LM-32
Digital Health and Bioinformatic Engineering

Digital Health Systems and Applications
Academic year 2022 / 2023

Project Report

Project 3

Authors	Ceglia Salvatore Ferrara Luigina Gargiulo Anna Maruotto Ida	0622900145 0622900144 0622900139 0622900135
Group Number	3	

Dipartimento di Ingegneria dell’Informazione e Elettrica e Matematica Applicata (DIEM)
Università degli Studi di Salerno
Via Giovanni Paolo II, 132 - 84084 Fisciano (SA)

Contents

Project 3 ... **1**

Hospital Information System ... **3**

Introduction ... **3**
Project Description ... 3
State of The Art ... 3

Solution ... **6**
Requirements ... 6
Architecture ... 7
Data ... 7
UML ... 7
Use Cases ... 11
Patient Admission ... 11
Order Registration ... 14
MRI ... 15
Import CSV ... 19

Application ... **19**
User Interface ... 23
CSV Mapping ... 23
Reading a CSV ... 23
Patient ... 24
Encounter ... 25
Providers ... 25
Organizations ... 26
Payers ... 26
Payer Transitions ... 27
Observation ... 27
Imaging Studies ... 28
Immunizations ... 28
Medications ... 29
Procedures ... 29
CarePlans ... 30
Fhir Server Interaction ... 32
Upload a Resource on the FHIR Server ... 32
Updating a Resource on the FHIR Server ... 32
Getting a Resource from the FHIR Server ... 33
DICOM attachments ... 34
DICOM files' structure in dataset ... 34
Visualization ... 35
Fhir mapping ... 35

Hospital Information System

Introduction

Project Description

The goal of the project is the realization of different applications to mimic processes of a Hospital Information System using the Coherent Dataset.

The objective is to interpret the information contained in the CSV files and translate it into FHIR records. These records need to be loaded on a FHIR server that needs to be installed and configured.

The possible applications can simulate:

1. Patient admission and registration;
2. Order entry for analysis or investigation;
3. Execution of an MRI analysis and loading of the corresponding DICOM file;
4. Execution of other analysis (blood, genomics, and so on).

All the applications need to update the patient FHIR record on the server.

If possible, define a BPMN process and execute it through a BPMN Engine to show the progress of the process.

State of The Art

The best Hospital Management softwares[1] available on the US market are **MediXcel**, **AHMS**, **Halemind**, **Caresoft**, and **GeniPluse**. They help to automate and simplify the process of hospital management.

1. The **MediXcelTM[2]** platform caters to the ever-evolving clinical and business software needs of clinics of all sizes, hospital OPDs & wellness providers. The platform includes solutions which provide for electronic medical records, practice management, business practice management, including Accounting and Inventory, patient engagement and relationship management, laboratory services, information management, disease management systems, disease surveillance, systems and healthcare analytics. **MediXcel** is a simple browser-based software deployable on Local Area Network or the Cloud, as per costumer's need, that provide

[1] Source: https://www.softwaresuggest.com/us/hospital-management-software
[2] MediXcelTM: https://www.plus91online.com/about-medixcel/

accessible, on time and patient centric care with managed workflow and reminders, manage multi-clinic access, accounts, inventory, memberships and patient appointments.

2. **Advanced Health Management System** [3](**AHMS**) is a robust electronic health record (EHR) and practice management software designed to meet the unique needs of small and midsize healthcare practices. The software can be deployed either in the cloud or on-premise. It is a HIPAA (Health Insurance Portability and Accountability Act) and HCFA (Health Care Financing Administration), compliant system that provides reports and enables clinicians to complete manual and electronic insurance processing. Key features of AHMS include e-prescribing, billing, lab, scheduling, electronic and manual insurance billing, clinical and RX interfaces, and patient demographics. The platform also features examination records, data records, optical scanning, integration with pharmacies & diagnostic devices, and a Graphical User Interface (GUI) for the Patients.
3. **Halemind**[4] is a cloud-based hospital management solution, which helps medical colleges, pharmacies, laboratories and clinics streamline processes related to patient registration, electronic health records (EHR), appointment scheduling and more. Key features of Halemind include automated billing, electronic prescriptions, customizable templates, digital signatures and queue management. Pharmacies can view customer's history, track orders, receive purchase invoices and manage suppliers in real-time. Additionally, hospitals can schedule surgeries, generate invoices, collect payments, track patients' samples for testing and visualize data according to requirements.

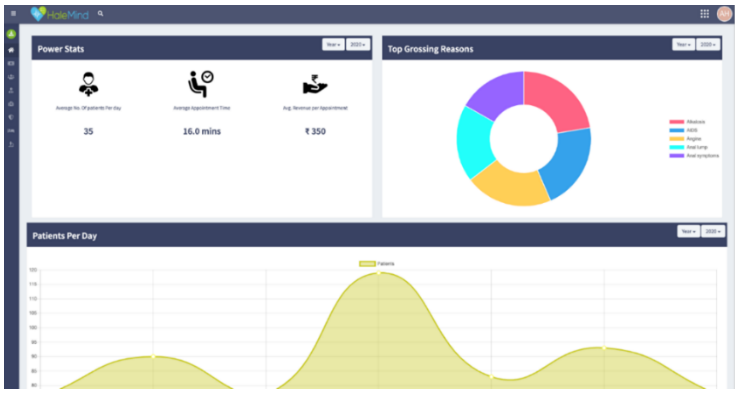


Figure 1: Halemind Dashboard. Source [https://www.softwareadvice.com/medical/halemind-profile/]

4. **CARESOFT Clinic Information System**[5] (**CIS**) helps healthcare organizations adopting Information Technology in the automation of the Clinics management process to face a mix of challenges such as budgetary pressures, cost efficiency, regulatory changes, achieving their goals with improving health care standards. Caresoft offers a unified suite of digital solutions

[3] AHMS: https://www.medicalrecords.com/mrcbase/emr/advanced-health-management-systems-ahms
[4] Halemind: https://www.softwareadvice.com/medical/halemind-profile/
[5] Caresoft: https://caresoft.co.in/clinic-information-system/

proven to streamline administration, reduce costs, control revenue leakages and enhance patient safety and increase the overall profitability of clinics or physician practices. Caresoft clinic information system is a healthcare application that manages the day-to-day activities of medical practice. The medical staff can enter patient details, arrange meetings and appointments, keep track of insurance payers, keep track of billing records, and generate medical reports. Clinic software provides intuitive and dynamic powerful features such as patient scheduling, office scheduling, patient billing, appointment reminders.

Figure 2: Caresoft CIS. Source[https://caresoft.co.in/clinic-information-system/]

5. **GeniPluse HMS** [6]is install based, the most robust, paperless and comprehensive solution for the hospital. The cost is affordable. By this hospital management software, you can Streamline the workflow, manage patient's data, OPD, IPD to doctor's schedule with very accurately. Its Desktop based solution for greater security. Due to easy-to-use interface, it reduces user learning curve & minimizes training cost. Tabbed based interface provides faster accessibility while working multiple interfaces concurrently. Reporting is available on both desktop as well as web, tablets & Mobile devices.

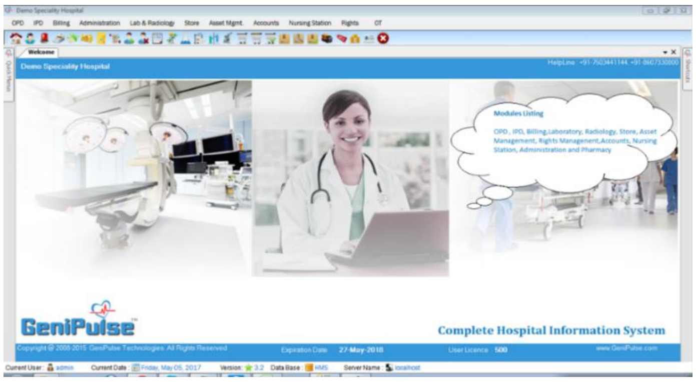

Demo Speciality Hospital
Modules Listing
OPD, IPD, Billing,Laboratory, Radiology, Store, Asset Management, Rights Management,Accounts, Nursing Station, Administration and Pharmacy
GeniPulse
Complete Hospital Information System

Figure 3: GeniPluse HMS Welcome Screen. Source[http://genipulse.com/Hospital-Management-System]

[6] GeniPluse: http://genipulse.com/Hospital-Management-System

Solution

Requirements

Functional Requirements:

- **FR001** - The application must map all the records of the CSV files in the corresponding FHIR Resources, compatible with the R4 version.
- **FR002** - For each CSV file, all fields must be suitably mapped according to the available attributes of the corresponding FHIR Resource.
- **FR003** - The application must allow the insertion of a new patient in the system, whose resource must be loaded on the server.
- **FR004** - The application must allow to search among all patients stored on the server and view personal details of each one.
- **FR005** - The application must allow the insertion of a new medical service request for a patient registered on the system.
- **FR006** - The application must allow to search among medical service request registered on the system and view its details.
- **FR007** - The application must allow load the results of a MRI study, to show the produced DICOM files and automatically compile, save, and show a medical report related to that study.

Non-Functional Requirements:

- **NFR001** - The FHIR Resources must be created using HAPI FHIR API[7].
- **NFR002** - The FHIR Resources must be compliant with the specifications of US Core Profile[8] of the HL7 Standard.
- **NFR003** - Mapping from dataset entities to the FHIR data model, references between records must be maintained between resources, using entity identifiers and the referencing mechanism offered by the HAPI FHIR API.
- **NFR004** - All the codable information available on the entities records of the dataset must be encoded according to SNOMED[9], LOINC[10] and NUCC[11] systems.
- **NFR005** - The client part of the application must use HAPI FHIR client to fetch from or store resources to an external server.
- **NFR006** - The server part of the application must be based on the HAPI JPA/Database Server[12] to deploy a fully functional FHIR server.

[7] Source: https://hapifhir.io/hapi-fhir/apidocs/hapi-fhir-structures-r4/
[8] US Core Implementation Guide: https://hl7.org/fhir/us/core/
[9] SNOMED: http://snomed.info/sct
[10] LOINC: https://loinc.org/
[11] NUCC: https://www.nucc.org/
[12] HAPI FHIR JPA Starter Server: https://github.com/hapifhir/hapi-fhir-jpaserver-starter

Architecture

The architecture chosen for the application is Client-Server, that is a computing model in which the server hosts, delivers, and manages most of the resources and services requested by the client.

The **Client** is a Java-based application with Maven, implementing the MVC pattern (Model-View-Controller), using JavaFX platform and SceneBuilder as tool to develop the application GUI.

The **Server** used is configured starting from the HAPI FHIR JPA Starter Server[13] and installed through a Docker Container on a Virtual Machine provided by the Department, accessible with a VPN connection.

Data

The goal of the application is to interpret data contained in the folder *csv* of the Coherent Data Set[14]. This dataset is composed of several CSV Files[15] representing FHIR resources and some of their fields. The mapping is neither one-to-one, nor a full mapping. Not all the csv fields can be mapped using pure FHIR resources, but all "must have" elements have been filled through csv fields or through derivation from other fields. All elements that are denoted in FHIR as "should support" have been mapped if present in the CSVs.

Moreover, the dataset also contains a *dicom* folder containing DICOM files artificially created, and connected with imaging studies of the patients. These files have been added to the server as a part of one of the resources.

UML

Here is reported UML diagram of Application architecture. It is divided by package and not reporting Resources fields described next, neither enumeration's codes. *HIS_Project* package is represented with pseudo-UML and its graph is aimed to better describe the interaction within classes. Package *resources*' diagram just show the hierarchy of resources: all of them implements *interface BaseResource* and there are two different kinds of Organization.

[13] https://github.com/hapifhir/hapi-fhir-jpaserver-starter.git
[14] Walonoski J, Hall D, Bates KM, Farris MH, Dagher J, Downs ME, Sivek RT, Wellner B, Gregorowicz A, Hadley M, Campion FX, Levine L, Wacome K, Emmer G, Kemmer A, Malik M, Hughes J, Granger E, Russell S. The "Coherent Data Set": Combining Patient Data and Imaging in a Comprehensive, Synthetic Health Record. Electronics. 2022; 11(8):1199. https://doi.org/10.3390/electronics11081199
[15] CSV File Data Dictionary: https://github.com/synthetichealth/synthea/wiki/CSV-File-Data-Dictionary

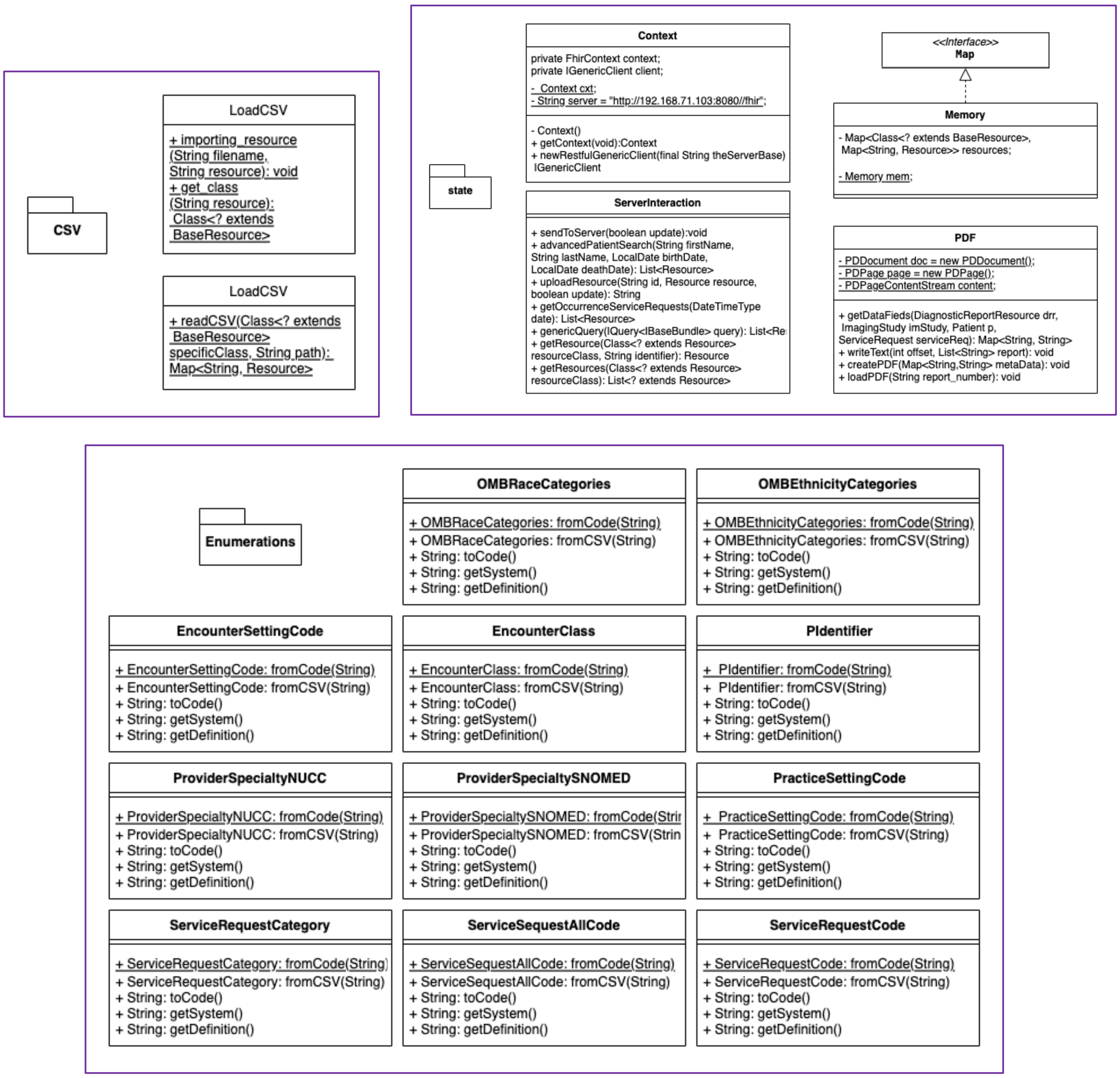

CSV
LoadCSV
+ importing_resource (String filename, String resource): void
+ get_class (String resource): Class<? extends BaseResource>
LoadCSV
+ readCSV(Class<? extends BaseResource> specificClass, String path): Map<String, Resource>
state
Context
private FhirContext context;
private IGenericClient client;
- Context cxt;
- String server = "http://192.168.71.103:8080//fhir";
- Context()
+ getContext(void):Context
+ newRestfulGenericClient(final String theServerBase) IGenericClient
ServerInteraction
+ sendToServer(boolean update):void
+ advancedPatientSearch(String firstName, String lastName, LocalDate birthDate, LocalDate deathDate): List<Resource>
+ uploadResource(String id, Resource resource, boolean update): String
+ getOccurrenceServiceRequests(DateTimeType date): List<Resource>
+ genericQuery(IQuery<IBaseBundle> query): List<Re
+ getResource(Class<? extends Resource> resourceClass, String identifier): Resource
+ getResources(Class<? extends Resource> resourceClass): List<? extends Resource>
<<Interface>>
Map
Memory
- Map<Class<? extends BaseResource>, Map<String, Resource>> resources;
- Memory mem;
PDF
- PDDocument doc = new PDDocument();
- PDPage page = new PDPage();
- PDPageContentStream content;
+ getDataFieds(DiagnosticReportResource drr, ImagingStudy imStudy, Patient p, ServiceRequest serviceReq): Map<String, String>
+ writeText(int offset, List<String> report): void
+ createPDF(Map<String,String> metaData): void
+ loadPDF(String report_number): void
Enumerations
OMBRaceCategories
+ OMBRaceCategories: fromCode(String)
+ OMBRaceCategories: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
OMBEthnicityCategories
+ OMBEthnicityCategories: fromCode(String)
+ OMBEthnicityCategories: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
EncounterSettingCode
+ EncounterSettingCode: fromCode(String)
+ EncounterSettingCode: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
EncounterClass
+ EncounterClass: fromCode(String)
+ EncounterClass: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
PIdentifier
+ PIdentifier: fromCode(String)
+ PIdentifier: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
ProviderSpecialtyNUCC
+ ProviderSpecialtyNUCC: fromCode(String)
+ ProviderSpecialtyNUCC: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
ProviderSpecialtySNOMED
+ ProviderSpecialtySNOMED: fromCode(Strir
+ ProviderSpecialtySNOMED: fromCSV(Strin
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
PracticeSettingCode
+ PracticeSettingCode: fromCode(String)
+ PracticeSettingCode: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
ServiceRequestCategory
+ ServiceRequestCategory: fromCode(String)
+ ServiceRequestCategory: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
ServiceSequestAllCode
+ ServiceSequestAllCode: fromCode(String)
+ ServiceSequestAllCode: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()
ServiceRequestCode
+ ServiceRequestCode: fromCode(String)
+ ServiceRequestCode: fromCSV(String)
+ String: toCode()
+ String: getSystem()
+ String: getDefinition()

Figure 4: Utility classes UML: packages csv, enumerations and state

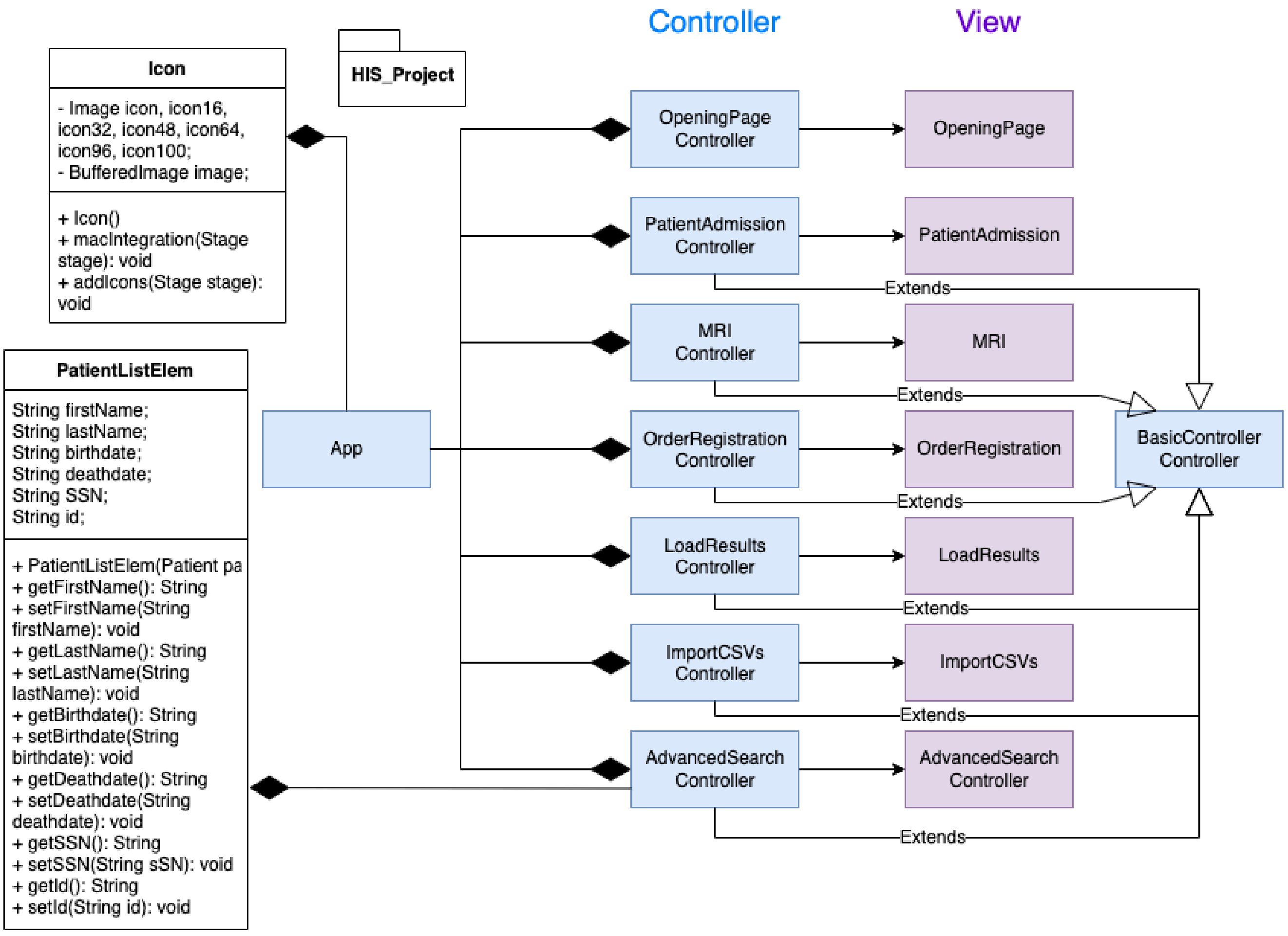


Figure 5: Pseudo-UML showing MVC architecture of client with its utility classes, controllers (blue) and fxml (purple)

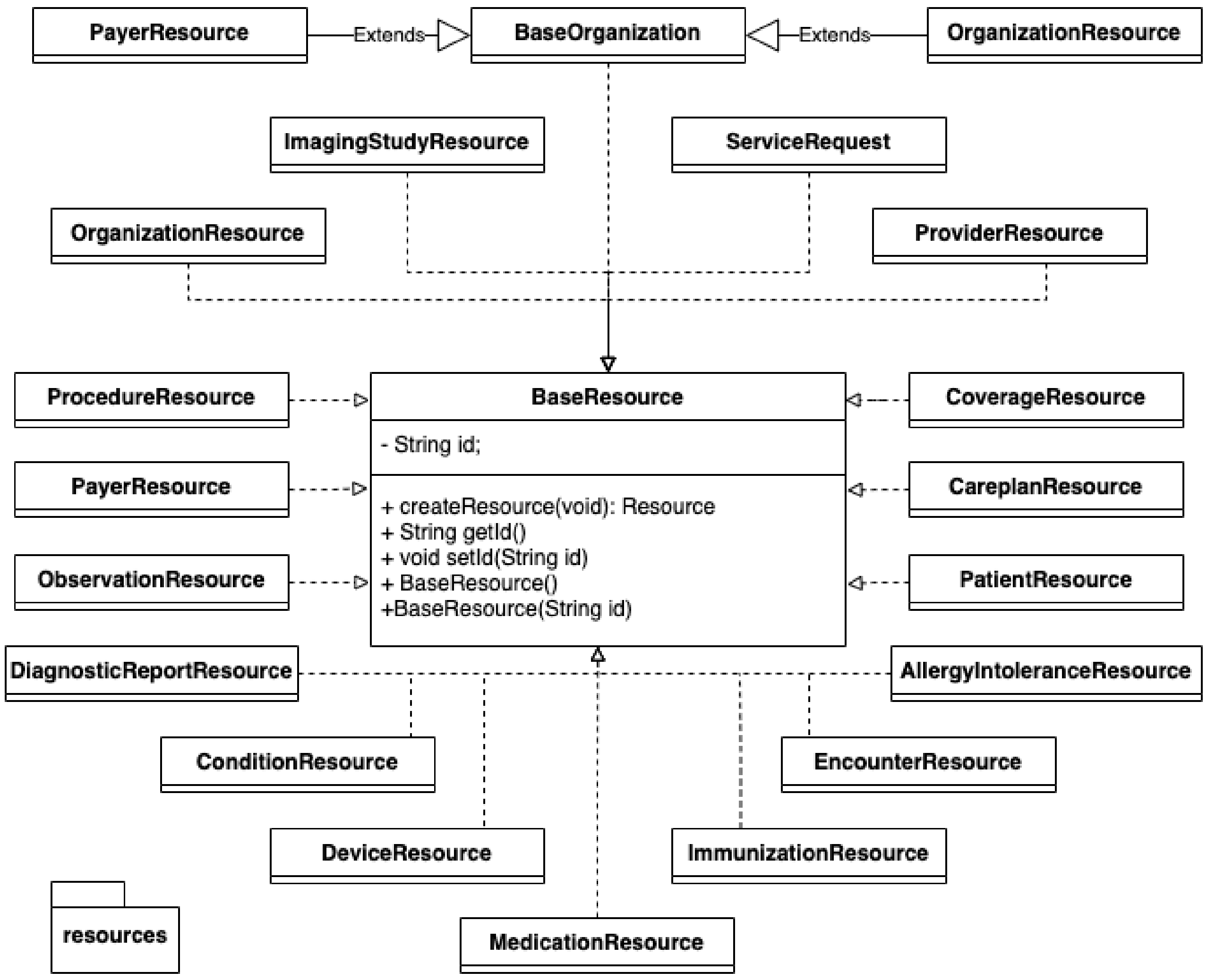


Figure 6: hierarchy of resources used to map from csv/gui to FHIR resources

Use Cases

1. On start, the application displays a logo and the possibility to choose between four different hospital processes:
    a. Patient admission;
    b. Order registration;
    c. MRI execution;
    d. Import CSV.

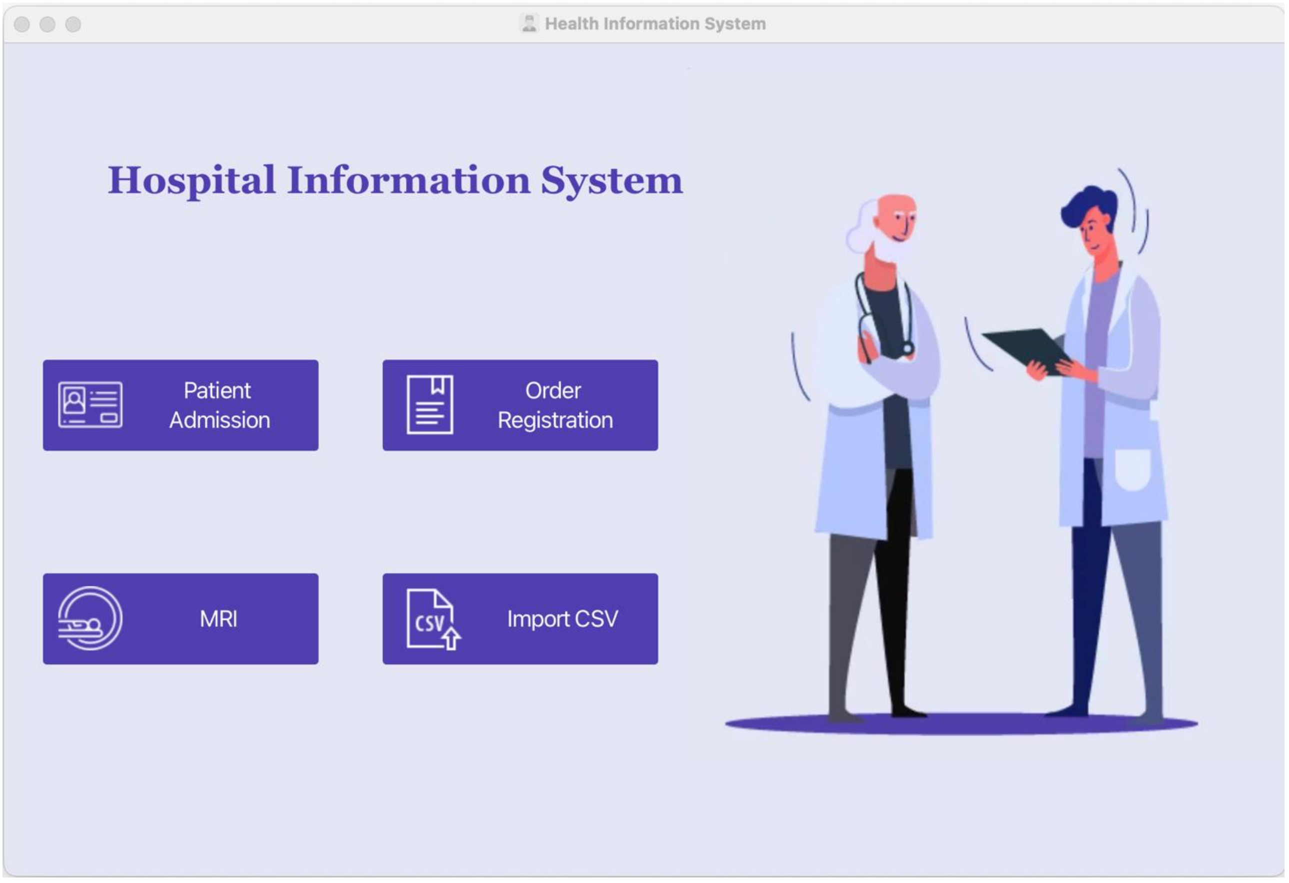


Figure 7: Application On Start

Depending on the process chosen by the personnel, different actions can be performed.

Patient Admission

The patient admission process allows to search for a patient already registered in the hospital by inserting their Patient Identification Code in the search bar and clicking "Search". If no ID is inserted, pressing the "Search" button will cause no action.

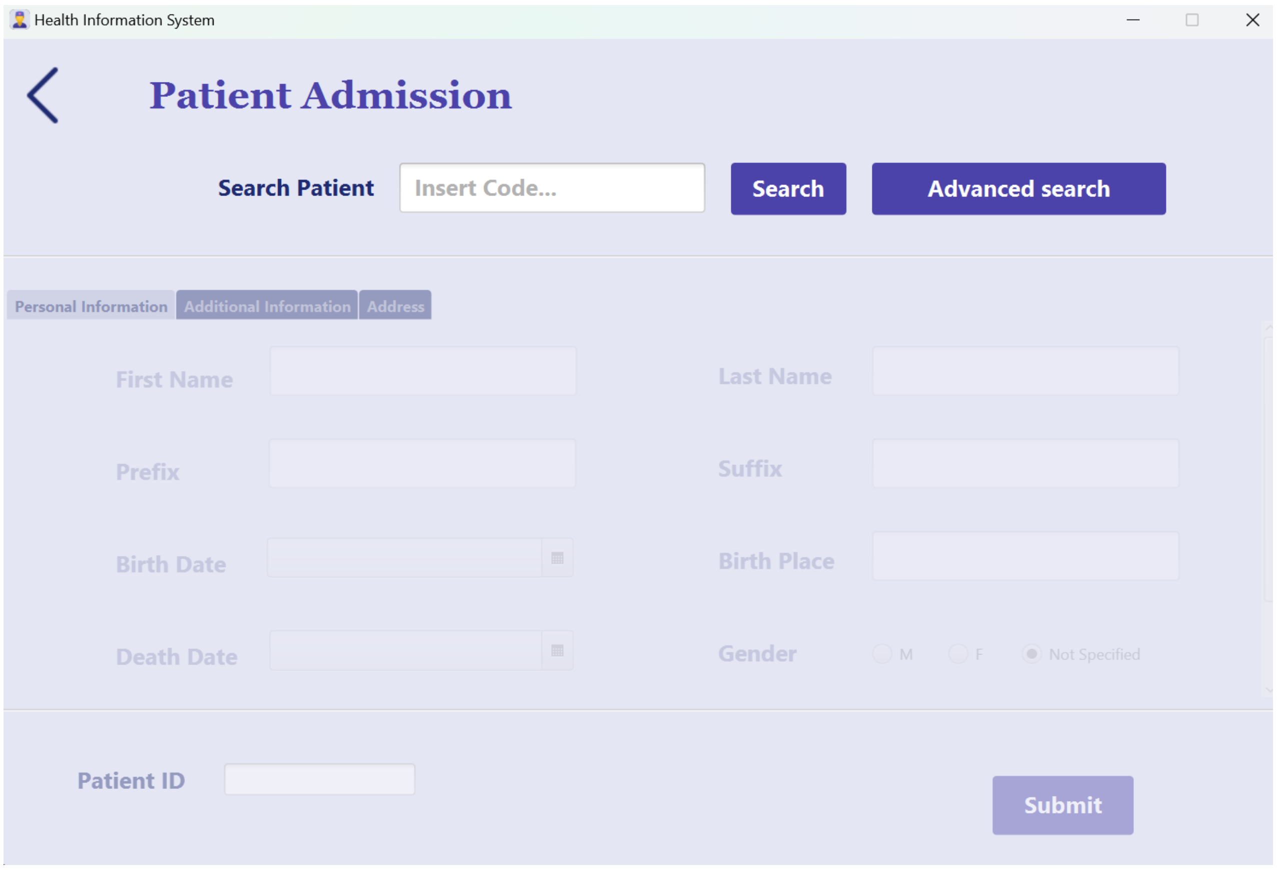
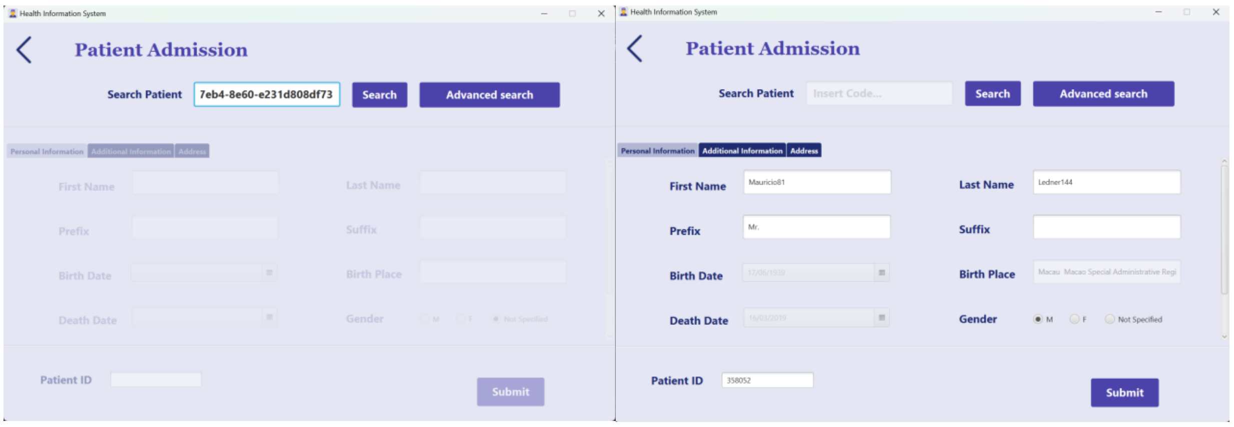


Figure 8: Patient Admission User Interface

If the patient is present on the FHIR Server, then the application unlocks the text fields and fills them with the information retrieved from the server.



Search Patient: 7eb4-8e60-e231d808df73

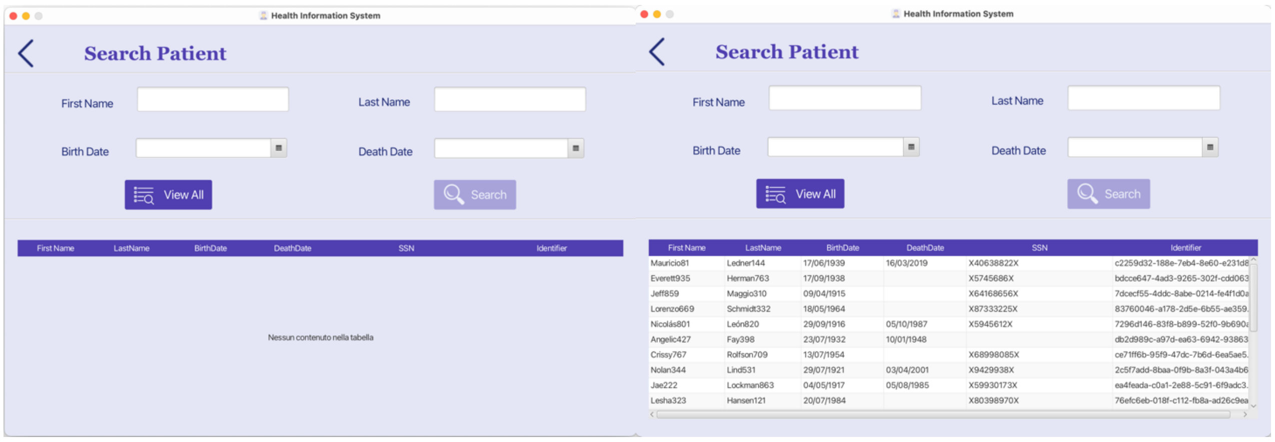
First Name: Mauricio81
Last Name: Ledner144
Prefix: Mr.
Suffix:
Birth Date: 17/06/1939
Birth Place: Macau Macao Special Administrative Regi
Death Date: 16/03/2019
Gender: [✓ M] F Not Specified
Patient ID: 358052

Figure 9: Search the patient by identifier on the Server

The text fields are divided into 3 subsections: Personal Information, Additional Information and Address.

Another possibility, if the patient's ID is unknown, is to search through the "Advanced Search" button in the up-right side of the application.

After clicking, the application shows the fields:



Left:

First Name	LastName	BirthDate	DeathDate	SSN	Identifier

Nessun contenuto nella tabella

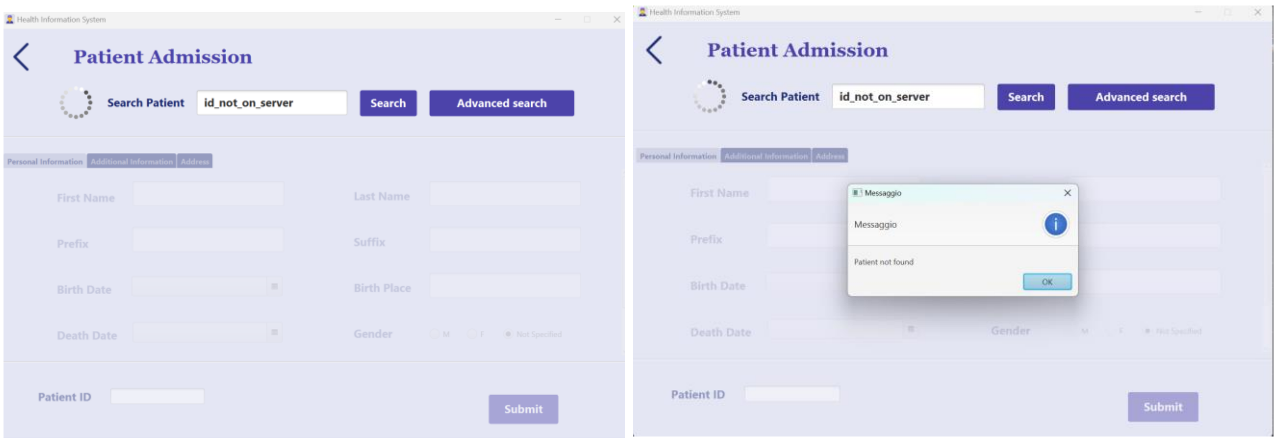
Right:

First Name	LastName	BirthDate	DeathDate	SSN	Identifier
Mauricio81	Ledner144	17/06/1939	16/03/2019	X40638822X	c2259d32-188e-7eb4-8e60-e231d8
Everett935	Herman763	17/09/1938		X5745686X	bdcce647-4ad3-9265-302f-cdd063
Jeff859	Maggio310	09/04/1915		X64168656X	7dcecf55-4ddc-8abe-0214-fe4f1d0a
Lorenzo669	Schmidt332	18/05/1964		X87333225X	83760046-a178-2d5e-6b55-ae359.
Nicolás801	León820	29/09/1916	05/10/1987	X5945612X	7296d146-83f8-b899-52f0-9b690a
Angelic427	Fay398	23/07/1932	10/01/1948		db2d989c-a97d-ea63-6942-93863
Crissy767	Rolfson709	13/07/1954		X68998085X	ce71ff6b-95f9-47dc-7b6d-6ea5ae5.
Nolan344	Lind531	29/07/1921	03/04/2001	X9429938X	2c5f7add-8baa-0f9b-8a3f-043a4b6
Jae222	Lockman863	04/05/1917	05/08/1985	X59930173X	ea4feada-c0a1-2e88-5c91-6f9adc3.
Lesha323	Hansen121	20/07/1984		X80398970X	76efc6eb-018f-c112-fb8a-ad26c9ea

Figure 10: Advanced Search - Patient Admission (left: when opened, right: after clicking view all)

The advanced search permits to search by common data that the patient can communicate. Since the search is not performed by ID, multiple entries for the same data can be found on the server. Thus, after clicking the "Search" button, the application displays a list of all the patients that satisfy the criteria, and shows their ID. The last possibility is to click the "View All" button, that displays a list of all the patients located on the server.

If, instead, the patient is not found on the server, the application allows the personnel performing the patient admission to insert a new patient on the server by filling, manually, the fields of the application in the Patient Admission page, and pressing the "Submit" button. The action associated with this button is the upload of a new patient on the server with the information contained in the fields. The user can choose an id for the patient.



Search Patient: id_not_on_server

Messaggio

Patient not found

Figure 11: Patient not found on server (left: search for the patient, center: message of patient not found

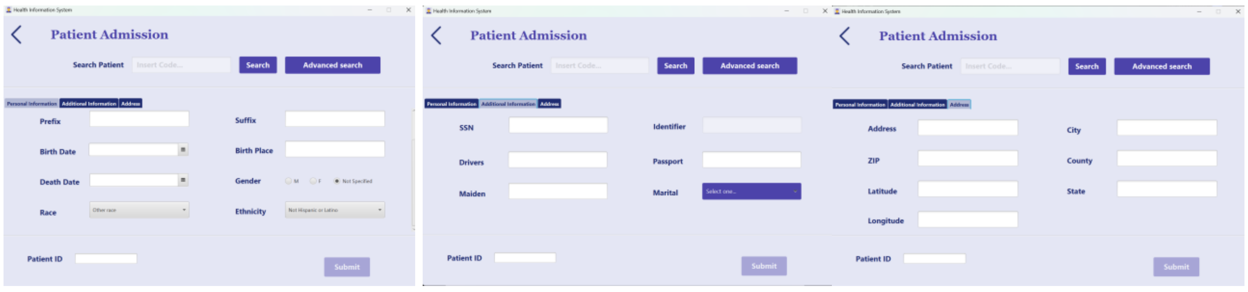


Figure 12: Fields are unlocked. left: Personal Information (cont.); center: Additional Information; right: Address

Order Registration

This page is built for the hospital personnel to insert a new order entry. It is characterized by a scroll bar so that the worker can insert all the information regarding the order.

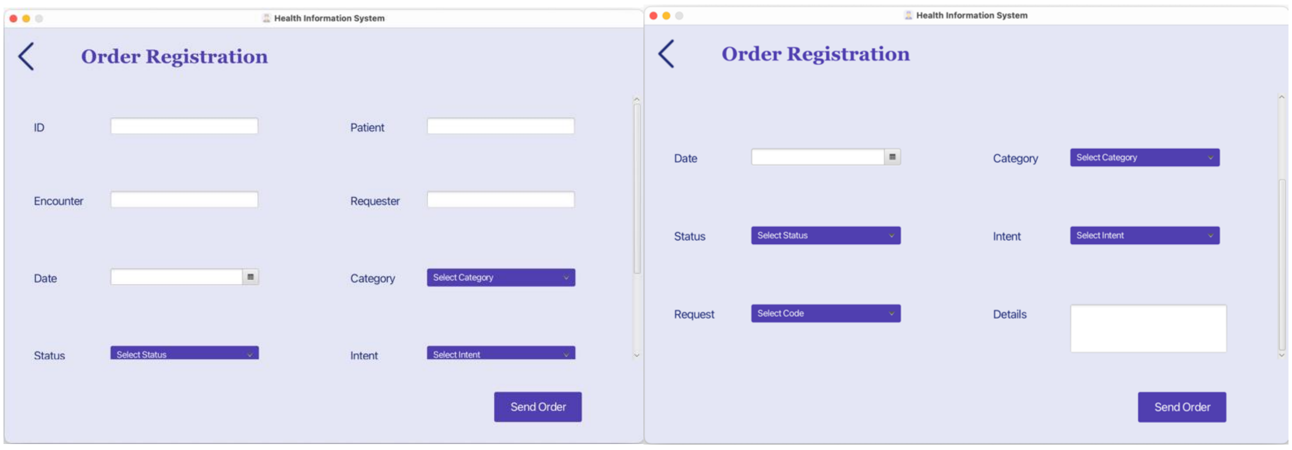
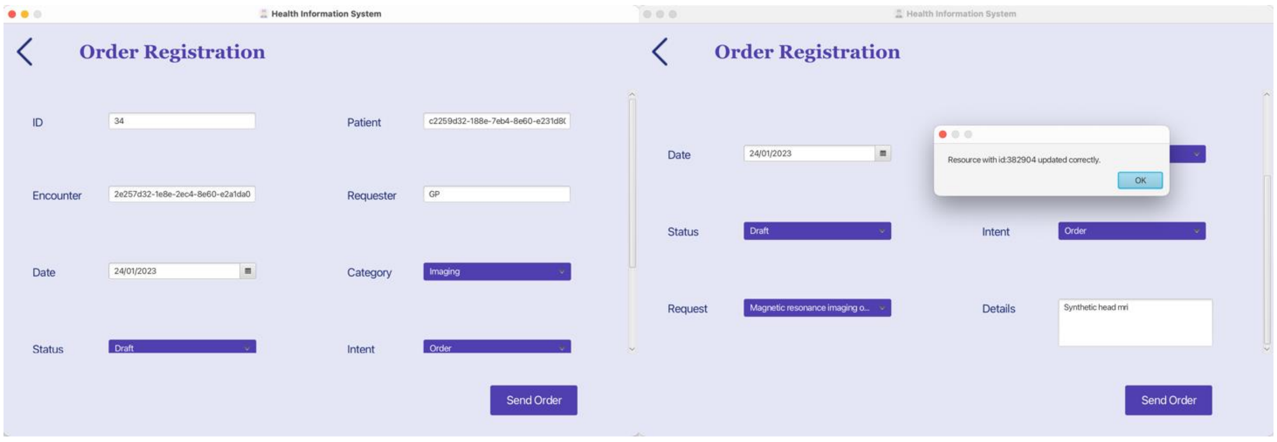


Figure 13: Order Registration User Interface. It can be scrolled down (right image).

The information put in this page is assumed to be correct, i.e. the patient, the requester (associated with the practitioner that requested the service), and the encounter are not searched on the server to check that their valid. For a complete application, this process should be done, but due to problems regarding the connection of the server it has been left to future upgrades. The "Category", "Status", "Intent", "Request" can be chosen among a set of valid possibilities through a Menu Button. Specifically, since the application only supports MRI analysis, the Requests acceptable are limited to the only possible MRI service requests.



ID: 34
Patient: c2259d32-188e-7eb4-8e60-e231d8(
Encounter: 2e257d32-1e8e-2ec4-8e60-e2a1da0
Requester: GP
Date: 24/01/2023
Category: Imaging
Status: Draft
Intent: Order

Date: 24/01/2023
Status: Draft
Intent: Order
Request: Magnetic resonance imaging o...
Details: Synthetic head mri

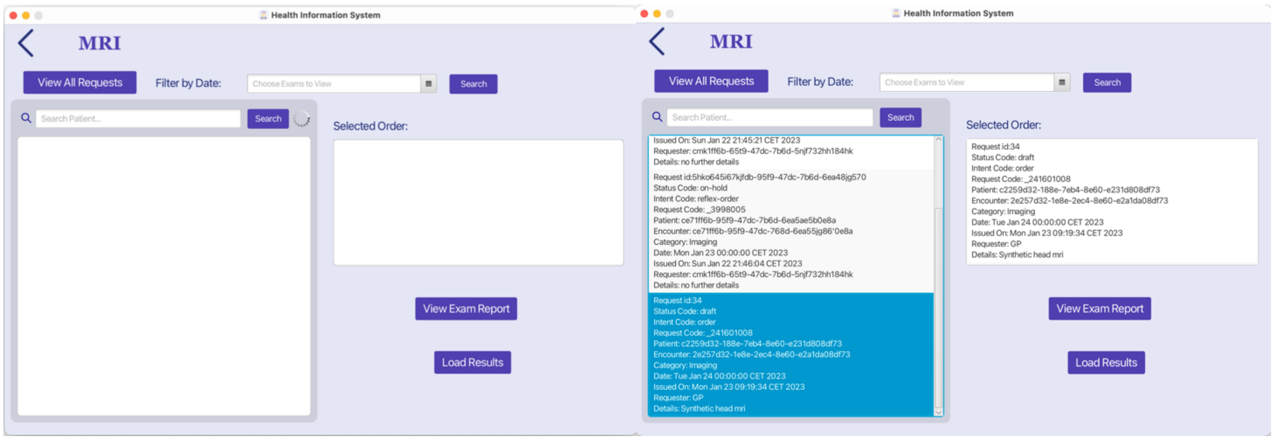
Resource with id:382904 updated correctly.

Figure 14: Send an order to server and successive popup highlighting correct update with server id

After filling *all* the fields, the order can be uploaded on the server.

MRI

This page is designed to perform different actions. First, the page displays the orders registered, currently present on the server. Then, the page gives different possibilities: either the requests can be filtered by date (displaying future requests), or they can be filtered by patient id. These actions are not mutually exclusive.



```
Issued On: Sun Jan 22 21:45:21 CET 2023
Requester: cmk1ff6b-65t9-47dc-7b6d-5njf732hh184hk
Details: no further details

Request id:5hko645i67kjfdb-95f9-47dc-7b6d-6ea48jg570
Status Code: on-hold
Intent Code: reflex-order
Request Code: _3998005
Patient: ce71ff6b-95f9-47dc-7b6d-6ea5ae5b0e8a
Encounter: ce71ff6b-95f9-47dc-768d-6ea55jg86'0e8a
Category: Imaging
Date: Mon Jan 23 00:00:00 CET 2023
Issued On: Sun Jan 22 21:46:04 CET 2023
Requester: cmk1ff6b-65t9-47dc-7b6d-5njf732hh184hk
Details: no further details

Request id:34
Status Code: draft
Intent Code: order
Request Code: _241601008
Patient: c2259d32-188e-7eb4-8e60-e231d808df73
Encounter: 2e257d32-1e8e-2ec4-8e60-e2a1da08df73
Category: Imaging
Date: Tue Jan 24 00:00:00 CET 2023
Issued On: Mon Jan 23 09:19:34 CET 2023
Requester: GP
Details: Synthetic head mri
```

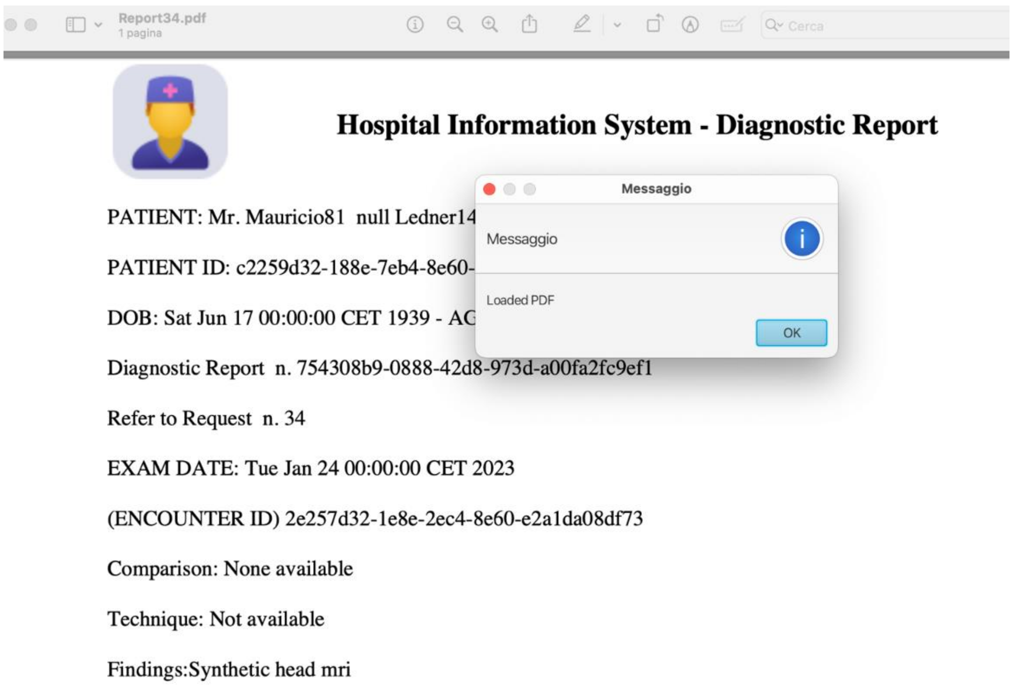
Selected Order:

```
Request id:34
Status Code: draft
Intent Code: order
Request Code: _241601008
Patient: c2259d32-188e-7eb4-8e60-e231d808df73
Encounter: 2e257d32-1e8e-2ec4-8e60-e2a1da08df73
Category: Imaging
Date: Tue Jan 24 00:00:00 CET 2023
Issued On: Mon Jan 23 09:19:34 CET 2023
Requester: GP
Details: Synthetic head mri
```

Figure 15: MRI User Interface (left empty, right with loaded orders)

If one of the filters is applied, the application searches on the server for the requests that satisfy the criteria,and shows them on the List View on the left. If the user wants, then, to visualize them all again, they can press the “View All Requests” button, to display them all.

It is possible to select the request of interest among those showed, and it will be displayed on the view on the right, under the “Selected Order” label. From there, if the examination has already been carried out, the Report of the exam can be seen as a PDF format. Instead, if the doctor has to upload the results of the analysis (considered as MRI), they can perform this operation by pressing the “Load Results” button.



Hospital Information System - Diagnostic Report

Messaggio

Messaggio

Loaded PDF

OK

PATIENT: Mr. Mauricio81 null Ledner14

PATIENT ID: c2259d32-188e-7eb4-8e60-

DOB: Sat Jun 17 00:00:00 CET 1939 - AG

Diagnostic Report n. 754308b9-0888-42d8-973d-a00fa2fc9ef1

Refer to Request n. 34

EXAM DATE: Tue Jan 24 00:00:00 CET 2023

(ENCOUNTER ID) 2e257d32-1e8e-2ec4-8e60-e2a1da08df73

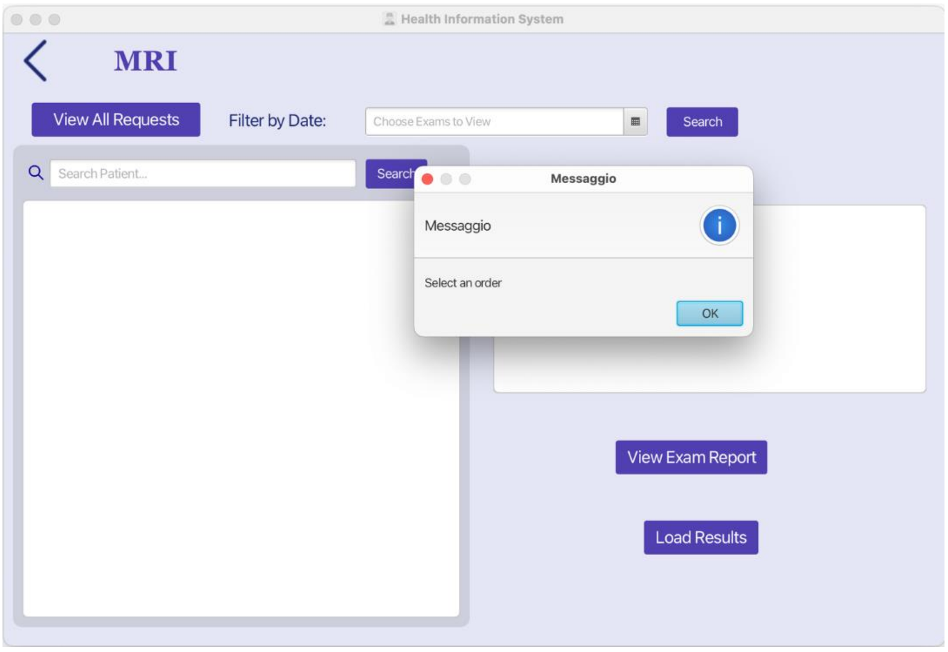
Comparison: None available

Technique: Not available

Findings:Synthetic head mri

Figure 16: Example of report with its popup signaling that the report has been opened

If no order has been selected, the application shows an information message that invites the user to select an order.



Messaggio

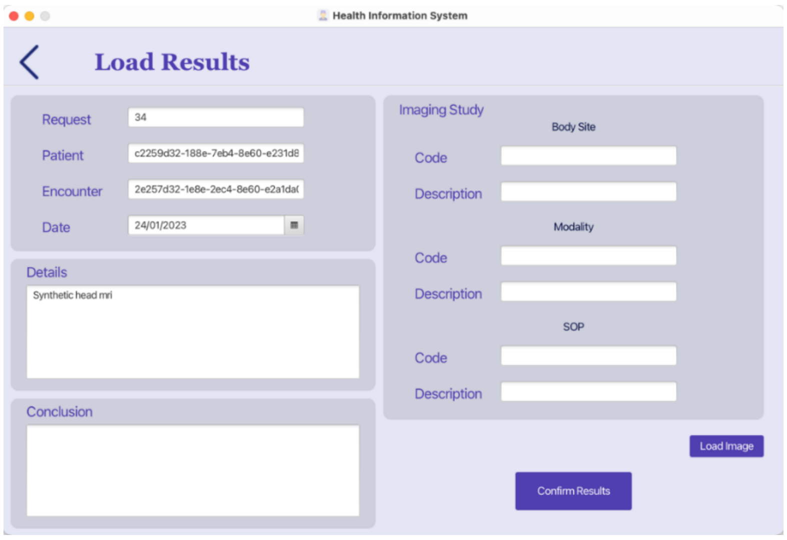
Messaggio

Select an order

OK

Figure 17: No order selected after pressing "Load Results"

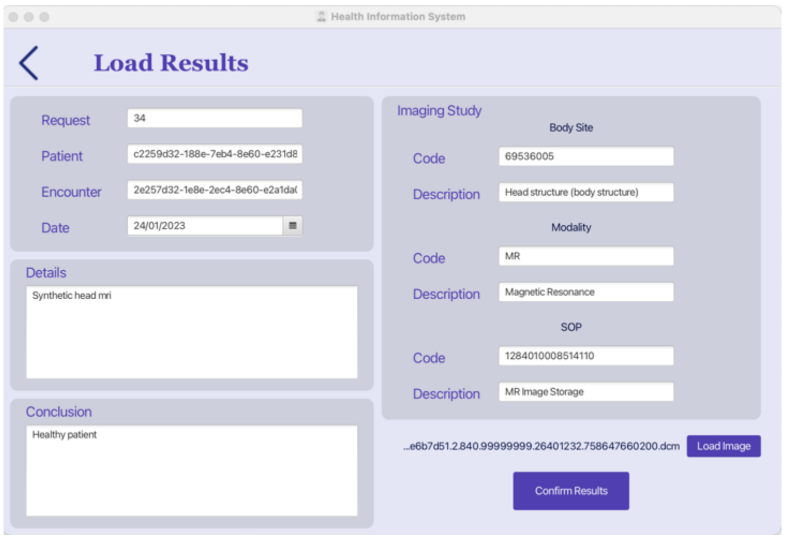
The output of the “LoadResults” button is shown below. In this page, some fields, such as “Request”, “Patient”, “Encounter”, “Date” and “Details”, are prefilled with information coming from the selected order.



Request	34
Patient	c2259d32-188e-7eb4-8e60-e231d8
Encounter	2e257d32-1e8e-2ec4-8e60-e2a1da(
Date	24/01/2023
Details	Synthetic head mri

Figure 18: LoadResult page right after opening

The doctor can insert a conclusion, and information about the study performed. Lastly, it is possible to select the DICOM image associated with the report, which will be visualized before performing the next step. Pressing the “Confirm Results” button will create a PDF file containing the information and will upload the report on the server.



Request	34
Patient	c2259d32-188e-7eb4-8e60-e231d8
Encounter	2e257d32-1e8e-2ec4-8e60-e2a1da(
Date	24/01/2023
Details	Synthetic head mri
Conclusion	Healthy patient

Body Site	Code	69536005
	Description	Head structure (body structure)
Modality	Code	MR
	Description	Magnetic Resonance
SOP	Code	1284010008514110
	Description	MR Image Storage

...e6b7d51.2.840.99999999.26401232.758647660200.dcm

Figure 19: Load Results after filling the fields

Import CSV

This page is a shortcut when first using the application. It allows the staff to upload the stored data of the hospital from csv formats. After clicking “Send”, the CSVs read are translated into FHIR Resources, and later uploaded on the server. There are some dependencies that need to be respected before sending the data to the server, such as the insert of the patient data before trying to upload encounters data, since they are highly dependent.

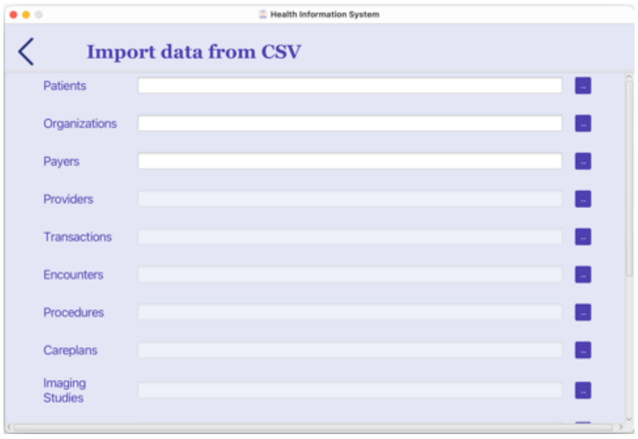


Figure 20: CSV User Interface

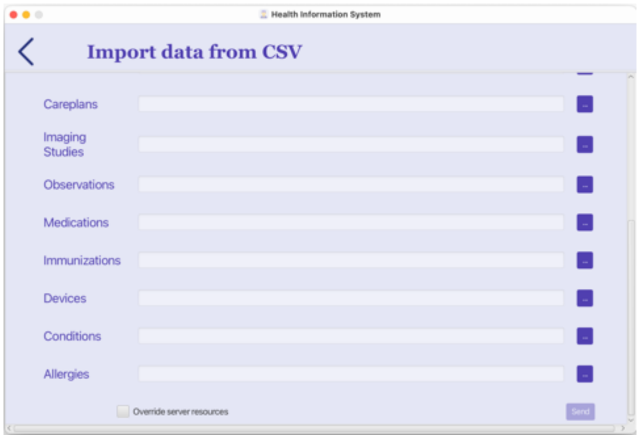


Figure 21: CSV User Interface – continued.

The fields highlight as it becomes possible to insert the CSV of the corresponding resource, a consequence of the insertion of the CSVs corresponding to the dependent resources.

Application

The application is organized in packages:

- **Java/CSV**:
  This package contains utility classes to manipulate the CSVs. The classes are:
  a. *LoadCSV*, to load a specific CSV in the memory of the application, to later upload the specific resource on the server;
  b. *ReadCSV*, the read the CSV and create the specific resource starting from the data contained in the file.
- **Java/enumerations**:
  This package contains all of the enumeration classes of the project, implemented by hand when they were not available as classes of HAPI FHIR packages.
  They are needed to encode the *CodableConcepts* of the *Resources*, in cases where, for example, a code associated to a procedure, a specialization of a practitioner, or the type of the encounter had to be explicit in the resource.
  The specific classes are:

a. *EncounterClass*, to encode the different possible classes of the encounter. The class is a concept representing classification of patient encounter such as ambulatory (outpatient), inpatient, emergency, home health or others due to local variations. Since the coherent dataset included classes not specified in the US Core Profile of the HL7 standard, somewhere added by hand as symbolic classes so that all of the encounters could be characterized by a class;
b. *OMBEthnicityCategories*, necessary to encode the Ethnicity of the Patients. The possible values are "Hispanic or Latino" and "Not Hispanic or Latino";
c. *OMBRaceCategories*, to encode the Race of the Patient, whose values are taken from the US Core Profile of the HL7 Standard;
d. *PIdentifier*, where the other possible identifiers for the Patient are encoded. The value set includes the passport number (PPN), the driver's license number (DL) and the social security number (SS);
e. *ProviderSpecialtySNOMED*, where the different possible SNOMED Practitioner official certifications, training, and licenses that authorize provision of care by the practitioner are encoded;
f. *ProviderSpecialtyNUCC*, where the different possible NUCC Practitioner official certifications, training, and licenses that authorize provision of care by the practitioner are encoded;
g. *ServiceRequestCategory*, used to define the category for the request, a code that classifies the service for searching, sorting and display purposes.
h. *ServiceRequestCode*, that represents the possible codes for the request. The codes describe the service requested for the patient.

- **Java/HIS_Project**:
  This package contains the main application class (App.java) and all the logic for the user interface.
  Specifically,
  a. *BasicController*, a base class that is extended by all the controllers. It contains the logic to initialize the controllers and to go back to the opening page of the application when needed;
  b. *OpeningPageController*, where the only aspect to manage is to select the right interface when the user selects a specific button;
  c. *PatientAdmissionController*, it manages the logic behind the "Patient Admission" user interface. In this Controller, the research for a patient in the server by their identifier can be carried out, or a new patient can be inserted by filling the fields. The only mandatory field to fill is the name for the patient. Then, a new Patient FHIR Resource is created and uploaded on the server;

d. *AdvancedSearchController*, needed to run the logic behind the "Advanced Search" user interface. It allows to search a patient by name, surname, date of birth or death date (or to search them all) on the server and show the results in a table using the support of the *PatientListElem* class;
e. *OrderRegistrationController*, the controller behind the "Order Registration" user interface. The module reads the input inserted by the patient and creates a new *ServiceRequest* Resource, and uploads it to the server. As constraints, the date of the examination cannot be prior to the current day, and all the fields need to be filled;
f. *MRIController,* the logic behind the "MRI" user interface. In the page, the controller searches the server for the ServiceRequest resources available and displays them in a ListView; it allows to filter them by the date by searching them on the server or to filter them by patient by only reading the information in the ObservableList in which the resources are stored, and displays only the ones that satisfy the criteria chosen. One of the resources can be selected and showed in a secondary ListView representing the Selected Order;
g. *LoadResultsController,* the controller behind the "Load Results" user interface. It implements the logic to visualize the necessary information about the request chosen (request id, patient id, encounter id, date of the service, details about the service) without having the user inserting them once again. Then, a *DiagnosticReport* Resource is created and uploaded on the server using the information provided by the user. The report contains references to the *ServiceRequest* and to the *ImagingStudy* Resource created when uploading the data about the analysis of the patient, which gets uploaded to the server as well. All of the fields need to be filled;
h. *ImportCSVController*, the controller behind the "Import CSVs" user interface. In this module, necessary checks about dependencies of the data is implemented, locking and unlocking the fields that can be filled depending on the information provided by the user. Later, it uploads all the inserted resources on the server using a map as a support for the created resources. This page is not intended to be used again.

The controllers provide user feedback if one of the requirements is not satisfied by displaying alert boxes. Moreover, the user has visual feedback of a progress bar image when pressing buttons in different interfaces, and a final feedback when the operations is concluded, either correctly or incorrectly.

- **Java/resources**:
  This package contains all of the classes used to create the FHIR Resources starting from data either present in the CSVs or coming from user input when using the interface. More details

about the classes of this package in the ***CSV Mapping*** section. All of the classes of this package extend a *BaseResource* class that manages the logic behind the assignment of the Identifier of the classes and defines an abstract function *createResource* which is implemented differently by each extending class depending on the FHIR Resource to create.
An important note has to be made about the Identifier chosen. Actually, in the BaseResource class it is assigned depending on whether the id can be found in the CSVs or not: if it is present, then the resource is identified by that value; otherwise, an identifier is created and assigned using the Universally Unique Identifier[16] Java Package.
Another relevant class is *BaseOrganization*, that is later extended by classes such as *OrganizationResource* and *CoverageResource*. It contains the common attributes that identify a generic organization, such as a Name and a Address, whereas the other attributes, specific to a definite type of organization, are implemented in the latter classes, and for each organization a type is specified.

- **Java/state**:
  This is a utility package for the classes that serve a purpose throughout the whole application. The classes are:
    a. *Memory*, a class necessary to keep the memory of the operations performed in the application. Here, the map saves all of the resources created but not yet uploaded on the server are stored. Thus, it is implemented as a Singleton;
    b. *Context*, also implemented using the Singleton pattern, defines the FHIR client that will connect to the server throughout the application;
    c. *PDF*, this class manages the creation of the report in PDF format, with the information provided by the user in the "Load Results" interface and the and the information of patient and the service request it refers to;
    d. *ServerInteraction*, that manages the logic behind the interaction with the server. It allows to initially populate the server with resources created from CSV files and then support the client retrieving from and uploading on the server the resources created by the user through the application.

[16] The UUID library generates random identifiers of 128 characters. Using the birthday theorem the probability to have two equals identifiers is:$\frac{(2^{128})!}{(2^{128}-n_{elem})!}$. There are less than 2 million elements in total in the demo. Supposing to have 10 million elements in total the probability of having two equals identifiers is: $1-\frac{(2^{128})!}{(2^{128}-10^{7})!} \approx 1-\frac{(10^{38})!}{(10^{38}-10^{7})!} \approx 1-1$

User Interface

As mentioned above, the user interface is implemented through different *.fxml* files that can be found in the package **main/resources,** one for each of the operations that can be performed using the application:

a. *OpeningPage.fxml;*
b. *PatientAdmission.fxml;*
c. *AdvancedSearch.fxml;*
d. *OrderRegistration.fxml;*
e. *MRI.fxml;*
f. *LoadResults.fxml;*
g. *ImportCSVs.fxml.*

The application has an icon[17] that shows on most common latest version of OS. Tests have been performed on MacOS X Ventura 13.0.1, Windows 11, Ubuntu 20.0.4.

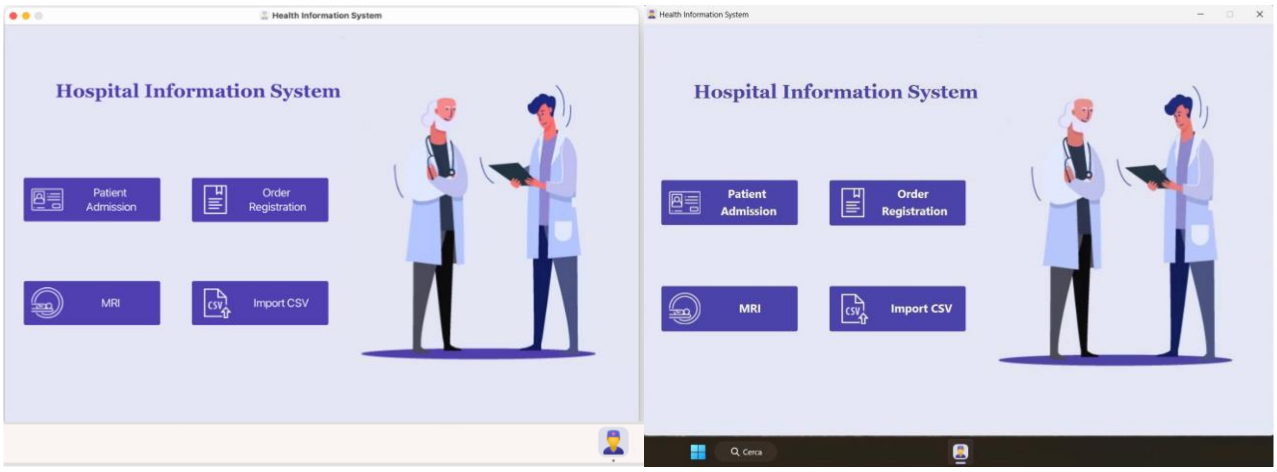

Health Information System
Hospital Information System
Patient Admission
Order Registration
MRI
Import CSV
Health Information System
Hospital Information System
Patient Admission
Order Registration
MRI
Import CSV
Cerca

Figure 22: Icon showing on MacOS X Ventura and Windows 11

CSV Mapping

Reading a CSV

The CSVs are read using the *opencsv* library, and, in particular, the *CsvToBeanBuilder* object.
It reads the CSV by assigning the value of the column read to the corresponding field of the resource to map thanks to the identification between names. In fact, the fields of the resources are named after the header of the CSV to map, with the addition of a *@CsvBindByName* tag.

[17] icon8 medical doctor: https://icons8.com/icon/OESdlMYQIKWs/medico

Once the CSV is read, a mapping between the id of the resource and the resource itself is inserted inside a utility map.
Then, after each resource is put inside of the utility map, a further mapping is created, with key equal to the Class of the Resource stored in the location, and as a value the utility map for that resource, where all the entries read from the CSV are stored.
Each resource is mapped to a FHIR Resource using the classes described below, where every class is characterized by the attributes read in the CSV, constructors as needed, getters and setters, the toString function, and the ***createResource*** function.
Every resource is created accounting for all the *must – have* information that it should have according to the HL7 standard, therefore when this information was missing in the CSV, a replacement was chosen taking into consideration the values of other fields.

Patient

The Patient is mapped through the *PatientResource* class.

The patient is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Patient;
- Definition of a profile using the US Core Profile for the Patient;
  The Mandatory values according to the profile are the gender, an identifier and the patient name.
- BIRTHDATE, DEATHDATE to BirthDate, Deceased;
- BIRTHPLACE to Extension(Address - birthplace);
- ID, SS, PPN, DL to Identifier, Identifier, Identifier, Identifier, with system v2-0203;
- FIRST, LAST, MAIDEN, PREFIX, SUFFIX to Name, with different uses OFFICIAL, MAIDEN;
- CITY, ADDRESS, COUNTY, ZIP, STATE to Extension(Address);
- LAT, LON to Extension(Geolocation);
- MARITAL to MaritalStatus, from the V3MaritalStatus enumeration;
- RACE, ETHNICITY to Extension(US Core Race) and Extension (US Core Ethnicity);
- GENDER to Gender from the V3AdministrativeGender enumeration.
- There are two CSV Fields that were not possible to map in the resource. They are HEALTHCARE_EXPENSES and HEALTHCARE_COVERAGE.

Encounter

Encounters are mapped through the *EncounterResource* class.

The encounter is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Encounter;
- Definition of a profile using the US Core Profile for the Encounter;
  The Mandatory values according to the profile are a status, a classification such as inpatient, outpatient or emergency, an encounter type, a patient.
- ID to Identifier;
- START, STOP to Period(Start, Stop);
- The status for the encounter, since it is a must have attribute and it is not available on the CSV, is derived from information about the dates of start and end of the encounter. This information is mapped to Status;
- PATIENT to Subject(Reference(Patient));
- ORGANIZATION to ServiceProvider(Reference(Organization));
- PROVIDER to Participant(Reference(Practitioner));
- PAYER_COVERAGE to Account(Reference(Account – where the coverage is added);
- ENCOUNTERCLASS to Class, through the EncounterClass enumeration;
- CODE, DESCRIPTION to Type, with system SNOMED;
- REASONCODE, REASONDESCRIPTION to ReasonCode, with system SNOMED.
- There are two CSV Fields that were not possible to map in the resource. They are BASE_ENCOUNTER_COST and TOTAL_CLAIM_COST.

Providers

Providers correspond to Practitioners, and they are mapped using the *ProviderResource* class.

The practitioner is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Practitioner;
- Definition of a profile using the US Core Profile for the Practitioner;
  The Mandatory values according to the profile are an identifier and the name.
- ID to Identifier;
- NAME to Family and Given;
- GENDER to Gender through the V3AdministrativeGender enumeration;
- CITY, ADDRESS, ZIP, STATE, to Extension(Address);
- LAT, LON to Extension(Geolocation);
- Organization to Issuer(Reference(Organization));
- SPECIALTY to Qualification through PractitionerQualificationComponent. This is done both for the SNOMED and NUCC encoding of the specialty.

- There is a CSV Field that was not possible to map in the resource. It is UTILIZATION.

Organizations

Organizations are mapped using the *OrganizationResource* class. This class extends a *BaseOrganization* class, that encapsulates all the attributes that are common between different types of organizations. The organizations described below are generic organizations.

The organization is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Organization;
- Definition of a profile using the US Core Profile for the Organization;
  The Mandatory values according to the profile are the name and the status (whether is still active or not).
- ID to Identifier;
- The status of the organization, since it is a must have attribute and it is not available on the CSV, is set to ACTIVE and mapped to Status;
- NAME to Name;
- A type is chosen for the organization; in this case, it set to PROV, an organization that provides healthcare services, and mapped to Type, through the OrganizationType enumeration;
- CITY, ADDRESS, POSTALCODE, ZIP, STATE to Address;
- LAT, LON to Extension(Geolocation);
- PHONE to Telecom with system Contact Point.
- There are two CSV Fields that were not possible to map in the resource. They are REVENUE and UTILIZATION.

Payers

Payers correspond to Organizations, and they are mapped using the *PayerResource* class.

The payer is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Organization;
- Definition of a profile using the US Core Profile for the Organization;
  The Mandatory values according to the profile are the name and the status (whether is still active or not).
- ID to Identifier;
- The status of the payer, since it is a must have attribute and it is not available on the CSV, is set to ACTIVE and mapped to Status through the OrganizationType enumeration;
- NAME to Name;

- A type is chosen for the organization; in this case, it set to PAY, a company, charity, or governmental organization, which processes claims and/or issues payments to providers on behalf of patients or groups of patients. It is mapped to Type;
- CITY, ADDRESS, POSTALCODE, ZIP, STATE_HEADQUARTERED to Address;
- PHONE to Telecom through the Contact Point enumeration.
- There are several CSV Fields that were not possible to map in the resource. They are AMOUNT_COVERED, AMOUNT_UNCOVERED, REVENUE, COVERED_ENCOUNTERS, UNCOVERED_ENCOUNTERS, COVERED_MEDICATIONS, UNCOVERED_MEDICATIONS, COVERED_PROCEDURES, UNCOVERED_PROCEDURES, COVERED_IMMUNIZATIONS, UNCOVERED_IMMUNIZATIONS, UNIQUE_CUSTOMERS, QOLS_AVG, MEMBER_MONTHS.

Payer Transitions

Payer Transitions correspond to Coverages, since they a financial instrument which may be used to reimburse or pay for health care products and services including both insurance and self-payment. They are mapped using the *CoverageResource* class.

The coverage is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Coverage;
- No profile available by US Core;
- Definition of a new Identifier (since it is not available on the CSV) mapped to Identifier;
- PATIENT to Reference(Subscriber);
- START_YEAR, END_YEAR to Period(Start, Stop);
- PAYER to Reference(PolicyHolder);
- OWNERSHIP to Relationship through the SubcriberRelationship enumeration.
- All the CSV Fields have been mapped in the resource.

Observations

Observations are mapped using the *ObservationResource* class.

The Observation is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Observation;
- There are different profiles for the Observation resource defined by US Core which differ from each other for the category. Since the CSV contains observations of different categories which is not possible to discriminate, it is not defined a specific profile but only the common mandatory values have been inserted. They are a LOINC code, a patient and a status.
- Definition of a new Identifier (since it is not available on the CSV) mapped to Identifier;
- DATE to Issued;

- The status of the observation, since it is a must have attribute and it is not available on the CSV, is set depending on the value of the field DATE and mapped to Status;
- PATIENT to Reference(Subject);
- ENCOUNTER to Reference(Encounter);
- CODE, DESCRIPTION to Code, with system LOINC;
- TYPE, VALUE, UNITS to Value.
- All the CSV Fields have been mapped in the resource.

Imaging Studies

Imaging studies are mapped using the *ImagingStudyResource* class.

The Imaging Study is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is ImagingStudy;
- No profile available by US Core;
- ID to Identifier;
- DATE to Started;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- BODYSITE_CODE, BODYSITE_DESCRIPTION to BodySite, with system SNOMED;
- MODALITY_CODE, MODALITY_CODE _DESCRIPTION to Modality, with system DICOM NEMA, through the ImagingStudySeriesComponent;
- SOP_CODE, SOP_DESCRIPTION to SopClass, with system DICOM NEMA, through the ImagingStudySeriesInstanceComponent.
- All the CSV Fields have been mapped in the resource.

Immunizations

Immunizations are mapped through the *ImmunizationResource* class.

The Immunization is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Immunization;
- Definition of a profile using the US Core Profile for the Immunization;
  The Mandatory values according to the profile are a patient, a vaccine code that identifies the kind of vaccine administered, a date the vaccine was administered and a status.
- Definition of a new Identifier, since it is not available on the CSV, mapped to Identifier;
- DATE to Recorded;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to VaccineCode;
- Since there is no direct mapping for the CSV Field BASE_COST in the FHIR Resource Immunization, this field has been defined as the text of a Note through Annotation;

- The status for the immunization, since it is a must have attribute and it is not available on the CSV, is derived from information about the date the immunization was administered. This information is mapped to Status, through the ImmunizationStatus enumeration.
- All the CSV Fields have been mapped in the resource.

Medications

Medications are mapped using the *MedicationResource* class.

The Medication is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is MedicationStatement;
- No profile available by US Core;
- Definition of a new Identifier (since it is not available on the CSV) mapped to Identifier;
- START, STOP to Effective(Period);
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Context(Reference(Encounter));
- CODE, DESCRIPTION to Medication, with system rxnorm;
- BASE_COST through MedicationKnowledge to Cost;
- REASONCODE, REASONDESCRIPTION to ReasonCode, with system SNOMED;
- PAYER to PolicyHolder through Coverage;
- PAYER_COVERAGE to CostToBeneficiary through Account;
- DISPENSES to Note through Annotation.
- There is a CSV Field that was not be possible to map in the resource. It is TOTALCOST.

Procedures

Procedures are mapped through the *ProcedureResource* class.

The procedure is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Procedure;
- Definition of a profile using the US Core Profile for the Procedure;
  The Mandatory elements according to the profile are a patient, a status, a code that identifies the type of procedure performed on the patient and when the procedure was performed.
- Definition of a new Identifier, since it is not available on the CSV, mapped to Identifier;
- DATE to Performed;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to Code, with system SNOMED;
- Since there is no direct mapping for the CSV Field BASE_COST in the FHIR Resource Procedure, this field has been defined as the text of a Note through Annotation;
- REASONCODE, REASONDESCRIPTION to ReasonCode, with system SNOMED;

- The status for the procedure, since it is a must have attribute and it is not available on the CSV, is derived from information about the date the procedure was performed. This information is mapped to Status, through the ProcedureStatus enumeration.
- All the CSV Fields have been mapped in the resource.

CarePlans

CarePlans are mapped through the *CarePlanResource* class.

The careplan is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is CarePlan;
- Definition of a profile using the US Core Profile for the CarePlan;
  The Mandatory elements according to the profile are a patient, a status, an intent, a category code and a narrative summary of the patient assessment and plan of treatment.
- ID to Identifier;
- START, STOP to Period(Start, Stop);
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to Category, with system SNOMED;
- REASONCODE, REASONDESCRIPTION to Address, with system SNOMED;
- The status for the careplan, since it is a must have attribute and it is not available on the CSV, is derived from information about the period the plan is effective. If the period is ended, the status is set to COMPLETED, otherwise to ACTIVE. This information is mapped to Status through the CarePlanStatus enumeration;
- The intent for the careplan, since it is a must have attribute and it is not available on the CSV, is set to PLAN. This information is mapped to Intent through the CarePlanIntent enumeration;
- The narrative summary of the patient assessment and plan of treatment, since it is a must have attribute and it is not available on the CSV, is set thanks to the fields DESCRIPTION and REASONDESCRIPTION. This information is mapped to Description.
- All the CSV Fields have been mapped in the resource.

Allergies

Allergies are mapped through the *AllergyIntoleranceResource* class.

The allergy is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is AllergyIntolerance;
- Definition of a profile using the US Core Profile for the Allergy Intolerance;
  The Mandatory elements according to the profile are a clinical status of the allergy (e.g., active or resolved), a code which tells you what the patient is allergic to and a patient.
- Definition of a new Identifier, since it is not available on the CSV, mapped to Identifier;
- START to OnSet;

- STOP to LastOccurrence;
- The status for the allergy, since it is a must have attribute and it is not available on the CSV, is derived from information about the period the allergy is present. This information is mapped to ClinicalStatus;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to Code, with system SNOMED.
- All the CSV Fields have been mapped in the resource.

Devices

Devices are mapped using the *DeviceResource* class.

The Device is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is DeviceUseStatement;
- No profile available by US Core;
- Definition of a new Identifier (since it is not available on the CSV) mapped to Identifier;
- START, STOP to Occurrence(Period);
- The category for the condition, since it is a must have attribute and it is not available on the CSV, is set to ENCOUNTERDIAGNOSIS. This information is mapped to Category through the ConditionCategory enumeration;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to Code, with system SNOMED.
- UDI to CarrierHRF through the DeviceUdiCarrierComponent
- All the CSV Fields have been mapped in the resource.

Conditions

Conditions are mapped using the *ConditionResource* class.

The Condition is created with the following mapping CSV Field to FHIR Resource Field:

- The FHIR Resource is Condition;
- Definition of a profile using the US Core Profile for the Condition;
  The Mandatory elements according to the profile are a category code of “problem-list-item” or “health-concern”, a code that identifies the condition and a patient;
- Definition of a new Identifier (since it is not available on the CSV) mapped to Identifier;
- START, STOP to OnSet(Period);

- The category for the condition, since it is a must have attribute and it is not available on the CSV, is set to ENCOUNTERDIAGNOSIS. This information is mapped to Category through the ConditionCategory enumeration;
- PATIENT to Subject(Reference(Patient));
- ENCOUNTER to Encounter(Reference(Encounter));
- CODE, DESCRIPTION to Code, with system SNOMED.
- All the CSV Fields have been mapped in the resource.

Fhir Server Interaction

All the possible interactions with the server are mapped in *ServerInteraction* utility class. The methods of the class are described below, but are used encapsulated into a service when called in the application.

Upload a Resource on the FHIR Server

When uploading a resource, the server does not perform any check on the already present resources. This means that if two resources have the same fields, even if they have the same id[18], the server duplicate the resource assigning a new id to the fresh resource. Considering as wrong the duplication of a resource, the only way to know if a resource is already present is to search on the server ("get" the resource). The search is performed on the identifier associated to the resource, as it is unique and uniquely associated with each instance of a resource.

Even if two resources share all the fields apart from the identifier, they are considered different from each other. For example, there could be two (or more) patients that have the same *HumanName*[19] and it could be the only information about these two patients we know about one of them. They can represent the same person, but they could also represent two different instances of *Patient* with poor information. The only field that cannot be common is the unique identifier. Therefore, if they differ only by the UUID they are different.

This behavior leads to two ways of performing upload: override the already present resources by performing an update or do nothing (assuming the version on the server as the latest with respect to the one that might be uploaded). In the arranged function to do so, the decision is leaved to the programmer. And this choice reflects on the GUI where the user themselves can choice if update the resources already present or do nothing when uploading csv resources.

Updating a Resource on the FHIR Server

The logic of updating a resource on the server was already mapped in upload function, so it is sufficient to pass the new resource and the will to override the resource if already present on the

[18] Id assigned by the server and used as unique key by itself.
[19] Composition of Given Name, Surname, Prefix and Suffix

server. The function retrieves the old resource using the identifier (that must be the same as previously described) as search parameter. Having the old resource, we can know the id assigned by the server to that and assigning it to the new resource. In this way, the instruction given to the server is to update a resource with that id. Using the upload function to perform also update has a desired benefit: if a resource didn't exist it is created. In the application, the only part in which the user can perform an update of the resource, excluding the upload of csv data, is when admitting a patient. In such situation, even the GUI to update and to create a new patient is the same.

Getting a Resource from the FHIR Server

To retrieve a resource from the server some information must be specified when building the query. Parameters like resource type (that is FHIR class associated to the resource).

Generic Resource retrieving

The generic *Resource* get is done using the identifier as single parameter. This function is used when uploading or updating a resource and works independently from the resource type. The server answers with a bundle of resource. The application was built to allow only one resource with a given identifier. So, the function returns the first element in the bundle, because it can be composed of only one element. Although appears impossible, if the bundle is composed of more than one element, it is assumed that the first element is what we were looking for.

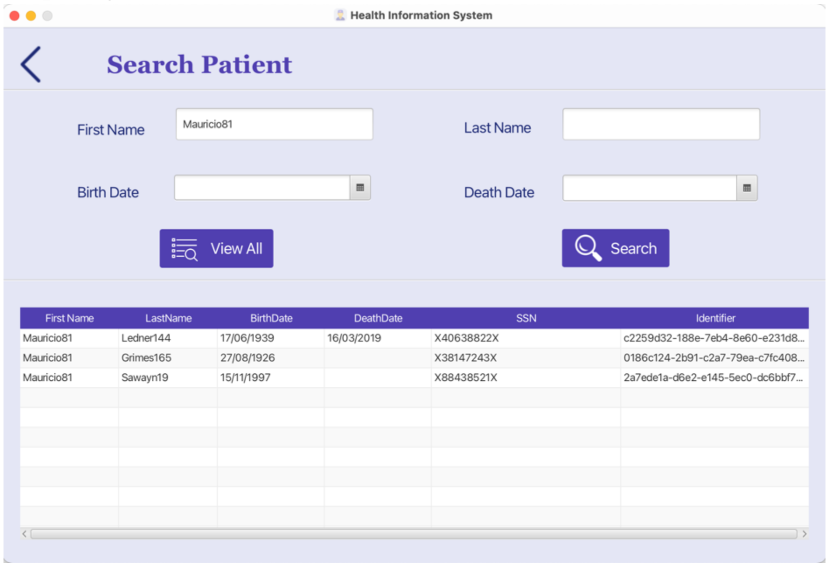
More than one resource retrieving

To get more than one resource of the same type[20] a specific function was built. This function returns a list of all the resources of that type on the server. This is done not specifying any parameter in the query besides the resource type.

[20] Fhir Resource class

More than one resource: Patient advanced search

In the application, there is a GUI that allows the user to give some parameters to narrow the number of patients down. The server query is built incrementally, adding the parameters that the user had insert. The user can specify name, surname, date of birth and date of death.



First Name	LastName	BirthDate	DeathDate	SSN	Identifier
Mauricio81	Ledner144	17/06/1939	16/03/2019	X40638822X	c2259d32-188e-7eb4-8e60-e231d8...
Mauricio81	Grimes165	27/08/1926		X38147243X	0186c124-2b91-c2a7-79ea-c7fc408...
Mauricio81	Sawayn19	15/11/1997		X88438521X	2a7ede1a-d6e2-e145-5ec0-dc6bbf7...

Figure 23: Retrieve only some patients. (This example: based on name)

More than one resource: ServiceRequest filter

In the application, there is a GUI that allows the user to filter out *ServiceRequest*s. This is done specifying the single parameter given by the user (the date) in the query, to retrieve only interesting resources.

DICOM attachments

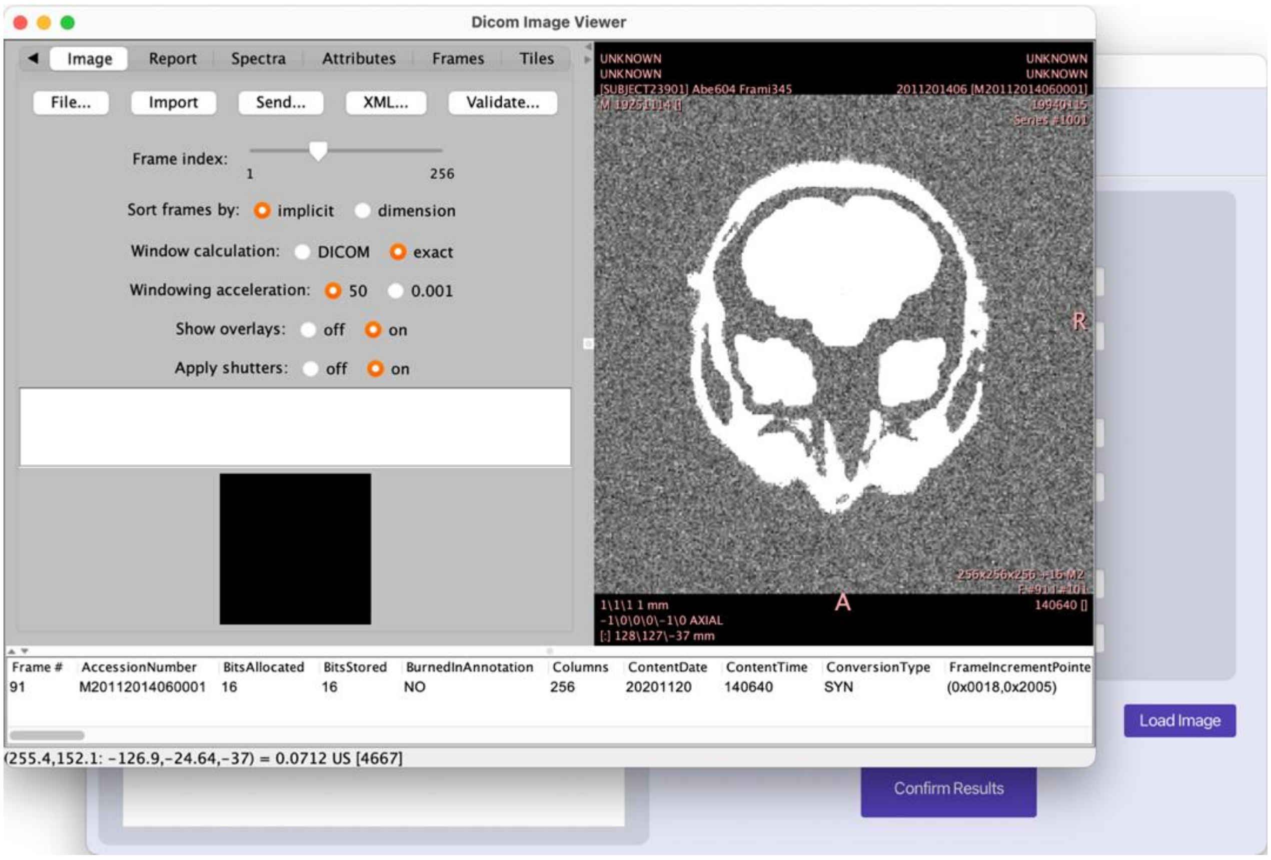
DICOM files' structure in dataset

Each dicom file in Coherent dataset[21] is a Synthetic Magnetic Resonance Imaging (MRI) brain scan, composed of 256 frames. A dicom file contains some information related to the patient, the study, the device, neglecting pixel data. These data are not used in the application to build the resource *DiagnosticReport* because they are already contained in references inside the resource itself.

[21] Walonoski J, Hall D, Bates KM, Farris MH, Dagher J, Downs ME, Sivek RT, Wellner B, Gregorowicz A, Hadley M, Campion FX, Levine L, Wacome K, Emmer G, Kemmer A, Malik M, Hughes J, Granger E, Russell S. The "Coherent Data Set": Combining Patient Data and Imaging in a Comprehensive, Synthetic Health Record. Electronics. 2022; 11(8):1199. https://doi.org/10.3390/electronics11081199

Visualization

To visualize the DICOM file an entire side application is needed because of their multi-frame nature. This is done thanks to *pixelmed* library. It can extract frames from the DICOM file and perform modifications on the images to better visualize them.



Frame index: 1 256

Sort frames by: [✓ implicit] dimension

Window calculation: DICOM [✓ exact]

Windowing acceleration: [✓ 50] 0.001

Show overlays: off [✓ on]

Apply shutters: off [✓ on]

UNKNOWN
UNKNOWN
[SUBJECT23901] Abe604 Frami345
M 19251114 []

UNKNOWN
UNKNOWN
2011201406 [M20112014060001]
19940115
Series #1001

256x256x256 +16 M2
F#91 I#101
140640 []

1\1\1 1 mm
-1\0\0\0\-1\0 AXIAL
[:] 128\127\-37 mm

Frame #	AccessionNumber	BitsAllocated	BitsStored	BurnedInAnnotation	Columns	ContentDate	ContentTime	ConversionType	FrameIncrementPointe
91	M20112014060001	16	16	NO	256	20201120	140640	SYN	(0x0018,0x2005)

(255.4,152.1: -126.9,-24.64,-37) = 0.0712 US [4667]

Figure 24: Image showing after chosen a file dicom

Fhir mapping

In FHIR each image is seen as an *Attachment* to a *Media* resource linked to a *DiagnosticReport*. The attachment is a byte array in base64. Therefore, each media is composed of raw information of pixels and number of frames. These two basic records allow to reconstruct the image and jointly with other data in *DiagnosticReport* even reconstruct the original DICOM. A file that cannot be uploaded in any other way in a FHIR server.