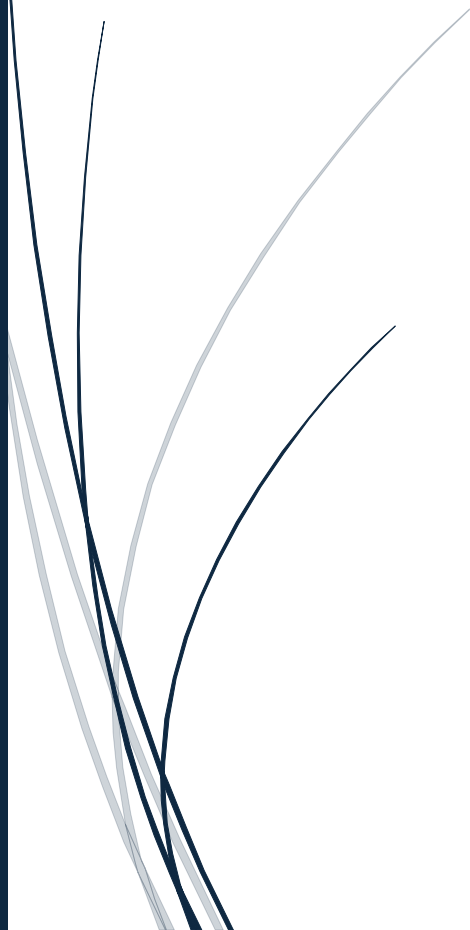




1-12-2025

TELEMEDICINE PRACTICE MEMORY

CEU SAN PABLO – 4º BIOMEDICAL ENGINEERING



Raquel Hernanz, Pablo García, Marta Álvarez,
Yolanda Luchelli, Sandra Cabrera

Indexe

1. Introduction	2
1.1. Proposal	2
1.2. Main objectives of the practice	2
2. Available actions in the system	3
3. Communication Protocol.....	5
3.1. TCP VS UDP selection	5
3.2. JSON and Decorators	5
4. Code and applications' diagrams	7
4.1. Protocol Communication Diagrams	7
4.2. Sequence Diagrams (SD)	8
4.3. Use Case Diagram	13
4.4. E-R Diagram	14
4.5. UMLs	14
5. Optional points.....	19
5.1. Extra functionalities implemented	19
5.2. About the graphical interface	20
6. User manuals	21
6.1. Patient	21
6.2. Doctor	23
6.3. Administrator/Server	26

1. Introduction

1.1. Proposal

Cardiac arrhythmias are chronic conditions in which the heart beats irregularly, too fast, or too slow, often disrupting normal systole-diastole cycles (**Figure 1**). These irregular episodes frequently appear unpredictably and may not be captured during routine check-ups, which complicates early detection and increases the risk of complications such as syncope, heart failure, or stroke. Continuous home monitoring therefore plays a crucial clinical role.



Figure 1. Cardiac arrhythmias in an ECG

Our project proposes a telemedicine application designed to remotely supervise patients with suspected or confirmed arrhythmias, integrating self-reported symptoms together with physiological signals captured by a BITalino device, specifically ECG and EDA. This approach enables real-time monitoring from the patient's home while allowing the hospital to remotely receive, store, and analyse data relevant to arrhythmic events.

The system aligns with the course's requirement of building a telemedicine platform for chronic-disease supervision, including patient reporting, physiological recording, Bluetooth communication with BITalino, and multi-client server management. It also follows the disease-specific design described in the team's proposal on arrhythmia monitoring.

Links for GitHub repositories:

https://github.com/RaquelHernanz/ServerCode_Telemedicine_2025.git

https://github.com/RaquelHernanz/DoctorCode_Telemedicine_2025.git

https://github.com/RaquelHernanz/PatientCode_Telemedicine_2025.git

1.2. Main objectives of the practice

- 1- **Enable remote monitoring of cardiac arrhythmias:** the system will allow patients to report symptoms such as palpitations, dizziness, chest pain, or

fatigue and transmit physiological parameters (ECG and EDA) from home to the server, as required by the final project guidelines.

- 2- **Integrate BITalino-based signal acquisition:** the patient application will establish a Bluetooth connection with the BITalino device (**Figure 2**) and record ECG and EDA signals, which are clinically relevant for detecting arrhythmias and associated stress responses as highlighted in the proposal.

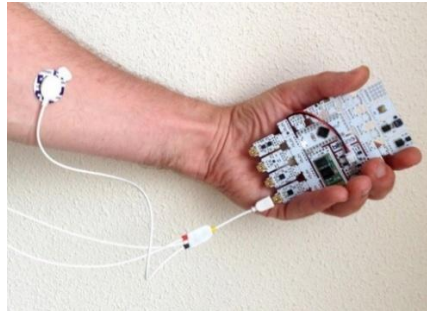


Figure 2. *Bitalino device*

- 3- **Enable doctor-side access:** the design of the telemedicine application anticipates the integration of a doctor client capable of accessing and reviewing patient data.
- 4- **Implement a multi-client server:** The server must support concurrent patient and doctor connections, sending and receiving symptom forms and physiological data, and store them. It must also provide a controlled shutdown mechanism with administrator authentication, as mandated by the project specification.
- 5- **Provide secure, organized data management:** the user information will be stored with timestamps and patient identifiers, following the project requirement to maintain clinical history in a structured manner on the server side.

2. Available actions in the system

This section summarizes the set of actions available to each role within the telemedicine system. Patients, doctors, and administrators interact with the platform through different capabilities aligned with their responsibilities, enabling data submission, clinical review, communication, and system control. The following overview organizes these actions into functional areas, clarifying how each user type participates in the overall workflow of the application.

User and access management

Related to the authentication and identity management

- Register user – Patient, Doctor.
- Login – Patient, Doctor.
- Authenticate administrator for server shutdown – Administrator.

- View connected clients on server – Administrator.
- View personal information – Doctor, Patient.
- Log out – Patient, Doctor

Clinical Data Input

Actions sent from the patient application to the server. Includes:

- Submit symptom report (palpitations, dizziness, chest pain...) – Patient.
- Record and send ECG/EDA via BITalino – Patient.
- Request appointment – Patient.

Clinical Data Review

From doctor to server

- View patient symptoms – Doctor.
- View patients ECG/EDA measurements – Doctor.
- View upcoming appointments – Doctor, Patient.

Messages and communication

Message exchange between patients and doctors.

- ❖ Send messages – Patient, Doctor.
- ❖ View received messages – Patient, Doctor.

Server management

Exclusive actions for the administrator

- Shutdown the server – Administrator.
- Monitor client connections – Administrator.
- Centralize data storage – Administrator.
- Synchronize patient–doctor data flow – Administrator.

Summary table

The following table (

Table 1) resumes the available actions for each user:

Patient	Doctor	Administrator/Server
Register	Register	Server shutdown
Login	Login	Monitor the connections
Log out	Log out	Control the system database
Send measurement of EDA/ECG to doctor	View measurements from patient	-
Send symptoms to doctor	View symptoms from patient	-
Send and view messages	Send and view messages	-

Request and view appointment	View appointments	-
------------------------------	-------------------	---

Table 1. Available actions summary

3. Communication Protocol

3.1. TCP VS UDP selection

For the communication layer of the telemedicine system, we chose TCP sockets instead of UDP. The application must reliably transmit clinical information such as ECG/EDA measurements, symptom reports and appointments between patients, the central server and doctors. These data cannot be lost, duplicated or arrive out of order without compromising medical interpretation. TCP provides a connection-oriented, reliable byte stream with built-in error detection, retransmission and in-order delivery, which fits the project requirement of sending patient signs and symptoms to a server and storing them as part of the clinical history.

In contrast, UDP is connectionless, does not guarantee delivery or ordering, and would force us to implement our own reliability mechanisms on top of the protocol. Since our priority is integrity and robustness rather than ultra-low latency, TCP is the most appropriate choice.

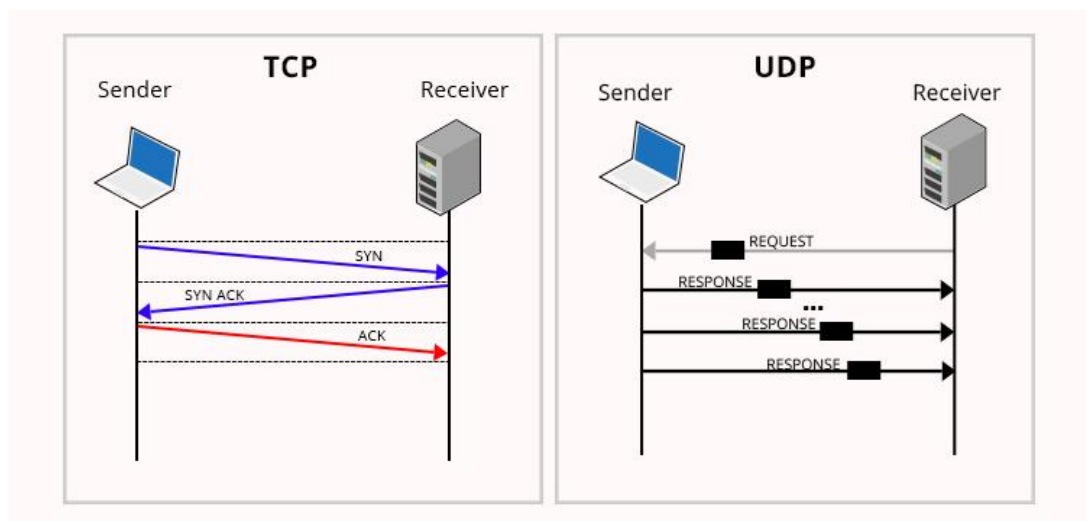


Figure 3. TCP VS UDP comparison

3.2. JSON and Decorators

On top of the TCP sockets, we apply a decorator-based design to exchange data. Instead of letting the rest of the application work directly with raw *InputStream* and *OutputStream* objects, each socket is wrapped by several layers (decorators) that progressively add responsibilities: buffering, character encoding, message framing and, if needed, logging or encryption.

Both server and clients wrap the socket streams with higher-level reader/writer classes that send and receive text messages following our JSON-based protocol (**Figure 4**). We chose JSON format because it has a clear and predictable structure for information exchange, the administrator and programmers can easily read and debug, avoid ambiguity, and is widely used across the internet networks and medical platforms. As a result, the application code only deals with domain messages from enumerates (e.g. LIST_SYMPTOMS, SEND_MEASUREMENT, REGISTER, REQUEST_APPOINTMENT) and not with low-level byte manipulation.

```
How the messages are going to be structure.

{
  "type": "REQUEST",
  "action": "REGISTER_PATIENT",
  "role": "PATIENT",
  "requestId": "req-1",
  "payload": {
    "user": {
      "username": "rachel@example.com",
      "password": "secreto123",
      "role": "PATIENT"
    },
    "patient": {
      "name": "Rachel",
      "surname": "Hernanz",
      "email": "rachel@example.com",
      "phoneNumber": "+34999999999",
      "dob": "1999-04-10",
      "sex": "FEMALE"
    }
  }
}
```

Figure 4. Example of JSON format message

This use of decorators brings three main advantages:

- **Separates concerns:** networking mechanics (framing, encoding, retries) remain isolated from business logic (handling patients, doctors, symptoms and measurements).
- **Improves extensibility:** allow the addition of features such as symmetric encryption, compression or additional logging by inserting new decorators around the socket without changing the core protocol handlers, as suggested in the project's optional security features.
- **Simplifies testing and maintenance:** each layer (protocol parser, data storage) can be tested independently while relying on the same decorated interface for sending and receiving JSON messages over TCP.

4. Code and applications' diagrams

The following schemes represent sequence diagrams, use case diagrams, E-R diagram and the UMLs for each repository.

4.1. Protocol Communication Diagrams

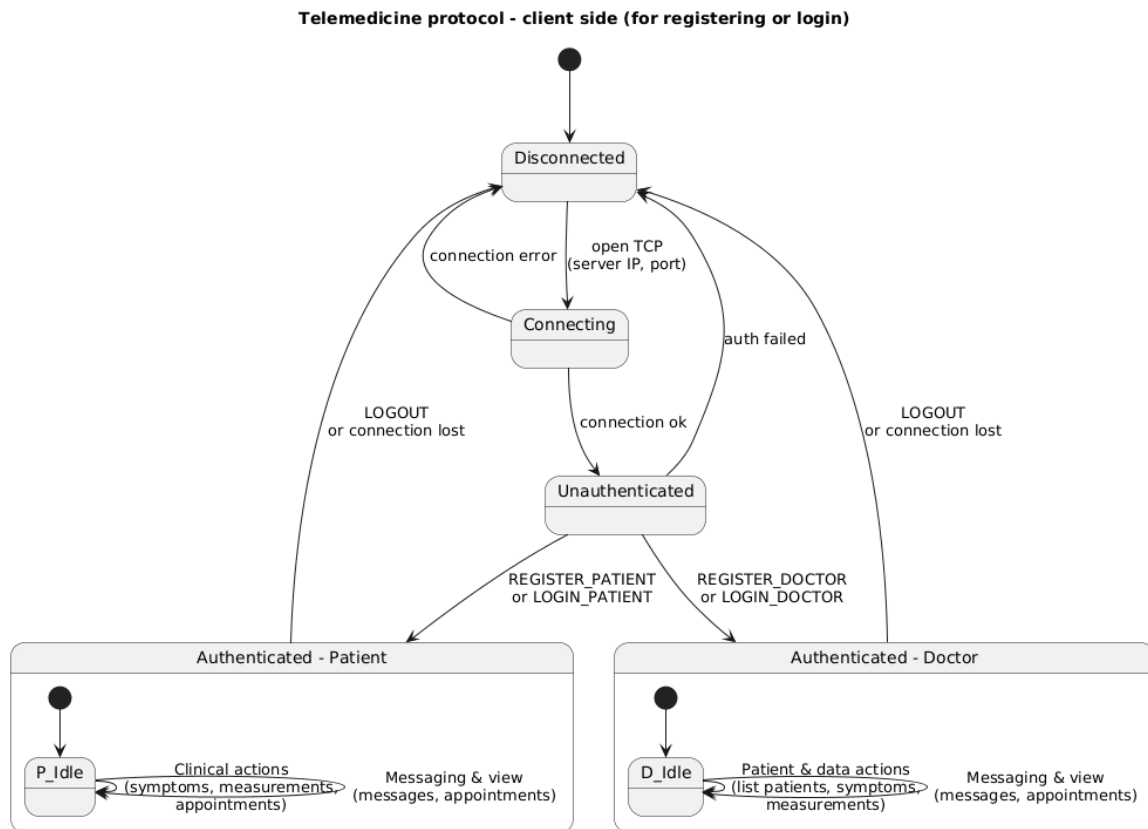


Figure 5. Communication protocol diagram for client

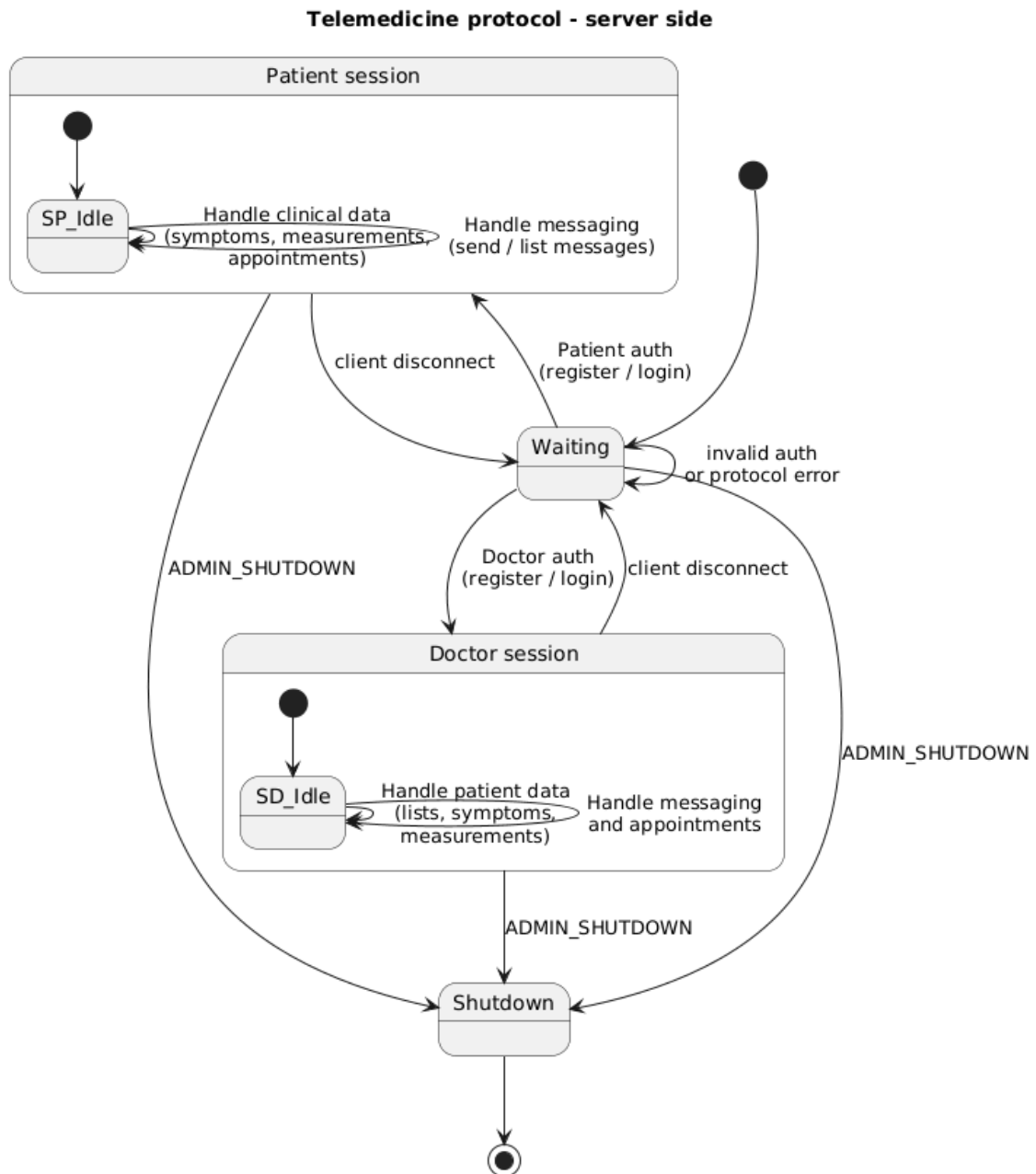


Figure 6. Communication protocol diagram for server

4.2. Sequence Diagrams (SD)

REGISTER PATIENT

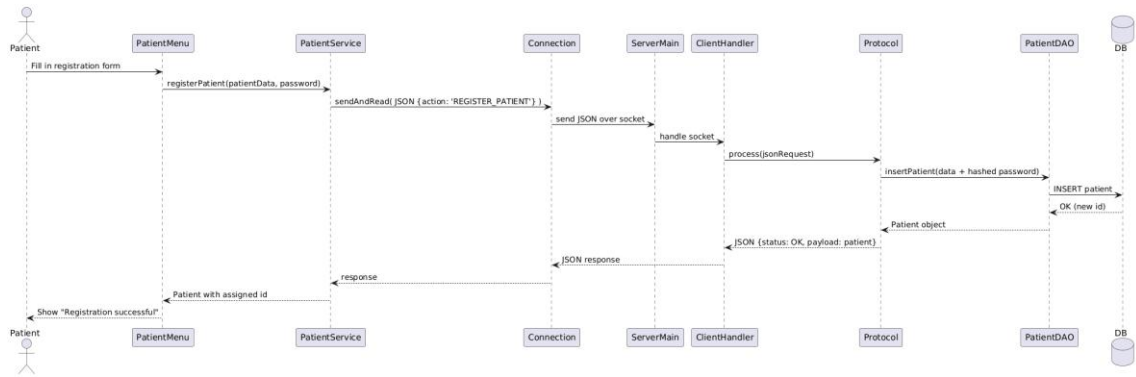


Figure 7. Patient registration SD

REGISTER DOCTOR

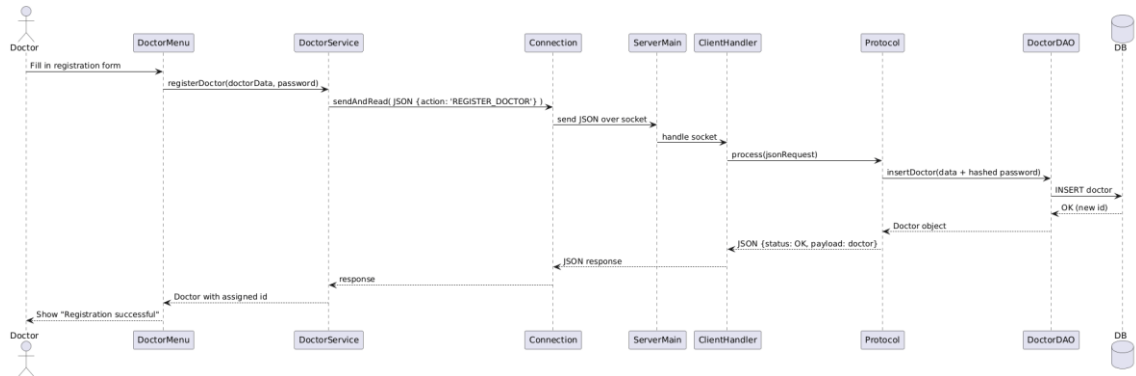


Figure 8. Doctor registration SD

LOGIN

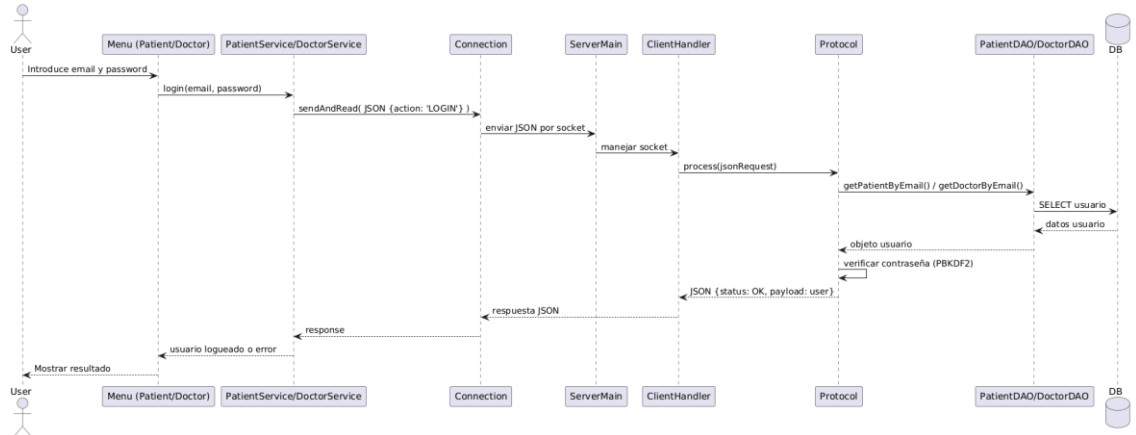


Figure 9. Login SD

SEND SYMPTOMS

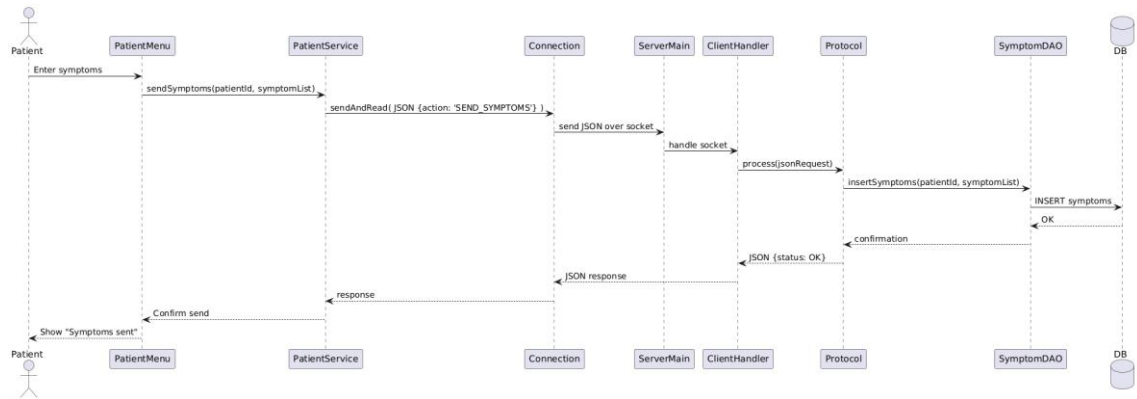


Figure 10. Send symptoms SD

SEND MEASUREMENT

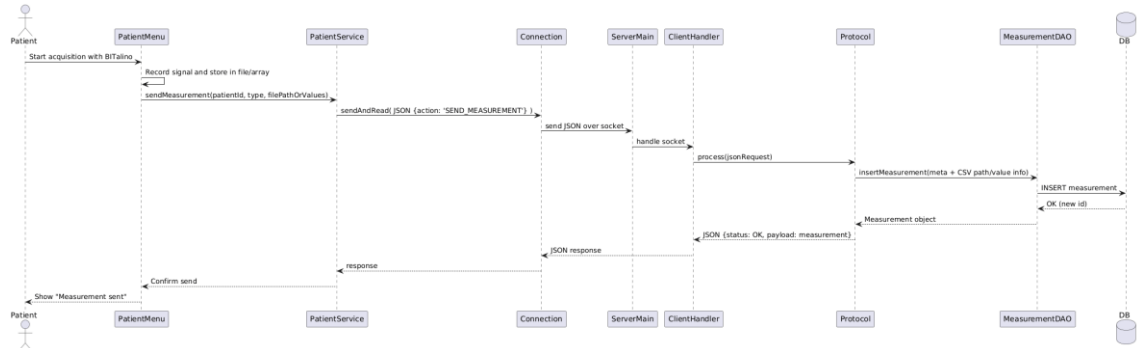


Figure 11. Send measurement SD

LIST PATIENTS

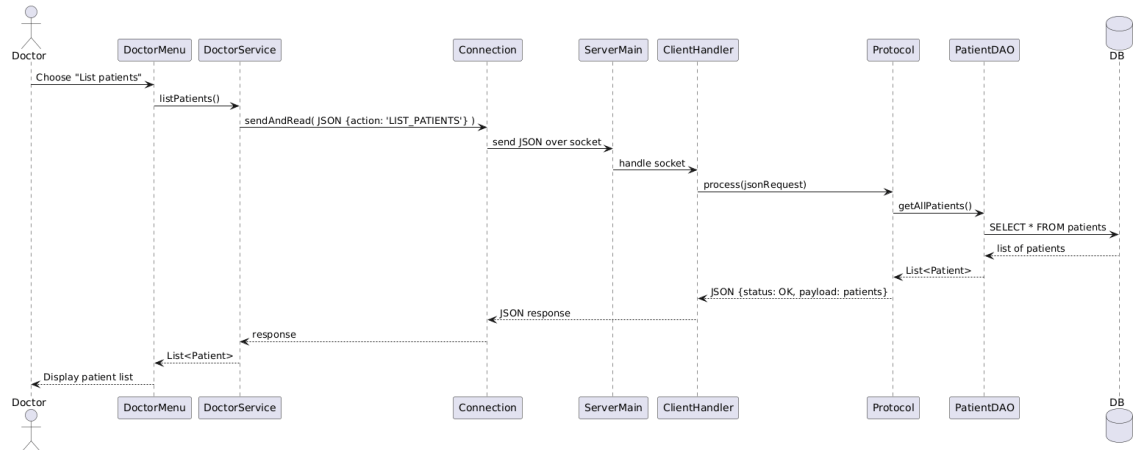


Figure 12. List of patients SD

LIST DOCTORS

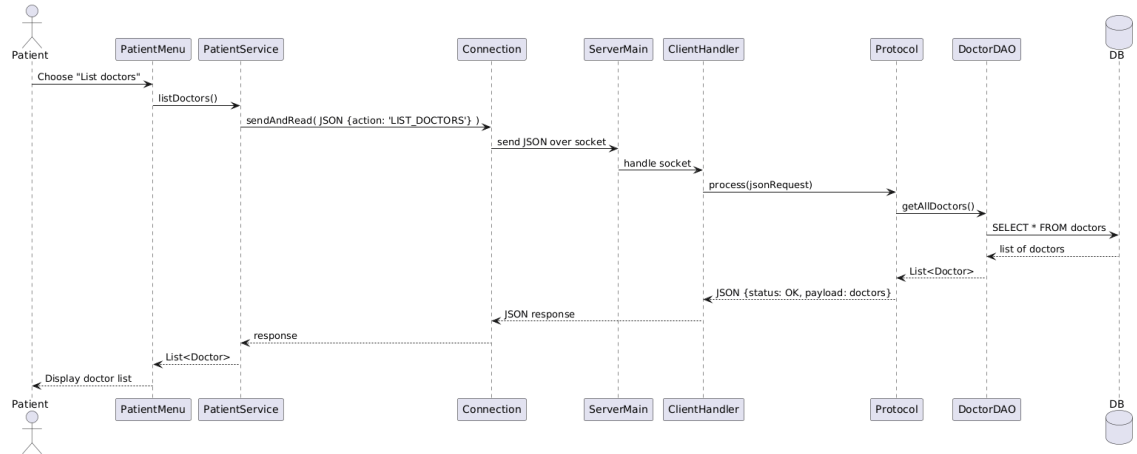


Figure 13. List of doctors SD

REQUEST APPOINTMENTS

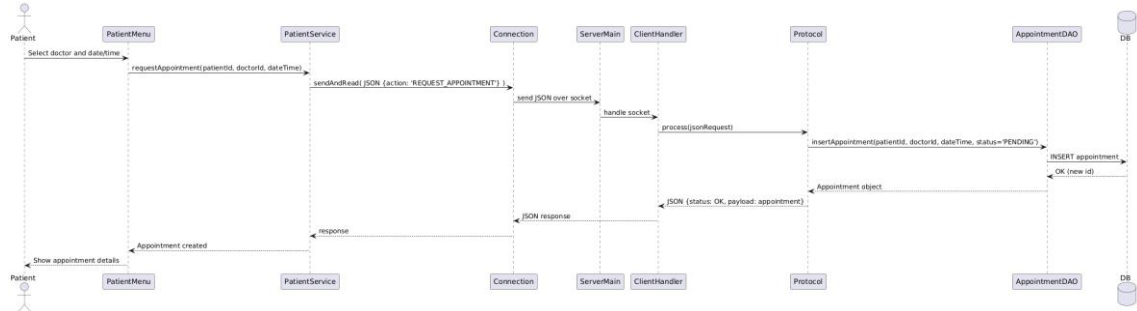


Figure 14. Request of appoint SD

LIST APPOINTMENTS

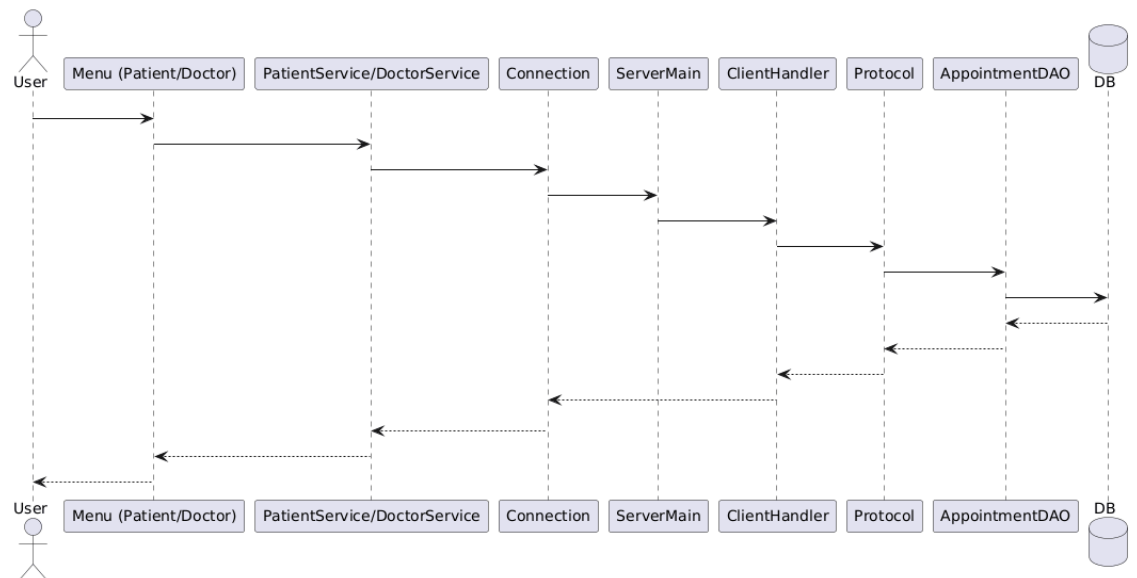


Figure 15. List of appointments SD

LIST MEASUREMENTS

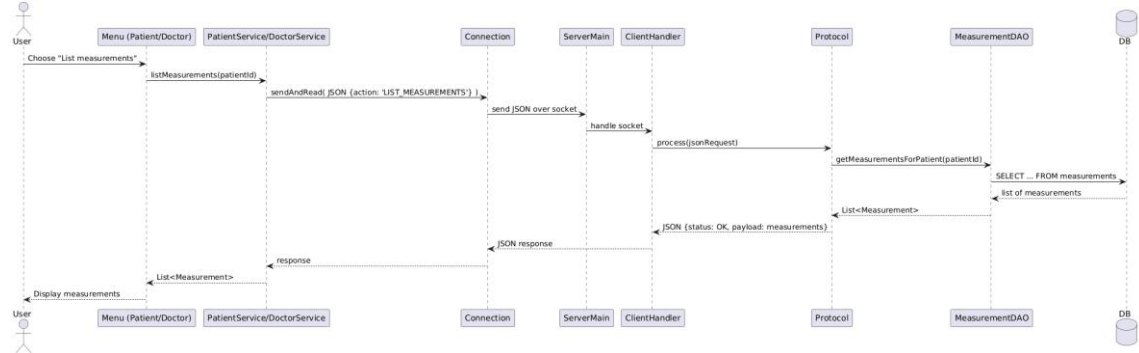


Figure 16. List of measurements SD

GET MEASUREMENT VALUES

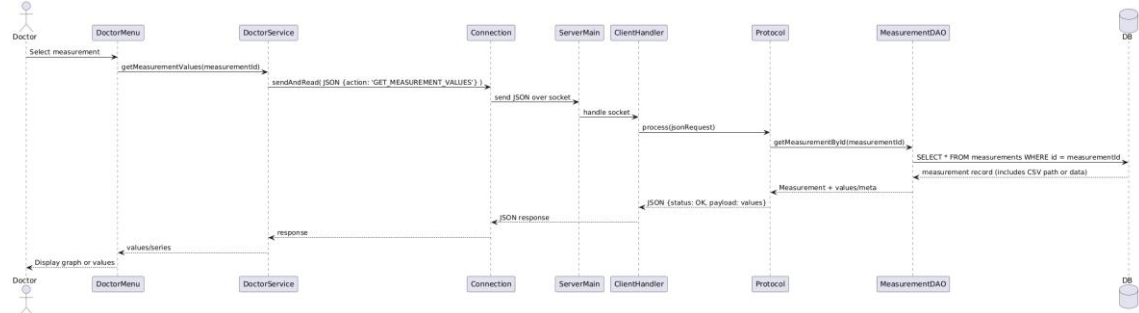


Figure 17. Get measurement values SD

LIST SYMPTOMS

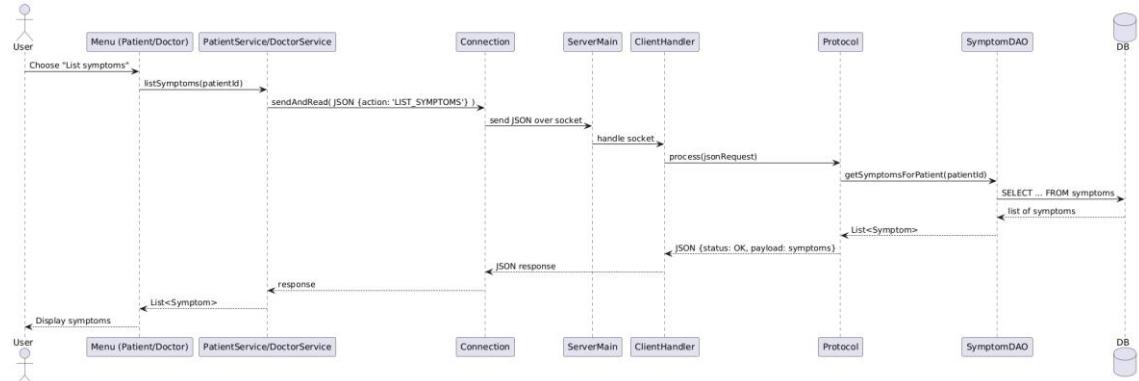


Figure 18. List of symptoms SD

SEND MESSAGE

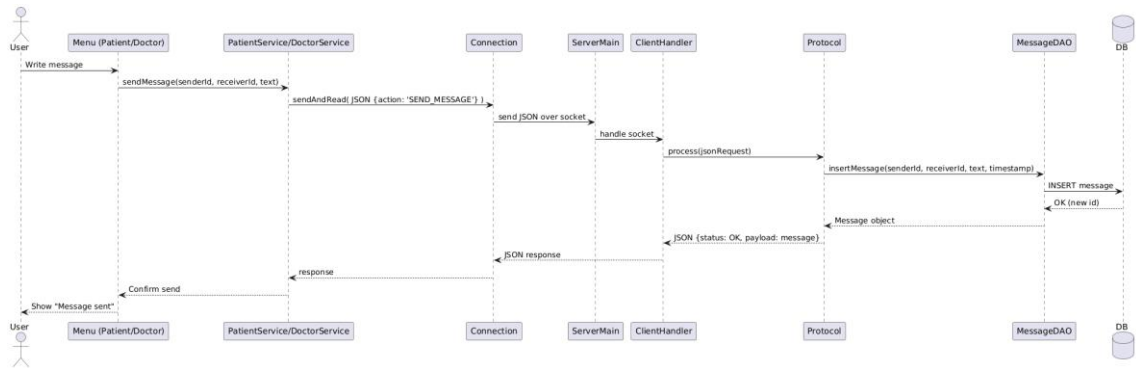


Figure 19. Send message SD

LIST MESSAGES

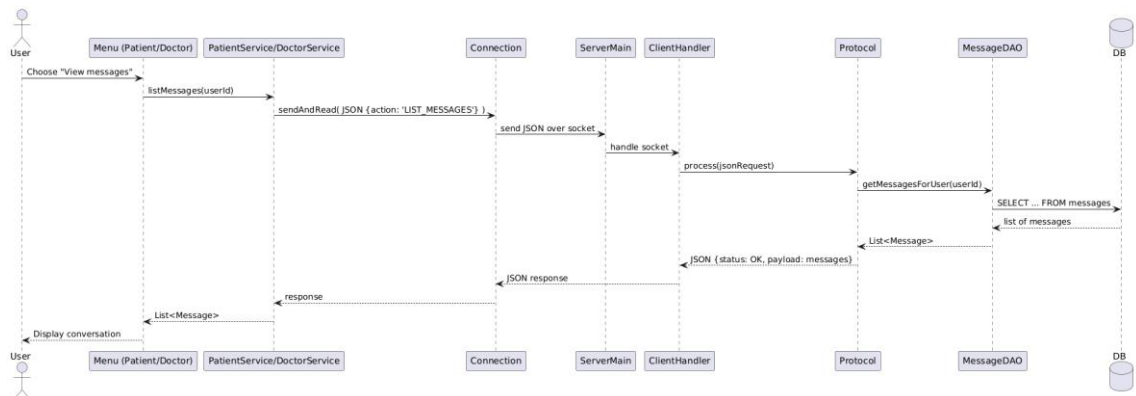
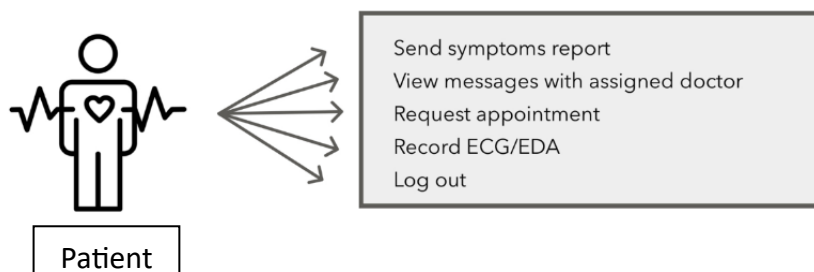


Figure 20. List of messages SD

4.3. Use Case Diagram



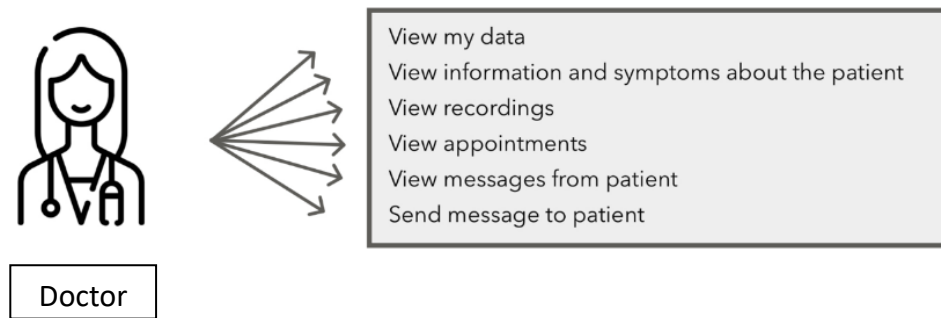


Figure 21. Use Case Diagram

4.4. E-R Diagram

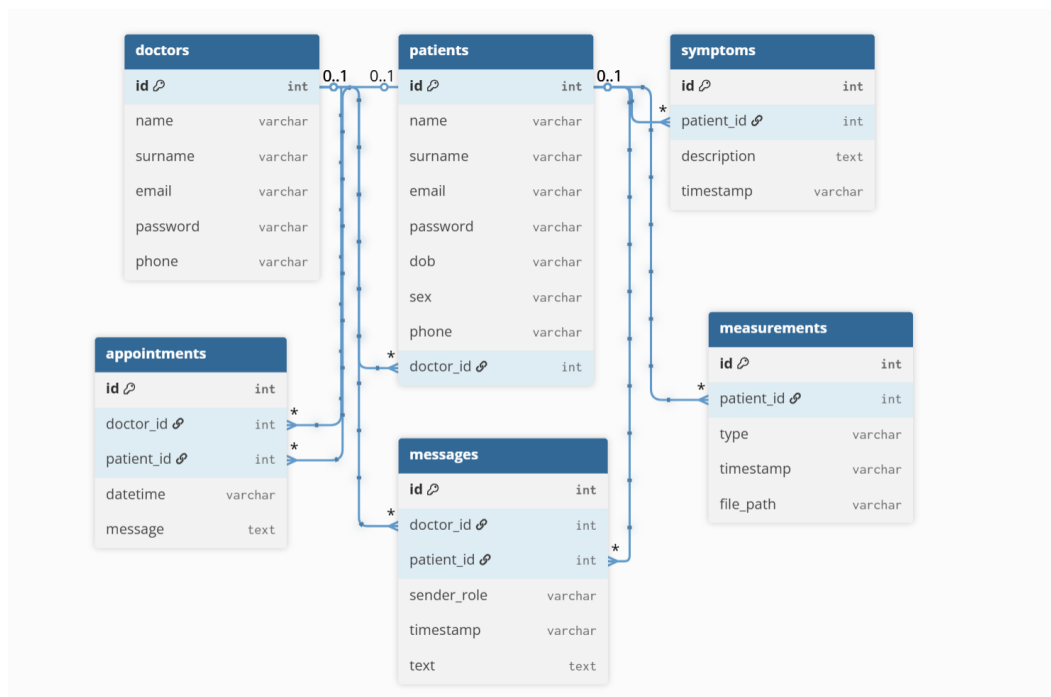


Figure 22. E-R Diagram

4.5. UMLs

UML Doctor

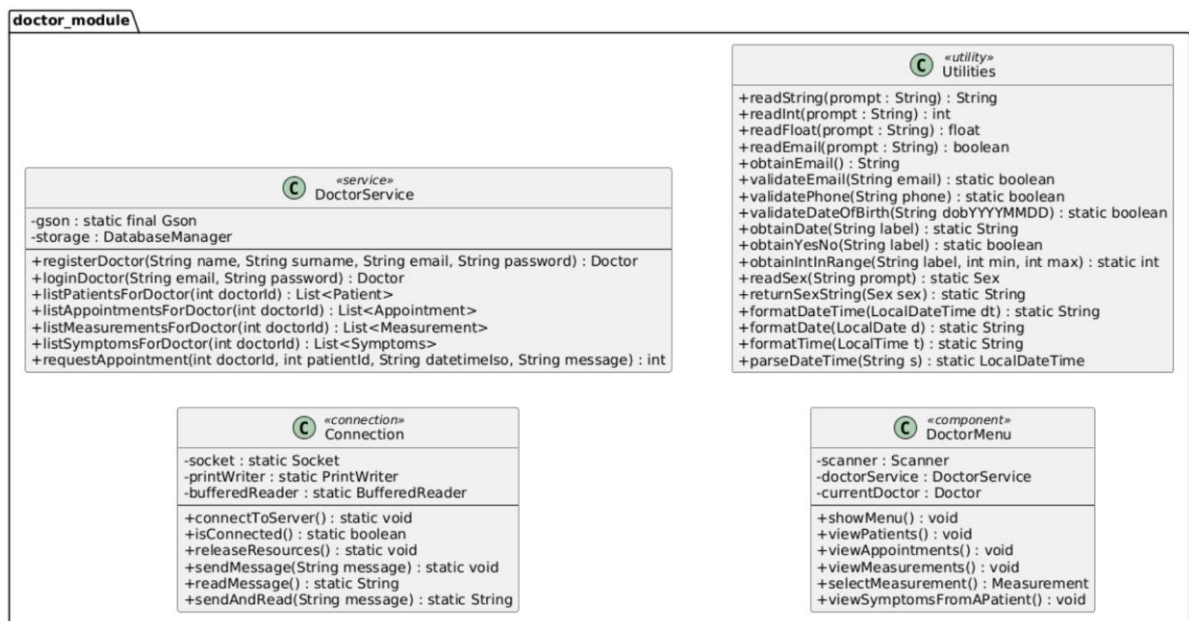
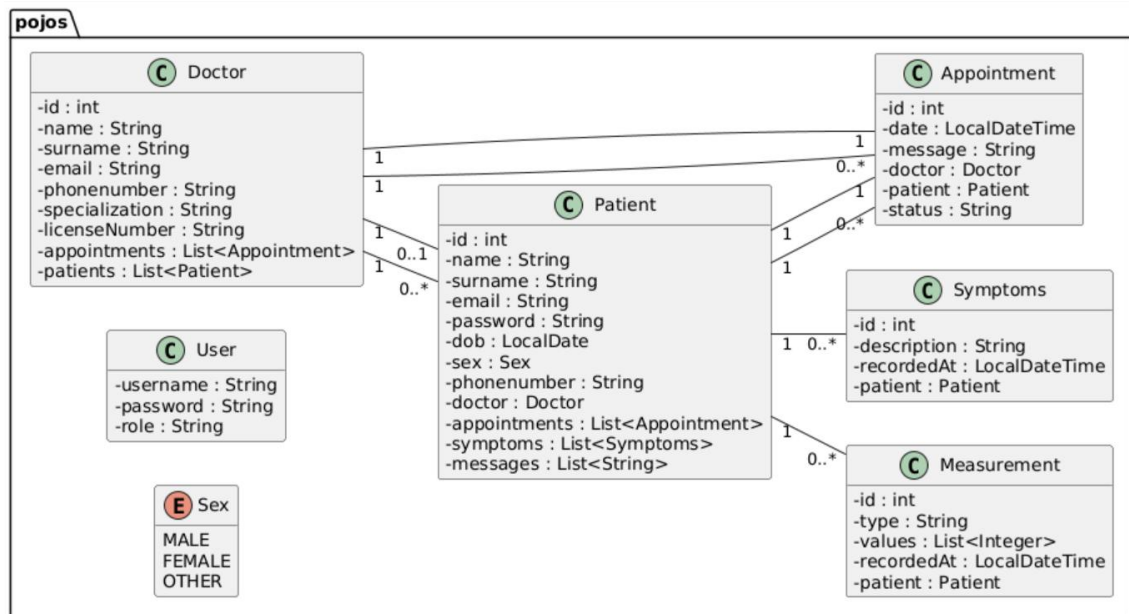


Figure 23. Doctor UML

UML Patient

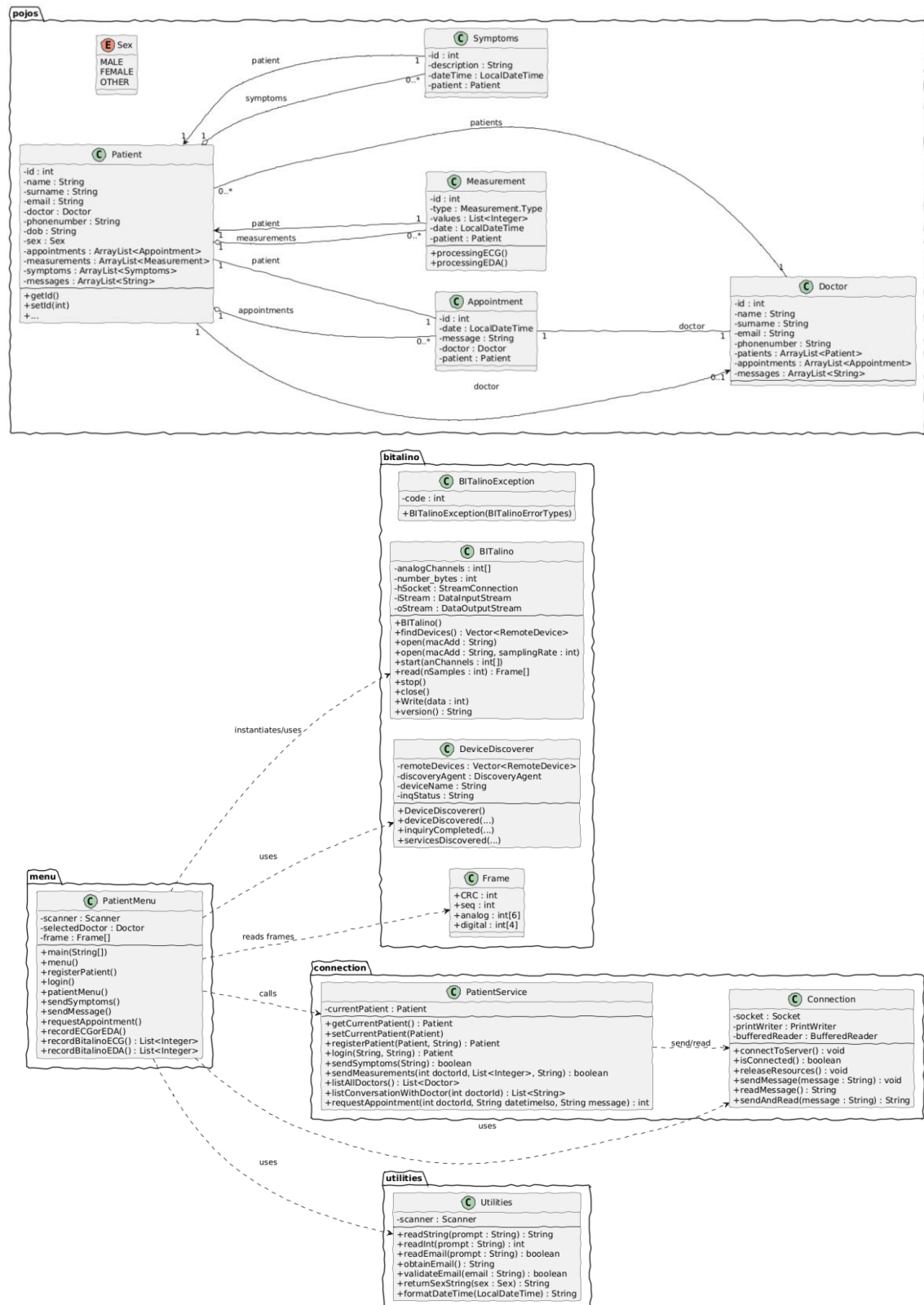
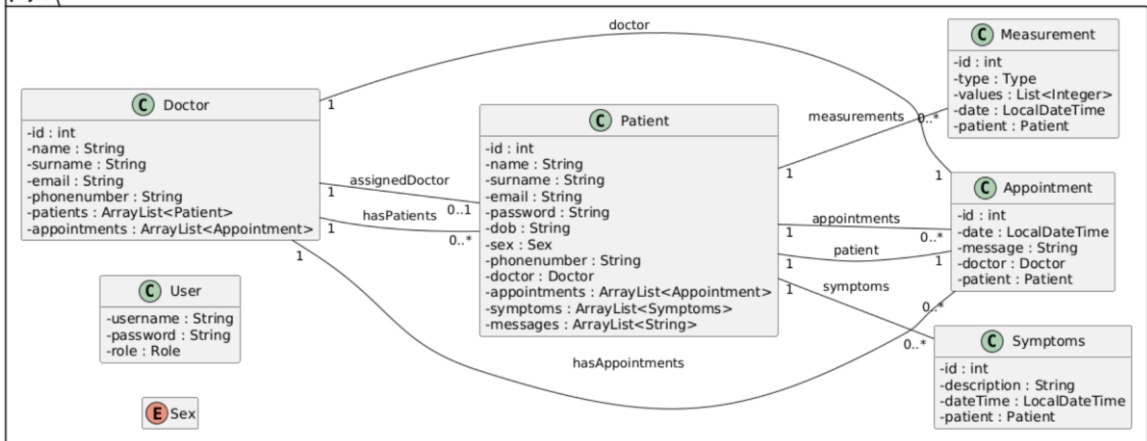


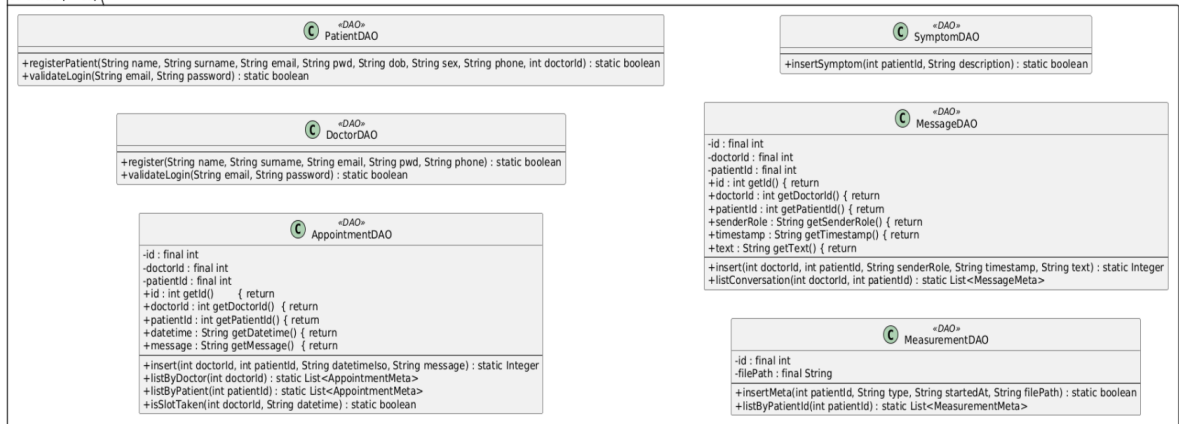
Figure 24. Patient UML

UML Server

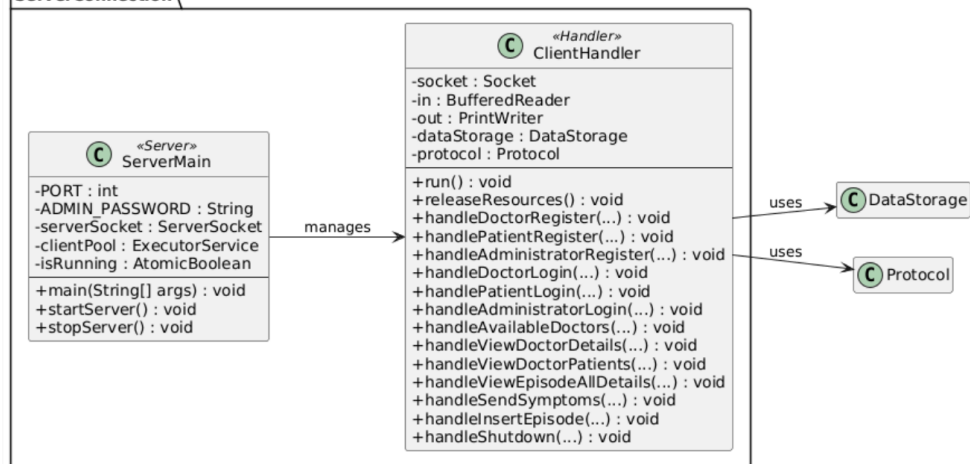
pojos



database (DAOs)



ServerConnection



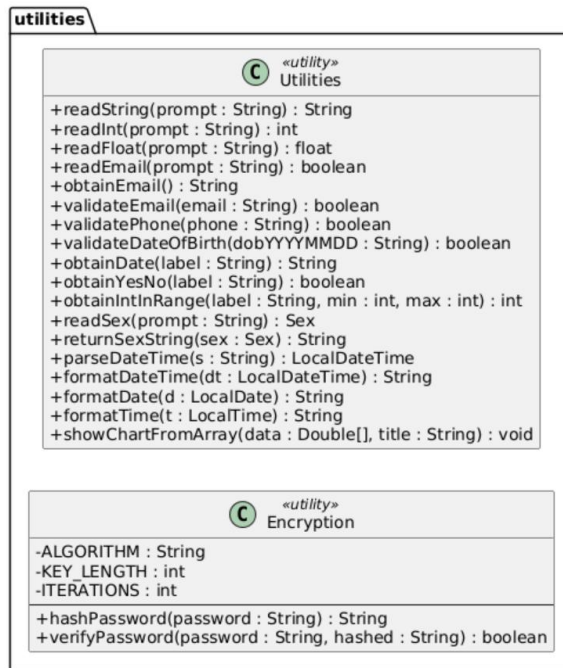


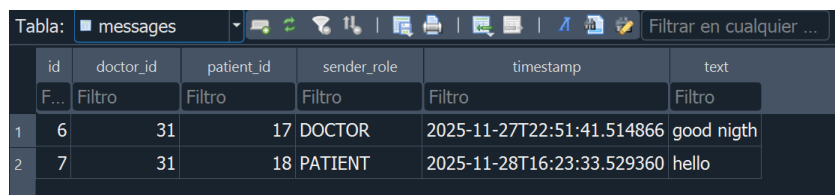
Figure 25. Server UML

5. Optional points

5.1. Extra functionalities implemented

With these extra functionalities, we accomplished to build a more complex and functional telemedicine system for our practice.

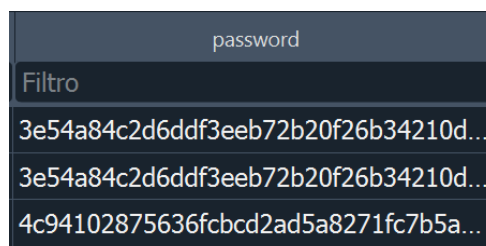
- ✓ **Database storage:** serves to store the information about the clients connecting to our application. In the E-R Diagram scheme the relationships of each POJO in the database are mentioned in this practice memory. The only values that are not uploaded to the database directly are the measurements' numerical values, which are sent from the Patient repository to the Server repository (contain in a CVS file in the data folder), and later they are retrieved for visualization in the Doctor repository.



	id	doctor_id	patient_id	sender_role	timestamp	text
1	6	31	17	DOCTOR	2025-11-27T22:51:41.514866	good nigh
2	7	31	18	PATIENT	2025-11-28T16:23:33.529360	hello

Figure 26. Example of data base table

- ✓ **Password encryption thought symmetric key:** we searched for various encryption methods; however, we chose Password-Based Key Derivation Function (PBKDF2) with Hash-based Message Authentication Code (HMAC) and SHA256. The idea is to transform a human password into a strong cryptographic key by repeatedly applying HMAC-SHA256 thousands of times together with a unique salt or random number, avoiding the passwords to have the same hash. This intentional computational cost makes each password-guess expensive, preventing attackers from performing fast brute-force or rainbow-table attacks. Instead of storing the password itself, the system stores the derived key, which can only be reproduced by providing the correct password and running the same iterative process.



password
Filtro
3e54a84c2d6ddf3eeb72b20f26b34210d...
3e54a84c2d6ddf3eeb72b20f26b34210d...
4c94102875636fcbcd2ad5a8271fc7b5a...

Figure 27. Encrypted password

- ✓ **Visualization of the clinical signal:** the doctor in the console or graphical interface is capable of viewing the measurement's signals from a specific patient, employing an interface frame. The doctor can distinguish between an EDA and ECG signals.

5.2. About the graphical interface

We decided not to integrate a graphical interface in the doctor and patient applications for the practice delivery on 1st of December due to time constraints during the development, and because we wanted to ensure the applications have a fully functional and reliable system over having incomplete implementation. For later versions we would add additional functionalities such as the graphical interface, connection to the server without introducing the specific IP, and also the asymmetric key encryption for the database information.

6. User manuals

Reminder: the clients must always specify the server IP, because it is not static (the server can be run in any computer but the IP changes).

6.1. Patient

Purpose of the Patient Application

The patient application allows individuals with cardiac arrhythmias to remotely send relevant clinical information to the hospital. This includes:

- Self-reported symptoms (palpitations, dizziness, chest pain, fatigue).
- ECG and EDA measurements recorded through a BITalino device.
- Messages and appointment requests to their assigned doctor.

Starting the Application

The patient opens the program and is presented with an initial menu that allows:

- Registering as a new user.

```
--- Patient Registration ---  
Email: patienttest@gmail.com  
Password: Patient  
Name: Jane  
Surname: Hopper  
Phone Number: 74398745  
Sex (M/F/O): F
```

Figure 28. Patient registration

- Logging in with email and password.
- Exiting the application.

Both registration and login are required to access the full functionality.

Main Functions Available (after login)

Once logged in, the patient accesses the main menu containing the following actions:

```
0. Log out  
1. Send symptoms report  
2. View messages with assigned doctor  
3. Send message to assigned doctor  
4. Request appointment  
5. Record ECG/EDA (Bitalino)  
6. View my personal information  
7. View appointments
```

Figure 29. Patient Menu

1. Send Symptoms Report

The patient can submit a brief description of current symptoms.
This report is automatically sent to the server and stored in their clinical profile.

```
--- Send Symptoms Report ---  
Describe your current symptoms (e.g., chest pain, dizziness): I had chest pain during exercise
```

Figure 30. Patient sends symptoms to doctor

2. View Messages

Displays all the messages previously exchanged with the assigned doctor.

```
=== Conversation with Dr. Duncan ===  
[2025-11-29T14:01:42.130210100] PATIENT: Hello  
=== End of conversation ===  
Logged as: patienttest@gmail.com (ID: 21)
```

Figure 31. View patient's messages

3. Send Message to the Doctor

Allows the patient to write and send a new message directly to the doctor.

```
--- Send Message to Dr. Duncan ---  
Message: Hello
```

Figure 32. Patient sends a message to the doctor

4. Request an Appointment

The patient can request a date and time for a future medical appointment.
The server registers the request so the doctor can review it.

```
--- Request Appointment with Dr. Duncan ---  
Appointment Date (yyyy-MM-dd): 2026-03-12  
Appointment Time (HH:mm): 12:30  
Reason for the appointment: Monthly check-up
```

Figure 33. Patient requests an appointment

5. Record ECG/EDA (BITalino)

The patient can start a BITalino recording session to capture:

- ECG (cardiac rhythm): port A2.
- EDA (stress-related skin conductance): port A3.

The application connects to the device via Bluetooth, add the MAC address, records a short measurement, and sends the data to the hospital server automatically.

```
Choose an option: 5  
Select 1 for ECG or 2 for EDA  
1  
Recording ECG  
BlueCove version 2.1.1-SNAPSHOT on winsock  
javax.bluetooth.LocalDevice@26f67b76  
[ ]  
Enter MAC address for Bitalino: 
```

Figure 34. ECG recording action

Since we are not recording, it should send a message that the ECG won't be sent.

```
No ECG data recorded. Measurement will not be sent.  
Logged as: patienttest@gmail.com (ID: 21)
```

Figure 35. No data recorded message

And if we are recording properly the signal it should send this message:

```
Enter MAC address for Bitalino: 20:17:11:20:52:32  
[Patient] Enviado: {"type":"REQUEST","action":"SEND_MEASUREMENT"  
[Patient] Recibido: {"type":"RESPONSE","action":"SEND_MEASUREMENT"  
ECG sent successfully.
```

Figure 36. ECG send message

6. View Personal Information

Shows the patient's stored personal detail.

```
PATIENTS INFORMATION:  
Name: Jane  
Surname: Hopper  
Email: patienttest@gmail.com  
Date of Birth: 1987-08-11  
Sex: FEMALE  
Phone Number: 74398745  
Logged as: patienttest@gmail.com (ID: 21)
```

Figure 37. Personal information

7. View Appointments

Displays upcoming or previously scheduled appointments registered on the server.

```
=== Appointments ===  
Date/Time: 2026-03-12T12:30  
Patient ID: 21  
Message: Monthly check-up
```

Figure 38. View patient's appointments

8. Log Out

Ends the current session and returns to the initial login menu.

6.2. Doctor

Purpose of the Doctor Application

The doctor application allows medical professionals to connect to the hospital server and supervise their assigned patients. The doctor can:

- View their own personal data.
- See information and symptoms of a selected patient.
- View physiological recordings (ECG/EDA) previously sent by the patient.
- Review the patient's appointments.
- Read messages received from the patient.
- Send messages to the patient.

Starting the Application

The doctor opens the program and is presented with an initial menu that allows:

- **Registering as a new doctor.**

```
Enter your personal details to register as a Doctor
Email: doctortest@gmail.com
Password: Doctor
Phone Number: 63483653
Name: George
Surname: Duncan
Enter the IP address of the server: [REDACTED]
```

Figure 39. Doctor registration

- **Logging in using email and password.**

```
Introduce email to enter the system:
Email: doctortest@gmail.com
Introduce your password
Password: Doctor
Enter the IP address of the server: [REDACTED]
Connecting to server...
```

Figure 40. Log in

- **Exiting the application.**

```
0
Logging out...
```

Figure 41. Log out

A successful registration and login are required to access the full functionality.

Main Functions Available (after login)

Once logged in, the doctor accesses the main menu containing the following actions:

```
0. Log out
1. View my data
2. View information and symptoms about the patient
3. View recordings
4. View appointments
5. View messages from patient
6. Send message to patient
```

Figure 42. Doctor Menu

1. View My Data

Displays the doctor's personal information stored in the system, including name, surname, email, phone number, and ID.

```

DOCTOR INFORMATION:
id: 33
Name: George
Surname: Duncan
Email: doctortest@gmail.com
Phone Number: 63483653

```

Figure 43. View personal information

2. View Information and Symptoms About a Patient

The doctor can select a patient from their assigned patient list (searched by ID).

For the selected patient, the application displays:

- Personal data
- All symptoms previously submitted by the patient

This information is retrieved from the server and stored in the doctor's patient list.

```

ID: 21, Name and surname: Jane Hopper
Enter the ID of the patient: 21

PATIENTS INFORMATION:
Name: Jane
Surname: Hopper
Email: patienttest@gmail.com
Date of Birth: 1987-08-11
Sex: FEMALE
Phone Number: 74398745
Symptoms for patient Jane Hopper:
Symptoms[date_hour=2025-11-29T14:00:56.835953300, description='I had chest pain during exercise'].

```

Figure 44. View a patient's symptoms

3. View Recordings

The doctor can view physiological recordings sent by the patient.

These include:

- ECG measurements
- EDA measurements

The application retrieves the selected measurement from the server and displays the processed ECG or EDA signal as a line chart.

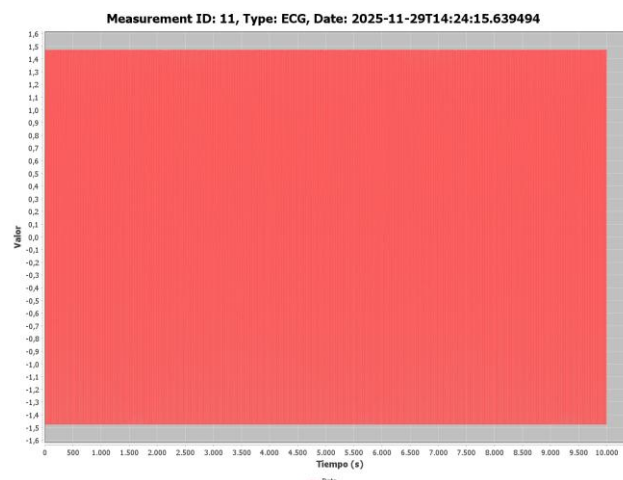


Figure 45. Measurement's signal

4. View Appointments

Displays the list of medical appointments associated with the patient, including:

```
ID: 21, Name and surname: Jane Hopper
Enter the ID of the patient: 21
=== Appointments for patient 21 ===
Date/Time: 2026-03-12T12:30
Patient ID: 21
Message: Monthly check-up
```

Figure 46. View appointments from a specific patient

- Appointment date and time
- Appointment message
- Patient and doctor ID

The information is requested from the server.

5. View Messages from Patient

Shows all text messages previously received from the selected patient.

```
=== Conversation with Jane Hopper ===
[2025-11-29T14:01:42.130210100] PATIENT: Hello
[2025-11-29T14:37:07.991962800] DOCTOR: Hello there, Jane
=== End of conversation ===
```

Figure 47. View messages

6. Send Message to Patient

Allows the doctor to write and send a new text message to the patient through the server.

```
ID: 21, Name and surname: Jane Hopper
Enter the ID of the patient: 21
Write your message: Hello there, Jane
```

Figure 48. Send a message to the patient

7. Log Out

Ends the doctor's session and closes the server connection, returning to the initial menu.

6.3. Administrator/Server

Purpose of the server

The administrator can access to the server, controlling the flow of protocols and the database, able to:

- Change the database, although it requires a Database browser.
- View the protocol messages in the console while the clients are interacting.
- Shutdown directly the server through a command and prefix password.

```
[Server] [OUT] Replied : {"type":"RESPONSE","action":"LIST_APPOINTMENTS","requestId":"list_mes
[Server] [IN] Received: {"type":"REQUEST","action":"LIST_MESSAGES","role":"DOCTOR","requestId":"list_mes
[Server] [OUT] Replied : {"type":"RESPONSE","action":"LIST_MESSAGES","requestId":"list_mes
[Server] [IN] Received: {"type":"REQUEST","action":"LIST_MEASUREMENTS","role":"DOCTOR","requestId":"list_mes
[Server] [OUT] Replied : {"type":"RESPONSE","action":"LIST_MEASUREMENTS","requestId":"list_mes
[Server] [IN] Received: {"type":"REQUEST","action":"GET_MEASUREMENT_VALUES","role":"DOCTOR","requestId":"list_mes
[Server] [OUT] Replied : {"type":"RESPONSE","action":"GET_MEASUREMENT_VALUES","requestId":"list_mes
```

Figure 49. Protocol messages flow in console

```
[Server] Connection closed for / [REDACTED]
shutdown
Enter admin password to confirm shutdown: [REDACTED]
[Admin] Shutting down server...
[Server] Shutting down thread pool and DB connection...
[DB] Connection closed.
-----
[Server] Server stopped successfully.
```

Figure 50. Server shutdown through command and password

How to make it work?

- 1- Initialize the server by *run* in IntelliJ. Can be also done by running the jar file.
- 2- The server will function automatically, meaning it doesn't require any action from the administrator to accept new clients.
- 3- The administrator can observe through the console the protocol messages, viewing who is connecting and if the client has closed the connection.
- 4- To shutdown the server must write "shutdown" command in the console and password (it is stabilised as "admin").