UNIVERSITETET I AGDER

FEST on FHIR

A Standardized Service Interface for Norwegian Medicines Registry

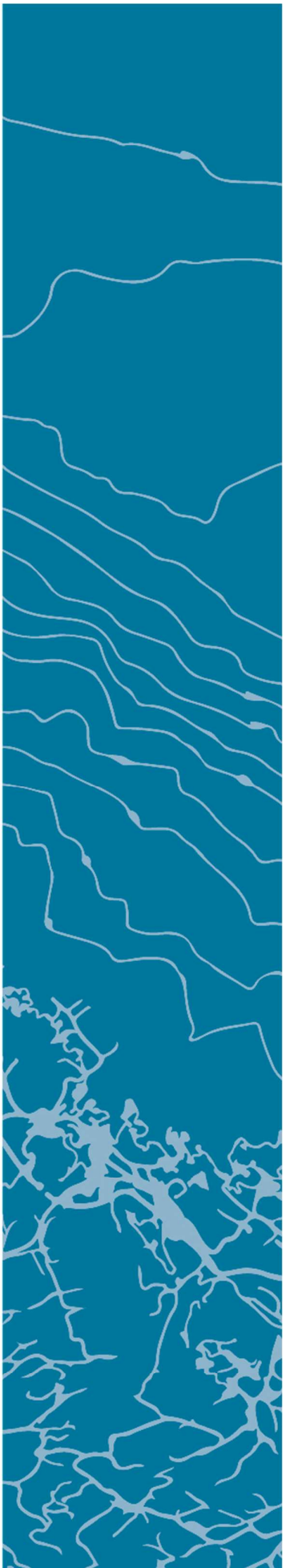
Islam Fathi Hussein Al Khaldi

Supervisors

Associate Professor Jan Pettersen Nytun

Assistant Professor Martin Wulf Gerdes

University of Agder, 2018
Faculty of Engineering and Science
Department ICT



UNIVERSITY OF AGDER

FEST on FHIR

A Standardized Service Interface for Norwegian Medicines Registry

by

Islam Fathi Hussein Al Khaldi

Supervisor: Associate Professor Jan Pettersen Nytun

Supervisor: Assistant Professor Martin Wulf Gerdes

Master Thesis
IKT 590

University of Agder
Faculty of Engineering and Science

Grimstad, 4 June 2018

Abstract

The Norwegian Electronic Prescription Support System (FEST) is a Norwegian drugs database in XML format offered by the Norwegian Medicines Agency (SLV). FEST includes all information about drugs that can be prescribed electronically in Norway. FEST is used as a pharmaceutical data source by practitioners, pharmaceutical companies, and hospitals. The Norwegian hospitals are divided into four health regions. Each region has its own drug registry, but they use FEST as a resource of drug information and take the responsibility of updating their own systems with the latest up-to-date version of FEST.

Having access to one common national source of information about drugs which is always up-to-date, relevant and quality assured would allow a more efficient development of interoperable systems and solutions based on one well defined, standardized interface and protocol. Our main goal of this thesis is to automatically map FEST XML into Fast Healthcare Interoperability Resources (HL7 FHIR). Thereby, drug information will always become available/accessible through a REST-based service, avoiding that the hospitals have to download the latest up-to-date version of FEST periodically.

Our first step is to use Java XML integration technology to automatically map FEST XML data model into a Java classes model. The outcome of this step is that all FEST data elements will be accessible as Java instances of its corresponding generated Java classes. The next step is to define HL7 FHIR profiles for FEST data and implement these profiles on a RESTful FHIR server where FEST data will be stored as instances of HL7 FHIR resources. The last step is to develop Web-based back-end components to access the RESTful service developed in the former step.

As a proof of concept, we developed a front-end interface to provide web-based access to our FEST on FHIR resources, and a use case of drug-drug interaction detection was applied as well. The result of our solution is that the latest version of FEST data can be represented in terms of HL7 FHIR international standards and is always available/accessible through FHIR RESTful service. Consequently, the Norwegian hospitals may have more opportunity to develop more efficient interoperable systems by using FEST data available through the FHIR RESTful service.

Keywords: *HL7, FHIR, FEST, RESTful, Medicine Norway, Health Information Services (HIS) Infrastructure, Norwegian Health Network (NHN)*

Preface

This thesis represents the result of the work that has been taken in order to fulfil the requirement of IKT 590 master's thesis course, 30 ECTS, at the faculty of engineering and science, department of information and communication technology (ICT), University of Agder (UiA), Grimstad, Norway.

The work has started on 15th January 2018 and ended on 4th June 2018.

The main goal behind this work is to automatically map The Norwegian Electronic Prescription Support System (FEST) into HL7 FHIR resources and provide a RESTful service where FEST data become accessible. As a result, various healthcare information systems will obtain an instant access to an up-to-date FEST data.

I would like to show gratefulness and appreciation to my thesis supervisors, Associate Professor Jan Pettersen Nytun, and Assistant Professor Martin Gerdes for all the help, guidance, encouragement and support through the period of this thesis work.

Special thanks go to my mother for her prayers and my father who is my source of enthusiasm.

Special thanks go to my wife and kids for their support and patience.

Special thanks go to my sister for her support.

Special thanks go to my brother for his guidance and encouragement.

I would like to thank my friends and classmates for being them.

Islam Fathi Hussein Al Khaldi

4th June 2018

University of Agder

Grimstad, Norway

Table of Figures

List of Tables

List of Abbreviations

AMR	Automated Medical Records
API	Application Programming Interface
ATC	Anatomical therapeutic chemical
CDA	Clinical Document Architecture
CPM	Common Product Model
CRUD	Create Read Update Delete Operations
DDI	Drug-drug Interaction
EHCR	Electronic Healthcare Records
EHR	Electronic Health Record
EMR	Electronic Medical Records
ePrescribing	Electronic Prescribing
FEST	Norwegian Electronic Prescription Support System Forskrivnings- og ekspedisjonsstøtte
FHIR	Fast Healthcare Interoperability Resources
GP	General Practitioner
HL7	Health Level Seven International
HTTP	Hypertext transfer protocol
ICT	Information and communication technology
IDMP	Identification of Medicinal Products
IHE	Interoperability in eHealth Industry
ISO	International Organization of Standardization
IT	Information technology
JAXB	Java Architecture for XML Binding
JDK	Java Development Kit
NDE	Norwegian Directorate of eHealth
NHN	Norwegian Health Network Norsk Helsenett
NUFA	Academic committee for health and architecture in Norway
OOM	Object-Oriented Modelling
OSI	Open System Interconnections
OWL	Web Ontology Language
POM	Project Object Model
REST	Representational State Transfer
RIM	Reference Information Model
SLV	Norwegian Medicines Agency Statens Legemiddelverk
UML	Unified Modelling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

Table of Contents

1 Introduction

Proper electronic exchange of medical information provides patients, practitioners, pharmacists and all other healthcare professionals with the ability to easy access and securely share medical information of patients. Consequently, advancing medical systems' interoperability will expand patient's care quality and reduce the risk of patient injuries and deaths related to incorrect medication. Standardization supports interoperability of medical information systems in many aspects; meaning by using standardized healthcare vocabularies, information and message structures as standardized by Health Level Seven International (HL7), information transfer by using secure protocols, and availability/accessibility by using services [1].

The field of eHealth is an emerging field where information and communication technologies (ICTs) are used to support health and its related areas such as healthcare, health surveillance and health education [2]. According to the Norwegian Directorate of eHealth (NDE), ICT systems which are used today by healthcare professionals still need more development/improvement to follow the patient throughout the patient's course completely [3]. NDE claims that one of the reasons is that the healthcare sector in Norway consists of many independent organizations which are responsible for the priorities, procurement, and operation of their own systems. Consequently, there are many individual and different solutions in operation [3].

The ministry of health and care services in Norway has the role of the responsible supervisor for hospitals which are organised into four regional health organizations. Under each one of these four regional health organizations, there are many hospitals. Each hospital covers a specific local area and maybe distributes over a number of local hospitals. Each local hospital consists of different departments, centres and clinics [4]. In addition to local hospitals, there are specialist health services which include psychiatric hospitals, training and rehabilitation institutions, institutions specialized in drug abuse treatment, prehospital services, private practitioners and laboratory [5].

In the current structure of health and care services in Norway, a major need for a common national journal solution for each patient has emerged and a project called "Én innbygger - én journal" is currently under development by NDE. One of the challenges in the "Én innbygger - én journal" project is that healthcare staff in municipalities do not have access to an up-to-date source of drugs information [6]. Consequently, there is a major need for a common national source of information about drugs that can be easily accessed, updated, and integrated across various medical information systems including the patient's journal.

The Norwegian Electronic Prescription Support System (Forskrivnings- og ekspedisjonsstøtte FEST) is a Norwegian database in XML format offered by the Norwegian Medicines Agency (SLV). FEST includes all information about drugs that can be prescribed electronically in Norway. FEST is used as a pharmaceutical data source by practitioners, hospitals and pharmaceutical companies. FEST provides practitioners with the required pharmaceutical information when they prescribe a medicine such as side effects, delivery difficulties in pharmacies, and warning in case a specialist should be contacted before prescribing a medicine. Today, hospitals have their own drug registry, but they use FEST as a resource of drug information, and they take the responsibility of updating their own systems. Pharmacies use FEST along with another system, FarmaPro, developed by Farmalogg; both systems synchronize information about drugs so that pharmacies have the best information about each drug from both systems [7].

Currently, users of FEST have two options to get an up-to-date version of FEST. They should use a solution that can automatically search for updates every 3 nights or manually download FEST from SLV website before the 1st and the 15th day of each month [8].According to the Norwegian Directorate of eHealth, FEST needs more development and improvements in many aspects such as data quality, content and update [9].

Applying international standards to the medical data stored in FEST will improve the integration process with other medical systems, allow better medical interoperability, increase patients' care quality and reduce the risk of patient injuries and deaths related to incorrect medication.

1.1 Background

Interoperability, i.e. the sharing and exchange of data between various information systems, is of great relevance; disparate healthcare information systems are no exception. Let's say, for example; a practitioner sends a patient to a specialist to address a complicated case. The specialist prescribes a drug for the patient who will get it from a pharmacy. If the prescribed drug is not available at the pharmacy, then the pharmacist will give the patient an alternative drug and send an instant update to the practitioner. Later, the patient gets another prescribed drug from the practitioner which may interact with the former drug. So even if the practitioner, specialist and the pharmacist are using different systems, still patient's information is exchanged properly and instantly.

Such interoperability, providing the capability of instant exchange and share of patient's information, requires the use of standards. The role of standards is to determine the structure of both shared data and exchanged messages from one healthcare information system to another. In this context, a list of six suggested standards and specifications have been mentioned by the academic committee (NUFA) for health and architecture in Norway [10]:

- **HL7 V3 Messaging**; the Health Level Seven Version 3 is an international standard based on HL7's Reference Information Model (RIM) and HL7 Version 3 Development Framework where messages and documents are in XML syntax with focus on information's semantics.
- **IHE XDS.b**; Integrating the Healthcare Enterprise – Cross-Enterprise Document Sharing is an international specification for exchange and share of documents with no restrictions on the type of documents being shared [11].
- **OpenEHR**; offers a set of specifications of interface implementation and a platform of journal systems, including a query language, decision support rules, and modelling of clinical information [12].
- **HL7 CDA R2**; Clinical Document Architecture is an international standard from HL7 used for documents exchange where a framework of document markup standard is used to describe the structure and semantics of clinical documents [13].
- **Linked Data**; a concept of describing and connecting data by using a set of standards from W3C (World Wide Web Consortium) where an URI is used to name a thing, HTTP-based URI is used to search for names, RDF and SPARQL standards are used to provide information, and links to other URIs are used to find more information [14].
- **HL7 FHIR**; Fast Healthcare Interoperability Resources is a standards framework from HL7 intended for integration/communication of health information between EPR systems (Electronic Patient Record systems). HL7 FHIR encompasses the best features of HL7's V2, HL7's V3, and CDA. FHIR uses the latest Web standards with a tense focus on implementation. FHIR solutions are based on modular components called resources which represent a wide variety of medical information [15].

Support of RESTful architectures is one of the important advantages that HL7 FHIR has over the previously mentioned standards where it supports service-based architectures [15]. In addition, according to an assessment of international standards and national coordination of standards published by the Norwegian Directorate of eHealth (NDE) [10], HL7 FHIR is the only international standard recommended for a solution where REST-based operations or a REST-based architecture being applied. The assessment takes into consideration four models of interaction; message exchange, document exchange, document sharing, and data sharing. HL7 FHIR is the only international standard that can be used in all the interaction models. Furthermore, base resources in HL7 FHIR can be adjusted for local requirements which supports interoperability as well [15].

Our main goal of this thesis is to represent the Norwegian Electronic Prescription Support System (FEST) in terms of Fast Healthcare Interoperability Resources (HL7 FHIR). Thereby, drug information will become available/accessible through a REST-based service. Sending a request/query to such a REST-based service will provide a query result in FHIR format. Furthermore, there will be no need to download an up-to-date FEST database before sending a query about drug information because the update process should be handled automatically.

1.2 Motivation

The motivation behind our work in this thesis is to reduce the potential risks in patients' treatment that may result from lack of interoperability between various medical information systems. Our concept is to enable a representation of FEST that follows an international standard which will improve the accessibility of the information about drugs in Norway.

1.3 Problem Statement and Goals

FEST is a national source of information about drugs which is mainly designed to support e-prescription, but it is also used today by Norwegian hospitals in the four health regions "Helse Sør-Øst RHF," "Helse Vest RHF," "Helse Midt-Norge RHF" and "Helse Nord RHF." Hospitals have increasing demands related to the content and the quality of the information that FEST currently offers. In addition, specialist health services "Spesialisthelsetjenesten" currently have no access to one complete and up-to-date national source for drug data that satisfies requirements such as decision support for example. Consequently, specialist health service institutions have to provide their own solutions, use source data from hospitals, maintain and manage source data locally and utilize resources for their own management.

Currently, each one of the four health regions implements and maintains its own local solutions on a regular basis to structure and quality assure the data offered by FEST. As a result, providing FEST data elements in a well-standardized form would give hospitals more flexibility regarding integration solutions as well. According to the SAFEST project [16], the current demands of hospitals and specialist health services regarding FEST are categorized under four categories shown in the figure below.

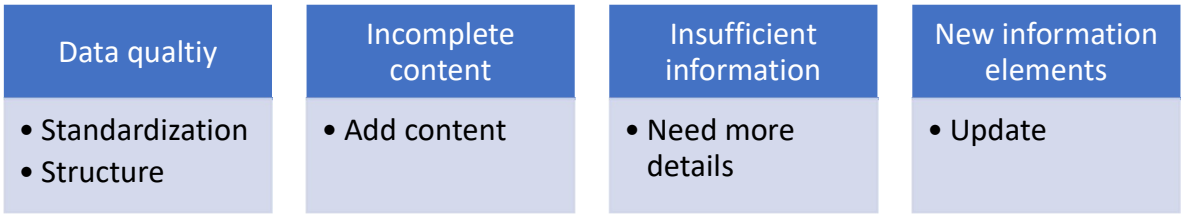


Figure 1.1: Categories of requirement

Furthermore, the fact that drug data sources can be national and international implies an essential requirement to establish standardized integration solutions for both national and international drug data providers, which can fulfil the increasing demands of hospitals for data exchange.

Having access to one common national source of information about drugs which is up-to-date, relevant and quality assured would save the resources used by each health region to compensate the deficiencies in today's FEST and enhance hospitals' performance as well. In addition, more precise coordination will reduce medication errors and confusion situations, increase patient's care quality and reduce the risk of patient injuries and deaths related to incorrect medication [9].

According to the report "IE-1005 Assessment of international standards, national coordination of standards" published by Norwegian e-health directorate, applying international standards and specifications to the medical data should improve the integration process with other medical systems and allow for better medical interoperability [10].

Having access to a common national drug register solution in which international standards and specifications are implemented will solve problems related to the following areas:

- Norwegian health services need a common national drug register solution following international standards instead of using local solutions to achieve safe, effective and correct prescribing, treatment and handling of drugs [9].
- Clear communication inside and across health care services. For example, the Norwegian medical agency SLV is looking for a new distribution service to be used when a warning message about safety information related to a specific drug should be sent to the GPs, pharmacies, and patients. In

the current FEST(M30), an SLV message is sent only to GPs, while in F5 FEST, the message is sent to GPs, pharmacies, and patients but SLV is looking for a new distribution service as it claims that F5 FEST is just a proof of concept which will not be further developed, improved or supported in any way [17].

- Benefits for patients, healthcare professionals, and society through a quality assured and coordinated source of public drug information.
- Reduce the need for the use of the professional resources in implementing, introducing and managing medical solutions in specialist health services.
- Operations become safer when services are centralized.
- Medication is a vital part of the healthcare process, and it's considered as an essential part of some eHealth research/development projects being conducted at UiA. Thus, accessibility to up-to-date drug information would improve development and research activities. In addition, when a solution follows a recommended international standard in the context of information exchange, it becomes more applicable. In some projects, an implementation of a complete eHealth system is composed of various software components; a drug information component could be one of them.

1.4 Research Questions, Focus and Limitations

1.4.1 Research Questions

RQ1: How to represent the FEST data model in terms of a HL7 FHIR data model?
RQ2: How to provide accessibility to FEST data through FHIR resources in a REST-based architecture?
RQ3: How to provide decision support information / queries, utilizing the FHIR resources that give access to underlying FEST data?

1.4.2 Focus and Limitations

Our goal is not to fully represent FEST in all details. Thereby, we will select some catalogues of the main FEST catalogues and demonstrate a use case.

1.5 Proposed Solution

Applying HL7 FHIR international standards on the medical data stored in FEST register will improve the integration process with other medical systems and allow for better medical interoperability. According to the report "Assessment of international standards: National coordination of standards" published by the Norwegian e-Health directorate, HL7 FHIR is recommended where REST-based operations or architecture is applicable. There are four different interaction models where HL7 FHIR can be used; message exchange, document exchange, document sharing and data sharing [10]. An adaption of HL7 FHIR specification to build a new HL7 FHIR-based data model and a distribution solution by mapping FEST ML30 data entities to HL7 FHIR resources. While there is waiting for a large-scale introduction of HL7 FHIR, We will introduce our focus area to develop a proposed solution based on HL7 FHIR standards and specifications as a proof of concept.

There are various FHIR-based solutions that are in the phase of development or deployment. DoseMeRx is a clinical decision support software used to determine precisely the optimal drug dose in real-time. Different data sources are used to provide the required information for clinicians; pharmacokinetic drug data, patient's profile, and concentrations of drug. DoseMeRx can be used as a standalone solution accessed through the Web or as a mobile app as well [18].

Evolve Integrated Care is a FHIR-based solution on cloud-based platform, it provides clinical care pathways and can be integrated with EHR systems. FHIR is used as both the main data repository and the service to exchange data with EHR systems. Evolve Integrated Care is currently used by 100 hospitals around UK and NHS to improve the automation of stroke, behavioural health, and emergency care pathways for clinics [19].

1.5.1 Mapping FEST M30 data entities into FHIR resources

As drug data is stored in FEST register in XML format, our first step is to use Java XML integration technology to map **automatically** FEST XML data model into a new data model represented by Java classes. As a result, all data elements represented in FEST register will be accessible as Java instances of its corresponding generated Java classes. The next step will be extending FHIR resources to fit various FEST drug data elements. The former two steps require analysing FEST M30 XML schema structure and provide the suitable corresponding FHIR resources if available. Otherwise, we need to extend the defined FHIR resources or define our own resources which must adhere to FHIR specifications in turn. In this perspective, the use of extensions in FHIR allows our solution to be more flexible with potential changes in the structure of FEST.

1.5.2 Develop a RESTful service

HAPI FHIR is a Java library in which HL7 FHIR specification is implemented for Java [20].
A model of classes for all resource types defined by FHIR specification and extended by us will be represented in a RESTful server where we will develop RESTful service to access and query the FHIR resources representing FEST M30 data entities. Our implementation will be developed in Java programming language.

1.5.3 Develop web application components to interact with RESTful service

We will develop Web-based back-end components to access the RESTful service developed in the former step. As a proof of concept, we will develop a front-end interface to provide Web access to our FEST on FHIR resources, and a use case of drug-drug interaction will be applied as well.
The figure below illustrates an overview of the system architecture to be designed and implemented.

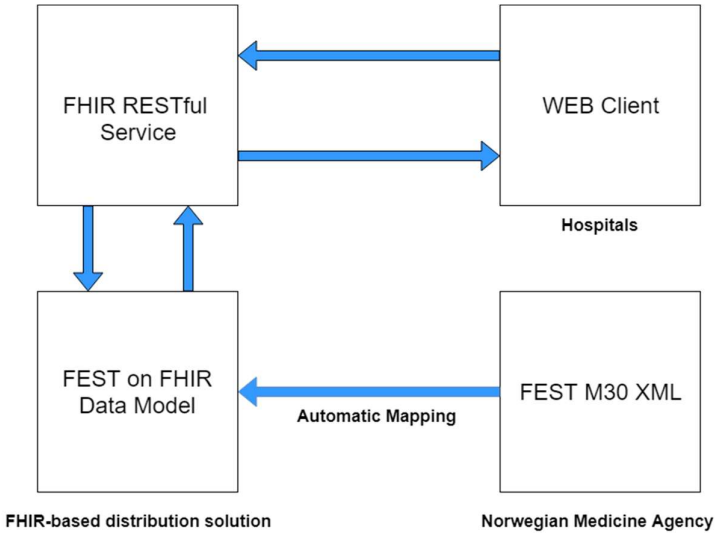


Figure 1.2: System architecture overview

On the other hand, our system design and implementation will be scalable and extensible. We take into consideration that in the future, the FHIR-based distribution solution can be adapted so that when the Norwegian medicines agency SLV updates and edits FEST XML file, a request to the FHIR-based distribution solution will be sent to trigger the automatic mapping of FEST M30 XML data into FHIR resources representing the structure and content of FEST M30 drug register according to FHIR specification.

1.6 Report structure

This thesis is divided into five chapters as follows:
Chapter two represents theoretical and technical background where this work is based on.
It includes a review of the related literature; electronic health record (HER), health level seven international (HL7) and its evolution, electronic prescribing, interoperability in eHealth industry, interoperability between EHR and ePrescription, fast healthcare interoperability resources (FHIR), state-of-the-art of medicines information systems, the Norwegian electronic prescription support system (FEST), Felleskatalogen portal for drug information, the anatomical therapeutic chemical (ATC), SAFEST project, the Norwegian electronic patient journal (Kjernejournal KJ), HL7 FHIR Release 3 (STU; v3.0.1), HAPI-FHIR, FHIR client, XML, FEST information model, FHIR profiling Forge tool, Java XML integration, Apache Maven, Apache Tomcat, and Web application development.

Chapter three represents the details and description of the implementation of our proposed solution. We describe step by step how we built our solution; starting from the required knowledge we needed, system design, the implementation of the system, and finally testing and validation of our solution. In addition, we introduced a practitioner use case to provide a proof of concept.

In chapter four, we discuss our proposed solution based on the outcome. We also describe some challenges we met through our implementation and compare our implementation to the current solution offered by Felleskatalogen.

In chapter five, we provide a brief conclusion about our problem, solution, and results. In addition, we introduce a system architecture for proposed future work.

2 Theoretical and Technical Background

2.1 Electronic Health Record EHR

The main idea behind having patient's records stored and shared electronically evolved through the need of solving issues related to written prescriptions by various providers such as practitioners, clinics and hospitals. These issues mainly include communication, efficiency, patient medical history, and quality of patient's care. EHR becomes a promising concept as the improvement of patient's care is the focus; improving the workflow of healthcare service is no exception. EHR is defined as "a longitudinal collection of electronic health information about individual patients and populations." [21]

As the concept of EHR being introduced and worldwide used, various categorization, classification, and customizations of patient's records were implemented. As a result, a variety of methods for recording patient's data has been introduced. The difference of these methods can be expressed in terms of using different names, different acronyms and having different purposes. Automated Medical Records (AMR), sometimes referred to as AHR, is one of the earlier methods to transform patient's data into digital form by simply scan written prescriptions. Although AMR reduces the usage of paper in patient care process, it is not that helpful when it comes to decision support process.

Electronic Medical Records (EMR) is considered more complicated system than AMR. All prescriptions, lab results, and nurse care records are included in the form of digital images. Decision support is enabled to a certain limit besides statistics which provides more support from business and industrial perspective.

The Medical Records Institute classified different electronic healthcare record systems into 5 levels of sophistication illustrated in the figure below [22].

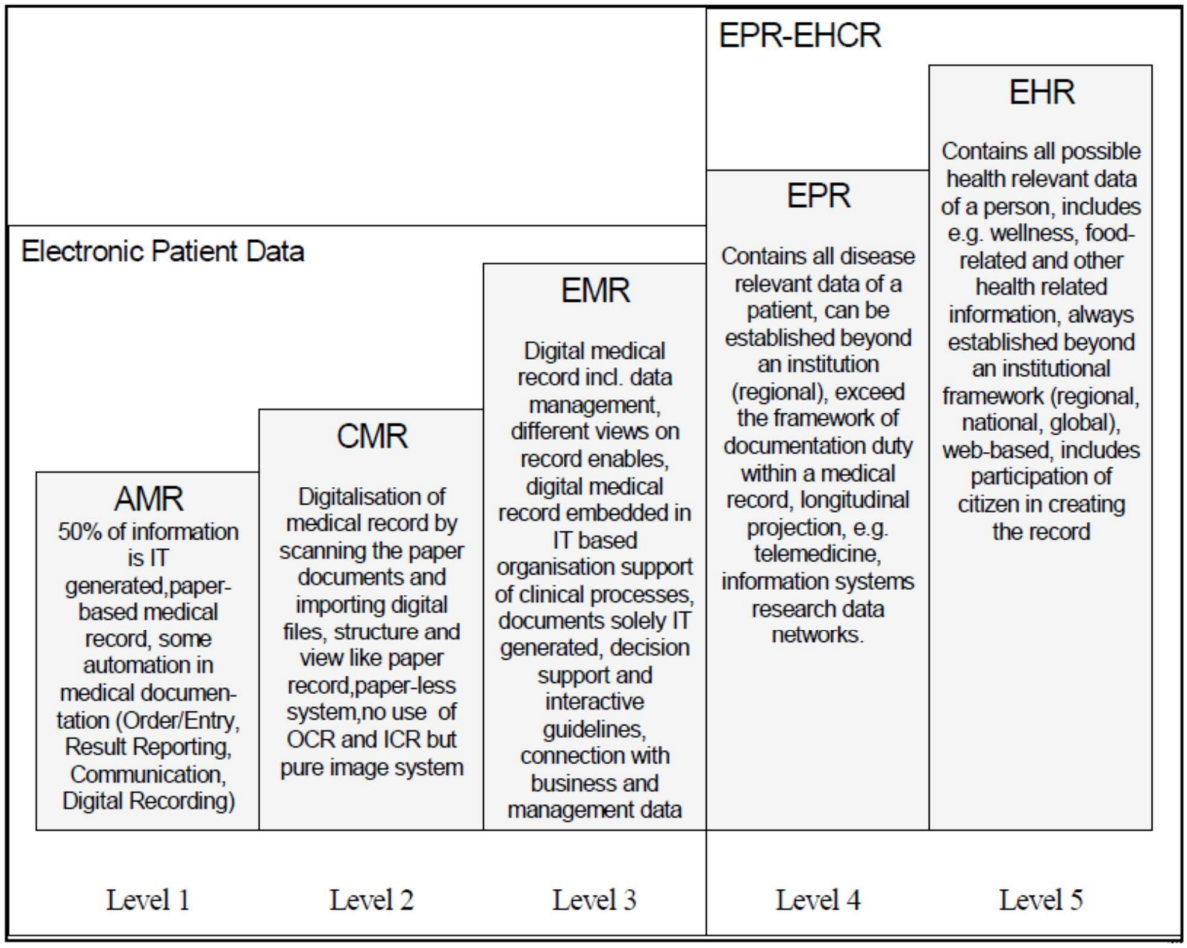


Figure 2.1: Levels of Electronic Healthcare Records (EHCR) [22]

From a technical point of view, the development of a healthcare system that can exchange messages and various medical information with other healthcare systems requires sophisticated communication standards and interoperability techniques as well. Defining such standards can prove to be a challenging task which can be illustrated by the figure below *[23]*.

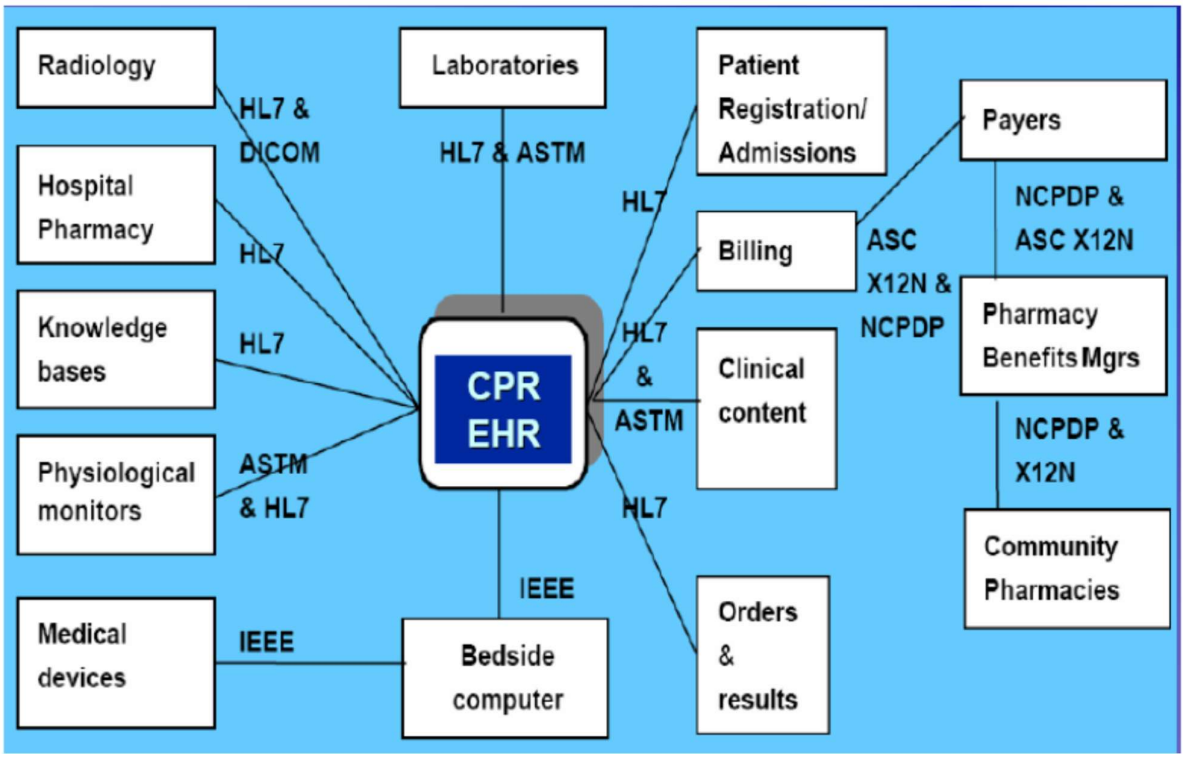


Figure 2.2: EHR Communication Standards [23]

The figure shows various standards developed and used for different purposes. One EHR system may be using HL7 standards for hospital pharmacy and clinical content but might be using IEEE standards for medical devices. Consequently, sharing and exchange of information become more complicated.

2.2 Health Level Seven International (HL7)

In general, development of standards that can fulfil the requirements of healthcare information systems takes long time as it includes technical committee and experts from different stakeholders, roundtable discussions, issuing drafts, and voting. If a draft can get agreement, then it becomes an approved standard to be used; HL7 is no exception.HL7 is considered one of the most worldwide used standards in healthcare information sector [24]. The number "Seven" refers to the seventh layer, application layer, of the communications model for Open System Interconnections (OSI) issued by the International Organization of Standardization (ISO) [25]. At the application layer of OSI model, common application services are handled such as sending requests and interpreting received responses.

HL7 has partners in all continents. European HL7 partners include 21 countries such as
Norway, Finland, Sweden, Britain, Germany, and Italy. According to HL7 working committee founded in 1987, "Health Level Seven International (HL7) is a not-for-profit, ANSI-accredited standard developing organization dedicated to providing a comprehensive framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services. HL7 is supported by more than 1,600 members from over 50 countries, including 500+ corporate members representing healthcare providers, government stakeholders, payers, pharmaceutical companies, vendors/suppliers, and consulting firms." [25].

HL7 is the state-of-the-art of international clinical information standards as it has a variety of agreements with other standards development organizations which allows HL7 to collaboratively facilitate the integrations between different healthcare systems to share data without interruption.

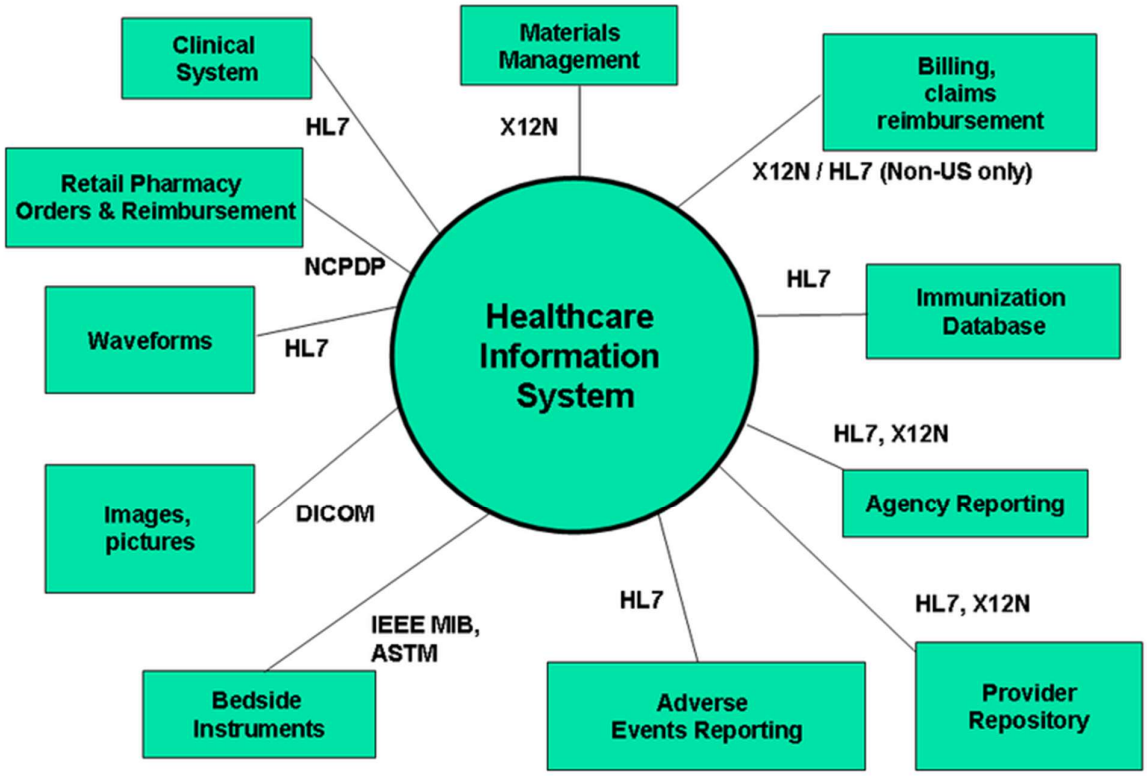


Figure 2.3: Healthcare Data Interchange Standards [26]

Some of the standards development organizations which have agreements with HL7 [27]:

- **SNOMED CT** is a clinical reference for health terminology, which provides a precise global language for health terms.
- **NCPDP** data processing standards for pharmacies.
- **ADA** data processing standards for dental services.
- **ANIA** data processing standards for nursing services.
- **ASTM** messages exchange for observations performed in clinics.
- **Centres for Disease Control Prevention.**
- **Clinical Data Interchange Standards Consortium** for medical and biopharmaceutical product development.
- **DICOM** clinical images exchange.
- **X12N** exchange of insurance, eligibility, and management information.
- **The Institute of Electrical and Electronics Engineers (IEEE).**

2.3 Evolution of HL7

From a practical point of view, HL7 represents a framework of clinical standards in which data can be exchanged. When HL7 v1 released, only the structure of the interface between systems was introduced, and there was a need to customize/develop an interface for each system to be able to exchange data, messages, with other systems. The motivation behind HL7 v2 comes from clinical interface specialists as they need to build a standard framework for interfacing, which includes the main common requirements for data exchange and reduces interface customization efforts at the same time. The main reason behind the market success of HL7 v2 is the 80/20 approach, where 80% of the interface is predefined, and 20% is left to be customized to meet the special rules of healthcare providers. Because of the widespread usage of HL7 v2, the focus was on refinement and addressing challenges related to consistency of data model, message elements relations, proper definition of user and application roles, and standard precision. In 2005, the first version of HL7 v3 was released, introducing advantages such as internationalization and consistency of data model [28].

```
MSH|^~\&|AcmeHIS|StJohn|ADT|StJohn|20060307110111||ADT^A04|MSGID20060307110111|P|2.4
EVN|A04
PID|||12001||Jones^John||19670824|M|||123 West St.^^Denver^CO^80020^USA
PV1||O|OP^PAREG^||||2342^Jones^Bob|||OP|||||||2|||||||||||||||||||||||20060307110111|
AL1|1||3123^Penicillin||Produces hives~Rash~Loss of appetite
```

Figure 2.4: Message format in HL7 v2 [28]

Figure 2.5: Message format in HL7 v3 [28]

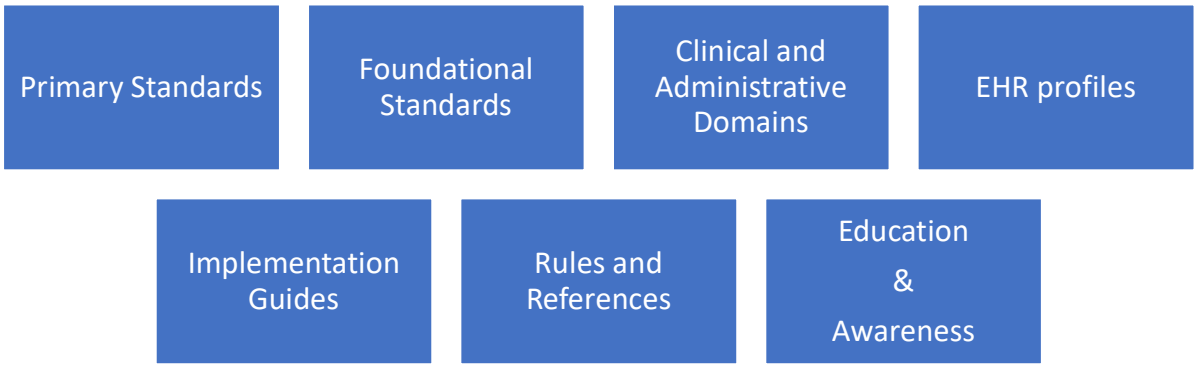


Figure 2.6: HL7 v3 standards categories

Example of HL7 v3 standard under "EHR profiles" category is "HL7 EHR-System ePrescribing Functional Profile Release 1". The goal of the standard is a mapping guidance for electronic prescription information [29]. The Norwegian Electronic Prescription Support System (FEST) is based on HL7 v3. FEST includes all information about drugs that can be prescribed electronically in Norway.

2.4 Electronic Prescribing (ePrescribing)

The term electronic prescribing (ePrescribing) is defined by eHealth initiative as "the use of computing devices to enter, modify, review, and output or communicate drug prescriptions" [30]. eHealth initiative divides ePrescribing into six levels where each level is considered to include all the functionalities of the level below it [30]. At top levels, ePrescribing is considered as a component of an EHR system [31].

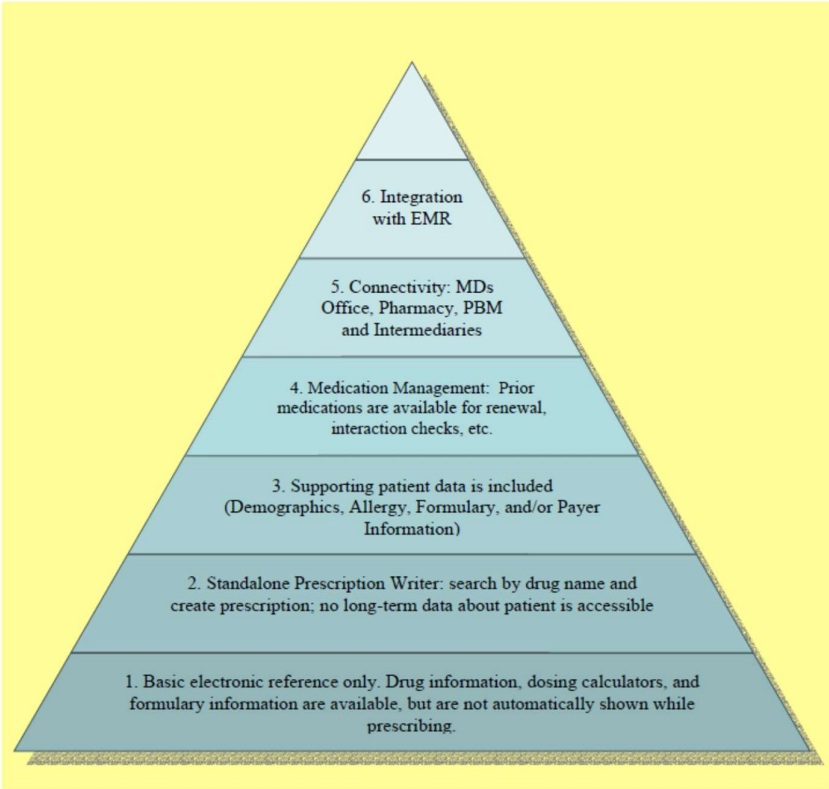


Figure 2.7: Levels of Electronic Prescribing [30]

2.5 Interoperability in eHealth Industry (IHE)

When it comes to healthcare services, there are various actors and factors affecting health system interoperability, for example:

- Patients from different ages.
- Healthcare professionals.
- Hospitals.
- Coordination between different departments within a hospital.
- Clinics of different specificities.
- Insurance.

Given the above examples, the interoperability of a healthcare system can be defined in terms of:
The system can interchange, recognize and process patient and health information collaboratively.
Differences in language or culture does not affect the system, patient, healthcare professional, or the workflow of a healthcare service [31].

2.6 Interoperability between EHR and ePrescription

According to the conceptual framework of interoperable electronic health record and ePrescribing systems [31], interoperability between EHR and ePrescribing systems is divided into levels where interoperability increases in ascending manner.
Level 1: "Availability/access to stand alone solutions" [31], which means EHR and/or ePrescribing solutions is available.
Level 2: "Potential for interoperability" [31], which means sharing information without real exchange.
Level 3: "Real inter-operation" [31], which means sharing and exchange of information as well.
Moreover, within level 3, the type of connectivity plays a vital role:

- Local connectivity: only on one site, sharing and exchange of information takes place.
- Multi-site connectivity: an organization has multi sites.
- Regional connectivity: organizations within the same region.
- National connectivity: organizations within the same country.
- Multi-national connectivity: cross country border.

2.7 Fast Healthcare Interoperability Resources (FHIR)

"FHIR® – Fast Healthcare Interoperability Resources – is a next generation standards framework created by HL7. FHIR combines the best features of HL7's v2, HL7 v3, and CDA product lines while leveraging the latest web standards and applying a tight focus on implement ability. 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. FHIR is suitable for use in a wide variety of contexts – mobile phone apps, cloud communications, EHR-based data sharing, server communication in large institutional healthcare providers, and much more." [32].

As FHIR mainly focusing on implementation, there are many libraries available for development with no restrictions on specification. In addition, FHIR supports various web standards such as XML, JSON, HTTP, and OAuth. Supporting the web standards enhances the flexibility and extensibility of FHIR which makes messages and documents exchange becomes easier.

The main challenge for healthcare standards is the continuous need to add more fields and options to the specification which increases the cost and complexity of implementations. "FHIR solves this challenge by defining a simple framework for extending and adapting the existing resources. All systems, no matter how they are developed, can easily read these extensions and extension definitions can be retrieved using the same framework as retrieving other resources." [32].


```
<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 2.8: Example of Patient FHIR Resource [32]

2.8 State-of-the-art of Medicines Information Systems

From information technology (IT) perspective, there is a growing range of users who need to instantly access medicines information. Consequently, there is a user's-focused development of IT services that meet the requirements of different users such as patients, healthcare professionals, and pharmaceutical companies.

"The International Organisation for Standardisation (ISO), Identification of Medicinal Products (IDMP) standards specify the use of standardised definitions for the identification and description of medicinal products for human use. The purpose of these standards is to facilitate the reliable exchange of medicinal product information in a robust and consistent manner, by providing a common product 'language' for stakeholders to use in their interactions. The use of these standards is a regulatory requirement as they are mandated by the EU legislation (Commission Implementing Regulation (EU) No 520/2012 [articles 25 and 26])." [33].
One of the use cases of ISO IDMP is HL7 Messaging Specification which defines the structure of the messages that will be used to exchange information. HL7 Messaging Specification is derived from HL7 standard called Common Product Model (CPM) [33].

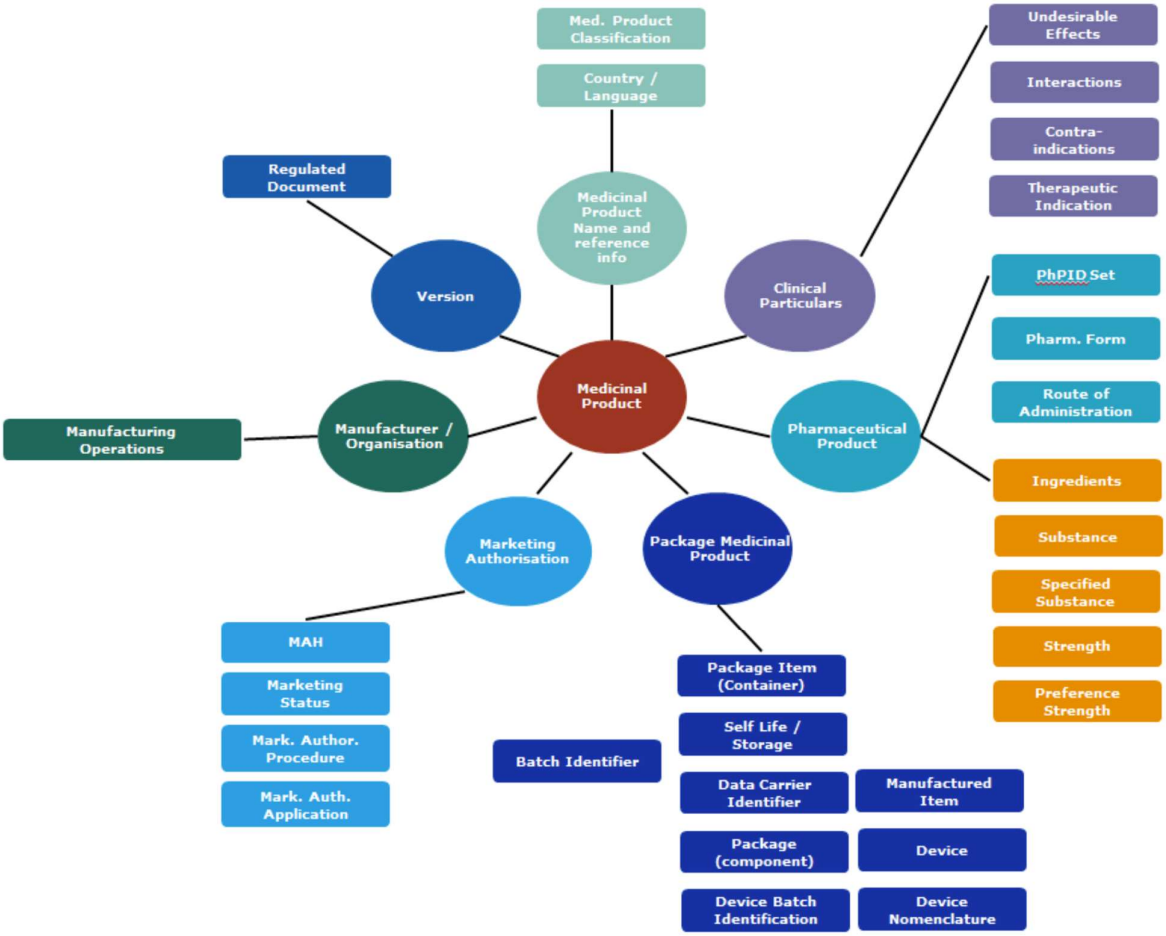


Figure 2.9: ISO IDMP Data Elements [33]

"The approach to implementing the ISO IDMP standards is based on the four domains of master data in pharmaceutical regulatory processes: Substance, Product, Organisations and Referentials (SPOR) data.

EMA will establish ISO IDMP compliant business services for the central management of data in each of the four SPOR areas. These include data management services for:

- **Substance data**: Harmonized data and definitions to uniquely identify the ingredients and materials that constitute a medicinal product.
- **Product data**: Harmonised data and definitions to uniquely identify a medicinal product based on regulated information (e.g. marketing authorisation, packaging and medicinal information).
- **Organisation data**: Data that comprises of organisation name and location address data for organisations such as MAH, sponsors, regulatory authority, manufacturers.
- **Referential data**: Lists of terms (controlled vocabularies) used to describe attributes of products, e.g. lists of dosage forms, units of measurement, routes of administration." [33].

2.9 The Norwegian Electronic Prescription Support System (FEST)

"The Norwegian Electronic Prescription Support System (FEST) is an information service provided to expose common pharmaceutical data to all members of the prescription chain; Physicians, hospitals, pharmacies and manufactures of surgical appliances receive updated information about all articles available for prescription/dispensing in Norway from one single source. Basic data provided by the Norwegian Medicines Agency through FEST can be used for different systems. It is not necessary to replace the systems

to use FEST, only to adapt them to receive data in the correct format. The suppliers of the various systems are responsible for developing and providing the required functionality and use of the information." [34]

Along with the information model provided by FEST M30 v2.5, the latest version of FEST, there is a prescription model (Forskrivning) which includes information that are not relevant to FEST M30 but used in the prescription and other notifications of E-prescription [34]. The following diagram illustrates the combination of FEST M30 information model and Prescription (Forskrivning):

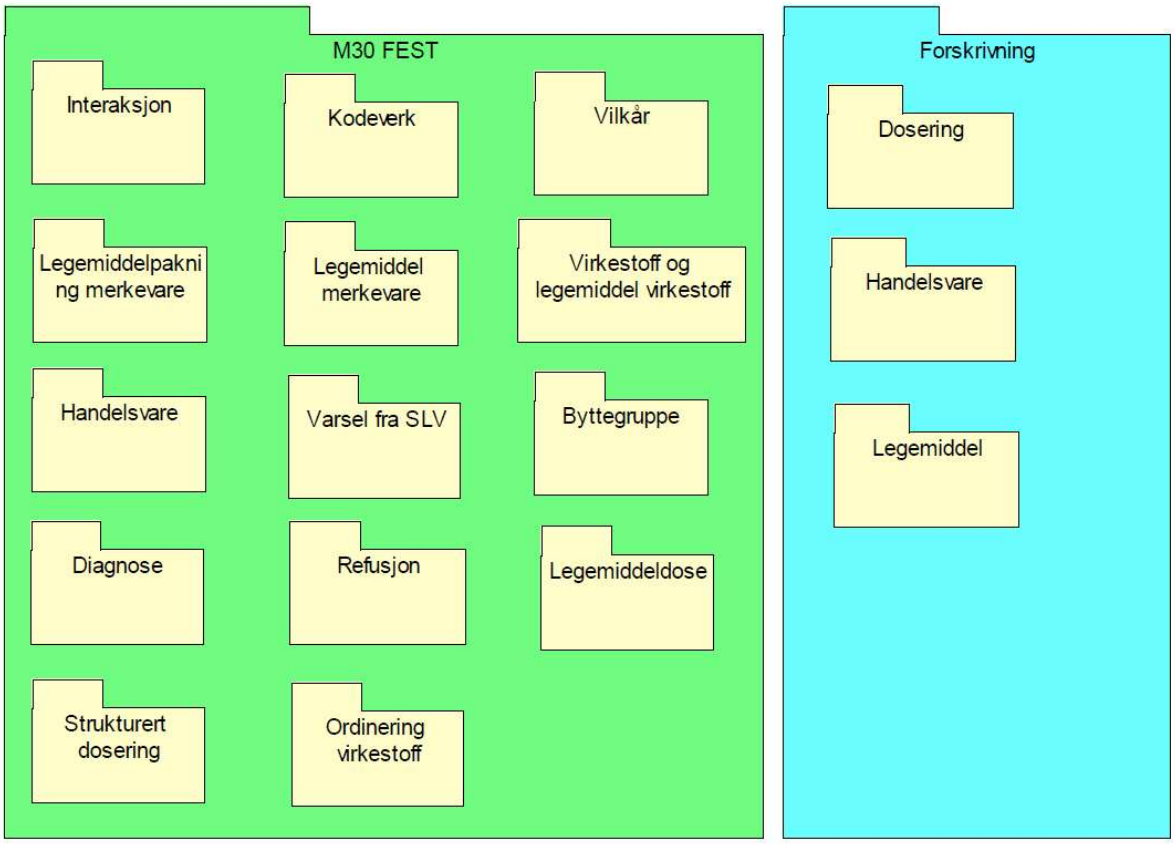


Figure 2.10: Structure of Information Model of FEST M30 and Prescripton(Forskrvning) [35]

As shown in the figure above, FEST M30 consists of various catalogues which used to describe drug information, prescribing rules, usage warnings and so on. On the other hand, catalogues of Prescription model are used mainly for drug prescription with administration information such as the price or discount on the drug in case of chronic disease for example (blå resept).

2.9.1 Felleskatalogen

www.felleskatalogen.no is a Norwegian portal, which freely provides drug information such as ATC-code, active substance, strength, form, manufacturer, package, and product number. Information is driven from FEST which is updated each 14 days. Drug-drug interaction analysis is given by entering the name of a drug, active substance, or ATC-code [36].

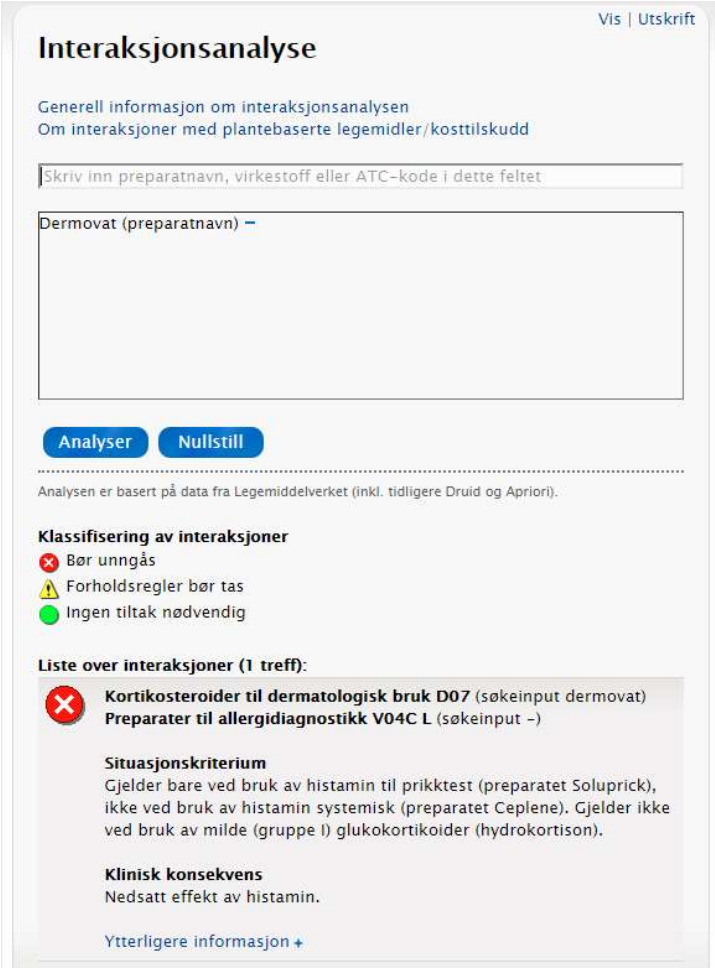


Figure 2.11: Felleskatalogen Drug-drug interaction analysis [36]

2.10 The Anatomical Therapeutic Chemical (ATC)

"The Anatomical Therapeutic Chemical (ATC) classification system and the Defined Daily Dose (DDD) as a measuring unit have become the gold standard for international drug utilization monitoring and research. The ATC/DDD system is a tool for exchanging and comparing data on drug use at international, national or local levels." [37]. ATC is used for medications and in interactions by the Norwegian Electronic Prescription Support System (FEST) [34]. The classification of the active substances is hierarchical and divided into five distinguished levels. For example, the following illustration depicts the code structure of metformin classification [38].

A	Alimentary tract and metabolism (1st level, anatomical main group)
A10	Drugs used in diabetes (2nd level, therapeutic subgroup)
A10B	Blood glucose lowering drugs, excl. insulins (3rd level, pharmacological subgroup)
A10BA	Biguanides (4th level, chemical subgroup)
A10BA02	metformin (5th level, chemical substance)

Figure 2.12: ACT code for metformin [38]

2.11 SAFEST

SAFEST is a Norwegian project, managed by National IKT organization, aimed to improve drug information provided by the Norwegian Medicines Agency (SLV) for use in the specialist health service (Forbedre legemiddelinformasjon til bruk i spesialisthelsetjenesten). The goal of the project is to develop national solutions to support medicines regulations in hospitals with an up-to-date source of drug information, FEST [39]. The concept behind the project includes the adaption of systems currently used by the Norwegian Medicines Agency (SLV), integration of national and international data sources, and development of new systems. The current project phase is realisation [40]. The figure below shows the proposed architecture of the project.

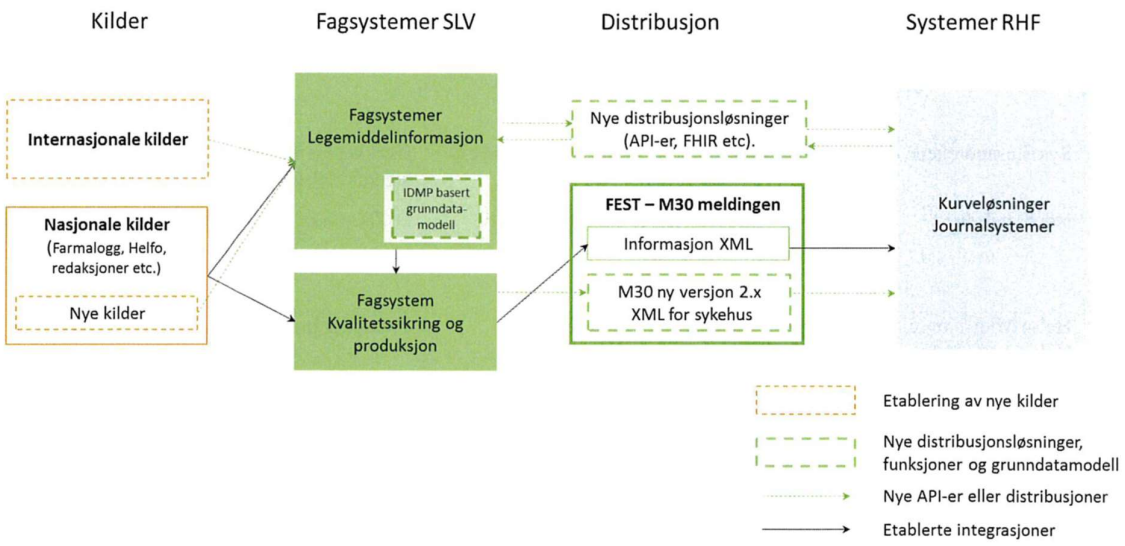


Figure 2.13: SAFEST Concept Overview [40]

When it comes to drug information provided by the Norwegian Medicines Agency (SLV), the data model structure is proposed to be based on IDMP. On the other hand, in our solution implementation, the data elements of FEST M30 be will be re-structured according to HL7 FHIR v3 specifications. In addition, both SAFEST and our thesis address the problem of distribution by using FHIR as a solution.

2.12 The Norwegian Electronic Patient Journal (Kjernejournal KJ)

According to the Norwegian eHealth Directory, Kjernjournal is a digital solution which includes vital health-related information about patients which can accessed by both patients and healthcare professionals and shared across various levels of healthcare systems.

Figure 2.14: Kjernjournal [41]

Alert information, kritisk informasjon in Norwegian, means information that in a treatment situation may lead to change decisions, save patient's life, or help in preventing patient from serious injury. Alert information is part of the content which is represented by Kjernejournal as shown in red in the figure below.

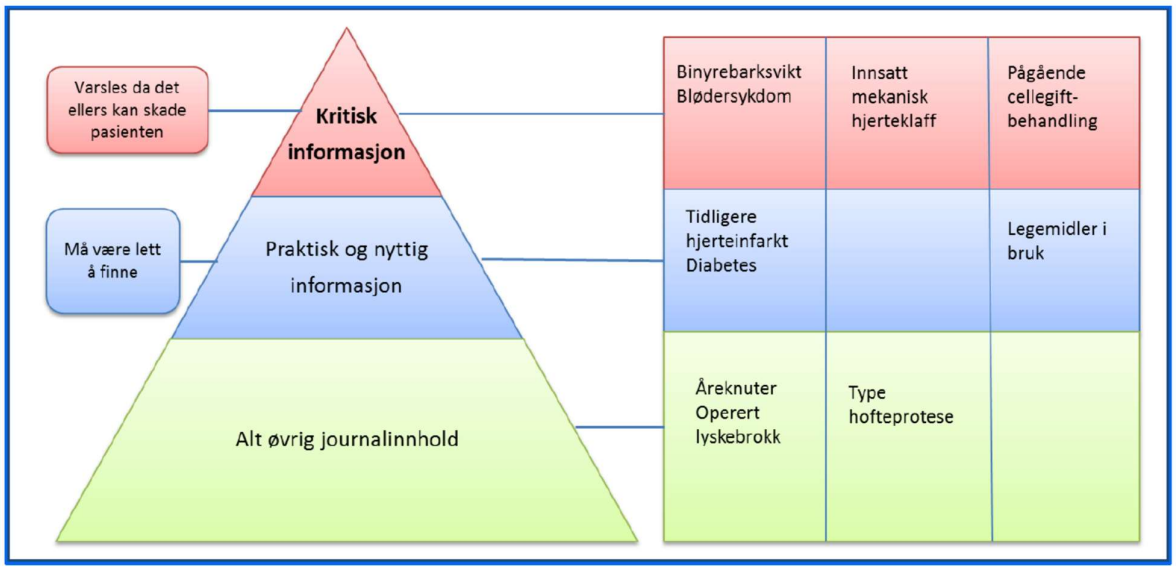


Figure 2.15: Content of Kjernejournal [42]

The concept of HL7 FHIR-profile, set of restrictions on FHIR resources to be used in the same context, is introduced here in Norway by many projects; Kjernejournal is no exception. According to the implementation guide, published by the Norwegian HL7 FHIR *[43]*, there are many Norwegian projects using HL7 FHIR resources. For example, Kjernejournal uses AllergyIntolerance, Medication, Device, and Procedure resources.

DIPS, a Norwegian eHealth solutions platform, uses HL7 FHIR resources Diagnosis and Procedure in order to support medical coding in its electronic patient journal (EPJ) solution used by Oslo University Hospital (Oslo universitetssykehus OSU) *[44]*.

2.13 HL7 FHIR Release 3 (STU; v3.0.1)

The version of HL7 FHIR we are going to use in our implementation is FHIR Release 3(STU; v3.0.1). In this context, STU means Standards for Trial Use. There are many reasons behind choosing FHIR in our work; focus on implementation, availability of implementation libraries,
no restrictions on use of specifications, based on web standards; XML, JSON, HTTP, support of RESTful and service-based architectures [45].

The structure of FHIR specification is divided into five levels; level 1, level 2, level 3, level 4 and level 5. Each level has a role(s) and includes a set of modules. Each module includes related-resources. The figure below illustrates the structure of FHIR specification.

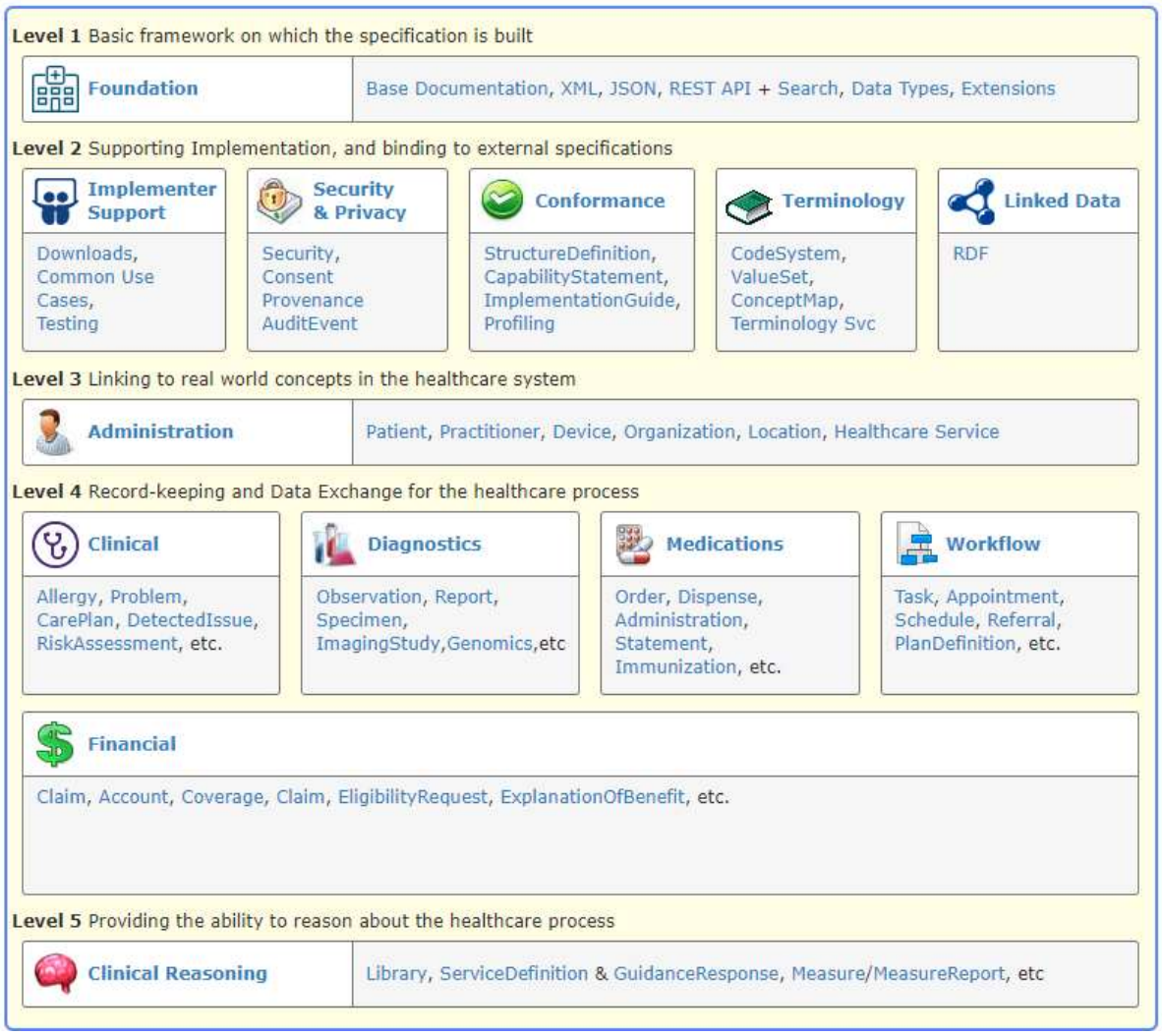


Figure 2.16: Structure of FHIR v3.0.1 [46]

2.13.1 FHIR Overview

From a technical perspective, FHIR is an implementation-oriented standard where information integrity is taken in consideration. FHIR is flexible as it can be used as a standalone standard or with other standards. The main building units of FHIR are resources which are used to exchange data. A resource is composed of a set of elements of various data types; primitive and complex types. Primitive data types have a value and no child elements while complex data types have child elements [47]. In addition, a resource has metadata and human readable part.

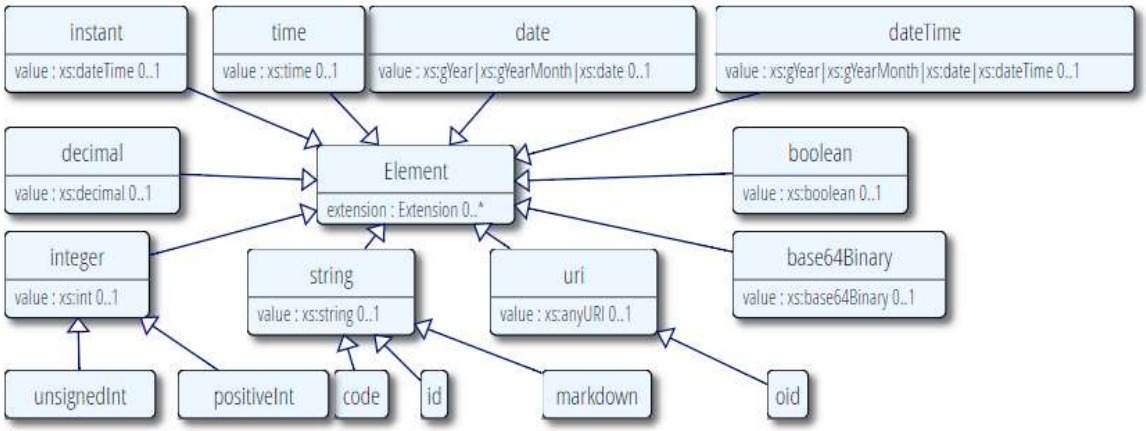


Figure 2.17: FHIR primitive data types [47]

Example of primitive data type in XML format:
<count value = “3”>

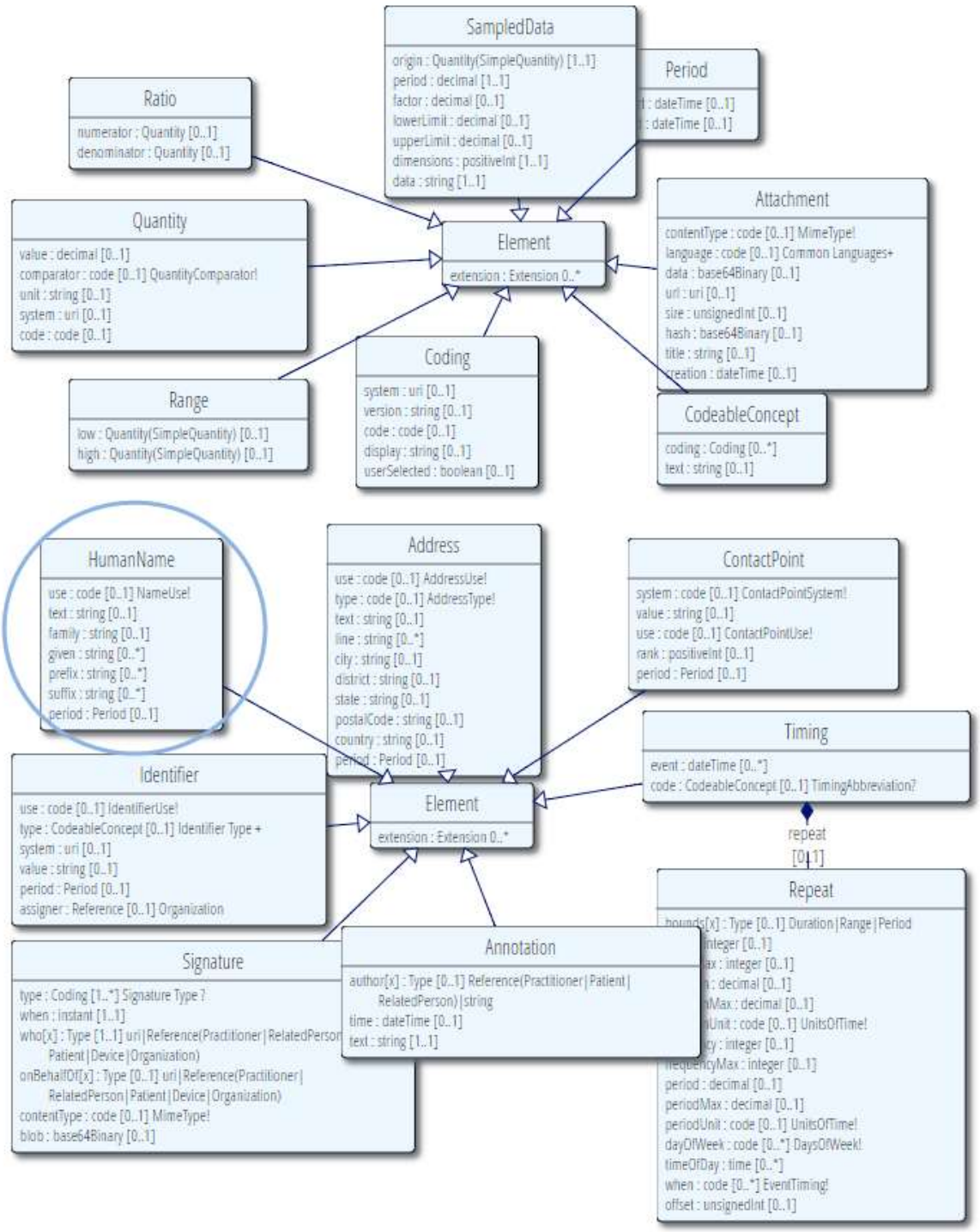


Figure 2.18: FHIR complex data types [47]

The following figure illustrates an example of a complex data type HumanName in XML format.

Structure | UML | XML | JSON | Turtle | R2 Diff | All

XML Template

```
<[name] xmlns="http://hl7.org/fhir">
 <!-- from Element: extension -->
 <use value="[code]"/><!-- 0..1 home | work | temp | old - purpose of this address -->
 <type value="[code]"/><!-- 0..1 postal | physical | both -->
 <text value="[string]"/><!-- 0..1 Text representation of the address -->
 <line value="[string]"/><!-- 0..* Street name, number, direction & P.O. Box etc. -->
 <city value="[string]"/><!-- 0..1 Name of city, town etc. -->
 <district value="[string]"/><!-- 0..1 District name (aka county) -->
 <state value="[string]"/><!-- 0..1 Sub-unit of country (abbreviations ok) -->
 <postalCode value="[string]"/><!-- 0..1 Postal code for area -->
 <country value="[string]"/><!-- 0..1 Country (e.g. can be ISO 3166 2 or 3 letter code) -->
 <period><!-- 0..1 Period Time period when address was/is in use --></period>
</[name]>
```

Figure 2.19: HumanName Complex data type [47]

In general, healthcare information modelling is achieved by using FHIR resources to define the information contents and the relations between different business objects of the most common use cases with the capability of using extensions for specific content requirements.

2.13.2 FHIR Infrastructure

From architectural perspective, FHIR is divided into four main components:

- Information Model (FHIR resources-related).
- Constraints (flexibility in different healthcare data system contexts).
- Terminology (ability to represent the values of various code systems).
- Usage (interoperability).

The figure below illustrates the relation between these components.

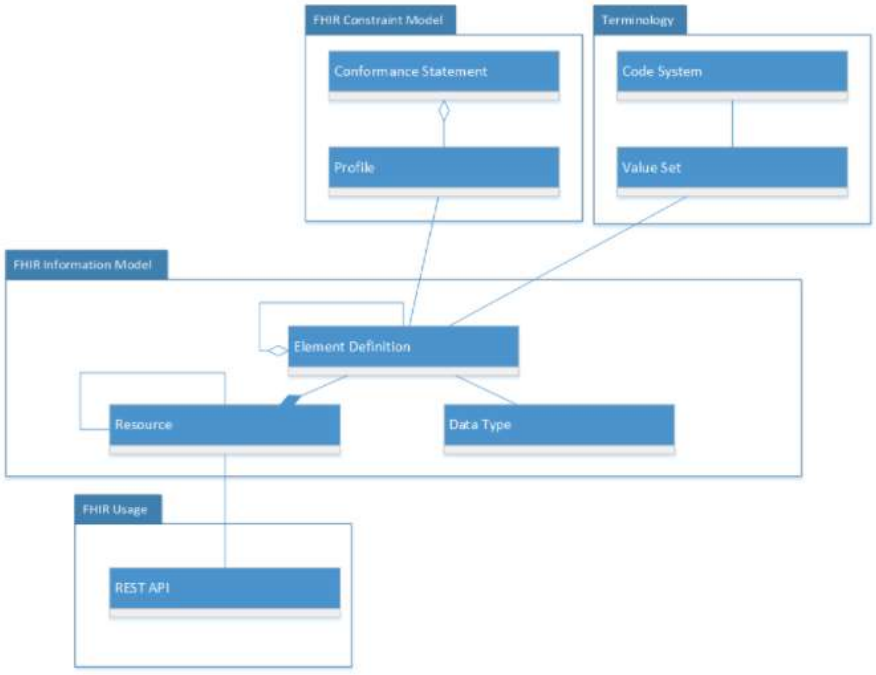


Figure 2.20: FHIR Specification Components [48]

2.13.2.1 FHIR Information Model

In order to model health data, FHIR information model represents the healthcare information domain as collection of small sub-domains. As FHIR resources are the basic building units, each sub-domain includes

related resources. The organization of FHIR resources is achieved in terms of layers of related categories. The purposes of such FHIR composition framework includes support of resources navigation and identification, grouping of related resources, flexibility in adding new resources as the top layers combine most commonly used resources. The figure below illustrates the organization of FHIR resources.

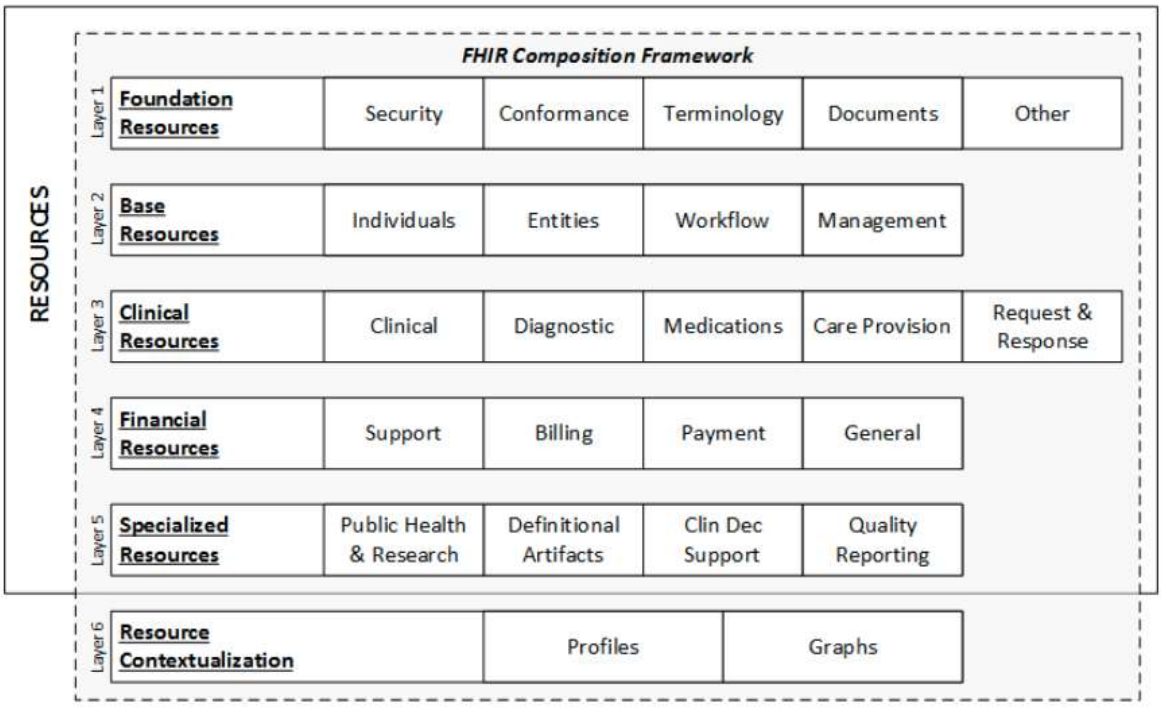


Figure 2.21: FHIR Composition Framework [48]

2.13.2.2 FHIR RESTful API

The term REST stands for **RE**presentational **S**tate **T**ransfer. In this context, the word "Representational" means that we request a representation of a specific resource, an instance of a resource, for example, some or the whole information about a patient by sending the patient's URI. The word "State" means that the information we request reflects the state of this particular instance of resource. The representation format can be in XML, JSON or CSV etc. The action of our request is in terms of HTTP verbs such as POST, GET, so HTTP is the interface between the client and the server that provides RESTful service. The response of our HTTP request includes the representation of the requested resource. Another word in this context is "stateless", which means that the server does not keep information about the client's state. In other words, from the client perspective, the response is stateful because it provides us with the state, information, about the requested resource. RESTful architecture supports the concept of separation of concerns by its layered architecture where there could be middle hardware or software between the client and the server. Consequently, RESTful architecture provides scalability which is a vital aspect of healthcare information systems.

FHIR RESTful API defines operations on resources in terms of interactions. There are several levels of interactions. "Instance" level of interactions includes actions such as read, update, and delete of instances. "Type" level interactions includes actions such as create and search for.

An example of a Read operation/interaction to retrieve a single instance of Patient resource looks like "GET http://fhir.someserver.org/fhir/Patient/1" and follows the definition:
VERB [base]/[type]/[id]
where VERB: GET,
service base URL: http://fhir.someserver.org/fhir,

resource type: Patient,
id: instance id value is 1 [49].

There are many interactions offered by FHIR REST API following the above definition [50]:

Create interaction:	POST	https://fhir.someserver.org/path/{resourceType}
Read interaction:	GET	https://fhir.someserver.org/path/{resourceType}/{id}
Update interaction:	PUT	https://fhir.someserver.org/path/{resourceType}/{id}
Delete interaction:	DELETE	https://fhir.someserver.org/path/{resourceType}/{id}
Search interaction:	GET	https://fhir.someserver.org/path/{resourceType}?search parameters

2.13.3 FHIR Key Concepts

From implementation perspective, there are main concepts in FHIR specification which we should understand in order to be able to map FEST data model and represent it using FHIR specification.

As we mentioned before, the main building unit of FHIR is resources which are used to store and exchange healthcare data. According to FHIR specification, each resource "has a known identity (a URL) by which it can be addressed, identifies itself as one of the types of resources defined in this specification, contains a set of structured data items as described by the definition of the resource type, and has an identified version that changes if the contents of the resource change" [51].

The identity of a resource is a logical identity for the resource which is given by the server storing it; an example of a logical id of a Patient resource can be http://fhir-server/Patient/123.

When the structure of a resource changed, the resource will have a version. For example, if the definition of patient resource is updated, then the old version can be accessed by using the URL

"http://fhir-server/Patient/123/_history/1" and the new version is accessible by using the URL

"http://fhir-server/Patient/123/_history/2".

"All resources have the following features in common:

- A URL that identifies the resource
- Common metadata
- A human-readable XHTML summary
- A set of defined data elements - a different set for each type of resource
- An extensibility framework to support variation in healthcare" [50].

In the context of FHIR specification, sometimes the terms "Resource" or "Resource instance" are used interchangeably which may lead to misunderstanding. We will use the term "Resource" when we refer to a resource as a data type. But when we refer to an instance of a resource, we use the term "Instance". According to FHIR specification, each instance of a resource is composed of resourceType, id, meta data, human readable text, extension, identifier, and data.

2.13.3.1 FHIR Profiling

The concept FHIR Profiling is intended to facilitate the customization of base FHIR specification in terms of the various local requirement of healthcare systems. Consequently, there should be some rules about which resource elements to be used, resource elements to be added, terminologies to be used [52].

A medication which has various levels of complication can be represented differently by using FHIR Profiling so that the local requirement can be handled [52]. A set of all customizations is gathered in "Implementation Guide" which organizes the customization in packages, each package includes a profile about the changes made to used resources. The following figure illustrates a simple example of Implementation Guide:

```
<ImplementationGuide>
    <meta />
    <url value="http://ehealth.uia.no/FHIR/profiles/FestImplementationGuide" />
    <version value="0.1" />
    <name value="FestImplementationGuide" />
    <status value="draft" />
    <date value="2018-04-22T17:47:06.513445+02:00" />
    <description value="FEST implementationguide for FHIR" />
    <fhirVersion value="3.0.1" />
    <package>
        <name value="FEST Medication" />
        <description value="Package for profiles related to FEST" />
        <resource>
            <example value="true" />
            <name value="Medication" />
            <description value="Profile for Medication" />
            <sourceReference>
                <reference value="http://ehealth.uia.no/FHIR/profiles/Medication" />
            </sourceReference>
        </resource>
    </package>

</ImplementationGuide>
```

Figure 2.22: ImplementationGuide Example

2.13.3.2 FHIR Extensions

"The paper forms (Resources) in FHIR are somewhat generic. They have to be usable in different countries and by different types of clinicians in different contexts (human care, veterinary care, public health, research, etc.). Recognizing that a one size fits all approach is not appropriate in the healthcare space, FHIR provides the ability to adjust the forms (Resources) to be able to handle the needs of different implementation spaces by defining "extensions" as well as enforcing constraints. For example, a "prescription" form might have extension elements added to support tracking of restricted medications while also constraining the codes that can be used to communicate types of drugs to a particular national standard. Forms are designed in such a way that these changes can be made without changing how systems pass forms around, enabling any system to consume completed forms even if they have additional elements added, whether or not those additional elements are used by the receiving system." [53].

The way FHIR Profiling handles the required customizations is achieved by adding structural definitions called "Extensions" where the added data elements are defined. For example, "MedicationStatement" is FHIR resource, organized under the Medication module, can be customized by adding an extension to refer to the prescriber who is responsible for issuing the prescription. The following figure illustrates an XML representation of the extension with a reference to "Practitioner" resource [54].

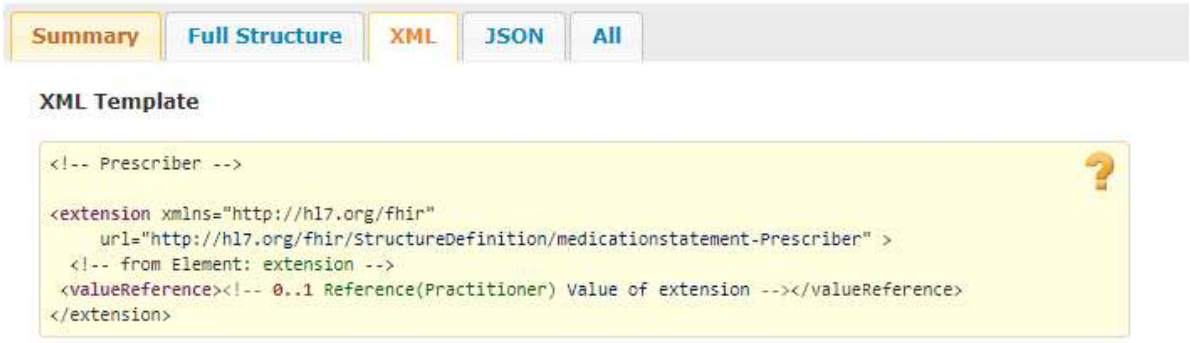


Figure 2.23: Extension Content Example [54]

Another example is the extension “doseType” which is part of the defined FHIR profile “International Realm Pharmacy Extensions”. The type of dose can be “initial”, “maintenance”, or “loading”. The following figure illustrates “doseType” extension content *[55]*.

Figure 2.24: doseType Extension Content [55]

2.13.3.3 FHIR Terminology

“The Terminology Module provides an overview and guide to the FHIR resources, operations, coded data types and externally-defined standard and FHIR-defined terminologies that are used for representing and communicating coded, structured data in the FHIR core specification and profiles. Collectively, these capabilities are used to provide the terminology service functionality required for supporting the use of coded data in FHIR resources throughout the specification as described in the other modules.” [56]. The figure below illustrates resources FHIR Terminology and the relationships between them.

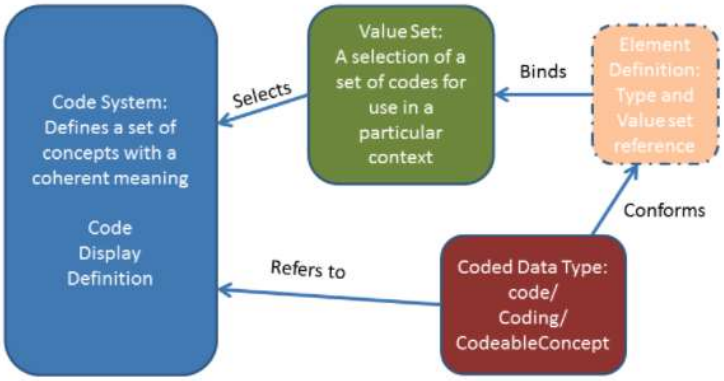


Figure 2.25: FHIR Terminology Resources and Relationships [56]

FHIR CodeSystem Resource defines a group of codes with their semantics. The following "ACME Codes for Cholesterol" is an example of a code system resource [57].

Code	Display	Definition
chol-mmol	SChol (mmol/L)	Serum Cholesterol, in mmol/L
chol-mass	SChol (mg/L)	Serum Cholesterol, in mg/L
chol	SChol	Serum Cholesterol

Figure 2.26: CodeSystem Example [57]

The Anatomical Therapeutic Chemical Classification System (ATC) is acknowledged by FHIR specification as an external published code system with URI “http://www.whocc.no/atc”. The following figure illustrates a representation in XML for the ACME Codes for Cholesterol CodeSystem.

```
<CodeSystem xmlns="http://hl7.org/fhir">
  <id value="example"/>
 <url value="http://hl7.org/fhir/CodeSystem/example"/>
  <identifier>
    <system value="http://acme.com/identifiers/codesystems"/>
    <value value="internal-cholesterol-inl"/>
  </identifier>
  <name value="ACME Codes for Cholesterol in Serum/Plasma"/>

  <concept>
    <code value="chol-mmol"/>
    <display value="SChol (mmol/L)"/>
    <definition value="Serum Cholesterol, in mmol/L"/>
  </concept>
  <concept>
    <code value="chol-mass"/>
    <display value="SChol (mg/L)"/>
    <definition value="Serum Cholesterol, in mg/L"/>
    </concept>
  <concept>
    <code value="chol"/>
    <display value="SChol"/>
    <definition value="Serum Cholesterol"/>
   </concept>
</CodeSystem>
```

Figure 2.27: ACME Codes for Cholesterol CodeSystem [58]

FHIR ValueSet is a collection of codes from one or more code systems; the following example illustrates ValueSet concept in FHIR:

```
<ValueSet xmlns="http://hl7.org/fhir">
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Possible Clinistix codes: neg, trace, +, ++, and +++</p>
    </div>
  </text>
  <url value="http://hl7.org/fhir/ValueSet/clinistix"/>
  <name value="Codes for Clinistix"/>
  <publisher value="HL7"/>
  <contact>
    <name value="FHIR project team"/>
    <telecom>
      <system value="url"/>
      <value value="http://hl7.org/fhir"/>
    </telecom>
  </contact>
  <description value="Clinistix Codes"/>
  <status value="draft"/>
  <experimental value="true"/>
  <date value="2013-10-01"/>
  <compose>
    <include>
      <system value="http://example.org/codes/simple-grades"/>
      <concept>
        <code value="neg"/>
      </concept>
      <concept>
        <code value="trace"/>
      </concept>
      <concept>
        <code value="+"/>
      </concept>
      <concept>
        <code value="+"/>
      </concept>
      <concept>
        <code value="++"/>
      </concept>
      <concept>
        <code value="+++"/>
      </concept>
    </include>
  </compose>
</ValueSet>
```

Figure 2.28: ValueSet of Codes for Clinistix [58]

2.13.4 FHIR Clinicians

We are going to focus on FHIR Clinical layer where most of resources that we are going to use in this thesis are categorized under the following modules [53]:

- Clinical Summary
- Care provision
- Diagnostics
- Medications
- Administrative

The use case of this thesis describes a practitioner who can display a list of medication statements which are currently active for a patient. The practitioner can read drug-drug interaction information related to the patient active medication statements. The practitioner decides to prescribe a new medicine. The practitioner always has access to an up-to-date medicine information. In other words, the practitioner does not need to worry about the update status of medicine information. In case a new drug-drug interaction information is added to this specific medicine, it will show up instantly providing the patient more safe medication by avoiding undesired consequences of drug-drug interaction .The instant update happens by an authorized member of the Norwegian Medicines Agency (SLV) who can search for a specific medicine, add a new medicine, update a medicine related information about interaction, and delete a medicine. In other words, any update to FEST XML database will be instantly reflected in the query result sent to the practitioner.

This scenario, include the following FHIR resources:
Medication: used to define and identify a medication with ingredients and packaging.
Medication Statement: a prescribed medication to a patient.
Medication Request : An order to supply a medication to a patient.
Patient: a person who needs medical help.
Practitioner: a physician.
DetectedIssue: represents a drug-drug interaction risk for this patient.

The relationships between the above-mentioned resources are expressed in terms of references, for example, the resource “Patient” has a reference “generalPractitioner” to resource “Practitioner”. The following diagram illustrates potential references:

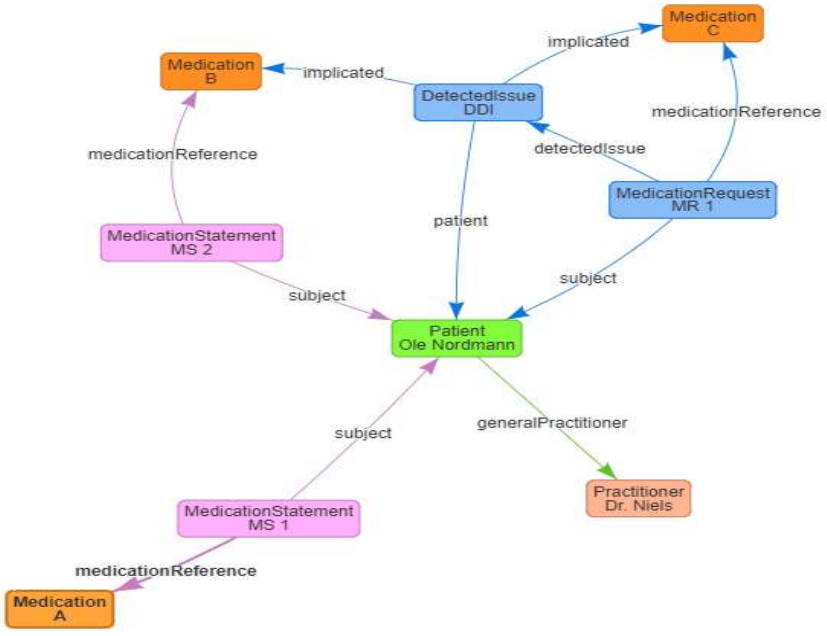


Figure 2.29: Use Case Resources References

2.14 HAPI-FHIR

HAPI-FHIR is a java implementation for FHIR specification developed by the University Health Network(UHN) group. UHN group describes HPAI-FHIR as "a chance to build our own integrated FHIR RESTful server which exposes data supported by several systems and repositories" [hapifhir.io/]. The following figures illustrate architectures that HAPI-FHIR supports and fit our solution implementation [20].

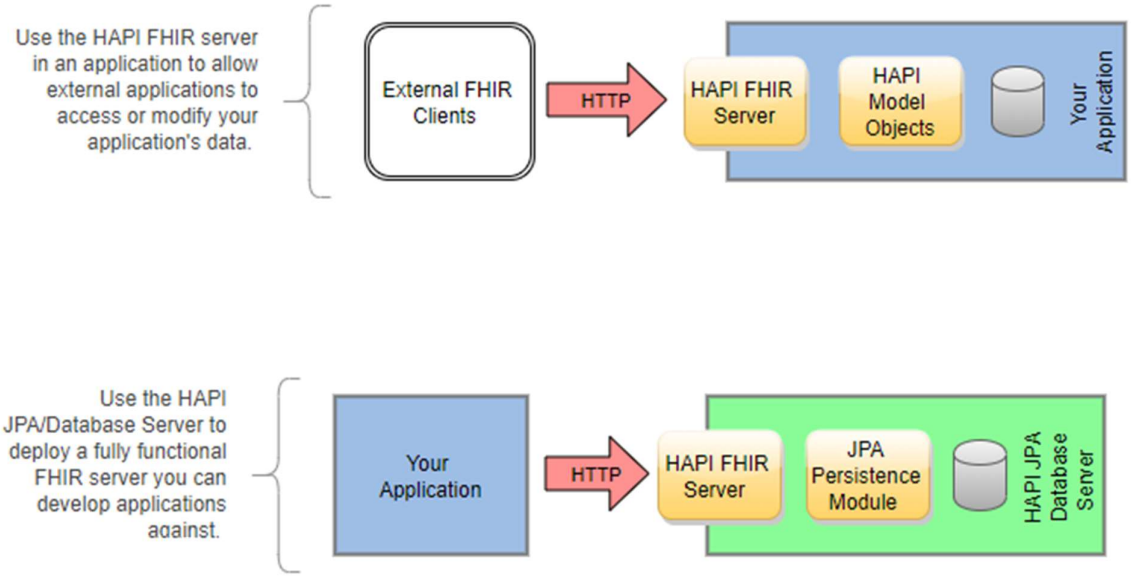


Figure 2.30: HAPI-FHIR Server [20]

2.14.1 FHIR Client

HAPI-FHIR library provides a RESTful client which supports searching for FHIR resource instances including various search parameters and returns a bundle of results that matches the specified query, if any. In addition, this RESTful client can be used to create, read, update, and delete FHIR resource instances.

2.14.2 JPA Persistence Module

HAPI-FHIR provides an embedded instance of a Java database called Apache Derby where FHIR resources instances are stored permanently. This database supports CRUD operations (create, read, update, and delete) on the stored FHIR resources instances.

2.15 Extensible Markup Language XML

In general, the syntax of XML language is like HTML language where data is represented by element tags. The key difference between XML and HTML is that XML is designed to carry, transport, data while HTML is designed only to display data along with using cascading style sheets for formatting purpose. XML is extensible because tags are not predefined, which mean we can define, invent, our own tags to carry data, but we must provide a schema file which describes the semantics of such defined tags.

In FEST, data about drugs are carried by XML tags, for example <legemiddelMerkevare> element includes all data about the medicine brand product. The semantics of this tag will not be clear unless the definition of its complex type is written inside a schema file which in turn must be referenced in the FEST XML file itself. In addition, FHIR specification uses XML to display the structure definition of FHIR resources. An instance of FHIR resource, medication for example, can be returned in XML format as well.

2.16 FEST Information Model

In this section, we are going to introduce an overview of the main components, catalogues, of FEST information model, which will be used in our solution implementation. We focus on the structure of each catalogue, and the connection between catalogues. In general, FEST catalogues are structured in a way that can match different purposes. In other words, a solution implementation based on FEST catalogues can be selective so that the solution can use some of the catalogues which meet its business logic [34]. Our main purpose here is to develop a good understanding of the catalogues we are going to represent in terms of FHIR data type structures with the necessary profiling to adopt the intended business logic of FEST.

2.16.1 Legemiddel (Medicine) Class

The Legemiddel (Medicine) class is an abstract class, circled in red , which is inherited by the main catalogues, circled in green, of FEST information model.

- LegemiddelMerkevare (MedicineBrandProduct)
- LegemiddelVirekestoff (MedicineActiveSubstance)
- LegemiddelPakning (MedicinePackage)
- LegemiddelDose (MedicineDose)

In each one of the main catalogues, there is a central class carries the same name of the catalogue and inherits the main class Legemiddel.

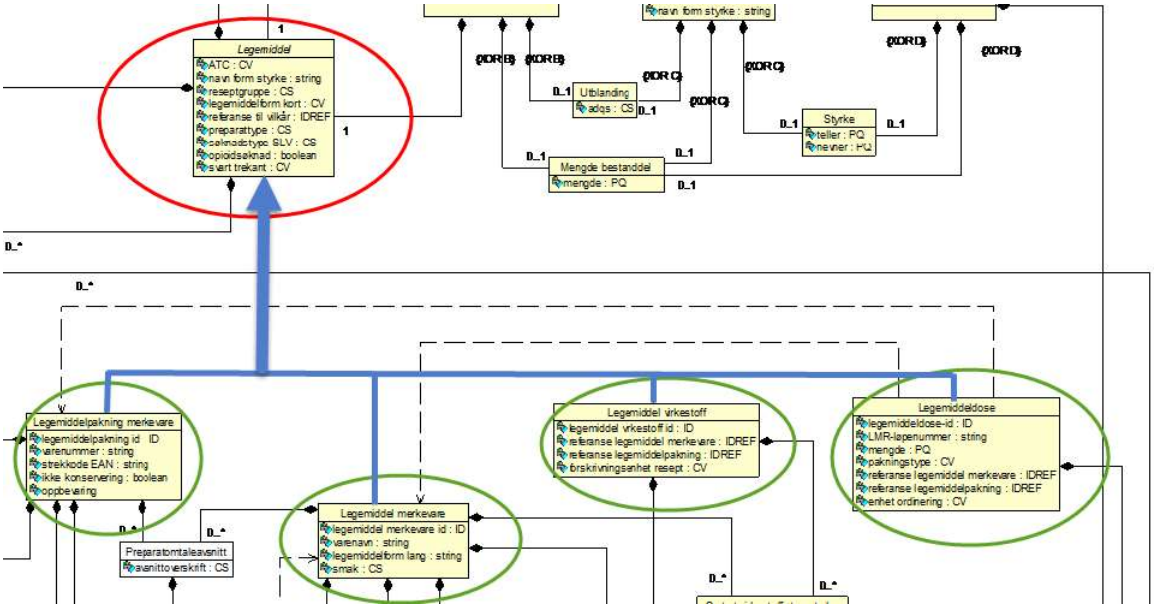


Figure 2.31: Class Legemiddel(Medicine) [34]

2.16.2 LegemiddelMerkvare (MedicineBrandedProduct) Catalogue

The LegemiddelMerkvare catalogue represents information about “prescription of a strength and form of a specific branded product” [34]. The following figure shows the information represented by this catalogue along with what is inherited from the main class Legemiddel (Medicine) [34].

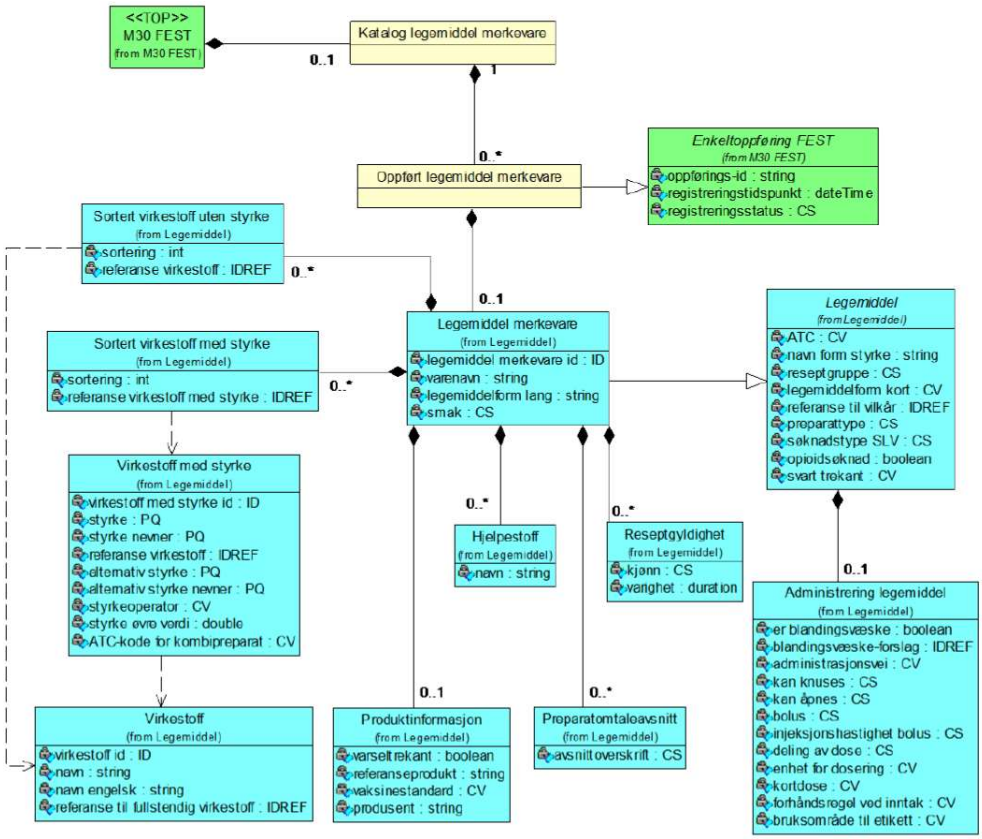


Figure 2.32: LegemiddelMerkevare (MedicineBrandedProduct) Catalogue Content [34]

2.16.3 LegemiddelVirkestoff (MedicineActiveSubstance) Catalogue

The LegemiddelVirkestoff catalogue represents information about "prescription of active pharmaceutical ingredients" [34]. The following figure shows the information represented by this catalogue along with what is inherited from the main class Legemiddel (Medicine) [34].

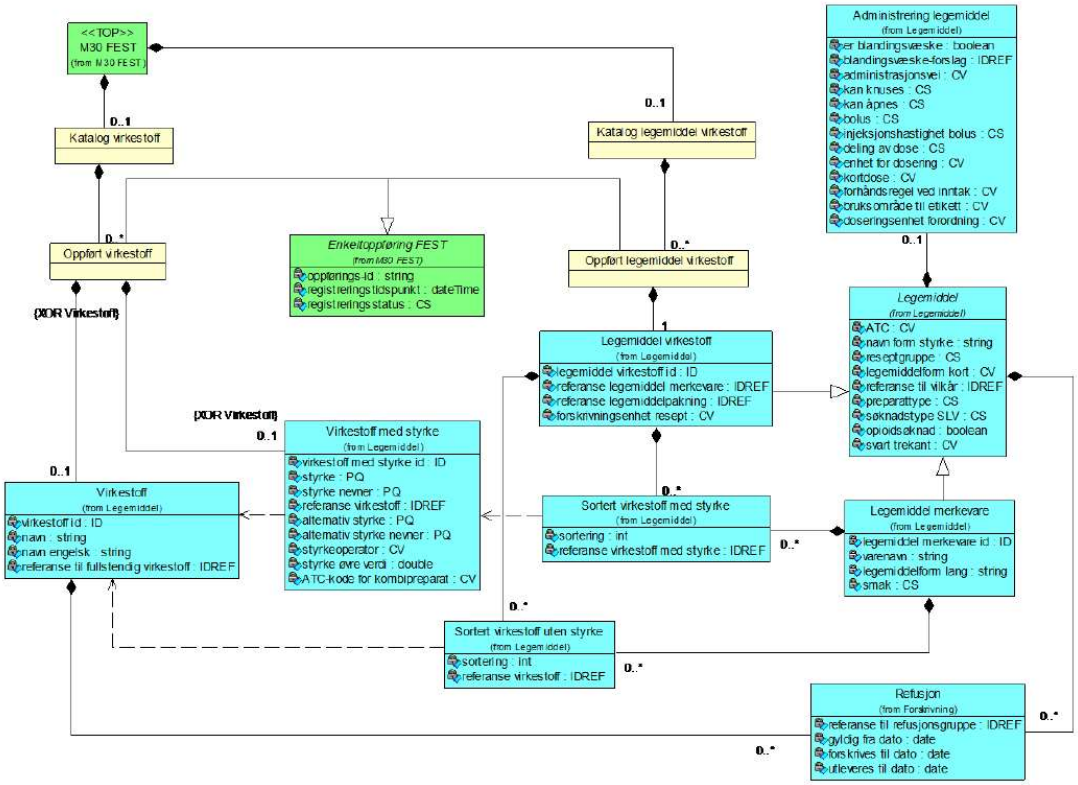


Figure 2.33: LegemiddelVirkestoff (MedicineActiveSubstance) Catalogue Content [34]

2.16.4 LegemiddelPakning (MedicinePackage) Catalogue

The LegemiddelPakning catalogue represents information about "prescription of a specific package of a branded product (article number)". The following figure shows the information represented by this catalogue along with what is inherited from the main class Legemiddel (Medicine) [34].

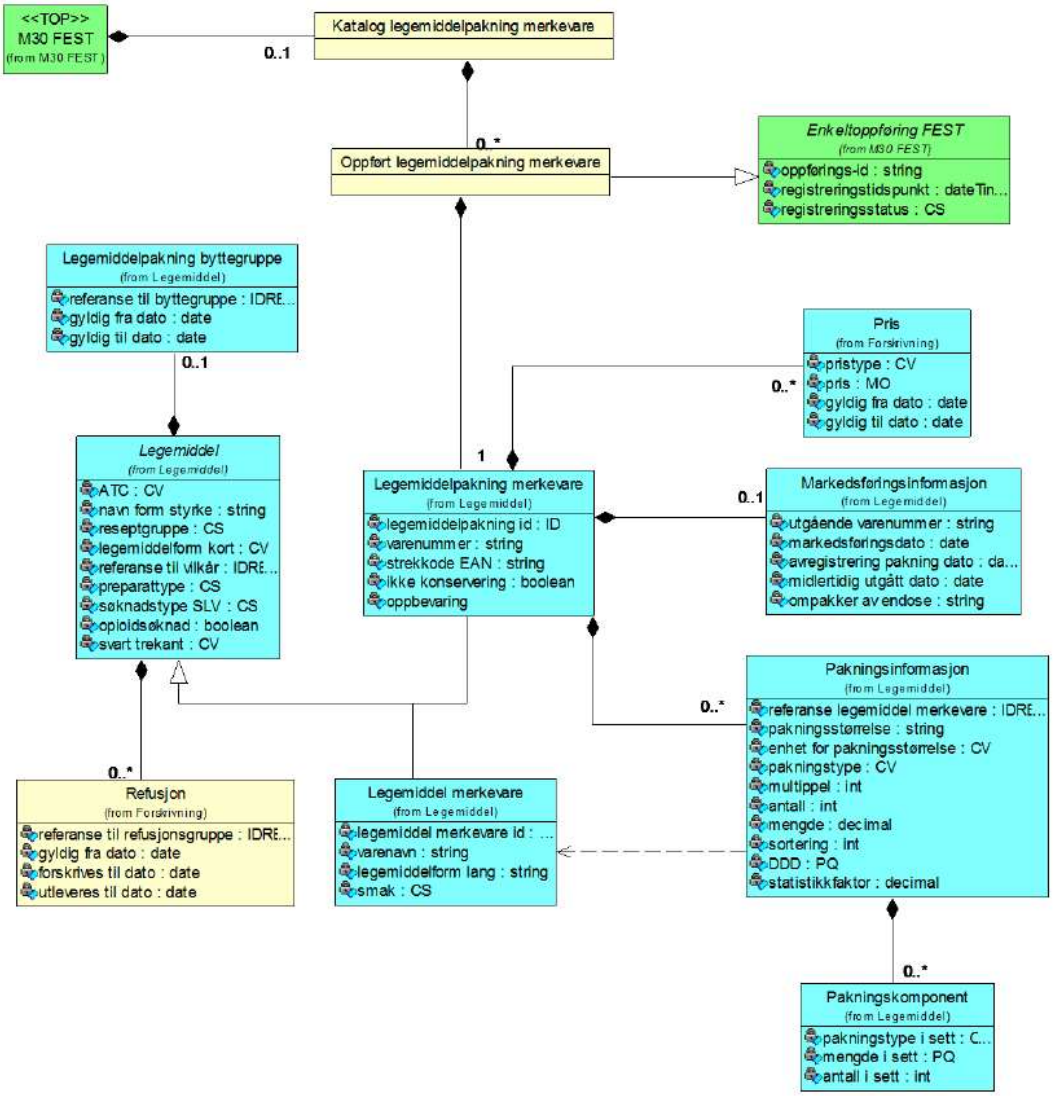


Figure 2.34: The LegemiddelPakning Catalogue Content [34]

2.16.5 LegemiddelDose (MedicineDose) Catalogue

The LegemiddelDose catalogue represents information about "prescription of a specific branded product with ID (LMR number) representing the smallest selectable unit, for instance, one ampoule or one tablet." [34] The following figure shows the information represented by this catalogue along with what is inherited from the main class Legemiddel (Medicine) [34].

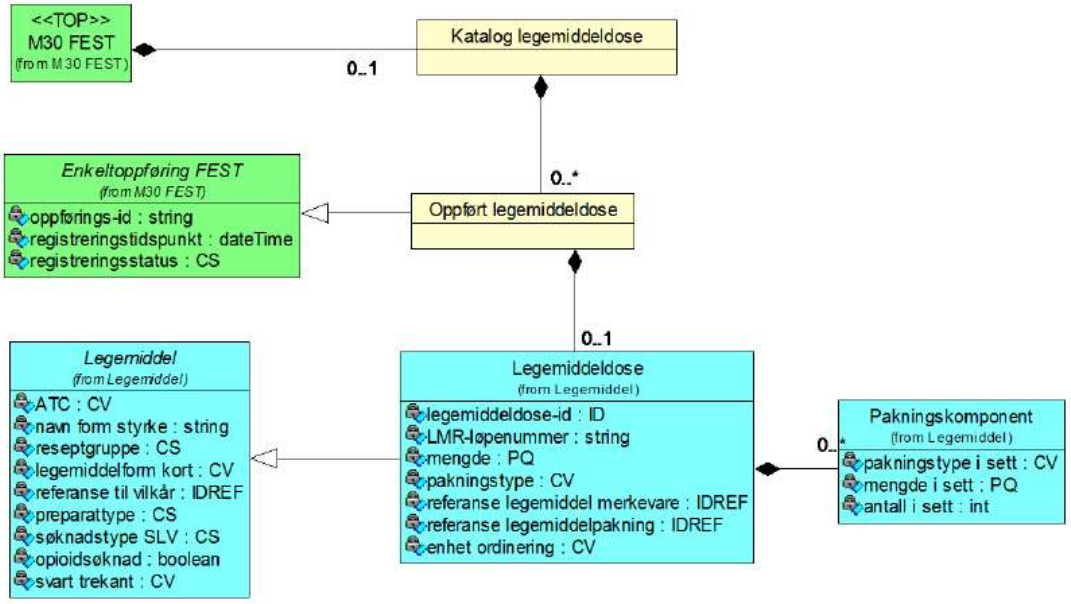


Figure 2.35: The LegemiddelDose Catalogue Content [34]

2.16.6 Interaksjoner (Interactions) Catalogue

The Interaksjoner catalogue represents information related to interactions between two substance groups. Let's say we have two medicines and we need to check drug-drug interaction between two medicines. In case the active substance of one medicine exists in one substance group, and the active substance of the other medicine exists in the second substance group, then there is an interaction. The severity of interaction can be "should be avoided", "precaution should be taken", or "no action necessary" [34]. Clinical consequences of such interaction provide information about the clinical risk resulted from the interaction. Each active substance is identified by its ATC-code.

2.16.7 FEST and FHIR

In our published paper, "Mapping FHIR Resources to Ontology for DDI reasoning", we used FHIR resources to extend the capabilities of Forskrivnings- og expedisjonsstøtte (FEST). We extracted
Interaction and medication data from FEST and represent it in terms of FHIR resources on HAPI-FHIR RESTful server. Then we used FHIR resource instances to build an ontology representing a family history use case and applied drug-drug interaction reasoning [59].

2.17 Forge FHIR Profile Designer

"Forge is the official HL7®FHIR® profile editor. Tailoring the FHIR specification to a specific use-case is an important aspect of the FHIR standard. This process is also known as *profiling*. While the FHIR spec targets all countries and all use cases, profiling allows you to define local use of FHIR." [60]. In this thesis, we use Forge FHIR Profile Designer to create and validate profiles and extensions.

2.18 Java XML Integration

When it comes to parsing XML content, we have to consider many factors before choosing the right API to start with; available memory, XML content size, and the complexity are no exception. The content of FEST XML file is huge, and it keeps growing because of the ongoing process of update carried out by Statens Legemiddelverket (SLV). Consequently, we consider using a Java API for XML processing which provides

us with a binding processor where the process of deserialization XML into Java classes works efficiently, and it's easy to maintain as well. Our choice is JAXB binding processor API.

2.19 Apache Maven

"Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information." [61]. In our implementation of the proposed solution of this thesis, we are going to use Apache Maven in two cases. The first case when we deserialize FEST XML content into Java Classes model to represent FEST information model. Apache Maven will be used in the "pom" file of our Java application to manage the dependency of the software components used in our project, compile and build the solution as well. The second case when we install our HAPI-FHIR server, where we need to compile and build the source code of the server before we install it. In addition, we will be adding some required configurations to the server to fit our work and recompile the server using Apache Maven.

2.20 Web Development

In general, the term web development describes a broad development environment where client-server model is adopted. On the web server side, all the files which are required to render the business logic and user interface are stored. On the client side, a web browser should be able to render the web page content sent by the web server as a response to the user's request.

2.20.1 Apache Tomcat Web Server

"The Apache Tomcat® software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies." [62]. In our implementation of the proposed solution of this thesis, we are going to use Apache Tomcat web server in two cases. The first case to deploy our HAPI-FHIR server in which FEST resource instances are persistently stored in the built-in Apache Derby database. The second case, to deploy our demo for the web components we will develop to display FEST medicines and drug-drug interaction information. All the software components we will develop through this thesis are coded in Java programming language.

3 Implementation of Solution

The goal behind this thesis is to map FEST domain model into HL7 FHIR resources-based model to support interoperability between various health information systems which require instant access to an up-to-date information about medicines in Norway. This chapter describes in detail the phases we have gone through, the functionalities we have developed in each phase, and the tools we have used in this thesis in order to achieve our goal of having FEST on FHIR. This chapter can also be considered as a manual for developers who need guidelines to implement a similar solution. The implementation is described in a step by step manner by following the time and sequence of operations. I provide visual illustrations through screenshots along with comments on code snippets.

In this thesis, we adopted an Object-Oriented Modelling (OOM) approach to construct collections of objects representing the FEST information model by using Java 8 programming language capabilities which provides us with object identification and allows for communication between objects along with all features of inheritance and encapsulation. In other words, OOM is the conceptual basis for understanding the concepts behind FEST data structures and the mapping of FEST.

In our implementation, we adopted a use-case driven approach in which we extract the required FEST information we need to represent drug-drug interaction. In addition, we have followed HL7 FHIR specifications in order to represent the extracted FEST data in terms of FHIR resources and retain the relations between FEST data in terms of FHIR resources references.

The following diagram illustrates briefly the processes workflow, top-down, we have implemented to achieve the above-mentioned goal. Each row represents a process starts from left to right where the process is defined as a task, then the steps and methodology we have taken to finally reach the outcome of the process at the most right of each row. Each process strongly depends on the outcome of its former process, which means we had to work in sequence.

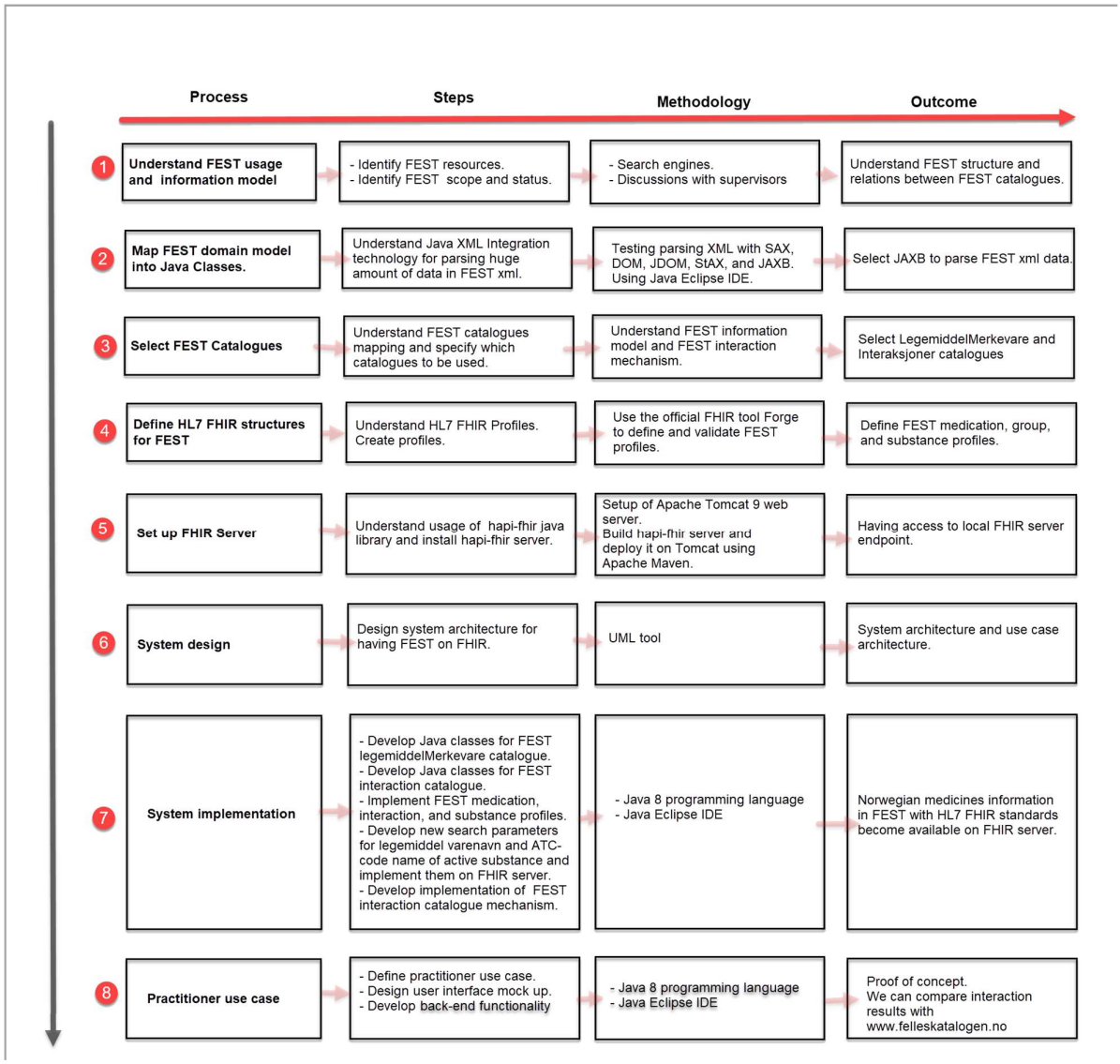


Figure 3.1:Solution processes workflow

3.1 Understand FEST usage and information model

As we mentioned in the previous chapter M30 FEST comes in the format of an xml file that comprises various elements where some elements can have attributes in the form of key-value pairs. Understanding the semantics of the structures in M30 FEST xml file is a vital point for the next processes of parsing the file. The following two diagrams illustrates how the M30 FEST information model is defined in UML and in terms of an XML schema in order.

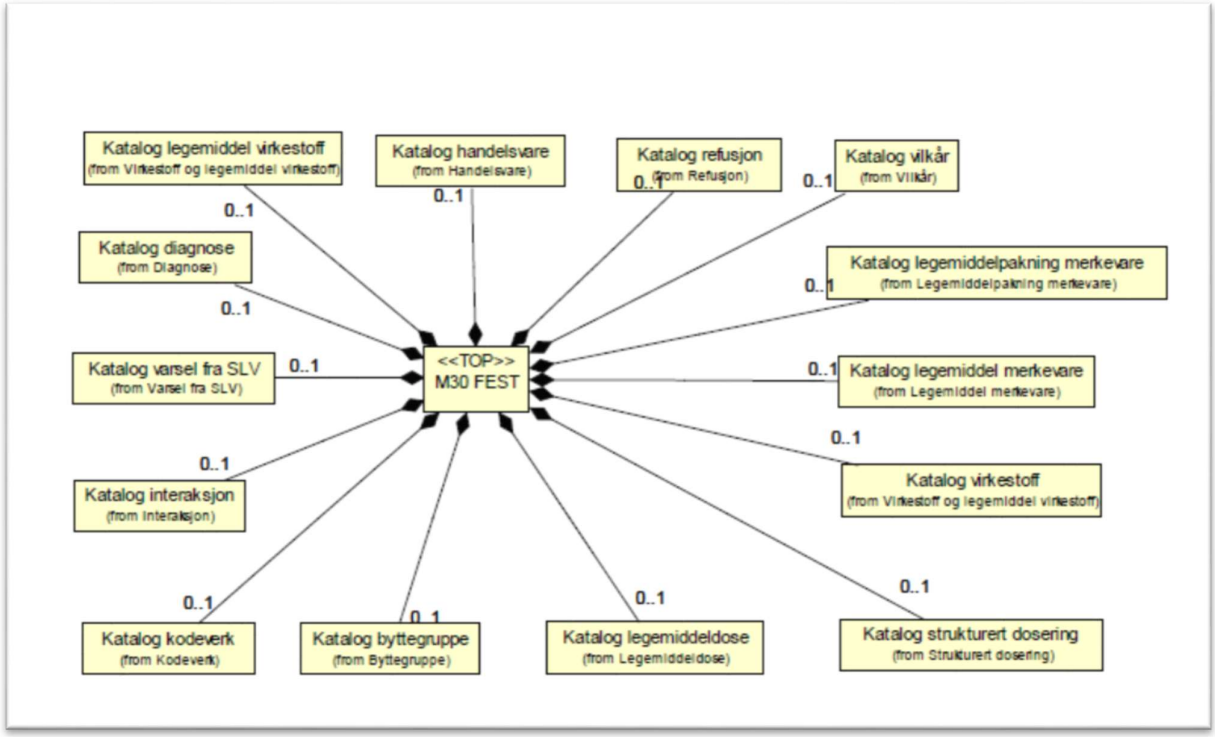


Figure 3.2: M30 FEST Information Model [34]

Figure 3.3: FEST complex type

Each M30 FEST catalogue is represented as an xml child element under FEST element and has its own definition as a complex type which comprises all child elements. For example, KatLegemiddelMerkevare catalogue is defined in the following diagram:

```
<element name="KatLegemiddelMerkevare">
    <annotation>
        <documentation>Katalog legemiddel merkevare</documentation>
    </annotation>
    <complexType>
        <sequence>
            <element name="OppfLegemiddelMerkevare" minOccurs="0" maxOccurs="unbounded">
                <annotation>
                    <documentation>Oppført medisinsk produkt</documentation>
                </annotation>
                <complexType>
                    <complexContent>
                        <extension base="m30:typeEnkeltoppforingFest">
                            <sequence minOccurs="0">
                                <element ref="fs:LegemiddelMerkevare"/>
                            </sequence>
                        </extension>
                    </complexContent>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

Figure 3.4 LegemiddelMerkevare catalogue definition in XML schema

It is shown from the above diagram that the catalogue has OppfLegemiddelMerkevare child element which represents a single entry for each LegemiddelMerkevare instance under the catalogue.

Understanding the data types used to represent M30 FEST information is also important before we start parsing the xml file. The primitive data types used in FEST are based on ISO's language independent datatypes. There are complex data types such as CV used to represent the identity of each medicine according to the ATC coding system. The following snippet illustrates CV complex type.

```
<complexType name="CV">
    <attribute name="V" type="token" use="optional"/>
    <attribute name="S" type="kith:oid" use="optional"/>
    <attribute name="DN" type="string" use="optional"/>
    <attribute name="OT" type="string" use="optional"/>
</complexType>
```

The following snippet illustrates how in a legemiddelmerkevare instance, the attribute V represents the ATC-code given to a legemiddel and the attribute DN represents the display name of the active substance of this legemiddel.

```
<LegemiddelMerkevare xmlns="http://www.kith.no/xmlstds/eresept/forskrivning/2014-12-01">
  <Atc V="H01BA02" S="2.16.578.1.12.4.1.1.7180" DN="Desmopressin" />
```

The importance of understanding mapping from one catalogue to another is no exception for our work in the next sections, so we tried to follow the UML diagrams and mapping notes offered in the implementation guide of FEST *[34]*. For example, there is a direct reference from LegemiddelVirkestoff catalogue to the Legemiddelmerkevare catalogue by using IDref (direct reference). Another example is to use IDref to map from LegemiddelMerkevare catalogue to get information about the relevant medicine packages in LegeMiddelPakningMerkevare catalogue.

The following snippet illustrates how an actual medicine is represented in LegemiddelMerkevare catalogue.

```
<LegemiddelMerkevare xmlns="http://www.kith.no/xmlstds/eresept/forskrivning/2014-12-01">
  <Atc V="H01BA02" S="2.16.578.1.12.4.1.1.7180" DN="Desmopressin" />
  <NavnFormStyrke>Minirin Tab 0,1 mg</NavnFormStyrke>
  <Reseptgruppe V="C" DN="Reseptgruppe C" />
  <LegemiddelformKort V="53" S="2.16.578.1.12.4.1.1.7448" DN="Tablett" />
  <RefVilkar>ID_491E7F6C-23A9-45C3-A906-95A1332AD5BD</RefVilkar>
  <Preparattype V="11" DN="Krever godkj. Fritak" />
  <TypeSoknadSlv V="2" DN="Søknad vurderes av apotek" />
  <AdministreringLegemiddel>
    <Administrasjonsvei V="53" S="2.16.578.1.12.4.1.1.7477" DN="Oral bruk" />
    <EnhetDosering V="13" S="2.16.578.1.12.4.1.1.7480" DN="tablett" />
  </AdministreringLegemiddel>
  <Id>ID_000C0AFE-42AE-4522-AE22-BAAF5746D81E</Id>
  <Varenavn>Minirin</Varenavn>
  <LegemiddelformLang>Tablett</LegemiddelformLang>
  <SortertVirkestoffMedStyrke>
    <Sortering>0</Sortering>
    <RefVirkestoffMedStyrke>ID_1F464BED-F3CE-43C8-87D3-024E3202DD48</RefVirkestoffMedStyrke>
  </SortertVirkestoffMedStyrke>
  <ProduktInfo>
    <Produsent>Ferring</Produsent>
  </ProduktInfo>
  <Reseptgyldighet>
    <Varighet>P1Y</Varighet>
  </Reseptgyldighet>
</LegemiddelMerkevare>
```

Figure 3.5: Medicine instance in LegemiddelMerkevare catalogue

3.2 Map FEST domain model into Java classes model

In order to choose the right Java XML API to parse FEST XML file, we had to read about a set of Java APIs for XML Processing (JAXP). We had to take in consideration the size and complexity of FEST XML file and to pay attention to the required memory to process such a huge XML file. In addition, as our task in this section is to read FEST xml file and create its corresponding Java classes model, our focus was on JAXP APIs that meet our needs. The main idea is to read the xml file and use the annotations in the schema files of FEST to generate the Java classes model.
We started by testing SAX which is a simple API for xml, it can read the XML file, but it works as a streaming API with events such as start document, end document. So it did not help in creating Java classes model. We had the same result with DOM, JDOM, StAX APIs.

Java Architecture for XML Binding (JAXB) API is our best choice because it reads the XML file and uses the annotations defined in XML elements and attributes to bind the XML content into corresponding java classes with properties and the required accessor methods.

We developed a maven project in Eclipse with the following directory structure:

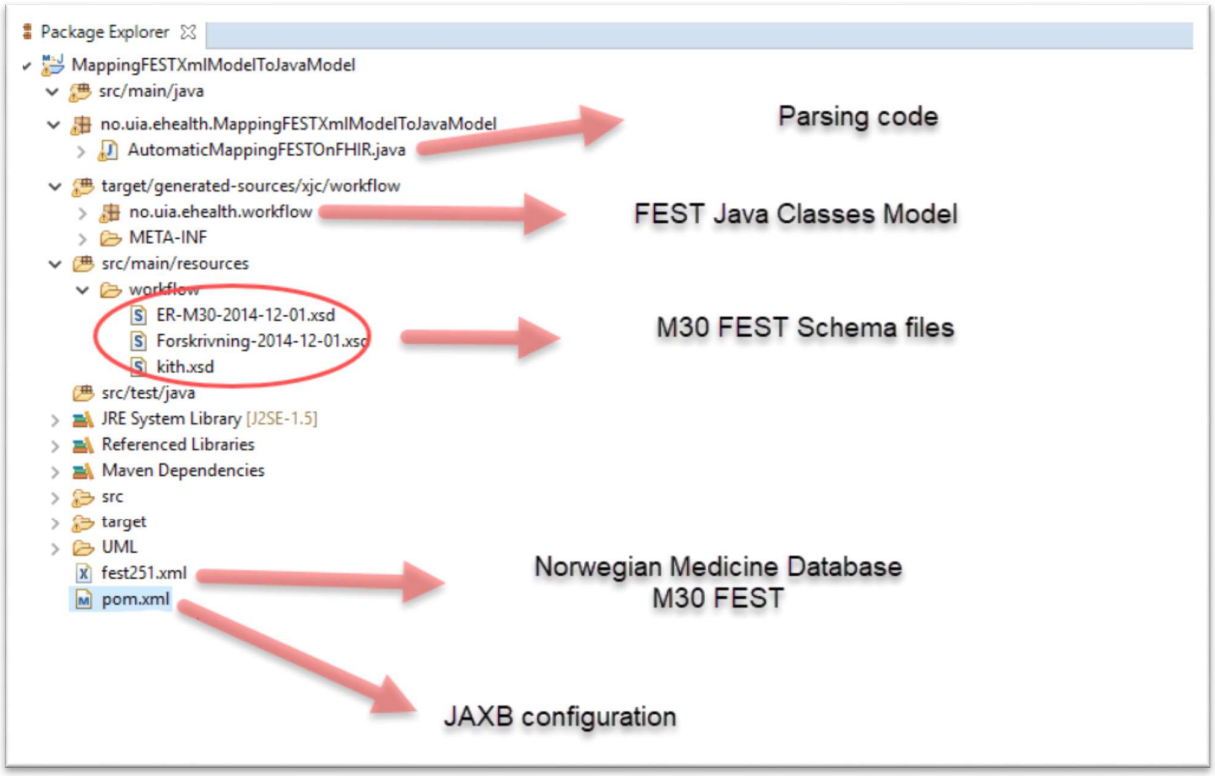


Figure 3.6: Parsing FEST

3.2.1 Functional requirements

In this section, we explain how we added the required Java Architecture XML binding library (JAXB2) to our Java project which we used to compile FEST XML schema in order to deserialize the XML content of FEST file and generate the Java classes model. The following snippet code illustrates how we used the library.

```
// create JAXB context and generate FEST information model java classes
JAXBContext context = JAXBContext.newInstance("no.uia.ehealth.workflow");
Unmarshaller um = context.createUnmarshaller();
// read FEST 2.5.1 xml file
BufferedInputStream bis = new BufferedInputStream(new FileInputStream(new File(XML_FILE)));

try {

    FEST fest = (FEST) um.unmarshal(bis);// End of Generate java classes for FEST  informat
```

Id	1	**Requirement type**	Functional requirement 1	**Event**	Setup project config. file pom.xml
Description	Add required configuration to pom file and configure maven plugins.				
Rationale	To add required libraries/API to parse FEST xml file and generate Java classes model.				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 1: POM file configuration

The configuration file of our Maven project is “pom.xml”, which is used to manage the dependencies of the software components/plugins we use in our project in addition to determine the build configuration of the project. The following snippet illustrates the content of our POM file related to parse FEST XML file.

Figure 3.7: Dependency configuration

In the build section of pom file, we add the following plugins.

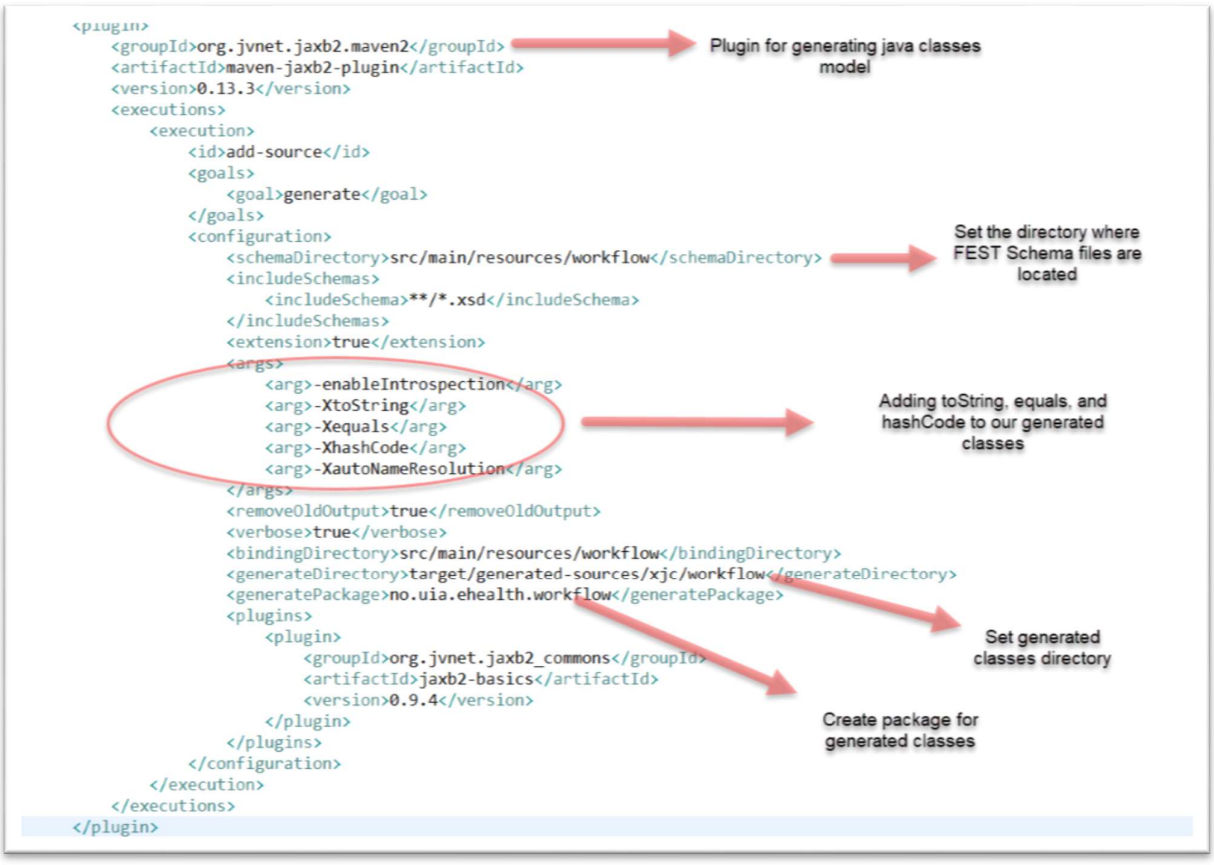


Figure 3.8: Configuration of JAXB2 plugin

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <version>1.9</version>
    <executions>
        <execution>
            <id>add-source</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>add-source</goal>
            </goals>
            <configuration>
                <sources>
                    <source>${basedir}/target/generated-sources/xjc/workflow</source>
                </sources>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Set generated source code directory as a source directory in the project.

Figure 3.9: Set source directory plugin

As shown in the figure below, inside class AutomaticMappingFESTOnFHIR there is our main method where we start the execution of the application and call the constructors of other classes to proceed with mapping.

```
import java.io.BufferedInputStream;

public class AutomaticMappingFESTOnFHIR {
    // private static final String XML_FILE = "miniFest.xml";
    private static final String XML_FILE = "fest251.xml";
    private static List<MedicationFHIRInformation> medicationFHIRInformationList = new ArrayList<MedicationFHIRInformation>();
    private static List<InteractionFHIRInformation> interactionFHIRInformationList = new ArrayList<InteractionFHIRInformation>();
    private static final String baseURL = "http://localhost:8081/fest/baseDstu3"; // to be replaced with url of production web server
    private static FhirContext ctx = FhirContext.forDstu3();

    public static void main(String[] args) throws JAXBException, FileNotFoundException {

        // **************************Generate FEST java class model**************************************
        // For more details about the Apache Maven configuration behind FEST java model
        // generation | please refer to the "pom.xml" file of this project. A lot of setup, testing,
        // installation and  re-installation efforts have been paid to get the model mapped into java
        // classes model.
        // **********************************************************************************************

        // create JAXB context and generate FEST information model java classes
        JAXBContext context = JAXBContext.newInstance("no.uia.ehealth.workflow");
        Unmarshaller um = context.createUnmarshaller();
        // read FEST 2.5.1 xml file
        BufferedInputStream bis = new BufferedInputStream(new FileInputStream(new File(XML_FILE)));

        try {

            FEST fest = (FEST) um.unmarshal(bis);// End of Generate java classes for FEST  information model
```

Set FEST xml file

Create a context object of JAXB API and assign the generated package name we defined in "pom.xml" file.

Unmarshaller means parser from XML to Java

Read the file

Parsing result stored in an instance of FEST class.
The fest variable is an instance of FEST xml file but in terms of Java model. In other words, all the content of FEST xml file are mapped and can be accessed through fest object. All the relations, navigation capabilities between FEST catalogues are mapped and can be represented in terms of Java classes hierarchy, inheritance , methods and properties inside classes.

Figure 3.10: Mapping FEST to Java model

3.2.2 FEST Java Classes Model

Our FEST Java classes model fully represents FEST data structures. By using the Java Architecture for XML binding (JAXB), we managed to compile FEST schema and generate Java classes for each complex type of FEST. Each generated Java class represents all the XML elements of its corresponding FEST complex type as properties. In addition, Each Java class has getters and setters methods (accessor methods) which are required to access its properties and connect the class instance with other parts of FEST as well. The generated Java classes model reflects and keeps the cardinalities and relations between FEST catalogues in the same way as FEST data structures are designed. All the generated Java classes are available in a GitHub repository, for more details refer to Appendix D: FEST Java Classes Model.

We are going to represent some of them in the following sections. The following diagram illustrates the created FEST catalogues Java classes.

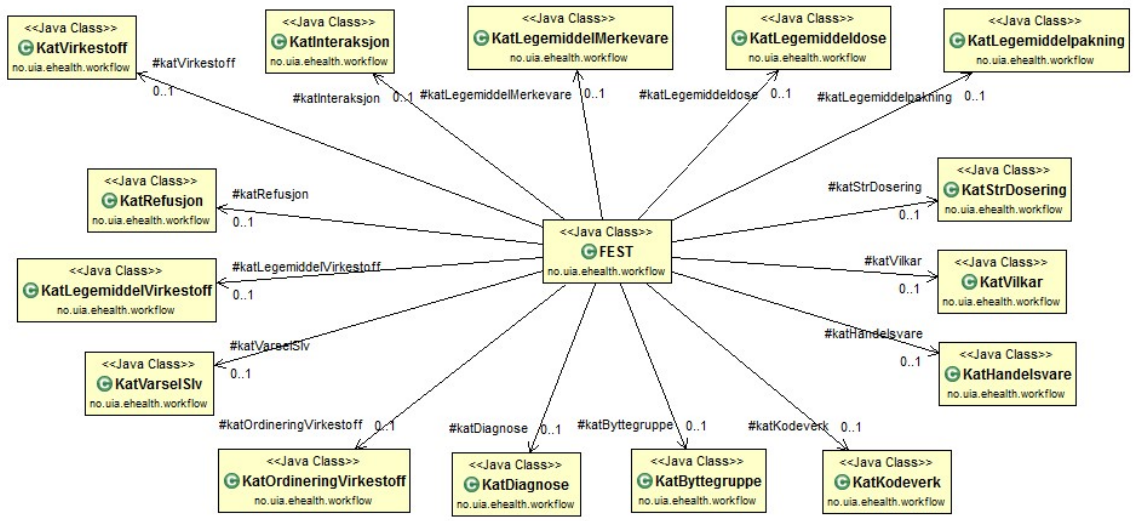


Figure 3.11: FEST catalogues Java classes

3.2.2.1 FEST Class

The following two diagrams illustrate how the properties of the FEST class generated in our solution completely represents the FEST complex type defined by FEST schema.

```
<element name="FEST">
    <complexType>
        <sequence>
            <element name="HentetDato" type="dateTime"/>
            <element name="GyldigFradatoHelfo" type="date" minOccurs="0"/>
            <element ref="m30:KatLegemiddelMerkevare" minOccurs="0"/>
            <element ref="m30:KatLegemiddelpakning" minOccurs="0"/>
            <element ref="m30:KatVirkestoff" minOccurs="0"/>
            <element ref="m30:KatOrdineringVirkestoff" minOccurs="0"/>
            <element ref="m30:KatLegemiddelVirkestoff" minOccurs="0"/>
            <element ref="m30:KatHandelsvare" minOccurs="0"/>
            <element ref="m30:KatDiagnose" minOccurs="0"/>
            <element ref="m30:KatRefusjon" minOccurs="0"/>
            <element ref="m30:KatVilkar" minOccurs="0"/>
            <element ref="m30:KatVarselSlv" minOccurs="0"/>
            <element ref="m30:KatKodeverk" minOccurs="0"/>
            <element ref="m30:KatByttegruppe" minOccurs="0"/>
            <element ref="m30:KatLegemiddeldose" minOccurs="0"/>
            <element ref="m30:KatInteraksjon" minOccurs="0"/>
            <element ref="m30:KatStrDosering" minOccurs="0"/>
        </sequence>
    </complexType>
</element>
```

Figure 3.12: FEST complex type

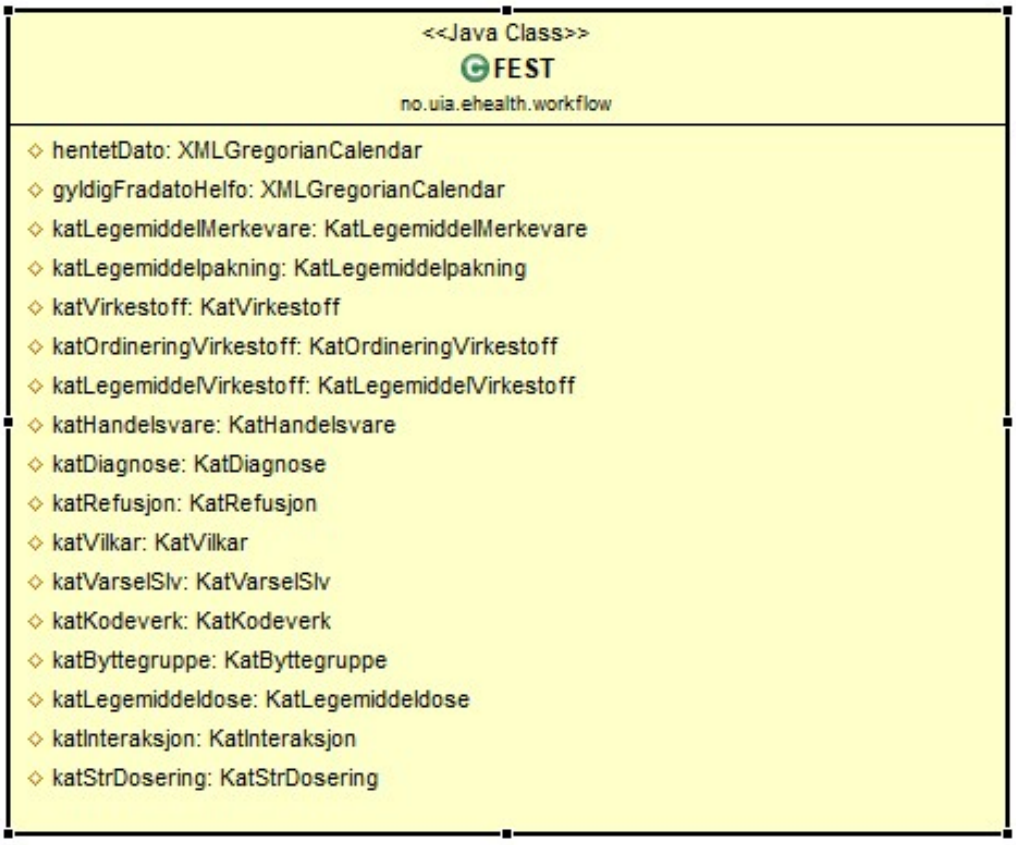


Figure 3.13: FEST class properties

The following diagram illustrates the getters, setters, and the constructor of the FEST Java class.

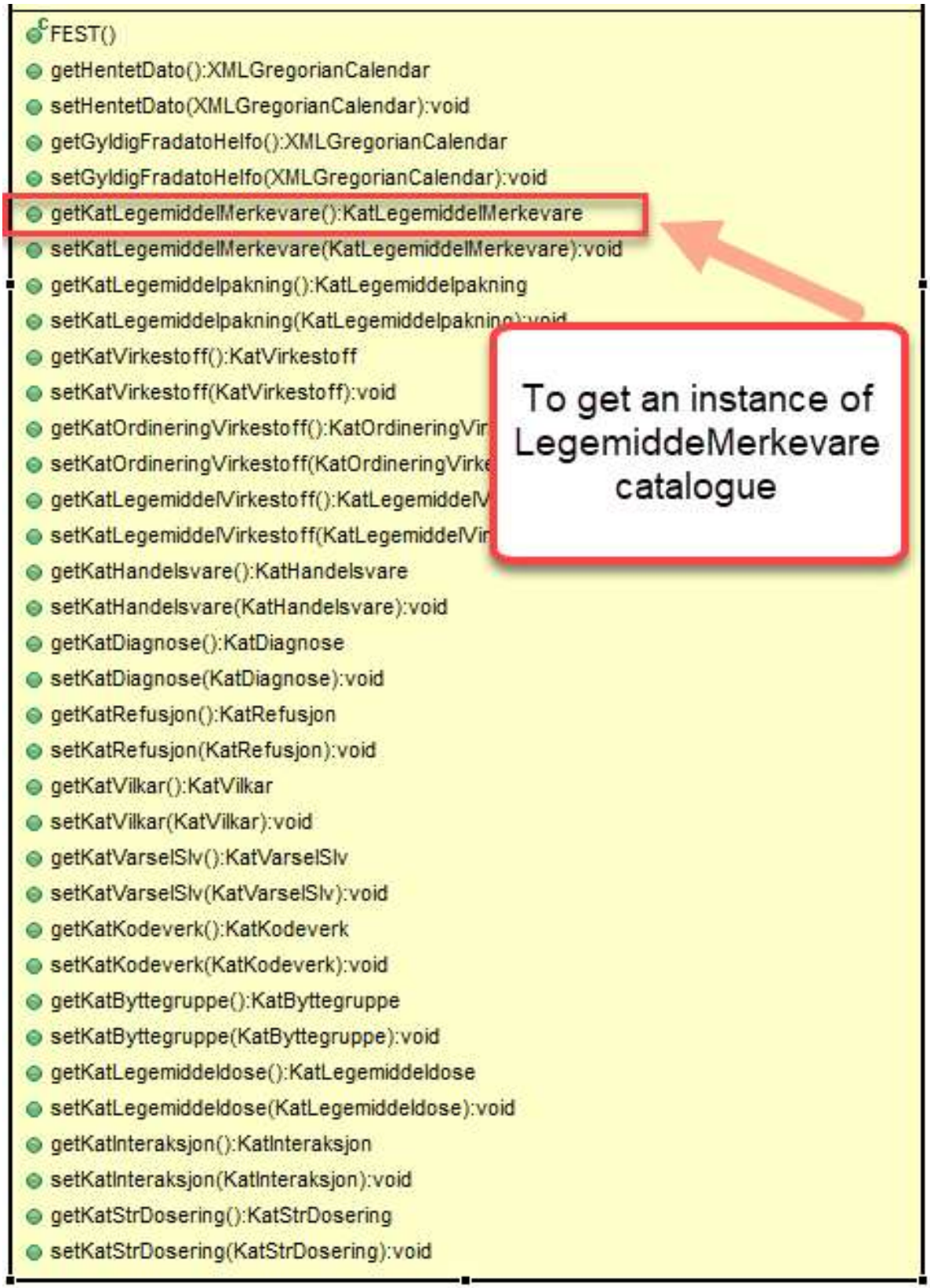


Figure 3.14: FEST class methods and constructor

3.2.2.2 LegemiddelMerkevare Catalogue

The following UML diagram represents *LegemiddelMerkavare* catalogue in our FEST Java classes model.
The UML diagram in the right side is taken from the information model documentation of FEST [34].

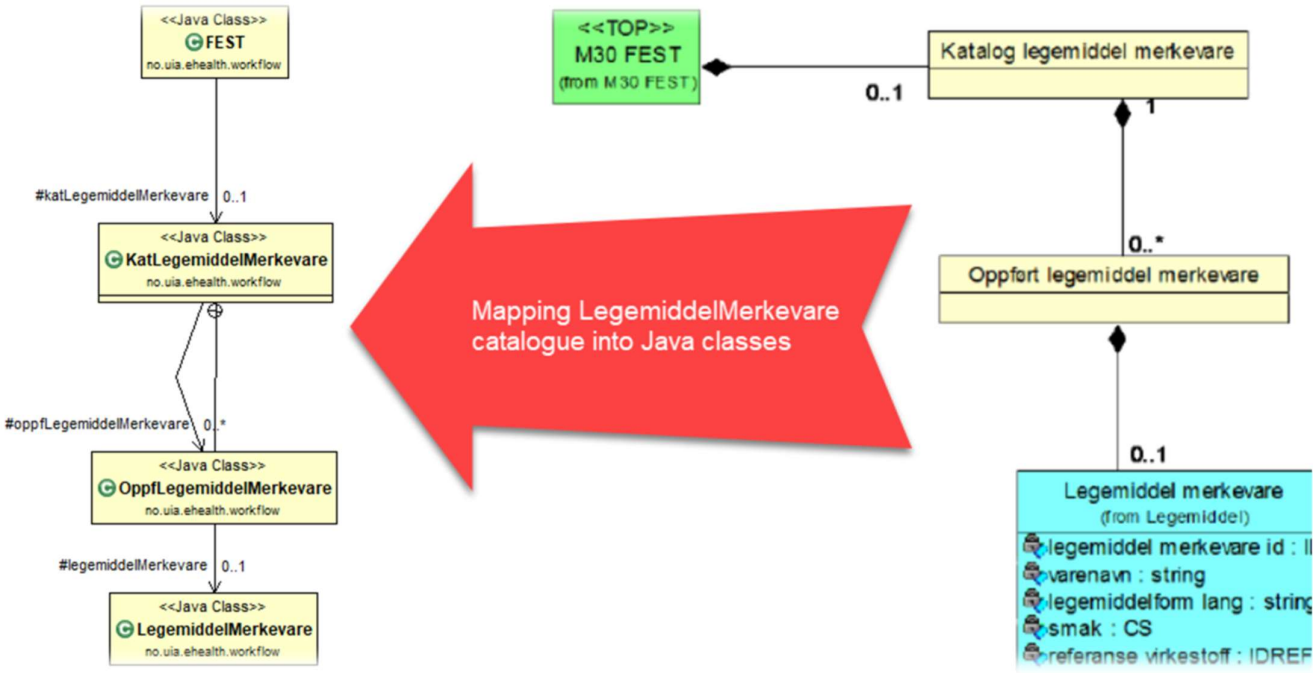


Figure 3.15: Mapping LegemiddelMerkevare catalogue into Java classes

3.2.2.3 Interaction Catalogue

The following UML diagram represents Interaction catalogue in our FEST Java classes model.
The UML diagram in the right side is taken from the information model documentation of FEST [34].

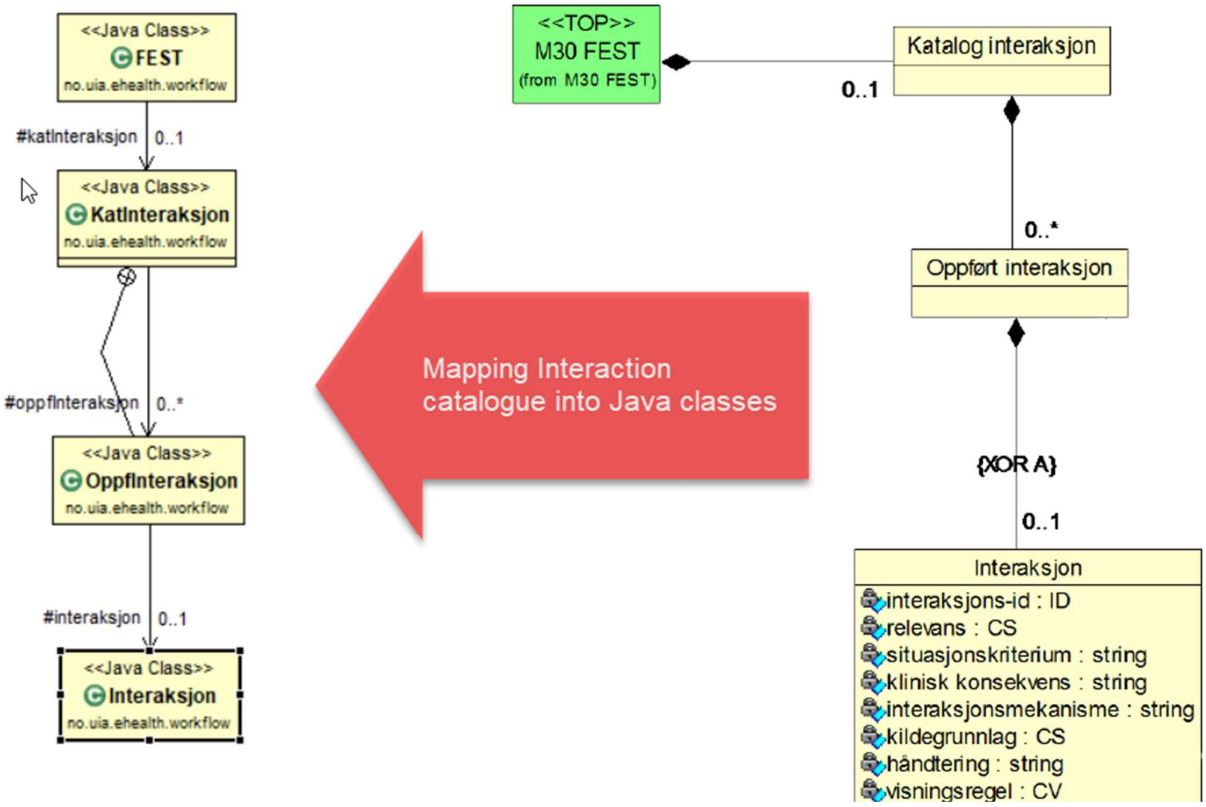


Figure 3.16: Mapping Interaction catalogue into Java classes

3.2.3 Non-functional requirement

The compilation of FEST schema should be fully completed without any compiling errors.
Java jdk 1.8.0_171 is used in our Java development environment.
Compilation and building of the solution is managed by Apache Maven version 3.5.2 which must be installed properly. The solution is written in Java so it is platform independent and can run on Windows, Mac, or Unix/Linux operating systems.

3.3 Select FEST catalogues

The outcome of the previous step is that FEST data is fully mapped to FEST Java classes (i.e., instances the Java classes produced by the parser). An object of class FEST contains all FEST data with all required methods to navigate between FEST catalogues. In this thesis, we focus on two catalogues; *LegemiddelMerkevare*, and *Interaksjon*.

3.3.1 LegemiddelMerkevare class

LegemiddelMerkevare Java class, shown in the figure below, is the core of *LegemiddelMerkevare* catalogue. The class has accessor methods (getters and setters) which provides connections with other catalogue of FEST. For example, the getter method getProduktInfo() returns information about the packaging of the medicine and the producer. By using such methods, we can retrieve information from various catalogues. On the other hand, the setter method setProduktInfo(ProduktInfo) can be used to update information about packaging and producer of the medicine.

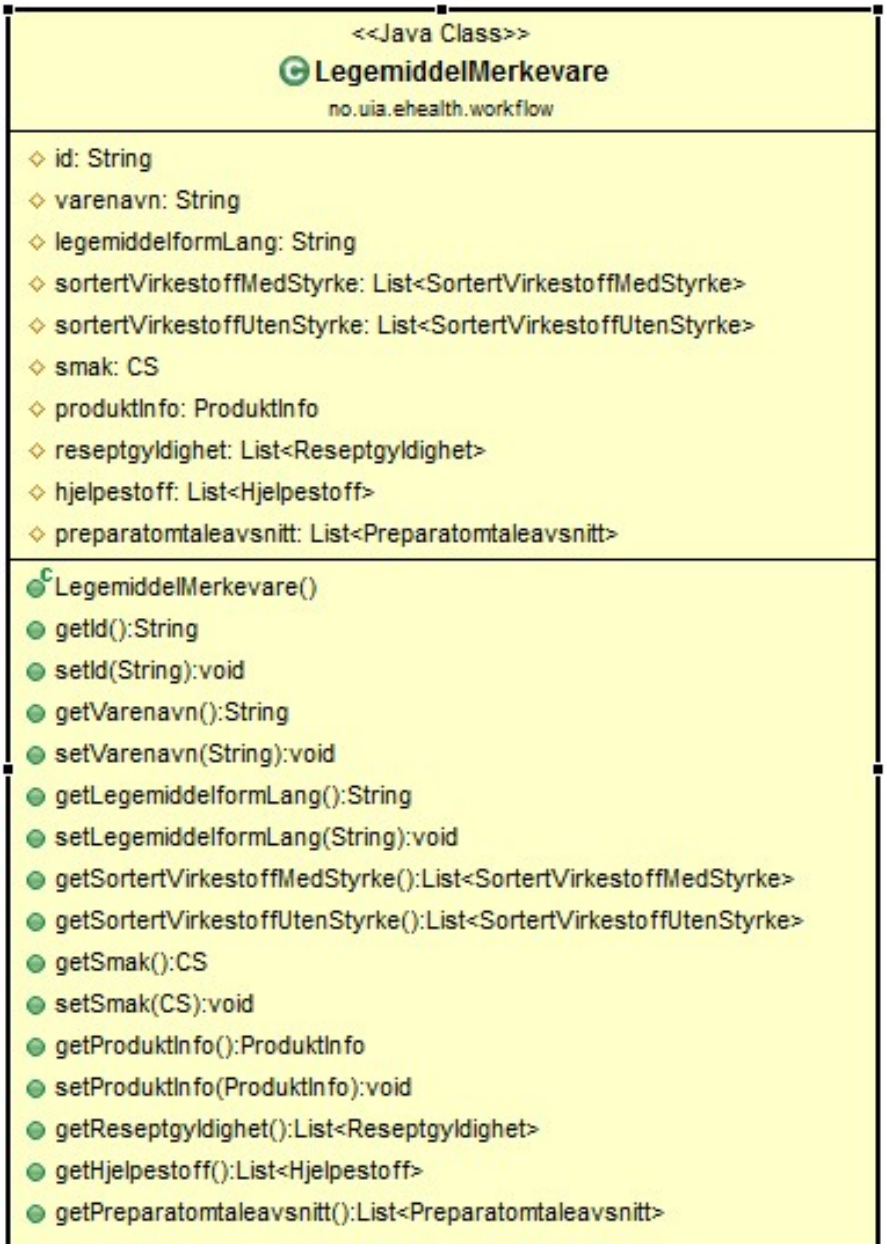


Figure 3.17: LegemiddelMerkevare Java Class

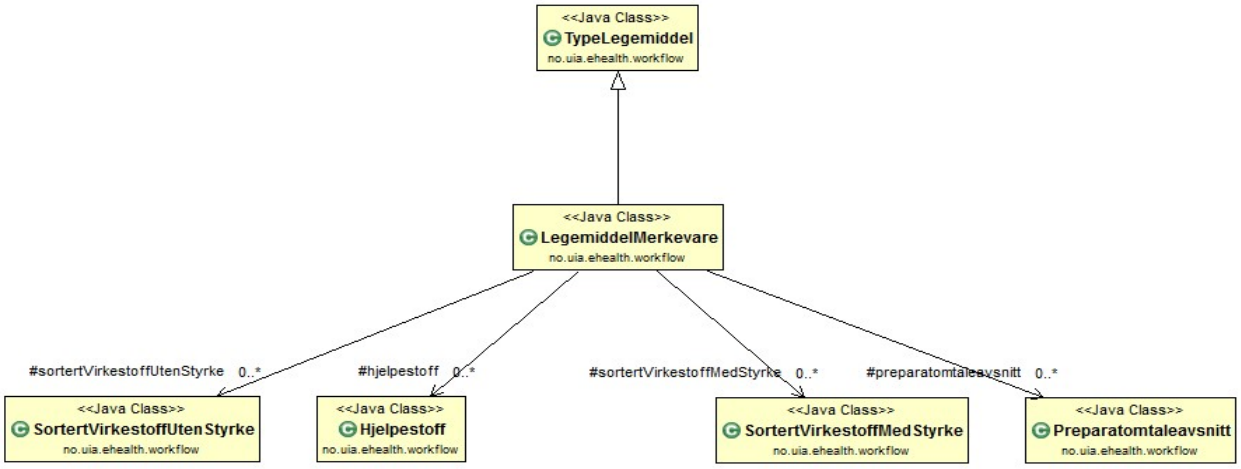


Figure 3.18: LegemiddelMerkevare class relations

Class *LegemiddelMerkevare* extends class *TypeLegemiddel*. Consequently, all the data elements defined for a legemiddel by FEST schema and mapped to FEST *TypeLegemiddel* Java class become totally accessible in *LegemiddelMerkevare* class. In addition, the getter and setter methods of *TypeLegemiddel* provide connection between *LegemiddelMerkevare* class and the other main catalogues of FEST. For example, to get information about the dosage of a medicine, we can use getAdministreringLegemiddel() which in turn returns dosage information such as unit dose.

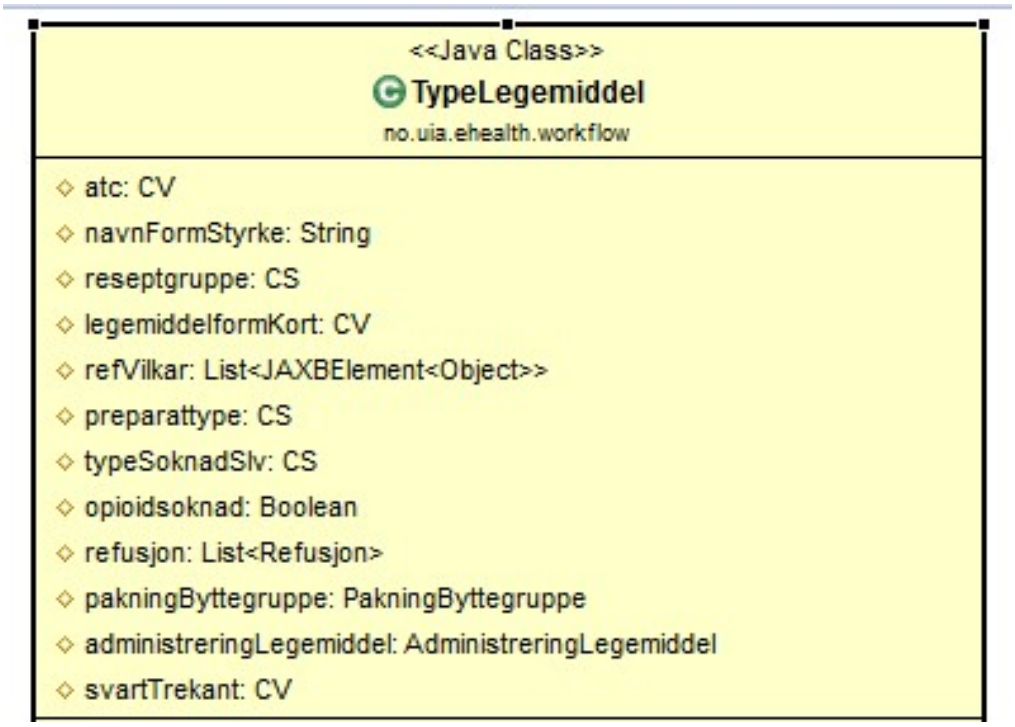


Figure 3.19: TypeLegemiddel Java class

3.3.2 Interaksjon class

Interakjson class is the core of the *Interaksjon* catalogue; it defines all the data elements defined by FEST schema about interaction along with the required getter and setter methods required to communicate with other parts of the FEST model.

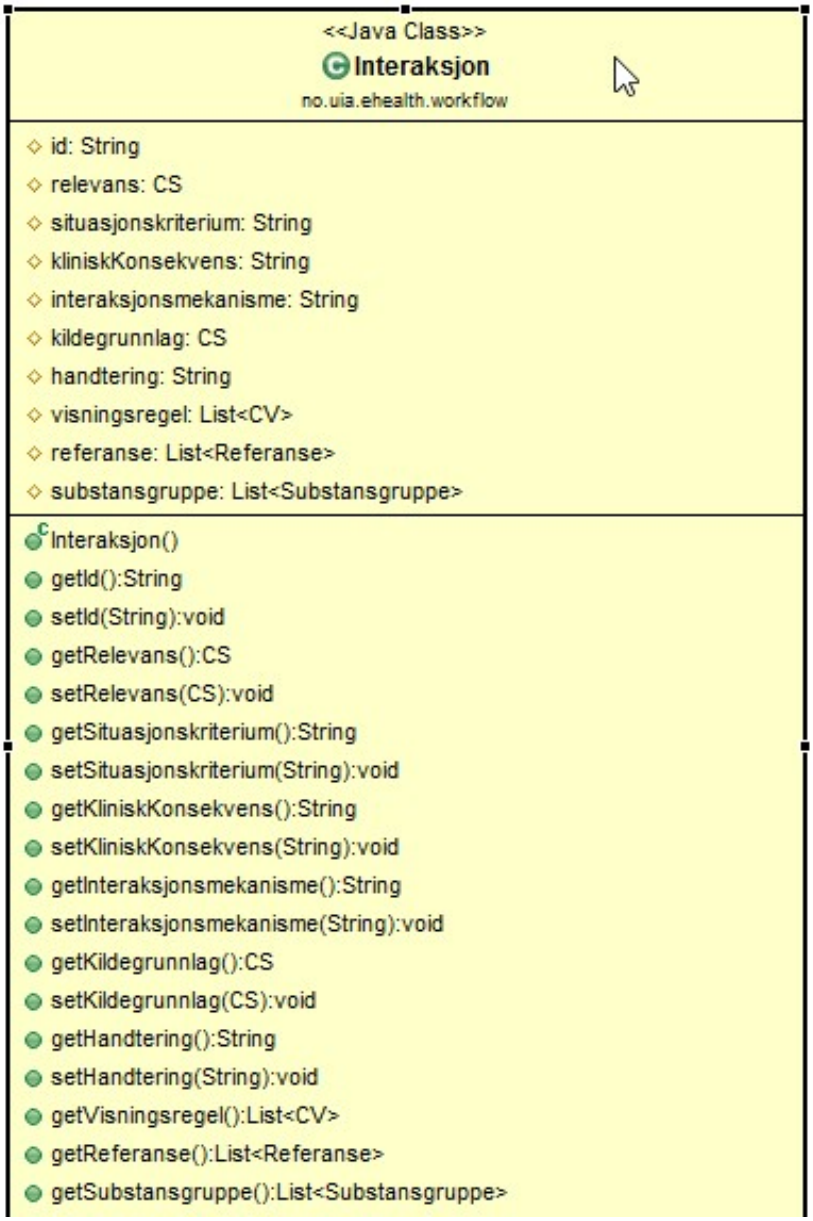


Figure 3.20: Interaskjon class

3.3.2.1 FEST Interaction mechanism

Let's say we have two medicines and we need to check drug-drug interaction between them. In case the active substance of one medicine exists in one substance group, and the active substance of the other

medicine exists in the second substance group, then there is an interaction [34]. The following flowchart illustrates FEST logic of finding interaction. More details about how we implemented this mechanism in system implementation section.

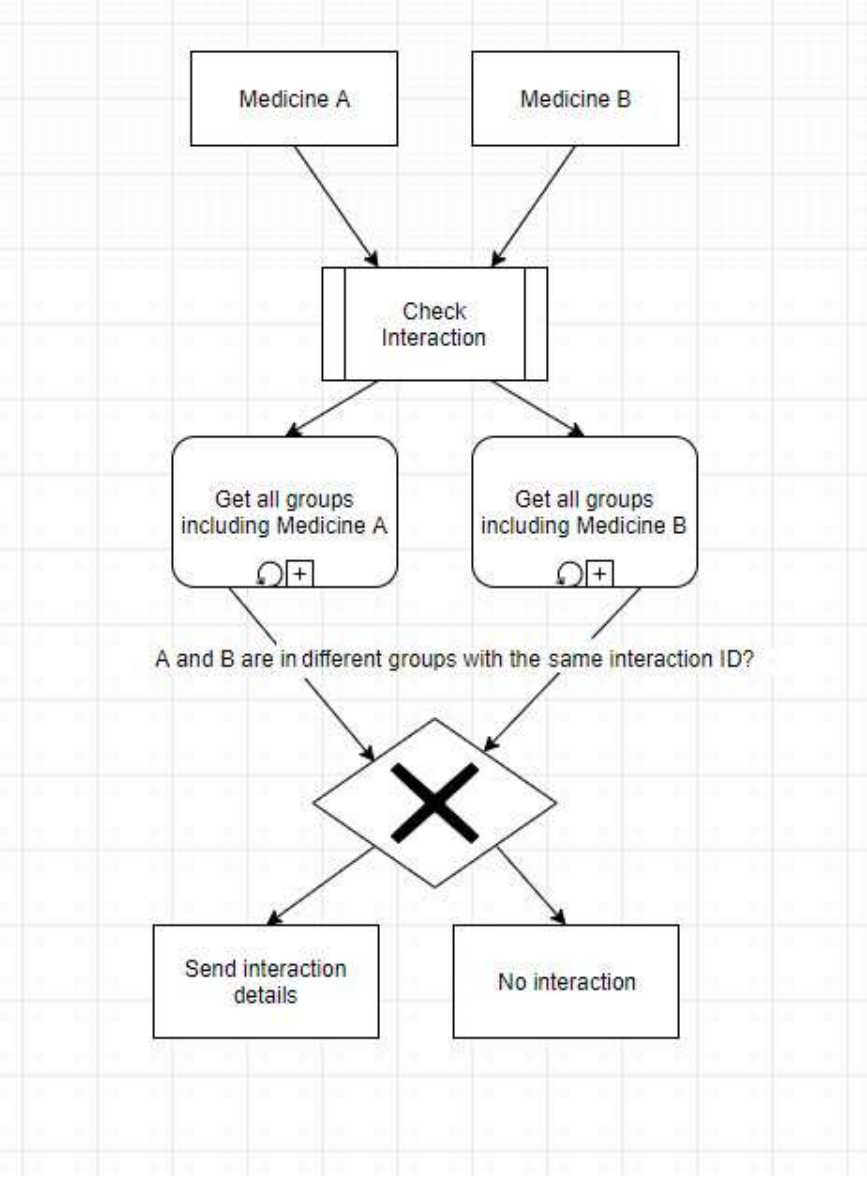


Figure 3.21: FEST interaction mechanism flowchart

3.4 Define HL7 FHIR structures for FEST

In this section, we define FHIR profiles for “Medication”, “Substance”, and “Group” resources.
The main point here is to use Forge tool which is the official FHIR tool to create and validate profiles. By validating our profiles this way, we guarantee that we did not violate FHIR specification so that resource instances of our profiles can be understood, and exchanged with other health information. The definition of

each structure we defined in the profiles is accessible through the url value attached to each structure. For example, the URL of our FEST medication resource definition is :
<url value="http://ehealth.uia.no/StructureDefinition/FESTMedication" />

The following diagram illustrates how we created an implementation guide for medication, group and substance profiles.

```
<ImplementationGuide>
    <meta />
    <url value="http://ehealth.uia.no/StructureDefinition/FestImplementationGuide" />
    <version value="0.1" />
    <name value="FestImplementationGuide" />
    <status value="draft" />
    <date value="2018-05-20T17:47:06.513445+02:00" />
    <description value="FEST implementationguide for FHIR" />
    <fhirVersion value="3.0.1" />
    <package>
    <name value="FEST on FHIR"/>
    <description value="Package for profiles related to FEST" />
        <resource>
            <example value="true" />
            <name value="FESTMedication" />
            <description value="Profile for Medication" />
            <sourceReference>
            <reference value="http://ehealth.uia.no/StructureDefinition/FESTMedication" />
            </sourceReference>
        </resource>
        <resource>
            <example value="true" />
            <name value="FESTGroup" />
            <description value="Profile for Group" />
            <sourceReference>
            <reference value="http://ehealth.uia.no/StructureDefinition/FESTGroup" />
            </sourceReference>
        </resource>
        <resource>
            <example value="true" />
            <name value="FESTSubstance" />
            <description value="Profile for Substance"/>
            <sourceReference>
            <reference value="http://ehealth.uia.no/StructureDefinition/FESTSubstance" />
            </sourceReference>
        </resource>
    </package>
</ImplementationGuide>
```

Figure 3.22: FEST on FHIR implementation guide

3.4.1 FEST Medication Profile

3.4.1.1 Functional requirements

Id	2	**Requirement type**	Functional requirement 2	**Event**	Create FEST Medication Profile
Description	Create FEST Medication Profile according to HL7 FHIR v3 specification				

Rationale	To map FEST medication elements to FHIR resource elements
Source	Islam Al Khaldi
Dependencies	Functional requirement 4, this profile has reference to substance profile
Conflict	None

Table 2: Create FEST medication profile

3.4.1.2 Mapping specification

The following figure illustrates how FEST LegemidddelMerkevare elements mapped into FHIR Medication profile elements:

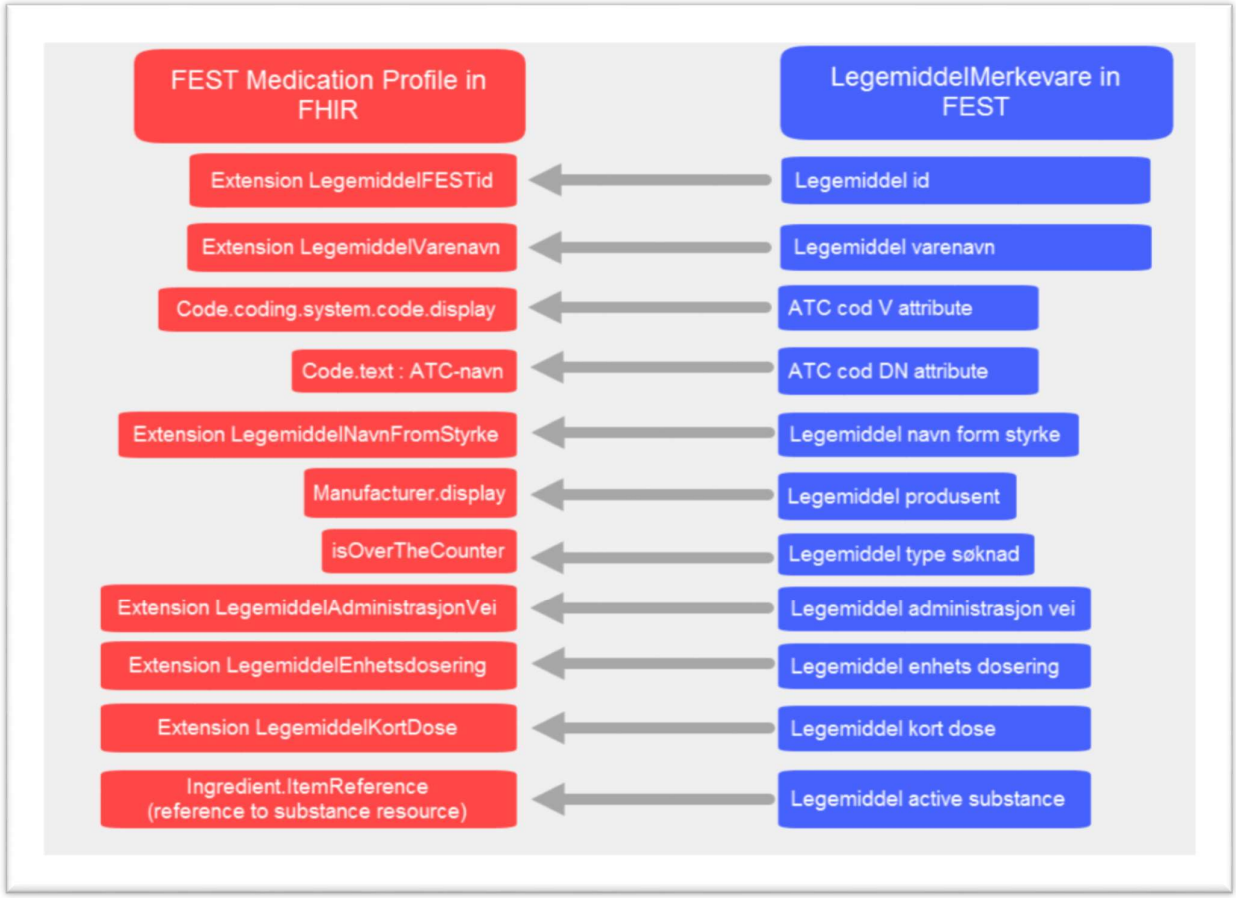


Figure 3.23: Mapping elements of LegemiddelMerkevare to FHIR medication resource

The structure definitions of the FEST medication profile and its extensions are in XML format and can be found in thesis appendix. HL7 FHIR validation rules are applied on each element we create and define in FEST medication profile so that each profile's element is considered complying to HL7 FHIR specifications. The following screenshot illustrates how we use Forge FHIR Profile Designer to define the coding system, ATC, in our FEST medication profile and set the cardinality value for the element system.

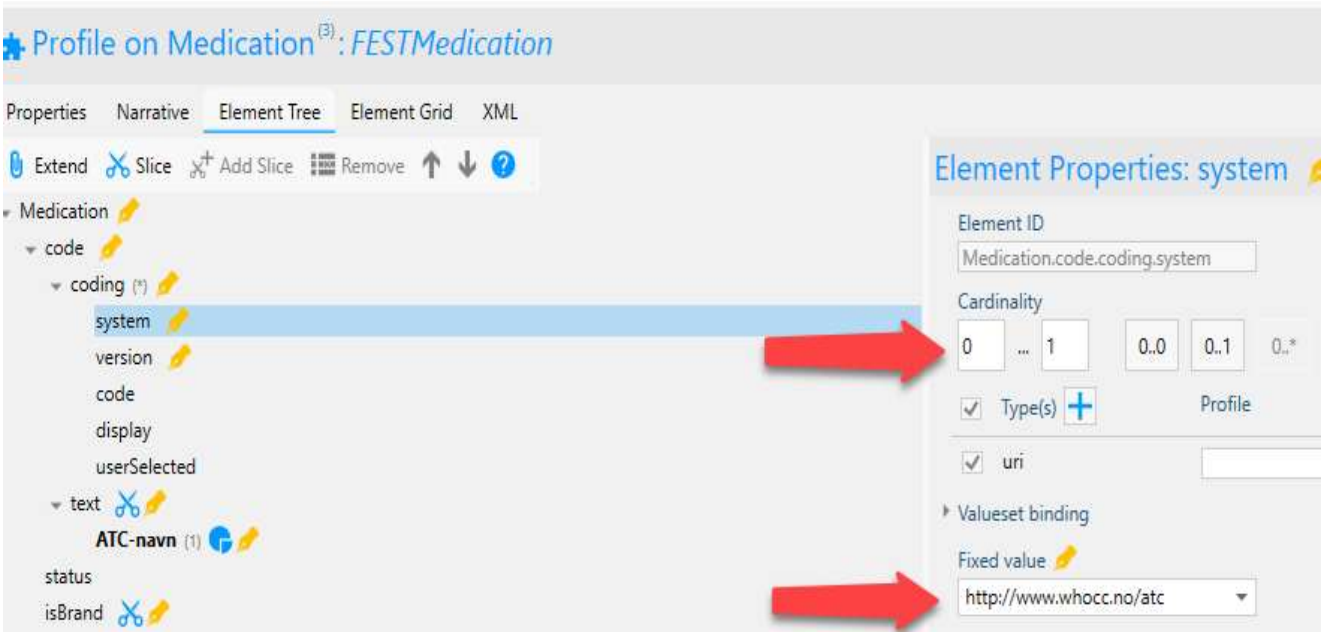


Figure 3.24: Set system element properties for FEST Medication profile

3.4.2 Group Profile

3.4.2.1 Functional requirements

Id	3	**Requirement type**	Functional requirement 3	**Event**	Create FEST Group Profile
Description	Create FEST Group Profile according to HL7 FHIR v3 specification				
Rationale	To map FEST interaction elements to FHIR resource elements				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 2				
Conflict	None				

Table 3: Create FEST group profile

3.4.2.2 Mapping specification

Based on section 3.2.2.1 FEST Interaction mechanism, we are going to represent FEST interaction information as a FHIR Group resource where the members of the group are medication resources which share the same interaction information. The reason behind our selection of FHIR Group resource is discussed in the discussion section of this thesis. The following figure illustrates how FEST Interaksjon elements mapped into FHIR Group profile elements:

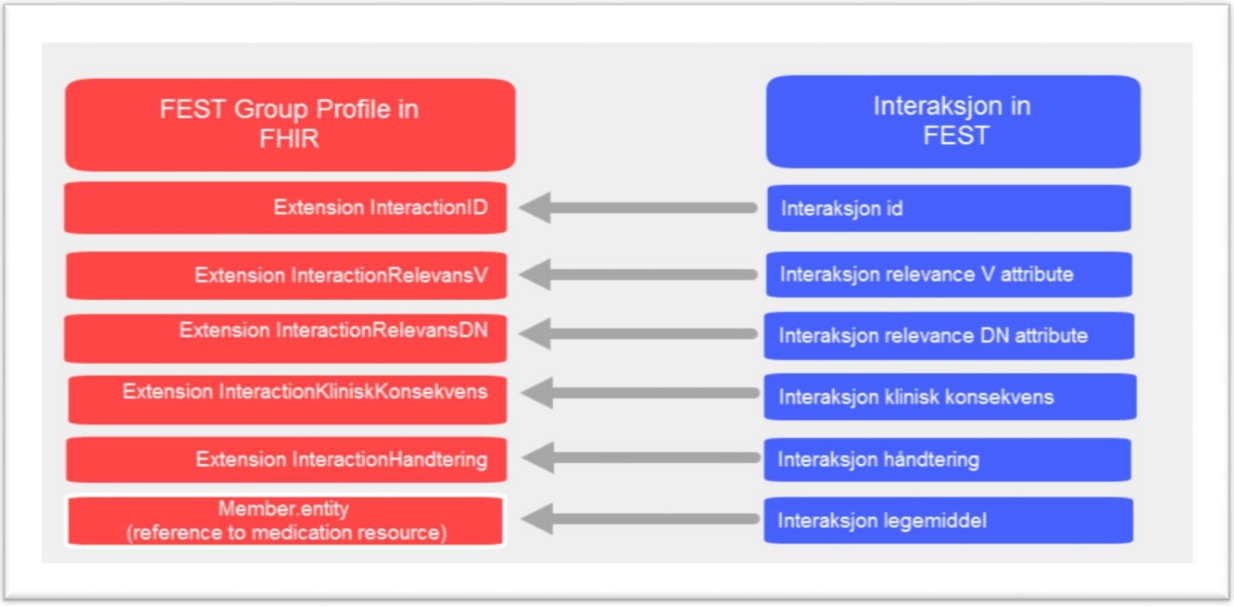


Figure 3.25: Mapping elements of Interaksjon to FHIR group resource

The structure definitions of the FEST group profile and its extensions are in XML format and can be found in thesis appendix. HL7 FHIR validation rules are applied on each element we create and define in FEST group profile so that each profile's element is considered complying to HL7 FHIR specifications. The following screenshot illustrates how we use Forge FHIR Profile Designer to set the entity type of the FEST group profile to be FESTMedication which is based on FHIR medication resource. If we specify an entity type that is not valid by HL7 FHIR specification, we get a validation error message.

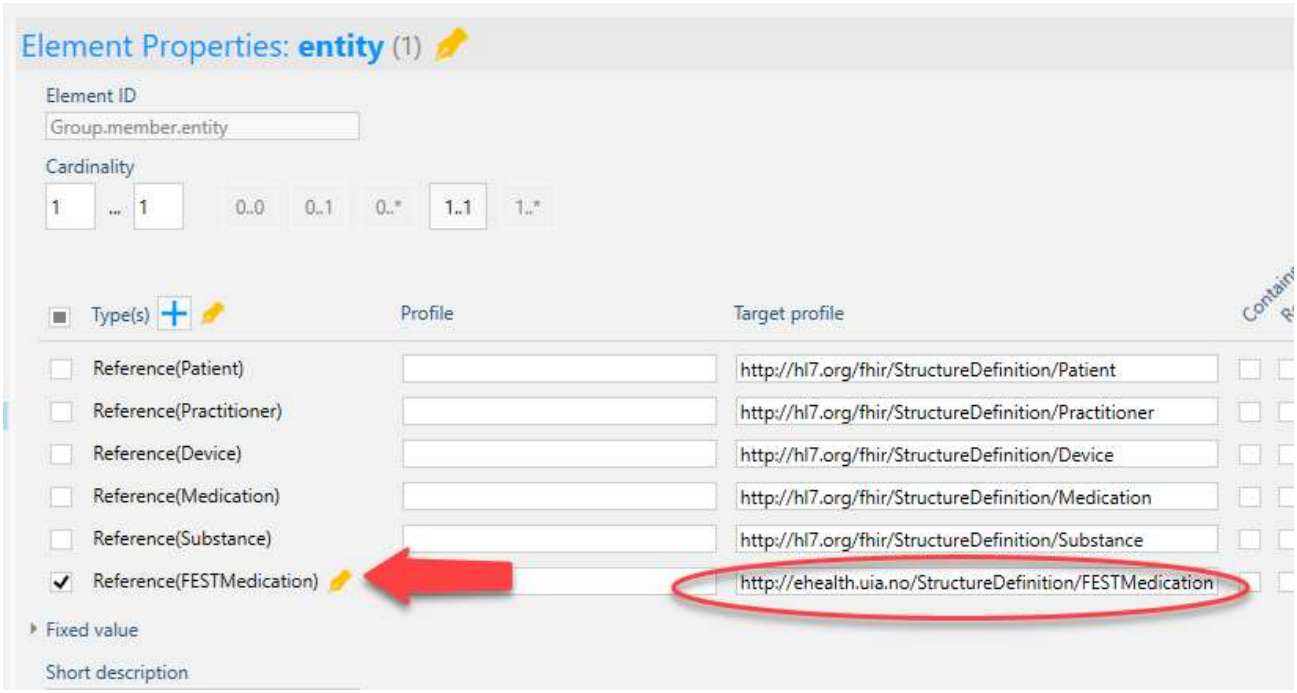


Figure 3.26: Set entity properties of FESTGroup profile.

Forge tool has validation error feature which checks that the values we assign to the element properties of the profile are valid and conform to HL7 FHIR specification. For example, HL7 FHIR specification restricts the type of members of the group profile to be one of the following resources; person, animal, practitioner, device, medication, or substance. The following screenshot illustrates an example of such validation error feature if we set the member type to be observation resource.

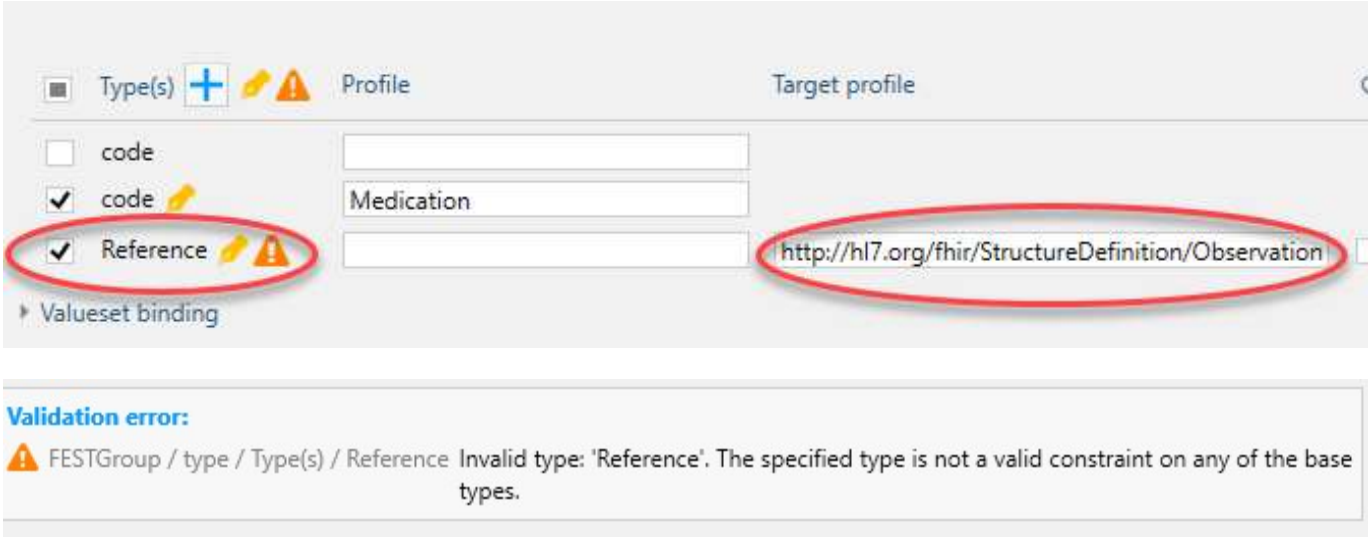


Figure 3.27: Validation error

3.4.3 Substance Profile

3.4.3.1 Functional requirements

Id	4	**Requirement type**	Functional requirement 4	**Event**	Create FEST Substance Profile
Description	Create FEST Substance Profile according to HL7 FHIR v3 specification				
Rationale	To map FEST virkestoff elements to FHIR resource elements				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 2				
Conflict	None				

Table 4: Create FEST substance profile

3.4.3.2 Mapping specification

The structure definitions of the FEST substance profile and its extensions are in xml format and can be found in thesis appendix. The following figure illustrates how FEST *Virkestoff* elements mapped into FHIR Substance profile elements:

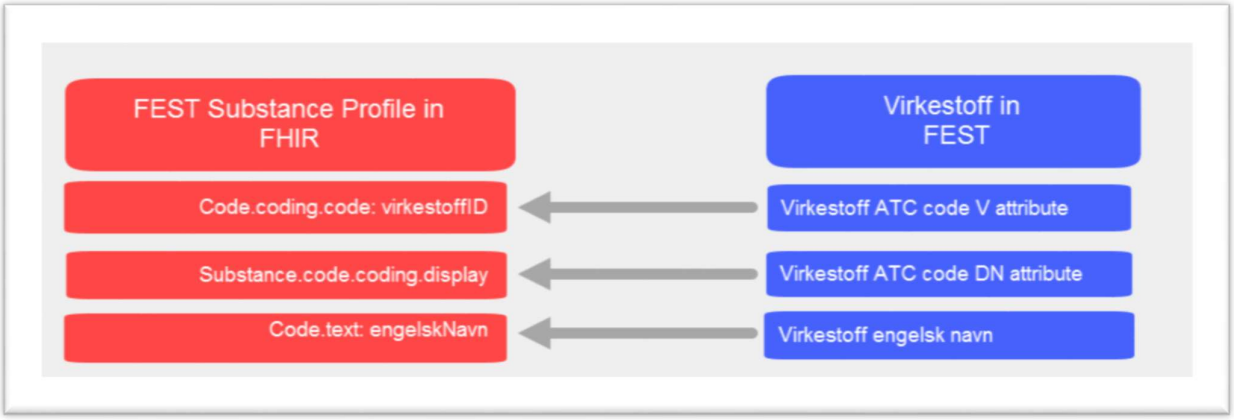


Figure 3.28:Mapping elements of Virkestoff to FHIR substance resource

3.4.4 FEST on FHIR Simplifier Project

We have created a project on www.simplifier.net to host our FEST on FHIR profiles and extensions. The following figures illustrates the structure of each profile in our FEST on FHIR project.

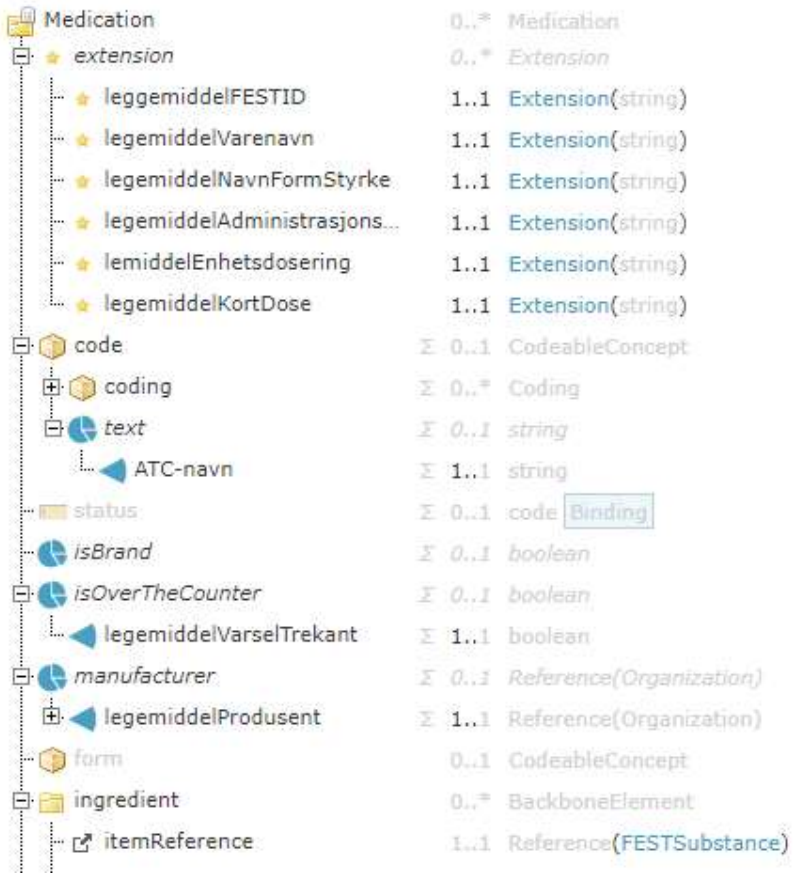


Figure 3.29: FEST Medication Profile

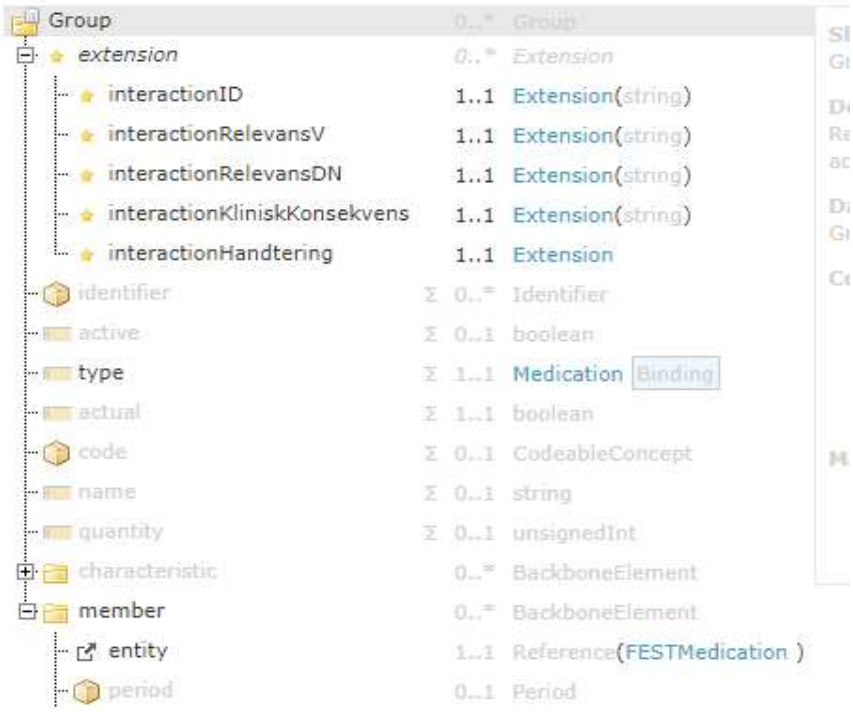


Figure 3.30: FEST Group Profile

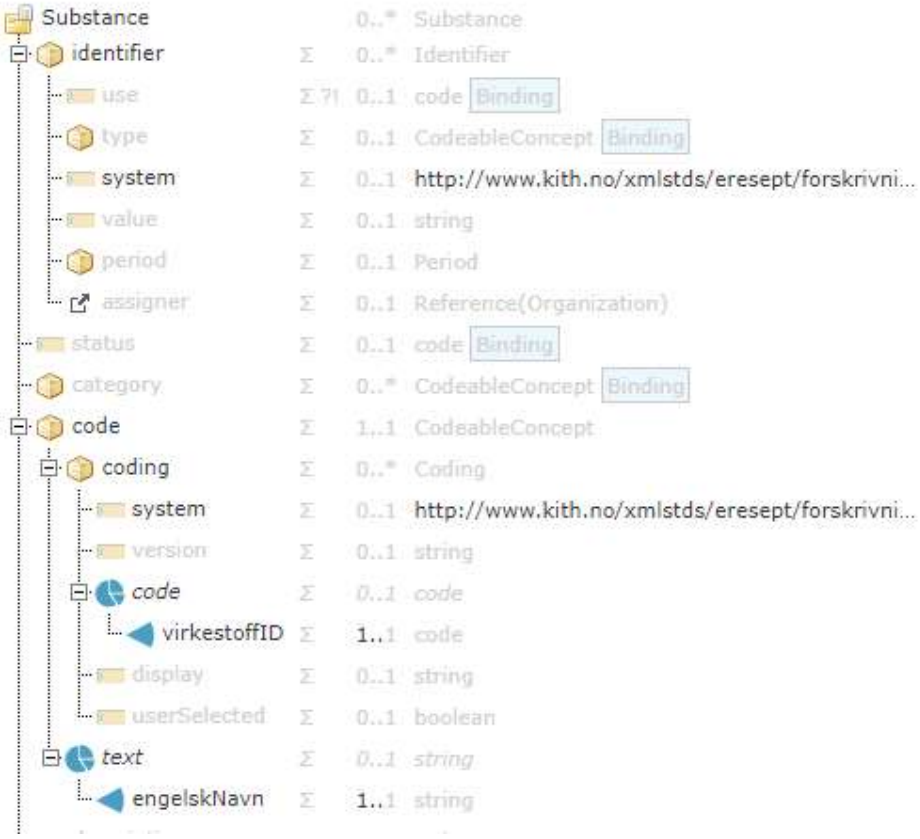


Figure 3.31: FEST Substance Profile

3.4.5 Non-functional Functional requirements

- Install Forge FHIR Profile Designer tool.
- Sign up in www.simplifier.net

3.5 Set up FHIR Server (HAPI-FHIR)

In this thesis, we use HAPI-FHIR as our FHIR server end point. The following functional requirements are implemented in sequence to set up our environment for FHIR RESTful operations so that we can create instances of medication and group resources and save them on Apache Derby database hosted on FHRI server for persistency. This step is very vital for our project and must be implemented properly as FHIR server will hold the whole FEST data structures we mapped according to HL7 FHIR specification.

3.5.1 Functional requirements

Id	5	**Requirement type**	Functional requirement 5	**Event**	Setup Apache Tomcat v9
Description	Install and configure Apache Tomcat v9 as our web application server (Servlet container)				
Rationale	To host FHIR server (hapi-fhir)				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 5: Setup Apache Tomcat v9

Id	6	**Requirement type**	Functional requirement 6	**Event**	Build hapi-fhir server
Description	Download hapi-fhir-jpaserver-example server from github repository of hapi-fhir project and build it using Apache Maven command “mvn install”.				
Rationale	To get the WAR file of hapi-fhir-jpaserver-example which will be deployed on Apache Tomcat web server.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 5				
Conflict	None				

Table 6: Build HAPI-FHIR server

Id	7	**Requirement type**	Functional requirement 7	**Event**	Deploy hapi-fhir server
Description	Deploy HPAI FHIR server on Apache Tomcat web server by uploading the WAR file through the management console of Tomcat.				
Rationale	To access our FHRI server via web by using the URL http://localhost:8081/fest				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 6				
Conflict	The default port of Tomcat is 8080, so we changed it in the configuration file to 8081 to avoid conflict with jetty web server we used during the development of web applications of practitioner and update FEST use cases.				

Table 7: Deploy HAPI-FHIR server

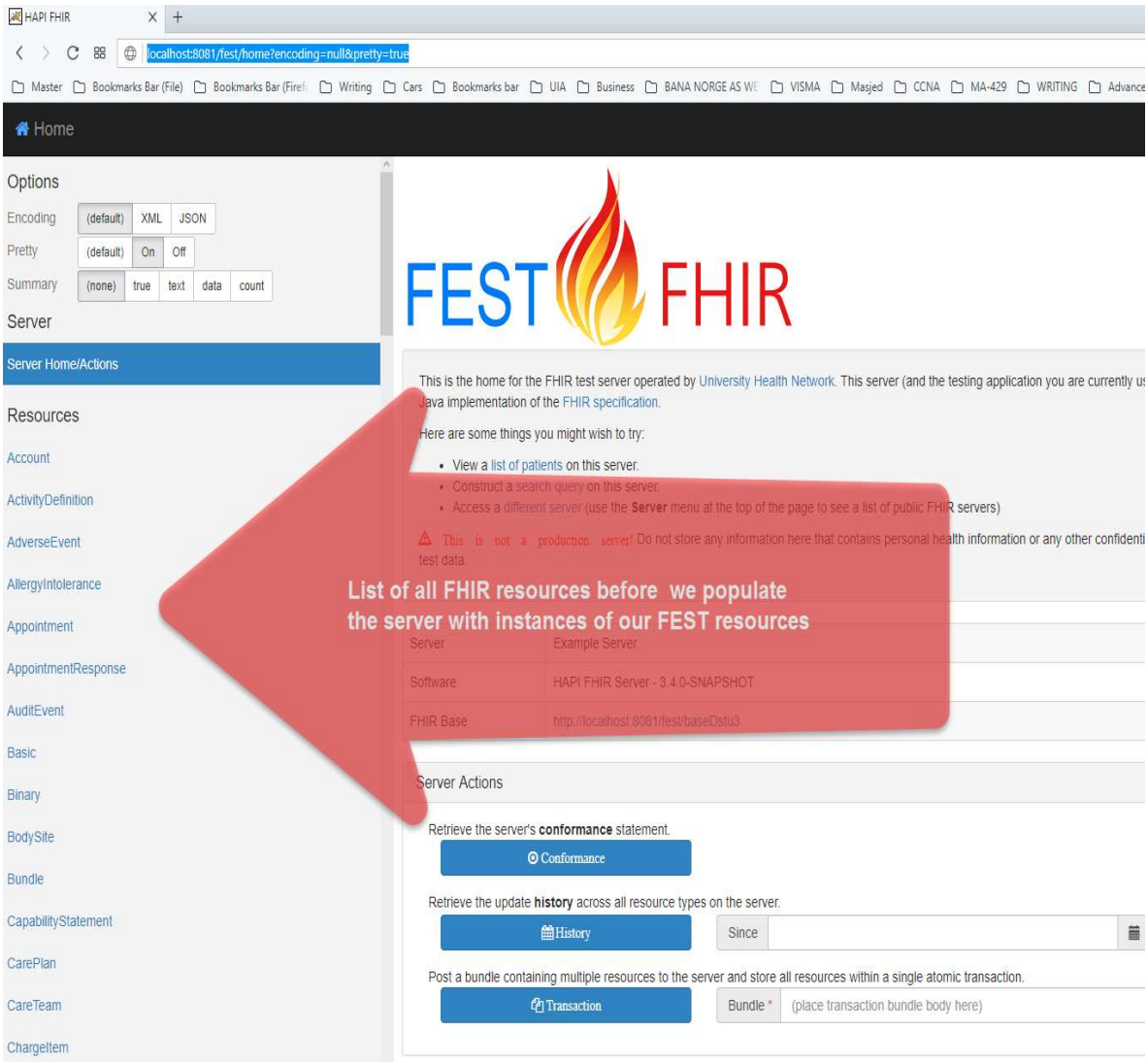


Figure 3.32: Home page of FEST on FHIR HAPI-FHIR server

3.5.2 Non-functional requirements

Java jdk 1.8 must be installed, and the path of the *bin* folder of Apache Maven must be added to the PATH environment variable. We have to select Java home directory when installing Apache Tomcat.

3.6 System Design

In this section, we introduce our system architecture to map FEST on FHIR. In addition, we introduce our Web application architecture for the practitioner use case.

3.6.1 FEST on FHIR system architecture

The following diagram illustrates the system architecture. From the right side, we start parsing and mapping FEST database. As a result, on the left side, parsed FEST data elements are stored as FHIR resources instances in HAPI-FHIR database (Apache Derby).

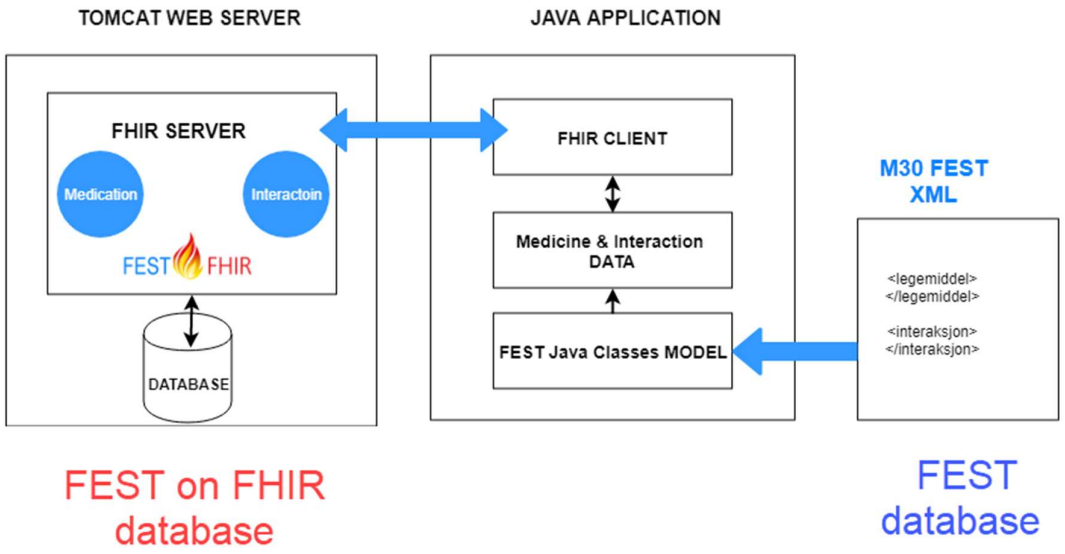


Figure 3.33: FEST on FHIR system architecture

The following diagram illustrates the execution workflow:

Step#	Description	Tool/Software
1	Parsing M3 FEST xml and schema files and create FEST Java Classes Model	Java/Eclipse
2	Develop Java classes to hold data of selected FEST catalogues	Java/Eclipse
3	Develop Java classes for FEST medication, interaction and substance profiles. Develop Java classes for adding search parameters to FHIR server for extensions. Implement FEST interaction mechanism.	Java/Eclipse
4	Create instances of FEST profiles and upload them on FHIR server. Read instances of FEST profiles from FHIR server.	Java/Eclipse

Table 8: System execution workflow

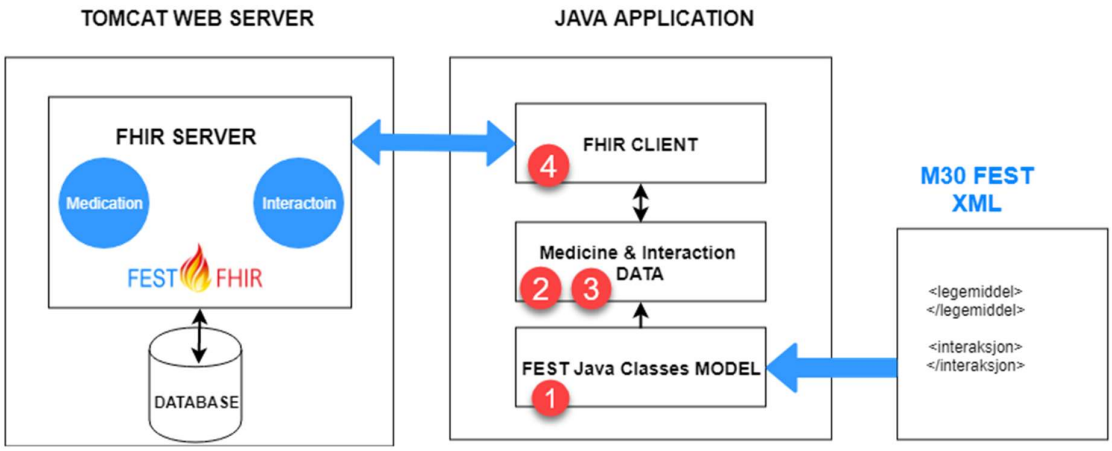


Figure 3.34: System execution workflow

3.6.2 Practitioner Web Application Architecture

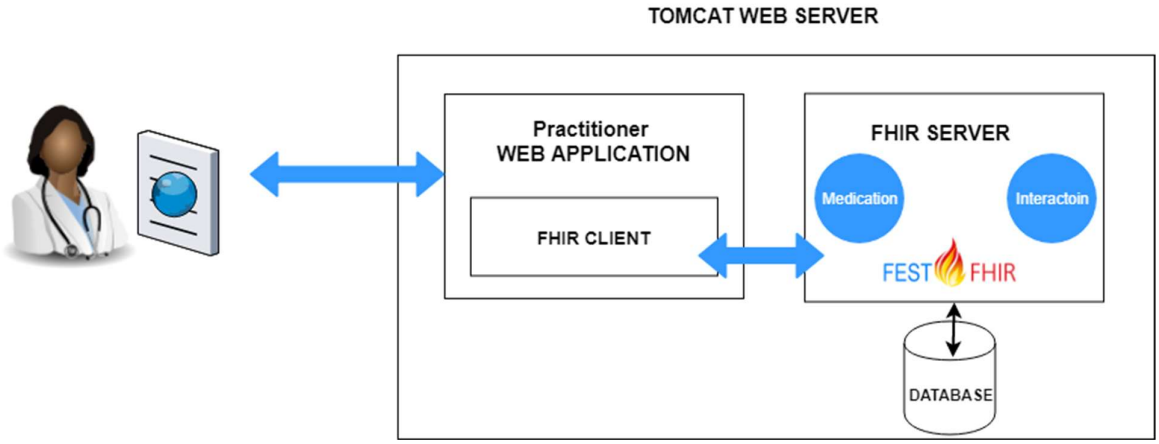


Figure 3.35: Practitioner web application architecture

3.7 System Implementation

At this very stage, we have a FEST object which is an instance of FEST class in our FEST Java model. Our task here in this section is to extract FEST data from FEST object and initiate objects of classes that can hold FEST data for later use when we start creating instances of medication and interaction resources to be saved on FHIR server. In our implementation, we apply the concept of modularity where we separate the code that handles *LegemiddelMerkevare* catalogue from the code that handles *Interaksjon* catalogue and drug-drug interaction detection as well.

As the main challenge at this part was the lack of documentation, I tried my best to explain each step in detail. As a result, this system implementation section can be used as user manual when it comes to implement FEST profiles on HAPI-FHIR server.

3.7.1 Develop Java classes for FEST LegemiddelMerkevare Catalogue

First, we create *LegemiddelMerkevareCatalogueMapper* class which takes FEST object as an argument.

Id	8	**Requirement type**	Functional requirement 8	**Event**	Develop class LgemiddelMerkevareCatalogueMapper
Description	This class takes FEST object as an argument so that all catalogue data are available for mapping operations.				
Rationale	To loop through FEST catalogue data elements and store them.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1, 9,10				
Conflict	None				

Table 9: Class LegemiddelMerkevareCatalogueMapper

Figure 3.36: LegemiddelMerkevareCatalogueMapper class

```
// ********************************************************************************
// LegemiddelMerkevare Catalogue
// ********************************************************************************

LegemiddelMerkevareCatalogueMapper lm = new LegemiddelMerkevareCatalogueMapper(fest);
medicationFHIRInformationList = lm.getMedicationFHIRInformationList();
```

Figure 3.37: Start mapping FEST LegemiddelMerkevare catalogue

The main goal behind this class is to loop through *LegemiddelMerkevare* catalogue and store each *LegemiddelMerkevare* FEST data element in class MedicationFHIRInformation that inherits *LegemiddelMerkevare* class. But why we created this class MedicationFHIRInformation while we have already class *LegemiddelMerkevare*? The answer is simply that we apply here the concept of separation of concerns. In other words, if the data structure of FEST changes, our FEST Java model will automatically represent the new changes because we have our own class MedicationFHIRInformation in which we can apply any modification in order to fit the new FEST data structures.

Let's first introduce the classes we created which are used in LegemiddelMiddelCatalogueMapper class.

3.7.1.1 MedicationFHIRInformation Class

Id	9	Requirement type	Functional requirement 9	Event	Develop class MedicationFHIRInformation
Description	It extends LegemiddelMerkevare class				
Rationale	Represent the selected FEST data elements from the main four catalogues to map on FHIR resources.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1,8,10				
Conflict	None				

Table 10: Class MedicationFHIRInformation

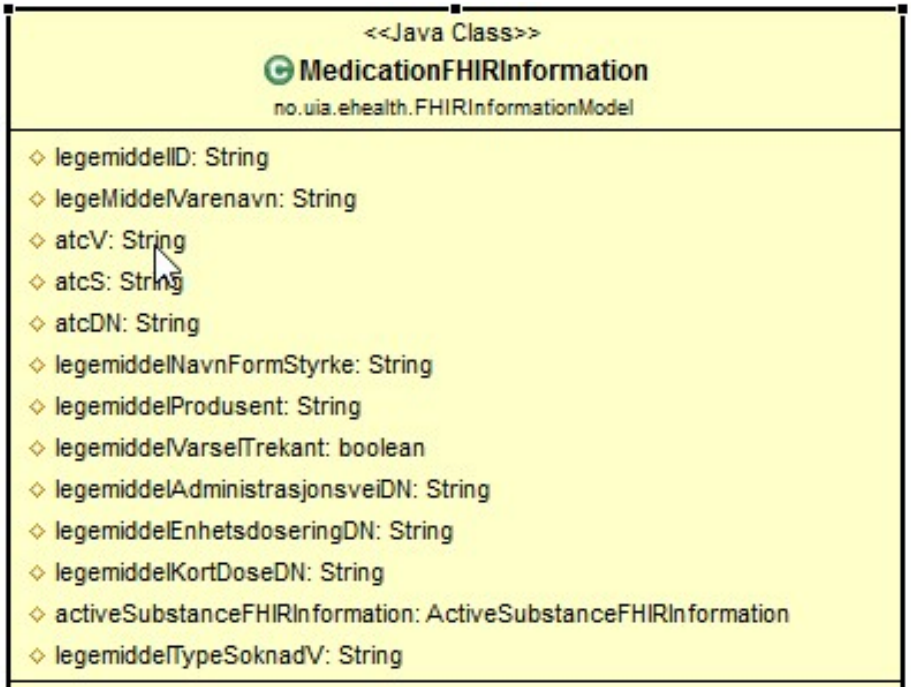


Figure 3.38: MedicationFHIRInfromation

```
1. package no.uia.ehealth.FHIRInformationModel;
2. import no.uia.ehealth.workflow.*;
3. import no.uia.ehealth.workflow.KatLegemiddelMerkevare.OppfLegemiddelMerkevare;
4. import java.util.ArrayList;
5. import java.util.List;
6. public class MedicationFHIRInformation extends LegemiddelMerkevare {
7. protected String legemiddelID;
8. protected String legeMiddelVarenavn;
9. protected String atcV;
10. protected String atcS;
11. protected String atcDN;
12. protected String legemiddelNavnFormStyrke;
13. protected String legemiddelProdusent;
14. protected boolean legemiddelVarselTrekant;
15. protected String legemiddelAdministrasjonsveiDN;
16. protected String legemiddelEnhetsdoseringDN;
17. protected String legemiddelKortDoseDN;
18. protected ActiveSubstanceFHIRInformation activeSubstanceFHIRInformation;
19. protected String legemiddelTypeSoknadV;
```

The above snippet illustrates the properties of LegemiddelMerkevare that will be mapped to elements of FHIR medication resource. Various information about medicines are gathered through the methods available by LegemiddelMerkevare and its ancestor TypeLegemiddel class. For example, to get product information such as producer, we call getProduktInfo() method which in turn will access another catalogue and return the required information. The same procedure applies for the other information about dosage and administration of the medicine brand product.The description of each property will be further explained later when we discuss the implementation of FESTMedication resource in this chapter.

3.7.1.2 ActiveSubstanceFHIRInformation class

We have also created ActiveSubstanceFHIRInformation class which extends Substansgruppe.Substans class to hold information about the active substance (virkestoff) used in the medicine. The class holds information about the name, ATC-code, and the English name of the active substance.

Id	10	**Requirement type**	Functional requirement 10	**Event**	Develop class ActiveSubstanceFHIRInformation
Description	It extends Substansgruppe.Substans				
Rationale	Represent the properties of active substance of the medicine				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1,8, 9				
Conflict	None				

Table 11: Class ActiveSubstanceFHIRInformation

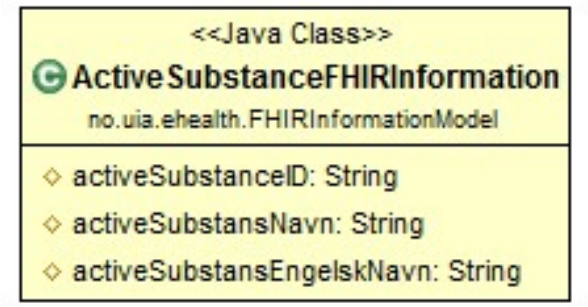


Figure 3.39: ActiveSubstanceFHIRInformation class

The following diagram illustrates the relations between the three classes we created to hold information about LegemiddelMerkevare catalogue.

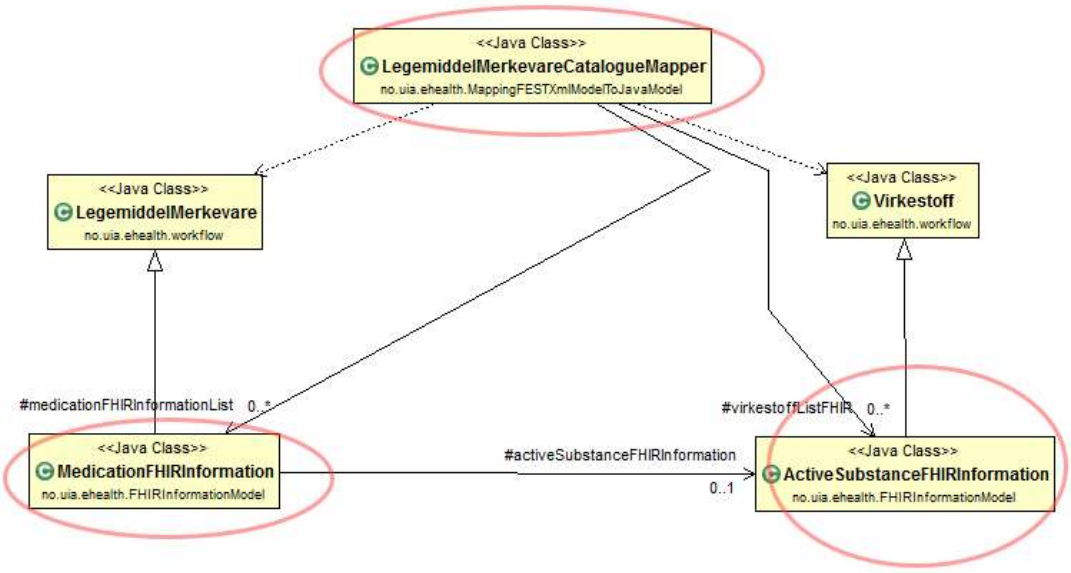


Figure 3.40: Classes for mapping LegemiddelMerkevare catalogue

3.7.1.3 Logic inside LegemiddelMerkevareCatalogueMapper class

As we introduced the above classes, we come back to explain the logic of Legemiddel Merkevare Catalogue mapper class LgemiddelMerkevareCatalogueMapper.

```
13 public class LegemiddelMerkevareCatalogueMapper {
14 protected List<ActiveSubstanceFHIRInformation> virkestoffListFHIR = new ArrayList<ActiveSubstanceFHIRInf
15 protected List<MedicationFHIRInformation> medicationFHIRInformationList = new ArrayList<MedicationFHIRIn
16 public LegemiddelMerkevareCatalogueMapper(FEST fest) {
17     List<OppfLegemiddelMerkevare> oppfLegemiddelMerkevareList = new ArrayList<OppfLegemiddelMerkevare>()
18     oppfLegemiddelMerkevareList = fest.getKatLegemiddelMerkevare().getOppfLegemiddelMerkevare();
19     for (OppfLegemiddelMerkevare element : oppfLegemiddelMerkevareList) {
20         LegemiddelMerkevare medicine = element.getLegemiddelMerkevare();
21         MedicationFHIRInformation medicationFHIR = new MedicationFHIRInformation();
```

In lines 14, 15 we define two lists to hold the extracted information from FEST object. The FEST object is passed to the constructor of the class (line 16). In line 17, we create a list to extract OppfLegemiddelMerkevare objects (each object contains one LegemiddelMerkevare instance) under LegemiddelMerkevare catalogue. In line 19, we start to loop through the list and extract each LegemiddelMerkevare property and create an instance of MedicationFHIRInformation class and store the extracted data in their corresponding properties.

```
81
82   medicationFHIRInformationList.add(medicationFHIR);
83
```

Before the end of for loop (line 82), we add the instance of MedicationFHIRInformation to the list medicationFHIRInformationList.

As shown in the snippet below, by calling the method getMedicationFHIRInformationList() of LgemiddelMerkevareCatalogueMapper, we get the list in the main method of our Java application, and we are ready to implement FHRI profiles we defined previously in the Forge tool.

```
// ********************************************************************************
// LegemiddelMerkevare Catalogue
// ********************************************************************************

LegemiddelMerkevareCatalogueMapper lm = new LegemiddelMerkevareCatalogueMapper(fest);
medicationFHIRInformationList = lm.getMedicationFHIRInformationList();
```

3.7.1.4 Non-functional Functional requirements

All over our code and starting from the FEST object, we have to make sure that the value of the object we need to access is not null. Otherwise we get "Null Pointer Exception", and the following part of the code will not work properly.

3.7.2 Develop Java classes for FEST interaction catalogue.

First, we create InteraksjonCatalogueMapper class which takes fest object as an argument.

Id	11	**Requirement type**	Functional requirement 11	**Event**	Develop class InteraksjonCatalogueMapper
Description	This class takes fest object as an argument so that all catalogue data are available for mapping operations.				
Rationale	To loop through FEST catalogue data elements and store them.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1, 12,13				
Conflict	None				

Table 12: Class InteraksjonCatalogueMapper

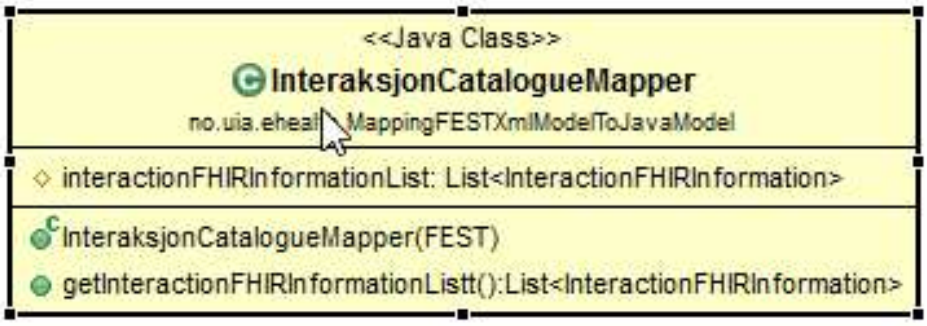


Figure 3.41: InteraksjonCatalogueMapper class

```
// ********************************************************************************
// Interaksjoner Catalogue
// ********************************************************************************

InteraksjonCatalogueMapper im = new InteraksjonCatalogueMapper(fest);
interactionFHIRInformationList = im.getInteractionFHIRInformationListt(); // End of Get FEST catalogues information
```

Figure 3.42: Start mapping FEST Interaksjoner catalogue

The main goal behind this class is to loop through Interakjsoner catalogue and store each Interaksjon FEST data element in class InteractionFHIRInformation that inherits Interaksjon class. We applied here the concept of separation of concerns as well. Consequently, if the data structure of FEST changes, our system will not be broken because we have our own class InteractionFHIRInformation in which we can apply any

modification in order to fit the new FEST data structures. Let's first introduce the classes we created which are used in InteraskjonCatalogueMapper class.

3.7.2.1 InteractionFHIRInformation class

Id	12	**Requirement type**	Functional requirement 12	**Event**	Develop class InteractionFHIRInformation
Description	It extends Interaksjon class				
Rationale	Represent the selected FEST data elements from the interaksjon catalogue to be mapped on FHIR group resource.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1, 11,13				
Conflict	None				

Table 13: Class InteractionFHIRInformation

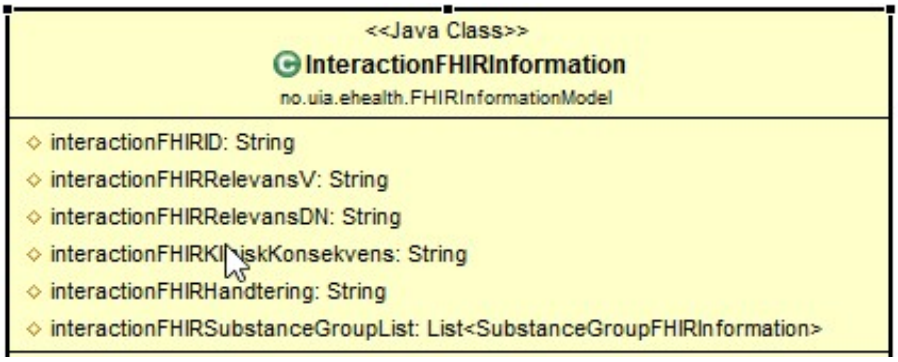


Figure 3.43: InteractionFHIRInfromatin class

The above figure illustrates the properties of Interaksjon that will be mapped to elements of FHIR group resource. Various information about interactions are gathered through the methods available by Interaksjon class.The description of each property will be further explained later when we discuss the implementation of FESTGrooup resource in this chapter.

At this very stage, we must explain the structure of an interaction entity in FEST so that the reason of creating the next class makes sense for the reader of this thesis. The following diagram illustrates this structure.

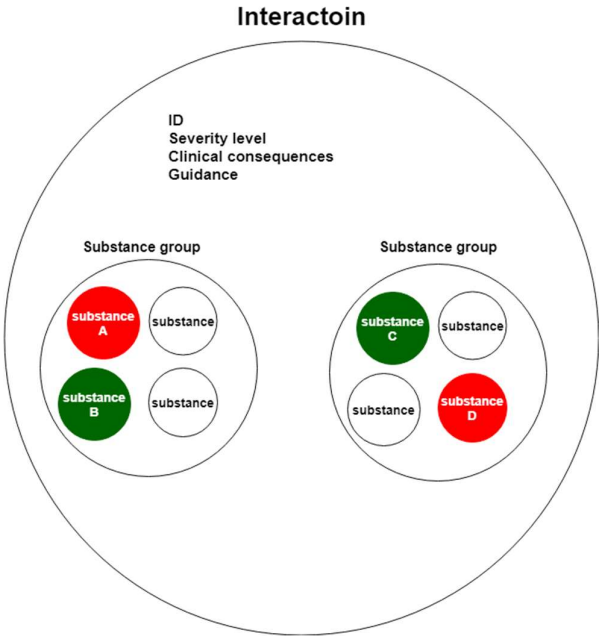


Figure 3.44: FEST interaction structure

Each interaction in FEST includes two substance groups. To say that an interaction is found between two medicines/substances, the active substance of both medicines must exist in two different substance groups at the same interaction entity. For example, substance A and D interact together because they belong to different substance groups and the same for substance A and C, but substance A and B has no interaction because they belong to the same group and the same for substance C and D.

3.7.2.2 SubstanceGroupFHIRInformation class

Id	13	**Requirement type**	Functional requirement 13	**Event**	Develop class SubstanceGroupFHIRInformation
Description	It extends Substansgruppe class				
Rationale	Represent the list of SubstanceFHIRInformation class				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 1,11,12				
Conflict	None				

Table 14: Class SubstanceGroupFHIRInformation

Figure 3.45: SubstanceGroupFHIRInformation class

The following diagram illustrates the relations between the three classes we created to hold information about Interaksjon catalogue.

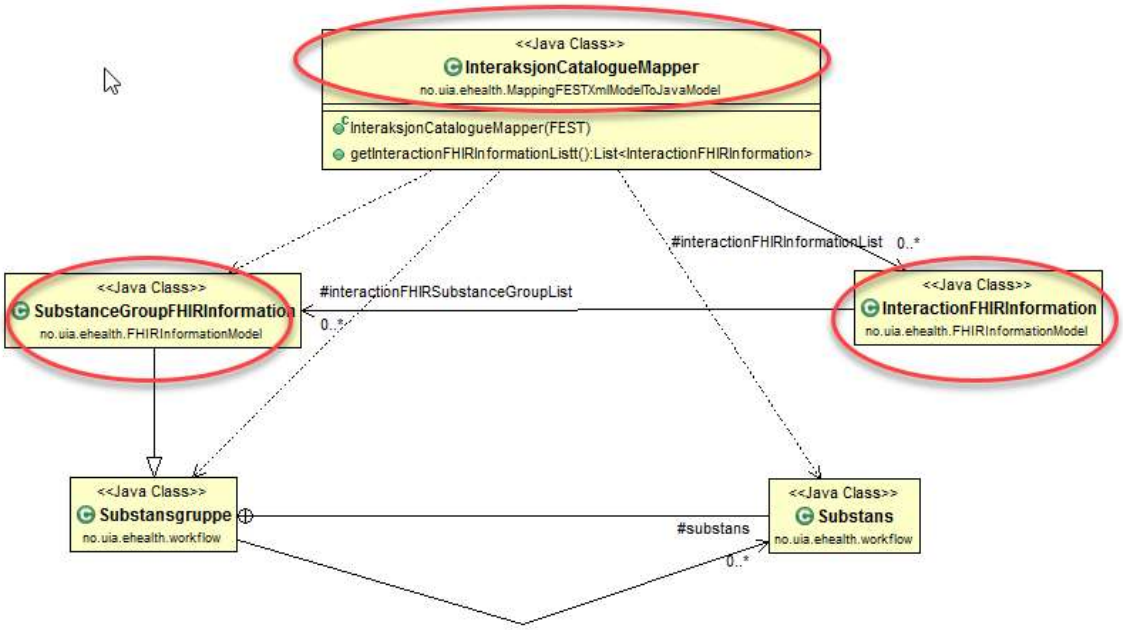


Figure 3.46: Classes for mapping Interaksjon catalogue

3.7.2.3 Logic in InteraksjonCatalogueMapper class

As we introduced the above classes, we come back to explain the logic of *Interaksjon* catalogue mapper class InteraksjonCatalogueMapper.

```
12 public class InteraksjonCatalogueMapper
13 {
14     protected List<InteractionFHIRInformation> interactionFHIRInformationList = new ArrayList<Inte
15
16     public InteraksjonCatalogueMapper (FEST fest)
17     {
18         List<OppfInteraksjon> oppfInteraksjonList = new ArrayList<OppfInteraksjon>();
19         oppfInteraksjonList = fest.getKatInteraksjon().getOppfInteraksjon();
20         for ( OppfInteraksjon element : oppfInteraksjonList)
21         {
22             if(element != null)
23             {
```

In lines 14 we define a list to hold the extracted information from fest object. The fest object is passed to the constructor of the class (line 16). In line 18, we create a list to extract *OppfInteraksjon* objects (each object contains one *Interaksjon* instance) under *Interaksjon* catalogue. In line 20, we start to iterate over the list and extract each Interaksjon property and create an instance of InteractionFHIRInformation class and store the extracted data in corresponding properties respectively.

```
47 // gathering required FHIR information for all substance groups in each interaction
48 List<SubstanceGroupFHIRInformation> substanceGroupFHIRInformationList = new ArrayList<SubstanceGr
```

In line 48, we create a list of type SubstanceGroupFHIRInfromation to extract all substance group instances from the current interaction instance.

```
54 interaksjonSubstansgruppeList = interaction.getSubstansgruppe();
55 for (Substansgruppe sg : interaksjonSubstansgruppeList )
56     {
```

In line 54, we iterate over each group list of the current interaction instance. In line 80, we add each substance to its current substance list.

```
84                  substanceGroupFHIRInformation.setSubstanceFHIRInformation(substanceFHIRInformat
85                  substanceGroupFHIRInformationList.add(substanceGroupFHIRInformation); // add gr
86              }
87
88          interactionFHIR.setSubstanceGroupFHIRInformation(substanceGroupFHIRInformationList); //
89      }
90      interactionFHIRInformationList.add(interactionFHIR); // add the current interaction to list
91  }
```

In line 84, we add the current substance list to its group. In line 85, we add the group to its list of groups. In line 88, we add the groups list to the current interactionFHIRInformation instance. Finally, at line 90, we add each interactionFHIRInformation instance to the interactionFHIRInformationList.

As shown in the snippet below, by calling the method getInteractionFHIRInformationList() of InteraskjonCatalogueMapper, we get the list in the main method of our Java application, and we are ready to implement FHRI profiles we defined previously in the forge tool.

```
// ********************************************************************************************
// Interaksjoner Catalogue
// ********************************************************************************************

InteraksjonCatalogueMapper im = new InteraksjonCatalogueMapper(fest);
interactionFHIRInformationList = im.getInteractionFHIRInformationListt(); // End of Get FEST catalogues information
```

3.7.2.4 Non-functional Functional requirements

All over our code and starting from the fest object, we have to make sure that the value of the object we need to access is not null. Otherwise we get "Null Pointer Exception", and the following part of the code will not work properly.

3.7.3 Implement FEST medication profile.

At this very stage, we implement our FESTMedication profile we properly defined in FEST Medication Profile section of this chapter. Starting again from class AutomaticMappingFESTOnFHIR, we call the constructor of FESTMedicatonHapiFHIRController, as shown in the code snippet below.

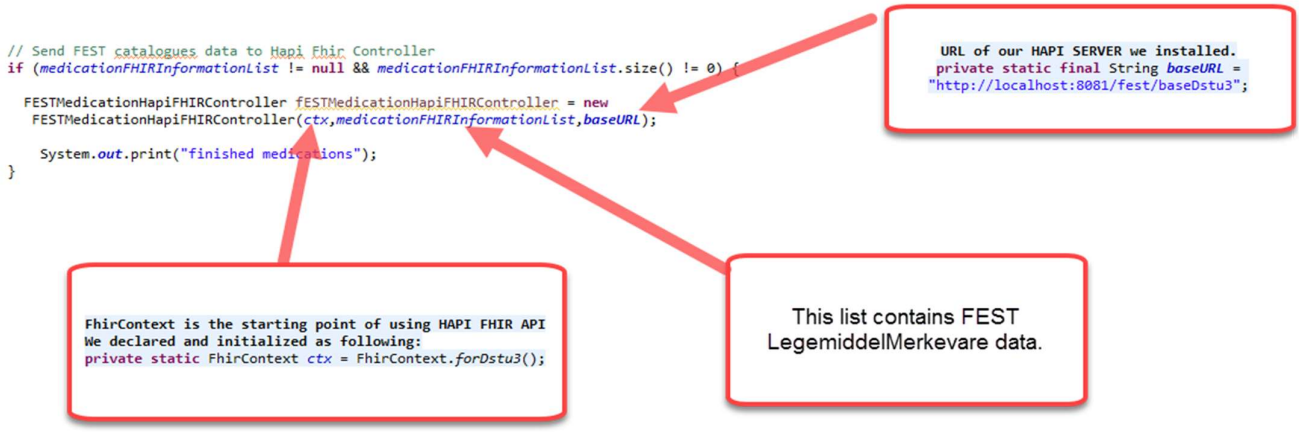


Figure 3.47: Calling FESTMedicationHapiFHIRController

Id	14	Requirement type	Functional requirement 14	Event	Develop class FESTMedicationHapiFHIRController
Description	Create instances of FESTMedication resource and send it to hapi-fhir server.				
Rationale	Save instances of FESTMedication resource on the server.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 2,5,6,7,8,15 and 16				
Conflict	None				

Table 15: Class FESTMedicationHapiFHIRController

The following diagram illustrates class FESTMedicationHapiFHIRController and the other classes used in the implementation of this step. We provide further details by following the execution flow step by step.

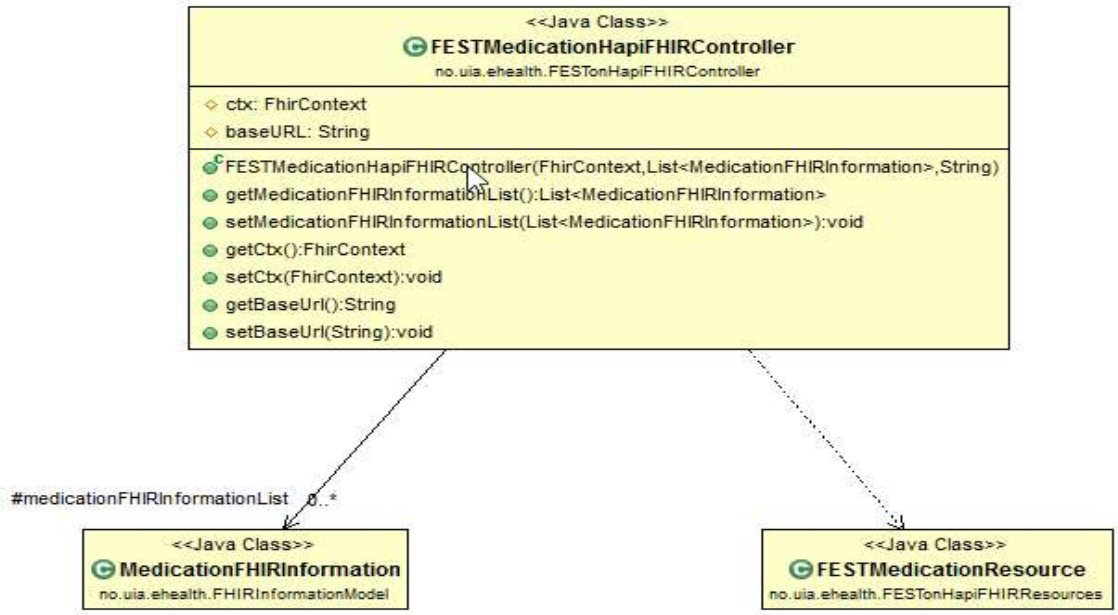


Figure 3.48: FESTMedicationHapiFHIRController

3.7.3.1 Define new search parameters

By default, HAPI FHIR server provides us with search parameters to filter the result of resource instances we search for, i.e. we can search by code, ingredient, language and other parameters.
On the other hand, in Norway, we can search for a medication by its ATC-code, product name (*varenavn*), or the name of active substance (*virkestoff*). Consequently, we need to customize our HAPI-FHIR server to be able to search by *varenavn* or *virkestoff*. But why we introduce this step before creating FEST medication instances and save them on the server? The answer is as simple as when HAPI FHIR server detects new added search parameters; it enforces a re-indexing process for all resource instances it has stored before. From performance perspective, adding the new search parameters before creating and saving the instances will improve the performance of RESTful operations. Inside FESTMedicationHapiFHIRController class and before we start creating instances of FESTMedication resource, we created the following two search parameter and added them to HAPI-FHIR server.

Id	15	Requirement type	Functional requirement 15	Event	Develop search parameter varenavn
Description	Create and add search parameter varenavn to hapi-fhir server.				
Rationale	It allows filtering medication search result by varenavn				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 16: Search parameter varenavn

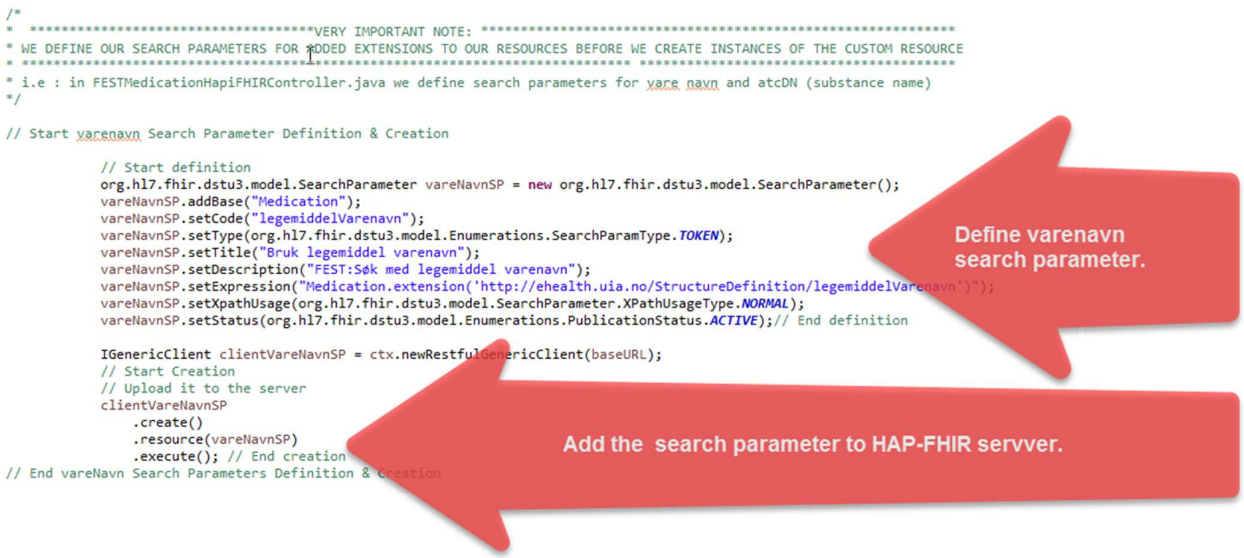


Figure 3.49: Define and add varenavn search parameter.

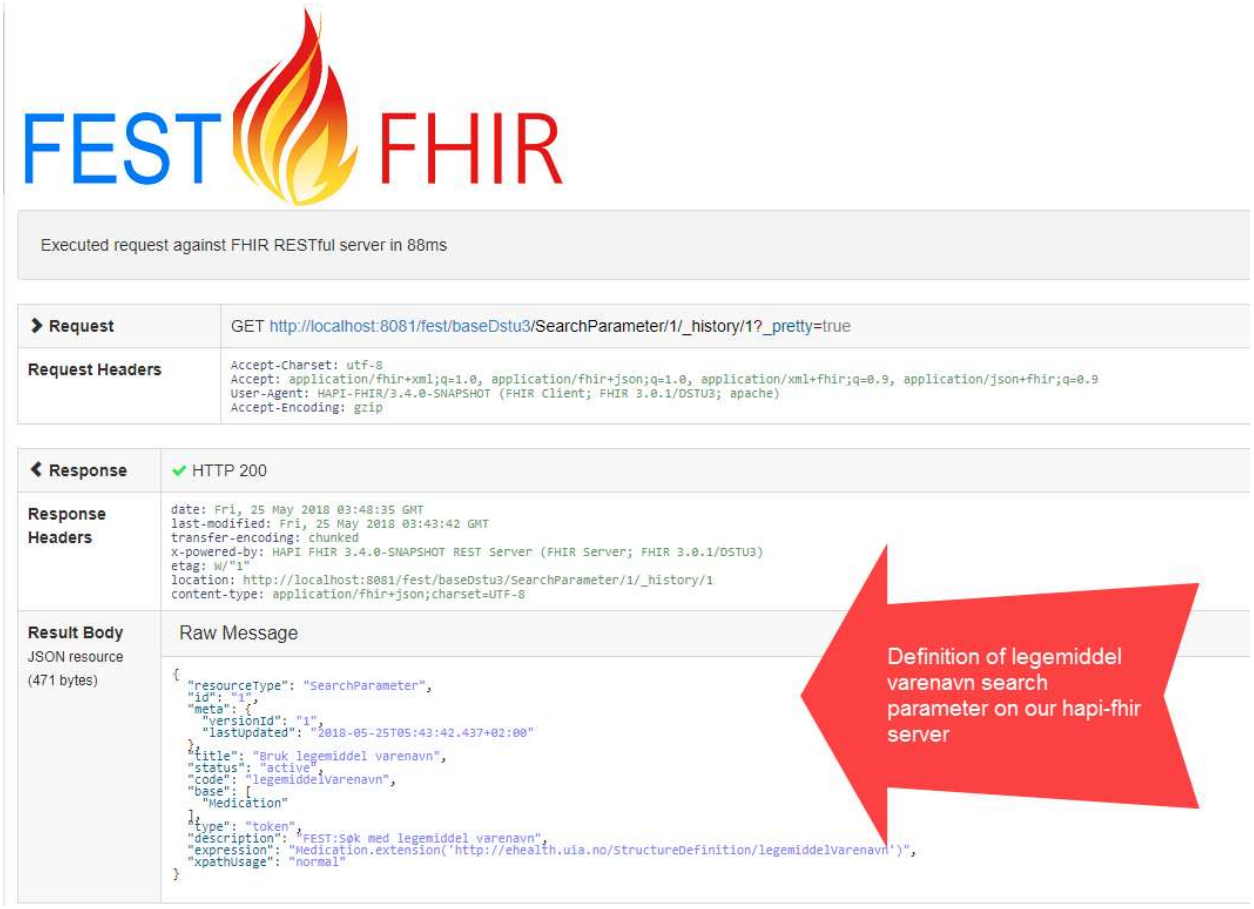


Figure 3.50: Legemiddel varenavn search parameter on HAPI-FHIR server

Id	16	**Requirement type**	Functional requirement 16	**Event**	Develop search parameter Virkestoff navn
Description	Create and add search parameter virkestoff navn to hapi-fhir server.				
Rationale	It allows filtering medication search result by virkestoff navn.				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 17: Search parameter virkestoff navn

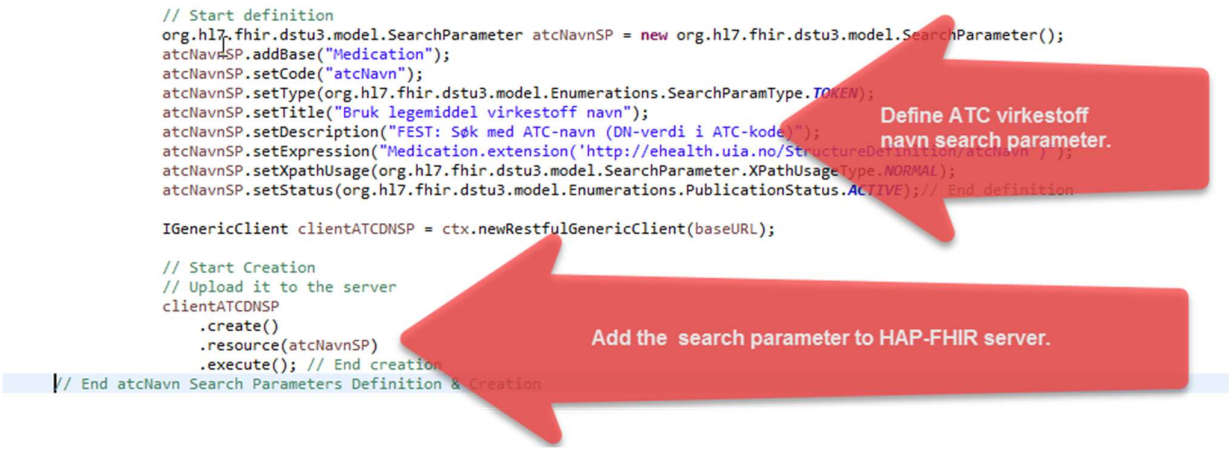


Figure 3.51: Define and add ATC virkestoff navn search parameter.

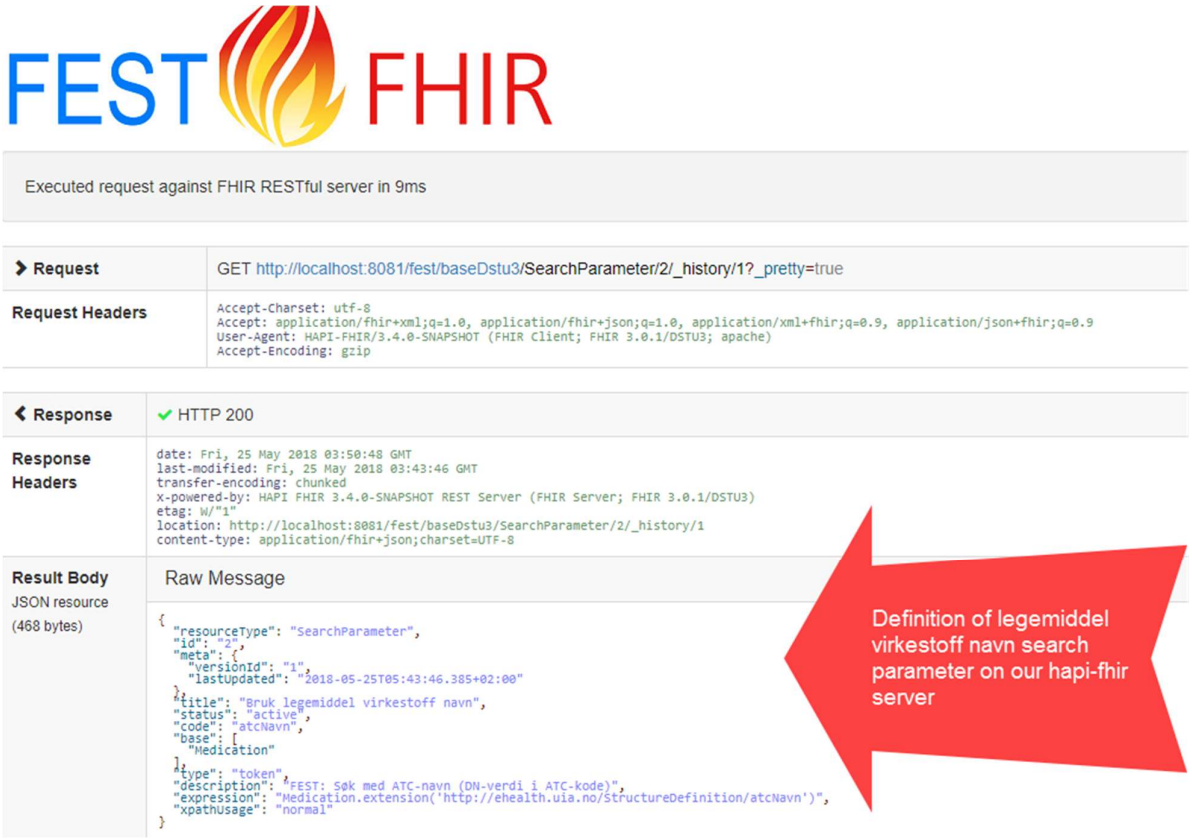


Figure 3.52: Virkestoff navn search parameter on HAPI-FHIR server

Now we can use the added search parameter as they appear in the list of search parameters offered by HAPI-FHIR server.

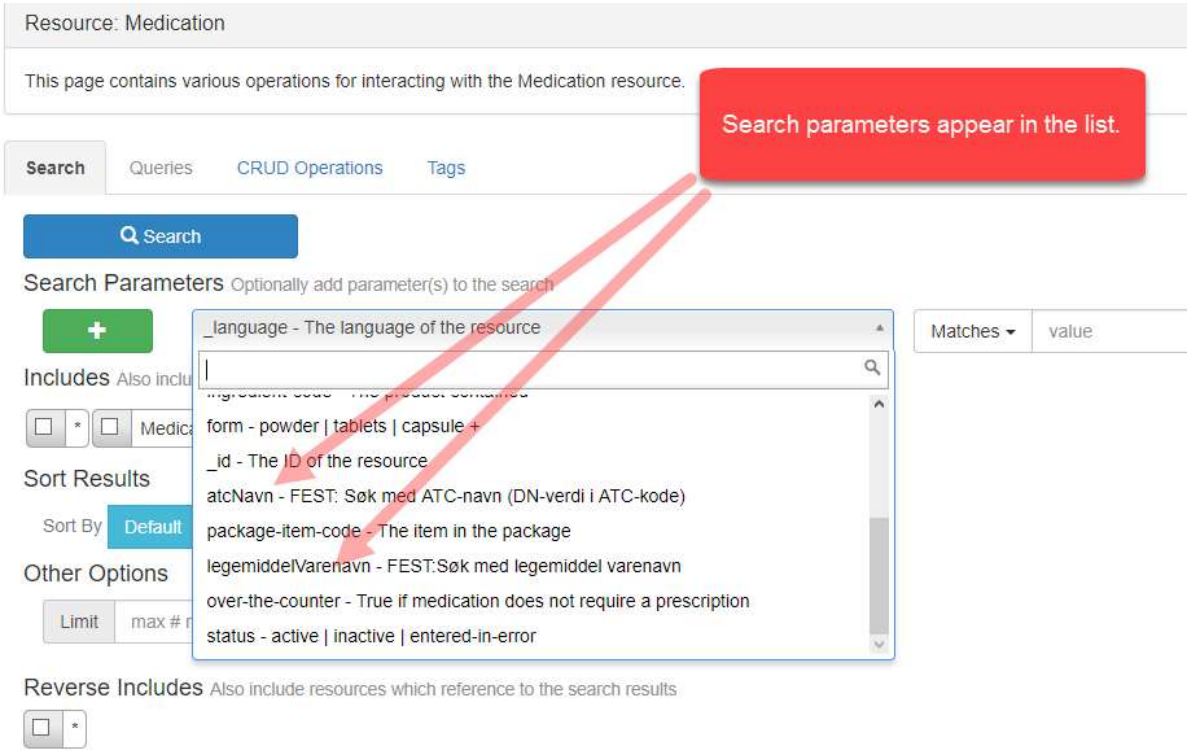


Figure 3.53: Updated list of search parameters for FEST medication instances

3.7.3.2 FESTMedicationResource class

In this section we are going to implement FEST medication profile with the extensions we defined previously.

Id	17	**Requirement type**	Functional requirement 17	**Event**	Develop FESTMedicationResource class
Description	Implement FEST medication profile				
Rationale	Define FESTMedicationResource to initialize FEST medication profile instances and store them on hapi-fhir server.				
Source	Islam Al Khaldi				
Dependencies	15,16				
Conflict	None				

Table 18: Class FESTMedicationResource

Figure 3.54: FESTMedicationResource class

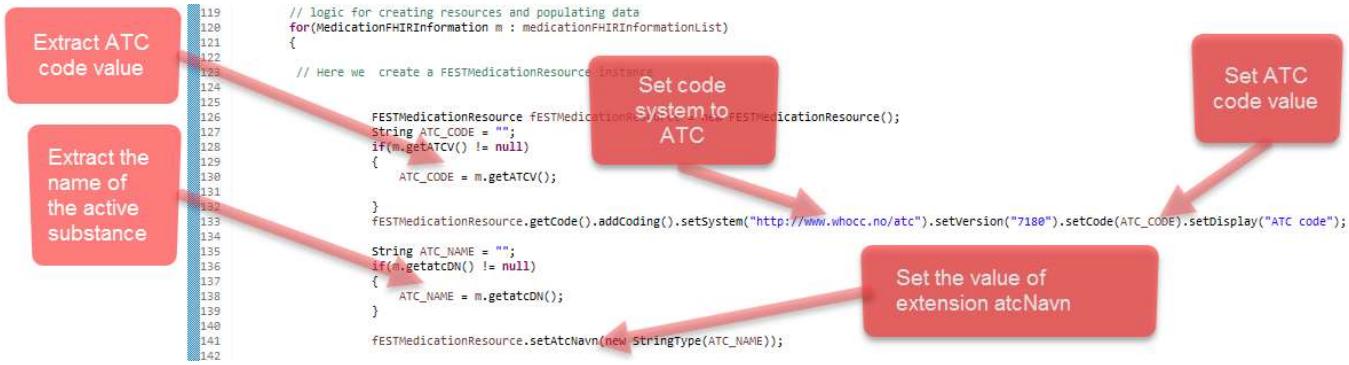
We use annotations to define the class, all extensions we created in our FEST medication profile are implemented as class properties with annotations as well. The following snippets illustrate the implementation of the class and its extensions.

```
13 /**
14  * Definition class for adding extensions to the built-in
15  * Medication resource type.
16  */
17 @ResourceDef(name="Medication", profile="http://ehealth.uia.no/StructureDefinition/FESTMedication")
18 public class FESTMedicationResource extends Medication {
19
```

```
// code.text.ATC-navn: get the ATC DN attribute
@Child(name="atcNavn")
@Extension(url="http://ehealth.uia.no/StructureDefinition/atcNavn", definedLocally=true, isModifier=false)
@Description(shortDefinition="The name associated with the medicine ATC code, DN attribute of element ATC in FEST")
private StringType atcNavn;
// Medication.isOverTheCounter:legemiddelVarselTrekant
@Child(name="legemiddelVarselTrekant")
@Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelVarselTrekant", definedLocally=true, isModifier=false)
@Description(shortDefinition="Indicates a red triangle")
private BooleanType legemiddelVarselTrekant;
// Extension: leggemiddelFESTID
 @Child(name="leggemiddelFESTID")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/LegemiddelFESTID", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel ID in FEST M30 2.5.1")
 private StringType legemiddelFESTID;
 // Extension: legemiddelVarenavn
 @Child(name="legemiddelVarenavn")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelVarenavn", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel varenavn in FEST M30 2.5.1")
 private StringType legemiddelVarenavn;
// Extension: legemiddelNavnFormStyrke
 @Child(name="legemiddelNavnFormStyrke")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelNavnFormStyrke", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel navn form styrke in FEST M30 2.5.1")
 private StringType legemiddelNavnFormStyrke;
 // Extension: legemiddelAdministrasjonsvei   lemiddelEnhetsdosering
 @Child(name="legemiddelAdministrasjonsvei")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelAdministrasjonsvei", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel administration guidance in FEST M30 2.5.1")
 private StringType legemiddelAdministrasjonsvei;
// Extension: lemiddelEnhetsdosering legemiddelKortDose
 @Child(name="legemiddelEnhetsdosering")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelEnhetsdosering", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel unit dosage in FEST M30 2.5.1")
 private StringType legemiddelEnhetsdosering;
 // Extension: legemiddelKortDose
 @Child(name="legemiddelKortDose")
 @Extension(url="http://ehealth.uia.no/StructureDefinition/legemiddelKortDose", definedLocally=true, isModifier=false)
 @Description(shortDefinition="legemiddel short dose in FEST M30 2.5.1")
 private StringType legemiddelKortDose;
```

3.7.3.3 Logic in FESTMedicationHapiFHIRController class

The main goal of FESTMedicationHapiFHIRController class is to create instances of FESTMedicationResource class and save it on HAPI-FHIR server. The following code snippets illustrate how we iterate over MedicationFHIRInformation list, create instances of FESTMedicationResource, and save them on our HAPI-FHIR server.




Set isBrand value
fESTMedicationResource.setIsBrand(true);
if((m.getLegemiddelVarselTrekant() != null) && (m.getLegemiddelVarselTrekant()))
{
fESTMedicationResource.setLegemiddelVarselTrekant(new BooleanType(true));
}
else
{
fESTMedicationResource.setLegemiddelVarselTrekant(new BooleanType(false));
}
if(m.getLegemiddelTypeSoknadV() != null)
{
if(m.getLegemiddelTypeSoknadV().equals("1"))
{
fESTMedicationResource.setIsOverTheCounter(false);
}
else
{
fESTMedicationResource.setIsOverTheCounter(true);
}
}
Set value of extension legemiddelVarselTrekant
The value of isOverTheCounter depends on value of TypeSoknadV, if it equals "1", then we set isOverTheCounter to false, otherwise to true. In FEST, TypeSoknadV means an approval is required.


String producer = "";
if(m.getLegemiddelProdusent() !=null)
{
producer = m.getLegemiddelProdusent();
}
Set manufacturer value
fESTMedicationResource.getManufacturer().setDisplay(producer);
String legemiddelFESTID = "";
if(m.getLegemiddelID() != null)
{
legemiddelFESTID = m.getLegemiddelID();
}
Set value of extension legemiddelFESTID
fESTMedicationResource.setLegemiddelFESTID(new StringType(legemiddelFESTID));
String legemiddelVarenavn ="";
if(m.getLegemiddelVarenavn() != null)
{
legemiddelVarenavn = m.getLegemiddelVarenavn();
}
Set value of extension legemiddelVarenavn
fESTMedicationResource.setLegemiddelVarenavn(new StringType(legemiddelVarenavn));
String legemiddelNavnFormStyrke = "";
if(m.getLegemiddelNavnFormStyrke() != null)
{
}
Set value of extension legemiddelNavnFormStyrke
legemiddelNavnFormStyrke = m.getLegemiddelNavnFormStyrke();
fESTMedicationResource.setLegemiddelNavnFormStyrke(new StringType(legemiddelNavnFormStyrke));

```
201
202                 String legemiddelAdministrasjonsvei = "";
203                 if(m.getLegemiddelAdministrasjonsveiDN() != null)
204                 {
205                     legemiddelAdministrasjonsvei = m.getLegemiddelAdministrasjonsveiDN();
206                 }
207
208                 fESTMedicationResource.setLegemiddelAdministrasjonsvei(new StringType(legemiddelAdministrasjonsvei));
209
210                 String legemiddelEnhetsdosering = "";
211                 if(m.getLegemiddelEnhetsdoseringDN() != null)
212                 {
213                     legemiddelEnhetsdosering = m.getLegemiddelEnhetsdoseringDN();
214                 }
215
216                 fESTMedicationResource.setLegemiddelEnhetsdosering(new StringType(legemiddelEnhetsdosering));
217
218                 String legemiddelKortDose = "";
219                 if(m.getLegemiddelKortDoseDN() != null)
220                 {
221                     legemiddelKortDose = m.getLegemiddelKortDoseDN();
222                 }
223
224                 fESTMedicationResource.setLegemiddelKortDose(new StringType(legemiddelKortDose));
225
226             client.create().resource(fESTMedicationResource).execute();
227
228         //End of create a FESTMedicationResource instance
229
```

Set value of extension legemiddelAdministrasjonvei

Set value of extension legemiddelEnhetsdosering

Set value of extension legemiddelKortDose

Save the instance on HAPI FHIR SERVER

3.7.4 Implement FEST Group profile.

At this very stage, we implement our FEST Group profile we properly defined in Group Profile section of this chapter. Starting again from class AutomaticMappingFESTOnFHIR, we call the constructor of FESTInteractionHapiFHIRController, as shown in the code snippet below.

```
100
101             if (interactionFHIRInformationList != null && interactionFHIRInformationList.size() != 0) {
102
103                 FESTInteractionHapiFHIRController fESTInteractionHapiFHIRController = new
104                 FESTInteractionHapiFHIRController(ctx,interactionFHIRInformationList,baseURL);
105                 System.out.print("finished interactions");
106
107             }
108
```

FhirContext is the starting point of using HAPI FHIR API
We declared and initialized as following:
`private static FhirContext ctx = FhirContext.forDstu3();`
It is an expensive object, so we try to initialize it once and pass a reference to the object.

This list contains FEST Interaksjon data.

URL of our HAPI SERVER we installed.
`private static final String baseURL = "http://localhost:8081/fest/baseDstu3";`

Figure 3.55: *Calling FESTInteractionHapiFHIRController*

Id	18	**Requirement type**	Functional requirement 18	**Event**	Develop class FESTInteractionHapiFHIRController
Description	Create instances of FESTGroup resource and send it to hapi-fhir server.				
Rationale	Save instances of FESTGroup resource on the server.				
Source	Islam Al Khaldi				
Dependencies	Functional requirement 2,5,6,7,8,14 and 19				
Conflict	None				

Table 19: Class FESTInteractionHapiFHIRController

The following diagram illustrates class FESTInteractionHapiFHIRController and the other classes used in the implementation of this step. We provide further details by following the execution flow step by step.

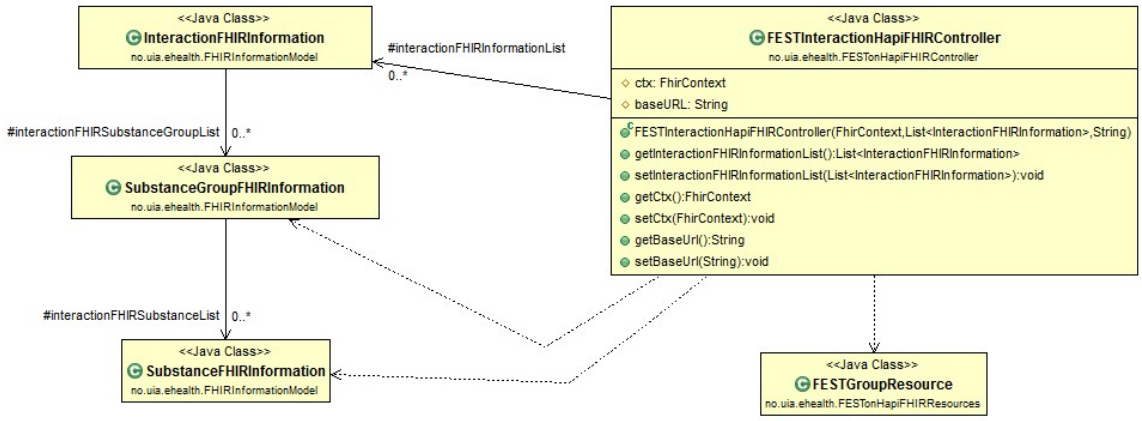


Figure 3.56: FESTInteractionHapiFHIRController

3.7.4.1 FESTGroupResource class

In this section we are going to implement FEST group profile with the extensions we defined previously.

Id	19	**Requirement type**	Functional requirement 19	**Event**	Develop FESTGroupResource class
Description	Implement FEST group profile				
Rationale	Define FESTGroupResource to initialize FEST group profile instances and store them on hapi-fhir server.				
Source	Islam Al Khaldi				
Dependencies	18				
Conflict	None				

Table 20: Class FESTGroupResource

Figure 3.57: FESTGroupResource class

FESTGroup class stores information about only one group of medicines which has drug-drug interaction with another group. Consequently, to represent an interaction, we need two FESTGroup instances having the same interaction id. The only difference in structure between our FESTGroup class and the interaction

complex type of FEST M30 is that in FEST one interaction instance includes two groups whereas an instance of FESTGroup class includes only one group. In each instance of FESTGroup, we must set the type property to the value “medication” and the actual property to the value “true” which means the group refers to a specific group of real individuals / instances.

The following snippets illustrate how each extension we defined in FESTGroup profile is defined in FESTGroupResource class.

```
// Group.extension:interactionID
@Child(name = "interactionID")
@Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionID", definedLocally = true, isModifier = false)
@Description(shortDefinition = "The given interaction ID  in FEST M30 2.5.1")
private StringType interactionID;

// Group.extension:interactionRelevansV
@Child(name = "interactionRelevansV")
@Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionRelevansV", definedLocally = true, isModifier = false)
@Description(shortDefinition = "The severity level of the interaction")
private StringType interactionRelevansV;

// Group.extension:interactionRelevansDN
@Child(name = "interactionRelevansDN")
@Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionRelevansDN", definedLocally = true, isModifier = false)
@Description(shortDefinition = "Severity short description of the interaction")
private StringType interactionRelevansDN;

// Group.extension:interactionKliniskKonsekvens
@Child(name = "interactionKliniskKonsekvens")
@Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionKliniskKonsekvens", definedLocally = true, isModifier = false)
@Description(shortDefinition = "Interaction clinical consequences.")
private StringType interactionKliniskKonsekvens;

// Group.extension:interactionHandtering
@Child(name = "interactionHandtering")
@Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionHandtering", definedLocally = true, isModifier = false)
@Description(shortDefinition = "Interaction guidance")
private StringType interactionHandtering;
```

3.7.4.2 Logic of FESTInteractionHapiFHIRController class

The main goal of FESTInteractionHapiFHIRController class is to create instances of FESTGroupResource class and save it on HAPI-FHIR server. The following code snippets illustrate how we iterate over InteractionFHIRInformation list, create instances of FESTGroupResource, and save them on our HAPI-FHIR server.

```
57          // logic for creating resources and populating data
58
59          for(InteractionFHIRInformation interaction : interactionFHIRInformationList)
60          {
61              String interactionID = "";
62              String interactionFHIRRelevansV = "";
63              String interactionFHIRRelevansDN = "";
64              String interactionFHIRHandtering = "";
65              String interactionFHIRKliniskKonsekvens = "";
66
67              if(interaction.getInteractionFHIRID() != null)
68              {
69                  interactionID = interaction.getInteractionFHIRID();
70              }
71
72              if(interaction.getInteractionFHIRRelevansV() != null)
73              {
74                  interactionFHIRRelevansV =  interaction.getInteractionFHIRRelevansV();
75              }
76
77              if(interaction.getInteractionFHIRRelevansDN() != null)
78              {
79                  interactionFHIRRelevansDN = interaction.getInteractionFHIRRelevansDN();
80              }
81
82              if(interaction.getInteractionFHIRHandtering() != null)
83              {
84                  interactionFHIRHandtering = interaction.getInteractionFHIRHandtering();
85              }
86
87              if(interaction.getInteractionFHIRKliniskKonsekvens() != null)
88              {
89                  interactionFHIRKliniskKonsekvens = interaction.getInteractionFHIRKliniskKonsekvens();
90              }
91
```

Extract interaction id value

Extract interaction guidance information

Extract interaction relevanse V attribue value which indicates level of severity

Extract interaction clinical consequences information

Extract interaction relevanse DN attribue value which describes level of severity

```
92
93          //each interaction has two medication groups, so we loop and process each group
94          List <SubstanceGroupFHIRInformation> interactionRelatedGroups = interaction.getSubstanceGroupFHIRInformation();
95
96          for(SubstanceGroupFHIRInformation medicationsGroup : interactionRelatedGroups)
97          {
98              /*
99               * Here we create the FESTGroupResource instance
100              */
101             //Start of create a FESTGroupResource instance
102             FESTGroupResource fESTGroupResource = new FESTGroupResource();
103             // Add interaction details.
104             fESTGroupResource.setInteractionID(new StringType(interactionID));
105             fESTGroupResource.setInteractionRelevansV(new StringType(interactionFHIRRelevansV));
106             fESTGroupResource.setInteractionRelevansDN(new StringType(interactionFHIRRelevansDN));
107             fESTGroupResource.setInteractionHandtering(new StringType(interactionFHIRHandtering));
108             fESTGroupResource.setInteractionKliniskKonsekvens(new StringType(interactionFHIRKliniskKonsekvens));
109
110
111             fESTGroupResource.setActual(true); // means the group refers to specific group of real medication instances
112             fESTGroupResource.setActive(true); // means the group refers to specific group of real medication instances
113
114             //GroupType.MEDICATION
115             fESTGroupResource.setType(FESTGroupResource.GroupType.valueOf("MEDICATION"));
```

Each interaction has two groups. We iterate over each group

Create FESTGroupResource instance

Set values of extensions we defined in FESTGroupResource class for the current instance.

Set actual, active and type properties of the instance.

3.7.4.3 FhirServerConfig.java

We had to customize the FhirServerConfig class to add our base URL in order to avoid Invalid Request Exceptions when we create extensions. Simply we edited “daoConfig” Bean and let our url http://localhost:8081/fest being treated as local. Then we had to re-build hapi-fhir server and run the Java application of our solution again.

```java
@Configuration
@EnableTransactionManagement()
public class FhirServerConfig extends BaseJavaConfigDstu3 {


    /**
     * Configure FHIR properties around the the JPA server via this bean
     */
    @Bean()
    public DaoConfig daoConfig() {
        DaoConfig retVal = new DaoConfig();
        retVal.setAllowMultipleDelete(true);
        retVal.setAllowExternalReferences(true);
        retVal.getTreatBaseUrlsAsLocal().add("http://localhost:8081/fest/");
        return retVal;
    }

```

Figure 3.58: FhirServerConfig class

At this very stage of our implementation, we have FEST medication resource instances and FEST group resource instances saved on our HAPI-FHIR server. The following figure illustrates the home page of our HAPI-FHIR server.

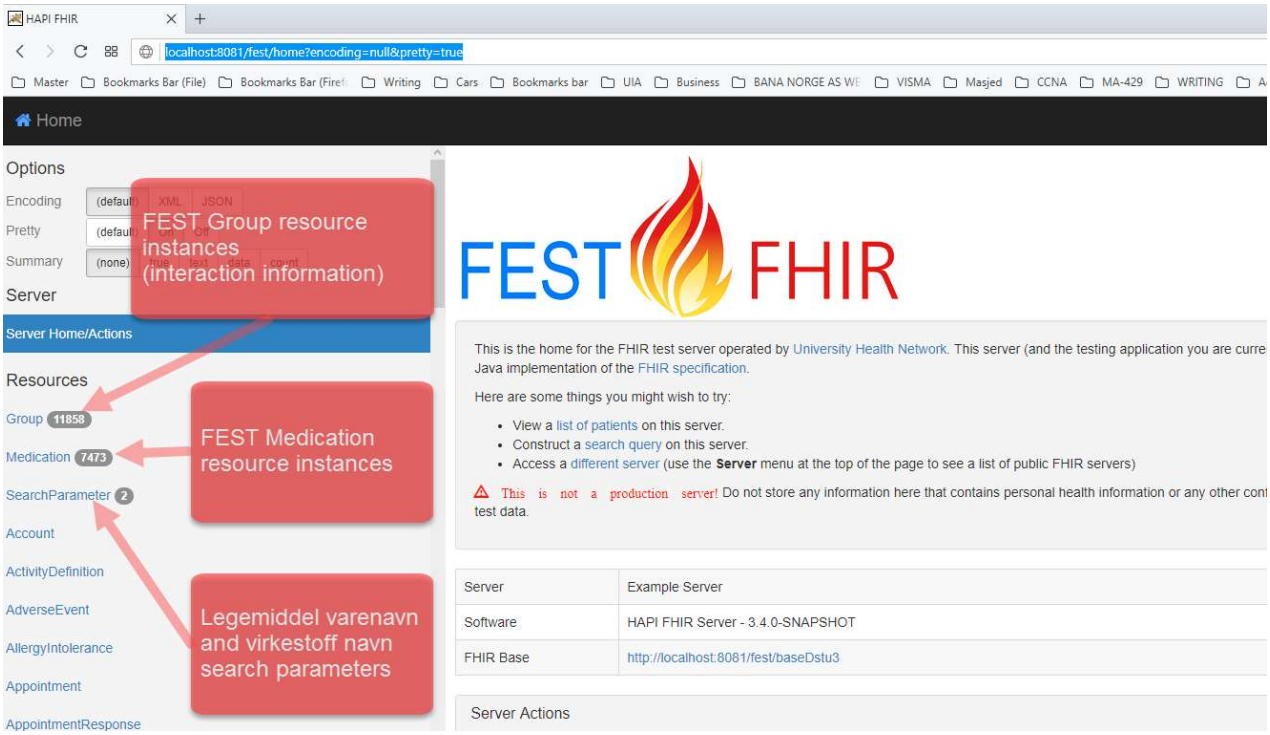


Figure 3.59: FEST on FHIR HAPI-FHIR server

3.7.5 Implementation of FEST Drug-drug interaction mechanism

The importance behind this section is that it is the proof of concept of our solution implementation. The user of our system will input the names/codes of two drugs and send a query to our HAPI-FHIR server where FEST data are stored in form of FEST medication resource instances and FEST group resource instances. Our implementation of FEST drug-drug interaction mechanism should be able to detect drug-drug interaction, if any, and return the details.
This interaction detection process comprises reading and searching for FHIR resource instances by using search parameters including the parameters we added. Consequently, if we get an interaction result which is identical to the result we may obtain by using drug-drug interaction analysis tools offered by www.interaksjoner.no or www.felleskatalogen.no sites which use FEST , then we can say that we have FEST on FHIR. Let's recall how FEST interaction mechanism is defined in FEST M30. When we check drug-drug interaction for two medicines. In case the active substance of one medicine exists in one substance group, and the active substance of the other medicine exists in the second substance group, then there is an interaction [34]. The following flowchart illustrates FEST logic of finding interaction.

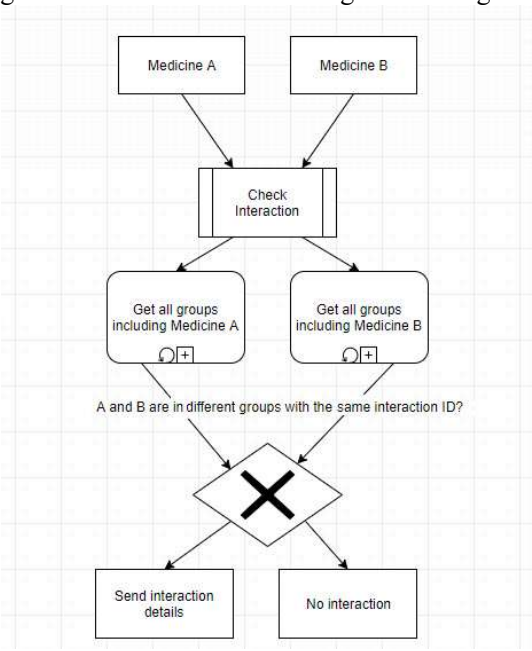


Figure 3.60: FEST interaction mechanism flowchart

We will follow the execution workflow step by step in our explanation. First, we start by determining the type of the search keyword the user can use to check for drug-drug interaction between two FEST medication resource instances hosted on our HAPI-FHIR server. In order to provide the user with search flexibility, our implementation can accept medicine brand name (*legemiddel varenavn*), ATC-code, or the name of active substance (*legemiddel virkestoff navn*).

Id	20	**Requirement type**	Functional requirement 20	**Event**	Develop CheckForInteraction class
Description	Implement FEST interaction mechanism				
Rationale	To find interaction details, if any, between two medication resource instances on hapi-fhir server				
Source	Islam Al Khaldi				
Dependencies	8,11,14,18,21,22 and 23				
Conflict	None				

Table 21: Class CheckForInteraction

The following diagram illustrates the relations between CheckForInteraction class and the other classes we created in our implementation.

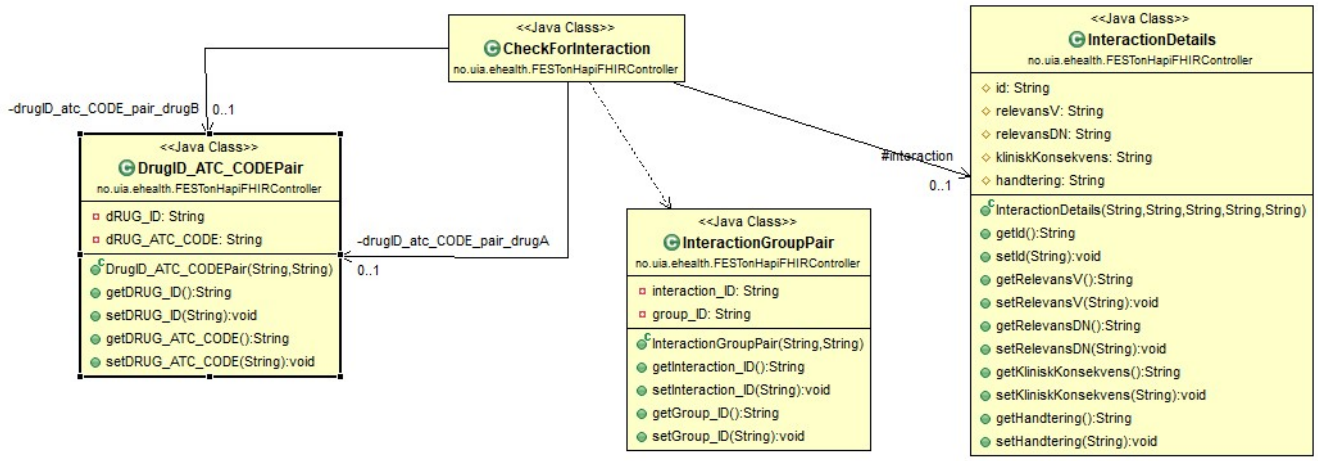


Figure 3.61: CheckForInteraction class relations

Id	21	**Requirement type**	Functional requirement 21	**Event**	Develop DrugID_ATC_CODEPair class
Description	Store drug id and ATC-code of the search keyword				
Rationale	Whatever the search keyword the user entered, we should be able to get the drug id and active substance ATC-code.				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 22: Class DrugID_ATC_CODEPair

Id	22	**Requirement type**	Functional requirement 22	**Event**	Develop InteractionGroupPair class
Description	Store interaction id and group id of the drug.				
Rationale	Interaction exists when two drugs have the same interaction id, but they belong to different groups. This class helps us when we compare two drugs by their interaction id and group id.				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 23: Class InteractionGroupPair

Id	23	**Requirement type**	Functional requirement 23	**Event**	Develop InteractionDetails class
Description	Store interaction id, severity, guidance and clinical consequences.				
Rationale	To send interaction details,if any, to the user.				
Source	Islam Al Khaldi				
Dependencies	None				
Conflict	None				

Table 24: Class InteractionDetails

The following code snippets illustrate the logic of drug-drug interaction mechanism step by step.

```
54
55      public CheckForInteraction(FhirContext ctx, String drugA, String drugB, String baseURL) {
56          this.ctx = ctx;
57          this.drugA = drugA;
58          this.drugB = drugB;
59          this.baseURL = baseURL;
60
61          // 1. we need to get the ATC-CODE of drugA and drugB.
62
63          // The practitioner can search by ATC-CODE, vareNavn, or active substance name.
64          // 1.1 Check whether drugA search keyword is ATC-CODE, product name (vareNavn) ,
65          // or active substance name
66          this.drugID_atc_CODE_pair_drugA = inputCheckerA(drugA);
67
```

Step#1
1.1 Determine the type of drugA search keyworkd.
1.2 Determine the type of drugB search keyworkd.

We call inputCheckerA method and send drug A as an argument

```
197     private DrugID_ATC_CODEPair inputCheckerA(String drug) {
198
199         // 1.1.1 ATC-CODE checker
200         IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
201         Bundle resultATCV = clientATCV.search().forResource(Medication.class)
202                 .where(Medication.CODE.exactly().code(drug)).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class)
203                 .execute();
204         if (resultATCV != null) {
205             if (resultATCV.getTotal() == 0) {
206                 this.flagATCCodeA = false;
207             } else {
208                 this.flagATCCodeA = true;
209                 if (resultATCV.getEntry().get(0).getResource() != null) {
210                     if (resultATCV.getEntry().get(0).getResource() instanceof Medication) {
211                         Medication m = (Medication) resultATCV.getEntry().get(0).getResource();
212                         this.aTC_CODE_A = drug;
213                         this.drugA_ID = m.getId();
214                         this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), drug);
```

Is drug A an ATC-code?
We send a query to our HAPI-FHIR server. If ATC-code, we store the pair medication instance id and ATC-code in an object of class DrugID_ATC_CODEPair. If not, we go to next step.

```
223         if (!flagATCCodeA) {
224             // 1.1.2 Legemiddel varenavn checker
225             // Note that we have defined the search parameter legemiddelVarenavn in
226             // FESTMedicationHapiFHIRController.java
227             // use the added search parameter
228             IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
229             Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class)
230                     .where(new TokenClientParam("legemiddelVarenavn").exactly().code(drug)).returnBundle(Bundle.class)
231                     .execute();
232
233             if (resultVareNavn != null) {
234
235                 if (resultVareNavn.getTotal() == 0) {
236                     this.flagVareNavnA = false;
237                 } else {
238                     this.flagVareNavnA = true;
239                     if (resultVareNavn.getEntry().get(0).getResource() != null) {
240                         if (resultVareNavn.getEntry().get(0).getResource() instanceof Medication) {
241                             FESTMedicationResource m = (FESTMedicationResource) resultVareNavn.getEntry().get(0)
242                                     .getResource();
243                             this.drugA_ID = m.getId();
244                             this.aTC_CODE_A = m.getCode().getCoding().get(0).getCode();
245
246                             this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(),
247                                     m.getCode().getCoding().get(0).getCode());
248                         }
```

Here we know that drug A is not ATC-code, so we send a query to our HAPI-FHIR server to check if drug A is varenavn. We use our search parameter "legemiddelVarenavn"

If Varenavn, we get the ATC-code and store the pair medication instance id and ATC-code in an object of class DrugID_ATC_CODEPair. If not, we go to next step.

```
259         if (!this.flagATCCodeA && !this.flagVareNavnA) {
260             // 1.1.3 active substance name checker
261             // Note that we have defined the search parameter atcNavn in
262             // FESTMedicationHapiFHIRController.java
263             // use the added search parameter
264             IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
265             Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class)
266                     .where(new TokenClientParam("atcNavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
267
268             if (resultSubstanceName != null) {
269
270                 if (resultSubstanceName.getTotal() == 0) {
271                     this.flagSubsNameA = false;
272                 } else {
273                     this.flagSubsNameA = true;
274                     if (resultSubstanceName.getEntry().get(0).getResource() != null) {
275                         if (resultSubstanceName.getEntry().get(0).getResource() instanceof Medication) {
276                             FESTMedicationResource m = (FESTMedicationResource) resultSubstanceName.getEntry().get(0)
277                                     .getResource();
278                             this.drugA_ID = m.getId();
279                             this.aTC_CODE_A = m.getCode().getCoding().get(0).getCode();
280                             this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(),
281                                     m.getCode().getCoding().get(0).getCode());
282                         }
283
284                     }
285
286                 }
287             }
288
289         }
290
291         return drugID_atc_CODE_pair_drugA;
292     }
293
```

Here we know that drug A is neither ATC-code nor varenavn, so we send a query to our HAPI-FHIR server to check if drug A is name of active substance (atcNavn). We use our search parameter "atcNavn"

If Varenavn, we get the ATC-code and store the pair medication instance id and ATC-code in an object of class DrugID_ATC_CODEPair. If not, we go to next step.

return pair of ATC-code and medication id

```
68          // 1.2 Check whether drugB search keyword is ATC-CODE, product name (vareNavn) ,
69          // or active substance name
70          this.drugID_atc_CODE_pair_drugB = inputCheckerB(drugB);
71
```

We call inputCheckerB method and send drug B as an argument

```
385
386         return drugID_atc_CODE_pair_drugB;
387     }
388
```

We follow the same steps for Drug A and return pair of ATC-code and medication id

Step# 1 finished.

Step# 2 start.

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
```

```
// 1. we need to get the ATC-CODE of drugA and drugB.

// The practitioner can search by ATC-CODE, vareNavn, or active substance name.
// 1.1 Check whether drugA search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugA = inputCheckerA(drugA);

// 1.2 Check whether drugB search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugB = inputCheckerB(drugB);

// 2. Search for all FESTGroup instances which include Medication reference
// entries for drugA
// store IDs of interactions and groups for drug A
List<InteractionGroupPair> interaction_group_drugA = drugGroups(this.drugID_atc_CODE_pair_drugA);

// 3. Search for all FESTGroup instances which include Medication reference
// entries for drugB
// store IDs of interactions and groups for drug B
List<InteractionGroupPair> interaction_group_drugB = drugGroups(this.drugID_atc_CODE_pair_drugB);

// 4. Compare both lists detect interaction, if any.
```

We need a list of all interactions and groups including drug A. Each entry in this list will be of type InteractionGroupPair class

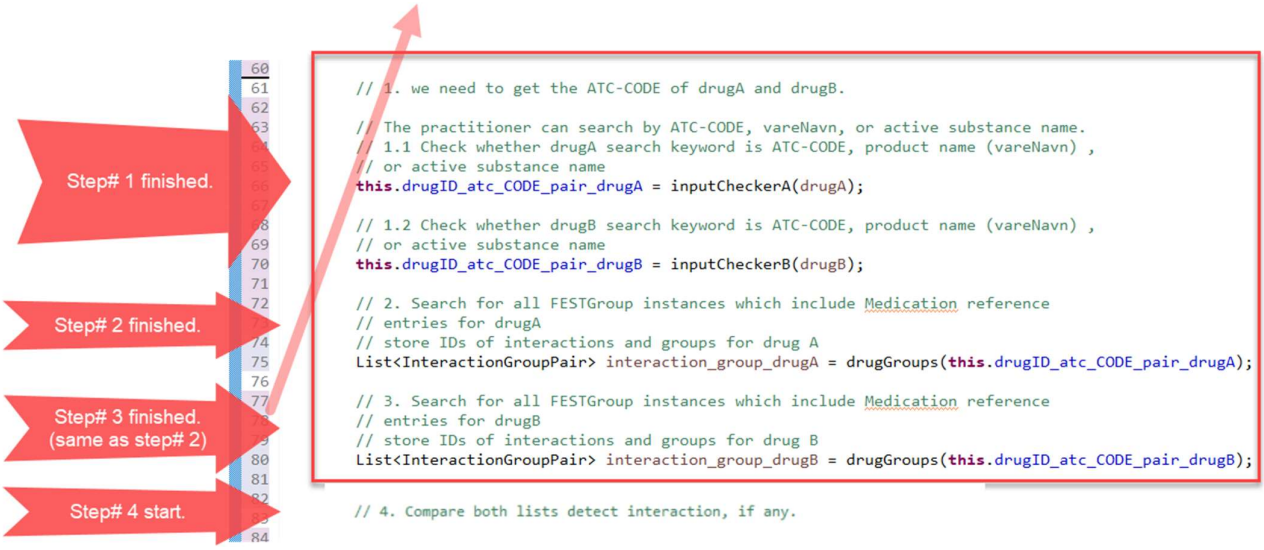
We send a query to hapi-fhir server to get all FEST Group instances including a FEST Medication instance with ID equals to the drug id we got in step# 1.

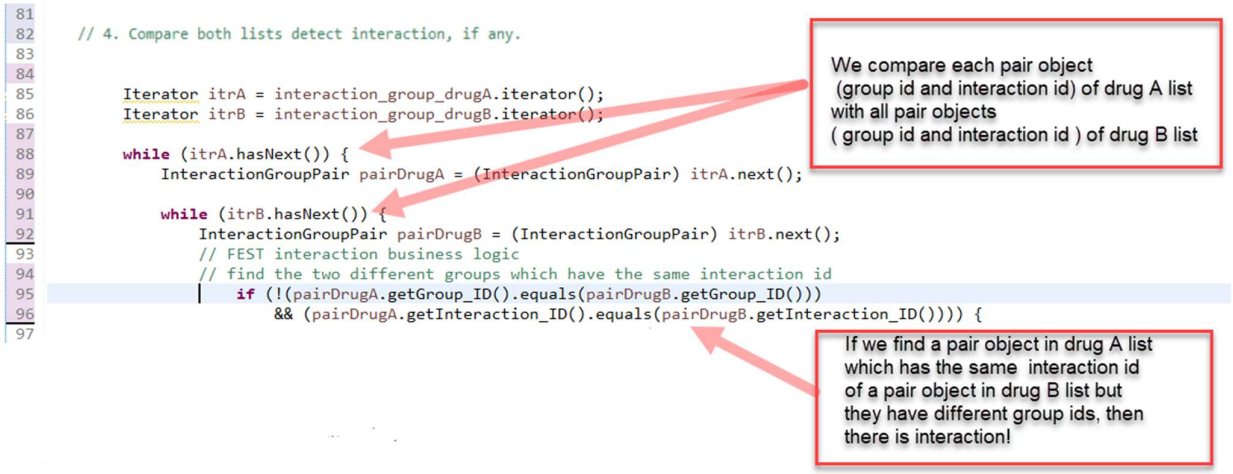
Add the pair interaction id and group id to the list.

Return a list of groups and interactions including this drug.

```
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
```

```
private List<InteractionGroupPair> drugGroups(DrugID_ATC_CODEPair pair) {
    List<InteractionGroupPair> interaction_group_for_drug = new ArrayList<InteractionGroupPair>();
    IGenericClient clientDrugAGroupIDs = ctx.newRestfulGenericClient(baseURL);

    Bundle resultGroups = clientDrugAGroupIDs.search().forResource(Group.class)
            .where(Group.MEMBER.hasId(pair.getDRUG_ID())).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class)
            .execute();
    if (resultGroups != null) {

        if (resultGroups.getTotal() > 0) {
            for (BundleEntryComponent e : resultGroups.getEntry()) {
                String groupID = "";
                String interactionID = "";

                if (e.getResource() instanceof Group) {
                    // get group id
                    Group g = (Group) e.getResource();
                    groupID = g.getId();

                    // get interaction id
                    List<Extension> resourceExts = g
                            .getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionID");
                    Extension exti = resourceExts.get(0);
                    interactionID = exti.getValue().toString();

                    InteractionGroupPair p = new InteractionGroupPair(interactionID, groupID);
                    interaction_group_for_drug.add(p);
                }
            }

        }

    }

    return interaction_group_for_drug;
}
```


Step# 1 finished.
Step# 2 finished.
Step# 3 finished.
(same as step# 2)
Step# 4 start.
// 1. we need to get the ATC-CODE of drugA and drugB.
// The practitioner can search by ATC-CODE, vareNavn, or active substance name.
// 1.1 Check whether drugA search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugA = inputCheckerA(drugA);
// 1.2 Check whether drugB search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugB = inputCheckerB(drugB);
// 2. Search for all FESTGroup instances which include Medication reference
// entries for drugA
// store IDs of interactions and groups for drug A
List<InteractionGroupPair> interaction_group_drugA = drugGroups(this.drugID_atc_CODE_pair_drugA);
// 3. Search for all FESTGroup instances which include Medication reference
// entries for drugB
// store IDs of interactions and groups for drug B
List<InteractionGroupPair> interaction_group_drugB = drugGroups(this.drugID_atc_CODE_pair_drugB);
// 4. Compare both lists detect interaction, if any.


// 4. Compare both lists detect interaction, if any.
Iterator itrA = interaction_group_drugA.iterator();
Iterator itrB = interaction_group_drugB.iterator();
while (itrA.hasNext()) {
InteractionGroupPair pairDrugA = (InteractionGroupPair) itrA.next();
while (itrB.hasNext()) {
InteractionGroupPair pairDrugB = (InteractionGroupPair) itrB.next();
// FEST interaction business logic
// find the two different groups which have the same interaction id
if (!(pairDrugA.getGroup_ID().equals(pairDrugB.getGroup_ID()))
&& (pairDrugA.getInteraction_ID().equals(pairDrugB.getInteraction_ID()))) {
We compare each pair object
(group id and interaction id) of drug A list
with all pair objects
(group id and interaction id) of drug B list
If we find a pair object in drug A list
which has the same interaction id
of a pair object in drug B list but
they have different group ids, then
there is interaction!

```
    // get interaction details for this group id
    String id = "";
    String relevansV = "";
    String relevansDN = "";
    String kliniskKonsekvens = "";
    String handtering = "";

    IGenericClient client = ctx.newRestfulGenericClient(baseURL);
    Group theOne = client.read().resource(Group.class).withId(pairDrugA.getGroup_ID()).execute();
    // get interaction ID
    List<Extension> resourceExtsInteractionID = theOne
            .getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionID");
    if (resourceExtsInteractionID != null) {
        Extension interactionID = resourceExtsInteractionID.get(0);
        id = interactionID.getValue().toString();
    }

    // get Interaction Relevans V attribute
    List<Extension> resourceExtsInteractionRelevansV = theOne
            .getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionRelevansV");
    if (resourceExtsInteractionRelevansV != null) {
        Extension interactionRelevansV = resourceExtsInteractionRelevansV.get(0);
        relevansV = interactionRelevansV.getValue().toString();
    }
    // get Interaction Relevans DN attribute
    List<Extension> resourceExtsInteractionRelevansDN = theOne
            .getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionRelevansDN");
    if (resourceExtsInteractionRelevansDN != null) {
        Extension interactionRelevansDN = resourceExtsInteractionRelevansDN.get(0);
        relevansDN = interactionRelevansDN.getValue().toString();
    }
    // get Interaction KliniskKonsekvens
    List<Extension> resourceExtsInteractionKliniskKonsekvens = theOne.getExtensionsByUrl(
            "http://ehealth.uia.no/StructureDefinition/interactionKliniskKonsekvens");
    if (resourceExtsInteractionKliniskKonsekvens != null) {
        Extension interactionKliniskKonsekvens = resourceExtsInteractionKliniskKonsekvens.get(0);
        kliniskKonsekvens = interactionKliniskKonsekvens.getValue().toString();
    }
    // get Interaction Handtering
    List<Extension> resourceExtsInteractionHandtering = theOne
            .getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionHandtering");
    if (resourceExtsInteractionHandtering != null) {
        Extension interactionHandtering = resourceExtsInteractionHandtering.get(0);
        handtering = interactionHandtering.getValue().toString();
    }

    // set interaction details
    this.interaction = new InteractionDetails(id, relevansV, relevansDN, kliniskKonsekvens, handtering);
    this.isInteraction = true;
    break;
}
```

We send query to hapi-fhir server to get the details of the interaction found between drug A and drug B.

A new object of type InteractionDetails class is created and we end the iterations.

Step# 1 finished.

Step# 2 finished.

Step# 3 finished. (same as step# 2)

Step# 4 finished.

```
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
```

```
// 1. we need to get the ATC-CODE of drugA and drugB.

// The practitioner can search by ATC-CODE, vareNavn, or active substance name.
// 1.1 Check whether drugA search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugA = inputCheckerA(drugA);

// 1.2 Check whether drugB search keyword is ATC-CODE, product name (vareNavn) ,
// or active substance name
this.drugID_atc_CODE_pair_drugB = inputCheckerB(drugB);

// 2. Search for all FESTGroup instances which include Medication reference
// entries for drugA
// store IDs of interactions and groups for drug A
List<InteractionGroupPair> interaction_group_drugA = drugGroups(this.drugID_atc_CODE_pair_drugA);

// 3. Search for all FESTGroup instances which include Medication reference
// entries for drugB
// store IDs of interactions and groups for drug B
List<InteractionGroupPair> interaction_group_drugB = drugGroups(this.drugID_atc_CODE_pair_drugB);


// 4. Compare both lists detect interaction, if any.
```

3.8 Practitioner use case

In this section we implement the required back-end functionalities for a practitioner use case where a practitioner uses a web application to do the following CRUD operations scenario in sequence against our HAPI-FHIR server:

1. Read patient information from HAPI-FHIR server.
2. Read patient's medication statements from HAPI-FHIR server.
3. Check for drug-drug interaction before assigning a new medication statement to the patient.
4. Read medication details from HAPI-FHIR server.

The following mockup illustrates a patient profile web page in which the patient information and his/her medication statements are read from HAPI-FHIR server.

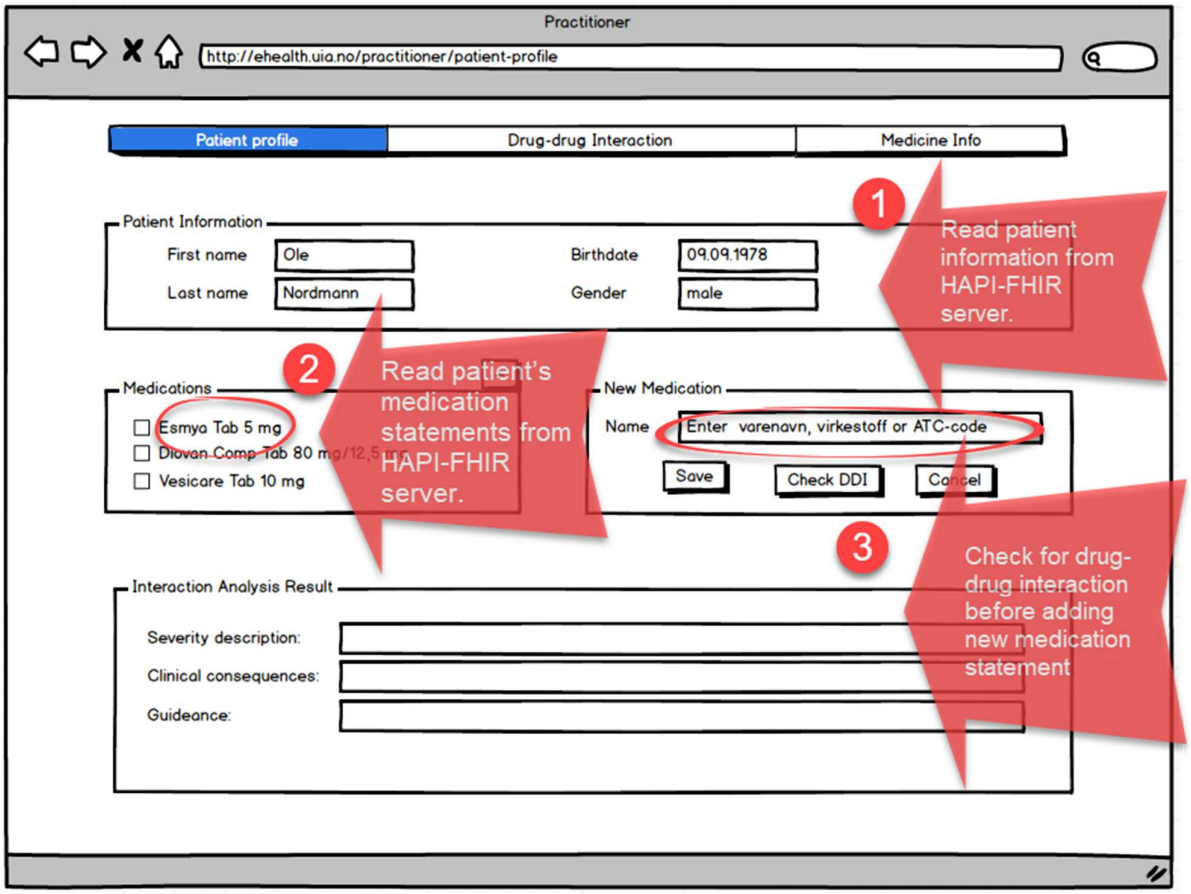


Figure 3.62: Patient Profile

3.8.1 Read patient information

The following snippet code can be used to read/retrieve patient information from HAPI-FHIR server.

```
47
48      private static FhirContext ctx = FhirContext.forDstu3();
49      private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
50      IGenericClient client = ctx.newRestfulGenericClient(baseURL);
51      // assume we know the id of the current patient is for example 19352
52      Patient patient = client.read().resource(Patient.class).withId(String.valueOf("19352")).execute();
53
```

3.8.2 Read patient's medication statements

The following snippet code can be used to read/retrieve patient's medication statements information from HAPI-FHIR server.

```
51
52      private static FhirContext ctx = FhirContext.forDstu3();
53      private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
54      IGenericClient client = ctx.newRestfulGenericClient(baseURL);
55      // assume we know the id of the current patient is for example 19352
56      IBaseBundle bundle = client.search().forResource(MedicationStatement.class)
57              .where(new ReferenceClientParam("patient").hasId("19352"))
58              .prettyPrint()
59              .execute();
60
```

3.8.3 Check for drug-drug interaction

We use our durg-drug interaction implementation we explained previously in section 3.7.5
In the following snippet code, we assume that the practitioner has checked one of the active medication statements and entered the varenavn, ATC-code, or active substance name of the new medication.

```
64      private static FhirContext ctx = FhirContext.forDstu3();
65      private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
66      // Assume the practitioner check for drug-drug interactin between a medication with ATC-code J02AB02
67      // and another medication with varenavn "Esmya"
68      CheckForInteraction interactionCheckup = new CheckForInteraction(ctx, "J02AB02", "Esmya", baseURL);
69
```

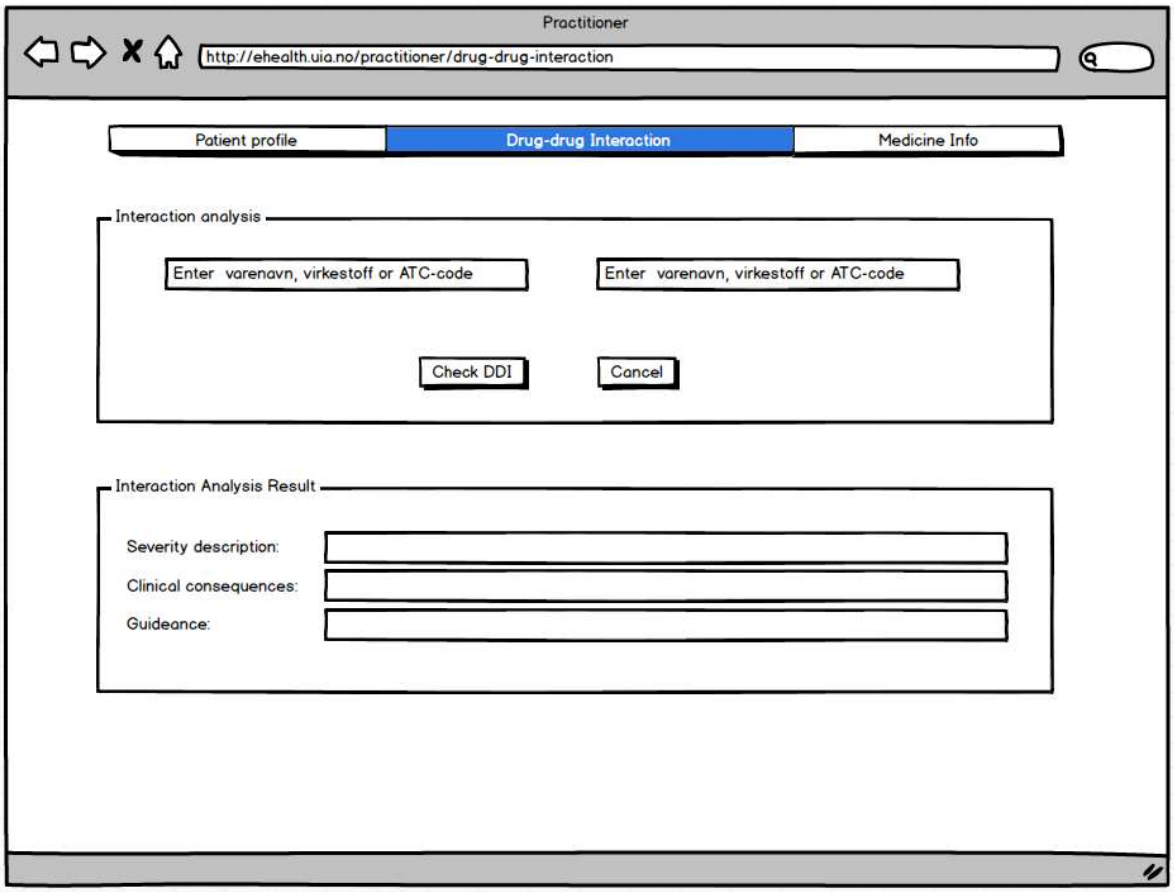


Figure 3.63: Drug-drug interaction

3.8.4 Read medication details from HAPI-FHIR server.

The following snippet codes can be used to read medication details.

```
private static FhirContext ctx = FhirContext.forDstu3();
private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
// if the practitioner use ATC CODE, for example J02AB02
Bundle resultATCV = clientATCV.search().forResource(Medication.class)
        .where(Medication.CODE.exactly().code("J02AB02")).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class)
        .execute();
```

```
private static FhirContext ctx = FhirContext.forDstu3();
private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
// if the practitioner use varenavn, for example Ketoconazole HRA
IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class)
        .where(new TokenClientParam("legemiddelVarenavn").exactly().code("Ketoconazole HRA")).returnBundle(Bundle.class)
        .execute();
```

```
private static FhirContext ctx = FhirContext.forDstu3();
private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
// if the practitioner use active substance name, for example Ketokonazol
IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class)
        .where(new TokenClientParam("atcNavn").exactly().code("Ketokonazol")).returnBundle(Bundle.class).execute();
```

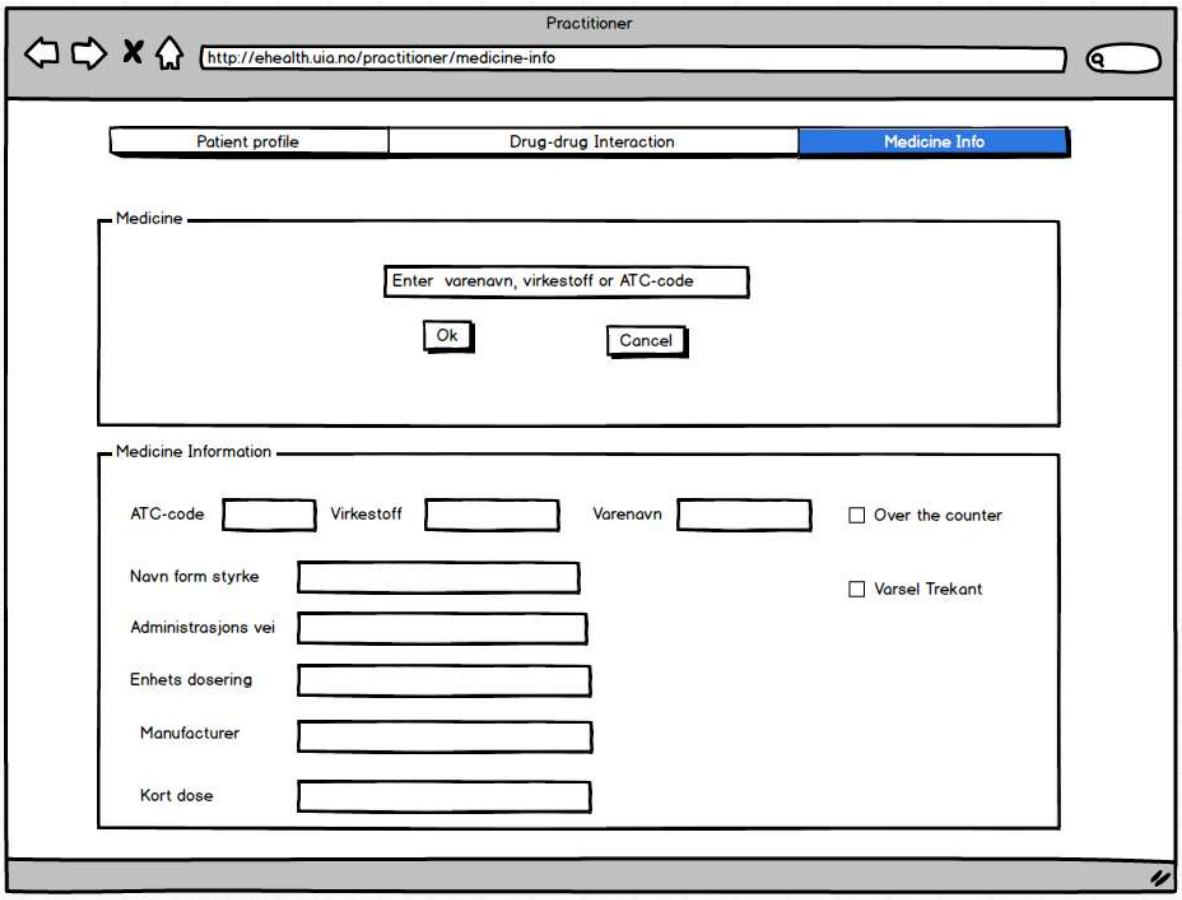


Figure 3.64: Medication details

3.8.5 Validation and testing of drug-drug interaction result

We have implemented the "check for drug-drug interaction" scenario in practitioner use case.
The following screenshot shows the home page of the drug-drug interaction analysis Java web application we developed where the practitioner checks for drug-drug interaction and gets interaction result, if any.
At the web server side, the practitioner's request is handled by the class "CheckForInteraction" where it sends a search query to our HAPI-FHIR server and returns the result.

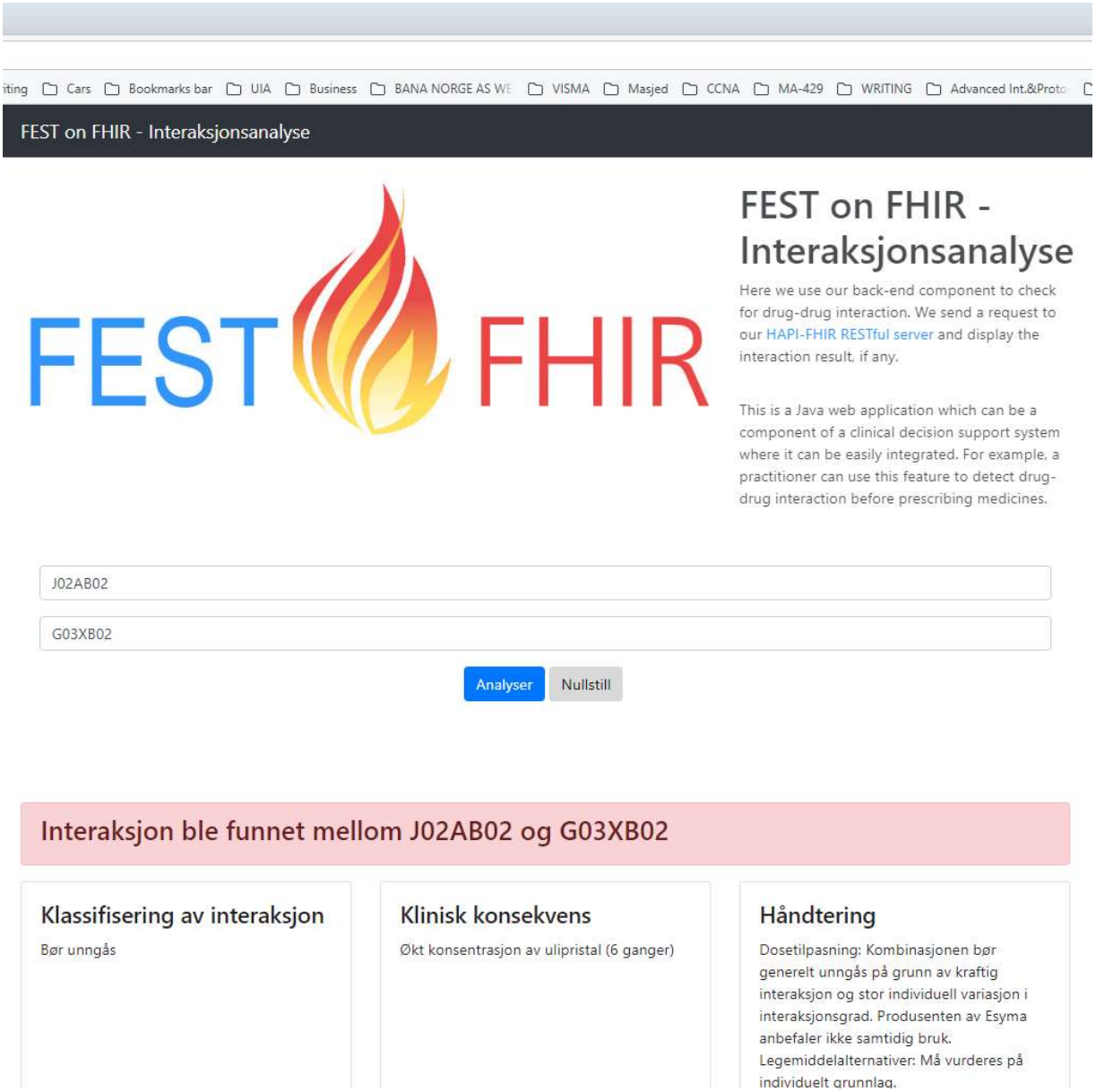


Figure 3.65: FEST on FHIR drug-drug interaction analysis Java web application

We take the following steps to ensure that the result of our solution is valid:
First, we select randomly two medications from the FEST xml file where there is an interaction defined by FEST M30 between them. Secondly, we use our drug-drug interaction analysis Java web application. We compare the interaction result we get with the interaction data stored in FEST xml file.
We also double check our drug-drug interaction result by using the interaction analysis tools offered by www.felleskatalogen.no and www.interaskjoner.no .

4 Discussion

The main objective of this thesis is to provide a standardized RESTful service interface for the Norwegian medicines registry (Forskrivnings- og ekspedisjonsstøtte FEST). According to an assessment study of international standards published by the Norwegian Directorate of eHealth (NDE) [10], Fast Healthcare Interoperability Resources (HL7 FHIR) is the recommended international standard framework for a service-oriented architecture, based on service interfaces. In addition, the base resources of HL7 FHIR can be adapted for national requirements. As a result, we decided to follow HL7 FHIR specifications as the chosen standard framework.

As the amount of data stored in the FEST XML file is huge, we used the Java Architecture for XML Binding (JAXB) technique to efficiently parse all FEST XML data elements and to automatically map the FEST information model into a Java classes model which is compliant to all FEST predefined complex types in schema files. At this very stage, we managed to represent FEST data elements in terms of Java objects where each FEST data element was automatically mapped to an instance of its corresponding Java class.

The most important advantage of our implemented RESTful-based architecture is that the client side of our FHIR RESTful server can be simply a web browser, while the business logic to handle requests about FEST data and return results, completely resides on the server side. In addition, the user of our system is neither required to download a local copy of FEST nor required to implement an update procedure to retain the latest version of FEST. Consequently, using our implemented solution will save the Norwegian health sector a lot of resources and efforts in general. In other terms, our architecture offers a common national up-to-date source of FEST data following the international HL7 FHIR standards and at the same time is easily accessible through HTTP protocol.

In contrast to our FEST on FHIR RESTful-based architecture, the current architecture of using FEST provided by -Fellekatalogen does not follow the HL7 FHIR international standard recommended for the future NHN reference architecture. Even though there are client solutions of the Felleskatalogen for local installation and access through a Web browser, obtaining the latest version of FEST still must be handled [36].

In the FHIR profiles that we created to represent FEST information, we added extensions to represent information such as the active substance name, and the product name of the medicine. We managed to explicitly add new search parameters for these extensions to our FHIR RESTful server. These new search parameters are implemented in line with HL7 FHIR standard mechanisms. Consequently, the user of our solution can use them in search queries along with the distinct parameters that are implicitly offered by FHIR RESTful server. Adding such search parameters provides our solution with the same search criteria as offered by Felleskatalogen which in turn adds a compatibility feature to our solution.

The components of the Web application we developed to demonstrate the practitioner use case, where a practitioner can send a request to our FHIR RESTful server to obtain medicines information and drug-drug interaction analysis, can be integrated in an electronic health record (EHR) systems and electronic patient journal (EPJ) systems following the guidelines of the future HL7-based reference architecture. For example, the practitioner can run drug-drug interaction analysis between the active medication statements taken by the patient and the new medication to be prescribed. The practitioner is free to search by the ATC-code, medicine product name, or the active substance name. In addition, the practitioner can obtain all information about a specific medicine such as short dosage, manufacturer, and usage guidance by using the same search criteria we mentioned above.

Another aspect of the work of this thesis is that FEST drug-drug interaction detection mechanism is implemented on our FHIR RESTful server side, where we mimic the business logic of FEST interaction analysis and represent interaction data by using group resource instances. In this context, we had to overcome a real implementation challenge where we could not use the FHIR DetectedIssue resource. The reason behind

this constraint is that DetectedIssue resource is designed to represent a detected drug-drug interaction related to a specific patient. On the other hand, our implementation of FEST interaction must represent drug-drug interaction in general without referring to any specific patient.
Consequently, using the DetectedIssue resource in such context would be violating the Object Oriented Design (OOD) concept as there is no one-to-one relation between FEST interaction class and FHIR DetectedIssue class.

By using FHIR group resource instead of DetectedIssue, we managed to overcome this problem. In the FEST Interaction class, drug-drug interaction is represented by two lists of medicines where one list includes medicines that interact with the medicines in the other list. We implemented this concept by adding the interaction details as extensions to the Group resource instance and representing one list of medicines as members of this Group resource instance. Then we repeated the same procedure for the second list of medicines. As a result, one FEST interaction instance is represented by two Group resource instances. The trade-off of this method is that we had to add interaction details (id, severity, clinical consequences, and guidance) in both Group instances.

I have participated in a project called "Mapping FHIR Resources to Ontology for DDI reasoning". We published a paper [59] with the same title in the 15th Scandinavian Conferecne on Health Informatics SHI2017, where we got the Best Paper Award. We utilized and customized the unofficial FHIR draft ontology in order to represent drug-drug interactions reasoning. The main purpose of the paper's work was to enhance the capabilities of FEST interactions. We used a FHIR client to store some FEST information on a FHIR server along with patient information. Then, we extracted and translated some of this information into Web Ontology Language (OWL) based ontology and applied drug-drug interaction reasoning mechanisms to expose any potential health risks implicitly caused by the interactions derived from FEST. The work of this thesis extends this paper's work by automatically, not manually, mapping all FEST data; interactions are no exception. In addition, the results of interactions reasoning obtained from the paper's project can be stored in our HAPI-FHIR server in terms of DetectedIssue resource instances, i.e. the potential risks can be easily exchanged as well.

The details in the system implementation part of this thesis reflect the search efforts and challenges I have overcome during my implementation journey. One of the main challenges was the lack of HAPI-FHIR DSTU 3 implementation documentation and guidelines. I had to go through the source code of HAPI-FHIR Java library in order to understand, test, and implement. The implementation of this thesis took long time to get an outcome that is reliable and valid.

The outcome of this solution is that FEST data can be automatically mapped and become available on a FHIR RESTful server. That means various health information systems can get access to one common national source of information about drugs which is complying with the international FHIR standard, up-to-date, and quality assured. Consequently, the interoperability between the various Norwegian health information systems can be improved which in turn a fundamental purpose of any ehealth project, i.e. a better patient's care quality.

5 Conclusion and future work

5.1 Conclusion

The Norwegian Electronic Prescription Support System (FEST) includes all information about drugs that can be prescribed electronically in Norway. FEST also is used today by Norwegian hospitals in the four health regions, where each region implements and maintains its own local solutions on a regular basis to structure and quality assure the data offered by FEST. As a result, the interoperability between these various medication system is negatively affected. In addition, users of FEST have two options to get an up-to-date version of FEST. They should use a solution that can automatically search for updates every 3 nights or manually download FEST from the Norwegian Medicines Agency (SLV) website before the 1st and 15th days of each month.

Our proposed solution is to automatically map FEST data into Fast Healthcare Interoperability Resources (HL7 FHIR standard) where drug information will become available and accessible through a FHIR RESTful-based service.

The result of our solution is that the latest version of FEST data can be represented in terms of HL7 FHIR international standards and is always available/accessible through FHIR RESTful-based service. We developed a Web application to verify and demonstrate the proof of concept of our solution result.

By using this solution, there will be no need to implement proprietary local solutions in order to structure, and quality assure FEST data because it already complies to HL7 FHIR international standards. In addition, the latest version of FEST will be always available and accessible through the proposed FHIR RESTful-based service. In other terms, when SLV updates the FEST XML file, they can use our solution to automatically map FEST XML into FHIR resources which are instantly accessible through the RESTful-based service.

5.2 Future work

5.2.1 FEST Administrator Dashboard

Develop a FEST administrator dashboard where the process of automatic mapping of FEST XML file to FHIR resources can be triggered by a click on an option displayed by the Web application.
The following diagram illustrates the execution workflow.

Step#	Description	Tool/Software
1	FEST technician clicks a button on FEST update web application to map the latest and up-to-date FEST xml file to FHIR resources.	Java/Eclipse
2	Parsing and creating FEST Java model. The same steps taken before.	Java/Eclipse
3	We have now the latest and up-to-date FEST on FHIR available online.	Java/Eclipse

Table 25: FEST Administrator Dashboard execution workflow

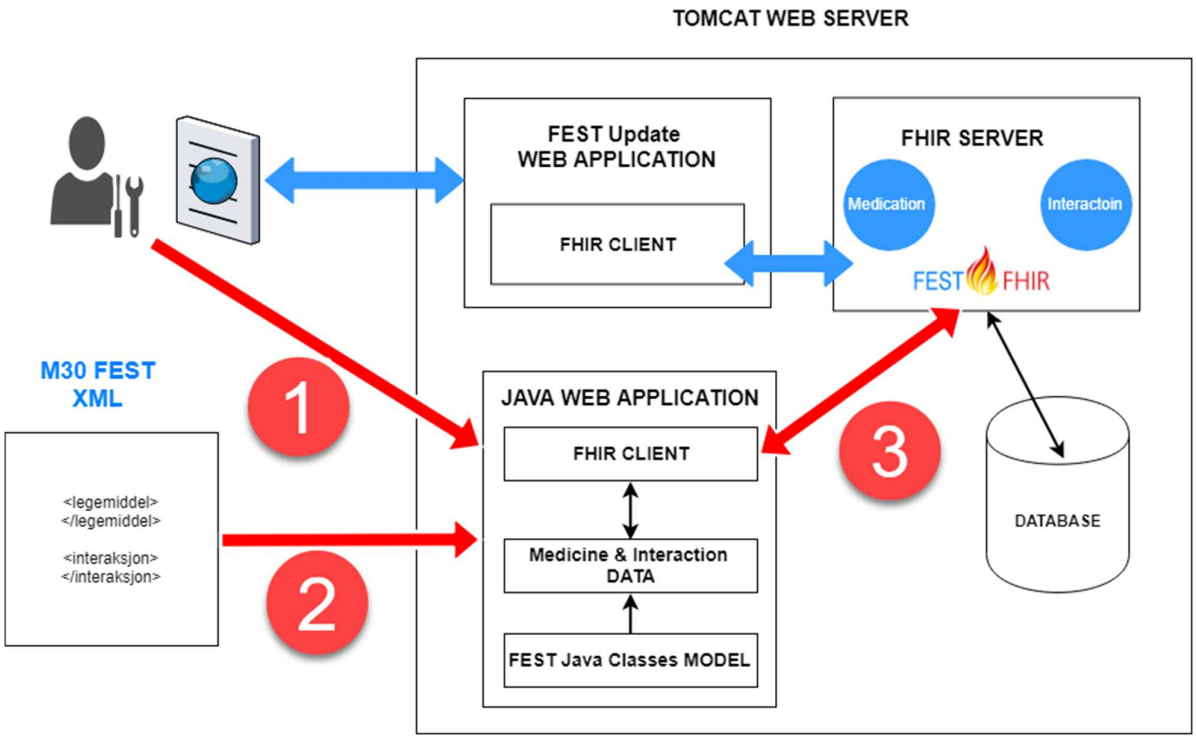


Figure 5.1: FEST administrator dashboard execution workflow

6 References

[1] H. I. E. (HIE), “Standards & Interoperability,” HealthIT.gov, 27 February 2014. [Online]. Available: https://www.healthit.gov/providers-professionals/standards-interoperability. [Accessed 24 March 2018].

[2] W. H. Organization, “The Fifty-eighth World Health Assembly,” 25 May 2005. [Online]. Available: http://www.who.int/healthacademy/media/WHA58-28-en.pdf. [Accessed 23 March 2018].

[3] D. f. e-helse, “E-helse,” Direktorate for e-helse, 6 June 2017. [Online]. Available: https://ehelse.no/e-helse. [Accessed 23 March 2018].

[4] S. Kontor, “Hospitals,” Regjeringen.no, [Online]. Available: https://www.regjeringen.no/en/topics/health-and-care/hospitals/id10935/. [Accessed 23 March 2018].

[5] S. Kontor, “Slik er spesialisthelsetjenesten bygd opp,” Kontor, Statsministerens, 24 November 2014. [Online]. Available: https://www.regjeringen.no/no/tema/helse-og-omsorg/sykehus/innsikt/nokkeltall-og-fakta---ny/slik--er-spesialisthelsetjenesten-bygd-o/id528748/. [Accessed 23 March 2018].

[6] D. f. e-helse, “Nasjonal journalløsning for kommunal helse- og omsorgstjeneste,” Direktorate for e-helse, 9 March 2018. [Online]. Available: https://ehelse.no/strategi/n-innbygger-n-journal/nasjonal-journallosning-for-kommunal-helse-og-omsorgstjeneste. [Accessed 23 March 2018].

[7] S. legemiddelverk, “FEST - informasjon om alt du kan få på resept i Noreg,” Statens legemiddelverk, 07 June 2016. [Online]. Available: https://legemiddelverket.no/andre-temaer/fest/fest-informasjon-om-alt-du-kan-fa-pa-resept-i-noreg. [Accessed 04 April 2018].

[8] S. legemiddelverk, “Nedlasting av FEST,” Statens legemiddelverk, 23 June 2017. [Online]. Available: https://legemiddelverket.no/andre-temaer/fest/nedlasting-av-fest. [Accessed 04 April 2018].

[9] D. f. e-helse, “Forbedre legemiddelinformasjon og FEST,” 13 March 2018. [Online]. Available: https://ehelse.no/nasjonale-prosjekter/forbedre-legemiddelinformasjon-og-fest.

[10] D. f. e-helse, “IE-1005 Vurdering av internasjonale standarder Nasjonal koordinering av standarder,” Direktorate for e-helse, Oslo, 2016.

[11] IHE International, Inc., “IHE Technical Frameworks,” 21 July 2017. [Online]. Available: http://www.ihe.net/uploadedFiles/Documents/ITI/IHE_ITI_TF_Vol4.pdf. [Accessed 02 April 2018].

[12] openEHR Foundation, “What is openEHR?,” openEHR Foundation, 2018. [Online]. Available: https://www.openehr.org/what_is_openehr. [Accessed 2 April 2018].

[13] H. L. S. INTERNATIONAL, “HL7 Standards Product Brief- CDA® Release 2,” [Online]. Available: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. [Accessed 02 April 2018].

[14] T. Berners-Lee, “Linked Data - Design issues,” www.w3.org, 18 June 2009. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed 02 April 2018].

[15] FHIR® – Fast Healthcare Interoperability Resources , “Summary - FHIR v3.0.1,” Health Level Seven® INTERNATIONAL, 2018. [Online]. Available: http://hl7.org/fhir/summary.html. [Accessed 02 April 2018].

[16] Nasjonal IKT, “Styringsdokument for 29.4 SAFEST planlegging,” NIKT, Oslo, 2016.

[17] S. legemiddelverk, “Femstjerners FEST – F5,” Statens legemiddelverk, 07 June 2016. [Online]. Available: https://legemiddelverket.no/andre-temaer/fest/femstjerners-fest-apne-lenkede-data. [Accessed 13 March 2018].

[18] DoseMeRx, “DoseMeRx - Worlds First Precision Dosing Software for Clinical Practice - DoseMeRx,” DoseMeRx, 2018. [Online]. Available: https://doseme-rx.com/introducing-dosemerx. [Accessed 06 April 2018].

[19] Alastair Allen, Kainos Evolve, “Partners in Interoperability & HL7 FHIR Applications Roundtable,” Kainos Evolve, 06 December 2017. [Online]. Available: http://www.hl7.org/events/fhir/roundtable/2017/12/presentations.cfm. [Accessed 06 April 2018].

[20] U. H. Network, “Introduction to HAPI FHIR,” University Health Network, 13 January 2018. [Online]. Available: http://hapifhir.io/doc_intro.html. [Accessed 13 March 2018].

[21] T. D. G. a. N. Terry, “The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions,” *Journal of Medical Internet Research,* vol. 7, p. 1, 2005.

[22] B. B, Architecture and tools for open, interoperable and portable EHRs., Nerlich M, Schaechinger U (Eds.): Integration of Health Telematics into Medical Practice, IOS Press, 2003.

[23] M. E., “Evolution and future of Electronic Health Records,” in *Int. Symposium on Health Informatics and Bioinformatic, Turkey*, Belek/Antalya-Turkey, 2005.

[24] T. Benson, Principles of Health Interoperability HL7 and SNOMED: Health Informatics, London: Springer, 2010.

[25] H. L. S. International, “About Health Level Seven® International,” Health Level Seven® International, 2007. [Online]. Available: http://www.hl7.org/about/. [Accessed 08 April 2018].

[26] H. L. S. International, “Introduction to HL7: Content,” Health Level Seven® International, 10 September 2006. [Online]. Available: http://wiki.hl7.org/index.php?title=Introduction_to_HL7:_Content. [Accessed 08 April 2018].

[27] H. L. S. International, “Related Organizations,” Health Level Seven International, 2018. [Online]. Available: http://www.hl7.org/about/relatedorganizations.cfm?ref=nav. [Accessed 08 April 2018].

[28] Dave Shaver , Corepoint Health., “The HL7 Evolution,” 2010. [Online]. Available: https://corepointhealth.com/wp-content/uploads/hl7-v2-v3-evolution.pdf. [Accessed 10 April 2018].

[29] H. L. S. International, “HL7 Standards Product Brief - HL7 EHR-System ePrescribing Functional Profile, Release 1,” Health Level Seven® International, 2018. [Online]. Available: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=368. [Accessed 10 April 2018].

[30] e. Initiative, “Electronic Prescribing:Toward Maximum Value and Rapid Adoption,” eHealth Initiative , Washington, D.C., 2004.

[31] K. A. S. V. N. S. J. A. T. J. R. H. Alexander Dobrev, “The conceptual framework of interoperable electronic health record and ePrescribing systems,” European Commission, Information Society and Media, Bonn, Germany, 2008.

[32] H. L. S. R. 3. (STU and v3.0.1-11917), “Summary - FHIR v3.0.1,” Health Level Seven®FHIR Release 3 (STU; v3.0.1-11917), 17 April 2017. [Online]. Available: https://www.hl7.org/fhir/summary.html. [Accessed 14 April 2018].

[33] E. M. A. EMA, “Introduction to ISO Identification of Medicinal Products, SPOR programme,” European Medicines Agency, London, 2016.

[34] N. M. Agency, “FEST Implementation guide,” Norwegian Medicines Agency, Oslo, 2016.

[35] T. T. R. Espen Stranger Seland, “eResept-M30 FEST og Forskrvning. Informasjonsmodell og XML meldingsbeskrivelse,” KITH INFORMASJONSTEKNOLOGI FOR HELSE OG VELFERD, Trondheim, 2014.

[36] ©. F. AS, “Medisin - Felleskatalogen,” © Felleskatalogen AS, [Online]. Available: https://www.felleskatalogen.no/medisin/interaksjon. [Accessed 15 April 2018].

[37] O. N. WHO Collaborating Centre for Drug Statistics Methodology, “International language for drug utilization research,” WHO Collaborating Centre for Drug Statistics Methodology, Oslo, Norway, 23 Februrary 2018. [Online]. Available: https://www.whocc.no. [Accessed 15 April 2018].

[38] N. I. o. P. H. WHO Collaborating Centre for Drug Statistics Methodology, “WHOCC- Structure and principles,” WHO Collaborating Centre for Drug Statistics Methodology, Norwegian Institute of Public Health, 15 February 2018. [Online]. Available: https://www.whocc.no/atc/structure_and_principles/. [Accessed 15 April 2018].

[39] N. IKT, “SAFEST- Nasjonal IKT,” Nasjonal IKT, 22 June 2017. [Online]. Available: https://nasjonalikt.no/prosjekter/safest. [Accessed 17 April 2018].

[40] Nasjonal IKT, “Projektforslag for SAFEST detaljeringsprojekt,” Nasjonal IKT, Statens Legemiddelverk, Oslo, 2017.

[41] B. A. L. A.-M. O. Aina Blix Bjelde, “Veiledning i god praksis for bruk av kjernejournal,” Direktorate for e-helse, Oslo, 2018.

[42] Direktorate for e-helse, “Kjernejournal - Kritisk informasjon IE-1006,” Direktorate for e-helse, Oslo, 2017.

[43] Norsk HL7 FHIR profil, “Norsk HL7 FHIR profil,” HL7 Norge, 2017. [Online]. Available: https://www.hl7.no/index.php/standarder/hl7-fhir-norsk. [Accessed 17 April 2018].

[44] DIPS, “En av verdens første FHIR-implementasjoner,” DIPS, 28 September 2015. [Online]. Available: https://www.dips.com/no/en-av-verdens-forste-fhir-implementasjoner. [Accessed 17 April 2018].

[45] Health Level Seven® INTERNATIONAL, “Summary - FHIR V3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/summary.html. [Accessed 19 April 2018].

[46] Health Level Seven® INTERNATIONAL, “INDEX- FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/. [Accessed 19 April 2018].

[47] Health Level Seven® INTERNATIONAL, “Datatypes - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/datatypes.html. [Accessed 20 April 2018].

[48] Health Level Seven® INTERNATIONAL, “Overview-arch FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/overview-arch.html. [Accessed 20 April 2018].

[49] Health Level Seven® INTERNATIONAL, “HTTP - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/http.html. [Accessed 20 April 2018].

[50] Health Level Seven® INTERNATIONAL, “Overview-dev - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/overview-dev.html. [Accessed 20 April 2018].

[51] Health Level Seven® INTERNATIONAL, “Resource - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/resource.html. [Accessed 21 April 2018].

[52] Health Level Seven® INTERNATIONAL, “Profiling - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/profiling.html. [Accessed 22 April 2018].

[53] Health Level Seven® INTERNATIONAL, “Overview - clinical - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: http://hl7.org/fhir/overview-clinical.html. [Accessed 22 April 2018].

[54] Health Level Seven® INTERNATIONAL, “Extensions - Prescriber - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/extension-medicationstatement-prescriber.html. [Accessed 22 April 2018].

[55] Health Level Seven® INTERNATIONAL, “Extension - doseType - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/extension-pharmacy-core-dosetype.html. [Accessed 22 April 2018].

[56] Health Level Seven® INTERNATIONAL, “Terminology-module - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/terminology-module.html. [Accessed 22 April 2018].

[57] Health Level Seven® INTERNATIONAL, “Codesystem example - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/codesystem-example.html. [Accessed 22 April 2018].

[58] Health Level Seven® INTERNATIONAL, “Datatypes - examples - FHIR v3.0.1,” ®© HL7.org 2011+, 19 April 2017. [Online]. Available: https://www.hl7.org/fhir/datatypes-examples.html. [Accessed 22 April 2018].

[59] R. A. G. A. a. J. P. N. Islam Fathi Hussein Al Khaldi, “Mapping FHIR Resources to Ontology for DDI reasoning,” in *Scandinavian Conference on Health Informatics SHI2017*, Kristiansand, Norway, 2017.

[60] M. Rutten, “Forge - FHIR,” Firely, 22 May 2018. [Online]. Available: https://fire.ly/forge/. [Accessed 22 May 2018].

[61] Apache Software Foundation, “Maven - Welcome to Apache Maven,” Apache Software Foundation, 2018. [Online]. Available: https://maven.apache.org. [Accessed 22 May 2018].

[62] Apache Software Foundation, “Apache Tomcat - Welcome,” Apache Software Foundation, 2018. [Online]. Available: http://tomcat.apache.org. [Accessed 24 May 2018].

[63] Health Level Seven International, “HL7 Standards Product Brief- HL7 Version 3 Product Suite,” Health Level Seven International, [Online]. Available: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186. [Accessed 02 April 2018].

[64] University Health Network, “HAPI-FHIR - The open source FHIR API for java,” University Health Network, 2018. [Online]. Available: http://hapifhir.io. [Accessed 25 April 2018].

7 Appendices

7.1 Appendix A: Java Application Code

```
1.  package no.uia.ehealth.MappingFESTXmlModelToJavaModel;
2.  import java.io.BufferedInputStream;
3.  import java.io.File;
4.  import java.io.FileInputStream;
5.  import java.io.FileNotFoundException;
6.  import java.util.ArrayList;
7.  import java.util.List;
8.  import javax.xml.bind.JAXBContext;
9.  import javax.xml.bind.JAXBElement;
10. import javax.xml.bind.JAXBException;
11. import javax.xml.bind.Marshaller;
12. import javax.xml.bind.Unmarshaller;
13. import ca.uhn.fhir.context.FhirContext;
14. import no.uia.ehealth.FHIRInformationModel.InteractionFHIRInformation;
15. import no.uia.ehealth.FHIRInformationModel.MedicationFHIRInformation;
16. import no.uia.ehealth.FESTonHapiFHIRController.*;
17. import no.uia.ehealth.workflow.*;
18. public class AutomaticMappingFESTOnFHIR {
19. private static final String XML  FILE = "fest251.xml";
20. private static List < MedicationFHIRInformation > medicationFHIRInformationList = new ArrayList < MedicationFHIRInformat
    ion > ();
21. private static List < InteractionFHIRInformation > interactionFHIRInformationList = new ArrayList < InteractionFHIRInformati
    on > ();
22.    private static final String baseURL = "http://localhost:8081/fest/baseDstu3";
23.    private static FhirContext ctx = FhirContext.forDstu3();
24.    public static void main(String[] args) throws JAXBException, FileNotFoundException {
25.       JAXBContext context = JAXBContext.newInstance("no.uia.ehealth.workflow");
26.       Unmarshaller um = context.createUnmarshaller();
27.       BufferedInputStream bis = new BufferedInputStream(new FileInputStream(new File(XML  FILE)));
28.       try {
29.          FEST fest = (FEST) um.unmarshal(bis);
30.          if (fest != null) {
31.             LegemiddelMerkevareCatalogueMapper lm = new LegemiddelMerkevareCatalogueMapper(fest);
32.             medicationFHIRInformationList = lm.getMedicationFHIRInformationList();
33.             InteraksjonCatalogueMapper im = new InteraksjonCatalogueMapper(fest);
34.             interactionFHIRInformationList = im.getInteractionFHIRInformationListt();
35.             if (medicationFHIRInformationList != null && medicationFHIRInformationList.size() != 0) {
36.                FESTMedicationHapiFHIRController fESTMedicationHapiFHIRController = new FESTMedicationHapiFHIRContro
    ller(ctx, medicationFHIRInformationList, baseURL);
37.             }
38.             if (interactionFHIRInformationList != null && interactionFHIRInformationList.size() != 0) {
39.                FESTInteractionHapiFHIRController fESTInteractionHapiFHIRController = new FESTInteractionHapiFHIRControll
    er(ctx, interactionFHIRInformationList, baseURL);
40.             }
41.          }
42.       } finally {}
43.    }
44. }
```

```
1.  package no.uia.ehealth.MappingFESTXmlModelToJavaModel;
2.  import java.util.ArrayList;
3.  import java.util.List;
4.  import no.uia.ehealth.FHIRInformationModel.ActiveSubstanceFHIRInformation;
5.  import no.uia.ehealth.FHIRInformationModel.MedicationFHIRInformation;
```

```
6.  import no.uia.ehealth.FHIRInformationModel.SubstanceGroupFHIRInformation;
7.  import no.uia.ehealth.workflow.*;
8.  import no.uia.ehealth.workflow.KatLegemiddelMerkevare.OppfLegemiddelMerkevare;
9.  public class LegemiddelMerkevareCatalogueMapper {
10.    protected List < ActiveSubstanceFHIRInformation > virkestoffListFHIR = new ArrayList < ActiveSubstanceFHIRInformation > ();
11.    protected List < MedicationFHIRInformation > medicationFHIRInformationList = new ArrayList < MedicationFHIRInformation > ();
12.    public LegemiddelMerkevareCatalogueMapper(FEST fest) {
13.       List < OppfLegemiddelMerkevare > oppfLegemiddelMerkevareList = new ArrayList < OppfLegemiddelMerkevare > ();
14.       oppfLegemiddelMerkevareList = fest.getKatLegemiddelMerkevare().getOppfLegemiddelMerkevare();
15.       for (OppfLegemiddelMerkevare element: oppfLegemiddelMerkevareList) {
16.          LegemiddelMerkevare medicine = element.getLegemiddelMerkevare();
17.          MedicationFHIRInformation medicationFHIR = new MedicationFHIRInformation(); // gathering required FHIR information for each medicine in FEST
18.          if (medicine.getId() != null && !medicine.getId().isEmpty()) medicationFHIR.setLegemiddelID(medicine.getId());
19.          if (medicine.getVarenavn() != null && !medicine.getVarenavn().isEmpty()) medicationFHIR.setLegemiddelVarenavn(medicine.getVarenavn());
20.          if (medicine.getAtc() != null) {
21.             if (medicine.getAtc().getV() != null && !medicine.getAtc().getV().isEmpty()) medicationFHIR.setatcV(medicine.getAtc().getV());
22.             if (medicine.getAtc().getS() != null && !medicine.getAtc().getS().isEmpty()) medicationFHIR.setatcS(medicine.getAtc().getS());
23.             if (medicine.getAtc().getDN() != null && !medicine.getAtc().getDN().isEmpty()) medicationFHIR.setatcDN(medicine.getAtc().getDN());
24.          }
25.          if (medicine.getTypeSoknadSlv() != null) {
26.             if (medicine.getTypeSoknadSlv().getV() != null && !medicine.getTypeSoknadSlv().getV().isEmpty()) medicationFHIR.setLegemiddelTypeSoknadV(medicine.getTypeSoknadSlv().getV());
27.          }
28.          if (medicine.getNavnFormStyrke() != null && !medicine.getNavnFormStyrke().isEmpty()) medicationFHIR.setLegemiddelNavnFormStyrke(medicine.getNavnFormStyrke());
29.          if (medicine.getProduktInfo() != null) {
30.             if (medicine.getProduktInfo().getProdusent() != null && !medicine.getProduktInfo().getProdusent().isEmpty()) medicationFHIR.setLegemiddelProdusent(medicine.getProduktInfo().getProdusent());
31.             if (medicine.getProduktInfo().getVarseltrekant() != null) medicationFHIR.setLegemiddelVarselTrekant(medicine.getProduktInfo().getVarseltrekant());
32.          }
33.          if (medicine.getAdministreringLegemiddel() != null) {
34.             if (medicine.getAdministreringLegemiddel().getAdministrasjonsvei().size() != 0 && !medicine.getAdministreringLegemiddel().getAdministrasjonsvei().get(0).getDN().isEmpty()) medicationFHIR.setLegemiddelAdministrasjonsveiDN(medicine.getAdministreringLegemiddel().getAdministrasjonsvei().get(0).getDN());
35.             if (medicine.getAdministreringLegemiddel().getEnhetDosering().size() != 0 && !medicine.getAdministreringLegemiddel().getEnhetDosering().get(0).getDN().isEmpty()) medicationFHIR.setLegemiddelEnhetsdoseringDN(medicine.getAdministreringLegemiddel().getEnhetDosering().get(0).getDN());
36.             if (medicine.getAdministreringLegemiddel().getKortdose().size() != 0 && !medicine.getAdministreringLegemiddel().getKortdose().get(0).getDN().isEmpty()) medicationFHIR.setLlegemiddelKortDoseDN(medicine.getAdministreringLegemiddel().getKortdose().get(0).getDN());
37.          } // logic to get the related active substances of this medicine.
38.          if ((medicine.getSortertVirkestoffMedStyrke() != null) && (!medicine.getSortertVirkestoffMedStyrke().isEmpty())) {
39.             VirkestoffMedStyrke e = (VirkestoffMedStyrke) medicine.getSortertVirkestoffMedStyrke().get(0).getRefVirkestoffMedStyrke();
40.             Virkestoff v = (Virkestoff) e.getRefVirkestoff();
41.             String id = v.getId();
42.             String navn = v.getNavn();
43.             String navnEngelsk = v.getNavnEngelsk();
44.             ActiveSubstanceFHIRInformation n = new ActiveSubstanceFHIRInformation(id, navn, navnEngelsk);
45.             medicationFHIR.setActiveSubstanceFHIRInformation(n);
46.          }
47.          medicationFHIRInformationList.add(medicationFHIR);
48.       }
49.    }
50.    public List < MedicationFHIRInformation > getMedicationFHIRInformationList() {
51.       return medicationFHIRInformationList;
52.    }
53. }
```

```
1.  package no.uia.ehealth.MappingFESTXmlModelToJavaModel;
2.  import java.util.ArrayList;
3.  import java.util.List;
4.  import no.uia.ehealth.FHIRInformationModel.InteractionFHIRInformation;
5.  import no.uia.ehealth.FHIRInformationModel.SubstanceFHIRInformation;
6.  import no.uia.ehealth.FHIRInformationModel.SubstanceGroupFHIRInformation;
7.  import no.uia.ehealth.workflow.*;
8.  import no.uia.ehealth.workflow.KatInteraksjon.OppfInteraksjon;
9.  public class InteraksjonCatalogueMapper {
10. protected List < InteractionFHIRInformation > interactionFHIRInformationList = new ArrayList < InteractionFHIRInformation >
    ();
11.   public InteraksjonCatalogueMapper(FEST fest) {
12.     List < OppfInteraksjon > oppfInteraksjonList = new ArrayList < OppfInteraksjon > ();
13.     oppfInteraksjonList = fest.getKatInteraksjon().getOppfInteraksjon();
14.     for (OppfInteraksjon element: oppfInteraksjonList) {
15.       if (element != null) {
16.         if (element.getInteraksjon() != null) {
17.           Interaksjon interaction = element.getInteraksjon();
18.           InteractionFHIRInformation interactionFHIR = new InteractionFHIRInformation(); // gathering required FHIR inform
    ation for each interaction in FEST
19.           if (interaction.getId() != null && !interaction.getId().isEmpty()) interactionFHIR.setInteractionFHIRID(interaction.ge
    tId());
20.           if (interaction.getRelevans() != null) {
21.             if (interaction.getRelevans().getV() != null && !interaction.getRelevans().getV().isEmpty()) interactionFHIR.setIn
    teractionFHIRRelevansV(interaction.getRelevans().getV());
22.             if (interaction.getRelevans().getDN() != null && !interaction.getRelevans().getDN().isEmpty()) interactionFHIR.se
    tInteractionFHIRRelevansDN(interaction.getRelevans().getDN());
23.             interactionFHIR.setInteractionFHIRRelevansDN(interaction.getRelevans().getDN());
24.           }
25.           if (interaction.getKliniskKonsekvens() != null && !interaction.getKliniskKonsekvens().isEmpty()) interactionFHIR.s
    etInteractionFHIRKliniskKonsekvens(interaction.getKliniskKonsekvens());
26.           if (interaction.getHandtering() != null && !interaction.getHandtering().isEmpty()) interactionFHIR.setInteractionFHI
    RHandtering(interaction.getHandtering()); // gathering required FHIR information for all substance groups in each // interaction
27.           List < SubstanceGroupFHIRInformation > substanceGroupFHIRInformationList = new ArrayList < SubstanceGroupF
    HIRInformation > ();
28.           List < Substansgruppe > interaksjonSubstansgruppeList = new ArrayList < Substansgruppe > ();
29.           if (interaction.getSubstansgruppe() != null) {
30.             interaksjonSubstansgruppeList = interaction.getSubstansgruppe();
31.             for (Substansgruppe sg: interaksjonSubstansgruppeList) { // gathering required FHIR information for each substanc
    e group in the current // interaction
32.               SubstanceGroupFHIRInformation substanceGroupFHIRInformation = new SubstanceGroupFHIRInformation();
33.               List < SubstanceFHIRInformation > substanceFHIRInformationList = new ArrayList < SubstanceFHIRInformati
    on > ();
34.               List < Substansgruppe.Substans > sgList = new ArrayList < Substansgruppe.Substans > ();
35.               if (sg.getSubstans() != null) {
36.                 sgList = sg.getSubstans();
37.                 for (Substansgruppe.Substans s: sgList) { // gathering required FHIR information for each substance in the curr
    ent group
38.                   SubstanceFHIRInformation sFHIR = new SubstanceFHIRInformation();
39.                   if (s.getSubstans() != null && !s.getSubstans().isEmpty()) sFHIR.setSubstansNavn(s.getSubstans());
40.                   if (s.getAtc() != null) {
41.                     if (s.getAtc().getV() != null && !s.getAtc().getV().isEmpty()) sFHIR.setatcV(s.getAtc().getV());
42.                     if (s.getAtc().getS() != null && !s.getAtc().getS().isEmpty()) sFHIR.setatcS(s.getAtc().getS());
43.                     if (s.getAtc().getDN() != null && !s.getAtc().getDN().isEmpty()) sFHIR.setatcDN(s.getAtc().getDN());
44.                   }
45.                   substanceFHIRInformationList.add(sFHIR);
46.                 }
47.               }
48.               substanceGroupFHIRInformation.setSubstanceFHIRInformation(substanceFHIRInformationList);
49.               substanceGroupFHIRInformationList.add(substanceGroupFHIRInformation);
50.             }
51.             interactionFHIR.setSubstanceGroupFHIRInformation(substanceGroupFHIRInformationList);
52.           }
```

```
53.                 interactionFHIRInformationList.add(interactionFHIR); // add the current interaction to list of // interactions of FHIR
    model
54.             }
55.         }
56.     }
57.   }
58.   public List < InteractionFHIRInformation > getInteractionFHIRInformationListt() {
59.     return interactionFHIRInformationList;
60.   }
61. }
```

```
1.  package no.uia.ehealth.FHIRInformationModel;
2.  import org.hl7.fhir.dstu3.model.StringType;
3.  import no.uia.ehealth.workflow.*;;
4.  public class ActiveSubstanceFHIRInformation extends Virkestoff {
5.    protected String activeSubstanceID;
6.    protected String activeSubstansNavn;
7.    protected String activeSubstansEngelskNavn;
8.    public String getActiveSubstansEngelskNavn() {
9.      if (activeSubstansEngelskNavn == null) {
10.       activeSubstansEngelskNavn = new String();
11.     }
12.     return activeSubstansEngelskNavn;
13.   }
14.   public void setActiveSubstansEngelskNavn(String activeSubstansEngelskNavn) {
15.     this.activeSubstansEngelskNavn = activeSubstansEngelskNavn;
16.   }
17.   public String getActiveSubstanceID() {
18.     if (activeSubstanceID == null) {
19.       activeSubstanceID = new String();
20.     }
21.     return activeSubstanceID;
22.   }
23.   public void setActiveSubstanceID(String activeSubstanceID) {
24.     this.activeSubstanceID = activeSubstanceID;
25.   }
26.   public String getActiveSubstansNavn() {
27.     if (activeSubstansNavn == null) {
28.       activeSubstansNavn = new String();
29.     }
30.     return activeSubstansNavn;
31.   }
32.   public void setActiveSubstansNavn(String value) {
33.     this.activeSubstansNavn = value;
34.   }
35.   public ActiveSubstanceFHIRInformation(String id, String navn, String navnEngelsk) {
36.     this.activeSubstanceID = id;
37.     this.activeSubstansNavn = navn;
38.     this.activeSubstansEngelskNavn = navnEngelsk;
39.   }
40. }
```

```
1.  package no.uia.ehealth.FHIRInformationModel;
2.  import no.uia.ehealth.workflow.*;
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  public class InteractionFHIRInformation extends Interaksjon {
6.    protected String interactionFHIRID;
7.    protected String interactionFHIRRelevansV;
8.    protected String interactionFHIRRelevansDN;
9.    protected String interactionFHIRKliniskKonsekvens;
10.   protected String interactionFHIRHandtering;
11.   protected List < SubstanceGroupFHIRInformation > interactionFHIRSubstanceGroupList;
12.   public String getInteractionFHIRID() {
13.     if (interactionFHIRID == null) {
```

```
            interactionFHIRID = new String();
        }
        return interactionFHIRID;
    }
    public void setInteractionFHIRID(String value) // this.id
        {
            this.interactionFHIRID = value;
        }
    public String getInteractionFHIRRelevansV() {
        if (interactionFHIRRelevansV == null) {
            interactionFHIRRelevansV = new String();
        }
        return interactionFHIRRelevansV;
    }
    public void setIinteractionFHIRRelevansV(String value) // this.relevans.getV()
        {
            this.interactionFHIRRelevansV = value;
        }
    public String getInteractionFHIRRelevansDN() {
        if (interactionFHIRRelevansDN == null) {
            interactionFHIRRelevansDN = new String();
        }
        return interactionFHIRRelevansDN;
    }
    public void setInteractionFHIRRelevansDN(String value) // this.relevans.getDN()
        {
            this.interactionFHIRRelevansDN = value;
        }
    public String getInteractionFHIRKliniskKonsekvens() {
        if (interactionFHIRKliniskKonsekvens == null) {
            interactionFHIRKliniskKonsekvens = new String();
        }
        return interactionFHIRKliniskKonsekvens;
    }
    public void setInteractionFHIRKliniskKonsekvens(String value) // this.kliniskKonsekvens
        {
            this.interactionFHIRKliniskKonsekvens = value;
        }
    public String getInteractionFHIRHandtering() {
        if (interactionFHIRHandtering == null) {
            interactionFHIRHandtering = new String();
        }
        return interactionFHIRHandtering;
    }
    public void setInteractionFHIRHandtering(String value) // this.handtering
        {
            this.interactionFHIRHandtering = value;
        }
    public List < SubstanceGroupFHIRInformation > getSubstanceGroupFHIRInformation() {
        if (interactionFHIRSubstanceGroupList == null) {
            interactionFHIRSubstanceGroupList = new ArrayList < SubstanceGroupFHIRInformation > ();
        }
        return this.interactionFHIRSubstanceGroupList;
    }
    public void setSubstanceGroupFHIRInformation(List < SubstanceGroupFHIRInformation > list) // this.substansgruppe
        {
            this.interactionFHIRSubstanceGroupList = list;
        }
}
```

```
package no.uia.ehealth.FHIRInformationModel;
import no.uia.ehealth.workflow.*;
import no.uia.ehealth.workflow.KatLegemiddelMerkevare.OppfLegemiddelMerkevare;
import java.util.ArrayList;
import java.util.List;
public class MedicationFHIRInformation extends LegemiddelMerkevare {
```

```
7.      protected String legemiddelID; // in Legemiddelmerkevare.id
8.      protected String legeMiddelVarenavn; // in Legemiddelmerkevare.varenavn, // OppfLegemiddelMerkevare.legemiddelmerkevare.varenavn ( can be used as // identifier in FHIR resource instance) // used together to identify a medicine, for example <Atc V="P01AB03" // S="2.16.578.1.12.4.1.1.7180" DN="Ornidazol" />
9.      protected String atcV; // OppfLegemiddelMerkevare.legemiddelmerkevare.atc v attribute ( atc code for // active substance)
10.     protected String atcS; // OppfLegemiddelMerkevare.legemiddelmerkevare.atc s attribute ( no display on // web app)
11.     protected String atcDN; // OppfLegemiddelMerkevare.legemiddelmerkevare.atc dn attribute ( name of active // substance)
12.     protected String legemiddelNavnFormStyrke; // in TypeLegemiddel get navnFormStyrke() method // //OppfLegemiddelMerkevare.legemiddelmerkevare.NavnFormStyrke
13.     protected String legemiddelProdusent; // in Legemiddelmerkevare getProduktInfo() method // // OppfLegemiddelMerkevare.legemiddelmerkevare.produktInfo.produsent
14.     protected boolean legemiddelVarselTrekant; // in Legemiddelmerkevare getProduktInfo() method // // OppfLegemiddelMerkevare.legemiddelmerkevare.produktInfo.varselTerkant // true/false
15.     protected String legemiddelAdministrasjonsveiDN; // in TypeLegemiddel getAdminstreringLegeMiddel() method// // OppfLegemiddelMerkevare.legemiddelmerkevare.Adminstrasjonvei // dn attribute
16.     protected String legemiddelEnhetsdoseringDN; // in TypeLegemiddel getAdminstreringLegeMiddel() method // //OppfLegemiddelMerkevare.legemiddelmerkevare.AdminstreringLAdminstreringLegeMiddel.Adminstrasjonvei.Enhetsdosering // dn attribute
17.     protected String legemiddelKortDoseDN; // in TypeLegemiddel getAdminstreringLegeMiddel() // method//OOppfLegemiddelMerkevare.legemiddelmerkevare.AdminstreringLegeMiddel.Adminstrasjonvei.Kortdose // dn attribute
18.     protected ActiveSubstanceFHIRInformation activeSubstanceFHIRInformation;
19.     protected String legemiddelTypeSoknadV; // in medicine.getTypeSoknadSlv().getV(), if value == 1, then isOverCounter // false

20.     public String getLegemiddelTypeSoknadV() {
21.         if (legemiddelTypeSoknadV == null) {
22.             this.legemiddelTypeSoknadV = legemiddelTypeSoknadV;
23.         }
24.         return legemiddelTypeSoknadV;
25.     }
26.     public void setLegemiddelTypeSoknadV(String legemiddelTypeSoknadV) {
27.         this.legemiddelTypeSoknadV = legemiddelTypeSoknadV;
28.     }
29.     public String getLegemiddelID() {
30.         if (legemiddelID == null) {
31.             legemiddelID = new String();
32.         }
33.         return legemiddelID;
34.     }
35.     public void setLegemiddelID(String value) // legemiddelMerkevare.id
36.         {
37.             this.legemiddelID = value;
38.         }
39.     public String getLegemiddelVarenavn() {
40.         if (legeMiddelVarenavn == null) {
41.             legeMiddelVarenavn = new String();
42.         }
43.         return legeMiddelVarenavn;
44.     }
45.     public void setLegemiddelVarenavn(String value) // legemiddelMerkevare.varenavn
46.         {
47.             this.legeMiddelVarenavn = value;
48.         }
49.     public String getATCV() {
50.         if (atcV == null) {
51.             atcV = new String();
52.         }
53.         return atcV;
54.     }
55.     public void setatcV(String value) // legemiddelMerkevare.atc.getV();
56.         {
57.             this.atcV = value;
58.         }
59.     public String getatcS() {
60.         if (atcS == null) {
61.             atcS = new String();
62.         }
63.         return atcS;
64.     }
```

```
65.    public void setatcS(String value) // legemiddelMerkevare.atc.getS();
66.       {
67.          this.atcS = value;
68.       }
69.    public String getatcDN() {
70.       if (atcDN == null) {
71.          atcDN = new String();
72.       }
73.       return atcDN;
74.    }
75.    public void setatcDN(String value) // legemiddelMerkevare.atc.getDN();
76.       {
77.          this.atcDN = value;
78.       }
79.    public String getLegemiddelNavnFormStyrke() {
80.       if (legemiddelNavnFormStyrke == null) {
81.          legemiddelNavnFormStyrke = new String();
82.       }
83.       return legemiddelNavnFormStyrke;
84.    }
85.    public void setLegemiddelNavnFormStyrke(String value) // legemiddelMerkevare.navnFormStyrke
86.       {
87.          this.legemiddelNavnFormStyrke = value;
88.       }
89.    public String getLegemiddelProdusent() {
90.       if (legemiddelProdusent == null) {
91.          legemiddelProdusent = new String();
92.       }
93.       return legemiddelProdusent;
94.    }
95.    public void setLegemiddelProdusent(String value) // legemiddelMerkevare.produktInfo.getProdusent()
96.       {
97.          this.legemiddelProdusent = value;
98.       }
99.    public Boolean getLegemiddelVarselTrekant() {
100.      return legemiddelVarselTrekant;
101.   }
102.   public void setLegemiddelVarselTrekant(Boolean value) // legemiddelMerkevare.produktInfo.getVarseltrekant()
103.      {
104.         this.legemiddelVarselTrekant = value;
105.      }
106.   public String getLegemiddelAdministrasjonsveiDN() {
107.      if (legemiddelAdministrasjonsveiDN == null) {
108.         legemiddelAdministrasjonsveiDN = new String();
109.      }
110.      return legemiddelAdministrasjonsveiDN;
111.   }
112.   public void setLegemiddelAdministrasjonsveiDN(String value) // legemiddelMerkevare.getAdministreringLegemiddel().getAd
      ministrasjonsvei().get(0).getDN() // or // legemiddelMerkevare.administreringLegemiddel.getAdministrasjonsvei().get(0).getDN()

113.      {
114.         this.legemiddelAdministrasjonsveiDN = value;
115.      }
116.   public String getLegemiddelEnhetsdoseringDN() {
117.      if (legemiddelEnhetsdoseringDN == null) {
118.         legemiddelEnhetsdoseringDN = new String();
119.      }
120.      return legemiddelEnhetsdoseringDN;
121.   }
122.   public void setLegemiddelEnhetsdoseringDN(String value) // legemiddelMerkevare.getAdministreringLegemiddel().getEnhetD
      osering().get(0).getDN() // or // legemiddelMerkevare.administreringLegemiddel.getEnhetDosering().get(0).getDN()
123.      {
124.         this.legemiddelEnhetsdoseringDN = value;
125.      }
126.   public String getLegemiddelKortDoseDN() {
127.      if (legemiddelKortDoseDN == null) {
128.         legemiddelKortDoseDN = new String();
```

```
129.        }
130.        return legemiddelKortDoseDN;
131.    }
132.    public void setLlegemiddelKortDoseDN(String value) // legemiddelMerkevare.administreringLegemiddel.getKortdose().get(0).
    getDN()
133.        {
134.            this.legemiddelKortDoseDN = value;
135.        }
136.    public ActiveSubstanceFHIRInformation getActiveSubstanceFHIRInformation() {
137.        if (activeSubstanceFHIRInformation == null) {
138.            activeSubstanceFHIRInformation = new ActiveSubstanceFHIRInformation("", "", "");
139.        }
140.        return activeSubstanceFHIRInformation;
141.    }
142.    public void setActiveSubstanceFHIRInformation(ActiveSubstanceFHIRInformation activeSubstanceFHIRInformation) {
143.        this.activeSubstanceFHIRInformation = activeSubstanceFHIRInformation;
144.    }
145. }
```

```
1.  package no.uia.ehealth.FHIRInformationModel;
2.  import no.uia.ehealth.workflow.*;
3.  import java.util.List;
4.  public class SubstanceFHIRInformation extends Substansgruppe.Substans {
5.      protected String substansNavn;
6.      protected String atcV;
7.      protected String atcS;
8.      protected String atcDN;
9.      public String getSubstansNavn() {
10.         if (substansNavn == null) {
11.             substansNavn = new String();
12.         }
13.         return substansNavn;
14.     }
15.     public void setSubstansNavn(String value) // this.substans
16.         {
17.             this.substansNavn = value;
18.         }
19.     public String getatcV() {
20.         if (atcV == null) {
21.             atcV = new String();
22.         }
23.         return atcV;
24.     }
25.     public void setatcV(String value) // this.atc.getV();
26.         {
27.             this.atcV = value;
28.         }
29.     public String getatcS() {
30.         if (atcS == null) {
31.             atcS = new String();
32.         }
33.         return atcS;
34.     }
35.     public void setatcS(String value) // this.atc.getS();
36.         {
37.             this.atcS = value;
38.         }
39.     public String getatcDN() {
40.         if (atcDN == null) {
41.             atcDN = new String();
42.         }
43.         return atcDN;
44.     }
45.     public void setatcDN(String value) // this.atc.getDN();
46.         {
47.             this.atcDN = value;
```

```
48.         }
49.     }
```

```
1.  package no.uia.ehealth.FHIRInformationModel;
2.  import no.uia.ehealth.workflow.*;
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  public class SubstanceGroupFHIRInformation extends Substansgruppe {
6.      protected List < SubstanceFHIRInformation > interactionFHIRSubstanceList;
7.      public List < SubstanceFHIRInformation > getSubstanceFHIRInformation() {
8.          if (interactionFHIRSubstanceList == null) {
9.              interactionFHIRSubstanceList = new ArrayList < SubstanceFHIRInformation > ();
10.         }
11.         return this.interactionFHIRSubstanceList;
12.     }
13.     public void setSubstanceFHIRInformation(List < SubstanceFHIRInformation > list) // this.substans
14.             {
15.         this.interactionFHIRSubstanceList = list;
16.     }
17. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRResources;
2.  import org.hl7.fhir.dstu3.model.*;
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  import ca.uhn.fhir.model.api.annotation.Child;
6.  import ca.uhn.fhir.model.api.annotation.Description;
7.  import ca.uhn.fhir.model.api.annotation.Extension;
8.  import ca.uhn.fhir.model.api.annotation.ResourceDef;
9.  import ca.uhn.fhir.util.ElementUtil;
10. import japa.parser.ast.type.ReferenceType;
11. /** * Definition class for adding extensions to the built-in Group resource type. */
12. @
13. ResourceDef(name = "Group", profile = "http://ehealth.uia.no/StructureDefinition/FESTGroup") public class FESTGroupResourc
    e extends Group {
14.     private static final long serialVersionUID = 1 L; // FESTGroup reference store information about a group of medicines which a
    re // part of drug-
    drug interaction // with another group. // The point is, to represent an interaction, two FESTGroup instances must have // the same i
    nteraction id // The structure of the reference reflects the structure of interaction // catalogue in FEST M30 2.5.1 // Interaction infor
    mation stored along with a group of this interaction // FESTGroup have the following elements which represents data elements fro
    m // FEST // Group.type : Medication // Group.actual: true which means the group refers to to a specific group of // real individuals
    /instances. // interactionID // Extensions start // Group.extension:interactionID
15.     @
16.     Child(name = "interactionID")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionID", definedLocally = tru
    e, isModifier = false)@ Description(shortDefinition = "The given interaction ID  in FEST M30 2.5.1") private StringType interact
    ionID; // interactionRelevansV // Group.extension:interactionRelevansV
17.     @
18.     Child(name = "interactionRelevansV")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionRelevansV", defi
    nedLocally = true, isModifier = false)@ Description(shortDefinition = "The severity level of the interaction") private StringType
    interactionRelevansV; // interactionRelevansDN // Group.extension:interactionRelevansDN
19.     @
20.     Child(name = "interactionRelevansDN")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionRelevansDN", d
    efinedLocally = true, isModifier = false)@ Description(shortDefinition = "Severity short description of the interaction") private S
    tringType interactionRelevansDN; // interactionKliniskKonsekvens // Group.extension:interactionKliniskKonsekvens
21.     @
22.     Child(name = "interactionKliniskKonsekvens")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionKliniskK
    onsekvens", definedLocally = true, isModifier = false)@ Description(shortDefinition = "Interaction clinical consequences.") priva
    te StringType interactionKliniskKonsekvens; // interactionID // Group.extension:interactionHandtering
23.     @
24.     Child(name = "interactionHandtering")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/interactionHandtering", defi
    nedLocally = true, isModifier = false)@ Description(shortDefinition = "Interaction guidance") private StringType interactionHan
    dtering; // Extensions end
25.     /**  * adding isEmpty() check for newly added extensions.    */
26.     @
27.     Override public boolean isEmpty() {
28.         return super.isEmpty() && ElementUtil.isEmpty(interactionID, interactionRelevansV, interactionRelevansDN, interactionK
    liniskKonsekvens, interactionHandtering);
```

```
    }
    public StringType getInteractionID() {
        if (interactionID == null) {}
        return interactionID;
    }
    public void setInteractionID(StringType interactionID) {
        this.interactionID = interactionID;
    }
    public StringType getInteractionRelevansV() {
        if (interactionRelevansV == null) {}
        return interactionRelevansV;
    }
    public void setInteractionRelevansV(StringType interactionRelevansV) {
        this.interactionRelevansV = interactionRelevansV;
    }
    public StringType getInteractionRelevansDN() {
        if (interactionRelevansDN == null) {}
        return interactionRelevansDN;
    }
    public void setInteractionRelevansDN(StringType interactionRelevansDN) {
        this.interactionRelevansDN = interactionRelevansDN;
    }
    public StringType getInteractionKliniskKonsekvens() {
        if (interactionKliniskKonsekvens == null) {}
        return interactionKliniskKonsekvens;
    }
    public void setInteractionKliniskKonsekvens(StringType interactionKliniskKonsekvens) {
        this.interactionKliniskKonsekvens = interactionKliniskKonsekvens;
    }
    public StringType getInteractionHandtering() {
        if (interactionHandtering == null) {}
        return interactionHandtering;
    }
    public void setInteractionHandtering(StringType interactionHandtering) {
        this.interactionHandtering = interactionHandtering;
    }
}
```

```
package no.uia.ehealth.FESTonHapiFHIRResources;
import org.hl7.fhir.dstu3.model.*;
import java.util.ArrayList;
import java.util.List;
import ca.uhn.fhir.model.api.annotation.Child;
import ca.uhn.fhir.model.api.annotation.Description;
import ca.uhn.fhir.model.api.annotation.Extension;
import ca.uhn.fhir.model.api.annotation.ResourceDef;
import ca.uhn.fhir.util.ElementUtil;
import japa.parser.ast.type.ReferenceType;
/** * Definition class for adding extensions to the built-in Medication resource * type. */
@
ResourceDef(name = "Medication", profile = "http://ehealth.uia.no/StructureDefinition/FESTMedication") public class FESTMedicationResource extends Medication {
    private static final long serialVersionUID = 1 L; // set code.coding.system coding system version code display values. // code.coding : http://www.whocc.no/atc // code.coding.system: http://www.whocc.no/atc // code.coding.version: 7180 // code.coding.code : ATC // display: get the ATC V attribute // Medication.manfacturer.display ( store value of produsent) // Medication.ingredient.isActive (true) // Slicing start // code.text.ATC-navn: get the ATC DN attribute
    @
    Child(name = "atcNavn")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/atcNavn", definedLocally = true, isModifier = false)@ Description(shortDefinition = "The name associated with the medicine ATC code, DN attribute of element ATC in FEST") private StringType atcNavn; // Medication.isOverTheCounter:legemiddelVarselTrekant
    @
    Child(name = "legemiddelVarselTrekant")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelVarselTrekant", definedLocally = true, isModifier = false)@ Description(shortDefinition = "Indicates a red triangle") private BooleanType legemiddelVarselTrekant; // Medication.ingredient.itemReference:itemReference.reference ( store a // reference to FESTSubstance)
```

```
19.     /*   * @Child(name="itemReference") *   * @Extension(url="http://ehealth.uia.no/StructureDefinition/itemReference",     * definedLocally=true, isModifier=false)    *   * @Description(shortDefinition="a reference to FESTSubstance") private  * FESTSubstanceResource itemReference;  */ // Slicing end // Extesnions start // Extension: leggemiddelFESTID
20.     @
21.     Child(name = "leggemiddelFESTID")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/LegemiddelFESTID", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel ID in FEST M30 2.5.1") private StringType legemiddelFESTID; // Extension: legemiddelVarenavn
22.     @
23.     Child(name = "legemiddelVarenavn")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelVarenavn", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel varenavn in FEST M30 2.5.1") private StringType legemiddelVarenavn; // Extension: legemiddelNavnFormStyrke
24.     @
25.     Child(name = "legemiddelNavnFormStyrke")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelNavnFormStyrke", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel navn form styrke in FEST M30 2.5.1") private StringType legemiddelNavnFormStyrke; // Extension: legemiddelAdministrasjonsvei lemiddelEnhetsdosering
26.     @
27.     Child(name = "legemiddelAdministrasjonsvei")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelAdministrasjonsvei", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel administration guidance in FEST M30 2.5.1") private StringType legemiddelAdministrasjonsvei; // Extension: lemiddelEnhetsdosering legemiddelKortDose
28.     @
29.     Child(name = "legemiddelEnhetsdosering")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelEnhetsdosering", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel unit dosage in FEST M30 2.5.1") private StringType legemiddelEnhetsdosering; // Extension: legemiddelKortDose
30.     @
31.     Child(name = "legemiddelKortDose")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/legemiddelKortDose", definedLocally = true, isModifier = false)@ Description(shortDefinition = "legemiddel short dose in FEST M30 2.5.1") private StringType legemiddelKortDose; // Extensions End
32.     /**  * adding isEmpty() check for newly added extensions.    */
33.     @
34.     Override public boolean isEmpty() { // return super.isEmpty() && // ElementUtil.isEmpty(atcNavn,legemiddelVarselTrekant, // legemiddelFESTID,itemReference,legemiddelVarenavn,legemiddelNavnFormStyrke,legemiddelAdministrasjonsvei,legemiddelEnhetsdosering,legemiddelKortDose);
35.         return super.isEmpty() && ElementUtil.isEmpty(atcNavn, legemiddelVarselTrekant, legemiddelFESTID, legemiddelVarenavn, legemiddelNavnFormStyrke, legemiddelAdministrasjonsvei, legemiddelEnhetsdosering, legemiddelKortDose);
36.     }
37.     public StringType getAtcNavn() {
38.         if (atcNavn == null) {
39.             atcNavn = new StringType();
40.         }
41.         return atcNavn;
42.     }
43.     public void setAtcNavn(StringType value) {
44.         this.atcNavn = value;
45.     }
46.     public BooleanType getLegemiddelVarselTrekant() {
47.         if (legemiddelVarselTrekant == null) {
48.             legemiddelVarselTrekant = new BooleanType();
49.         }
50.         return legemiddelVarselTrekant;
51.     }
52.     public void setLegemiddelVarselTrekant(BooleanType legemiddelVarselTrekant) {
53.             this.legemiddelVarselTrekant = legemiddelVarselTrekant;
54.         }
55.         /*   * public FESTSubstanceResource getItemReference() { if(itemReference == null) {     * itemReference = new FESTSubstanceResource(); } return itemReference; }    *   * public void setItemReference(FESTSubstanceResource itemReference) {   * this.itemReference = itemReference; }     */
56.     public StringType getLegemiddelFESTID() {
57.         if (legemiddelFESTID == null) {
58.             legemiddelFESTID = new StringType();
59.         }
60.         return legemiddelFESTID;
61.     }
62.     public void setLegemiddelFESTID(StringType legemiddelFESTID) {
63.         this.legemiddelFESTID = legemiddelFESTID;
64.     }
65.     public StringType getLegemiddelVarenavn() {
66.         if (legemiddelVarenavn == null) {
```

```
67.             legemiddelVarenavn = new StringType();
68.         }
69.         return legemiddelVarenavn;
70.     }
71.     public void setLegemiddelVarenavn(StringType legemiddelVarenavn) {
72.         this.legemiddelVarenavn = legemiddelVarenavn;
73.     }
74.     public StringType getLegemiddelNavnFormStyrke() {
75.         if (legemiddelNavnFormStyrke == null) {
76.             legemiddelNavnFormStyrke = new StringType();
77.         }
78.         return legemiddelNavnFormStyrke;
79.     }
80.     public void setLegemiddelNavnFormStyrke(StringType legemiddelNavnFormStyrke) {
81.         this.legemiddelNavnFormStyrke = legemiddelNavnFormStyrke;
82.     }
83.     public StringType getLegemiddelAdministrasjonsvei() {
84.         if (legemiddelAdministrasjonsvei == null) {
85.             legemiddelAdministrasjonsvei = new StringType();
86.         }
87.         return legemiddelAdministrasjonsvei;
88.     }
89.     public void setLegemiddelAdministrasjonsvei(StringType legemiddelAdministrasjonsvei) {
90.         this.legemiddelAdministrasjonsvei = legemiddelAdministrasjonsvei;
91.     }
92.     public StringType getLegemiddelEnhetsdosering() {
93.         if (legemiddelEnhetsdosering == null) {
94.             legemiddelEnhetsdosering = new StringType();
95.         }
96.         return legemiddelEnhetsdosering;
97.     }
98.     public void setLegemiddelEnhetsdosering(StringType legemiddelEnhetsdosering) {
99.         this.legemiddelEnhetsdosering = legemiddelEnhetsdosering;
100.    }
101.    public StringType getLegemiddelKortDose() {
102.        if (legemiddelKortDose == null) {
103.            legemiddelKortDose = new StringType();
104.        }
105.        return legemiddelKortDose;
106.    }
107.    public void setLegemiddelKortDose(StringType legemiddelKortDose) {
108.        this.legemiddelKortDose = legemiddelKortDose;
109.    }
110. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRResources;
2.  import org.hl7.fhir.dstu3.model.*;
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  import ca.uhn.fhir.model.api.annotation.Child;
6.  import ca.uhn.fhir.model.api.annotation.Description;
7.  import ca.uhn.fhir.model.api.annotation.Extension;
8.  import ca.uhn.fhir.model.api.annotation.ResourceDef;
9.  import ca.uhn.fhir.util.ElementUtil;
10. import japa.parser.ast.type.ReferenceType;  /** * Definition class for adding extensions to the built-
    in Substance resource * type. */  @
11. ResourceDef(name = "Substance", profile = "http://ehealth.uia.no/StructureDefinition/FESTSubstance") public class FESTSu
    bstanceResource extends Substance {
12.     private static final long serialVersionUID = 1 L; // FESTSubstanceResource stores the following values about virkestoff //
     1. virkestoff id Substance.code.coding.display // Substance.code.coding.userSelected true // 2. navn Substance.identifier.value // 3.
     engelskNavn slice extension on Substance.code.text // Slicing start
13.       @
14.     Child(name = "engelskNavn")@ Extension(url = "http://ehealth.uia.no/StructureDefinition/engelskNavn", definedLocally =
    false, isModifier = false)@ Description(shortDefinition = "The english name of the active substance in FEST M30 2.5.1") pri
    vate StringType engelskNavn; // Slicing end
15.       /**  * adding isEmpty() check for newly added extensions.  */  @
```

```
16.     Override public boolean isEmpty() {
17.         return super.isEmpty() && ElementUtil.isEmpty(engelskNavn);
18.     }
19.     public StringType getEngelskNavn() {
20.         if (engelskNavn == null) {
21.             engelskNavn = new StringType();
22.         }
23.         return engelskNavn;
24.     }
25.     public void setEngelskNavn(StringType engelskNavn) {
26.         this.engelskNavn = engelskNavn;
27.     }
28. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRController;
2.  import java.io.BufferedInputStream;
3.  import java.io.File;
4.  import java.io.FileInputStream;
5.  import java.io.FileNotFoundException;
6.  import java.util.ArrayList;
7.  import java.util.Iterator;
8.  import java.util.List;
9.  import org.hl7.fhir.dstu3.model.Bundle;
10. import org.hl7.fhir.dstu3.model.Medication;
11. import org.hl7.fhir.dstu3.model.Bundle.BundleEntryComponent;
12. import org.hl7.fhir.dstu3.model.Extension;
13. import org.hl7.fhir.dstu3.model.Group;
14. import org.hl7.fhir.dstu3.model.Group.GroupMemberComponent;
15. import ca.uhn.fhir.context.FhirContext;
16. import ca.uhn.fhir.rest.client.api.IGenericClient;
17. import ca.uhn.fhir.rest.gclient.TokenClientParam;
18. import no.uia.ehealth.FESTonHapiFHIRResources.FESTMedicationResource;
19. import no.uia.ehealth.FHIRInformationModel.InteractionFHIRInformation;
20. import no.uia.ehealth.FHIRInformationModel.MedicationFHIRInformation;
21. import no.uia.ehealth.FHIRInformationModel.SubstanceFHIRInformation;
22. public class CheckForInteraction {
23.     protected FhirContext ctx = FhirContext.forDstu3();
24.     protected String drugA;
25.     protected String drugB;
26.     protected String baseURL;
27.     protected Boolean flagATCCodeA = false;
28.     protected Boolean flagVareNavnA = false;
29.     protected Boolean flagSubsNameA = false;
30.     protected Boolean flagATCCodeB = false;
31.     protected Boolean flagVareNavnB = false;
32.     protected Boolean flagSubsNameB = false;
33.     protected String aTC_CODE_A = "";
34.     private String drugA_ID = "";
35.     protected String aTC_CODE_B = "";
36.     private String drugB_ID = "";
37.     private DrugID_ATC_CODEPair drugID_atc_CODE_pair_drugA;
38.     private DrugID_ATC_CODEPair drugID_atc_CODE_pair_drugB;
39.     protected InteractionDetails interaction;
40.     private Boolean isInteraction = false;
41.     public CheckForInteraction(FhirContext ctx, String drugA, String drugB, String baseURL) {
42.         this.ctx = ctx;
43.         this.drugA = drugA;
44.         this.drugB = drugB;
45.         this.baseURL = baseURL; // 1. we need to get the ATC-CODE of drugA and drugB. // The practitioner can search by ATC-
    CODE, vareNavn, or active substance name. // 1.1 Check whether drugA search keyword is ATC-
    CODE, product name (vareNavn) , // or active substance name
46.         this.drugID_atc_CODE_pair_drugA = inputCheckerA(drugA); // 1.2 Check whether drugB search keyword is ATC-
    CODE, product name (vareNavn) , // or active substance name
47.         this.drugID_atc_CODE_pair_drugB = inputCheckerB(drugB); // 2. Search for all FESTGroup instances which include Medic
    ation reference // entries for drugA // store IDs of interactions and groups for drug A
```

```
48.         List < InteractionGroupPair > interaction group drugA = drugGroups(this.drugID atc CODE pair drugA); // 3. Search for all FESTGroup instances which include Medication reference // entries for drugB // store IDs of interactions and groups for drug B
49.         List < InteractionGroupPair > interaction group drugB = drugGroups(this.drugID atc CODE pair drugB); // 4. Compare b oth lists detect interaction, if any.
50.         Iterator itrA = interaction_group_drugA.iterator();
51.         Iterator itrB = interaction_group_drugB.iterator();
52.         while (itrA.hasNext()) {
53.           InteractionGroupPair pairDrugA = (InteractionGroupPair) itrA.next();
54.           while (itrB.hasNext()) {
55.             InteractionGroupPair pairDrugB = (InteractionGroupPair) itrB.next(); // FEST interaction business logic // find the two d ifferent groups which have the same interaction id
56.             if (!(pairDrugA.getGroup ID().equals(pairDrugB.getGroup ID())) && (pairDrugA.getInteraction ID().equals(pairDrug B.getInteraction ID()))) { // get interaction details for this group id
57.               String id = "";
58.               String relevansV = "";
59.               String relevansDN = "";
60.               String kliniskKonsekvens = "";
61.               String handtering = "";
62.               IGenericClient client = ctx.newRestfulGenericClient(baseURL);
63.               Group theOne = client.read().resource(Group.class).withId(pairDrugA.getGroup_ID()).execute(); // get interaction ID
64.               List < Extension > resourceExtsInteractionID = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition /interactionID");
65.               if (resourceExtsInteractionID != null) {
66.                 Extension interactionID = resourceExtsInteractionID.get(0);
67.                 id = interactionID.getValue().toString();
68.               } // get Interaction Relevans V attribute
69.               List < Extension > resourceExtsInteractionRelevansV = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureD efinition/interactionRelevansV");
70.               if (resourceExtsInteractionRelevansV != null) {
71.                 Extension interactionRelevansV = resourceExtsInteractionRelevansV.get(0);
72.                 relevansV = interactionRelevansV.getValue().toString();
73.               } // get Interaction Relevans DN attribute
74.               List < Extension > resourceExtsInteractionRelevansDN = theOne.getExtensionsByUrl("http://ehealth.uia.no/Structure Definition/interactionRelevansDN");
75.               if (resourceExtsInteractionRelevansDN != null) {
76.                 Extension interactionRelevansDN = resourceExtsInteractionRelevansDN.get(0);
77.                 relevansDN = interactionRelevansDN.getValue().toString();
78.               } // get Interaction KliniskKonsekvens
79.               List < Extension > resourceExtsInteractionKliniskKonsekvens = theOne.getExtensionsByUrl("http://ehealth.uia.no/Str uctureDefinition/interactionKliniskKonsekvens");
80.               if (resourceExtsInteractionKliniskKonsekvens != null) {
81.                 Extension interactionKliniskKonsekvens = resourceExtsInteractionKliniskKonsekvens.get(0);
82.                 kliniskKonsekvens = interactionKliniskKonsekvens.getValue().toString();
83.               } // get Interaction Handtering
84.               List < Extension > resourceExtsInteractionHandtering = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureD efinition/interactionHandtering");
85.               if (resourceExtsInteractionHandtering != null) {
86.                 Extension interactionHandtering = resourceExtsInteractionHandtering.get(0);
87.                 handtering = interactionHandtering.getValue().toString();
88.               } // set interaction details
89.               this.interaction = new InteractionDetails(id, relevansV, relevansDN, kliniskKonsekvens, handtering);
90.               this.isInteraction = true;
91.               break;
92.             }
93.           }
94.         }
95.       }
96.       private List < InteractionGroupPair > drugGroups(DrugID ATC CODEPair pair) {
97.         List < InteractionGroupPair > interaction group for drug = new ArrayList < InteractionGroupPair > ();
98.         IGenericClient clientDrugAGroupIDs = ctx.newRestfulGenericClient(baseURL);
99.         Bundle resultGroups = clientDrugAGroupIDs.search().forResource(Group.class).where(Group.MEMBER.hasId(pair.getDRU G ID())).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
100.        if (resultGroups != null) {
101.          if (resultGroups.getTotal() > 0) {
102.            for (BundleEntryComponent e: resultGroups.getEntry()) {
```

```
103.                 String groupID = "";
104.                 String interactionID = "";
105.                 if (e.getResource() instanceof Group) { // get group id
106.                     Group g = (Group) e.getResource();
107.                     groupID = g.getId(); // get interaction id
108.                     List < Extension > resourceExts = g.getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionID");

109.                     Extension exti = resourceExts.get(0);
110.                     interactionID = exti.getValue().toString();
111.                     InteractionGroupPair p = new InteractionGroupPair(interactionID, groupID);
112.                     interaction group for drug.add(p);
113.                 }
114.             }
115.         }
116.     }
117.     return interaction group for drug;
118.  }
119.  private DrugID ATC CODEPair inputCheckerA(String drug) { // 1.1.1 ATC-CODE checker
120.     IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
121.     Bundle resultATCV = clientATCV.search().forResource(Medication.class).where(Medication.CODE.exactly().code(drug)).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
122.     if (resultATCV != null) {
123.         if (resultATCV.getTotal() == 0) {
124.             this.flagATCCodeA = false;
125.         } else {
126.             this.flagATCCodeA = true;
127.             if (resultATCV.getEntry().get(0).getResource() != null) {
128.                 if (resultATCV.getEntry().get(0).getResource() instanceof Medication) {
129.                     Medication m = (Medication) resultATCV.getEntry().get(0).getResource();
130.                     this.aTC CODE A = drug;
131.                     this.drugA_ID = m.getId();
132.                     this.drugID atc CODE pair drugA = new DrugID ATC CODEPair(m.getId(), drug);
133.                 }
134.             }
135.         }
136.     }
137.     if (!flagATCCodeA) { // 1.1.2 Legemiddel varenavn checker // Note that we have defined the search parameter legemiddelVarenavn in // FESTMedicationHapiFHIRController.java // use the added search parameter
138.         IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
139.         Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class).where(new TokenClientParam("legemiddelVarenavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
140.         if (resultVareNavn != null) {
141.             if (resultVareNavn.getTotal() == 0) {
142.                 this.flagVareNavnA = false;
143.             } else {
144.                 this.flagVareNavnA = true;
145.                 if (resultVareNavn.getEntry().get(0).getResource() != null) {
146.                     if (resultVareNavn.getEntry().get(0).getResource() instanceof Medication) {
147.                         Medication m = (Medication) resultVareNavn.getEntry().get(0).getResource();
148.                         this.drugA_ID = m.getId();
149.                         this.aTC CODE A = m.getCode().getCoding().get(0).getCode();
150.                         this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0).getCode());
151.                     }
152.                 }
153.             }
154.         }
155.     }
156.     if (!this.flagATCCodeA && !this.flagVareNavnA) { // 1.1.3 active substance name checker // Note that we have defined the search parameter atcNavn in // FESTMedicationHapiFHIRController.java // use the added search parameter
157.         IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
158.         Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class).where(new TokenClientParam("atcNavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
159.         if (resultSubstanceName != null) {
160.             if (resultSubstanceName.getTotal() == 0) {
161.                 this.flagSubsNameA = false;
162.             } else {
```

```
163.                 this.flagSubsNameA = true;
164.                 if (resultSubstanceName.getEntry().get(0).getResource() != null) {
165.                     if (resultSubstanceName.getEntry().get(0).getResource() instanceof Medication) {
166.                         Medication m = (Medication) resultSubstanceName.getEntry().get(0).getResource();
167.                         this.drugA_ID = m.getId();
168.                         this.aTC_CODE_A = m.getCode().getCoding().get(0).getCode();
169.                         this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
170.                     }
171.                 }
172.             }
173.         }
174.     }
175.     return drugID_atc_CODE_pair_drugA;
176.   }
177.   private DrugID_ATC_CODEPair inputCheckerB(String drug) { // 1.1.1 ATC-CODE checker
178.     IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
179.     Bundle resultATCV = clientATCV.search().forResource(Medication.class).where(Medication.CODE.exactly().code(drug)).re
    turnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
180.     if (resultATCV != null) {
181.       if (resultATCV.getTotal() == 0) {
182.         this.flagATCCodeB = false;
183.       } else {
184.         this.flagATCCodeB = true;
185.         if (resultATCV.getEntry().get(0).getResource() != null) {
186.           if (resultATCV.getEntry().get(0).getResource() instanceof Medication) {
187.             Medication m = (Medication) resultATCV.getEntry().get(0).getResource();
188.             this.aTC_CODE_B = drug;
189.             this.drugB_ID = m.getId();
190.             this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), drug);
191.           }
192.         }
193.       } // 1.1.2 Legemiddel varenavn checker // Note that we have defined the search parameter legemiddelVarenavn in // FEST
    MedicationHapiFHIRController.java // use the added search parameter
194.     }
195.     if (!flagATCCodeB) {
196.       IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
197.       Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class).where(new TokenClientParam("legemid
    delVarenavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
198.       if (resultVareNavn != null) {
199.         if (resultVareNavn.getTotal() == 0) {
200.           this.flagVareNavnB = false;
201.         } else {
202.           this.flagVareNavnB = true;
203.           if (resultVareNavn.getEntry().get(0).getResource() != null) {
204.             if (resultVareNavn.getEntry().get(0).getResource() instanceof Medication) {
205.               Medication m = (Medication) resultVareNavn.getEntry().get(0).getResource();
206.               this.drugB_ID = m.getId();
207.               this.aTC_CODE_B = m.getCode().getCoding().get(0).getCode();
208.               this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
209.             }
210.           }
211.         }
212.       }
213.     } // 1.1.3 active substance name checker // Note that we have defined the search parameter atcNavn in // FESTMedicationHap
    iFHIRController.java // use the added search parameter
214.     if (!this.flagATCCodeB && !this.flagVareNavnB) {
215.       IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
216.       Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class).where(new TokenClientParam("atc
    Navn").exactly().code(drug)).returnBundle(Bundle.class).execute();
217.       if (resultSubstanceName != null) {
218.         if (resultSubstanceName.getTotal() == 0) {
219.           this.flagSubsNameB = false;
220.         } else {
221.           this.flagSubsNameB = true;
222.           if (resultSubstanceName.getEntry().get(0).getResource() != null) {
```

```
223.                     if (resultSubstanceName.getEntry().get(0).getResource() instanceof Medication) {
224.                         Medication m = (Medication) resultSubstanceName.getEntry().get(0).getResource();
225.                         this.drugB_ID = m.getId();
226.                         this.aTC_CODE_B = m.getCode().getCoding().get(0).getCode();
227.                         this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
228.                     }
229.                 }
230.             }
231.         }
232.     }
233.     return drugID_atc_CODE_pair_drugB;
234. }
235. public String getATC_CODE_A() {
236.     if (aTC_CODE_A == null) {
237.         aTC_CODE_A = new String();
238.     }
239.     return aTC_CODE_A;
240. }
241. public void setATC_CODE_A(String aTC_CODE_A) {
242.     this.aTC_CODE_A = aTC_CODE_A;
243. }
244. public String getATC_CODE_B() {
245.     if (aTC_CODE_B == null) {
246.         aTC_CODE_B = new String();
247.     }
248.     return aTC_CODE_B;
249. }
250. public void setATC_CODE_B(String aTC_CODE_B) {
251.     this.aTC_CODE_B = aTC_CODE_B;
252. }
253. public Boolean getFlagATCCodeA() {
254.     if (flagATCCodeA == null) {
255.         flagATCCodeA = false;
256.     }
257.     return flagATCCodeA;
258. }
259. public void setFlagATCCodeA(Boolean flagATCCodeA) {
260.     this.flagATCCodeA = flagATCCodeA;
261. }
262. public Boolean getFlagVareNavnA() {
263.     if (flagVareNavnA == null) {
264.         flagVareNavnA = false;
265.     }
266.     return flagVareNavnA;
267. }
268. public void setFlagVareNavnA(Boolean flagVareNavnA) {
269.     this.flagVareNavnA = flagVareNavnA;
270. }
271. public Boolean getFlagSubsNameA() {
272.     if (flagSubsNameA == null) {
273.         flagSubsNameA = false;
274.     }
275.     return flagSubsNameA;
276. }
277. public void setFlagSubsNameA(Boolean flagSubsNameA) {
278.     this.flagSubsNameA = flagSubsNameA;
279. }
280. public Boolean getFlagATCCodeB() {
281.     if (flagATCCodeB == null) {
282.         flagATCCodeB = false;
283.     }
284.     return flagATCCodeB;
285. }
286. public void setFlagATCCodeB(Boolean flagATCCodeB) {
287.     this.flagATCCodeB = flagATCCodeB;
288. }
```

```
289.    public Boolean getFlagVareNavnB() {
290.        if (flagVareNavnB == null) {
291.            flagVareNavnB = false;
292.        }
293.        return flagVareNavnB;
294.    }
295.    public void setFlagVareNavnB(Boolean flagVareNavnB) {
296.        this.flagVareNavnB = flagVareNavnB;
297.    }
298.    public Boolean getFlagSubsNameB() {
299.        if (flagSubsNameB) {
300.            flagSubsNameB = false;
301.        }
302.        return flagSubsNameB;
303.    }
304.    public void setFlagSubsNameB(Boolean flagSubsNameB) {
305.        this.flagSubsNameB = flagSubsNameB;
306.    }
307.    public InteractionDetails getInteraction() {
308.        return interaction;
309.    }
310.    public void setInteraction(InteractionDetails interaction) {
311.        this.interaction = interaction;
312.    }
313.    public Boolean getIsInteraction() {
314.        if (isInteraction == null) {
315.            isInteraction = false;
316.        }
317.        return isInteraction;
318.    }
319.    public void setIsInteraction(Boolean isInteraction) {
320.        this.isInteraction = isInteraction;
321.    }
322. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRController;
2.  public class DrugID_ATC_CODEPair {
3.      private String dRUG_ID;
4.      private String dRUG_ATC_CODE;
5.      public DrugID_ATC_CODEPair(String dRUG_ID, String dRUG_ATC_CODE) {
6.          this.dRUG_ID = dRUG_ID;
7.          this.dRUG_ATC_CODE = dRUG_ATC_CODE;
8.      }
9.      public String getDRUG_ID() {
10.         if (dRUG_ID == null) {
11.             dRUG_ID = new String();
12.         }
13.         return dRUG_ID;
14.     }
15.     public void setDRUG_ID(String dRUG_ID) {
16.         this.dRUG_ID = dRUG_ID;
17.     }
18.     public String getDRUG_ATC_CODE() {
19.         if (dRUG_ATC_CODE == null) {
20.             dRUG_ATC_CODE = new String();
21.         }
22.         return dRUG_ATC_CODE;
23.     }
```

```
24.     public void setDRUG_ATC_CODE(String dRUG_ATC_CODE) {
25.         this.dRUG_ATC_CODE = dRUG_ATC_CODE;
26.     }
27. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRController;
2.  import java.security.Principal;
3.  import java.security.acl.Group;
4.  import java.util.ArrayList;
5.  import java.util.List;
6.  import org.hl7.fhir.dstu3.model.Bundle;
7.  import org.hl7.fhir.dstu3.model.Bundle.BundleEntryComponent;
8.  import org.hl7.fhir.dstu3.model.DateTimeType;
9.  import org.hl7.fhir.dstu3.model.DetectedIssue;
10. import org.hl7.fhir.dstu3.model.Enumeration;
11. import org.hl7.fhir.dstu3.model.Identifier.IdentifierUse;
12. import org.hl7.fhir.dstu3.model.ListResource;
13. import org.hl7.fhir.dstu3.model.ListResource.ListEntryComponent;
14. import org.hl7.fhir.dstu3.model.Medication;
15. import org.hl7.fhir.dstu3.model.Patient;
16. import org.hl7.fhir.dstu3.model.Reference;
17. import org.hl7.fhir.dstu3.model.Resource;
18. import org.hl7.fhir.dstu3.model.StringType;
19. import org.hl7.fhir.dstu3.model.codesystems.GroupType;
20. import org.hl7.fhir.dstu3.model.codesystems.GroupTypeEnumFactory;
21. import org.hl7.fhir.instance.model.api.IBaseOperationOutcome;
22. import org.hl7.fhir.dstu3.model.Extension;
23. import org.hl7.fhir.dstu3.model.Group.GroupMemberComponent;
24. import org.hl7.fhir.dstu3.model.HumanName;
25. import ca.uhn.fhir.context.FhirContext; //import ca.uhn.fhir.model.api.annotation.Extension;
26. import ca.uhn.fhir.model.primitive.IdDt;
27. import ca.uhn.fhir.model.primitive.StringDt;
28. import ca.uhn.fhir.rest.client.api.IGenericClient;
29. import no.uia.ehealth.FESTonHapiFHIRResources.*;
30. import no.uia.ehealth.FHIRInformationModel.InteractionFHIRInformation;
31. import no.uia.ehealth.FHIRInformationModel.MedicationFHIRInformation;
32. import no.uia.ehealth.FHIRInformationModel.SubstanceFHIRInformation;
33. import no.uia.ehealth.FHIRInformationModel.SubstanceGroupFHIRInformation;
34. import no.uia.ehealth.MappingFESTXmlModelToJavaModel.AutomaticMappingFESTOnFHIR;
35. public class FESTInteractionHapiFHIRController {
36.     protected FhirContext ctx = FhirContext.forDstu3(); // very expensive object, create as less as we can!
37.     protected List < InteractionFHIRInformation > interactionFHIRInformationList;
38.     protected String baseURL;
39.     public FESTInteractionHapiFHIRController(FhirContext ctx, List < InteractionFHIRInformation > interactionFHIRInformation
    List, String baseURL) {
40.         this.ctx = ctx;
41.         this.interactionFHIRInformationList = interactionFHIRInformationList;
42.         this.setBaseUrl(baseURL); // logic for creating resources and populating data
43.         for (InteractionFHIRInformation interaction: interactionFHIRInformationList) {
44.             String interactionID = "";
45.             String interactionFHIRRelevansV = "";
46.             String interactionFHIRRelevansDN = "";
47.             String interactionFHIRHandtering = "";
48.             String interactionFHIRKliniskKonsekvens = "";
49.             if (interaction.getInteractionFHIRID() != null) {
50.                 interactionID = interaction.getInteractionFHIRID();
51.             }
52.             if (interaction.getInteractionFHIRRelevansV() != null) {
53.                 interactionFHIRRelevansV = interaction.getInteractionFHIRRelevansV();
54.             }
55.             if (interaction.getInteractionFHIRRelevansDN() != null) {
56.                 interactionFHIRRelevansDN = interaction.getInteractionFHIRRelevansDN();
57.             }
58.             if (interaction.getInteractionFHIRHandtering() != null) {
59.                 interactionFHIRHandtering = interaction.getInteractionFHIRHandtering();
60.             }
```

```
61.             if (interaction.getInteractionFHIRKliniskKonsekvens() != null) {
62.                 interactionFHIRKliniskKonsekvens = interaction.getInteractionFHIRKliniskKonsekvens();
63.             } // each interaction has two medication groups, so we loop and process each group
64.             List < SubstanceGroupFHIRInformation > interactionRelatedGroups = interaction.getSubstanceGroupFHIRInformation();
65.             for (SubstanceGroupFHIRInformation medicationsGroup: interactionRelatedGroups) {
66.                 /*                 * Here we create the FESTGroupResource instance                 */ // Start of create a FESTGroupResource instance
67.                 FESTGroupResource fESTGroupResource = new FESTGroupResource(); // Add interaction details.
68.                 fESTGroupResource.setInteractionID(new StringType(interactionID));
69.                 fESTGroupResource.setInteractionRelevansV(new StringType(interactionFHIRRelevansV));
70.                 fESTGroupResource.setInteractionRelevansDN(new StringType(interactionFHIRRelevansDN));
71.                 fESTGroupResource.setInteractionHandtering(new StringType(interactionFHIRHandtering));
72.                 fESTGroupResource.setInteractionKliniskKonsekvens(new StringType(interactionFHIRKliniskKonsekvens));
73.                 fESTGroupResource.setActual(true); // means the group refers to specific group of real medication // instances
74.                 fESTGroupResource.setActive(true); // means the group refers to specific group of real medication // instances // Group Type.MEDICATION
75.                 fESTGroupResource.setType(FESTGroupResource.GroupType.valueOf("MEDICATION"));
76.                 /*                 * FESTGroup.member Here we populate FESTGroupResource.member by adding entity                 * reference for each FESTMedicationResource/ Medication instance we stored                 * previously                 */ // each group has a list of medication entities
77.                 List < SubstanceFHIRInformation > medicationsList = new ArrayList < SubstanceFHIRInformation > ();
78.                 if (medicationsGroup.getSubstanceFHIRInformation() != null) medicationsList = medicationsGroup.getSubstanceFHIRInformation();
79.                 for (SubstanceFHIRInformation medication: medicationsList) { // get a reference to an instance of FESTMedicationResource // which has the same atc-v and atc-dn values.
80.                     String atcV = "";
81.                     if (medication.getatcV() != null) {
82.                         atcV = medication.getatcV();
83.                     }
84.                     IGenericClient client = ctx.newRestfulGenericClient(baseURL);
85.                     Bundle result = client.search().forResource(Medication.class).where(Medication.CODE.exactly().code(atcV)).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
86.                     if (result.getEntry() != null && result.getEntry().size() != 0) {
87.                         for (BundleEntryComponent entry: result.getEntry()) {
88.                             Medication drug = (Medication) entry.getResource();
89.                             GroupMemberComponent member = new GroupMemberComponent();
90.                             member.getEntity().setReference(drug.getId());
91.                             member.setEntityTarget(drug);
92.                             member.setEntity(member.getEntity().setReference(drug.getId()));
93.                             fESTGroupResource.addMember(member);
94.                         }
95.                     }
96.                 } // End of create a FESTGroupResource instance // Send the FESTGroupResource instance to hapi-fhir server
97.                 IGenericClient clientSender = ctx.newRestfulGenericClient(baseURL);
98.                 clientSender.create().resource(fESTGroupResource).execute();
99.             }
100.         }
101.     }
102.     public List < InteractionFHIRInformation > getInteractionFHIRInformationList() {
103.         if (interactionFHIRInformationList == null) {
104.             interactionFHIRInformationList = new ArrayList < InteractionFHIRInformation > ();
105.         }
106.         return interactionFHIRInformationList;
107.     }
108.     public void setInteractionFHIRInformationList(List < InteractionFHIRInformation > interactionFHIRInformationList) {
109.         this.interactionFHIRInformationList = interactionFHIRInformationList;
110.     }
111.     public FhirContext getCtx() {
112.         if (ctx == null) {
113.             ctx = FhirContext.forDstu3();
114.         }
115.         return ctx;
116.     }
117.     public void setCtx(FhirContext ctx) {
118.         this.ctx = ctx;
119.     }
120.     public String getBaseUrl() {
```

```
121.        if (baseURL == null) {
122.            baseURL = new String();
123.        }
124.        return baseURL;
125.    }
126.    public void setBaseUrl(String value) {
127.        this.baseURL = value;
128.    }
129. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRController;
2.  import java.security.Principal;
3.  import java.security.acl.Group;
4.  import java.util.ArrayList;
5.  import java.util.Date;
6.  import java.util.List;
7.  import java.util.Random;
8.  import org.hl7.fhir.dstu3.model.BooleanType;
9.  import org.hl7.fhir.dstu3.model.Bundle;
10. import org.hl7.fhir.dstu3.model.CodeType;
11. import org.hl7.fhir.dstu3.model.DateTimeType;
12. import org.hl7.fhir.dstu3.model.DetectedIssue;
13. import org.hl7.fhir.dstu3.model.Identifier.IdentifierUse;
14. import org.hl7.fhir.dstu3.model.ListResource;
15. import org.hl7.fhir.dstu3.model.ListResource.ListEntryComponent;
16. import org.hl7.fhir.dstu3.model.Medication;
17. import org.hl7.fhir.dstu3.model.Medication.MedicationIngredientComponent;
18. import org.hl7.fhir.dstu3.model.MedicationStatement;
19. import org.hl7.fhir.dstu3.model.Patient;
20. import org.hl7.fhir.dstu3.model.StringType;
21. import org.hl7.fhir.dstu3.model.Substance;
22. import org.hl7.fhir.instance.model.api.IBaseOperationOutcome;
23. import org.hl7.fhir.dstu3.model.Extension;
24. import org.hl7.fhir.dstu3.model.HumanName;
25. import org.hl7.fhir.dstu3.model.Enumerations;
26. import ca.uhn.fhir.context.FhirContext; //import ca.uhn.fhir.model.api.annotation.Extension;
27. import ca.uhn.fhir.model.primitive.IdDt;
28. import ca.uhn.fhir.model.primitive.StringDt;
29. import ca.uhn.fhir.rest.client.api.IGenericClient;
30. import ca.uhn.fhir.rest.gclient.TokenClientParam;
31. import no.uia.ehealth.FESTonHapiFHIRResources.FESTMedicationResource;
32. import no.uia.ehealth.FESTonHapiFHIRResources.FESTSubstanceResource;
33. import no.uia.ehealth.FHIRInformationModel.InteractionFHIRInformation;
34. import no.uia.ehealth.FHIRInformationModel.MedicationFHIRInformation;
35. import no.uia.ehealth.MappingFESTXmlModelToJavaModel.AutomaticMappingFESTOnFHIR;
36. public class FESTMedicationHapiFHIRController {
37.     protected FhirContext ctx; // very expensive object, create as less as we can! this is why we get it in as // an argument in the con
    structor.
38.     protected List < MedicationFHIRInformation > medicationFHIRInformationList;
39.     protected String baseURL;
40.     public FESTMedicationHapiFHIRController(FhirContext ctx, List < MedicationFHIRInformation > medicationFHIRInformatio
    nList, String baseURL) {
41.         this.ctx = ctx;
42.         this.medicationFHIRInformationList = medicationFHIRInformationList;
43.         this.setBaseUrl(baseURL);
44.         IGenericClient client = ctx.newRestfulGenericClient(baseURL);
```

```
45.         int counter = 1; // to control the number of created instances ( for testing purpose)
46.         /*      * ************************************VERY IMPORTANT NOTE:      * ******************************
    ****************************** WE DEFINE OUR        * SEARCH PARAMETERS FOR ADDED EXTENSIONS TO OU
    R RESOURCES BEFORE WE CREATE      * INSTANCES OF THE CUSTOM RESOURCE      * ************************
    ************************************************          * ************************************** i.e : in
         * FESTMedicationHapiFHIRController.java we define search parameters for vare        * navn and atcDN (substance name)
     */ // Start varenavn Search Parameter Definition & Creation // Start definition
47.         org.hl7.fhir.dstu3.model.SearchParameter vareNavnSP = new org.hl7.fhir.dstu3.model.SearchParameter();
48.         vareNavnSP.addBase("Medication");
49.         vareNavnSP.setCode("legemiddelVarenavn");
50.         vareNavnSP.setType(org.hl7.fhir.dstu3.model.Enumerations.SearchParamType.TOKEN);
51.         vareNavnSP.setTitle("Bruk legemiddel varenavn");
52.         vareNavnSP.setDescription("FEST:Søk med legemiddel varenavn");
53.         vareNavnSP.setExpression("Medication.extension('http://ehealth.uia.no/StructureDefinition/legemiddelVarenavn')");
54.         vareNavnSP.setXpathUsage(org.hl7.fhir.dstu3.model.SearchParameter.XPathUsageType.NORMAL);
55.         vareNavnSP.setStatus(org.hl7.fhir.dstu3.model.Enumerations.PublicationStatus.ACTIVE); // End definition
56.         IGenericClient clientVareNavnSP = ctx.newRestfulGenericClient(baseURL); // Start Creation // Upload it to the server
57.         clientVareNavnSP.create().resource(vareNavnSP).execute(); // End creation // End vareNavn Search Parameters Definition &
    Creation // Start definition
58.         org.hl7.fhir.dstu3.model.SearchParameter atcNavnSP = new org.hl7.fhir.dstu3.model.SearchParameter();
59.         atcNavnSP.addBase("Medication");
60.         atcNavnSP.setCode("atcNavn");
61.         atcNavnSP.setType(org.hl7.fhir.dstu3.model.Enumerations.SearchParamType.TOKEN);
62.         atcNavnSP.setTitle("Bruk legemiddel virkestoff navn");
63.         atcNavnSP.setDescription("FEST: Søk med ATC-navn (DN-verdi i ATC-kode)");
64.         atcNavnSP.setExpression("Medication.extension('http://ehealth.uia.no/StructureDefinition/atcNavn')");
65.         atcNavnSP.setXpathUsage(org.hl7.fhir.dstu3.model.SearchParameter.XPathUsageType.NORMAL);
66.         atcNavnSP.setStatus(org.hl7.fhir.dstu3.model.Enumerations.PublicationStatus.ACTIVE); // End definition
67.         IGenericClient clientATCDNSP = ctx.newRestfulGenericClient(baseURL); // Start Creation // Upload it to the server
68.         clientATCDNSP.create().resource(atcNavnSP).execute(); // End creation // End atcNavn Search Parameters Definition & Cre
    ation // logic for creating resources and populating data
69.         for (MedicationFHIRInformation m: medicationFHIRInformationList) { // Here we create a FESTMedicationResource instan
    ce
70.             FESTMedicationResource fESTMedicationResource = new FESTMedicationResource();
71.             String ATC_CODE = "";
72.             if (m.getATCV() != null) {
73.                 ATC_CODE = m.getATCV();
74.             }
75.             fESTMedicationResource.getCode().addCoding().setSystem("http://www.whocc.no/atc").setVersion("7180").setCode(AT
    C_CODE).setDisplay("ATC code");
76.             String ATC  NAME = "";
77.             if (m.getatcDN() != null) {
78.                 ATC_NAME = m.getatcDN();
79.             }
80.             fESTMedicationResource.setAtcNavn(new StringType(ATC  NAME));
81.             fESTMedicationResource.setIsBrand(true);
82.             if ((m.getLegemiddelVarselTrekant() != null) && (m.getLegemiddelVarselTrekant())) {
83.                 fESTMedicationResource.setLegemiddelVarselTrekant(new BooleanType(true));
84.             } else {
85.                 fESTMedicationResource.setLegemiddelVarselTrekant(new BooleanType(false));
86.             }
87.             if (m.getLegemiddelTypeSoknadV() != null) {
88.                 if (m.getLegemiddelTypeSoknadV().equals("1")) {
89.                     fESTMedicationResource.setIsOverTheCounter(false);
90.                 } else {
91.                     fESTMedicationResource.setIsOverTheCounter(true);
92.                 }
93.             }
94.             String producer = "";
95.             if (m.getLegemiddelProdusent() != null) {
96.                 producer = m.getLegemiddelProdusent();
97.             }
98.             fESTMedicationResource.getManufacturer().setDisplay(producer);
99.             String legemiddelFESTID = "";
100.             if (m.getLegemiddelID() != null) {
101.                 legemiddelFESTID = m.getLegemiddelID();
102.             }
```

```
103.            fESTMedicationResource.setLegemiddelFESTID(new StringType(legemiddelFESTID));
104.            String legemiddelVarenavn = "";
105.            if (m.getLegemiddelVarenavn() != null) {
106.               legemiddelVarenavn = m.getLegemiddelVarenavn();
107.            }
108.            fESTMedicationResource.setLegemiddelVarenavn(new StringType(legemiddelVarenavn));
109.            String legemiddelNavnFormStyrke = "";
110.            if (m.getLegemiddelNavnFormStyrke() != null) {}
111.            legemiddelNavnFormStyrke = m.getLegemiddelNavnFormStyrke();
112.            fESTMedicationResource.setLegemiddelNavnFormStyrke(new StringType(legemiddelNavnFormStyrke));
113.            String legemiddelAdministrasjonsvei = "";
114.            if (m.getLegemiddelAdministrasjonsveiDN() != null) {
115.               legemiddelAdministrasjonsvei = m.getLegemiddelAdministrasjonsveiDN();
116.            }
117.            fESTMedicationResource.setLegemiddelAdministrasjonsvei(new StringType(legemiddelAdministrasjonsvei));
118.            String legemiddelEnhetsdosering = "";
119.            if (m.getLegemiddelEnhetsdoseringDN() != null) {
120.               legemiddelEnhetsdosering = m.getLegemiddelEnhetsdoseringDN();
121.            }
122.            fESTMedicationResource.setLegemiddelEnhetsdosering(new StringType(legemiddelEnhetsdosering));
123.            String legemiddelKortDose = "";
124.            if (m.getLegemiddelKortDoseDN() != null) {
125.               legemiddelKortDose = m.getLegemiddelKortDoseDN();
126.            }
127.            fESTMedicationResource.setLegemiddelKortDose(new StringType(legemiddelKortDose));
128.            client.create().resource(fESTMedicationResource).execute();
129.            System.out.println(counter++);
130.            System.out.println();
131.         }
132.      }
133.      public List < MedicationFHIRInformation > getMedicationFHIRInformationList() {
134.         if (medicationFHIRInformationList == null) {
135.            medicationFHIRInformationList = new ArrayList < MedicationFHIRInformation > ();
136.         }
137.         return medicationFHIRInformationList;
138.      }
139.      public void setMedicationFHIRInformationList(List < MedicationFHIRInformation > medicationFHIRInformationList) {
140.         this.medicationFHIRInformationList = medicationFHIRInformationList;
141.      }
142.      public FhirContext getCtx() {
143.         if (ctx == null) {
144.            ctx = FhirContext.forDstu3();
145.         }
146.         return ctx;
147.      }
148.      public void setCtx(FhirContext ctx) {
149.         this.ctx = ctx;
150.      }
151.      public String getBaseUrl() {
152.         if (baseURL == null) {
153.            baseURL = new String();
154.         }
155.         return baseURL;
156.      }
157.      public void setBaseUrl(String value) {
158.         this.baseURL = value;
159.      }
160. }
```

```
1.    package no.uia.ehealth.FESTonHapiFHIRController;
2.    public class InteractionDetails {
3.       protected String id;
4.       protected String relevansV;
5.       protected String relevansDN;
6.       protected String kliniskKonsekvens;
7.       protected String handtering;
```

```
8.      public InteractionDetails(String id, String relevansV, String relevansDN, String kliniskKonsekvens, String handtering) {
9.          this.id = id;
10.         this.relevansV = relevansV;
11.         this.relevansDN = relevansDN;
12.         this.kliniskKonsekvens = kliniskKonsekvens;
13.         this.handtering = handtering;
14.     }
15.     public String getId() {
16.         if (id == null) {
17.             this.id = new String();
18.         }
19.         return id;
20.     }
21.     public void setId(String id) {
22.         this.id = id;
23.     }
24.     public String getRelevansV() {
25.         if (relevansV == null) {
26.             this.relevansV = new String();
27.         }
28.         return relevansV;
29.     }
30.     public void setRelevansV(String relevansV) {
31.         this.relevansV = relevansV;
32.     }
33.     public String getRelevansDN() {
34.         if (relevansDN == null) {
35.             this.relevansDN = new String();
36.         }
37.         return relevansDN;
38.     }
39.     public void setRelevansDN(String relevansDN) {
40.         this.relevansDN = relevansDN;
41.     }
42.     public String getKliniskKonsekvens() {
43.         if (kliniskKonsekvens == null) {
44.             this.kliniskKonsekvens = new String();
45.         }
46.         return kliniskKonsekvens;
47.     }
48.     public void setKliniskKonsekvens(String kliniskKonsekvens) {
49.         this.kliniskKonsekvens = kliniskKonsekvens;
50.     }
51.     public String getHandtering() {
52.         if (handtering == null) {
53.             this.handtering = new String();
54.         }
55.         return handtering;
56.     }
57.     public void setHandtering(String handtering) {
58.         this.handtering = handtering;
59.     }
60. }
```

```
1.  package no.uia.ehealth.FESTonHapiFHIRController;
2.  public class InteractionGroupPair {
3.      private String interaction_ID;
4.      private String group  ID;
5.      public InteractionGroupPair(String interaction  ID, String group  ID) {
6.          this.interaction_ID = interaction_ID;
7.          this.group_ID = group_ID;
8.      }
9.      public String getInteraction_ID() {
10.         if (interaction  ID == null) {
11.             interaction_ID = new String();
```

```
        }
        return interaction_ID;
    }
    public void setInteraction_ID(String interaction_ID) {
        this.interaction_ID = interaction_ID;
    }
    public String getGroup_ID() {
        if (group_ID == null) {
            group_ID = new String();
        }
        return group_ID;
    }
    public void setGroup_ID(String group_ID) {
        this.group_ID = group_ID;
    }
}
```

7.2 Appendix B: Java Web Application

```
package no.uia.ehealth;
import java.util.Optional;
import org.apache.catalina.startup.Tomcat;
public class Main {
    public static final Optional < String > PORT = Optional.ofNullable(System.getenv("PORT"));
    public static final Optional < String > HOSTNAME = Optional.ofNullable(System.getenv("HOSTNAME"));
    public static void main(String[] args) throws Exception {
        String contextPath = "/";
        String appBase = ".";
        Tomcat tomcat = new Tomcat();
        tomcat.setPort(Integer.valueOf(PORT.orElse("8080")));
        tomcat.setHostname(HOSTNAME.orElse("localhost"));
        tomcat.getHost().setAppBase(appBase);
        tomcat.addWebapp(contextPath, appBase);
        tomcat.start();
        tomcat.getServer().await();
    }
}
```

```
package no.uia.ehealth;
import java.io.IOException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;@
WebServlet(name = "InteractionServlet", urlPatterns = {
    "/interaction"
}) public class InteractionServlet extends HttpServlet {
    /*   * Steps to compile 1. mvn clean compile 2. mvn package 3. cd target 4. java     * -jar ddi-app-snapshot.jar     */
    @
    Override protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String action = req.getParameter("searchAction");
        if (action != null) {
            switch (action) {
                case "checkDDI":
                    System.out.println("checkDDI parameter is detected");
```

```
                System.out.println("calling checkForDDI() method");
                checkForDDI(req, resp);
                break;
            }
        } else { // List<Employee> result = employeeService.getAllEmployees();
            System.out.println("no action parameters");
            forwardDisplayDDI(req, resp);
        }
    }
    private void forwardDisplayDDI(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String nextJSP = "/jsp/check-ddi.jsp";
        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(nextJSP);
        System.out.println("display empty ddi page");
        dispatcher.forward(req, resp);
    }
    private void checkForDDI(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String drugA = req.getParameter("drugA");
        String drugB = req.getParameter("drugB");
        InteractionService interactionService = new InteractionService();
        System.out.println("before calling our code for ddi check");
        InteractionDetails interactionResult = interactionService.ddiCheck(drugA, drugB);
        System.out.println("after calling our code for ddi check");
        displayInteractionResult(req, resp, interactionResult);
    }
    private void displayInteractionResult(HttpServletRequest req, HttpServletResponse resp, InteractionDetails interactionResult) t
hrows ServletException, IOException {
        String nextJSP = "/jsp/check-ddi.jsp";
        System.out.println("before display ddi result");
        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(nextJSP);
        req.setAttribute("interactionResult", interactionResult);
        dispatcher.forward(req, resp);
    }
}
```

```
package no.uia.ehealth;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import ca.uhn.fhir.context.FhirContext;
import no.uia.ehealth.CheckForInteraction;
public class InteractionService {
    public InteractionDetails ddiCheck(String drugA, String drugB) {
        InteractionDetails result;
        String baseURL = "http://localhost:8081/fest/baseDstu3"; // to be replaced with url of production web server
        FhirContext ctx = FhirContext.forDstu3();
        CheckForInteraction interactionCheckup = new CheckForInteraction(ctx, drugA, drugB, baseURL);
        result = interactionCheckup.getInteraction();
        return result;
    }
}
```

```
package no.uia.ehealth;
public class InteractionGroupPair {
    private String interaction_ID;
    private String group_ID;
    public InteractionGroupPair(String interaction  ID, String group  ID) {
        this.interaction_ID = interaction_ID;
        this.group  ID = group  ID;
    }
    public String getInteraction  ID() {
        if (interaction_ID == null) {
            interaction_ID = new String();
```

```
        }
        return interaction_ID;
    }
    public void setInteraction_ID(String interaction_ID) {
        this.interaction_ID = interaction_ID;
    }
    public String getGroup_ID() {
        if (group_ID == null) {
            group_ID = new String();
        }
        return group_ID;
    }
    public void setGroup_ID(String group_ID) {
        this.group_ID = group_ID;
    }
}
```

```
package no.uia.ehealth;
public class InteractionDetails {
    protected String id;
    protected String relevansV;
    protected String relevansDN;
    protected String kliniskKonsekvens;
    protected String handtering;
    protected String drugA;
    protected String drugB;
    public InteractionDetails(String id, String relevansV, String relevansDN, String kliniskKonsekvens, String handtering, String drugA, String drugB) {
        this.id = id;
        this.relevansV = relevansV;
        this.relevansDN = relevansDN;
        this.kliniskKonsekvens = kliniskKonsekvens;
        this.handtering = handtering;
        this.drugA = drugA;
        this.drugB = drugB;
    }
    public String getDrugA() {
        return drugA;
    }
    public void setDrugA(String drugA) {
        this.drugA = drugA;
    }
    public String getDrugB() {
        return drugB;
    }
    public void setDrugB(String drugB) {
        this.drugB = drugB;
    }
    public String getId() {
        if (id == null) {
            this.id = new String();
        }
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getRelevansV() {
        if (relevansV == null) {
            this.relevansV = new String();
        }
        return relevansV;
    }
    public void setRelevansV(String relevansV) {
        this.relevansV = relevansV;
    }
```

```
49.     public String getRelevansDN() {
50.         if (relevansDN == null) {
51.             this.relevansDN = new String();
52.         }
53.         return relevansDN;
54.     }
55.     public void setRelevansDN(String relevansDN) {
56.         this.relevansDN = relevansDN;
57.     }
58.     public String getKliniskKonsekvens() {
59.         if (kliniskKonsekvens == null) {
60.             this.kliniskKonsekvens = new String();
61.         }
62.         return kliniskKonsekvens;
63.     }
64.     public void setKliniskKonsekvens(String kliniskKonsekvens) {
65.         this.kliniskKonsekvens = kliniskKonsekvens;
66.     }
67.     public String getHandtering() {
68.         if (handtering == null) {
69.             this.handtering = new String();
70.         }
71.         return handtering;
72.     }
73.     public void setHandtering(String handtering) {
74.         this.handtering = handtering;
75.     }
76. }
```

```
1.  package no.uia.ehealth;
2.  public class DrugID_ATC_CODEPair {
3.      private String dRUG_ID;
4.      private String dRUG_ATC_CODE;
5.      public DrugID_ATC_CODEPair(String dRUG_ID, String dRUG_ATC_CODE) {
6.          this.dRUG_ID = dRUG_ID;
7.          this.dRUG_ATC_CODE = dRUG_ATC_CODE;
8.      }
9.      public String getDRUG_ID() {
10.         if (dRUG_ID == null) {
11.             dRUG_ID = new String();
12.         }
13.         return dRUG_ID;
14.     }
15.     public void setDRUG_ID(String dRUG_ID) {
16.         this.dRUG_ID = dRUG_ID;
17.     }
18.     public String getDRUG_ATC_CODE() {
19.         if (dRUG_ATC_CODE == null) {
20.             dRUG_ATC_CODE = new String();
21.         }
22.         return dRUG_ATC_CODE;
23.     }
24.     public void setDRUG_ATC_CODE(String dRUG_ATC_CODE) {
25.         this.dRUG_ATC_CODE = dRUG_ATC_CODE;
26.     }
27. }
```

```
1.  package no.uia.ehealth;
2.  import java.io.BufferedInputStream;
3.  import java.io.File;
4.  import java.io.FileInputStream;
5.  import java.io.FileNotFoundException;
```

```
6.  import java.util.ArrayList;
7.  import java.util.Iterator;
8.  import java.util.List;
9.  import org.hl7.fhir.dstu3.model.Bundle;
10. import org.hl7.fhir.dstu3.model.Medication;
11. import org.hl7.fhir.dstu3.model.Bundle.BundleEntryComponent;
12. import org.hl7.fhir.dstu3.model.Extension;
13. import org.hl7.fhir.dstu3.model.Group;
14. import org.hl7.fhir.dstu3.model.Group.GroupMemberComponent;
15. import ca.uhn.fhir.context.FhirContext;
16. import ca.uhn.fhir.rest.client.api.IGenericClient;
17. import ca.uhn.fhir.rest.gclient.TokenClientParam;
18. public class CheckForInteraction {
19.     protected FhirContext ctx;
20.     protected String drugA;
21.     protected String drugB;
22.     protected String baseURL;
23.     protected Boolean flagATCCodeA = false;
24.     protected Boolean flagVareNavnA = false;
25.     protected Boolean flagSubsNameA = false;
26.     protected Boolean flagATCCodeB = false;
27.     protected Boolean flagVareNavnB = false;
28.     protected Boolean flagSubsNameB = false;
29.     protected String aTC_CODE_A = "";
30.     private String drugA_ID = "";
31.     protected String aTC  CODE  B = "";
32.     private String drugB  ID = "";
33.     private DrugID_ATC_CODEPair drugID_atc_CODE_pair_drugA;
34.     private DrugID  ATC  CODEPair drugID  atc  CODE  pair  drugB;
35.     protected InteractionDetails interaction;
36.     private Boolean isInteraction = false;
37.     public CheckForInteraction(FhirContext ctx, String drugA, String drugB, String baseURL) {
38.         this.ctx = ctx;
39.         this.drugA = drugA;
40.         this.drugB = drugB;
41.         this.baseURL = baseURL; // 1. we need to get the ATC-CODE of drugA and drugB. // The practitioner can search by ATC-
    CODE, vareNavn, or active substance name. // 1.1 Check whether drugA search keyword is ATC-
    CODE, product name (vareNavn) , // or active substance name
42.         this.drugID  atc  CODE  pair  drugA = inputCheckerA(drugA); // 1.2 Check whether drugB search keyword is ATC-
    CODE, product name (vareNavn) , // or active substance name
43.         this.drugID  atc  CODE  pair  drugB = inputCheckerB(drugB); // 2. Search for all FESTGroup instances which include Medic
    ation reference // entries for drugA // store IDs of interactions and groups for drug A
44.         List < InteractionGroupPair > interaction_group_drugA = drugGroups(this.drugID_atc_CODE_pair_drugA); // 3. Search for
    all FESTGroup instances which include Medication reference // entries for drugB // store IDs of interactions and groups for drug B

45.         List < InteractionGroupPair > interaction  group  drugB = drugGroups(this.drugID  atc  CODE  pair  drugB); // 4. Compare b
    oth lists detect interaction, if any.
46.         Iterator itrA = interaction  group  drugA.iterator();
47.         Iterator itrB = interaction  group  drugB.iterator();
48.         while (itrA.hasNext()) {
49.             InteractionGroupPair pairDrugA = (InteractionGroupPair) itrA.next();
50.             while (itrB.hasNext()) {
51.                 InteractionGroupPair pairDrugB = (InteractionGroupPair) itrB.next(); // FEST interaction business logic // find the two d
    ifferent groups which have the same interaction id
52.                 if (!(pairDrugA.getGroup  ID().equals(pairDrugB.getGroup  ID())) && (pairDrugA.getInteraction  ID().equals(pairDrug
    B.getInteraction_ID()))) { // get interaction details for this group id
53.                     String id = "";
54.                     String relevansV = "";
55.                     String relevansDN = "";
56.                     String kliniskKonsekvens = "";
57.                     String handtering = "";
58.                     IGenericClient client = ctx.newRestfulGenericClient(baseURL);
59.                     Group theOne = client.read().resource(Group.class).withId(pairDrugA.getGroup_ID()).execute(); // get interaction ID

60.                     List < Extension > resourceExtsInteractionID = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition
    /interactionID");
61.                     if (resourceExtsInteractionID != null) {
```

```
62.                     Extension interactionID = resourceExtsInteractionID.get(0);
63.                     id = interactionID.getValue().toString();
64.                 } // get Interaction Relevans V attribute
65.                 List < Extension > resourceExtsInteractionRelevansV = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureD
    efinition/interactionRelevansV");
66.                 if (resourceExtsInteractionRelevansV != null) {
67.                     Extension interactionRelevansV = resourceExtsInteractionRelevansV.get(0);
68.                     relevansV = interactionRelevansV.getValue().toString();
69.                 } // get Interaction Relevans DN attribute
70.                 List < Extension > resourceExtsInteractionRelevansDN = theOne.getExtensionsByUrl("http://ehealth.uia.no/Structure
    Definition/interactionRelevansDN");
71.                 if (resourceExtsInteractionRelevansDN != null) {
72.                     Extension interactionRelevansDN = resourceExtsInteractionRelevansDN.get(0);
73.                     relevansDN = interactionRelevansDN.getValue().toString();
74.                 } // get Interaction KliniskKonsekvens
75.                 List < Extension > resourceExtsInteractionKliniskKonsekvens = theOne.getExtensionsByUrl("http://ehealth.uia.no/Str
    uctureDefinition/interactionKliniskKonsekvens");
76.                 if (resourceExtsInteractionKliniskKonsekvens != null) {
77.                     Extension interactionKliniskKonsekvens = resourceExtsInteractionKliniskKonsekvens.get(0);
78.                     kliniskKonsekvens = interactionKliniskKonsekvens.getValue().toString();
79.                 } // get Interaction Handtering
80.                 List < Extension > resourceExtsInteractionHandtering = theOne.getExtensionsByUrl("http://ehealth.uia.no/StructureD
    efinition/interactionHandtering");
81.                 if (resourceExtsInteractionHandtering != null) {
82.                     Extension interactionHandtering = resourceExtsInteractionHandtering.get(0);
83.                     handtering = interactionHandtering.getValue().toString();
84.                 } // set interaction details
85.                 this.interaction = new InteractionDetails(id, relevansV, relevansDN, kliniskKonsekvens, handtering, drugA, drugB);
86.                 this.isInteraction = true;
87.                 break;
88.             }
89.         }
90.     }
91. }
92. private List < InteractionGroupPair > drugGroups(DrugID_ATC_CODEPair pair) {
93.     List < InteractionGroupPair > interaction_group_for_drug = new ArrayList < InteractionGroupPair > ();
94.     IGenericClient clientDrugAGroupIDs = ctx.newRestfulGenericClient(baseURL);
95.     Bundle resultGroups = clientDrugAGroupIDs.search().forResource(Group.class).where(Group.MEMBER.hasId(pair.getDRU
    G_ID())).returnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
96.     if (resultGroups != null) {
97.         if (resultGroups.getTotal() > 0) {
98.             for (BundleEntryComponent e: resultGroups.getEntry()) {
99.                 String groupID = "";
100.                String interactionID = "";
101.                if (e.getResource() instanceof Group) { // get group id
102.                    Group g = (Group) e.getResource();
103.                    groupID = g.getId(); // get interaction id
104.                    List < Extension > resourceExts = g.getExtensionsByUrl("http://ehealth.uia.no/StructureDefinition/interactionID");

105.                    Extension exti = resourceExts.get(0);
106.                    interactionID = exti.getValue().toString();
107.                    InteractionGroupPair p = new InteractionGroupPair(interactionID, groupID);
108.                    interaction_group_for_drug.add(p);
109.                }
110.            }
111.        }
112.    }
113.    return interaction_group_for_drug;
114. }
115. private DrugID_ATC_CODEPair inputCheckerA(String drug) { // 1.1.1 ATC-CODE checker
116.     IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
117.     Bundle resultATCV = clientATCV.search().forResource(Medication.class).where(Medication.CODE.exactly().code(drug)).re
    turnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
118.     if (resultATCV != null) {
119.         if (resultATCV.getTotal() == 0) {
120.             this.flagATCCodeA = false;
121.         } else {
```

```
122.             this.flagATCCodeA = true;
123.             if (resultATCV.getEntry().get(0).getResource() != null) {
124.                 if (resultATCV.getEntry().get(0).getResource() instanceof Medication) {
125.                     Medication m = (Medication) resultATCV.getEntry().get(0).getResource();
126.                     this.aTC_CODE_A = drug;
127.                     this.drugA_ID = m.getId();
128.                     this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), drug);
129.                 }
130.             }
131.         }
132.     }
133.     if (!flagATCCodeA) { // 1.1.2 Legemiddel varenavn checker // Note that we have defined the search parameter legemiddelVar
    enavn in // FESTMedicationHapiFHIRController.java // use the added search parameter
134.         IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
135.         Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class).where(new TokenClientParam("legemid
    delVarenavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
136.         if (resultVareNavn != null) {
137.             if (resultVareNavn.getTotal() == 0) {
138.                 this.flagVareNavnA = false;
139.             } else {
140.                 this.flagVareNavnA = true;
141.                 if (resultVareNavn.getEntry().get(0).getResource() != null) {
142.                     if (resultVareNavn.getEntry().get(0).getResource() instanceof Medication) {
143.                         Medication m = (Medication) resultVareNavn.getEntry().get(0).getResource();
144.                         this.drugA_ID = m.getId();
145.                         this.aTC_CODE_A = m.getCode().getCoding().get(0).getCode();
146.                         this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
147.                     }
148.                 }
149.             }
150.         }
151.     }
152.     if (!this.flagATCCodeA && !this.flagVareNavnA) { // 1.1.3 active substance name checker // Note that we have defined the
    search parameter atcNavn in // FESTMedicationHapiFHIRController.java // use the added search parameter
153.         IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
154.         Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class).where(new TokenClientParam("atc
    Navn").exactly().code(drug)).returnBundle(Bundle.class).execute();
155.         if (resultSubstanceName != null) {
156.             if (resultSubstanceName.getTotal() == 0) {
157.                 this.flagSubsNameA = false;
158.             } else {
159.                 this.flagSubsNameA = true;
160.                 if (resultSubstanceName.getEntry().get(0).getResource() != null) {
161.                     if (resultSubstanceName.getEntry().get(0).getResource() instanceof Medication) {
162.                         Medication m = (Medication) resultSubstanceName.getEntry().get(0).getResource();
163.                         this.drugA_ID = m.getId();
164.                         this.aTC_CODE_A = m.getCode().getCoding().get(0).getCode();
165.                         this.drugID_atc_CODE_pair_drugA = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
166.                     }
167.                 }
168.             }
169.         }
170.     }
171.     return drugID_atc_CODE_pair_drugA;
172. }
173.  private DrugID_ATC_CODEPair inputCheckerB(String drug) { // 1.1.1 ATC-CODE checker
174.     IGenericClient clientATCV = ctx.newRestfulGenericClient(baseURL);
175.     Bundle resultATCV = clientATCV.search().forResource(Medication.class).where(Medication.CODE.exactly().code(drug)).re
    turnBundle(org.hl7.fhir.dstu3.model.Bundle.class).execute();
176.     if (resultATCV != null) {
177.         if (resultATCV.getTotal() == 0) {
178.             this.flagATCCodeB = false;
179.         } else {
180.             this.flagATCCodeB = true;
181.             if (resultATCV.getEntry().get(0).getResource() != null) {
```

```
182.                 if (resultATCV.getEntry().get(0).getResource() instanceof Medication) {
183.                     Medication m = (Medication) resultATCV.getEntry().get(0).getResource();
184.                     this.aTC_CODE_B = drug;
185.                     this.drugB_ID = m.getId();
186.                     this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), drug);
187.                 }
188.             }
189.         } // 1.1.2 Legemiddel varenavn checker // Note that we have defined the search parameter legemiddelVarenavn in // FEST
    MedicationHapiFHIRController.java // use the added search parameter
190.     }
191.     if (!flagATCCodeB) {
192.         IGenericClient clientVareNavn = ctx.newRestfulGenericClient(baseURL);
193.         Bundle resultVareNavn = clientVareNavn.search().forResource(Medication.class).where(new TokenClientParam("legemid
    delVarenavn").exactly().code(drug)).returnBundle(Bundle.class).execute();
194.         if (resultVareNavn != null) {
195.             if (resultVareNavn.getTotal() == 0) {
196.                 this.flagVareNavnB = false;
197.             } else {
198.                 this.flagVareNavnB = true;
199.                 if (resultVareNavn.getEntry().get(0).getResource() != null) {
200.                     if (resultVareNavn.getEntry().get(0).getResource() instanceof Medication) {
201.                         Medication m = (Medication) resultVareNavn.getEntry().get(0).getResource();
202.                         this.drugB_ID = m.getId();
203.                         this.aTC_CODE_B = m.getCode().getCoding().get(0).getCode();
204.                         this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
205.                     }
206.                 }
207.             }
208.         }
209.     } // 1.1.3 active substance name checker // Note that we have defined the search parameter atcNavn in // FESTMedicationHap
    iFHIRController.java // use the added search parameter
210.     if (!this.flagATCCodeB && !this.flagVareNavnB) {
211.         IGenericClient clientATCDN = ctx.newRestfulGenericClient(baseURL);
212.         Bundle resultSubstanceName = clientATCDN.search().forResource(Medication.class).where(new TokenClientParam("atc
    Navn").exactly().code(drug)).returnBundle(Bundle.class).execute();
213.         if (resultSubstanceName != null) {
214.             if (resultSubstanceName.getTotal() == 0) {
215.                 this.flagSubsNameB = false;
216.             } else {
217.                 this.flagSubsNameB = true;
218.                 if (resultSubstanceName.getEntry().get(0).getResource() != null) {
219.                     if (resultSubstanceName.getEntry().get(0).getResource() instanceof Medication) {
220.                         Medication m = (Medication) resultSubstanceName.getEntry().get(0).getResource();
221.                         this.drugB_ID = m.getId();
222.                         this.aTC_CODE_B = m.getCode().getCoding().get(0).getCode();
223.                         this.drugID_atc_CODE_pair_drugB = new DrugID_ATC_CODEPair(m.getId(), m.getCode().getCoding().get(0)
    .getCode());
224.                     }
225.                 }
226.             }
227.         }
228.     }
229.     return drugID_atc_CODE_pair_drugB;
230.   }
231.   public String getATC_CODE_A() {
232.     if (aTC_CODE_A == null) {
233.         aTC_CODE_A = new String();
234.     }
235.     return aTC_CODE_A;
236.   }
237.   public void setATC_CODE_A(String aTC_CODE_A) {
238.     this.aTC_CODE_A = aTC_CODE_A;
239.   }
240.   public String getATC_CODE_B() {
241.     if (aTC_CODE_B == null) {
242.         aTC_CODE_B = new String();
```

```
243.     }
244.     return aTC_CODE_B;
245.   }
246.   public void setATC_CODE_B(String aTC_CODE_B) {
247.     this.aTC_CODE_B = aTC_CODE_B;
248.   }
249.   public Boolean getFlagATCCodeA() {
250.     if (flagATCCodeA == null) {
251.       flagATCCodeA = false;
252.     }
253.     return flagATCCodeA;
254.   }
255.   public void setFlagATCCodeA(Boolean flagATCCodeA) {
256.     this.flagATCCodeA = flagATCCodeA;
257.   }
258.   public Boolean getFlagVareNavnA() {
259.     if (flagVareNavnA == null) {
260.       flagVareNavnA = false;
261.     }
262.     return flagVareNavnA;
263.   }
264.   public void setFlagVareNavnA(Boolean flagVareNavnA) {
265.     this.flagVareNavnA = flagVareNavnA;
266.   }
267.   public Boolean getFlagSubsNameA() {
268.     if (flagSubsNameA == null) {
269.       flagSubsNameA = false;
270.     }
271.     return flagSubsNameA;
272.   }
273.   public void setFlagSubsNameA(Boolean flagSubsNameA) {
274.     this.flagSubsNameA = flagSubsNameA;
275.   }
276.   public Boolean getFlagATCCodeB() {
277.     if (flagATCCodeB == null) {
278.       flagATCCodeB = false;
279.     }
280.     return flagATCCodeB;
281.   }
282.   public void setFlagATCCodeB(Boolean flagATCCodeB) {
283.     this.flagATCCodeB = flagATCCodeB;
284.   }
285.   public Boolean getFlagVareNavnB() {
286.     if (flagVareNavnB == null) {
287.       flagVareNavnB = false;
288.     }
289.     return flagVareNavnB;
290.   }
291.   public void setFlagVareNavnB(Boolean flagVareNavnB) {
292.     this.flagVareNavnB = flagVareNavnB;
293.   }
294.   public Boolean getFlagSubsNameB() {
295.     if (flagSubsNameB) {
296.       flagSubsNameB = false;
297.     }
298.     return flagSubsNameB;
299.   }
300.   public void setFlagSubsNameB(Boolean flagSubsNameB) {
301.     this.flagSubsNameB = flagSubsNameB;
302.   }
303.   public InteractionDetails getInteraction() {
304.     return interaction;
305.   }
306.   public void setInteraction(InteractionDetails interaction) {
307.     this.interaction = interaction;
308.   }
309.   public Boolean getIsInteraction() {
```

```
310.        if (isInteraction == null) {
311.            isInteraction = false;
312.        }
313.        return isInteraction;
314.    }
315.    public void setIsInteraction(Boolean isInteraction) {
316.        this.isInteraction = isInteraction;
317.    }
318. }
```

```
1.  <% @taglib uri = "http://java.sun.com/jsp/jstl/core"
2.  prefix = "c" %> < html lang = "no" > < head > < meta charset = "utf-8" > < meta name = "viewport"
3.  content = "width=device-width, initial-scale=1, shrink-to-fit=no" > < meta name = "description"
4.  content = "" > < meta name = "author"
5.  content = "" > < title > FEST on FHIR - Interaksjonsanalyse < /title> <!-- Bootstrap core CSS -->
6.  <link rel="stylesheet" href="../css/bootstrap.min.css">
7.
8.  <!-- Custom styles for this template -->
9.  <link href="../css/template.css" rel="stylesheet">
10.
11. </head>
12.
13. <body>
14.
15.     <!-- Navigation -->
16.     <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
17.         <div class="container">
18.             <a class="navbar-brand" href="#">FEST on FHIR -
19.                 Interaksjonsanalyse</a>
20.             <button class="navbar-toggler" type="button" data-toggle="collapse"
21.                 data-target="#navbarResponsive" aria-controls="navbarResponsive"
22.                 aria-expanded="false" aria-label="Toggle navigation">
23.                 <span class="navbar-toggler-icon"></span>
24.             </button>
25.             <div class="collapse navbar-collapse" id="navbarResponsive"></div>
26.         </div>
27.     </nav>
28.
29.     <!-- Page Content -->
30.     <div class="container">
31.
32.         <!-- Heading Row -->
33.         <div class="row my-4">
34.             <div class="col-lg-8">
35.                 <img id="logo" class="img-fluid rounded"
36.                     src="../img/hapi_fhir_banner.jpg" alt="">
37.             </div>
38.             <!-- /.col-lg-8 -->
39.             <div id="sideBox" class="col-lg-4">
40.                 <h1>FEST on FHIR - Interaksjonsanalyse</h1>
41.                 <p>
42.                     Here we use our back-end component to check for drug-drug
43.                     interaction. We send a request to our <a
44.                         href="http://localhost:8081/fest/baseDstu3">HAPI-FHIR RESTful
45.                         server</a> and display the interaction result, if any. <br>
46.                 <hr>
47.                 This is a Java web application which can be a component of a
48.                 clinical decision support system where it can be easily integrated.
49.                 For example, a practitioner can use this feature to detect drug-drug
50.                 interaction before prescribing medicines.
51.                 </p>
```

```
52.
53.            </div>
54.            <!-- /.col-md-4 -->
55.        </div>
56.        <!-- /.row -->
57.
58.        <!-- Call to Action Well -->
59.        <!--  <div class="card text-white bg-secondary my-4 text-center"> -->
60.        <div class="text-center">
61.            <div class="card-body">
62.                <!--Search Form -->
63.                <form action="/interaction" method="get" id="detectInteractionForm"
64.                    role="form">
65.                    <input type="hidden" id="searchAction" name="searchAction"
66.                        value="checkDDI">
67.                    <div class="form-group col-xs-5">
68.                        <input id="drugA" type="text" name="drugA" id="drugA"
69.                            class="form-control" required="true"
70.                            placeholder="Skriv inn varenavn, virkestoff eller ATC kode" />
71.                    </div>
72.
73.                    <div class="form-group col-xs-5">
74.                        <input id="drugB" type="text" name="drugB" id="drugB"
75.                            class="form-control" required="true"
76.                            placeholder="Skriv inn varenavn, virkestoff eller ATC kode" />
77.                    </div>
78.                    <button type="submit" class="btn btn-primary  btn-md">
79.                        <span class=""></span> Analyser
80.                    </button>
81.                    <button type="reset" class="btn btn-md">
82.                        <span class=""></span> Nullstill
83.                    </button>
84.                    <br></br> <br></br>
85.                </form>
86.            </div>
87.        </div>
88.
89.        <!-- Content Row -->
90.        <c:if test="${not empty message}">
91.            <div class="row">
92.                <div id="notEmptyMessage" class="alert alert-success">
93.                    ${message}</div>
94.            </div>
95.        </c:if>
96.        <c:choose>
97.            <c:when test="${not empty interactionResult}">
98.
99.                <div class="row">
100.                    <div class="col-md-12 mb-12">
101.                        <div class="alert alert-danger" role="alert">
102.                            <h3 class="alert-heading">Interaksjon ble funnet mellom
103.                                ${interactionResult.drugA} og ${interactionResult.drugB}</h3>
104.                        </div>
105.                    </div>
106.                </div>
107.                <div class="row">
108.                    <div class="col-md-4 mb-4">
109.                        <div class="card h-100">
110.                            <div class="card-body">
111.                                <h4 class="card-title">Klassifisering av interaksjon</h4>
112.                                <p class="card-text">${interactionResult.relevansDN}</p>
113.                            </div>
114.
115.                        </div>
116.                    </div>
117.                    <!-- /.col-md-4 -->
118.
```

```
119.                 <div class="col-md-4 mb-4">
120.                     <div class="card h-100">
121.                         <div class="card-body">
122.                             <h4 class="card-title">Klinisk konsekvens</h4>
123.                             <p class="card-text">${interactionResult.kliniskKonsekvens}</p>
124.                         </div>
125.
126.                     </div>
127.                 </div>
128.                 <!-- /.col-md-4 -->
129.                 <div class="col-md-4 mb-4">
130.                     <div class="card h-100">
131.                         <div class="card-body">
132.                             <h4 class="card-title">Håndtering</h4>
133.                             <p class="card-text">${interactionResult.handtering}</p>
134.                         </div>
135.
136.                     </div>
137.                 </div>
138.                 <!-- /.col-md-4 -->
139.             </div>
140.         </c:when>
141.         <c:otherwise>
142.             <div class="row">
143.                 <div class="col-md-12 mb-12">
144.                     <div class="alert alert-info">Ingen relevante interaksjoner
145.                         ble funnet!</div>
146.                 </div>
147.             </div>
148.         </c:otherwise>
149.     </c:choose>
150.     <!-- /.row -->
151.
152.
153.     <!-- /.row -->
154.
155.   </div>
156.   <!-- /.container -->
157.
158.   <!-- Footer -->
159.   <footer class="py-5 bg-dark">
160.     <div class="container">
161.       <p class="m-0 text-center text-white">Copyright &copy; FEST on
162.         FHIR. Master thesis 2018, Islam Al Khaldi. UiA</p>
163.     </div>
164.     <!-- /.container -->
165.   </footer>
166.
167.   <!-- Bootstrap core JavaScript -->
168.   <script src="../js/bootstrap.min.js"></script>
169.   <script src="../js/jquery.min.js"></script>
170.
171. </body>
172.
173. </html>
```

```
1.    < html > < body > < h2 > Practitioner < /h2> < jsp: forward page = "/interaction" / > < /body> < /html>
```

7.3 Appendix C: FEST on FHIR Profiles and Extensions

All the profiles and extension defined in this thesis can be downloaded from the following URL:

https://simplifier.net/FESTonFHIR?fhirVersion=STU3

7.4 Appendix D: FEST Java Classes Model

FEST Java classes model can be downloaded from the following GitHub repository URL:

https://github.com/islamalkhaldi/FESTonFHIR