resources in XML or JSON based on a configuration	Yes/No

Indicators FF8 and FF9 will be tested on the amount of concepts the server is able to list with all required attributes, comparing with the amount of concepts it should list. The metrics are: 0% of requirement fulfillment for no concepts listed; 25% for less than half of the concepts listed; 50% for half of the concepts listed; 75% for more than half but less than the total of concepts it should list; 100% for the all of the concepts listed. Tests will be performed on every code system, then a mean of the results is calculated, which will represent the requirements fulfillment.

Indicators FF7 and FF14 will be tested on the amount of properties the server accepts for searching. The metrics are: 0% of requirement satisfaction for no properties accepted; 50% for a single property accepted and 100% for multiple properties accepted in a single request.

The US01 indicator will be tested making invalid requests to the server. If the server warns against all the invalid requests, the requirement is fulfilled at 100%.

Indicators RI01 and RI02 will be tested comparing the data from a response with the data contained in the database.

Indicator RS01 will be tested on the protocol the server can be accessed by. If the server only accepts HTTP requests, the requirement is only satisfied by 50%. On the other hand, if the server accepts requests via HTTP and HTTPS, the requirement is completely fulfilled (100%).

The SC01 indicator will be tested on the amount of formats it can display the responses. If only one format is supported, 50% of the requirement is fulfilled, while if the server supports both formats, the requirement fulfillment raises to 100%.

The remaining indicators' metrics are binary, meaning that they either are satisfied or not.

7.2.2 Performance Testing

With the existence of a performance requirement, it becomes necessary to conduct some tests when it comes to the performance of the software. The performance should be tested in both testing phases, since the response times will most likely differ from development to production environment.

7.2.2.1 Methodology and Indicators

Table 21 - Performance indicators and metrics

Id	Indicator	Metric
PS01	The server responds to a request of listing all code systems or value sets in 5 seconds or less	Yes/No
PS02	The server responds to a request for a search, including find-matches and lookup operations in 2 seconds or less	Yes/No
PS03	The server responds to a request for a validation in 2 seconds or less	Yes/No
PS04	The server responds to a request for a subsumes operation in 2 seconds or less	Yes/No
PS05	The server responds to a request for an expand operation in 4 seconds or less	Yes/No

For this evaluation, the methodology applies statistical knowledge so that the results are significant. Therefore, the sample should contain at least 30 iterations for each request. After the data collection, this data should be analyzed from a statistical point of view. In order to determine what type of statistical test should be performed, the distribution of the values should be calculated, since the tests should be parametric. Then, the statistical test to the hypothesis can be performed so that the probability value is calculated and conclusions are made. Postman can be used since it registers the response time of a request.

7.2.3 Security Testing

Security testing is a form of system testing whereby the security mechanisms of an application or system are tested, trying to compromise them (Myers and Sandler, 2004, p. 150). The test plan regarding the security requirements should also be performed in both testing phases (development and production-like environment).

7.2.3.1 Methodology and Indicators

Table 22 demonstrates the indicators that fall under this testing category, as well as the metrics used to evaluate them. These all are indicators relative to requirements in the functional dimension and factor of the QEF.

Table 22 - Security indicators and metrics

Id	Indicator	Metric
FF16	The server is capable of authenticating an user	Yes/No
FF17	The server is capable of verifying an user's authorization to access data	Yes/No

The FF16 indicator will be tested by performing a request to the server using correct credentials of access. The request is to the endpoint that returns the server's

capabilities. In this case, the server should be able to authenticate the user and return a response, since all authenticated users will have access to this information. Than the same test should be performed but using incorrect credentials, in order to guarantee that the authentication system works. In this scenario, the response should warn against wrong credentials. The requirement is fulfilled only if both results are as expected.

Indicator FF17's testing methodology follows the same logic has the previous one. Using credentials of a user who has access to a specific information that not all users have access to, like a code system that requires users to have a license for usage, a request is made to access this information. The server should be able to authorize the user and return a response with the information requested. Afterwards, the same test should be conducted, now using credentials from a user who does not have access to that information. In this case, the server should warn the client about the lack of authorization to access the requested information. Again, the requirement is satisfied only if both results are as expected.

7.2.4 Other approaches

In this section are explained other approaches to requirements testing for requirements not approached in the sections above, due to their scope not being related with the system's functionality, performance or security, or even because functional and end-to-end testing does not apply. Just like the other approaches, these tests should be encompassed in both testing phases.

7.2.4.1 Methodology and Indicators

In Table 23 are displayed the indicators that are missing a testing plan so far. These relate to requirements within the Support factor from the Usability dimension, the Consistency factor in the Reliability dimension, the Scalability factor in the Supportability dimension, and lastly, the Knowledge factor, which belongs to the dimension related with other categories.

Table 23 – Other indicators and metrics

Id	Indicator	Metric
US02	The system comes with extra information about its functionality and usage	Yes/No
RC01	The server works continuously without failure or error	Yes/No
ST01	The server comes with a set of tests to test and server has an example for clients	Amount of tests VS amount of capabilities
SS01	The server was built with a good architectural structure, following design patterns and engineering best practices	Yes/No
OK01	The development of the solution gives preference to Oracle technologies - Java and SQL. If not, proper justification must be presented	Yes/No

These indicators' metrics are binary, meaning that either they are satisfied or not, with the exception of ST01, which evaluation will be based on the relationship between the amount of tests and the amount of capabilities the server supports. The metrics are: 0% of requirement satisfaction for no tests provided; 25% for tests for less than 50% of the capabilities; 50% for tests for 50% of the server's capabilities; 75% for tests for more than 50% but less than 100% of the server's capabilities; 100% for a set of tests containing tests for all the server's capabilities. These capabilities include all of the supported operations for each endpoint, including all the server's available endpoints.

7.3 Results

There were two phases of testes besides the incremental integration testing that occurred during the development. This incremental testing allowed was when it was constructed the Postman collection, with requests and tests being added as functionality was implemented. The results of both phases are presented below and will be discussed later on in this document.

7.3.1 Tests

A collection with 62 requests and was created using Postman to test both the functionality, security and performance of the server. Here are exemplified some of these tests and their workflow.

All requests have around 3 tests but sometimes requests are skipped because there is no data to perform that operation. One of these testes is a performance test, which verifies if the response time was smaller or equal to the ones defined in the speed requirements. Throughout the iterations, the package tests distinct Code Systems by

the order they are listed, so for example, if a Code System is not hierarchical, the requests dedicated to the subsumes operation are skipped.

In order to test some operations on concepts, a known concept of the code system needs to be provided. The first request in the collection involving concepts is a Lookup operation and contains a pre-request script to be executed before the request is made. This script uses the Code System stored for that iteration and makes an expand request to its generic Value Set. If this Value Set is composed of categories, select a random category and performs an expand operation on that partial Value Set. Then, either on the response of the generic or the partial Value Set, a random concept is chosen and stored. In case it being from a partial Value Set, the name of the part is also stored for future requests I the iterations. Since the Lookup operation is also the one that allows to retrieve more information on a concept, from the response of the request, other data is stored in the collection, like a related concept to perform a Subsumes operation.

For the Subsumes operation, in order to test a coherent result, the request is made twice, switching the values of the codeA and codeB parameters in between requests. In the first request there is a pre-request tests script which defines a codeB based on the stored information from the Lookup operation. Then the request is performed, and if the status code of the response is successful, the result is stored in an environmental variable as seen in Figure 32.

```
let skipTest = pm.environment.get('SkipSubsumes');
var res = JSON.parse(responseBody);

(skipTest ? pm.test.skip : pm.test)("Status code", () => {
    pm.expect(pm.response.code).to.equal(200);
    if(pm.response.code == 200) {
        pm.environment.set("ResultSubsumes",res.parameter[0].valueCode);
    }
});

(skipTest ? pm.test.skip : pm.test)("Performance Pass", () => {
    pm.expect(pm.response.responseTime).to.be.below(2000);
});
```

Figure 32 – Test Subsumes Operation 1

Next step is to switch the values and verify if the result was coherent with the first request. Figure 33 showcases the tests on the second request. These tests are skipped if the code selected in that iteration has no relationship to another, therefore not existing enough information to perform this operation.

```
let skipTest = pm.environment.get('SkipSubsumes');
var res = JSON.parse(responseBody);

(skipTest ? pm.test.skip : pm.test)("Status code", () => {
    pm.expect(pm.response.code).to.equal(200);
});

(skipTest ? pm.test.skip : pm.test)("Performance Pass", () => {
    pm.expect(pm.response.responseTime).to.be.below(2000);
});

(skipTest ? pm.test.skip : pm.test)("Functionality Pass", () => {
    var resultSubsumes = pm.environment.get("ResultSubsumes");
    var coherent = false;
    if((resultSubsumes == "Subsumes" && res.parameter[0].valueCode == "Subsumed-By") ||
    (resultSubsumes == "Subsumed-By" && res.parameter[0].valueCode == "Subsumes")) {
        coherent = true;
    }
    pm.expect(coherent).to.equal(true);
});
```

Figure 33 – Test Subsumes Operation 2

Other important features tested by the Postman collection are the error handling, related to the requirement “The server warns the client against unexpected and unresolved requests”. Figure 34 presents a Lookup request which tests the server response when the code provided is not in the Code System. According to FHIR, the response should be an OperationOutcome resource with an issue. This issue should have its code value “not-found” for errors of this type. Figure 35 demonstrates the tests on this request.

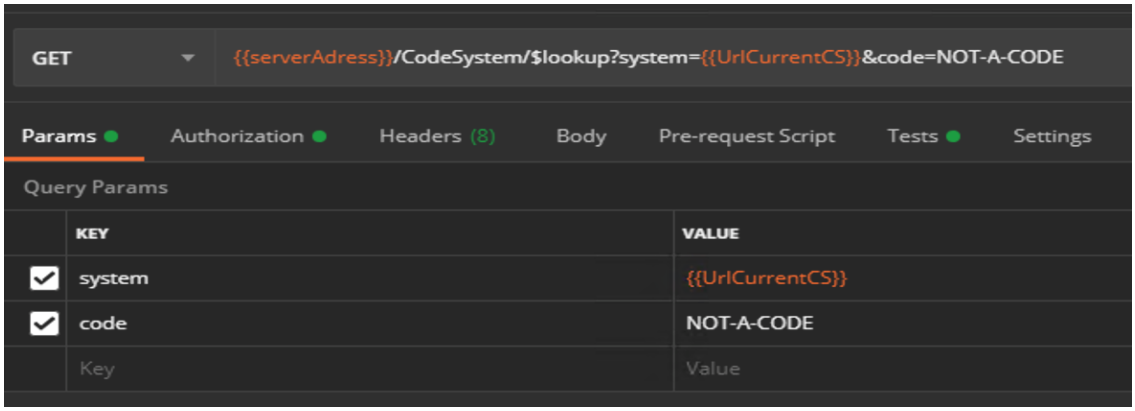


Figure 34 – Test Lookup Operation 1

```
var res = JSON.parse(responseBody);

// Response status and time
pm.test("Status code", function () {
    pm.response.to.have.status(404);
        pm.expect(res.issue[0].code).to.be.equal("not-found");
});

pm.test("Performance Pass", function () {
    pm.expect(pm.response.responseTime).to.be.below(400);
});
```

Figure 35 – Test Lookup Operation 2

7.3.2 First Phase Results

In the initial phase, midway through the implementation, performance and security were not tested. The phase consisted only on testing the system existing functionalities with QEF in order to understand the progress made so far and that still need to be made to complete the product.

The results gathered are summarized in the table below, which demonstrates the completion stage of each dimension and factor on QEF.

Table 24 – QEF First phase results

Factor	Qj (%)	Dimension	Qi (%)	Q (%)
Functionality	67,74	Functionality	67,74	70
Support	42,86	Usability and Reliability	80,95	
Consistency	100			
Integrity	100			
Security	100			
Speed	0	Performance, Supportability and Others	42,5	
Testability	25			
Configurability	100			
Scalability	100			
Knowledge	100			

At this point, the project already represented 70% of the ideal solution, according to the QEF results. The best performing dimension is the Usability and Reliability one, with 3 of its 4 factors being 100% complete. The lowest factor is the Speed on Performance, Supportability and Other dimension, which is on itself also the lowest dimension, and

this is explained by this factor being completely designated to performance testing, a form of testing that did not occurred in this phase.

In terms of functionality, the following requirements were still to be completely fulfilled.

- FF03 - List all code systems;
- FF04 - Display a code system's structure;
- FF07 - Search a code system by other properties;
- FF12 - Search categories by name;
- FF14 - Search a concept by its code, term or other properties;
- FF16 – Authentication;
- F17 – Authorization.

Note that not all of them were on the 0% mark, some of them had a 25% or even 50% completion state.

In the Support factor the only requirement not completely satisfied at this stage was "US02 - Provides information for usage" since the usage and maintenance documentation of the server was still not produced. In the Testability factor, the requirement is only one and was implemented only 25%, since the requirement is "ST01 - Provides a set of tests" and the tests made at this point were not a final version, not testing for even 50% of the necessities.

7.3.3 Final Phase Results

By the end of the development phase, the requirements were once again analyzed and its fulfillment verified. This phase of results is highly based on results from postman. The collection of tests was executed 30 times and the results were the following.

7.3.3.1 Functional/End-to-End Results

When it comes to functional and end-to-end testing, the results from the execution of the Postman collection dictated the fulfillment of the requirements, since it tested against every Code System in the server more than once.

In the first phase of results there were not a lot of requirements missing implementation from the functionality dimension. The postman collection tested every single functionality in the system, including these requirements. These functionality tests were performed having in mind that some Code Systems will not allow to perform some operations, and in that case, the test is skipped, just like explained previously

about the Subsumes operation. The functionality tests demonstrated that only the requirements related to authentication and authorization suffered changes in its metrics. The search of concepts by multiple parameters requirement is 50% complete, since the server allows to search for a concept by one property, and sometimes even multiple at once as described by the indicators, but due to localization issues and some special characters in codes and descriptions this last option has demonstrated to fail in 22 of the 30 iterations (around 74% of the sample), so it is not considered to be functional.

In this category of tests are also present requirements related to other FURPS+ categories that cannot be tested using a Web Client like Postman. The only requirements of this type missing fulfilment since the initial testing phase were the support and testability ones, related to the usage and maintenance manual for the server and the postman collection of testes, which are both completed. Therefore, the requirements are fulfilled.

Finally, there were still some requirements not being completely implemented but the evaluation of the project needed to still be done, since there was no more time for development. These are listed below, with the percentage of fulfilment achieved according to the metrics and indicators defined previously:

Table 25 – Requirements not fulfilled

Id	Indicator	Achievement (%)
FF03	List all code systems	0
FF04	Display a code system's structure	0
FF12	Search categories by name	0
PS01	Has the responds in 5s or less for listing operations	0

Since all of these requirements have binary indicators, since they are not entirely fulfilled, the indicator represents a 0% achievement on every one of them. The last here visible is related with performance and its tests results are presented in the next section.

7.3.3.2 Performance Results

The performance results need to be merged to fit into the performance requirements, since these are relative to a group of requests and not just one single request. So for each request, an analysis on the response times is performed. Then, if the mean time of all of the requests was higher than the hypothesized, the requirements fails.

The "PS01 - The server responds to a request of listing all code systems or value sets in 5 seconds or less" requirement relates to the request presented in Table 26 that also

showcases mean times, in milliseconds, for each request in a 30 iterations run. Figure 36 and Figure 37 demonstrate the times captured by the tests in milliseconds

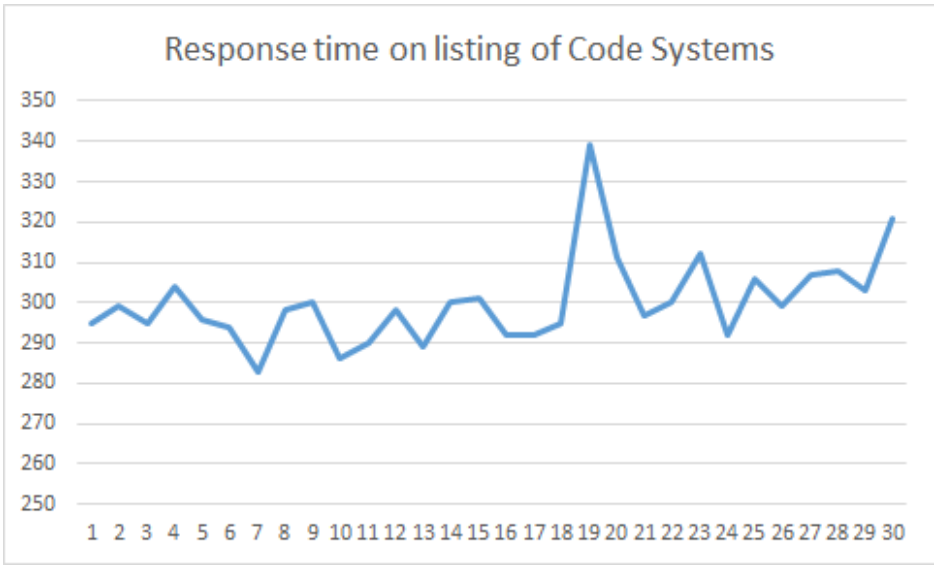


Figure 36 – Response times on listing of Code Systems

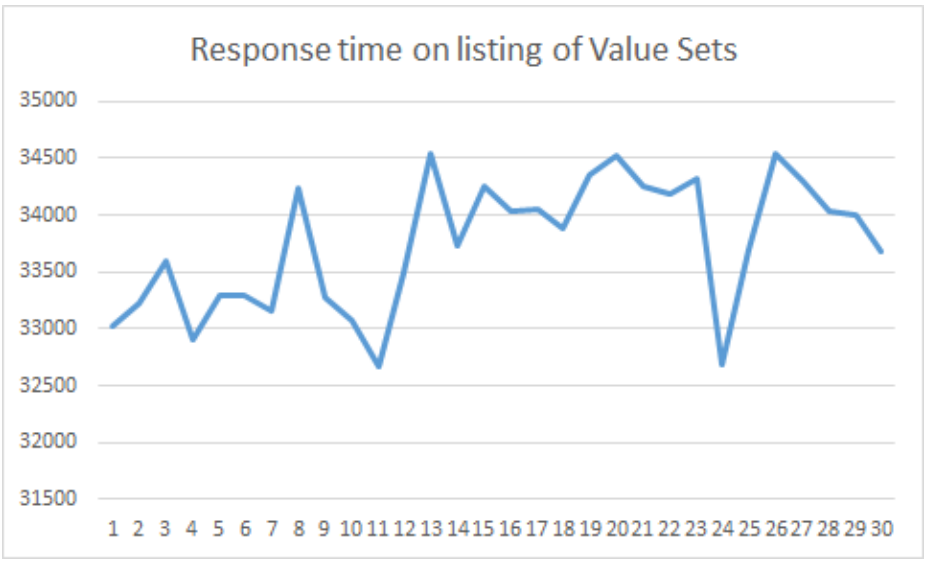


Figure 37 – Response time on listing Value Sets

The final average time is around 17 seconds, which is bigger than the one defined by the metric so the requirement did not got fulfilled.

Table 26 – Average times for List operations

Request	Average time	Metric
/CodeSystem	300,07	17022,6
/ValueSet	33745,03	

As for the “PS02 - Has the responds in 2s or less for searching operations” requirement, Figure 38 and Figure 39 demonstrate times collected by the tests on the different endpoints affected by this requirement. Table 27 showcases the average times in milliseconds.

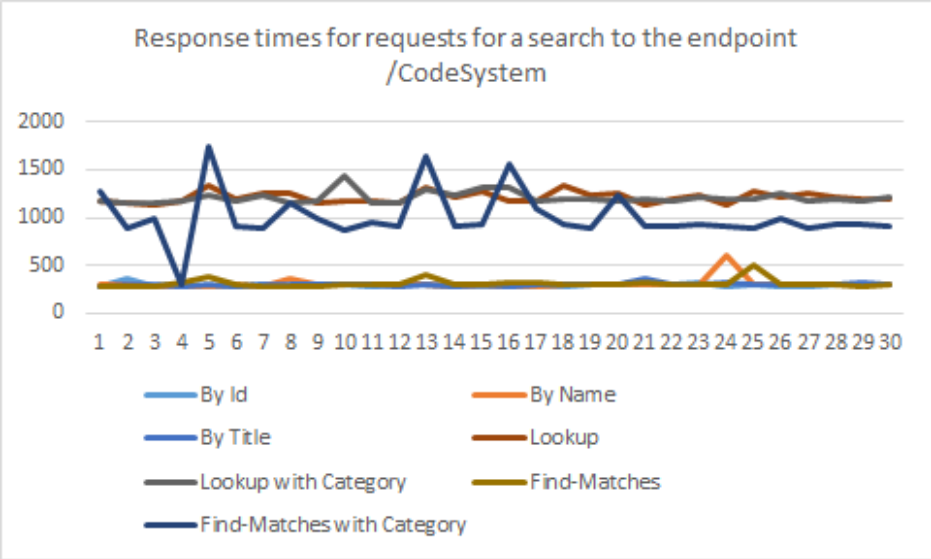


Figure 38 – Response times on searching on the /CodeSystem endpoint

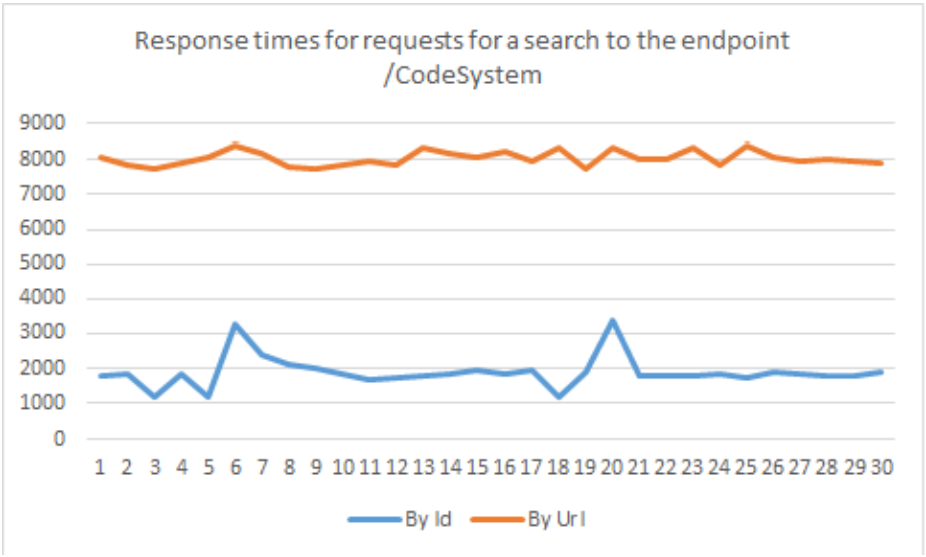


Figure 39 – Response time on searching on the /ValueSet endpoint

The final average time is around 1.6 seconds meaning that the requirement was achieved.

Table 27 – Average times for Search operations

Name	Request	Average	Metric
By Id	/CodeSystem/{id}	295,433	1618,6
By	/CodeSystem?name={name}	307,33	
By Title	/CodeSystem?title={title}	298,97	
Lookup	/CodeSystem/$lookup?system={system}&code={code}	1209,20	
Lookup with Categor	/CodeSystem/$lookup?system={system}&code={code}&category={category}	1207,80	
Find-matches	/CodeSystem/{id}/$find-matches?code={code}&display={display}	310,87	
Find-matches categor	/CodeSystem/{id}/$find-matches?code={code}&display={display}&category={category}	1007,67	
By Id	/ValueSet/{id}	1887,63	
By Url	/ValueSet/?url={url}	8042,40	

Relatively to the “PS03 - Has the responds in 2s or less for validation operations” requirement, Table 28 showcases the mean times in milliseconds. The values in these times are so alike that a line graphic would not provide any important information on the matter. The average result time for this ended up being 433.5 milliseconds, which is way smaller than the indicated metric, resulting in a successful completion of the requirement.

Table 28 – Average times for Validation operations

Request	Average	Metric
/CodeSystem/$validate-code?system={system}&code={code}	318,93	433,5
/CodeSystem/$validate-code?system={system}&code={code}&display={display}	300,53	
/CodeSystem/{id}/$validate-code?system={system}&code={code}	302,40	
/CodeSystem/{id}/$validate-code?system={system}&code={code}&display={display}	309,67	
/ValueSet/$validate-code?system={system}&code={code}	311,30	
/ValueSet/$validate-code?system={system}&code={code}&display={display}	318,63	
/ValueSet/{id}/$validate-code?system={system}&code={code}	298,00	
/ValueSet/{id}/$validate-code?system={system}&code={code}&display={display}	309,20	
/ValueSet/{id_category}/$validate-code?system={system}&code={code}	944,23	
/ValueSet/{id_category}/$validate-code?system={system}&code={code}&display={display}	922,03	

Related with “PS04 - Has the responds in 2s or less for subsumes operation” requirement, Figure 40 showcases times collected by the tests while Table 29 presents the average times in milliseconds.

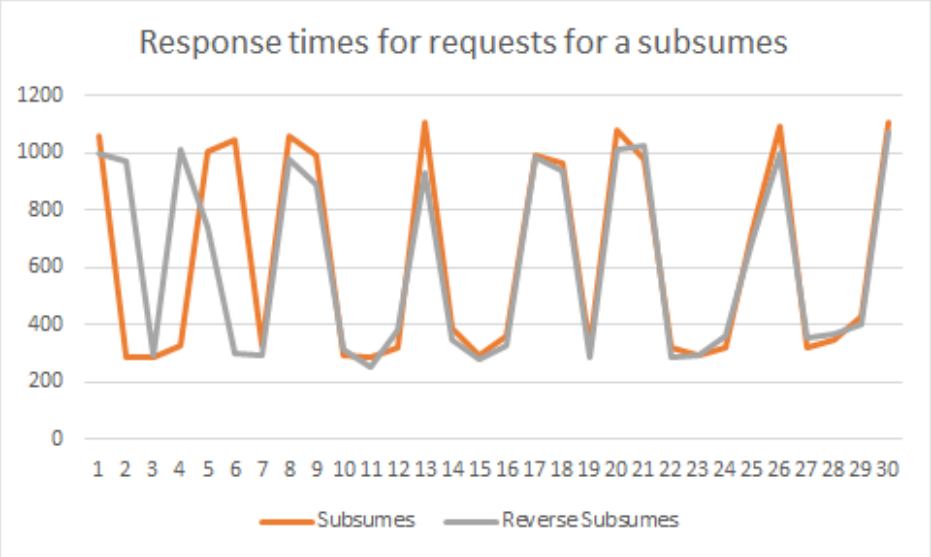


Figure 40 – Response times on subsumes operations

The final average time is around 600 milliseconds meaning that the requirement was fulfilled.

Table 29 – Average times for Subsumes operations

Request	**Average**	**Metric**
/CodeSystem/$subsumes?system={system}&codeA={codeA}&codeB={codeB}	624,87	619,2
REVERSE /CodeSystem/$subsumes?system={system}&codeA={codeA}&codeB={codeB}	613,53	

Finally, with relation with the “PS05 - Has the responds in 4s or less for expand operation” requirement, Figure 41 presents times collected by the tests while Table 30 demonstrates the average times in milliseconds.

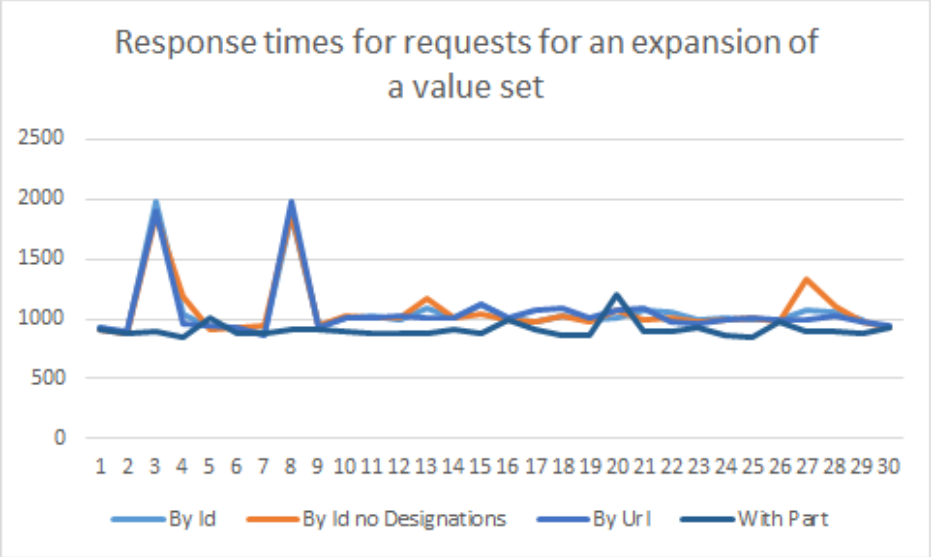


Figure 41 – Response times on expand operations

The final average time is around 1 second, so the requirement was completed.

Table 30 – Average times for Expand operations

Request	**Average**	**Metric**
/ValueSet/{id}/$expand	1060,67	1024,3
/ValueSet/{id}/$expand?includeDesignations=false	1069,37	
/ValueSet//$expand?url={url}	1057,47	
/ValueSet/{id-part}/$expand	909,70	

7.3.3.3 QEF Results

Translating these conclusions values to the QEF metrics, the following values were calculated in each factor and dimension, resulting in a final classification of 80% of the ideal product achieved.

Table 31 – QEF Last phase results

Factor	**Qj (%)**	**Dimension**	**Qi (%)**	**Q (%)**
Functionality	77,42	Functionality	77,42	80
Support	100	Usability and Reliability	100	
Consistency	100			
Integrity	100			
Security	100			
Speed	80	Performance, Supportability and Others	90	
Testability	100			
Configurability	100			
Scalability	100			
Knowledge	100			

8 Conclusion

This chapter presents an analyses of the results of the project according to the goals defined and results provided by the experiments, while making some considerations regarding possible improvements to the finished product.

8.1 Overview

By analyzing the results demonstrated in both testing phases, it is clear that there wasn't much difference in the amount of implemented features in the product. Some features were added, performance improved, and documentations and tests were created from the first to the last phase of testing, resulting in a 10% improvement of the product according to the requirements, metrics and indicators defined at the beginning of the project. The finished product is still far from the ideal solution to the problem but it is a major step to solving the problem at hand.

8.2 Future Improvements

After analyzing the success of the server, it is important to state which improvements could have been or can be done, in order to achieve the perfect desirable product.

In terms of design, it could have been a good decision to create more than one repository class, using the strategy design pattern. In order to delegate responsibilities better, a repository should have been created for dealing with data on the code systems, and two other repositories class for dealing with complex operations involving suffixes, one for categories and another for main ancestors. These repository should

extend the main one, who is responsible for taking care of all other operations, and delegating responsibilities to the other two for more complex ones. Dependencies between packages should also be reduced.

The authentication and authorization mechanism should also be improved in order to provide a token on the moment of authentication, which can later be used to authenticate other requests, instead of having to authenticate every request with an e-mail and a password. Other improvements are obviously to implement the remaining requirements that were not fulfilled so far.

8.3 Final Thoughts

At the end of the project, my opinion on the finished product of this project is overall positive, despite the perfect solution not being fully achieved. The finished product at the time of the experiments and evaluation second phase is still remarkably good for the time it had to grow. The initially thought to be the ideal solution and its requirements were too ambitious, even being defined by the developer itself. This represents a lesson for the future, not to underestimate work and not to overestimate capacities.

The extensive investigation and analyses here demonstrated represent one crucial step to the project, being able to provide important knowledge and promote a better understanding of the necessities and difficulties of the project. Moreover, it allowed for a thoughtful design and implementation decisions that contributed positively to the project. The design and development phases were the most challenging ones due to design implications that required smart decisions that could affect the success of the final product. Despite the improvements that need to be done, the server was designed and implemented with a constant growth mindset and it is ready to continue being improved overtime.

References

(ALERT Life Sciences Computing, 2010)	ALERT Life Sciences Computing, S. A. (2010) *Mission and values | ALERT® ONLINE*. Available at: http://www.alert-online.com/mission-values (Accessed: 20 January 2020).
(Benson and Grieve, 2016, p. 26)	Benson, T. and Grieve, G. (2016) *Principles of Health Interoperability*. Cham: Springer International Publishing (Health Information Technology Standards). doi: 10.1007/978-3-319-30370-3.
(Bodenreider, 2004)	Bodenreider, O. (2004) 'The Unified Medical Language System (UMLS): Integrating biomedical terminology', *Nucleic Acids Research*. Oxford University Press, 32(DATABASE ISS.). doi: 10.1093/nar/gkh061.
(Buchanan and Gardiner, 2003)	Buchanan, J. and Gardiner, L. (2003) 'A comparison of two reference point methods in multiple objective mathematical programming', *European Journal of Operational Research*, 149(1), pp. 17–34.
Chute (2000)	Chute, C. G. (2000) 'Clinical Classification and Terminology: Some History and Current Observations', *Journal of the American Medical Informatics Association*, 7(3), pp. 298–303. doi: 10.1136/jamia.2000.0070298.
(Computational Health Informatics Program, 2019a)	Computational Health Informatics Program (2019a) *SMART Health IT – Connecting health system data to innovators' apps*. Available at: https://smarthealthit.org/ (Accessed: 16 January 2020).
(Computational Health Informatics Program, 2019b)	Computational Health Informatics Program (2019b) *SMART on FHIR*. Available at: https://docs.smarthealthit.org/ (Accessed: 16 January 2020).
(CSIRO, 2019)	CSIRO (2019) *Our purpose - CSIRO*. Available at: https://www.csiro.au/en/About/We-are-CSIRO (Accessed: 8 January 2020).
(Evans, 2015, p. 6) (Evans, 2015, p. 17)	Evans, E. (2015) *Domain---Driven Design Reference Definitions and Pattern Summaries*. Available at: http://creativecommons.org/licenses/by/4.0/.ii (Accessed: 29 January 2020).
(Firely, no date a)	Firely (no date a) *About - FHIR*. Available at: https://fire.ly/about-firely/ (Accessed: 16 January 2020).
(Firely, no date b)	Firely (no date b) *FHIR .NET API - FHIR*. Available at: https://fire.ly/fhir-api/ (Accessed: 16 January 2020).
(Firely, no date c)	Firely (no date c) *Services - FHIR*. Available at: https://fire.ly/services/ (Accessed: 16 January 2020).
(Foundation, 2018)	Foundation, A. S. (2018) *Apache Subversion*. Available at: https://subversion.apache.org/ (Accessed: 18 November 2019).
(Fowler, 2004, p. 401)	Fowler, M. (2004) *Patterns of Enterprise Application Architecture*.
(Gallagher, Dunleavy and Reeves, 2019)	Gallagher, A., Dunleavy, J. and Reeves, P. (2019) *The Waterfall Methodology: Why You Should Use It*, *IBM Developer*. Available at: https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/ (Accessed: 14 November 2019).
(Gamma *et al.*, 1994, p. 144; Stelting and Maassen, 2002, p. 31) (Gamma *et al.*, 1994, p. 144; Stelting and Maassen, 2002, p. 98)	Gamma, E. *et al.* (1994) *Design patterns : elements of reusable object-oriented software*.
Geller *et al.* (2009)	Geller, J. *et al.* (2009) 'Comparing inconsistent relationship configurations indicating UMLS errors.', *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2009, pp. 193–7. Available at:

http://www.ncbi.nlm.nih.gov/pubmed/20351848 (Accessed: 8 January 2020).

(Grady and Caswell, 1987) Grady and Caswell (1987, p. 159) — Grady, R. B. and Caswell, D. L. (1987) *Software metrics : establishing a company-wide program*. Prentice-Hall, Inc.

(Health Level Seven International, 2019a) — Health Level Seven International (2019a) *2018 ANNUAL REPORT*. Available at: http://www.hl7.org/documentcenter/public/HL7/HL7 2018 Annual Report v9.pdf (Accessed: 9 January 2020).

(Health Level Seven International, 2019b) — Health Level Seven International (2019b) *About Health Level Seven International | HL7 International*. Available at: http://www.hl7.org/about/index.cfm?ref=footer (Accessed: 30 December 2019).

(Health Level Seven International, 2019c) — Health Level Seven International (2019c) *CodeSystem - FHIR v4.0.1*. Available at: http://hl7.org/fhir/codesystem.html (Accessed: 7 January 2020).

(Health Level Seven International, 2019d) — Health Level Seven International (2019d) *ConceptMap - FHIR v4.0.1*. Available at: http://hl7.org/fhir/conceptmap.html (Accessed: 7 January 2020).

(Health Level Seven International, 2019e) — Health Level Seven International (2019e) *FHIR Specification*. Available at: http://hl7.org/fhir/directory.html (Accessed: 2 January 2020).

(Health Level Seven International, 2019f) — Health Level Seven International (2019f) *HL7 Standards Product Brief - FHIR® R4 (HL7 Fast Healthcare Interoperability Resources, Release 4) | HL7 International*. Available at: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=491 (Accessed: 30 December 2019).

(Health Level Seven International, 2019g) — Health Level Seven International (2019g) *Summary - FHIR v4.0.1*. Available at: http://hl7.org/fhir/summary.html (Accessed: 2 January 2020).

(Health Level Seven International, 2019h) — Health Level Seven International (2019h) *Terminology-module - FHIR v4.0.1.* Available at: http://hl7.org/fhir/terminology-module.html (Accessed: 7 January 2020).

(Health Level Seven International, 2019i) — Health Level Seven International (2019i) *Terminology-service - FHIR v4.0.1.* Available at: https://www.hl7.org/fhir/terminology-service.html (Accessed: 7 January 2020).

(Health Level Seven International, 2019j) — Health Level Seven International (2019j) *ValueSet - FHIR v4.0.1*. Available at: http://hl7.org/fhir/valueset.html (Accessed: 7 January 2020).

(Health Level Seven International, 2019k) — Health Level Seven International (2019k) *Versions - FHIR v4.0.1*. Available at: https://www.hl7.org/fhir/versions.html (Accessed: 3 January 2020).

(Health Level Seven International, no date) — Health Level Seven International (no date) *Application Registry*. Available at: http://www.fhir.org/implementations/registry/ (Accessed: 9 January 2020).

Healthcare Information and Management Systems Society (HIMSS) (no date) — Healthcare Information and Management Systems Society (no date) *Interoperability and Health Information Exchange | HIMSS*. Available at: https://www.himss.org/interoperability-and-health-information-exchange (Accessed: 27 December 2019).

(Hedden, 2010) — Hedden, H. (2010) 'Taxonomies and controlled vocabularies best practices for metadata', *Journal of Digital Asset Management*. Springer Nature, 6(5), pp. 279–284. doi: 10.1057/dam.2010.29.

(Hufnagel, 2009) — Hufnagel, S. P. (2009) 'Interoperability', *Military Medicine*, 174(5S), pp. 43–50. doi: 10.7205/MILMED-D-03-9808.

(Hussain, Langer and Kohli, 2018) — Hussain, M. A., Langer, S. G. and Kohli, M. (2018) 'Learning HL7 FHIR Using the HAPI FHIR Server and Its Use in Medical Imaging with the SIIM Dataset', *Journal of Digital Imaging*. Springer New York LLC, 31(3), pp. 334–340. doi: 10.1007/s10278-018-0090-y.

(Jacobson, 1992) — Jacobson, I. (1992) *Object-oriented software engineering : a use case driven approach*. ACM Press.

(Jacobson, Spence and Bittner, 2011)	Jacobson, I., Spence, I. and Bittner, K. (2011) *USE-CASE 2.0 The Guide to Succeeding with Use Cases*.
(James Agnew, 2019)	James Agnew (2019) *News: The HAPI FHIR Blog*. Available at: https://hapifhir.io/hapi-fhir/blog/?page=0 (Accessed: 15 January 2020).
(Koen et al., 2002)	Koen, P. A. *et al.* (2002) 'Fuzzy Front End: Effective Methods, Tools, and Techniques', *The PDMA Toolbook*, 1(for new product development), pp. 5–35. Available at: http://www.stevens.edu/cce/NEW/PDFs/FuzzyFrontEnd_Old.pdfNEW/PDFs/FuzzyFrontEnd_Old.pdf.
(Larman, 2004)	Larman, C. (2004) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition*.
(Mandl and Kohane,	Mandl, K. D. and Kohane, I. S. (2009) 'No small change for the health information economy', *New England Journal of Medicine*. Massachussetts Medical Society, 360(13), pp. 1278–1281. doi: 10.1056/NEJMp0900411.
Myers and Sandler (2004, p. 150) (Myers and Sandler,	Myers, G. J. and Sandler, C. (2004) *The Art of Software Testing, Second Edition*. Available at: www.Wiley.com. (Accessed: 31 January 2020).
(Nicola, 2019)	Nicola, S. (2019) 'Análise_Valor_Aula_2_22OUT_2019_1hora_AHP'.
(Nicola, Ferreira and Ferreira, 2012)	Nicola, S., Ferreira, E. P. and Ferreira, J. J. P. (2012) 'A novel framework for modeling value for the customer, an essay on negotiation', *International Journal of Information Technology and Decision Making*, 11(3), pp. 661–703. doi: 10.1142/S0219622012500162.
(Nowlan, 1993)	Nowlan, W. A. (1993) *Structured Methods of Information Management for Medical Records*.
(Nuno Escudeiro, 2019)	Nuno Escudeiro (2019) '2_Qualidade-QEF'.
(OpenGALEN, 1999a)	OpenGALEN (1999a) *Overall purpose of the Model*. Available at: http://www.opengalen.org/themodel/purpose.html (Accessed: 8 January 2020).
(OpenGALEN, 1999b)	OpenGALEN (1999b) *Structure of the Model*. Available at: http://opengalen.org/themodel/structure.html (Accessed: 7 January 2020).
(Oracle and/or its affiliates, 2020)	Oracle and/or its affiliates (2020) *Securing RESTful Web Services*. Available at: https://docs.oracle.com/cd/E24329_01/web.1211/e24983/secure.htm#RESTF113 (Accessed: 24 January 2020).
(Oracle and/or its affiliates, no date)	Oracle and/or its affiliates (no date) *The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 339*. Available at: https://jcp.org/en/jsr/detail?id=339 (Accessed: 24 January 2020).
(Paul et al., 2007)	Paul, R. *et al.* (2007) *The thinker's guide to engineering reasoning*. Foundation for Critical Thinking.
(Pivotal Software, 2020)	Pivotal Software, I. (2020) *Spring Boot*. Available at: https://spring.io/projects/spring-boot (Accessed: 29 January 2020).
Posnack and Barker, 2018	Posnack, S. and Barker, W. (2018) 'Heat Wave: The U.S. is Poised to Catch FHIR in 2019 | Health IT Buzz', *Health IT Buzz*, 1 October. Available at: https://www.healthit.gov/buzz-blog/interoperability/heat-wave-the-u-s-is-poised-to-catch-fhir-in-2019 (Accessed: 9 January 2020).
(Rational Software, 2011)	Rational Software (2011) *Rational Unified Process Best Practices for Software Development Teams*. Available at: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf (Accessed: 14 November 2019).
(Rector, 1999)	Rector, A. L. (1999) 'Clinical Terminology: Why is it so hard?', *Methods of Information in Medicine*, 38(4–5), pp. 239–252. doi: 10.1055/s-0038-1634418.

(Regenstrief Institute, 2019a)	Regenstrief Institute, I. (2019a) *Developer Reference for LOINC in FHIR – LOINC.* Available at: https://loinc.org/fhir/dev/ (Accessed: 8 January 2020).
(Regenstrief Institute, 2019b)	Regenstrief Institute, I. (2019b) *LOINC FHIR Terminology Server – LOINC*. Available at: https://loinc.org/fhir/ (Accessed: 8 January 2020).
(Rich and Holweg, 2000)	Rich, N. and Holweg, M. (2000) *Value Analysis Value Engineering*. Cardiff, United Kingdom.
(Rogers et al., 2001)	Rogers, J. *et al.* (2001) 'GALEN ten years on: Tasks and supporting tools', in *Studies in Health Technology and Informatics*. IOS Press, pp. 256–260. doi: 10.3233/978-1-60750-928-8-256.
(S. O'Dea, 2019)	S. O'Dea (2019) *Number of Internet of Things (IoT) active connections in healthcare in the European Union (EU) in 2016, 2019, 2022 and 2025*. Available at: https://www.statista.com/statistics/691848/iot-active-connections-in-healthcare-in-the-eu/ (Accessed: 20 January 2020).
(Saaty, 1987)	Saaty, R. W. (1987) 'The analytic hierarchy process-what it is and how it is used', *Mathematical Modelling*, 9(3–5), pp. 161–176. doi: 10.1016/0270-0255(87)90473-8.
(Saaty, 1990)	Saaty, T. L. (1990) 'How to make a decision: The analytic hierarchy process', *European Journal of Operational Research*, 48(1), pp. 9–26. doi: 10.1016/0377-2217(90)90057-I.
(Simpatico Intelligent Systems Inc., 2020a)	Simpatico Intelligent Systems Inc. (2020a) *Enterprise-Class FHIR Clinical Data Repository | Smile CDR*. Available at: https://smilecdr.com/technology.html (Accessed: 15 January 2020).
(Simpatico Intelligent Systems Inc., 2020b)	Simpatico Intelligent Systems Inc. (2020b) *FAQ: Smile CDR*. Available at: https://smilecdr.com/faq.html (Accessed: 15 January 2020).
(Simpatico Intelligent Systems Inc., 2020c)	Simpatico Intelligent Systems Inc. (2020c) *For Business: Improve Interoperability in Healthcare Systems | Smile CDR*. Available at: https://smilecdr.com/business.html (Accessed: 15 January 2020).
(Simpatico Intelligent Systems Inc., 2020d)	Simpatico Intelligent Systems Inc. (2020d) *Implementation, Integration, & Support Services | Smile CDR*. Available at: https://smilecdr.com/services.html (Accessed: 15 January 2020).
(Simpatico Intelligent Systems Inc., 2020e)	Simpatico Intelligent Systems Inc. (2020e) *SMART on FHIR: Introduction - Smile CDR Documentation*. Available at: https://smilecdr.com/docs/security/smart_on_fhir_introduction.html (Accessed: 16 January 2020).
(Simpatico Intelligent Systems Inc., 2020f)	Simpatico Intelligent Systems Inc. (2020f) *Table of Contents - Smile CDR Documentation*. Available at: https://smilecdr.com/docs/ (Accessed: 15 January 2020).
(SNOMED International, 2019a)	SNOMED International (2019a) *Expression Constraint Language - Specification and Guide - Expression Constraint Language - SNOMED Confluence*. Available at: https://confluence.ihtsdotools.org/display/DOCECL/Expression+Constraint+Language+-+Specification+and+Guide (Accessed: 8 January 2020).
(SNOMED International, 2019b)	SNOMED International (2019b) *Features of Known Servers - SNOMED on FHIR - SNOMED Confluence*. Available at: https://confluence.ihtsdotools.org/display/FHIR/Features+of+Known+Servers (Accessed: 8 January 2020).
(SNOMED International, 2019c)	SNOMED International (2019c) *GitHub - IHTSDO/snowstorm: Scalable SNOMED CT Terminology Server using Elasticsearch*. Available at: https://github.com/IHTSDO/snowstorm (Accessed: 8 January 2020).
(SNOMED International, 2019d)	SNOMED International (2019d) *SNOMED CT Browser - Home*. Available at: https://browser.ihtsdotools.org/? (Accessed: 8 January 2020).
(Spring Team, 2016)	Spring Team (2016) *Core Technologies*. Available at: docs.spring.io/spring/docs/current/spring-framework-reference/core.html#beans-introduction (Accessed: 27 March 2020).

(Spring Team, no date)	Spring Team (no date) *JdbcTemplate (Spring Framework 5.2.7.RELEASE API*). Available at: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html (Accessed: 12 May 2020).
(Gamma et al., 1994, p. 144; Stelting and Maassen, 2002, p. 31) (Stelting and Maassen, 2002, p. 81) (Stelting and Maassen, 2002, p. 98)	Stelting, S. and Maassen, O. (2002) *Applied Java Patterns*. 1st edn. Hall, Prentice.
(Substitutable Medical Applications and Reusable Technologies, 2019)	Substitutable Medical Applications and Reusable Technologies (2019) *SMART on FHIR API – SMART Health IT*. Available at: https://smarthealthit.org/smart-on-fhir-api/ (Accessed: 16 January 2020).
(TEAM, 2019)	TEAM, T. T. (2019) *Home · TortoiseSVN*. Available at: https://tortoisesvn.net/ (Accessed: 18 November 2019).
(The McGraw-Hill Companies, 2003)	The McGraw-Hill Companies, I. (2003) 'Quality Function Deployment'. McGraw-Hill/Irwin.
(The McGraw-Hill Companies, 2013)	The McGraw-Hill Companies, I. (2013) 'QFD: Product Excellence using Six Sigma: QFD', pp. 1–61.
Tolk, Diallo and Turnitsa (2007)	Tolk, A., Diallo, S. Y. and Turnitsa, C. D. (2007) 'Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability, and Composability for System-of-Systems Engineering', *Journal of Systemics, Cybernetics, and Informatics*, 5(5), pp. 65–74. Available at: https://digitalcommons.odu.edu/msve_fac_pubs (Accessed: 27 December 2019).
(Ulaga and Eggert, 2006)	Ulaga, W. and Eggert, A. (2006) 'Relationship value and relationship quality: Broadening the nomological network of business-to-business relationships', *European Journal of Marketing*, 40(3–4), pp. 311–327. doi: 10.1108/03090560610648075.
(University Health Network, 2019)	University Health Network (2019) *Table of Contents - HAPI FHIR Documentation*. Available at: https://hapifhir.io/hapi-fhir/docs (Accessed: 15 January 2020).
(Woodall, 2003)	Woodall, T. (2003) *Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis is Senior Lecturer in Quality Management and Marketing, Department of Strategic Management and Market-ing*. Available at: http://www.amsreview.org/articles/woodall12-2003.pdf (Accessed: 21 January 2020).
(Zeithaml, 1988)	Zeithaml, V. A. (1988) 'Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence', *Journal of Marketing*. SAGE Publications, 52(3), p. 2. doi: 10.2307/1251446.

Appendix A – Criteria Consistency

Supports FHIR specification

The priority vector is:

$$\begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix}$$

Following the formulas:

$$Criteria\ Matrix \times Eigenvector \cong \lambda max \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 1 & 1 & 3 \\ 1 & 1 & 1 & 3 \\ 1 & 1 & 1 & 3 \\ 1/3 & 1/3 & 1/3 & 1 \end{bmatrix} \times \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 1.2 \\ 1.2 \\ 1.2 \\ 0.4 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$$\Leftrightarrow \lambda max \cong 4.04$$

With the λ_{max} value is now possible to calculate the IC, and then the RC. Since there is 4 tools, n = 4 and RI = 0.90.

$$IC = \frac{\lambda \max - n}{n - 1} \qquad CR = \frac{IC}{RC}$$

$$\Leftrightarrow IC = \frac{4 - 4}{4 - 1} \qquad \Leftrightarrow CR = \frac{0}{0.90}$$

$$\Leftrightarrow IC = \frac{0}{3} \qquad \Leftrightarrow CR = 0$$

$$\Leftrightarrow IC = 0$$

Since RC = 0 (0%) and 0 < 0.1 (10%), the priority values are consistent.

Implementation

The priority vector is:

$$\begin{bmatrix} 0.39 \\ 0.07 \\ 0.15 \\ 0.39 \end{bmatrix}$$

Following the formulas:

$$Criteria\ Matrix \times Eigenvector \cong \lambda max \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 5 & 3 & 1 \\ 1/5 & 1 & 1/3 & 1/5 \\ 1/3 & 3 & 1 & 1/3 \\ 1 & 5 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 0.39 \\ 0.07 \\ 0.15 \\ 0.39 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.39 \\ 0.07 \\ 0.15 \\ 0.39 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 1.58 \\ 0.28 \\ 0.62 \\ 1.58 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.39 \\ 0.07 \\ 0.15 \\ 0.39 \end{bmatrix}$$

$$\Leftrightarrow \lambda max \cong 4.06$$

With the λ_{max} value is now possible to calculate the IC, and then the RC. Since there is 4 alternatives, n = 4 and RI = 0.90.

$$IC = \frac{\lambda \max - n}{n - 1} \qquad CR = \frac{IC}{RC}$$

$$\Leftrightarrow IC = \frac{4.06 - 4}{4 - 1} \qquad \Leftrightarrow CR = \frac{0.02}{0.90}$$

$$\Leftrightarrow IC = \frac{0.06}{3} \qquad \Leftrightarrow CR = 0.02$$

$$\Leftrightarrow IC = 0.02$$

Since RC = 0.02 (2%) and 0.02 < 0.1 (10%), the priority values are consistent.

Maintainability

The priority vector is:

$$\begin{bmatrix} 0.52 \\ 0.13 \\ 0.04 \\ 0.30 \end{bmatrix}$$

Following the formulas:

$$Criteria\ Matrix \times Eigenvector \cong \lambda max\ \times\ Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 5 & 9 & 2 \\ 1/5 & 1 & 4 & 1/3 \\ 1/9 & 1/4 & 1 & 1/7 \\ 1/2 & 3 & 7 & 1 \end{bmatrix} \times \begin{bmatrix} 0.52 \\ 0.13 \\ 0.04 \\ 0.30 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.52 \\ 0.13 \\ 0.04 \\ 0.30 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 2.13 \\ 0.50 \\ 0.17 \\ 1.23 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.52 \\ 0.13 \\ 0.04 \\ 0.30 \end{bmatrix}$$

$$\Leftrightarrow \lambda max \cong 4.07$$

With the λ_{max} value is now possible to calculate the IC, and then the RC. Since there is 4 tools, n = 4 and RI = 0.90.

$$IC = \frac{\lambda \max - n}{n - 1}$$

$$\Leftrightarrow IC = \frac{4.07 - 4}{4 - 1}$$

$$\Leftrightarrow IC = \frac{0.07}{3}$$

$$\Leftrightarrow IC = 0.023$$

$$CR = \frac{IC}{RC}$$

$$\Leftrightarrow CR = \frac{0.023}{0.90}$$

$$\Leftrightarrow CR = 0.03$$

Since RC = 0.03 (3%) and 0.03 < 0.1 (10%), the priority values are consistent.

Security

The priority vector is:

$$\begin{bmatrix} 0.13 \\ 0.57 \\ 0.24 \\ 0.06 \end{bmatrix}$$

Following the formulas:

$$Criteria\ Matrix \times Eigenvector \cong \lambda max\ \times\ Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & {}^{1}/_{5} & {}^{1}/_{2} & 3 \\ 5 & 1 & 3 & 7 \\ 2 & {}^{1}/_{3} & 1 & 5 \\ {}^{1}/_{3} & {}^{1}/_{7} & {}^{1}/_{5} & 1 \end{bmatrix} \times \begin{bmatrix} 0.13 \\ 0.57 \\ 0.24 \\ 0.06 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.13 \\ 0.57 \\ 0.24 \\ 0.06 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 0.54 \\ 2.36 \\ 0.99 \\ 0.23 \end{bmatrix} \cong \lambda max \begin{bmatrix} 0.13 \\ 0.57 \\ 0.24 \\ 0.06 \end{bmatrix}$$

$$\Leftrightarrow \lambda max \cong 4.06$$

With the λ_{max} value is now possible to calculate the IC, and then the RC. Since there is 4 alternatives, n = 4 and RI = 0.90.

$$IC = \frac{\lambda \max - n}{n - 1}$$

$$\Leftrightarrow IC = \frac{4.06 - 4}{4 - 1}$$

$$\Leftrightarrow IC = \frac{0.06}{3}$$

$$\Leftrightarrow IC = 0.02$$

$$CR = \frac{IC}{RC}$$

$$\Leftrightarrow CR = \frac{0.02}{0.90}$$

$$\Leftrightarrow CR = 0.02$$

Since RC = 0.02 (2%) and 0.02 < 0.1 (10%), the priority values are consistent

Appendix B – FHIR R4 data model with selected extensions

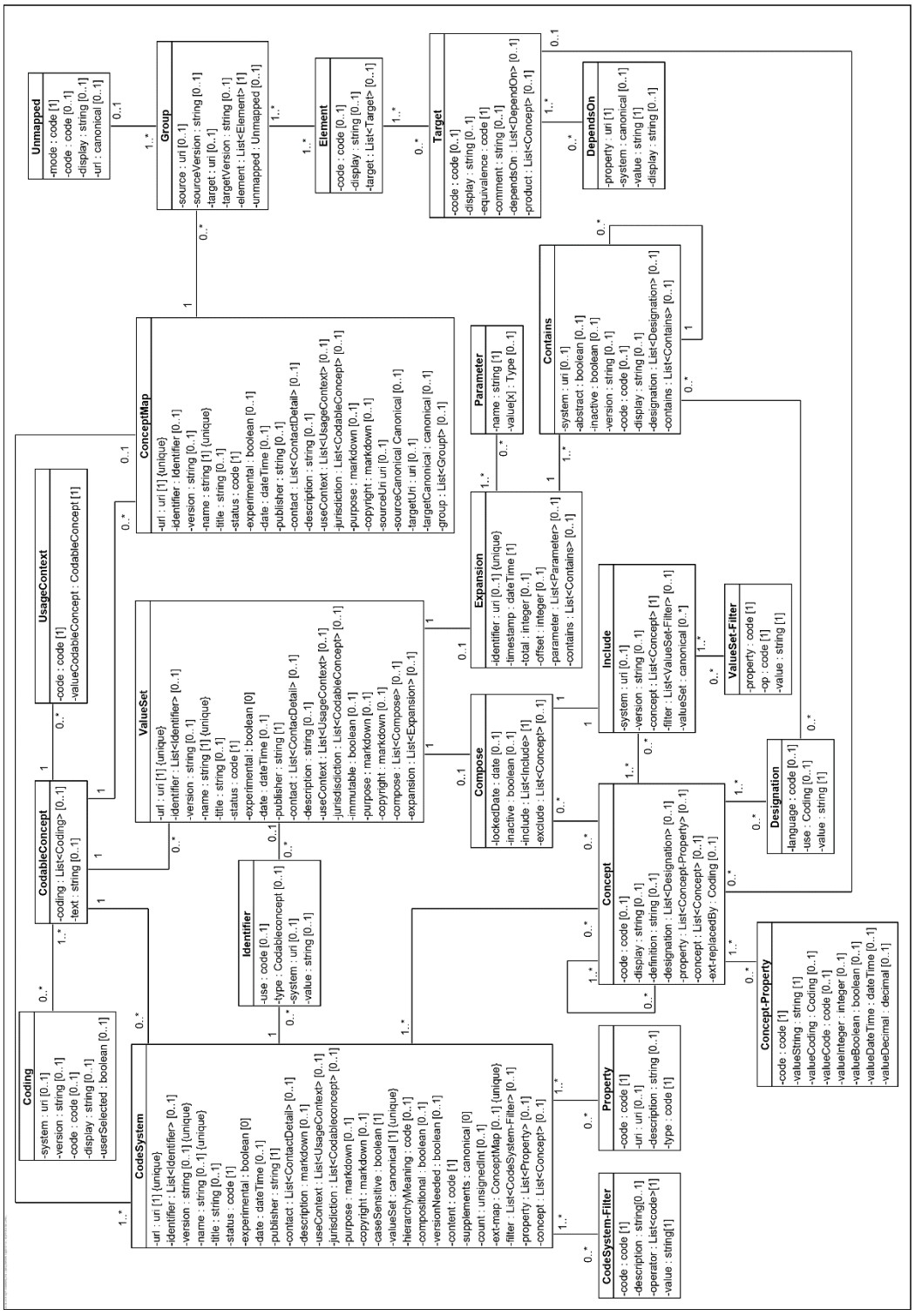


Figure 42 – FHIR R4 data model with attributes and selected extensions

Appendix C – Quantitative Evaluation Framework

Figure 43 displays the functionality requirements and respective metrics. These all fall under the Functional factor. Figure 44 displays the Usability and Reliability requirements and metrics. Under the usability dimension (red) there is the support factor, while in the reliability (green) there are the consistency, integrity and security factors. Finally, the last dimension (Figure 45) is the Performance (purple), supportability (orange) and others (blue), with speed, testability, configurability, scalability and knowledge factors.

Dimension	Functionality					
Factor	Functional					
		Wfk - Fullfilment (%)				
Requirement	Metric Evaluation	0	25	50	75	100
FF01 - Connects to a database	The server connects to a database that contains the terminologies basic information, their content, concept categories, languages and ALERT contexts	No	-	-	-	Yes
FF02 - List the server's capabilities	The server lists its capabilities	No	-	-	-	Yes
FF03 - List all code systems	The server lists all code systems and their attributes contained in the database	No	-	-	-	Yes
FF04 - Display a code system's structure	The server displays a code system's structure in a way interpretable by a FHIR client	No	-	-	-	Yes
FF05 - Validate if a concept belongs to a code system	The server allows validation of concepts within code systems	No	-	-	-	Yes
FF06 - Search a code system by name	The server allows to search for a code system by its name	No	-	-	-	Yes
FF07 - Search a code system by other properties	The server allows to search for a code system by its properties	No properties	-	Only one property at a time	-	Multiple properties simultaneously
FF08 - List all concepts from a code system	The server is able to list all concepts and their properties from a code system, in a single or multiple value sets	No concepts	Less than half of the concepts	Half of the concepts	More than half of the concepts	All concepts
FF09 - List all concepts of the same category and code system	The server is able to list all concepts and their properties from a single concept category within a code system in a single value set	No concepts	Less than half of the concepts	Half of the concepts	More than half of the concepts	All concepts
FF10- Validate if a concept belongs to a category	The server allows validation of concepts within concept categories	No	-	-	-	Yes
FF11 - List all categories	The server lists all concept categories contained in the database	No	-	-	-	Yes
FF12 - Search categories by name	The server allows to search for a concept category by its name	No	-	-	-	Yes
FF13 - List relationships between concepts	The server is capable of listing relationships between concepts	No	-	-	-	Yes
FF14 - Search a concept by its code, term or other properties	The server allows to search for a concept by its properties	No properties	-	Only one property at a time	-	Multiple properties simultaneously
FF15 - Pagination	The server is capable of displaying paginated resources	No	-	-	-	Yes
FF16 - Authentication	The server is capable of authenticating an user	No	-	-	-	Yes
FF17 - Authorization	The server is capable of verifying an user's authorization to access data	No	-	-	-	Yes

Figure 43 – QEF - Functional requirements and metrics

Dimension	Usability and Reliability								
Factor	Support, Consistency, Integrity, Security								
			Wfk - Fullfilment (%)						
Requirement	Metric Evaluation	0		25		50		75	100
US01 - Warns against unexpected and unresolved requests	The server warns the client against unexpected and unresolved requests	No		-		-		-	Yes
US02 - Provides information for usage	The system comes with extra information about its functionality and usage	No		-		-		-	Yes
RC01 - Works continuously without failures	The server works continuously without failure or error	Fails		-		-		-	Does not fail
RI01 - Ensures data integrity	The server displays accurate and consistent data according to the database	No		-		-		-	Yes
RI02 - Prevents against data repetition	The server does not display repeated data in the same resource	Repeated data		-		-		-	No repeated data
RS01 - Is accessed through a Web Client	A generic web client can access the server by HTTPS	No access		-		Access only by HTTP		-	Access by HTTPS

Figure 44 - QEF – Usability and Reliability requirements and metrics

Dimension	Performance, Supportability and Others						
Factor	Speed, Testability, Configurability, Scalability, Knowledge						
			Wrk - Fulfilment (%)				
Requirement	Metric Evaluation	0	25	50	75	100	
PS01 - Has the responds in 5s or less for listing operations	The server responds to a request of listing all code systems or value sets in 5 seconds or less	No	-	-	-	Yes	
PS02 - Has the responds in 2s or less for searching operations	The server responds to a request for a search, including find-matches and lookup operations in 2 seconds or less	No	-	-	-	Yes	
PS03 - Has the responds in 2s or less for validation operations	The server responds to a request for a validation in 2 seconds or less	No	-	-	-	Yes	
PS04 - Has the responds in 2s or less for subsumes operation	The server responds to a request for a subsumes operation in 2 seconds or less	No	-	-	-	Yes	
PS05 - Has the responds in 4s or less for expand operation	The server responds to a request for an expand operation in 4 seconds or less	No	-	-	-	Yes	
ST01 - Provides a set of tests	The server comes with a set of tests to test and server has an example for clients	No tests	Tests for less than 50% of the server's capabilities	Tests for 50% of the server's capabilities	Tests for more than 50% and less of 100% of the server's capabilities	Tests for 100% of the server's capabilities	
SC01 - Provides support for XML and JSON responses	The server is able to represent resources in XML and JSON	None of the formats	-	Only one format	-	Both formats	
SC02 - The format of the responses is configurable	The server is able to represent resources in XML or JSON based on a configuration	No	-	-	-	Yes	
SS01 - Is scalable	The server was built with a good architectural structure, following design patterns and engineering best practices	No	-	-	-	Yes	
OK01 - The development of the solution gives preference to technologies within ALERT's knowhow. If not, proper justification must be presented	The development of the solution gives preference to Oracle technologies - Java and SQL. If not, proper justification must be presented	No preference or justification	-	-	-	Gives preference or proper justification	

Figure 45 – QEF - Performance, Supportability and Others requirements and metrics