Health Informatics Project

Barichella Riccardo (77432301) - Daniele Francesca Mafalda (77212301)
Gugliotta Carmelo (77132301) - Leto Marco (77342301)

LINFO2381 — Academic year 2023-2024 — Report of Group n° 9

1 Introduction

Our web application has been projected with the aim of completely and efficiently managing the patient's medical data and clinical examinations. In particular, we have chosen to focus on the cardiology department, providing it specific tests such as those relating to ”Urine and Electrolytes” and ”Blood Pressure”, some basic tests that are useful for the cardiology department to have a basic overview of the cardiovascular health of patients and beyond.

Our goal was to develop an intuitive and functional interface that simplifies and optimizes the process of recording and accessing to medical data, allowing healthcare professionals to focus more on patients care. This is particularly relevant in the healthcare context, where worker's time and resources are precious and must be used as efficiently as possible.

The project was created following a client-server architecture, in which the backend takes care of managing the application logic, while the frontend provides an intuitive and responsive user interface.

2 Technologies used

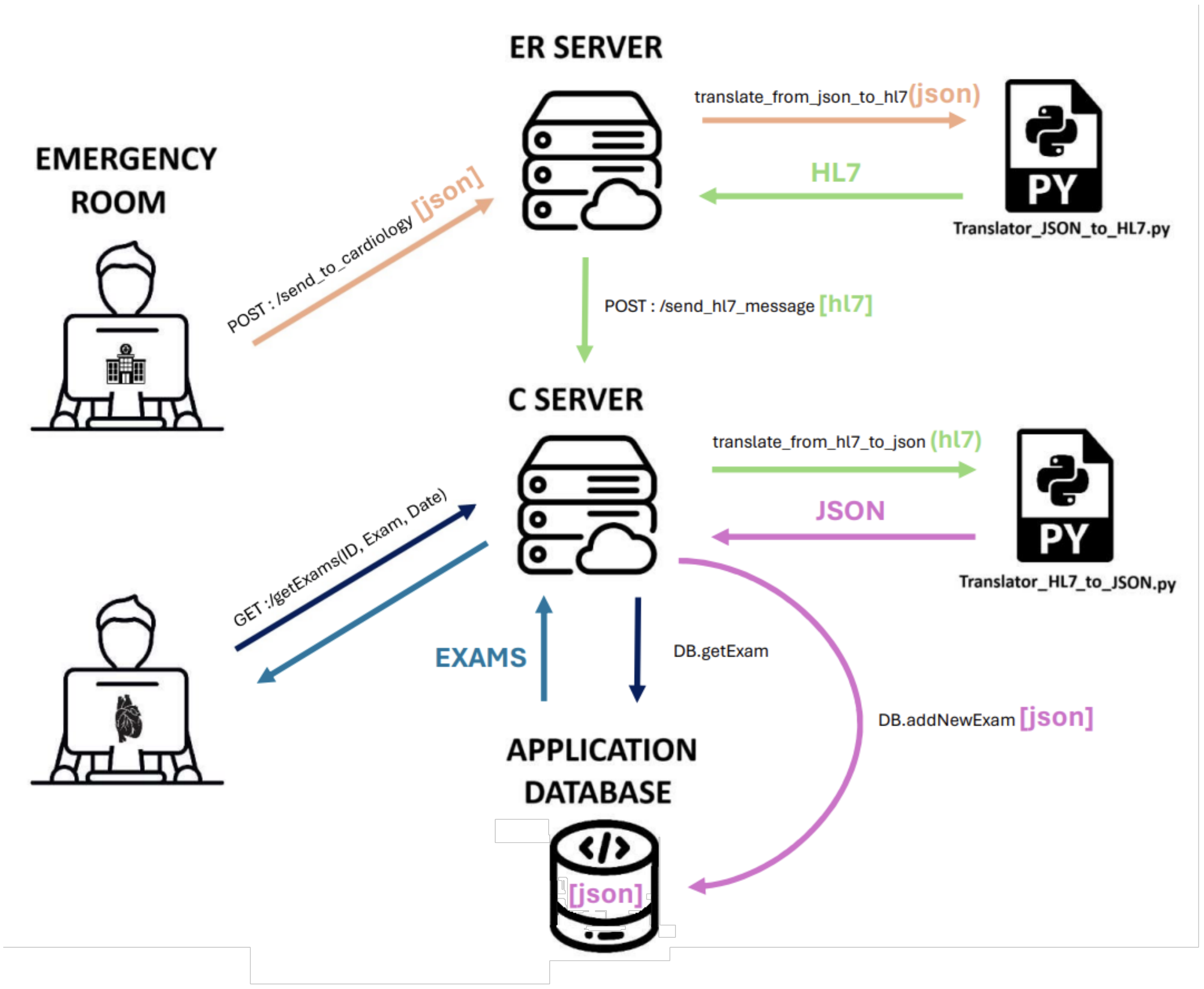
We utilise different tecnologies to develop our project:

- **Python** : Used to implement the backend logic, has been chosen for his versatility, ease of development, and the big amount of library that the programmer could use.
- **Flask**: Choosed as framework to create REST API (BSD-3 Clause Source License)
- **CouchDB**: Used as database framework to store the NoSQL data. Has been explained and studied during the course (Apache License Version 2.0)
- **HTML**, **CSS** e **JavaScript**: Used for the realization of the user interface
- **Bulma**: Used as CSS framework to create easily the UI
- **Docker**: Usad the the containers management

Every choosed technology have contribuited significantly for the succesfull realization of the project, giving us specific solution and support for the different system part. The open source license of this technology give the opportunity to use this tools for free and to modify the source code

3 Design and implementation

Our project can be divided into four sub-sections: backend, frontend, database, server. These are described as following and are weel explained by this flowchart


ER SERVER
translate_from_json_to_hl7(json)
HL7
Translator_JSON_to_HL7.py
EMERGENCY ROOM
POST : /send_to_cardiology [json]
POST : /send_hl7_message [hl7]
C SERVER
translate_from_hl7_to_json (hl7)
JSON
Translator_HL7_to_JSON.py
GET :/getExams(ID, Exam, Date)
EXAMS
DB.getExam
DB.addNewExam [json]
APPLICATION DATABASE
[json]

3.1 Backend

On the first start of the application, the user (the health professional present in the emergency room) has the possibility of sending the data of the patient and the results of some basic analysis to the cardiology department of the hospital or viewing the data of the patients stored in the database. Regarding the first part of the application which involves the form that stores data in the database, we first serialized a JSON object containing the data coming from the form, to translate it into an HL7 message as it is necessary for the hospital because it accepts only standardized messages.

These messages are being encoded and transformed into HL7 format for a better data management and sent to the cardiology department into the payload of a POST HTTP request. In the specific case we're dealing with, the HL7 message is shaped as an ADT A15 event, where ADT stands for Management of patient (admitting, discharging, updating, transferring, etc.) and we chose the A15 trigger event because the patient must be transferred to another location, after completion of a first basic health scan. The HL7 message is shaped like this:

- An MSH segment, that is the message header, which contains information about the message:
  - the sending application,
  - the sending facility,
  - the receiving application,
  - the receiving facility,
  - the current time,
  - the trigger event, set as ADTA15,
  - a hash code for the message ID, made by a fixed secret code, the current time, and the name of the receiving application,
  - the accepted acknowledgment type, set as “AL”, that means “always” accepted.
- A EVN segment, that is the event trigger, set as A15 and contains:
  - the current time,
  - the operator ID.
- A PID segment that contains data about the patient:
  - Patient ID, that is represented by their fiscal code,
  - Name,
  - Family name,
  - Date of birth,
  - Gender,
  - Address, which is represented by:
    - the street address,
    - the city,
    - the province,
    - the postal code,
    - the country code,
    - the type of address, which is fixed as “L” (“Legal address”),
  - Phone number,
  - Primary language,
- An OBR segment for the urine test data, which contains:
  - The placer order number,
  - A hash code for the filler order number, made by the fiscal code of the patient, the name of the test and the timestamp,
  - The Service Identifier made up of the SNOMED universal service identifier for “Urea and Electrolytes” and the test name,
  - The requested date and time,
  - The observation date and time,

- Three OBX segments containing the details of the urine test. Each one of the OBX segments must store data about the SNOMED universal service identifier, the observed value, the unit of measurement and the reference range for:
  - Potassium,
  - Sodium,
  - Urea,
- An OBR segment for the blood test data:
  - The placer order number,
  - A hash code for the filler order number, made by the fiscal code of the patient, the name of the test and the timestamp,
  - The Service Identifier made up of the SNOMED universal service identifier for “Blood Pressure” and the test name,
  - The requested date and time,
  - The observation date and time,
- Two OBX segments containing the details of the blood test. Each one of the OBX segments must store data about the SNOMED universal service identifier, the observed value, the unit of measurement and the reference range for:
  - systolic blood pressure,
  - diastolic blood pressure,
- A NTE segment representing the narrative text information about the visit.

We chose to use SNOMED Universal Identifier to easily implement the IDs into CouchDB and because we preferred a format that is the most common among the hospitals’ environments.

We implemented a single server that simulate both the Emergency Room server and the Cardiology Department one, so the parsing of the JSON object into the HL7 message is like it’s being done by the Emergency Room, then the HL7 message is parsed into JSON object in order to have neater data for storing it into the database and to view data about patients when the cardiology room needs it.

The function that translates the HL7 message into a JSON object uses the python-hl7 library that gives also indexed access to repetitions and components inside a field of a HL7 message, so there is no need to split manually each row of the message.

The new JSON object contains three dictionaries, one for the patient’s data, one for the operator’s data and the last one for both the tests of “Urea and Electrolytes” and “Blood Pressure” observed values.

We chose to retrieve data from the form in JSON format then convert in HL7, and the from HL7 to JSON again, to mantain interoperability. Now we are developing a little application but one day maybe we will need to implement another system to visualize data on mobile application and having data in a JSON format is a good practice for that purpose.

3.2 Database

The json produced by the Traslator, is sendend to the database to be memorized. The database implementation is composed of three different table:

- **Patient Table**: That store the data of the patients
- **Operator Table**: That store the data of the operators that work in the emergency romm
- **Exam Table**: That store the data of each exam

The table Exam put in a relationship the other 2 thanks to `idPatient` and `idOperator`.

We used the class `CouchDBClient.py` to build the structure of the database and then we implement all the method needed to store new Data and query the tables to visualize specific data in the frontend.

The following methods has been implemented:

- `addOperators` and `addPatient`, these functions's aim is to receive operator and patient data, format it correctly and store it securely in the corresponding databases.
- `addExam`, these functions's aim record the results of examinations carried out on patients. It receives all the necessary information about the test, including the type of test (such as "UREA AND ELECTROLYTES" or "BLOOD PRESSURE"), the measured values and the operator responsible for carrying it out.
- `addNewExam`, these functions's aim is to add a new Exam, taking in input the full JSON, and pass the good parameters to the functions previusly explained.

  Before recording a new exam, this function checks whether both the operator and the patient involved are already present in the respective databases. If they are not, it adds the operator and/or patient first.
- `getExamByFiscalCodeAndDate`, `getExamByFiscalCode`, `getExamByTestName`, `getExamByFiscalCodeAndDateAndTestName`, `getExamByFiscalCodeAndTestName` , are all method needed to take the information that a user from the Cardiology page could ask about. Note that each method return an array of `visit` and an array of `patient` that appear in the visits.

A series of other methods have been implemented to create a system that in the start generate randomly (following the correct syntax of the data) 20 visit to add to the database, this permise to have something to visualise at the start of the application.

Note that any of the records in the database cannot be deleted or modified with any method, this becouse, as seen during the lessons, in the context of health application nobody should be able to do that.

3.3 Server

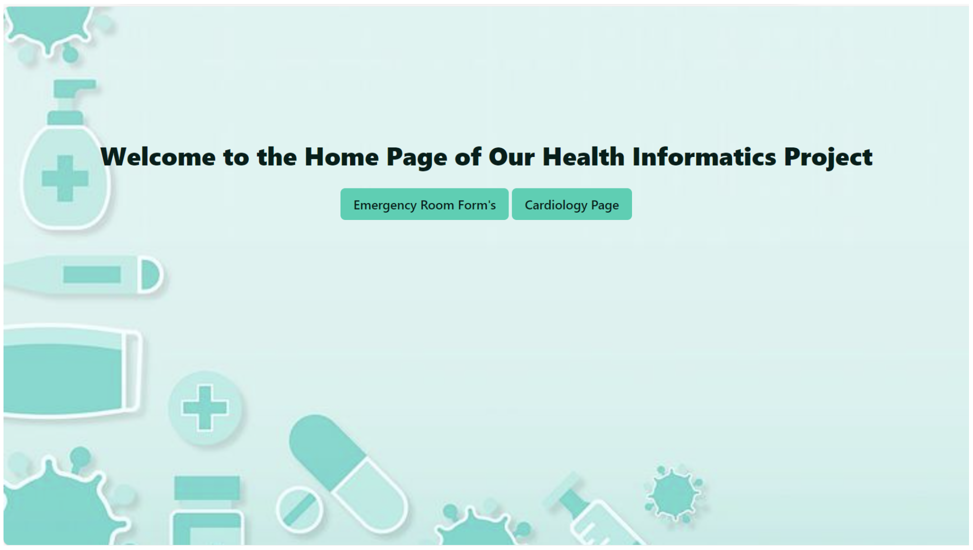
The `Server.py` is the class that manage the communication between the others implementation. It implements the REST API, to send and recive message as shoed in the previous image.

4 Results

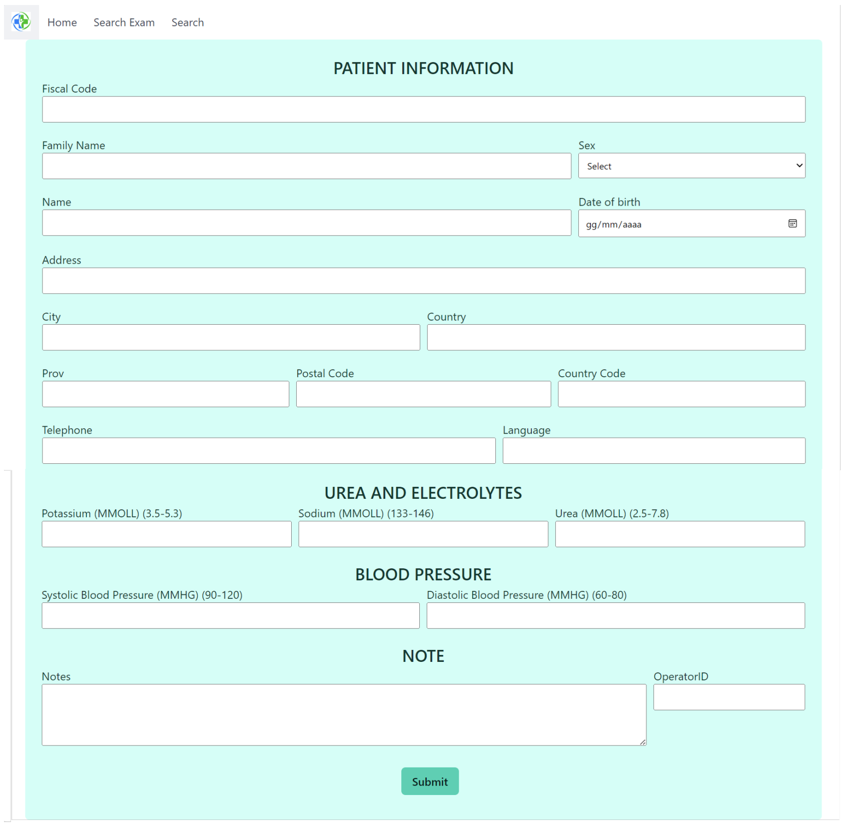
The final result is a meso-level application that simulates the communication between two distinct sectors of the same medical institution. The final goal is to optimize the process of registration and access to medical data for healthcare professionals.

The final user interface (frontend), which materializes all the backend work) is composed of 3 differents HTML pages:

- **Home** that give to the users the possibility to move among other pages, and have the following interface:



- **Emergency Room Form's** that permise two add a visit for a specific patient:



PATIENT INFORMATION

Fiscal Code

Family Name

Sex

Select

Name

Date of birth

gg/mm/aaaa

Address

City

Country

Prov

Postal Code

Country Code

Telephone

Language

UREA AND ELECTROLYTES

Potassium (MMOLL) (3.5-5.3)

Sodium (MMOLL) (133-146)

Urea (MMOLL) (2.5-7.8)

BLOOD PRESSURE

Systolic Blood Pressure (MMHG) (90-120)

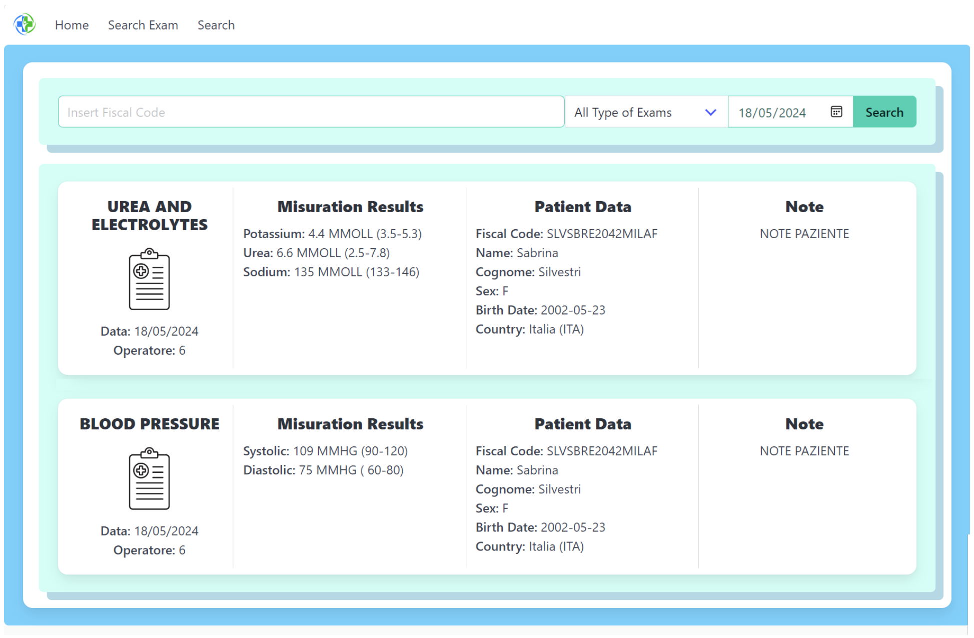
Diastolic Blood Pressure (MMHG) (60-80)

NOTE

Notes

OperatorID

- **Cardiology Page** that permise use to get visits searching for FiscalCode, Data and Visit Type:



UREA AND ELECTROLYTES
Data: 18/05/2024
Operatore: 6

Misuration Results
Potassium: 4.4 MMOLL (3.5-5.3)
Urea: 6.6 MMOLL (2.5-7.8)
Sodium: 135 MMOLL (133-146)

Patient Data
Fiscal Code: SLVSBRE2042MILAF
Name: Sabrina
Cognome: Silvestri
Sex: F
Birth Date: 2002-05-23
Country: Italia (ITA)

Note
NOTE PAZIENTE

BLOOD PRESSURE
Data: 18/05/2024
Operatore: 6

Misuration Results
Systolic: 109 MMHG (90-120)
Diastolic: 75 MMHG (60-80)

Patient Data
Fiscal Code: SLVSBRE2042MILAF
Name: Sabrina
Cognome: Silvestri
Sex: F
Birth Date: 2002-05-23
Country: Italia (ITA)

Note
NOTE PAZIENTE

- Present the results of your project. Highlight its key features and functionalities.
- Describe how users interact with your project and what they can achieve with it (this can include up to two pages of screenshots, but make sure not to be redundant with the video).
- Provide statistics about your source code (number of lines of code in the different programming languages).

5 Discussion

The project has been a success in achieving its main objectives and solving the stated problem. The interface is intuitive and easy to use for all healthcare professionals, ensuring a reduction in the time needed to record and access medical data and greater accuracy of the recorded medical information.

To ensure the effectiveness, reliability and correctness of our project, we adopted a testing technique for the translator, creating a JSON Object with fictitious data, in order to verify the efficiency of HL7 message translating.

6 Future work

Several improvements for the application are:

- An improvement that can be done on that project can be adding a second database for immediate data storing right after submitting the form. This database can be used for temporary backup, so this solution ensures to send data again in case of a loss during the transfer to the main database and facilitate a quick data recovery without having to resort to more complex backup procedures.
- A single server has been implemented to handle all operations for simplicity. However, to improve efficiency and security, it would be ideal to have two servers dedicated separately to the emergency department and the cardiology department, each on different machines.

7 Conclusion

Working on this project and on the topic of healthcare informatics has been interesting and educational for us. We have understood how important it is to have efficient systems for managing medical data, as they improve the quality of care and reduce response times for healthcare professionals. We have learned the best practices for protecting sensitive patient data, such as using backups and dedicated servers.

We have also deepened our knowledge of interoperability standards like HL7, which allow different healthcare systems to communicate securely and effectively. We have seen how crucial it is to design user-friendly systems for healthcare professionals, as an intuitive interface facilitates their work. Additionally, we have learned how to integrate advanced technologies to make the system more robust and efficient, taking into account the available resources.

8 References

For the SNOMED Universal Identifiers:

- Urea and electrolytes - https://www.findacode.com/snomed/252167001–urea-and-electrolytes.html?hl=urea+and+electrolytes
- Potassium - https://www.findacode.com/snomed/88480006–potassium.html?hl=88480006
- Sodium - https://www.findacode.com/snomed/39972003–sodium.html?hl=39972003
- Urea - https://www.findacode.com/snomed/387092000–urea.html?hl=387092000
- Blood pressure - https://www.findacode.com/snomed/75367002–blood-pressure.html
- Diastolic blood pressure - https://www.findacode.com/snomed/271650006–diastolic-blood-pressure.html
- Systolic blood pressure - https://www.findacode.com/snomed/271649006–systolic-blood-pressure.html