

POLITECNICO DI TORINO

Ingegneria Biomedica



Tesi di Laurea Magistrale

**Progettazione di un sistema distribuito
di archiviazione e distribuzione di dati
polisonnografici**

Relatore

Prof.ssa Gabriella OLMO

Laureando

Samuele TOZZI

Correlatori

Dott. Paolo ASTENGO

Ing. Irene RECHICHI

Ing. Camilla CERVETTO

Marzo 2021

Abstract

I sensori indossabili stanno diventando sempre meno invasivi, sempre più trasparenti, più comodi e di conseguenza più diffusi. L'utilizzo massivo di queste tecnologie permette di avere una grande mole di dati da utilizzare per l'addestramento di algoritmi di classificazione tali da fornire supporto ai clinici in fase di monitoraggio, diagnosi e riabilitazione. In particolare, più i sensori diventano "trasparenti", più è possibile usarli in applicazioni complesse, come il monitoraggio del sonno. In questo modo la polisonnografia (PSG) resta ancora il *gold standard* ma diventa pensabile sostituire parzialmente la PSG con esami effettuati in un contesto domiciliare, allestendo una opportuna BAN (Body Area Network). Tutti questi grandi quantità e varietà di informazioni necessita di essere archiviata e standardizzata, al fine di poterla analizzare in modo coerente e automatizzato. La presente tesi si pone l'obiettivo di realizzare un prototipo che prenda i dati in input tramite un portale web, in modo che sia estremamente accessibile. Una volta caricati, i dati vengono standardizzati e salvati in un database. Infine è possibile effettuare il download degli stessi, utilizzando lo standard FHIR. Quest'ultimo è personalizzabile in base alle esigenze dell'utente tramite l'impostazione di una serie di filtri dall'interfaccia web. Infine viene garantito un livello di sicurezza base e, soprattutto, il sistema è progettato per essere interoperabile e distribuito, in modo che possa subire aggiornamenti con sforzi modesti.

Indice

Acronimi	v
Introduzione	1
1 Gli studi sul sonno	3
1.1 I Segnali Elettrici	5
1.1.1 Elettroencefalogramma	6
1.1.2 Elettrooculogramma	7
1.1.3 Elettrocardiogramma	8
1.1.4 Elettromiografia	10
1.1.5 Impedenza transtoracica	11
1.2 I Segnali Meccanici	12
1.2.1 Saturazione dell'emoglobina	13
1.2.2 Estensimetro	13
1.2.3 Accelerometro	14
1.2.4 Giroscopio	14
1.2.5 Magnetometro	15
1.2.6 Termistori	15
1.3 Le fasi del sonno	16
2 Gli strumenti ICT	19
2.1 I linguaggi usati	20
2.1.1 Python	20
2.1.2 Javascript	21
2.1.3 HTML e CSS	21
2.2 HTTP	22
2.2.1 HTTPS	25
2.2.2 REST	26
2.2.3 Flask	27
2.3 UML	27
2.3.1 Use Case Diagram	29

2.3.2	Activity Diagram	30
2.3.3	Sequence Diagram	32
2.3.4	Class Diagram	32
2.3.5	Deployment Diagram	33
2.4	Nomenclatura di dati biomedici	36
2.4.1	SNOMED	36
2.5	I database	37
2.5.1	Modello concettuale	38
2.5.2	Modello logico	41
2.5.3	Algebra relazionale	43
2.5.4	SQL	44
2.6	HL7	45
2.6.1	Versione 2.x	45
2.6.2	Versione 3.0 e CDA	47
2.6.3	FHIR	47
2.7	Middleware	51
2.7.1	Mirth	52
3	SleepingRepo	56
3.1	Il contesto	57
3.1.1	Synopsis	57
3.1.2	Workflow	59
3.1.3	Use Case	61
3.2	Le comunicazioni	63
3.2.1	Deployment diagram	64
3.2.2	Sequence diagram	69
3.3	Il database	71
3.3.1	Il diagramma E-R	71
3.3.2	Verso il modello logico	74
3.4	Le classi	80
3.4.1	Le risorse FHIR	81
3.4.2	Le classi del server	87
3.5	Il prototipo	96
3.5.1	Il primo impatto: la registrazione	97
3.5.2	Upload dei dati	103
3.5.3	Download dei dati	109
	Conclusioni e sviluppi futuri	113
	Bibliografia	115
	Ringraziamenti	119

Acronimi

A

API Application Programming Interface

Ag Argento

AgCl Argento Clorurato

B

BAN Body Area Network

C

CDA Clinical Document Architecture

CHF Chronic Heart Failure

CNN Convolutional Neural Network

CSS Cascading Style Sheets

D

DAC Digital Analog Converter

DBMS Database Management System

DDoS Distributed Denial of Service

DDS Direct Digital Synthesis

E

E-R diagramma Entità-Relazione

ECG Elettrocardiogramma

EDF European Data Format

EEG Elettroencefalogramma

EMG Elettromiografia

EOG Elettrooculogramma

F

FHIR Fast Healthcare Interoperability Resources

G

GUI Graphic User Interface

H

HIS Hospital Information System

HL7 Health Level Seven

HTML HyperText Markup

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secured

I

ICD International Classification of Diseases

ICT Information and Communication Technology

IoT Internet of Things

IP Internet Protocol address

J

JSON JavaScript Object Notation

L

LA Left Arm

LAN Local Area Network

LL Left Leg

LOC Left Outer Canthus

LOINC Logical Observation Identifiers Names and Codes

N

NREM Non Rapid Eye Movement

NTC Negative Temperature Coefficient

P

PSG Polisonnografia

PTC Positive Temperature Coefficient

R

RA Right Arm

RAM Random Access Memory

REM Rapid Eye Movement

REST REpresentational State Transfer

RLS Restless Leg Syndrome

ROC Right Outer Canthus

RTT Round Trip Time

S

SNOMED Systematized Nomenclature of Human and Veterinary Medicine

SPO2 Saturazione Periferica dell'Ossigeno

SQL Structured Query Language

SSL Secure Socket Layer

T

TCP Transmission Control Protocol

TLS Transport Layer Security

U

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

V

VPN Virtual Private Network

W

W3C World Wide Web Consortium

WSGI Web Server Gateway Interface

X

XML eXtensible Markup Language

Introduzione

Negli ultimi anni si è assistito ad una forte espansione delle tecnologie informatiche e delle telecomunicazioni, denominate ICT¹: queste sono diventate sempre più accessibili economicamente, miniaturizzate e quindi invisibili agli occhi degli utenti, riuscendo dunque a penetrare in ogni ambito della società. Questa enorme diffusione di tecnologia è arrivata anche nell'ambito della salute, permettendo la progettazione di sensori indossabili in grado di acquisire moltissime tipologie di parametri fisiologici diversi. Grazie alla possibilità di trasmettere rapidamente le informazioni a distanza è possibile pensare a sistemi di telemedicina, ossia sistemi per la cura e il monitoraggio dei pazienti a distanza. Questo non significa che il ruolo del personale sanitario vanga meno, anzi tutt'altro: i sanitari diventano molto più efficienti, diventando quindi più importanti nel processo di monitoraggio e cura del paziente. Quest'ultimo diventa il centro del processo di cura. Il monitoraggio diventa indossabile e continuo e il baricentro dell'assistenza si sposta dall'ospedale, alleviando il carico sulla struttura ospedaliera per ricoveri e accessi inutili. Questa enorme diffusione di dispositivi meno invasivi ha prodotto una gran quantità di dati, che, potendo trasportarli molto facilmente, agevolano l'allenamento di algoritmi di machine learning, migliorando ulteriormente la qualità del supporto decisionale all'operatore sanitario.

Attualmente esistono banche dati sul sonno, ma non esistono piattaforme di questo genere. Quello che già esistono sono semplicemente piattaforme dove scaricare dati, senza interfacce e senza l'utilizzo di formati standard: non è possibile, quindi, fare quello che attualmente è molto diffuso in bioinformatica²[1]. In realtà, esistono delle piattaforme anche per dati medici, come il lavoro di Freund et al[2]. In questo contesto si muove il presente progetto di tesi che si pone l'obiettivo di realizzare un *repository*, ossia un deposito di dati sul sonno, cercando di armonizzare l'eterogeneità dei dati, uno dei principali problemi attuali. Proprio per comprendere

¹ICT sta per Information and Communication Technology. Sono le tecnologie dell'informazione e comprendono le tecniche per la trasmissione, ricezione ed elaborazione dell'informazione

²La bioinformatica è la disciplina che usa i metodi informatici per risolvere problemi biologici, in particolare i problemi di biologia molecolare.

a pieno questo fatto si è deciso di iniziare la trattazione con un capitolo sugli studi sul sonno, cercando di ricostruirne la fisiologia a partire dalla conoscenza dei dati raccolti e dei sensori utilizzati (Capitolo 1). Le informazioni raccolte sono state divise in base alla loro natura: meccanica ed elettrica. In generale, questo è importante per due motivi:

- comprendere l'origine del segnale e quindi della misura, per avere chiaro quale fenomeno fisiologico l'ha generata;
- comprendere la struttura del sistema di acquisizione, in particolare nel caso di segnali elettrici, poiché la struttura base per prelevare un segnale biomedico è sempre la stessa.

Una volta capito come si ottengono queste informazioni e che significato hanno, la trattazione prosegue con lo studio degli strumenti ICT che sono serviti per la buona riuscita del progetto (Capitolo 2). Tramite questo capitolo si avrà quindi una comprensione più tecnica di ciò che verrà poi trattato nel Capitolo 3, ossia il Capitolo finale. In quest'ultima parte verrà presentato il progetto, con le fasi dello sviluppo del software passo dopo passo, a partire dall'analisi dei requisiti, fino al codice vero e proprio. Saranno presentati dei diagrammi presi da UML per documentare il processo di sviluppo, in modo da far comprendere in modo più immediato che cosa sia stato fatto, senza però perdere dettaglio e rigore. Verranno presentate le interfacce con cui l'utente può interagire: quest'ultime sono state costruite in modo che siano accessibili da smartphone, tablet e computer. La trattazione include anche piccole parti di codice, soprattutto nei punti più cruciali dello sviluppo.

Capitolo 1

Gli studi sul sonno

Il sonno è una attività metabolica affascinante, con ancora molti segreti, che ricarica l'energia del corpo e migliora le difese immunitarie[3]. I disturbi del sonno possono causare problemi di vario genere nella quotidianità, fino a portare alla morte[4]. Lo strumento principe per studiare l'andamento del sonno è la polisonnografia (PSG). Un polisonnigrafo è uno strumento in grado di registrare diversi tipi di segnali:

- Elettroencefalogramma (EEG);
- Elettrocardiogramma (ECG);
- Elettrooculogramma (EOG);
- Elettromiografia (EMG);

A complemento di questi segnali spesso vengono aggiunti un saturimetro, dei dispositivi per la misurazioni dei movimenti addominali e uno per la valutazione del flusso oro-nasale. I segnali indicati sono segnali elettrici, che misurano l'attività elettrica dell'organismo, mentre per gli altri segnali si misurano grandezze meccaniche. Tutti i metodi di misura verranno approfonditi in seguito. In generale, il segnale principe quando si studia il sonno è l'EEG in quanto, dalle frequenze che si possono vedere studiando il segnae, e si ha l'intelaiatura principale delle fasi del sonno che il paziente sta attraversando. Queste, che verranno approfondite nell'apposita sezione, sono sostanzialmente 3: veglia, NREM e REM. Le ultime due fasi descrivono il sonno vero e proprio.

Si è detto che la polisonnografia è lo strumento principale per studiare e analizzare la qualità e la quantità del sonno di una persona; di fatto però non è l'unico metodo. Esistono, infatti, degli studi che mettono a confronto dei sistemi basati su sensori accelerometrici (approfonditi nell'apposito paragrafo) che si trovano comunemente in uno smartphone, con gli studi su una classica polisonnografia. Un sistema di questo tipo, utilizzando uno smartphone posto sullo sterno dei pazienti e usando

algoritmi di signal processing opportuni, ha consentito di individuare il disturbo dell'apnea notturna con una sensibilità del 90%[5]. Sensori inerziali di questo tipo possono essere integrati anche come sensori indossabili: interessante è la versione wearable come top sportivo prototipata presso Astel Electronic Engineering di Ivrea per il progetto della Regione Piemonte "ReHome". In un singolo top sportivo sono stati inseriti tre elettrodi per registrare ECG, impedenza transtoracica e movimenti baricentrici tramite un sensore inerziale contenente accelerometro, giroscopio e magnetometro. Il tutto utilizzando cavi e elettrodi in tessuto.



Figura 1.1: Disegno sul top prima di cucire le componenti. Si notino le posizioni degli elettrodi ai lati, mentre il microcontrollore con il sensore inerziale si posiziona in una tasca appositamente cucita davanti lo sterno

I sensori diventano via via sempre più accettabili per gli utilizzatori finali, si pensi che in alcuni casi vengono utilizzate smartband con cardiofrequenzimetro e accelerometro. Senza alcun dubbio questa grande varietà di sensori e segnali offre una tale quantità di dati da permettere di addestrare algoritmi di machine learning¹. Per esempio, nello studio condotto da Zang et al, sono stati classificati gli stadi del sonno a partire da un singolo canale EEG usando un metodo basato sulle convolutional neural networks (CNN)²[4].

¹ Letteralmente: apprendimento della macchina. Il machine learning è una branca dell'informatica che permette di "insegnare" ad un algoritmo a riconoscere particolari situazioni, classificandole

² Il CNN è un algoritmo basato sull'addestramento di reti neurali artificiali ispirato alla corteccia visiva animale

Infine bisogna considerare l'importanza nel trattamento di patologie neurodegenerative quali la malattia di Parkinson. Uno studio condotto da Navin Cooray et al dimostra come il disturbo della fase REM (RBD) sia un sintomo precoce del Parkinson. Un altro esempio riguardante il disturbo della fase REM è il lavoro di Matteo Cesari et al, in cui i ricercatori hanno messo a punto un sistema per predire il disturbo della fase REM (tipico del Parkinson) fin dallo stato prodromico.

1.1 I Segnali Elettrici

In questa sezione verranno brevemente presentati i segnali elettrici che possono essere utilizzati durante lo studio del sonno. La loro particolarità è quella di intercettare e misurare le correnti sulla superficie della pelle generate dall'attività elettrica del sistema nervoso nei vari organi, ad eccezione dell'impedenza transtoracica che, come si vedrà, prende in considerazione una misura di volume. È stata comunque inserita tra i segnali elettrici in quanto il dato viene individuato attraverso la misura di una grandezza di tipo elettrico, di conseguenza sono comunque valide le suddette considerazioni generali.

Una delle caratteristiche che accomuna questi segnali è la struttura base del sistema di acquisizione. Il front end può avere delle variazioni sui valori scelti dei componenti per seguire le specificità del segnale, mentre gli elettrodi sono tendenzialmente utilizzabili in ogni situazione. I materiali da cui sono costituiti generalmente sono argento (Ag) e argento clorurato (AgCl). Nel settore di interesse, ossia lo studio del sonno, sono sempre più diffusi elettrodi in tessuto, integrabili in soluzioni wearable³. In teoria, gli elettrodi si dividono in polarizzabili e non polarizzabili a seconda del loro comportamento: se si comportano come resistori sono non polarizzabili, se invece gli effetti capacitivi sono preponderanti, sono polarizzabili. Un elettrodo non polarizzabile è preferibile, in quanto ha un potenziale di semicella molto ridotto. Questo si traduce in minori artefatti da movimento. Nella realtà non esistono elettrodi perfettamente non polarizzabili, quindi il modello che si può assumere è quello riportato in Figura 1.2.

³Per tecnologia wearable si intende un dispositivo elettronico indossabile, spesso integrato negli abiti o negli accessori

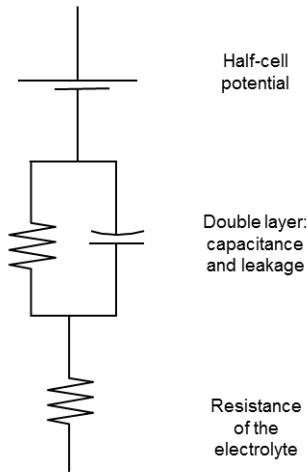


Figura 1.2: Modello elettrico di una interfaccia elettrodo/elettrolite, riprodotta con permesso da[6]

1.1.1 Elettroencefalogramma

L'elettroencefalogramma, abbreviato con EEG, è la misura dell'attività cooperativa spontanea della struttura neuronale della corteccia cerebrale. Il tipo più utilizzato, anche negli studi sul sonno, è quello non invasivo che consiste nella registrazione del segnale post sinaptico. La registrazione avviene sullo scalpo del paziente secondo lo standard "sistema internazionale 10-20" (Figura 1.3).

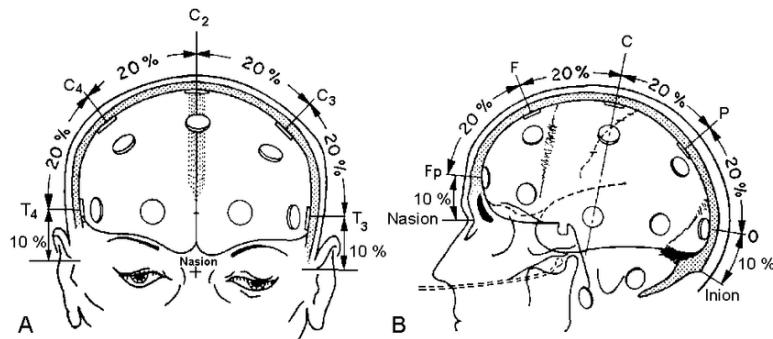


Figura 1.3: schema 10-20[7]

Vengono utilizzati fino a 20 elettrodi e una massa. I punti di riferimento sono: nasion, inion e i punti auricolari destro e sinistro. Vengono misurate le distanze tra questi così che gli elettrodi possano essere posizionati ad una distanza del 10% o del 20% della misura dello scalpo. Il potenziale registrato può essere differenziale o

monopolare. Con differenziale si intende o la differenza tra due elettrodi esploranti, mentre con monopolare si intende la differenza tra un elettrodo esplorante e il suo riferimento, considerato a potenziale nullo. Al fine di evitare interferenze, in particolare dovute a segnale ECG, le orecchie, il mastoide o il mento vengono usati come potenziali di riferimento.

I nomi dei segnali acquisiti derivano dalla posizione degli elettrodi che li generano. Questi ultimi sono nominati con una sigla che codifica la zona corticale esplorata e da un numero o un carattere che identifica l'emisfero: se è un numero dispari è sinistra, se pari è destra, se è il carattere z è la linea mediana.

L'EEG è usato per valutare l'attività cerebrale, ciò viene fatto attraverso la valutazione di diverse bande di frequenza:

- δ -> 0.3 Hz - 3.5 Hz
- θ -> 3.5 Hz - 7 Hz
- α -> 7 Hz - 14 Hz
- β_1 -> 14 Hz - 21 Hz
- β_2 -> 21 Hz - 40 Hz
- γ -> 40 Hz - 80 Hz

La spiegazione dei ritmi dell'EEG verrà approfondita successivamente, con particolare attenzione all'attivazione durante le fasi sonno/veglia.

1.1.2 Elettrooculogramma

Nell'occhio è possibile individuare un potenziale elettrico pressoché indipendente da qualsivoglia forma di energia luminosa. Questo potenziale è chiamato elettrooculogramma, abbreviato EOG. Si può schematizzare come un dipolo con polo positivo nella cornea e polo negativo nella retina. Quando gli occhi si muovono si ha un cambiamento di questo potenziale che permette quindi di individuare la presenza di movimento anche ad occhi chiusi, e il relativo verso. Questi elettrodi vengono posizionati 1 cm sopra il dotto lacrimale esterno dell'occhio destro (ROC) e 1 cm sotto il dotto lacrimale dell'occhio sinistro (LOC).

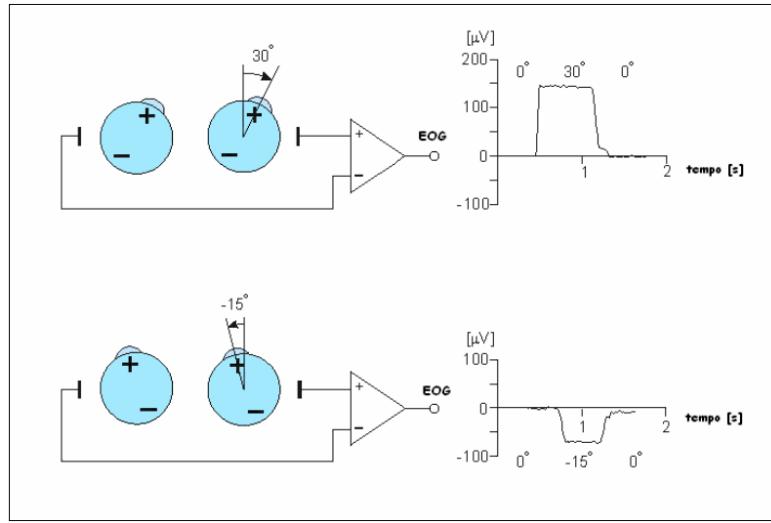


Figura 1.4: Schematizzazione della genesi del potenziale elettrooculografico indotto da rotazioni orizzontali oculari[8].

1.1.3 Elettrocardiogramma

L'elettrocardiogramma è un segnale preso dalla superficie della pelle con degli elettrodi e permette di ricostruire l'attività cardiaca. Il segnale è composto da cinque onde: P, Q, R, S e T.

Ad ogni parte del segnale corrisponde una fase del ciclo cardiovascolare. È possibile

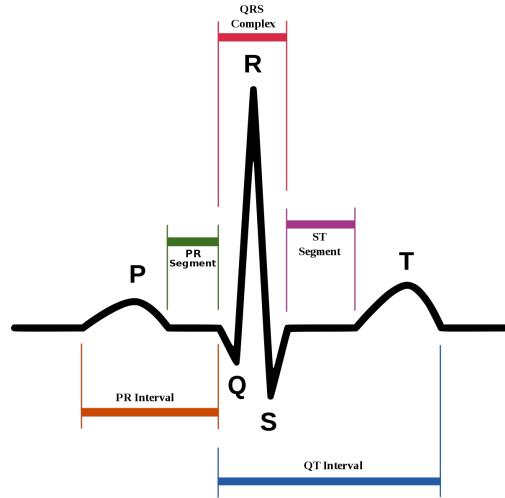


Figura 1.5: Tracciato ECG di un soggetto sano

individuare tre parti di segnale particolarmente importanti: l'onda P, il complesso

QRS e l'onda T. Durante gli studi sul sonno assume una grande rilevanza il picco R, in quanto consente di calcolare la frequenza cardiaca rapidamente.

La misura del segnale ECG può avvenire con un numero di elettrodi che varia da 3 a 10, portando i canali di acquisizione, detti "derivazioni" da 3 a 12. Le prime tre derivazioni sono dette "il triangolo di Einthoven".

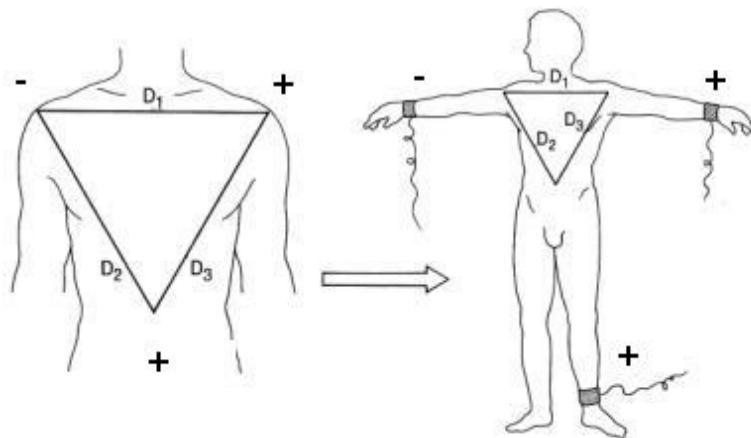


Figura 1.6: Triangolo di Einthoven

Se si aggiunge un elettrodo di riferimento al centro, anche detto "nodo di Wilson" si hanno altre tre derivazioni, dette "derivazioni aumentate" o "derivazioni di Goldberg". Infine, ponendo degli elettrodi sul torace è possibile aggiungere altre

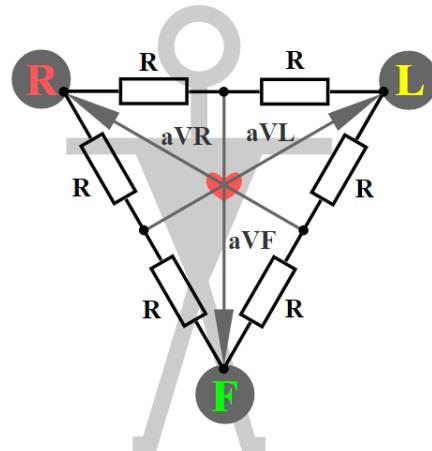


Figura 1.7: Derivazioni aumentate

sei derivazioni, dette "derivazioni precordiali". Lo studio dell'ECG durante il sonno

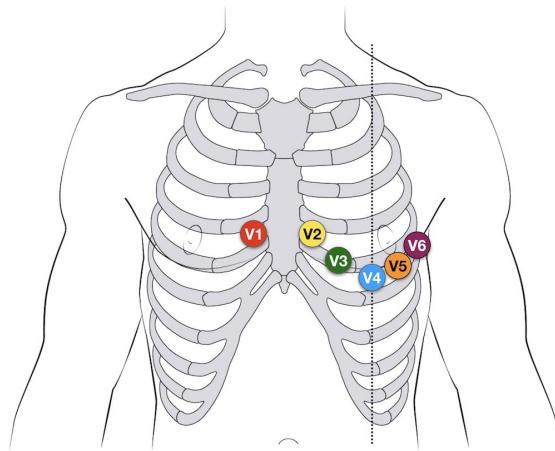


Figura 1.8: Derivazioni precordiali

è importante sia, come già detto, per rilevare la frequenza cardiaca, ma anche per valutare la fase del sonno in combinazione con altri segnali e valutare patologie cardiache.

1.1.4 Elettromiografia

L'elettromiografia è il segnale che viene acquisito durante contrazione muscolare indotta dal potenziale di azione lungo le fibre muscolari. Il principale uso di questa applicazione, secondo la *Food and Drug Administration*, è quello di intercettare la contrazione muscolare, dal momento che l'EMG non è un segnale deterministico, ycome l'ECG. L'EMG, infatti, ha una struttura complessa per cui lo studio della sua morfologia offre scarse informazioni. Negli studi sul sonno l'EMG è utile al fine di valutare la tensione muscolare e gli stadi del sonno, dal momento che il movimento è ridotto.

Normalmente due elettrodi sono posizionati sul mento: uno sopra e l'altro sotto la mascella. Altri due elettrodi sono posizionati sul muscolo tibiale anteriore, per controllare i movimenti della gamba. Gli elettrodi sotto il mento consentono di valutare eventuali casi di bruxismo⁴, mentre sulla gambe servono a identificare, in alternativa agli estensimetri la *Restless leg syndrome*⁵.

⁴Il bruxismo è un disturbo del sonno che consiste nel dignignare i denti al punto da farli sfregare tra di loro.

⁵Restless leg syndrome (RLS) è un disturbo caratterizzato da una fastidiosa sensazione alla gamba al momento di dormire che interferisce con l'inizio del sonno e lo rovina. Non si sa molto sulla frequenza con cui capita, ma spesso resta non diagnosticata.

1.1.5 Impedenza transtoracica

Come si è accennato in precedenza l’impedenza transtoracica è l’unica misura elettrica effettuata a non trattare una attività elettrica del corpo umano. Essa quantifica, infatti, la frequenza respiratoria. Nonostante nella pratica clinica questo parametro sia sottovalutato, risulta essere un parametro molto importante: infatti un ritmo molto variabile o un’evidente fatica nel compiere l’atto respiratorio possono essere sintomi di instabilità fisiologiche. Pazienti che mostrano questi sintomi possono avere problematiche a livello cardiaco come coloro che soffrono di CHF⁶. Per ottenere la frequenza respiratoria si utilizza la misurazione dell’impedenza transtoracica, ossia si va a valutare l’opposizione che incontra la corrente attraverso il torace. Per comprendere come avviene l’acquisizione bisogna considerare il funzionamento della meccanica respiratoria e il comportamento elettrico dell’aria. Quando si inspira, infatti, il diaframma si contrae, aumentando il volume della cassa toracica. Quando aumenta il volume la pressione interna diminuisce creando così un gradiente di pressione che permette all’aria di entrare dall’esterno. L’aria, inoltre, è un isolante elettrico, per cui se si misura l’impedenza di un torace appena prima di espirare e appena dopo aver espirato si noterà un comportamento capacitivo diverso. L’andamento di questo comportamento genera un segnale che permette di calcolare la frequenza respiratoria. La misurazione dell’impedenza può essere effettuata

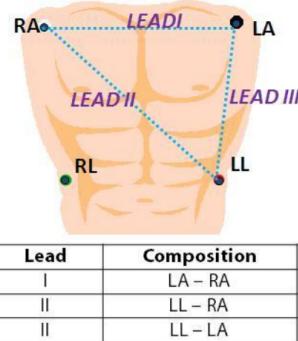


Figura 1.9: Derivazioni per il calcolo dell’impedenza[9].

con la stessa coppia di elettrodi (misura con due cavi) o con differenti coppie di elettrodi (misura con quattro cavi). Per effettuare la misura migliore bisogna tenere conto della posizione e dello stato del paziente, per esempio se il paziente è agitato respirerà tendenzialmente con il torace, quindi la prima derivazione ("Lead I" in

⁶CHF è un acronimo per "chronic heart failure", letteralmente "scompenso cronico cardiaco". È un disturbo che si verifica quando il cuore non riesce a pompare abbastanza sangue come dovrebbe.

figura 1.9) risulta più appropriata, mentre se un paziente sta dormendo ci sarà una maggiore variazione di volume nell'area addominale quindi le derivazioni due e tre saranno più adeguate. Si è detto, dunque che questa misura non è direttamente

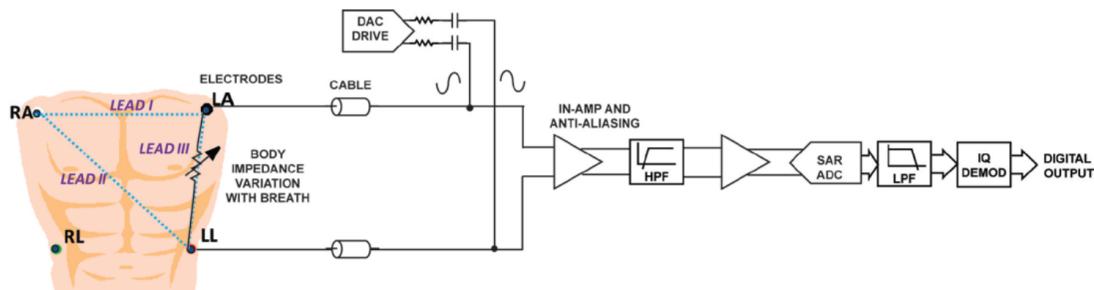


Figura 1.10: Tipica catena di acquisizione per il segnale di impedenza toracica[9].

correlata all'attività elettrica del corpo umano, ma resta tale proprio in virtù della grandezza che viene direttamente misurata. La differenza principale con i precedenti segnali, quindi, è che deve essere iniettata corrente nel corpo affinché si possa effettuare la misurazione, utilizzando, ad esempio, un circuito come quello in figura 1.10. Il circuito è composto da due parti, una per pilotare e una per misurare. Il pilotaggio può essere fatto da un DDS⁷ o da un DAC⁸. In entrambi i casi viene iniettata una corrente alternata ad alta frequenza negli elettrodi, poi viene effettuata la misura dell'impedenza.

1.2 I Segnali Meccanici

I segnali meccanici sono il risultato di un insieme di misurazioni che hanno lo scopo di acquisire grandezze di tipo meccanico. In ambito biomedico queste grandezze possono essere, per esempio, pressioni (pressione sanguigna), forze, misure invasive o non sul moto di un fluido, oppure possono essere misure inerziali, in relazione a diversi segmenti corporei. Negli studi sul sonno, in particolare, si usano molto i sensori inerziali, per monitorare il movimento del paziente durante la notte, ma si eseguono pure misure indirette su fluidi (aria nei polmoni e sangue) per capire il buon funzionamento o meno del sistema respiratorio.

⁷DDS sta per Direct Digital Synthesis, ed è un modo per generare digitalmente una singola onda periodica a partire da un singolo oscillatore di riferimento.

⁸DAC sta per digital analog converter, per convertire segnali digitali in analogici

1.2.1 Saturazione dell'emoglobina

Uno dei segnali meccanici più importanti è senza dubbio il livello di saturazione dell'emoglobina, che permette di valutare l'ossigenazione dei tessuti e quindi la salute del sistema cardiocircolatorio. Negli studi sul sonno è importante per individuare eventuali apnee notturne. Il dispositivo che si occupa di questa misurazione è detto pulsossimetro. Questo strumento effettua una misura ottica andando a irradiare il tessuto con due fasci di luce a diversa lunghezza d'onda, poi compara il differente assorbimento da parte del sangue, secondo la legge di Lambert-Beer:

$$I_{out} = I_{in} e^A$$

Dove I_{out} è l'intensità luminosa in uscita e I_{in} è quella in ingresso. A , invece, è il fattore di assorbimento. Secondo Yousuf Jawahar[10] La pulsossimetria può essere in due modi:

- **Metodo della trasmittanza.** La luce è trasmessa attraverso il tessuto con dei led ed è ricevuta dall'altra parte usando un foto-rilevatore.

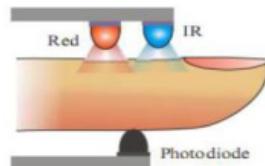


Figura 1.11: Metodo della trasmittanza

- **Metodo della reflettanza.** Led e foto-rilevatore sono dallo stesso lato quindi viene individuata la quantità di luce riflessa.



Figura 1.12: Metodo della reflettanza

1.2.2 Estensimetro

Gli estensimetri sono trasduttori che convertono estensioni in tensione. Questo avviene perché un corpo sottoposto a trazione subisce un allungamento in direzione

assiale e una restrizione in direzione trasversale. Inoltre, se si considerano materiali conduttori è noto che:

$$R = \rho \frac{L}{A}$$

Dove R è la resistenza, ρ è la resistività, L è la lunghezza e A è la sezione. Se si utilizzano dei conduttori con una dimensione preponderante, in fase di allungamento la diminuzione della sezione trasversale può essere considerata trascurabile rispetto alla lunghezza, quindi ci sarà una variazione di resistenza. In questo modo è possibile misurare l'allungamento andando a guardare quanto varia la resistenza. L'uso degli estensimetri è importante per il monitoraggio del *Restless leg syndrome* insieme a magnetometri e giroscopi e in alternativa all'EMG sulla gamba.

1.2.3 Accelerometro

Gli accelerometri sono dispositivi che misurano le vibrazioni o l'accelerazione del corpo su cui sono montati. I più diffusi sono quelli che sfruttano l'effetto piezoelettrico⁹.

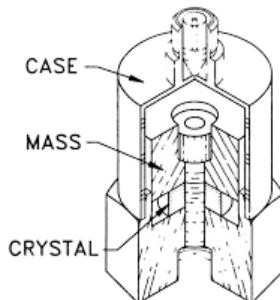


Figura 1.13: Accelerometro piezoelettrico base

1.2.4 Giroscopio

I giroscopi sono sensori che misurano la velocità angolare e quindi, dopo una integrazione nel tempo, gli angoli. Il loro principio di funzionamento si basa su una massa libera di vibrare all'interno del sensore con una velocità nota. Quando il sensore ruota la forza di Coriolis induce una vibrazione secondaria ortogonale al senso di vibrazione originale. Misurando la deformazione dovuta alla vibrazione

⁹L'effetto piezoelettrico è un fenomeno che si riscontra in alcuni materiali tra cui il quarzo, che se sollecitati lungo un asse detto *asse elettrico* si creano delle cariche sulle due facce proporzionali alla forza applicata

secondaria si può determinare la velocità angolare. Vengono usati insieme ad accelerometri e magnetometri.

1.2.5 Magnetometro

I magnetometri misurano il campo magnetico attraverso l'effetto Hall¹⁰, che mette in movimento le cariche all'interno del conduttore. Se sottoposte ad un campo magnetico e ad uno elettrico, esse si sposteranno in una direzione perpendicolare ad entrambi i campi. In base a dove si accumulano le cariche, noto il campo elettrico imposto, è possibile conoscere la direzione del campo magnetico terrestre e, di conseguenza, l'orientazione del sensore rispetto ad esso, in modo da sapere com'è posizionato il sensore nello spazio.

1.2.6 Termistori

Sono termometri a semiconduttore, per cui la resistenza è funzione della temperatura. Sono di due tipi:

- NTC (Negative Temperature Coefficient): la resistenza è inversamente proporzionale alla temperatura.
- PTC (Positive Temperature Coefficient): la resistenza è direttamente proporzionale alla temperatura.

Essendo basati su semiconduttori il loro comportamento è fortemente non lineare. Per esempio un termistore ntc ha una relazione di questo tipo:

$$R = R_0 e^{(\beta \frac{1}{T} - \frac{1}{T_0})}$$

Dove R_0 è la resistenza a T_0 , che di solito si assume essere 298.15 K, mentre β è un parametro del dispositivo. Siccome non sono dispositivi molto precisi possono essere usati per dare informazioni generali, in particolare negli studi sul sonno sono un buono strumento per misurare il ritmo del flusso nasale: quando c'è un aumento di temperatura, il paziente ha espirato.

¹⁰L'effetto Hall è un fenomeno fisico che prevede la generazione di una differenza di potenziale in senso trasversale in un conduttore quando è attraversato da una corrente in senso longitudinale ed è sottoposto ad un campo magnetico

1.3 Le fasi del sonno

Il sonno è uno stato di semi incoscienza complementare allo stato di veglia. Insieme le due componenti formano il ciclo *circadiano*¹¹ e il sonno occupa circa 8 ore, infatti un essere umano trascorre circa un terzo della sua vita a dormire. Questa periodicità si pensa sia mantenuta da alcuni "orologi biologici interni" [11]. Questi ultimi, sono comunque influenzati da fattori esterni come, ad esempio, l'alternanza giorno-notte. Queste influenze sono dette in tedesco *zeitgebers*¹². In ogni caso questi orologi sono in grado di distinguere i vari periodi anche in assenza di marca-tempo, ma con un periodo di circa 25 ore [12]. Questo fenomeno è detto *free-running*.

Il sonno, in realtà, segue un altro ciclo periodico, detto *ultradiano*, in quanto dura meno di 24 ore. Questa periodicità è l'alternanza delle fasi del sonno.

Mentre si dorme si oscilla principalmente tra due fasi principali:

- **NREM.** La parola sta per "Non Rapid Eye Movement". Questa fase è caratterizzata da una caduta di tono muscolare, in particolare nei muscoli antigravitari, da una assenza di movimenti rapidi degli occhi e da un EEG definito "sincronizzato", ossia a frequenze più basse. Lo stadio NREM è a sua volta suddiviso in tre sottostadi che portano via via ad un sonno sempre più profondo:
 - **N1** è lo stadio più leggero. Spesso viene trascurato in quanto occupa meno del 5% del tempo di sonno totale. È considerato uno stato di transizione tra la veglia e lo stato N2, infatti nell'EEG è caratterizzato da onde α (come in fase di veglia a riposo) e da un accenno di onde θ , caratteristiche della fase successiva.
 - **N2.** Insieme a N1 forma quello che viene comunemente classificato come "sonno leggero" e rappresenta circa il 50% del sonno totale. Questo stadio è caratterizzato da un EEG con una fase θ predominante (la stessa banda che si riscontra in un adulto sveglio in fasi ipnotiche o in stati di tensione emotiva) e da una fase detta Theta-Sigma in cui compaiono treni di onde Sigma (12-14 Hz) di ampiezza 5-50 μ V, detti fusi del sonno per la loro forma. Un altro elemento elettroencefalografico caratteristico è senza dubbio il complesso K. Esso non è altro che un'onda bi-trifasica con tensione maggiore di 75 μ V che presenta una componente negativa, seguita immediatamente da una positiva. La durata è di almeno 0.5 s.
 - **N3.** In alcune classificazioni a N3 si fanno corrispondere due fasi distinte, mentre secondo la nuova classificazione dell'AASM (American Accademy

¹¹Circadiano deriva dalle parole latine *circa* e *diem* ossia "intorno al giorno".

¹²In tedesco significa marca-tempo.

of Sleep Medicine) si ha solo N3. In questo stadio l'ampiezza del tracciato EEG aumenta, in media $150 \mu\text{V}$, mentre la frequenza diminuisce arrivando in banda δ . Questa frequenza non è fisiologica nello stato di veglia in età adulta, ma è predominante nei bambini e compare, inoltre, nelle persone sotto anestesia e in alcune malattie cerebrali. Attraverso l'analisi di Fourier si può ottenere una misura che quantifica l'attività in banda δ , detta *Slow Wave Activity* che può essere utilizzata come indice di intensità del sonno NREM stesso.

- **REM.** Il nome è un acronimo per "Rapid Eye Movement", in quanto una delle caratteristiche è proprio il movimento rapido degli occhi. Quando si entra in fase REM l'EEG da sincronizzato diventa desincronizzato con onde a bassa ampiezza e frequenza mista. La banda principale è la banda θ , come in fase N2, ma non è l'unica. Risulta, infatti, presente anche la banda α e durante il sonno onirico compare anche la banda β , componente che in un individuo adulto sano compare durante la veglia e in stato di attività. Per questa particolarità il sonno REM viene anche chiamato "sonno paradosso". Si è parlato di sonno onirico poiché, durante la fase REM, si sogna, infatti se si sveglia qualcuno durante la fase REM è molto probabile che racconti di aver sognato. In realtà si ritiene che si sogni anche durante la fase NREM, cambia solamente la qualità, poiché in fase REM il sogno è "immagine-simile", mentre in NREM è "pensiero-simile[13][14].

Un'altra caratteristica della fase REM è l'atonia muscolare, che in questo caso è totale. Questo fenomeno si verifica in particolare a carico dei muscoli antigravitari. Si possono riscontrare piccole clonie muscolari, ossia contrazioni involontarie dei muscoli. Infine si assiste ad una paralisi muscolare, per cui quando si sogna non ci si può muovere.

Durante la fase REM si possono notare anche modificazioni vegetative. Ad esempio, l'attività respiratoria diventa irregolare con incrementi durante le clonie muscolari. La frequenza cardiaca è variabile durante ogni episodio REM poiché si alternano periodi di bradicardia e tachicardia. Vengono sospesi i meccanismi termoregolatori, come la sudorazione. La temperatura aumenta coerentemente con l'aumento dell'attività del cervello.

Quando ci si addormenta inizialmente si entra in una fase N1, che è infatti un momento di transizione caratterizzato da onde α per poi passare rapidamente a fasi di sonno più profondo (N2 o N3). Questo passaggio avviene in circa 30 minuti. Successivamente dopo circa 70 minuti si passa alla fase REM, la cui durata è variabile (mediamente 15 minuti), da cui poi si torna alla fase N1 e si ricomincia il ciclo. In una notte si spendono approssimativamente cinque cicli NREM-REM ma la differenza consiste nella durata delle singole fasi: nelle fasi iniziali del sonno, infatti, si tende ad avere una fase NREM più lunga. Quando si avvicina la mattina

aumenta il tempo speso in fase REM.

Ci sono delle variazioni della durata delle fasi del sonno in funzione dell'età. Nel primo anno di vita, ad esempio, il passaggio dalla fase di veglia al sonno profondo è effettuato tramite fase REM e il ciclo NREM-REM è di circa 50-60 minuti, contro i 90 degli adulti. Il sonno profondo è massimo nei bambini per poi decrescere con l'età. Fino alla metà dell'adolescenza si può anche saltare la prima fase REM, dopodiché si nota un considerevole calo del sonno profondo, fino al 40%, anche a parità di durata del sonno totale. Questa diminuzione continua con l'età, fino al punto che negli uomini oltre i 60 anni può non essere presente alcuna fase di sonno profondo. Il numero di risvegli notturni aumenta con l'età. Infine il tempo in fase REM mantiene una percentuale abbastanza stabile con l'età. In termini assoluti, invece, declina rapidamente in concomitanza delle disfunzioni organiche del cervello della senilità[15].

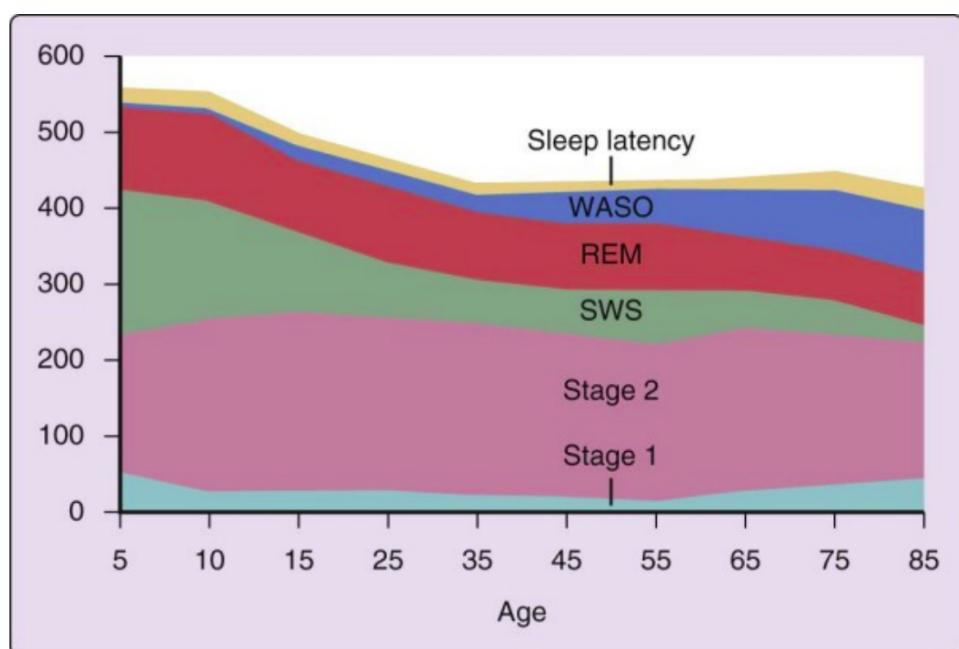


Figura 1.14: Andamento delle fasi nel sonno in funzione dell'età. In ascissa c'è il tempo, in ordinata il tempo in minuti. Per "WASO" si intende il tempo di veglia mentre per "SWS" si intende il sonno profondo[15]

Capitolo 2

Gli strumenti ICT

Il presente capitolo nasce con l'esigenza di dare una visione generale di alcuni strumenti che sono stati utilizzati al fine di realizzare il lavoro di tesi. Progettare un sistema informatico richiede un gran numero di strumenti differenti. Per questo motivo si tratteranno i linguaggi utilizzati, che siano essi di programmazione, come ad esempio Python, o di markup, come HTML. Ad esempio Javascript è di fatto lo standard per creare pagine web non statiche, mentre Python è estremamente più flessibile ed è eccellente anche stand-alone. Parlando di standard, un sistema che trasmette informazioni non può non utilizzare protocolli di comunicazione comunemente riconosciuti. Si è dunque optato per l'utilizzo di HTTP, di sicuro uno dei più diffusi al mondo e alla base di tutti i contenuti che si fruiscono su internet tramite browser. Omologare la comunicazione diventa un problema sempre più spinoso man mano che si entra all'interno del settore sanitario, dove l'interoperabilità dei dati e dei sistemi deve essere uno dei capisaldi al fine di riuscire a rendere disponibili le informazioni ovunque e in ogni momento: si è perciò deciso di adottare uno standard di ultima generazione, ossia FHIR.

Si tenga conto, però, che rendere standard i dati e far parlare diversi applicativi tra loro non sempre è possibile. In queste situazioni risulta importantissimo tenere presente che esistono i "middleware", ossia delle "collanti software" che aiutano moltissimo a raggiungere l'interoperabilità. Si sta parlando, in ogni caso, di dati e standard biomedici, per cui si è optato per un middleware che fosse specializzato nell'interoperabilità di sistemi informativi sanitari, ossia Mirth. Anche se non prettamente ICT è necessario codificare e standardizzare i nomi delle varie misurazioni mediche: seguendo la documentazione della regione Piemonte per il progetto "RE HOME" si è scelto per SNOMED.

Infine, ma non per importanza, si sono usati alcuni strumenti del linguaggio di modellazione UML per redigere una documentazione che fosse abbastanza esplicativa delle funzionalità del sistema, al fine di renderlo mantenibile e scalabile. Inoltre, aver formalizzato alcuni aspetti secondo UML, permette di comunicare e

presentare il lavoro in modo più chiaro e semplice. È stato necessario trattare i vari modelli implementati MySql, il database scelto, in particolare il logico e il concettuale. Infine, per concludere il discorso sui linguaggi è stato necessario trattare, seppur in forma abbastanza leggera, le varie operazioni possibili su un modello concettuale in particolare, ossia il relazionale, con l'algebra relazionale che viene implementata nel linguaggio di interrogazione e manipolazione SQL.

2.1 I linguaggi usati

Quando si pensa ad un software è importante pensare anche a come effettivamente questo codice venga scritto, per cui risulta molto importante conoscere i principali pro e contro di più linguaggi di programmazione in modo da avere un'idea di quale sia il più adatto alla situazione. In questa sezione verranno trattati due linguaggi di programmazione non compilati, ma interpretati, ossia Python e Javascript. I due linguaggi hanno funzionalità diverse. Se da un lato, Python è in grado di lavorare autonomamente, Javascript necessita almeno di un browser o, come si vedrà nella sezione sui middleware, di un ambiente come Mirth. Inoltre la principale funzione di Javascript è quella di rendere attive le interfacce, quindi è un linguaggio particolarmente indicato al *front-end* di una applicazione web. Nel front-end rientrano altri due linguaggi che verranno trattati, anche se non sono propriamente linguaggi di programmazione: HTML e CSS. Questi ultimi, come si vedrà in modo più approfondito, servono a generare le interfacce utente che il browser visualizza.

2.1.1 Python

Python è, ad oggi, uno dei linguaggi di programmazione più diffusi al mondo. Python è orientato a oggetti anche se in realtà supporta anche altri paradigmi, come la programmazione strutturata e alcune caratteristiche della programmazione funzionale, oltre a supportare la riflessione¹. Uno dei punti di forza di Python è senza alcun dubbio la sua sintassi pulita, per via dell'uso dell'indentazione al posto delle parentesi per delimitare i blocchi del programma, e vicina allo pseudocodice. Inoltre è un linguaggio interpretato. Questo comporta una maggiore immediatezza nello scrivere il codice. Spesso viene catalogato come linguaggio di scripting² anche se in realtà la grandissima quantità di librerie per le applicazioni più disparate

¹In informatica la riflessione è la capacità di programma di avere come oggetto di elaborazione sé stesso, arrivando a modificare pure il proprio codice sorgente.

²In informatica un linguaggio di scripting è un linguaggio interpretato che viene usato per automatizzare funzioni del sistema operativo o delle applicazioni, oppure nella programmazione web all'interno delle pagine

permettono a Python di scrivere un software molto modulare e quindi di gestire bene anche lo sviluppo di applicazioni complesse (si pensi che anche il famosissimo social network Instagram è scritto in Python).

Il linguaggio non è tipizzato, nonostante ci sia un forte controllo sui tipi in runtime: Python intende la variabile come il nome che si riferisce ad un oggetto. Di conseguenza con Python è possibile fare assegnamenti consecutivi alla stessa variabile, passando, per esempio, da un numero ad una stringa, perché il "tipo" della variabile non è altro che un attributo dell'oggetto. Inoltre il precedente oggetto, che a quanto punto non è più assegnato ad una variabile, viene pulito rapidamente in automatico dal garbage collector³.

Un'altra caratteristica distintiva è l'overloading⁴.

Il principale punto debole di Python, rispetto ai linguaggi con typing statico come C o Java, è la velocità di esecuzione. Il problema può essere aggirato utilizzando dei tool che permettono l'importazione di librerie C o C++.

2.1.2 Javascript

Javascript è uno dei linguaggi di programmazione più usati, insieme a Java e Python. Javascript è un linguaggio orientato debolmente a oggetti e ad eventi. Inoltre è debolmente tipizzato ed è interpretato.

Il nome "Javascript" non deve confondere con il linguaggio "Java": la somiglianza del nome è dovuta all'ispirazione nella sintassi che i creatori di Javascript hanno avuto. Infatti volevano che i programmatore di Java fossero agevolati nell'apprendimento del nuovo linguaggio.

Javascript è particolarmente usato nello sviluppo web lato client, infatti al momento tutti i browser supportano Javascript per rendere le pagine web interattive. Inoltre, essendo un linguaggio di scripting, può essere utilizzato all'interno di server per assolvere a determinati compiti importando librerie e classi dal programma che lo sta ospitando.

2.1.3 HTML e CSS

HTML, acronimo per *HyperText Markup Language* non è un vero linguaggio di programmazione, ma come dice il nome, un linguaggio di markup⁵. HTML è lo

³Letteralmente "raccoglitrice di spazzatura" il garbage collector è una parte importantissima della macchina astratta di Python, anche se quasi invisibile agli occhi del programmatore

⁴L'overloading in informatica è processo per cui si creano più funzioni con lo stesso nome ma con un diverso insieme di input (*signature*) e un diverso valore di ritorno in output

⁵Per linguaggio di markup si intende un sistema per mettere delle annotazioni in un documento in modo che siano distinguibili dal testo. Le annotazioni specificano la presentazione strutturale

standard per la rappresentazione di pagine web, rappresentando la struttura tramite l'uso di indicatori per la presentazione. Il browser legge il file e lo renderizza nella pagina che poi mostra all'utente. Nello specifico questi indicatori sono chiamati "tag" e sono rinchiusi tra delle parentesi angolate, come ad esempio `<html>`, anche chiamati elementi html.

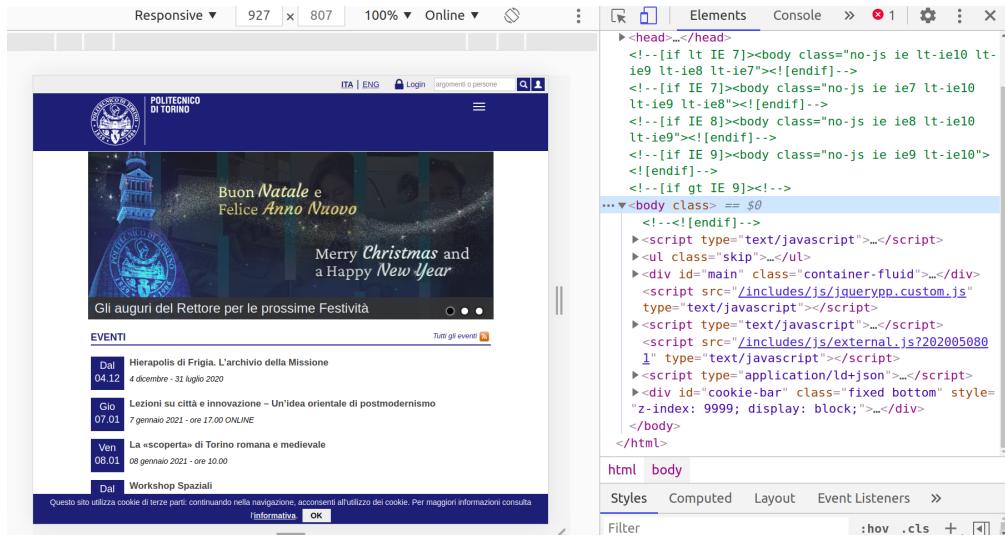


Figura 2.1: Esempio di pagina html

Il CSS, acronimo per Cascading Style Sheets, invece è un linguaggio il cui scopo è separare la formattazione dei documenti creati con un linguaggio di markup, come html o xml, dalla struttura del documenti. Questa separazione è particolarmente importante perché permette il riutilizzo di parti di codice e aumenta la leggibilità delle pagine. Il css è usato in particolare in accoppiata con html. Sia html che css sono gestiti dal World Wide Web Consortium (W3C).

2.2 HTTP

HTTP è un acronimo e sta per "*Hyper Text Transfer Protocol*". È il principale protocollo di comunicazione per la comunicazione su web. Il livello di astrazione di HTTP nella pila ISO-OSI⁶ è il livello 7, ossia l'application layer. Come protocollo

e gli aspetti semanticci del documento.

⁶La pila ISO-OSI è un modello di riferimento, proposto dall'ente di standardizzazione ISO, per l'interconnessione di sistemi aperti. Esso si basa su 7 livelli, che vanno dall'applicazione finale al livello fisico.

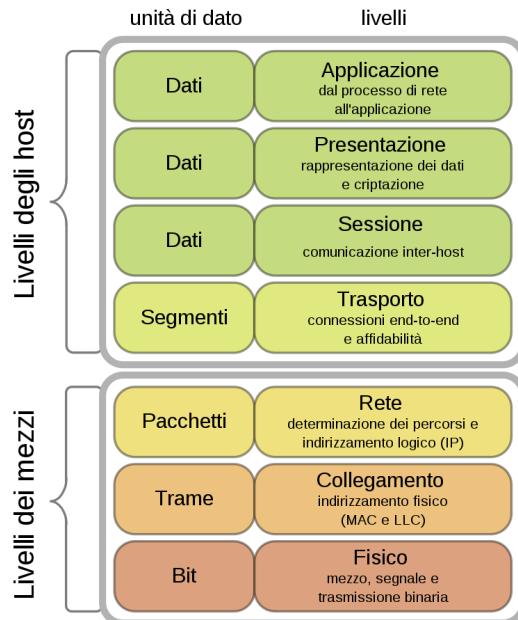


Figura 2.2: Pila ISO-OSI, il numero del livello è dal basso verso l'alto. Si noti l'Application layer, in alto dove si trova HTTP

sottostante HTTP si basa su TCP⁷ perché non sono ammesse perdite di pacchetti, mentre la sua porta⁸ di default è la 80. Il tempo impiegato per la trasmissione di un pacchetto con HTTP richiede un tempo pari a 2 RTT⁹ più il tempo di trasmissione del contenuto.

⁷Il protocollo TCP è uno dei due possibili protocolli del livello 4 (transport layer) della pila ISO-OSI. TCP garantisce la corretta consegna dei pacchetti, anche in ordine, usando un sistema detto *three-way handshake*. Si noti che in internet i layer 5 e 6 sono inglobati nel 7.

⁸Nel campo delle reti di calcolatori per porta si intende lo strumento per effettuare la multiplazione, ovvero la possibilità, per un singolo calcolatore di instradare correttamente i pacchetti, a livello di transport layer, verso il processo giusto.

⁹RTT sta per Round Trip Time, ossia il tempo che un pacchetto ci impiega ad arrivare ad una destinazione più il tempo che ci impiega il messaggio di acknowledge a tornare.

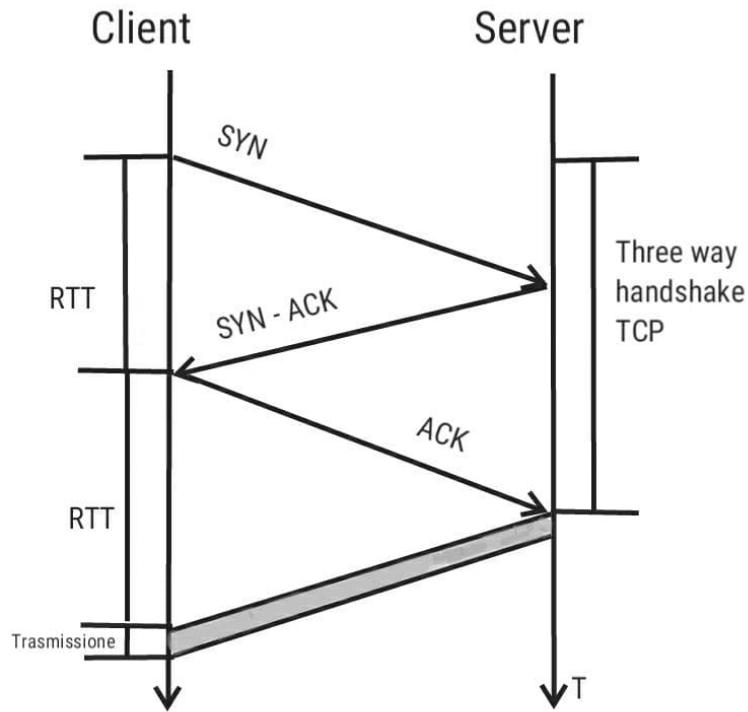


Figura 2.3: Schema per la trasmissione dell'informazione di un messaggio HTTP [16]

HTTP usa un paradigma di comunicazione denominato "request/response". In questo paradigma c'è un client che effettua una richiesta (request) per accedere alle risorse del server il quale risponde con una una risposta (response), contenente un codice che indica l'esito della request. Questo tipo di paradigma è definito sincrono, in quanto c'è una apertura della comunicazione e una chiusura con il responso. Per accedere alle risorse del server si usa un url (Uniform Resource Locator): l'url è composto dall'indirizzo del server, ad esempio <https://www.polito.it/>, da una parte detta uri che identifica la risorsa che si intende cercare e da una serie di parametri detti params, che sono delle informazioni utili alla richiesta.

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Figura 2.4: Esempio di Header HTTP

Una request di HTTP è composta da due parti: header e body. L'header è l'intestazione ed è composto da più parti tra cui il metodo REST utilizzato, di cui si parlerà in modo più approfondito in seguito, e l'url. Il body, che non è obbligatorio, contiene delle informazioni aggiuntive, come i dati di un form, o le misurazioni effettuate da dei sensori IoT.

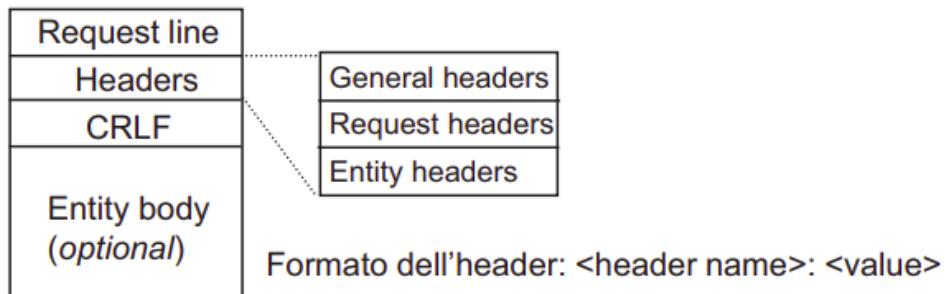


Figura 2.5: Struttura generale di una HTTP request.

HTTP, inoltre, è stateless: non tiene traccia delle precedenti richieste effettuate. Per sopperire a questa mancanza si usano i cookies: sono delle "briciole" di informazioni salvate nel browser del client che permettono al server di tenere traccia dell'identità del client.

2.2.1 HTTPS

HTTPS è una evoluzione del classico protocollo HTTP. Consiste in una "messa in sicurezza" della comunicazione, cifrandola. Per fare ciò si utilizza un protocollo che si colloca a cavallo tra il transport layer e l'application layer della pila ISO-OSI, ossia SSL (Secure Socket Layer). Dal 1999, in realtà SSL ha subito una evoluzione diventando TLS (Transport Layer Security). Questo protocollo si pone un gradino più in alto del TCP per sfruttare tutto ciò che c'è sotto, TCP incluso, senza trattare

la gestione dei dati. TLS offre una garanzia di sicurezza mediante tre caratteristiche principali:

- **Privacy del collegamento.** Per assicurare protezione i dati vengono protetti tramite algoritmi di crittografia a chiave simmetrica.
- **Autenticazione.** Per assicurare l'identità dei soggetti coinvolti è possibile usare sia della crittografia a chiave pubblica, sia dei certificati, che possono essere sia lato client, che lato server.
- **Affidabilità.** Viene effettuato un controllo di integrità sul messaggio tramite Message authentication code¹⁰

2.2.2 REST

REST è un acronimo che sta per "Representational State Transfer" ed è uno stile di architettura del software che impone costrizioni al fine di rendere interoperabili i servizi web. Un servizio web che supporta completamente REST è detto RESTful e permette di manipolare le risorse usando un set di operazioni predefinito e stateless. Queste caratteristiche permettono ad un servizio che supporta REST di essere veloce, affidabile e di crescere rapidamente riutilizzando vecchi componenti che possono essere modificati e aggiornati senza avere effetti sul sistema in generale, anche se questo è in funzione. I metodi standard sono:

- **GET.** È il metodo che usa il browser quando si connette ad un sito web. Serve per richiedere specifici dati da una risorsa nel server.
- **POST.** Si usa quando si vuole aggiornare o creare una risorsa. Se si effettua più volte, crea sempre nuove risorse.
- **PUT.** Si usa quando si vuole aggiornare o modificare o rimpiazzare una risorsa. Se si effettua più volte, quindi, non creerà risorse uguali ma le sostituirà.
- **PATCH.** Molto simile a PUT, ma può modificare la risorsa solo parzialmente, quindi non sostituisce tutto il contenuto.
- **DELETE.** Come suggerisce il nome, DELETE elimina completamente la risorsa.
- **HEAD.** Esattamente come GET, ma non ha il body nel response.
- **OPTIONS.** È un metodo poco usato ma che ha lo scopo di spiegare che cosa supportano gli altri metodi. È un po' come una funzione di "help".

¹⁰In crittografia un message authentication code (MAC) è un piccolo blocco di dati utilizzato per garantire l'autenticazione e integrità di un messaggio digitale, generato secondo un meccanismo di crittografia simmetrica[17].

2.2.3 Flask

Flask è uno dei web framework più utilizzati in Python, si pensi che LinkedIn e Pinterest sono sviluppati con Flask. Nello specifico è un micro-framework, perché il nucleo principale di Flask è estremamente ridotto, ma estensibile. Questo permette a Flask di essere estremamente flessibile e "trasparente", poiché permette all'utente di scegliere quello che vuole, come, ad esempio, il database usato. Supportando le estensioni, comunque, è abbastanza facile trovare ciò di cui si ha bisogno. Uno dei vantaggi è quello che ci sono poche dipendenze di base, riducendo il rischio di incappare in bug di sicurezza, ad esempio. Inoltre è relativamente facile iniziare con Flask, richiedendo poche righe di codice per le operazioni basilari. D'altro canto la sua estensibilità e il suo essere "micro" lasciano una enorme libertà all'utilizzatore che non è così vincolato da scelte fatte da altri[18].

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Figura 2.6: Esempio base di applicazione con Flask

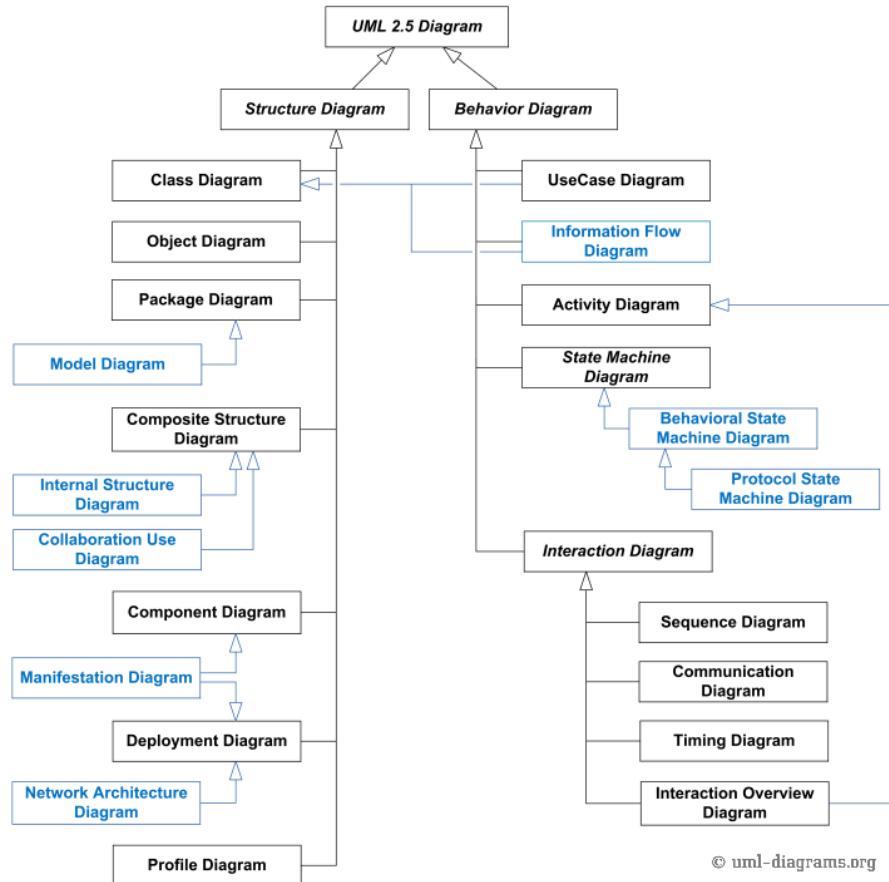
2.3 UML

Quando si costruisce un sistema informatico è normale scontrarsi con la complessità del sistema, non solo nell'immediato e nelle prime release ma, soprattutto, in termini di mantenibilità: un buon sistema, infatti, deve essere quanto più modulare e distribuito possibile, così che ogni modulo possa essere modificato, ampliato o sostituito in modo indipendente dagli altri. Così facendo il sistema può crescere molto più rapidamente. Distribuire il sistema, d'altro canto, porta fin dai primi stadi ad un aumento di complessità dello stesso, dal momento che introduce molti più componenti, sia hardware che software. Per questo motivo è importante utilizzare gli strumenti messi a disposizione da "UML" (Unified Modeling Language).

UML non è, ovviamente, un linguaggio di programmazione, bensì un linguaggio di modellazione. Creare dei modelli è importante perché permette di gestire la complessità, monitorare lo sviluppo e documentare le attività. Avere dei modelli è utile sia per permettere ai colleghi di comprendere più rapidamente cosa fa il sistema, sia per coordinare in parallelo lo sviluppo di diverse parti del software. Un esempio pratico può essere fatto da un sistema che esegue più funzioni. Si pensi, ad esempio, ad un sistema informativo che gestisce un reparto in un ospedale: se si modella il sistema a priori si possono portare avanti due team di sviluppo che si occupano di due funzioni (che UML chiama "casi d'uso"), come la prenotazione della visita o la distribuzione delle immagini o dei referti, in contemporanea. Non solo: utilizzare UML significa poter comunicare in modo semplice e schematico che cosa fa il software agli stakeholder¹¹, "nascondendo" la complessità sottostante in certi casi. In questo modo, anche un non addetto ai lavori può comunque capire il flusso di lavoro del sistema. UML può anche essere utilizzato anche come blueprint; utilizzando dei tool opportuni, si riesce a generare una serie di diagrammi a partire dal codice sorgente.

UML divide i diagrammi in due macro-sezioni: "Structure diagrams" e "Behavior diagrams". La prima tipologia descrive la struttura statica del sistema, mentre la seconda ne descrive il comportamento.

¹¹Letteralmente dall'inglese "titolare di una posta in gioco" è una persona che è coinvolta nel progetto.



© uml-diagrams.org

Figura 2.7: Divisione dei diagrammi in UML[19].

2.3.1 Use Case Diagram

Use Case Diagram significa letteralmente "diagramma dei casi d'uso" ed è il diagramma che modella le funzioni software del sistema. È utilizzatissimo come "riassunto" di tutte le funzionalità che il sistema implementa, mettendole in relazione agli attori che partecipano al processo, sia umani che non.

Lo Use Case usa due simboli principali mostrati in figura 2.8

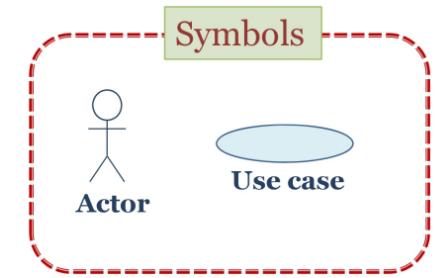


Figura 2.8: Simboli dello Use Case Diagram

Dove l'attore è chi interagisce con il sistema, sia esso un altro sistema o un utente, mentre il caso d'uso è la specifica funzione software correlata. Non ci sono solo delle correlazioni semplici nello Use Case ma è possibile includere un caso d'uso in un altro (funzione "include") o estendere un caso d'uso in un altro (funzione "extend"). Infine è possibile raggruppare più casi d'uso insieme in modo da avere, in un unico riquadro, i casi d'uso che appartengono ad una determinata categoria. Tutto ciò può essere fatto tramite dei "boundary box" ossia dei rettangoli intorno ai casi d'uso.

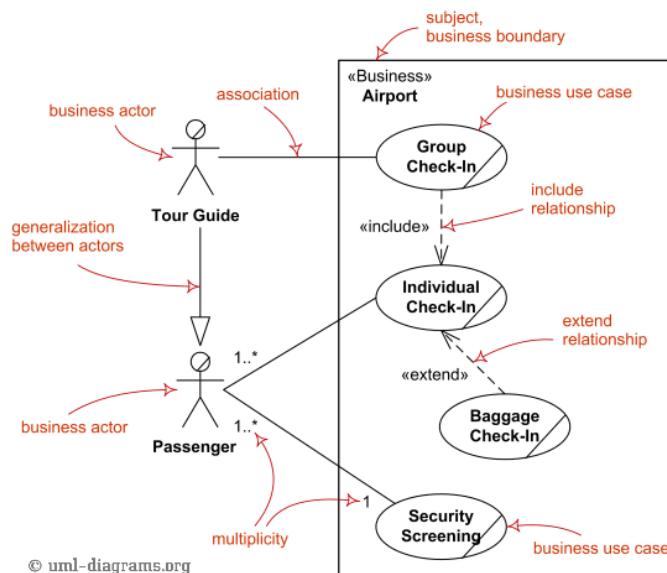


Figura 2.9: Esempio di un piccolo Use Case[19].

2.3.2 Activity Diagram

Appartenente all'insieme dei Behavior Diagrams l'Activity Diagram descrive il comportamento del software in funzione di ciò che fa l'utente. Per costruire

questo diagramma si deve partire quindi dalle interfacce, che possono essere delle "API"¹² o delle "GUI"¹³, per poi modellare il comportamento desiderato dal sistema. Questo diagramma è molto utile per testare il sistema, infatti seguendo il flusso del programma è possibile individuare eventuali situazioni che possono portare a bug.

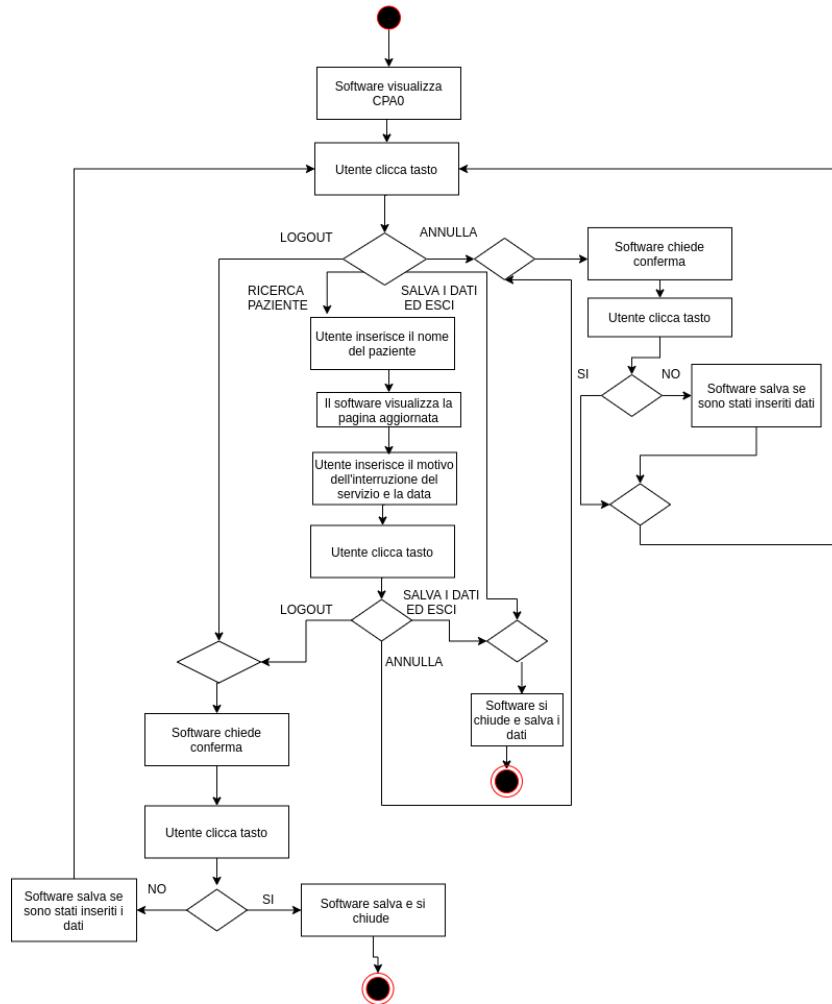


Figura 2.10: Activity Diagram fatto durante il corso di Progettazione di software medicali

¹²Per "API" si intende un tipo di interfaccia al solo livello di programmazione o comunicazione senza grafica.

¹³Letteralmente "interfaccia utente grafica" (graphical users interface) è un tipo di interfaccia che permette all'utente di comunicare graficamente con il sistema

2.3.3 Sequence Diagram

Il sequence diagram appartiene ad un sottogruppo dei Behavior diagram, gli "interaction diagram", di cui è il più comune. Il sequence si focalizza sullo scambio di messaggi attraverso delle "lifelines", ossia delle linee che indicano il tempo di attività di ogni attore che partecipa al processo.

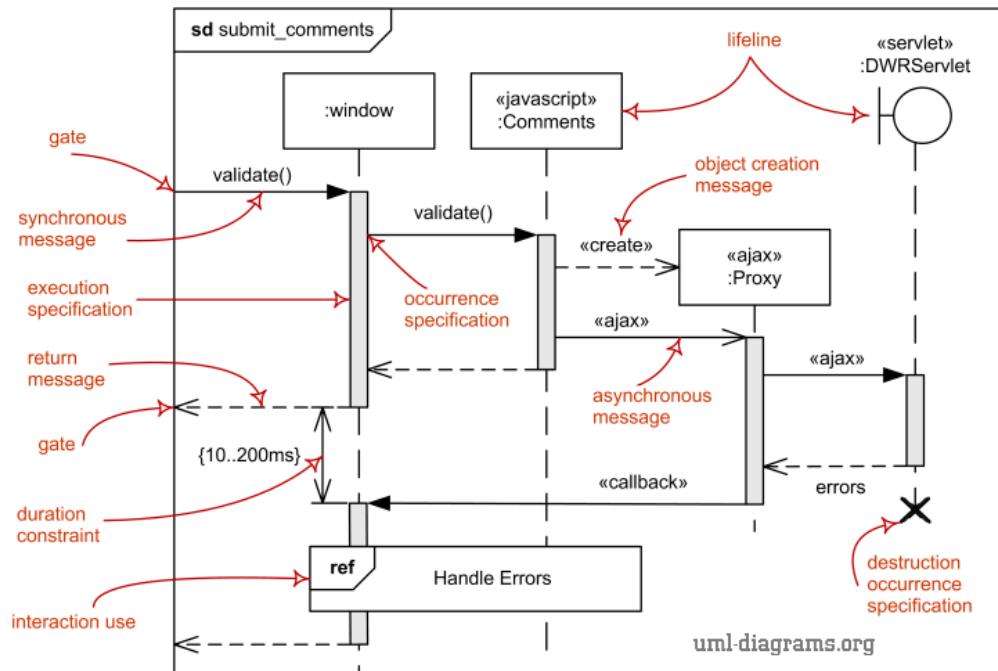


Figura 2.11: Esempio di Sequence Diagram[19].

2.3.4 Class Diagram

Il Class Diagram appartiene agli Structure Diagram, in quanto mostra le relazioni che ci sono tra le classi all'interno del sistema. Per classi vengono intese proprio quelle che vengono effettivamente programmate e poi instanziate come oggetti, sia come oggetti all'interno nel back-end¹⁴, sia a livello di front-end¹⁵. Uno dei più comuni modi di implementare il Class Diagram è quello di spiegare l'implementazione pratica delle classi includendo in questo caso pure gli attributi e i metodi della classe.

¹⁴Per "back-end" in informatica si fa riferimento alla parte di codice che non gestisce i comportamenti dell'interfaccia.

¹⁵Per front-end si intende la parte di software che gestisce il comportamento dell'interfaccia.

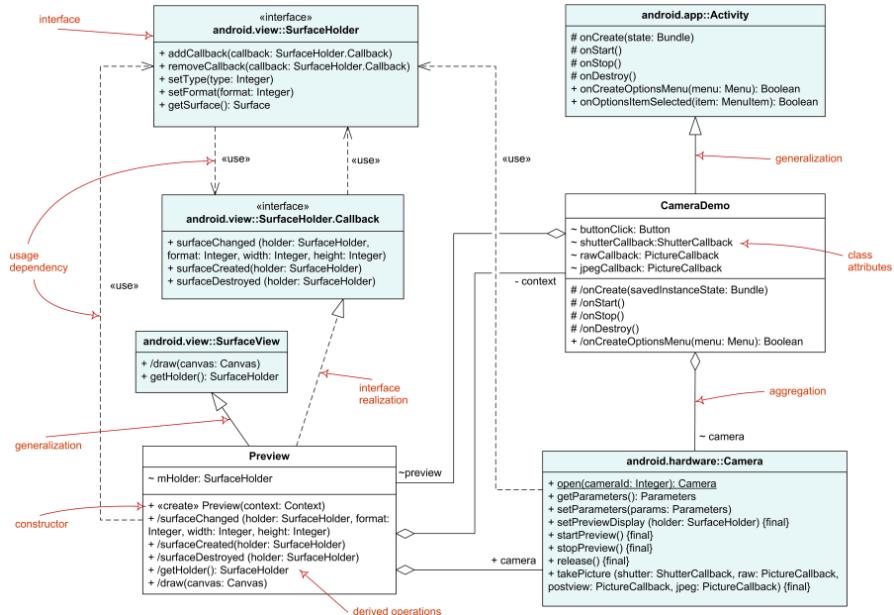


Figura 2.12: Esempio di Class Diagram

2.3.5 Deployment Diagram

Il Deployment Diagram è anch'esso uno Structure Diagram, in quanto mostra com'è sviluppato il sistema e com'è distribuito, anche all'interno del singolo dispositivo. Nelle presenti tesi, in realtà, verrà usato soprattutto per spiegare come vengono gestite le comunicazioni tra applicativi, dal momento che non esiste in UML un diagramma vero e proprio solo per le reti di computer. Questo caso particolare viene definito "Network Architecture Diagram" e, proprio perché è un caso particolare, alle volte si usano delle icone che non sono contemplate nello standard.

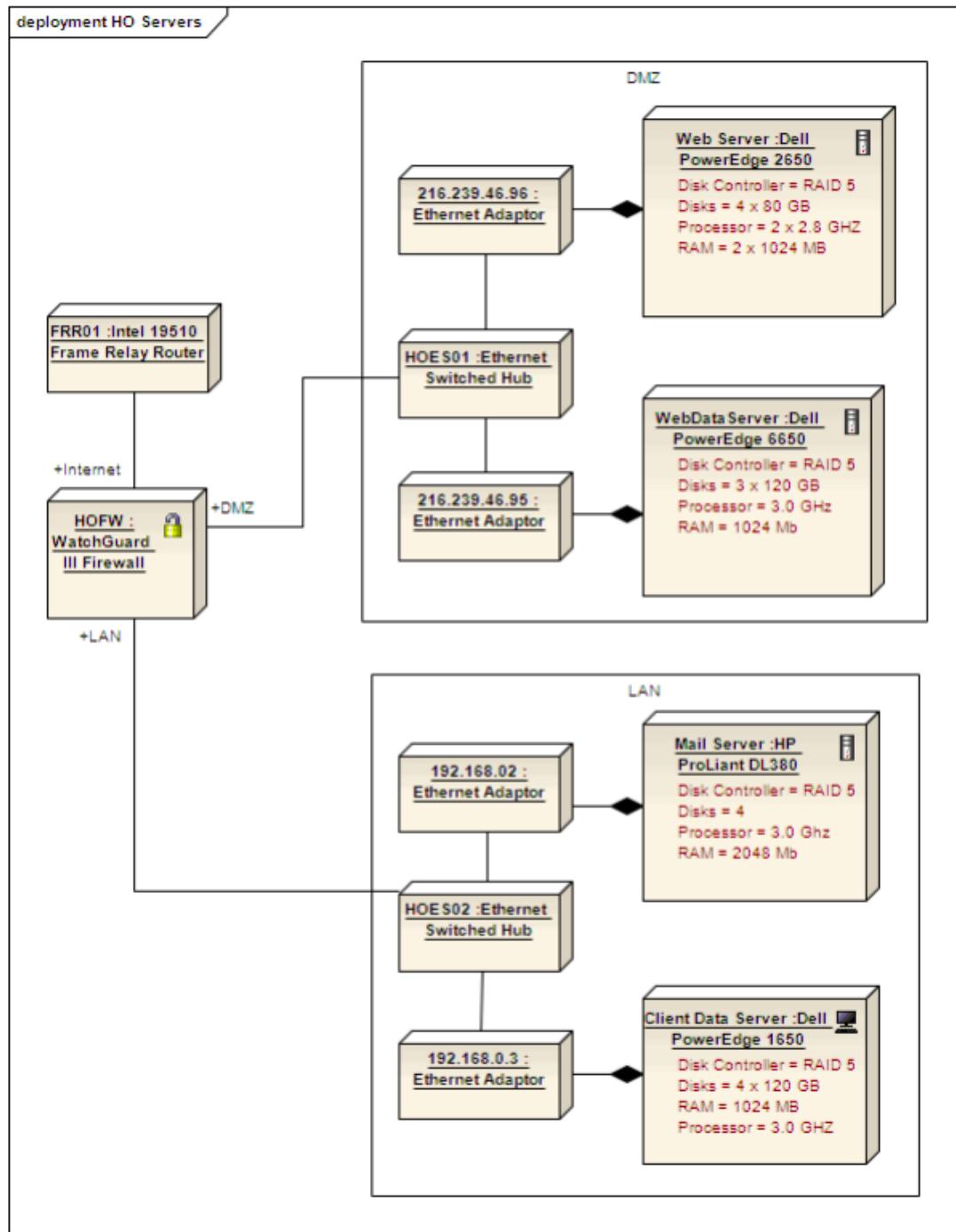


Figura 2.13: Esempio di Deployment Diagram con icone normali[21].

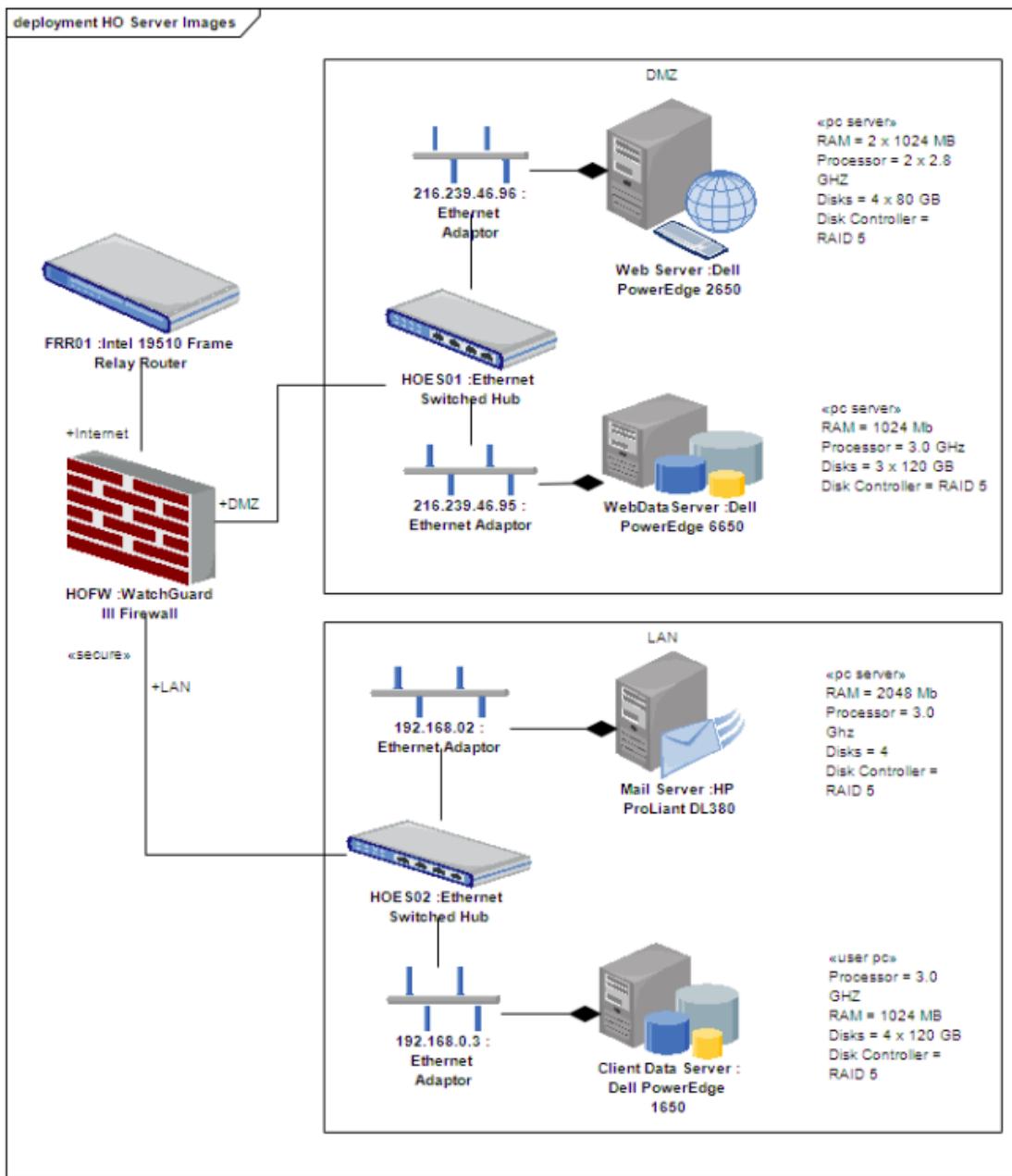


Figura 2.14: Esempio di Deployment Diagram identico al precedente ma con icone che non appartengono a UML[21].

2.4 Nomenclatura di dati biomedici

Per definire un concetto si possono utilizzare parole diverse, in modo estremamente soggettivo. Se si considerano pure diverse lingue la situazione non può che ingarbugliarsi, perché oltre alle possibili combinazioni di parole che una lingua offre si aggiungono quelle possibili nelle altre lingue. In una lingua come l'italiano, che spesso prende in prestito termini tecnici da altre lingue la situazione diventa ancora più complessa, in particolare se si considera l'implementazione in un sistema informatico. A meno che non si addestri specificatamente un sistema di machine learning al riconoscimento del linguaggio naturale tecnico, risulta praticamente impossibile far riconoscere ad un computer il contenuto di tutte le possibili frasi. La soluzione più semplice è quella di utilizzare dei sistemi di nomenclatura, ossia una standardizzazione dei termini medici in modo che si abbia un numero ridotto di classi.

2.4.1 SNOMED

FHIR normalmente usa LOINC, ma non è risultato essere adatto allo scopo della presente tesi, in quanto è più concentrato sui referti piuttosto che sui segnali. Infatti non possiede codici per definire le grandezze elettriche fisiologiche. Seguendo quindi la documentazione del progetto "RE HOME" si è deciso di adottare il sistema di classificazione SNOMED. SNOMED, acronimo per "Systematized Nomenclature of Human and Veterinary Medicine", è un ampio vocabolario strutturato di termini tecnici usati in medicina basato su una classificazione multiassiale a 11 assi, anche detti moduli, indipendenti. Ognuno di questi è completamente gerarchico. I vari codici dei vari assi, composti da una lettera che indica l'asse e un numero di 5-6 cifre, possono essere combinati tra di loro. Si prenda ad esempio il codice diagnostico D-13510, che sta per "polmonite pneumococcica". Esso è equivalente alla combinazione tra: T-28000 (codice topologico per il polmone), M-40000 (codice morfologico per l'infiammazione) e L-26116 (codice di organismi viventi per "streptococcus pneumoniae"). Si noti che la relazione che intercorre tra i tre codici combinati è data da relazione "è causato da". Uno dei problemi di questa combinazione è l'inesistenza di regole che stabiliscano come la combinazione debba avvenire, permettendo così di creare codici senza senso.

2.5 I database

Un database, anche detto "base di dati", è una collezione di dati volta a rappresentare le informazioni che interessano ad un sistema informativo¹⁶. Tale sistema è costituito da una grande quantità di dati che devono essere organizzati in insiemi omogenei in correlazione fra loro. I dati possono, inoltre, essere condivisi tra utenti e applicazioni in modo da ridurre le ridondanze. Un database è costituito da due componenti: uno schema, che è invariante nel tempo, e una istanza, che è variabile nel tempo in quanto rappresenta il valore effettivo del dato. Per gestire la base di dati si utilizza un "Database Management System" (DBMS). Il DBMS è un software che permette di nascondere all'utilizzatore la differenza di sistema operativo o file system sottostante, oltre che a farsi carico dell'organizzazione dei dati, permettendo di gestire i dati tramite "modelli". Inoltre un DBMS effettua i seguenti controlli per garantire un corretto funzionamento del sistema:

- **Controllo sulla concorrenza.** Il DBMS si occupa di gestire la contemporaneità degli accessi, in modo che non sorgano conflitti se più utenti (o processi) cercano di accedere contemporaneamente allo stesso dato.
- **Controllo sugli accessi.** Il DBMS si occupa di controllare l'accesso al database in base ai permessi dell'utente: ad esempio in un database ospedaliero è importante che il medico della radiologia, ad esempio, non abbia un accesso diretto ai dati della cardiologia.
- **Controllo sull'integrità e sulla consistenza.** Un DBMS deve occuparsi di controllare che i dati siano inseriti correttamente, secondo quanto inserito dal progettista nel modello concettuale (di cui si parlerà più avanti). Un esempio può essere che il DBMS deve accorgersi se un appuntamento per una visita è a carico di un medico che non è più presente nella struttura, o magari non è a nome di nessun medico.
- **Controllo della transazioni.** La transazione è un insieme di operazioni sul database che vengono eseguite tutte in blocco, se però, una di queste non può essere effettuata è necessario che il DBMS riporti il database allo stato precedente l'inserimento, perché questo causerebbe inconsistenze. Si prenda, ad esempio, l'inserimento del record di un paziente con tutti i suoi dati. Vengono inseriti uno ad uno gli attributi, finché non si incontra la data di nascita inserita come stringa, anziché in formato "data". In una situazione del genere il DBMS deve eliminare tutto il record del paziente per non rischiare

¹⁶Un sistema informativo può essere definito come “un insieme di elementi che intervengono e guidano il processo di trasformazione dell’evento in informazione.”[22]

di lasciare un paziente con informazioni lacunose. Il database deve quindi assicurarsi che una transazione sia "*ACID*", ossia goda delle seguenti proprietà:

- **Atomicity**: una transazione è un'entità di esecuzione indivisibile (come si diceva prima, o tutti gli effetti o nessuno);
 - **Consistency**: una transazione lascia il database in uno stato consistente, cioè opera una trasformazione corretta dallo stato del database (il DBMS garantisce che nessuno dei vincoli di integrità del database venga violato);
 - **Isolation**: una transazione viene eseguita indipendentemente dalle altre (se ci sono più transazioni in concorrenza il DBMS garantisce che l'effetto netto sia equivalente all'esecuzione seriale delle stesse);
 - **Durability**: gli effetti di una transazione che è stata eseguita correttamente devono durare nel tempo (il DBMS deve proteggere il database dai guasti);
- **Sicurezza dei dati.** Il DBMS deve farsi carico della protezione dei dati da fattori esterni come eventuali guasti all'hardware, utilizzando log files o backup.

Un DBMS è basato su tre livelli: interno, logico ed esterno. Il primo considera il database come un insieme di registrazioni in memoria di massa, in particolare viene posta l'attenzione alla distribuzione dei dati sui vari supporti e a come vengono memorizzati. Chiaramente questo livello del database è gestito dall'amministratore dello stesso. Il livello logico (o concettuale) presenta i dati in modo da far capire la loro struttura logica e descrive tutti i dati del database. Infine c'è il livello esterno che presenta i dati così come possono essere visti da una particolare classe di utenti in base ai permessi concessi.

I dati sono modellati secondo altre tre diverse tipologie: modello esterno, concettuale e logico. Il primo non è altro che ciò che può vedere l'utente del sistema, mentre per gli altri due è necessario approfondire nelle sotto-sezioni seguenti.

2.5.1 Modello concettuale

Il modello concettuale si pone come obiettivo quello di mettere in evidenza i concetti, da qui il nome, presenti del database e non tanto la struttura con cui poi sono effettivamente memorizzati. Il più famoso modello concettuale è il modello Entità-Relazione, che è basato, proprio come dice il nome, su due "pilastri": le entità e le relazioni.

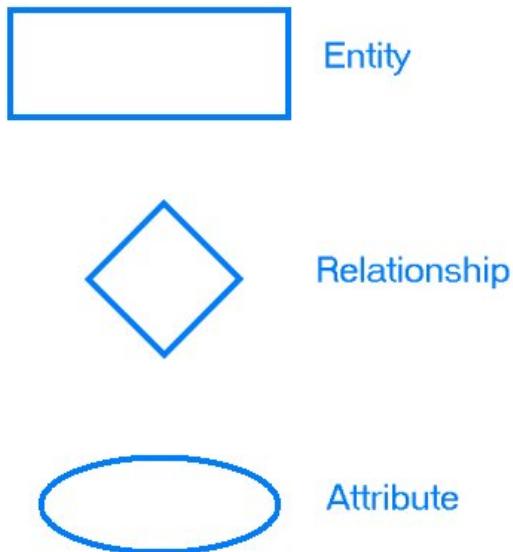


Figura 2.15: Elementi base di un diagramma E-R

Le entità sono l'astrazione dei dati che vengono inseriti. Detta in altri termini, sono dei raggruppamenti di dati con proprietà simili. Facendo un esempio pratico se in un database si vogliono raccogliere i pazienti ricoverati in una struttura sanitaria, tutti i record di ogni singola persona possono essere astratti sotto l'entità "paziente". Ogni entità a sua volta, come si è detto, possiede delle proprietà, denominate "attributi". L'entità paziente, ad esempio, può avere come attributi il nome, il cognome e il numero del letto in cui è ricoverato. Gli attributi possono essere a loro volta composti, cioè possono essere suddivisi in "sotto-attributi": si pensi all'attributo "indirizzo" che a sua volta può essere diviso in via, numero civico e città. Quando si ha una grossa mole di dati risulta difficile individuare uno specifico record, per questo è possibile utilizzare uno o più attributi come "chiave primaria" o "identificatore": in questo modo è possibile identificare in modo univoco uno specifico record. Si immagini un database di una realtà aziendale, nello specifico di una azienda sanitaria: ogni dipendente deve essere registrato all'interno del database. Una buona scelta su quale chiave scegliere è senza ombra di dubbio l'uso della matricola aziendale come identificativo, dal momento che la matricola, per definizione, è unica.

Si è parlato di entità, ma il modello Entità-Relazione (abbreviato E-R) è composto, per l'appunto, anche da relazioni. Una relazione non è altro che un legame tra entità. Il legame tra le entità ha un grado, che è definito come il numero di entità che partecipa alla relazione. Per esempio la relazione "partecipazione al processo di cura" coinvolge medico, infermiere e paziente e quindi ci sarà una relazione di

grado 3. Le relazioni, inoltre, possono essere ricorsive, ossia possono esserci altre entità che dipendono a loro volta da altre entità. Si prenda questo esempio: in una struttura ospedaliera si deve modellare la relazione che intercorre tra un infermiere caposala e un infermiere ordinario, considerando che nel database le due figure hanno gli stessi attributi. In questa situazione una relazione ricorsiva potrebbe risolvere la questione, a patto che si specifichi il "ruolo" che l'entità gioca nella relazione, come riportato in Figura 2.16.

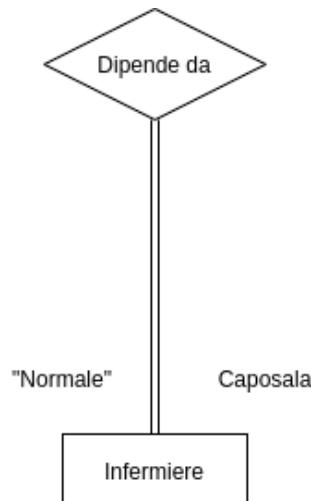


Figura 2.16: Esempio di relazione ricorsiva

In uno schema E-R sono presenti dei vincoli di integrità, cioè delle regole che si esprimono sullo schema e che esprimono una condizione che deve valere per ogni istanza dello schema. Ci sono quattro vincoli di integrità possibili:

- **Vincoli di cardinalità sulle relazioni.** Questo genere di vincoli indica il numero di entità che possono partecipare a tale relazione e, di conseguenza se è obbligatorio o meno. Ad esempio in un ospedale un infermiere lavora in uno e un solo reparto mentre un reparto può avere da zero a n infermieri, dove n è un numero arbitrario. Di conseguenza una istanza di infermiere non può non contenere una relazione con un reparto, mentre una istanza di un reparto può non contenere alcuna relazione con nessun infermiere, così come può averne benissimo dieci.
- **Vincoli di cardinalità sugli attributi.** Come nelle relazioni ci sono delle cardinalità, così negli attributi. Ad esempio una entità "persona" può avere una e una sola data di nascita, ma può non avere il numero di cellulare, così come può avere più numeri di cellulare. In questo caso, inoltre, è importante che il progettista del database decida quali attributi siano obbligatori, ponendo una

cardinalità minima >0 e quali sono opzionali, ponendo la cardinalità minima a 0. Nell'esempio fatto in precedenza sull'integrità della data di nascita, ad esempio, è sì vero che una persona non può non avere una data di nascita ma magari non è così importante all'interno del sistema, per cui si può accettare un record con questo dato lacunoso ponendo la cardinalità minima a 0.

- **Identifieri delle entità.** In precedenza si è introdotto il concetto di chiave primaria o identificatore. Si era detto che uno (chiave semplice) o più attributi (chiave composta) possono essere una chiave. In realtà è vero solo parzialmente: gli identifieri possono anche essere interni o esterni. Gli identifieri interni sono quelli che appartengono direttamente all'entità mentre gli identifieri esterni sono quelli che prendono in considerazione anche la relazione a cui partecipa l'entità. Un esempio inconscio di identificatore esterno composto si ha quando si identifica una persona con il suo nome e il nome del padre perché si prende un attributo dell'entità "persona" che ci si crea inconsciamente e la relazione "essere figlio/a di" in modo da creare un identificatore che comunemente funziona, perché, specie nelle piccole comunità, la possibilità che due genitori omonimi mettano lo stesso nome ai rispettivi figli/e è abbastanza limitata. Ovviamente questo è un esempio pratico ma non formalmente corretto in un database a meno che non si usino dei codici univoci per il genitore, perché la probabilità che una coppia padre-figlio/a abbia gli stessi nomi è fin troppo elevata.
- **Generalizzazioni.** La generalizzazione è utile quando si deve generalizzare una entità padre in una o più entità figlie. Se, ad esempio, si ha a che fare con un database ospedaliero le entità "medico" e "infermiere" possono essere generalizzate come una entità "dipendente", dal momento che condividono vari attributi. Infatti le entità figlie ereditano gli attributi dall'entità padre, oltre ad avere i propri.

2.5.2 Modello logico

Il modello logico è l'adattamento del modello concettuale in una rappresentazione presente nel DBMS, anche se ancora svincolata dai dettagli fisici, di cui si è detto se ne occupa il DBMS. Si possono individuare quattro tipi di modelli logici:

- **Modello gerarchico.** È stato sviluppato negli anni sessanta e rappresenta i dati come strutture ad albero gerarchico. Esso si basa su due concetti fondamentali: i record e la relazione genitore-figlio (PCR: Parent-child relationship).
- **Modello reticolare.** È stato sviluppato la decade successiva al modello gerarchico ed è una generalizzazione del precedente in cui una PCR può avere più di un genitore che partecipa alla relazione, oltre alla possibilità di contenere

riferimenti circolari (ossia un child può fare riferimento al parent da cui deriva). Questo modello non consente di superare i limiti dell'approccio gerarchico, ossia che bisogna partire dalla radice per poter accedere ai figli, passando per il padre.

- **Modello a oggetti.** Questo è il modello più recente di database che nasce dall'esigenza di gestire informazioni multimediali, come immagini, audio o video. Insieme ai dati sono quindi specificate le modalità di accesso più adatte al formato a cui si fa riferimento come metodi. I dati sono poi inglobati all'interno di classi, esattamente come nel paradigma di programmazione *Object Oriented*.
- **Modello relazionale.** È il modello logico più usato, oltre ad essere quello scelto nel presente progetto di tesi. Come unità fondante utilizza una serie di tabelle. Ogni tabella è una relazione, ma non nel senso che ha il termine nel modello concettuale. Tra le caratteristiche principali non si può non citare la semplicità e l'enorme diffusione che porta ad avere una trattazione teorica ricca e completa e l'esistenza di potenti linguaggi di interrogazione¹⁷ estremamente diffusi, come SQL, di cui si parlerà successivamente.

Come precedentemente esposto, per il progetto di tesi si è scelto un modello relazionale. Tale modello relazione si basa su delle tabelle, utilizzando degli strumenti come il calcolo relazionale e l'algebra relazionale. I concetti fondanti sono relazioni, tuple e attributi. La relazione, come si è già detto, è una tabella (per capire come si passa dal modello concettuale a quello relazionale si rimanda al Capitolo 3, dove verranno spiegati i passaggi eseguiti durante il presente lavoro di tesi), mentre una tupla è una riga di una istanza della relazione. Un attributo è una colonna della relazione.

Una relazione ha tre proprietà fondamentali: non esiste alcun ordinamento tra le tuple di una relazione, non esiste ordinamento tra gli attributi di una relazione ed infine non possono esistere due tuple completamente identiche. Per essere sicuri di soddisfare la terza proprietà è necessario introdurre il concetto di *superchiave*, che viene definita come un insieme di attributi di una relazione che soddisfa la proprietà di non avere la stessa combinazione di attributi per due o più tuple. A questo punto è utile definire una superchiave minimale, anche detta chiave: si definisce superchiave minimale la superchiave con meno attributi nell'insieme delle superchiavi. Viene da sé che possono esistere più superchiavi e più chiavi, di conseguenza bisogna definire il concetto di chiave primaria: la chiave primaria è la chiave che il progettista definisce allo scopo di individuare in modo univoco le tuple

¹⁷Un linguaggio di interrogazione è un linguaggio che permette di interrogare una base di dati o un sistema informativo da parte di un utente al fine di ottenere informazioni.

studente			
MATR	NOME	CITTA'	C-DIP
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

esame		corso				
MATR	COD-CORSO	DATA	VOTO	COD-CORSO	TITOLO	DOCENTE
123	1	7-9-97	30	1	matematica	Barozzi
123	2	8-1-98	28	2	informatica	Meo
702	2	7-9-97	20			

Figura 2.17: Tre istanze di tre relazioni diverse

della tabella. In questo modo si possono stabilire e formalizzare due proprietà della chiave primaria: unicità e non ridondanza. La prima accompagna la definizione di chiave, la seconda la definizione di superchiave minimale.

2.5.3 Algebra relazionale

Un modello relazione viene interrogato, come si è già detto, tramite linguaggi basati sul calcolo relazionale come SQL. Questi linguaggi si basano sulle regole dell’algebra relazionale, la quale prevede le seguenti operazioni:

- **Selezione**, che permette di scegliere da una tabella una o più tuple.
- **Proiezione**, che permette di creare una nuova relazione contenente le tuple precedenti ma limitate agli attributi selezionati.
- **Ridenominazione**, che consente di ridenominare gli attributi della relazione.
- **Unione**, che consente di costruire una nuova relazione che includa tutte le tuple a partire dalle relazioni di partenza. Si noti che le relazioni di partenza devono avere la stessa struttura e che le tuple presenti in due o più relazioni non vengono duplicate.
- **Intersezione**, che permette di costruire una nuova relazione che include tutte le tuple in comune nelle relazioni di partenza.
- **Differenza**, che permette di costruire una nuova relazione che include tutte le tuple che non sono in comune nelle relazioni di partenza.
- **Prodotto cartesiano**, che permette di costruire un nuova relazione a partire da due relazioni che possono anche avere una struttura diversa, poiché la

relazione risultato sarà composta da tuple aventi tutti gli attributi delle due relazioni di partenza. Si noti che il prodotto cartesiano avrà compre risultato una relazione con tutte le possibili combinazioni tra le due relazioni di partenza.

- **Join**, che consente di riunire in un'unica relazione tuple provenienti da più relazioni legate attraverso i valori di alcuni loro attributi. Una variante del join è il join naturale che ha lo stesso effetto ma non è necessario specificare gli attributi che legano le relazioni, poiché l'operatore userà gli attributi con lo stesso nome. Dal join è possibile arrivare ad altre due operazioni, ossia l'outer-join, che completa i campi che non corrispondono a "null" e il theta-join che è la combinazione di prodotto cartesiano e selezione

Come si evince dalla spiegazione, i primi tre operatori utilizzano operatori di base detti unari, cioè che hanno una sola tabella come operatore, mentre gli altri prevedono operatori di base binari, cioè prevedono l'uso di due tabelle.

2.5.4 SQL

SQL è un acronimo e sta per "Structured Query Language". È un linguaggio usato per operare su database relazionali, facendo interrogazioni e manipolazioni. Pur avendo delle caratteristiche da linguaggio imperativo¹⁸ SQL è un linguaggio dichiarativo, cioè un linguaggio che chiede di specificare le proprietà logiche delle informazioni ricercate.

SQL permette, come si diceva in precedenza, di fare interrogazioni sul database ma non solo. Esso, infatti, permette di effettuare anche altre operazioni. Si può quindi dividere SQL in quattro parti:

- **Data definition language (DDL)**. Permette di creare, modificare o cancellare la struttura del database attraverso tre comandi: *create*, *alter* e *drop*. Il primo comando serve a creare, ad esempio database o tabelle, alter serve a modificare la struttura interna ad una tabella o di altri oggetti interni ad una base di dati e infine drop serve per eliminare un intero database, una tabella o altri oggetti.
- **Data Manipulation Language (DML)**. È la parte di SQL che risulta più comune in quanto DML si occupa della gestione e del recupero dei record nel database. Attraverso i comandi *insert*, *update*, *delete*, *select*. Insert permette di inserire una tupla in una tabella, update di aggiornare dei record, delete permette di eliminare e infine select permette di richiedere dei dati.

¹⁸I linguaggi imperativi sono quei linguaggi che danno delle istruzioni al computer come se fossero "ordini". Esempi di linguaggi imperativi possono essere trovati in Python e Javascript, già trattati nella presente tesi, oppure in C e Java.

- **Data Control Language** (DCL). Permette di gestire gli utenti e i permessi.
- **Device Media Control Language** (DMCL). Permette di controllare i supporti dove vengono memorizzati i dati.

2.6 HL7

Uno dei problemi principali dell'interscambio di dati biomedici è il mantenimento di una forma che permetta ai vari applicativi di riconoscerla, in modo che possano comunicare tra di loro. La soluzione risiede quindi nello standardizzare le comunicazioni e i dati. Per questi ultimi, ad esclusione delle immagini, lo standard di riferimento è HL7.

HL7 sta per Health Level 7, in riferimento al livello più alto della pila ISO-OSI, anche se questo non significa che appartenga al livello 7. Significa, piuttosto, che corrisponde alla definizione concettuale di interfaccia di tipo paritario posta al settimo livello della pila ISO-OSI.

HL7 definisce le interfacce tra applicazioni dialoganti ma la realtà ospedaliera è così vasta che lo standard può essere applicato in modi differenti. Non è quindi possibile dire che sia *plug and play*¹⁹, poiché i due dispositivi possono avere implementato lo stesso messaggio in modi differenti sempre usando HL7, quindi occorre una negoziazione tra applicazioni concorrenti.

In realtà HL7 non è un unico standard, come si potrebbe pensare, ma una famiglia, perché sono state rilasciate tre versioni differenti: 2.x, 3.0 e FHIR.

2.6.1 Versione 2.x

La versione 2 è detta “trigger event”, cioè basata sulla centralità di eventi scatenanti, che danno inizio alla trasmissione del messaggio. L’evento scatenante dà avvio alla comunicazione e all’invio del messaggio, il quale, attraverso la rete, arriva a destinazione. Appena arriva, il sistema destinatario manda un “acknowledgement”, cioè una conferma di avvenuta ricezione, la quale viene poi recepita dal sistema che ha inviato il messaggio.

¹⁹Letteralmente "collega e usa" in informatica si intende una tecnologia che permette all’utente di usare la tecnologia all’interno del suo sistema senza conoscerne il funzionamento e senza che ci siano passaggi di installazione o configurazione

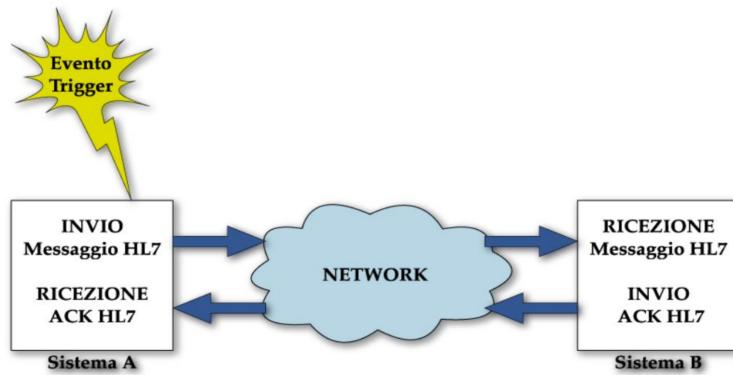


Figura 2.18: Invio di un messaggio HL7 v 2.x

La struttura di un messaggio di HL7 v2.x è composta da segmenti, davanti ai quali c'è un header, che identifica il tipo di messaggio²⁰ e l'evento trigger. I segmenti che sono dietro possono essere obbligatori o opzionali. Essi possono essere ripetuti o meno, mentre all'interno di ogni segmento le informazioni sono organizzate in campi. Ogni campo è concepito per contenere una stringa di caratteri e HL7 non si occupa di che cosa l'applicazione ne ricavi dal messaggio. Infatti, i campi possono essere per esempio stringhe di caratteri, numeri o indirizzi. I tipi di stringhe di caratteri nei campi sono elencate nella documentazione di HL7. È inoltre possibile trovare dei sottocampi. Inoltre, è possibile inviare il valore “null”²¹, utile soprattutto nell'aggiornare record di un database.

Nella figura 2.19 è possibile vedere un messaggio in HL7 v2.x. In grassetto è evidenziato l'header. Ogni riga è un segmento e i tre caratteri che stanno avanti sono il segment ID.

²⁰I messaggi HL7 vengono catalogati in varie categorie, in base alla loro funzione.

²¹Null in informatica è il sistema per indicare un campo senza valore. Piuttosto che lasciare vuoto si usa questa stringa.

```

MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126|SECURITY|ADT^A01|MSG00001|P|2.4
EVN|A01|198808181123
PID|||PATID1234^5^M11||JONES^WILLIAM^A^III||19610615|M||2106-3|1200 N ELM
STREET^^GREENSBORO^NC^27401-1020|GL|379-1212|271-3434~(919) 277
3114||S||PATID12345001^2^M10|123456789|9-87654^NC
NK1|1|JONES^BARBARA^K|SPO||||20011105
NK1|1|JONES^MICHAEL^A|FTH
PV1|1||2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||-||1|AO
AL1|1||^PENICILLIN||PRODUCES HIVES~RASH
AL1|2||^CAT DANDER
DG1|001|I9|1550|MAL NEO LIVER, PRIMARY|19880501103005|F||
PR1|2234|M11|111^CODE151|COMMON PROCEDURES|198809081123
ROL|45^RECODER^ROLE MASTER LIST|AD|CP|KATE^SMITH^ELLEN|199505011201
GT1|1122|1519|BILL^GATES^A
IN1|001|A357|1234|BCMD||||132987
IN2|ID1551001|SSN12345678
ROL|45^RECODER^ROLE MASTER LIST|AD|CP|KATE^ELLEN|199505011201

```

Figura 2.19: Esempio di un messaggio HL7 v 2.x

2.6.2 Versione 3.0 e CDA

La versione 3.0 di HL7 è nata nel 2005 con l'obiettivo di fornire un framework²² per accoppiare eventi, dati e messaggi, per muoversi in una direzione *plug and play* e per sfruttare le nuove tecnologie che nel frattempo erano nate, come XML²³. Infatti il formato in cui sono scritti i messaggi è proprio quest'ultimo metalinguaggio. Se la versione 2.x era trigger event, la versione 3.0 è object oriented e ingloba i principi di UML, così facendo si mira ad avere una maggiore chiarezza e ad aumentare il dettaglio. Un'altra novità, in direzione plug and play consiste nella possibilità di verificare la compatibilità, cosa molto complessa con la versione 2.x. La versione 3 introduce inoltre il CDA (Clinical Document Architecture). Esso non è altro che un veicolo per importare ed esportare dati clinici strutturati da e verso le applicazioni esistenti, in documenti autenticati e firmati [23]. In pratica esso specifica la semantica da usare in un documento XML, stabilendo il nome degli elementi del file XML che conterranno i dati. La novità del CDA, rispetto alla versione 2.x di HL7 è che può contenere testo, immagini, suoni e altro.

2.6.3 FHIR

FHIR è un acronimo per *Fast Healthcare Interoperability Resources* ed è l'ultimo standard proposto da HL7, nato con l'intento di ereditare le migliori caratteristiche dai suoi predecessori. FHIR può essere usato come strumento per scambiare informazioni sia "stand-alone" sia insieme alle precedenti versioni. Una delle principali caratteristiche di FHIR è quella di essere semplice da implementare, utilizzando

²²Framework in inglese significa "intelaiatura", "struttura"

²³XML è un metalinguaggio che permette la descrizione formale di linguaggi di markup.

concetti e strumenti ampiamente usati in informatica. FHIR, come la HL7 v3, è orientato a oggetti e il concetto principale da tenere presente è la "risorsa". Ogni entità del mondo reale è una "risorsa" in modo univoco. Nella v3 differenti modelli possono rappresentare lo stesso identico concetto, ad esempio esistono dieci modi diversi per rappresentare un "paziente". FHIR, invece, identifica il paziente in modo univoco. Certo, si possono creare più profili, ma hanno tutti lo stesso schema e la stessa serializzazione. FHIR permette di utilizzare due differenti formati: XML e JSON[20].

Le risorse sono un insieme di oggetti estremamente variegato: si va dal "Patient" allo "Schedule". Tutte le risorse, comunque, hanno tre componenti in comune:

- Un modo per definire la risorsa o per rappresentarla, in modo "human readable"²⁴ a partire da "data types" stabiliti.
- Un insieme di metadati, ossia di un insieme di informazioni come l'id o l'ultimo aggiornamento.
- Una parte "human readable" dove si spiega il contenuto della risorsa

Structure

Name	Flags	Card.	Type	Description & Constraints	?
Meta	Σ [N]		Element	Metadata about a resource	
versionId	Σ	0..1	id	Elements defined in Ancestors: id , extension	
lastUpdated	Σ	0..1	instant	Version specific identifier	
source	Σ	0..1	uri	Identifies where the resource comes from	
profile	Σ	0..*	canonical(StructureDefinition)	Profiles this resource claims to conform to	
security	Σ	0..*	Coding	Security Labels applied to this resource SecurityLabels (Extensible)	
tag	Σ	0..*	Coding	Tags applied to this resource Common Tags (Example)	

Figura 2.20: Struttura dei metadati in FHIR

Si è detto che lo standard HL7 è di vitale importanza nello scambio di informazioni in ambito sanitario e FHIR non risulta sprovvisto di metodi di comunicazione tra applicativi. In particolare, FHIR è descritto come un "RESTful", ossia possiede

²⁴Per "human readable" si intende un elemento informatico che può essere letto da un essere umano.

tutti i metodi REST implementati. Lo standard spiega anche come implementare questi metodi:

```
http(s):// VERB [base]/[type]/[id] {?_format=[mime-type]}
```

Dove "VERB" sta per il metodo REST che si intende utilizzare, "base" è l'URL del server con cui si vuole comunicare mentre "type" è il tipo di risorsa che si vuole modificare (ad esempio "Patient"). "Id" è opzionale in alcuni metodi e indica l'id unico per identificare una singola risorsa del determinato tipo specificato. All'interno delle parentesi graffe, che sono opzionali, si trovano i parameters che indicano i parametri della request che si intende fare. Infine, quando il metodo REST lo supporta è possibile inserire nel body la risorsa nel formato specificato nel parametro "format".

I metodi supportati, riconoscibili unicamente dalla sintassi standard riportata, possono essere divisi in tre macro-categorie: "Instance Level", "Type Level" e "Whole System". La prima racchiude i metodi che vanno a toccare una singola risorsa e contiene i seguenti metodi:

- **READ.** Il metodo READ legge, proprio come dice il nome, una specifica risorsa. La sua struttura base è la seguente:

```
GET [base]/[type]/[id] {?_format=[mime-type]}
```

- **VREAD.** Questo metodo è pressoché identico al precedente se non che legge una specifica versione di una determinata risorsa, per cui aggiunge la variabile tra parentesi graffe *vid*.

```
GET [base]/[type]/[id]/_history/[vid] {?_format=[mime-type]}
```

- **UPDATE.** Aggiorna una risorsa oppure la crea se non esiste.

```
PUT [base]/[type]/[id] {?_format=[mime-type]}
```

- **PATCH.** Modifica determinati campi di una risorsa. Il body di questo metodo può essere di tipo "JSON patch", "XML patch" o "FHIRPath Patch".

```
PATCH [base]/[type]/[id] {?_format=[mime-type]}
```

- **DELETE.** Elimina una risorsa. Il body deve essere vuoto, chiaramente.

```
DELETE [base]/[type]/[id]
```

- **HISTORY.** Riprende tutta la storia dei cambiamenti di una particolare risorsa. Di fatto è una estensione di vread.

```
GET [base]/[type]/[id]/_history {?_format=[mime-type]}
```

Il "Type Level" è su scala più larga perché va ad avere effetto su più risorse di un dato tipo. Contiene i seguenti metodi, che sono spesso estensione di quelli precedenti:

- **SEARCH.** Può essere visto come una estensione del metodo read, solo che ritorna più risorse. Qualora si effettuasse questo metodo da browser, lo standard raccomanda di usare il metodo POST, perché altrimenti compaiono i parametri di ricerca nella barra delle ricerche.

```
GET [base]/[type]{?parameters}{&_format=[mime-type]}
```

- **create.** Crea una nuova risorsa. In un certo senso pure il metodo update già descritto gli può essere equivalente.

```
POST [base]/[type] {?_format=[mime-type]}
```

- **HISTORY.** Identico al metodo già visto può essere fatto semplicemente senza inserire l'id come URI.

```
GET [base]/[type]/_history {?_format=[mime-type]}
```

Infine c'è il "Whole System", che va ad avere effetti sull'intero sistema.

- **CAPABILITIES.** Ritorna le possibilità del server.

```
GET [base]/metadata{?mode=[mode]} {&_format=[mime-type]}
```

- **BATCH/TRANSACTION.** Questi metodi effettuano una serie di operazioni sul server in una sola request.

```
POST [base] {?_format=[mime-type]}
```

- **search.** Questo metodo è identico a quello già visto ma nell'uri viene tolto il tipo, perché può riferirsi a qualsiasi risorsa.

```
GET [base]{?[parameters]{&_format=[mime-type]}}
```

- **HISTORY.** Il comando HISTORY si ripropone in tutte le macro-sezioni ma ogni volta perde un elemento dell'URI. In questo caso viene eliminato pure il tipo di risorsa, in modo analogo al metodo SEARCH.

```
GET [base]/_history {?_format=[mime-type]}
```

2.7 Middleware

L'idea di questo progetto di tesi è quella di costruire un sistema che sia "distribuito". Un sistema distribuito è definito come un insieme di processori che non condividono memorie o clock, la cui comunicazione avviene solo tramite linee di comunicazione [24]. È dunque chiaro che sono sistemi eterogenei, la cui principale caratteristica è la flessibilità e la scalabilità del sistema. Guadagnando questa caratteristica i sistemi distribuiti perdono capacità di integrazione tra di loro. Proprio per questo esistono middleware. Un middleware è "*un insieme di servizi di supporto alla distribuzione indipendenti dalle applicazioni. In definitiva il middleware è il software che risiede al di sopra della rete ed al di sotto delle applicazioni software.*"[25]

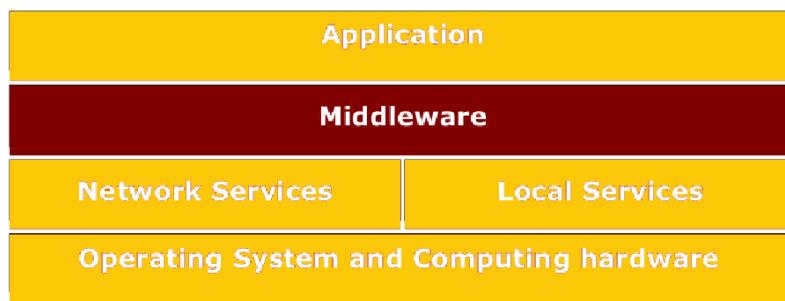


Figura 2.21: Posizione del middleware rispetto agli altri servizi

I middleware sono quindi dei “traduttori” che servono a tradurre dei messaggi con differenti tipologie di codifica e diversi parametri di ritorno. Sono indispensabili per lo scambio di dati complessi e per sincronizzare e far lavorare in parallelo i diversi applicativi. Per fare ciò nel modo più semplice possibile il middleware deve essere trasparente, cioè deve nascondere la complessità sottostante, dal momento che i sistemi distribuiti, per la loro eterogeneità, sono molto complessi.

2.7.1 Mirth

Mirth è un middleware open source²⁵ specifico dell’ambito sanitario e viene mantenuto dalla NextGen Healthcare. Il suo essere specifico dell’ambito sanitario e il suo essere open source ne fanno uno strumento estremamente diffuso, per permettere la comunicazione dei vari sistemi informativi sanitari. Mirth, infatti, è in grado di integrare una serie di messaggi in ingresso e modificarli tramite filtri e script in modo da essere tradotti per altri applicativi. Il software è molto potente anche grazie al suo cuore scritto in Java, permettendo dunque di utilizzare varie librerie di questo linguaggio. È però più immediato di Java grazie alla possibilità di scrivere script completamente personalizzabili in Javascript.

L’interfaccia base che usa Mirth per gestire i vari messaggi viene definita *Channel*, cioè canale. Ogni canale ha a sua volta dei componenti che hanno diversi ruoli, come mostrato in figura 2.22.

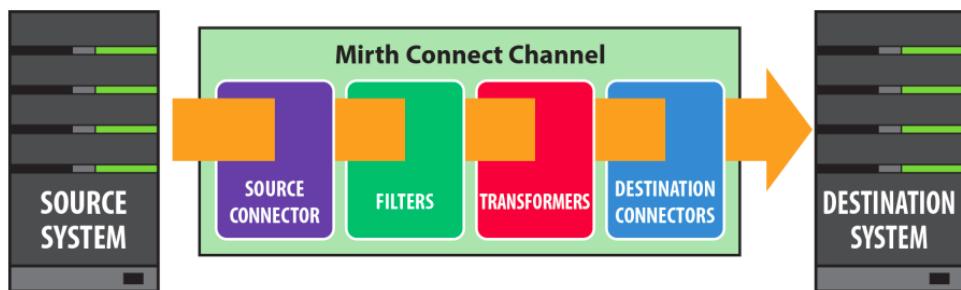


Figura 2.22: Schematizzazione di un canale di Mirth[26].

Aprendo l’interfaccia di amministratore (Mirth Connect Administrator) e cliccando sul canale si possono notare i quattro pannelli che suddividono il canale: *Summary*, *Source*, *Destination* e *Scripts*.

²⁵Un software è open source attraverso una licenza tramite cui i detentori dei diritti permettono la modifica, lo studio, l’utilizzo e la redistribuzione del codice sorgente.

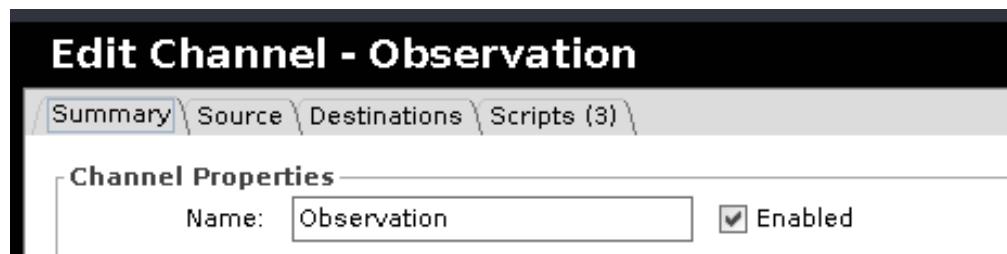


Figura 2.23: Dettaglio dei pannelli disponibili per il canale Observation

Summary è il pannello da cui si possono scegliere le impostazioni generali del canale, come la quantità di metadati da salvare o il formato dei dati attraverso il workflow del messaggio. Dopo Summary c'è il pannello Source che indica il primo elemento che entra in contatto con il messaggio. È possibile scegliere tra una vasta gamma di Source adatte a ogni esigenza. Dalla figura 2.24 si possono notare i tipi di connettori standard e quelle effettivamente utilizzato nel progetto, FHIR listener, installato tramite l'estensione apposita.

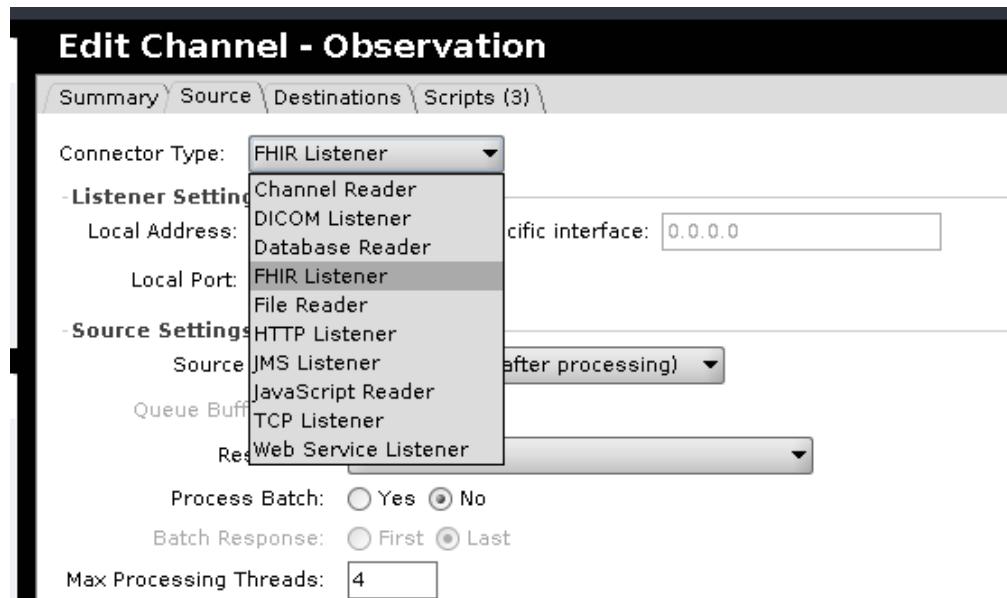


Figura 2.24: Dettaglio delle possibili opzioni come Source

Se Source si fa carico del messaggio appena arriva, Destination è il punto di arrivo del messaggio: è possibile salvare i dati come file o su un database oppure inviare nuovamente quel messaggio semplicemente "tradotto" verso un'altra destinazione. Bisogna notare che se per ogni canale c'è un solo pannello Source questo non è vero per il Destination: è proprio compito del Source instradare verso la corretta destinazione il messaggio.

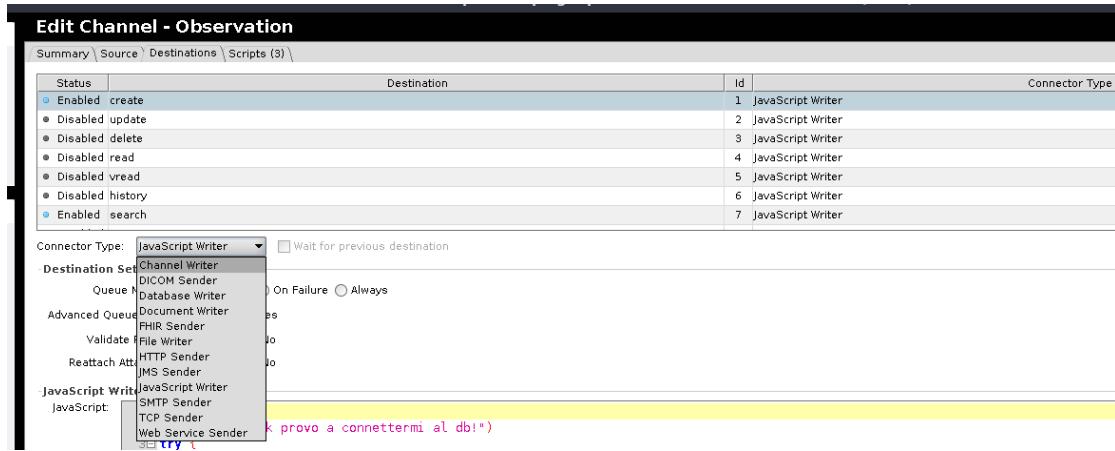


Figura 2.25: Dettaglio del pannello Destination: si noti la possibilità di avere più destinazioni.

È interessante notare come nelle destinazioni sia possibile scrivere un codice Javascript in maniera libera, in modo da non avere vincoli nell'operare sul messaggio. Ovviamente, ci sono altre possibilità di scrivere il proprio codice Javascript, ad esempio nel pannello *Script*. In questo punto si possono inserire degli script che facciano il loro corso quando viene avviato o dismesso il canale, oppure come preprocessing o postprocessing rispettivamente prima del Source e dopo la destinazione. Infine è importante ricordare che sia Source che Destination supportano filtri e trasformazioni. I filtri sono degli strumenti che permettono di, proprio come dice il nome, filtrare il messaggio in modo che sia possibile scegliere se farlo passare alla fase di trasformazione o meno. Quest'ultima permette di convertire il messaggio nel formato corretto.

Per riassumere la trattazione la figura 2.26 spiega che cosa avviene al messaggio quando viene inviato a Mirth.

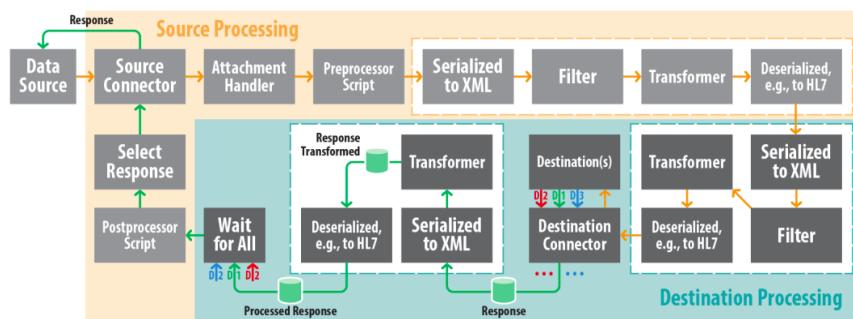


Figura 2.26: Workflow di un messaggio con Mirth[26].

Il messaggio arriva al Source, dove viene gestito un eventuale allegato, viene eseguito l'eventuale script di preprocessing e poi tutti i filtri e le trasformazioni. Successivamente si passa alla destinazione che applica a sua volta i suoi filtri e le sue trasformazioni, per poi finire al gestore delle varie destinazioni (nello schema si chiama Destination Connector), che indirizza l'informazione alla destinazione giusta, la quale, dopo aver fatto quello che deve, consegna il responso (response) al Destination Connector. Vengono così applicate eventuali trasformazioni al response e si aspetta finché non sono pronti tutti i risultati in uscita dalle varie destinazioni. Infine interviene il postprocessor e il Source sceglie il response definitivo da mandare indietro.

Capitolo 3

SleepingRepo

Il progetto *SleepingRepo* nasce dall'esigenza di integrare insieme i dati raccolti per studiare l'andamento del sonno dei pazienti. Come si è visto dal Capitolo 1 ci sono una grande varietà di sensori che possono essere utilizzati, alcuni in combinazione, altri possono introdurre ridondanze. Partendo da questa esigenza si è cominciato a fare l'analisi dei requisiti, cercando quindi di capire quali possano essere i processi in gioco e come semplificarli al massimo per rendere il sistema più *user friendly* possibile. A quel punto si è pensato al software e, come si vedrà nella prossima sezione, all'individuazione dei casi d'uso nello Use Case Diagram. Un altro step è stato stabilire le comunicazioni, i protocolli da utilizzare e la struttura della rete, dal momento che idealmente il sistema vuole essere distribuito, al fine di mantenere una maggiore flessibilità. Nello scegliere la rete si sono dovuti scegliere computer per la creazione del prototipo e si è optato per l'uso di una macchina virtuale all'interno di una reale. Entrambi i sistemi operativi sono basati su kernel Linux, per avere un più facile accesso ad un terminale bash¹. Per non perdere la semplicità di un sistema facile da utilizzare si sono scelte due distribuzioni basate su *Ubuntu*, essendo di fatto la base delle distribuzioni Linux più *user friendly*. Conoscendo la rete, conoscendo l'hardware a disposizione e conoscendo i processi si può cominciare ad entrare più nel dettaglio pensando alla creazione del database, dal suo modello concettuale, collegato dalla effettiva implementazione, alla scelta del DBMS con la relativa implementazione. Il processo di progettazione del database ha richiesto molta attenzione perché si è dovuto tenere conto sia delle eventuali problematiche implementative a livello di codice di interrogazione, sia di integrazione, sia di velocità e infine di aderenza alla realtà. Una volta conclusa la progettazione del database si è potuto pensare alla struttura delle classi in generale, con particolare attenzione alle risorse FHIR da utilizzare. Successivamente si è passato a costruire le interfacce

¹Bash è una shell testuale usata essenzialmente sui computer Unix e su macOS.

e quindi all'ideazione delle classi del webserver e infine all'implementazione del codice vero e proprio.

Anche dopo che si è arrivati a quest'ultima fase, le parti precedenti non sono state accantonate, ma hanno subito varie modifiche. Alcuni cambiamenti sono stati riportati per permettere di capire meglio il processo di sviluppo, anche se non tutti perché certi passaggi si sono rivelati fallimentari. Si pensi, ad esempio, che si è dovuto cambiare completamente il paradigma del web server in Python perché è stato necessario cambiare libreria in fase avanzata del progetto. Inizialmente si era scelto il framework CherryPy, per programmare in maniera *pythonica*², ma dopo svariati tentativi si è incappato in un possibile bug³ della libreria, apparentemente inspiegabile, che non permetteva il funzionamento del framework utilizzando la crittografia SSL su alcuni computer.

3.1 Il contesto

Prima di iniziare a progettare un sistema informativo bisogna comprendere a pieno come funziona il modo di lavorare delle persone per cui si progetta, in modo che il software sia il più possibile allineato con i loro bisogni. Per arrivare a questo obiettivo è necessario comprendere e studiare i requisiti e i processi. Per processo si intende un insieme di azioni portate avanti da attori, che siano umani o non. Dietro un processo c'è sempre un sistema, che non è altro che una serie di elementi legate da diverse relazioni.

Note queste definizioni è importante che si cominci a capire come funziona il sistema, modellarlo e automatizzare i processi che lo compongono. In questa sezione si mostrerà il lavoro svolto durante la tesi mediante tre modelli: Synopsis, Workflow e Use Case. I primi due non appartengono a UML e sono propriamente dei metodi per modellare i sistemi, indipendentemente dalle implementazioni software. Lo Use Case, come è noto dal Capitolo 2, è un linguaggio di modellazione per sistemi informatici, quindi con esso si vedrà il primo passo verso la costruzione del software.

3.1.1 Synopsis

Il Synopsis Diagram è particolarmente importante nel riassumere i processi di un sistema: esso tiene conto dei partecipanti, degli input e output, dell'evento scatenante e dello stato del sistema alla fine del processo[27].

²Per programmazione pythonica si intende un modo di programmare che non solo segue le regole e la sintassi di Python, ma che ne usa le convenzioni degli sviluppatori in Python e utilizza il linguaggio nel modo in cui è stato pensato.

³In informatica un bug è un comportamento anomalo del software

Nel caso specifico del progetto di tesi sono stati individuati due processi: upload e download. Essi sono i principali usi che verranno fatti dai ricercatori quindi è utile concentrarsi su questi.

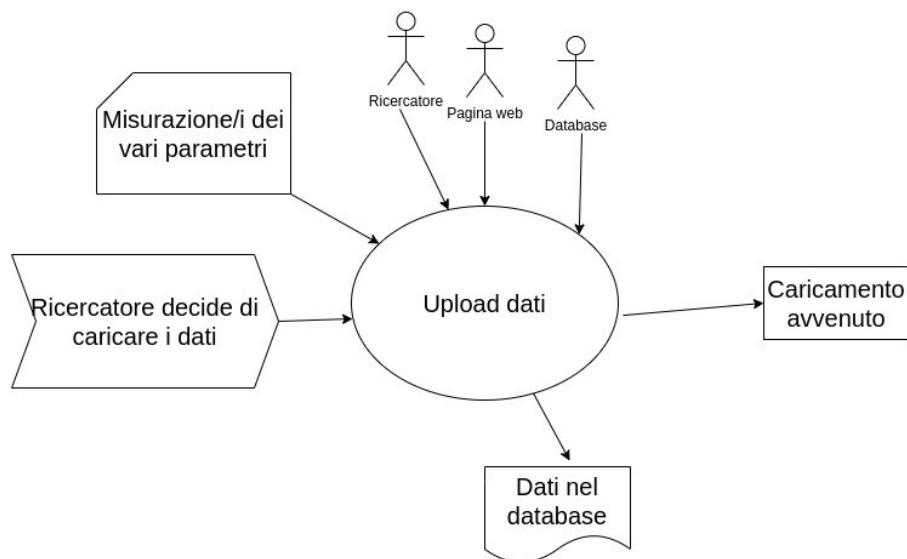


Figura 3.1: Synopsis del processo di upload.

Nella figura 3.1 è riassunto tutto il processo di upload. Nonostante sembri semplice è importante soffermarsi sulle varie parti del modello. In alto, con la figura di un uomo stilizzato sono rappresentati gli attori in gioco. Sicuramente un attore non può che essere il ricercatore, che è colui che carica i dati. Gli altri due attori sono macchine, di cui uno è la pagina web, con cui il ricercatore si interfaccia e l'altro è il database, colui che tiene i dati. Come dati in input, in alto a sinistra ci sono i vari parametri. Al centro a sinistra invece c'è l'evento scatenante, ossia la decisione del ricercatore di caricare i dati. Come stato finale (si vede tutto a destra) c'è l'avvenimento del caricamento. Infine come dati in uscita ci sono i dati salvati nel database.

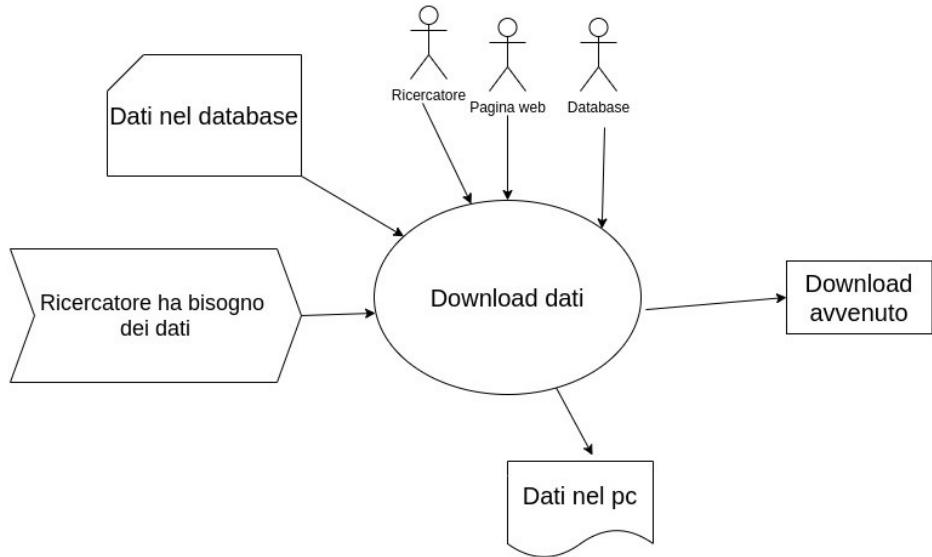


Figura 3.2: Synopsis del processo di download.

Guardando la figura 3.2 si può notare che il processo è pressoché identico a quello di upload, ma invertito. Gli attori presi in considerazione sono gli stessi, lo stato finale è la buona riuscita dell'operazione e l'evento scatenante è il bisogno di dati da parte del ricercatore. Se in fase di upload le misurazioni appena fatte erano i dati in input e i dati nel database erano i dati di output qui è esattamente l'inverso: i dati nel pc sono l'output e i dati nel database sono i dati di partenza.

3.1.2 Workflow

Il Workflow è usato per descrivere le azioni fatte durante il processo. È basato sul formalismo delle reti di Petri⁴ e si basa su due elementi fondamentali: il posto, rappresentato con un cerchio, e la transizione, rappresentata con un rettangolo. Si è cercato quindi di immaginare il flusso di lavoro che un sistema di questo tipo possa percorrere per i due processi individuati.

⁴La rete di Petri è una delle varie rappresentazioni matematiche di un sistema distribuito discreto.

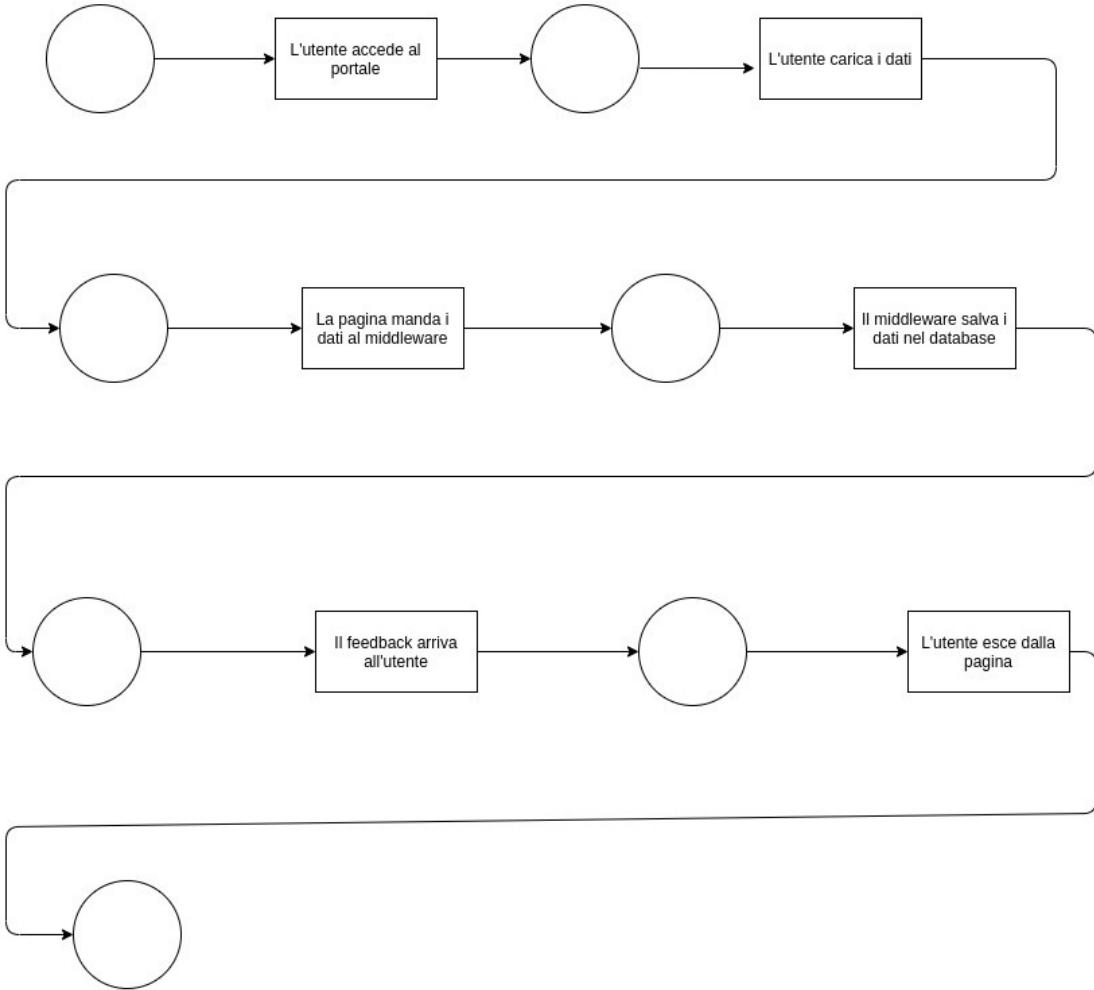


Figura 3.3: Workflow del processo di upload.

In fase di upload sono stati individuati alcune transizioni che portano a vari posti, ossia alle circonference che si vedono nelle Figure 3.3 e 3.4. Si inizia con l'accesso al portale, per poi caricare i dati. La pagina manda i dati al middleware che si occupa di salvare i dati nel database. Infine arriva un feedback all'utente e quest'ultimo chiude la pagina, perché ha finito. Come si è già visto in precedenza la fase di download è per molti aspetti speculare a quella di upload. Come mostrato in figura 3.4 l'utente accede al portale in ogni caso. A questo punto farà in modo di selezionare i parameter della query, ovviamente attraverso una interfaccia user friendly. I parametri arrivano al middleware che deve preoccuparsi di effettuare la

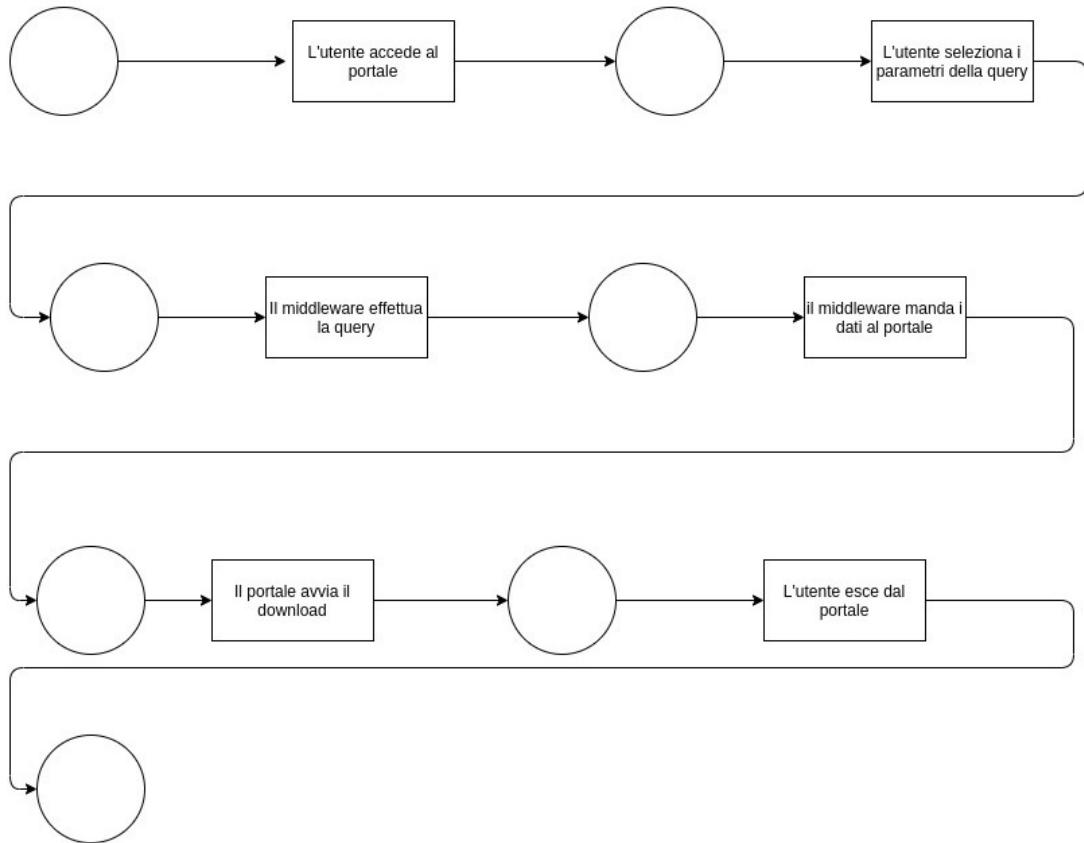


Figura 3.4: Workflow del processo di download.

query. Al termine della query sul database il middleware deve rimandare indietro i dati. Una volta arrivati al portale i dati verranno scaricati dal browser e l’utente a questo punto può chiudere il portale.

3.1.3 Use Case

Si è già trattato in maniera teorica lo Use Case diagram come parte di UML, nel capitolo 2. Arrivati a questo punto è necessario distinguere tra processo e caso d’uso. Un processo, come si è già detto, è un insieme di azioni, che servono a raggiungere lo scopo. Vengono dunque escluse le altre funzionalità software che fungono da corollario al processo vero e proprio. Di conseguenza, una volta che si sono modellati i processi si è passati all’individuazione dei "casi d’uso", ossia delle funzioni software da inserire affinché il sistema porti a termine i processi desiderati.

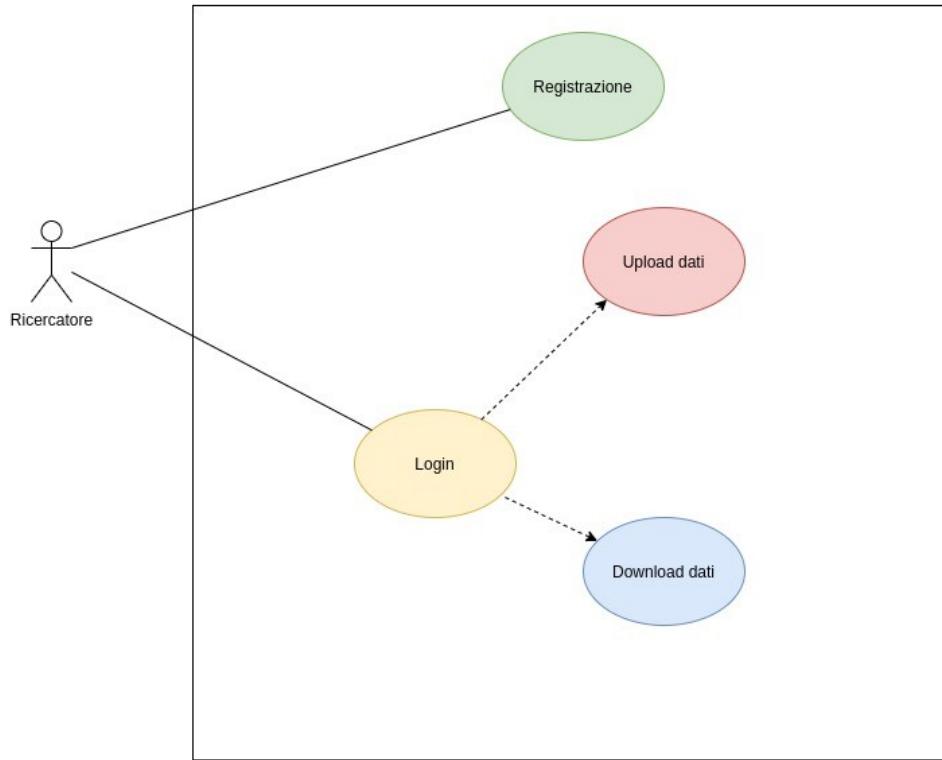


Figura 3.5: Use Case del progetto SleepingRepo.

Come si può vedere dalla figura 3.5 ci sono due casi d'uso principali: la registrazione per i nuovi utenti e il login. Dopo il login ci sono due extend: Upload e Download dei dati. Si è pensato a questa rappresentazione per marcare l'importanza del login. I dati sono anonimizzati, è vero, ma si tratta sempre di dati biomedici, ossia di dati estremamente preziosi, quindi è importante notare come solo attraverso il caso d'uso del login si può accedere al download o all'upload. C'è un unico attore, ossia il ricercatore. In realtà nelle prime fasi di progettazione lo Use Case era leggermente diverso, come mostrato in figura 3.6 La prima cosa che salta all'occhio è un ulteriore extend, poiché si pensava di utilizzare un'altra finestra per la selezione dei parametri. Si è poi optato per mantenere tutto in un'unica finestra per ragioni di semplicità di programmazione e di facilità di utilizzo, dal momento che il form usato è coerente con la GUI, ma di questo si parlerà nelle sezioni successive. L'altra particolarità che salta all'occhio è la suddivisione tra ricercatore "client" e ricercatore "server". Inizialmente si è pensato di mantenere

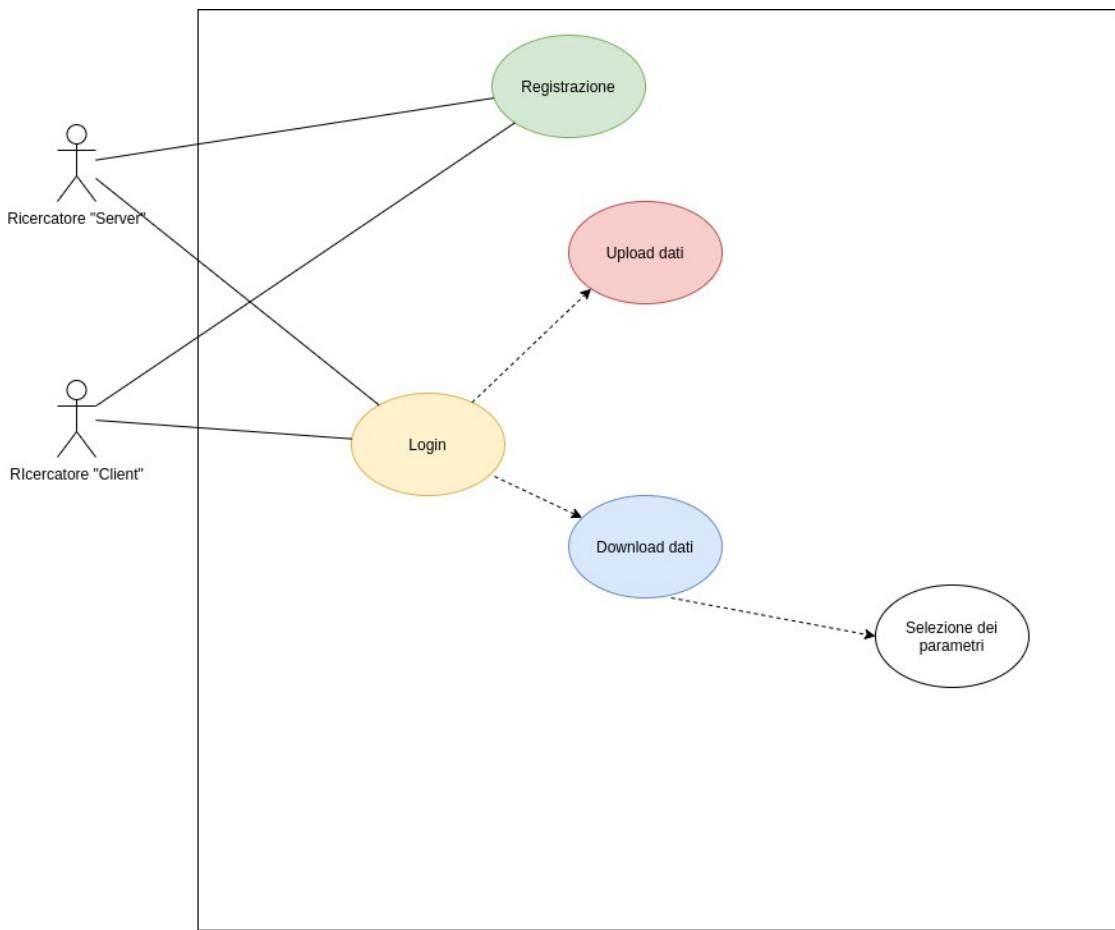


Figura 3.6: Primo Use Case del progetto SleepingRepo.

le due figure separate nel sistema informativo in modo da avere la possibilità di pensare a funzioni differenti a seconda dell'utente. Si specifica che le diciture "client" e "server" sono usi un po' impropri e colloquiali per indicare un ricercatore che carica i dati (server, perché offre un servizio) e un ricercatore che usufruisce dei dati, ossia un ricercatore "client", perché è cliente del servizio. Ad ogni modo durante la realizzazione non sono stati trovati validi motivi per tenere separate le due figure, perché esistono solo due "categorie" di utenti all'interno del portale: chi ha confermato l'indirizzo email e chi no.

3.2 Le comunicazioni

Un elemento importante affinché il sistema sia distribuito e fruibile da più luoghi è senza dubbio il sistema di comunicazione utilizzato. La prima caratteristica è

importante per rendere il sistema più modulare: se si volesse cambiare la struttura del portale si può rifarne una da capo e lasciare inalterato il middleware e il database. D'altro canto, se si volesse tagliare fuori il database e dirottare tutti i dati verso un altro sistema non ci sarebbe alcun ostacolo, basterebbe specificare il nuovo url ed essere sicuri della compatibilità con FHIR. La seconda caratteristica è infine quella che da senso all'idea, ossia la possibilità, per chiunque voglia, di caricare i dati senza la limitazione del doverli consegnare in loco. Per concettualizzare questi due obiettivi si è dovuto pensare prima alla struttura del sistema, rappresentata dal deployment diagram. Infine si è pensato alla sequenza che deve effettivamente essere programmata usando il Sequence Diagram.

3.2.1 Deployment diagram

La teoria del deployment diagram si è vista nel capitolo 2. In particolare, si ricordi come il deployment diagram abbia diversi usi a seconda di cosa si vuole raffigurare, a tal punto che ne esiste una versione "derivata" chiamata "Network Architecture Diagram". In questo caso l'architettura di rete è una "Three-tier architecture", ossia una architettura formata da tre strati. Il primo è il livello del client, ossia il dispositivo che richiede il servizio e su cui funzioneranno i codici HTML, CSS e Javascript in simultanea. Tale dispositivo può essere un qualsiasi dispositivo che ha un browser. Il secondo strato è quello dell'applicazione, ossia del web server, e infine c'è il "data tier", ossia la parte del database.

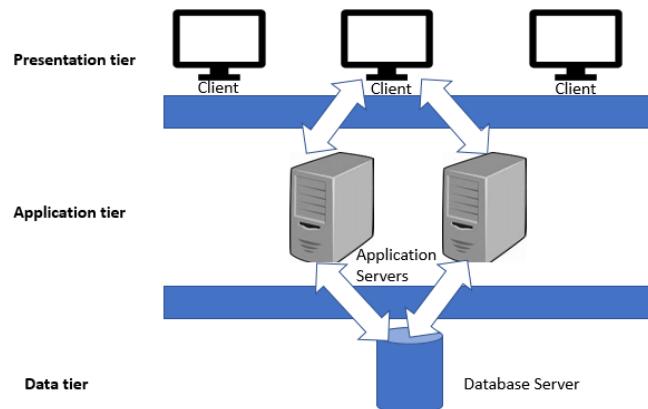


Figura 3.7: Architettura three-tier.

Partendo da questa architettura si è quindi sviluppato il deployment diagram in figura 3.8. Come si può notare è stato modellato il desktop client in modo generico. All'interno del client c'è il web browser. Questo comunica tramite HTTPS con il

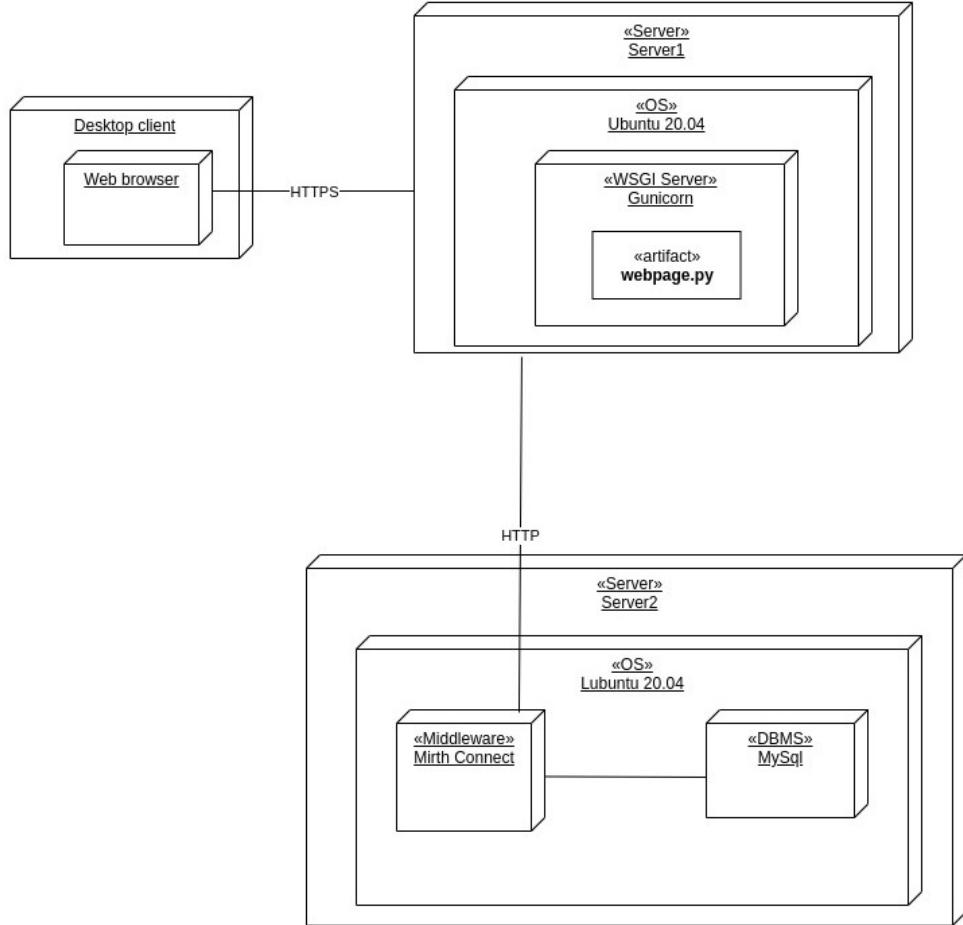


Figura 3.8: Deployment Diagram del progetto SleepingRepo.

web server chiamato "Server1". Nello specifico è stato utilizzato un server con una distribuzione Linux, ossia Ubuntu 20.04, come si vede nella figura 3.9.

Si noti che il quantitativo di RAM è in realtà inferiore poiché per fare le simulazioni con un unico dispositivo si è usata una macchina virtuale. All'interno di Ubuntu si è pensato di inserire un WSGI server⁵ Gunicorn, poiché permette di far funzionare applicazioni scritte con Flask molto facilmente. In realtà nelle fasi di debug non è stato utilizzato Gunicorn, ma va incluso nella documentazione dal momento che diventa importante in fase di installazione. Ad ogni modo, all'interno

⁵Per WSGI, acronimo per "Web Server Gateway Interface", si intende un protocollo di trasmissione che descrive comunicazioni e interazioni tra applicazioni web e server scritte in Python. Di fatto è l'interfaccia web standard dei servizi web scritti in Python.

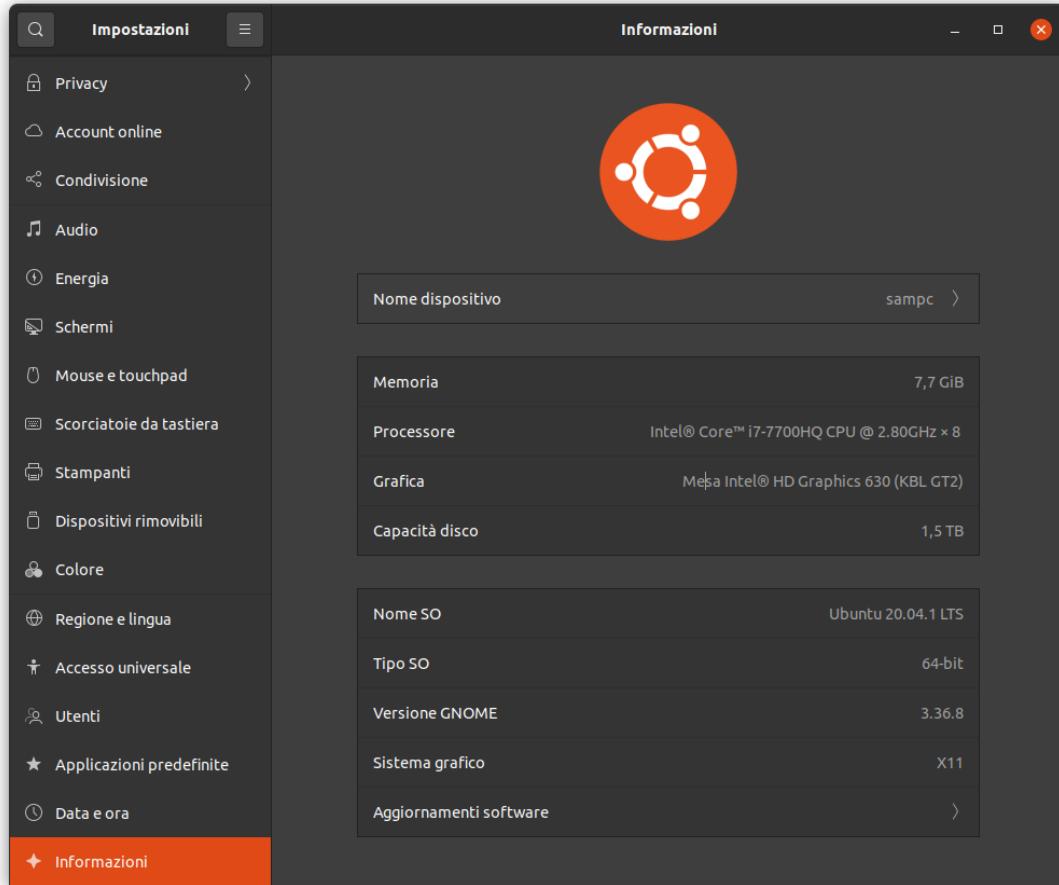


Figura 3.9: Specifiche della macchina usata per il web server.

di Gunicorn viene fatto funzionare lo script Python "webpage.py", ossia lo script principale contenente l'applicazione scritta con Flask.

Dopo il web server viene il server che tiene il database. Come si è accennato in precedenza si è simulato questo server usando una macchina virtuale costruita nell'ambiente "Oracle VM Virtual Box". Uno dei motivi per cui è stato usato è stato la maggiore semplicità nel debuggare il sistema in modo che non si debbano usare più dispositivi. Si è usato il protocollo HTTP perché non è possibile scaricare l'estensione Mirth che supporta SSL gratuitamente. Al momento, però, la comunicazione tra web server e middleware è sempre avvenuta tramite LAN. In fase di implementazione si potrebbe creare o una sotto rete o una VPN⁶ in cui giace il

⁶Una VPN (Virtual Private Network), è una rete, come dice il nome, privata. Viene utilizzata per creare delle reti private all'interno di reti più grandi pubbliche in modo da proteggere i

sistema in modo che solo il web server possa comunicare con il middleware. Come sistema operativo scelto si ha Lubuntu 20.04. Si è scelto Lubuntu perché è di fatto Ubuntu ma progettato per essere più "leggero" (da qui la lettera "L"), quindi più adatto per hardware ridotti. In particolare la memoria RAM, come si accennava poco sopra, ammonta a 3 GB, come si vede in figura 3.10, lasciando ad Ubuntu meno di 5 GB di RAM.

Dettagli di configurazione		Informazioni di esecuzione	Monitor delle prestazioni	Controllo del guest
	Generale			
Nome	Tesi			
Sistema operativo	Ubuntu (64-bit)			
	Sistema			
Memoria di base	3072 MB			
Ordine di avvio	Floppy, Ottico, Disco fisso			
Accelerazione	VT-x/AMD-V, Paginazione nidificata, Paravirtualizzazione KVM			
	Schermo			
Memoria video	16 MB			
Scheda grafica	VMSVGA			
Server di desktop remoto	Disabilitato			
Registrazione	Disabilitata			
	Archiviazione			
Controller: IDE				
IDE master secondario	[Lettore ottico] Vuoto			
Controller: SATA				
Porta SATA 0	Tesi.vdi (Normale, 50,00 GB)			
	Audio			
Driver host	PulseAudio			
Controller	ICH AC97			

Figura 3.10: Configurazione della macchina virtuale usata.

Sempre in figura 3.10 viene segnalato che il sistema operativo è Ubuntu, poiché Lubuntu condivide, di fatto, la stessa matrice. Le informazioni sul sistema operativo sono in figura 3.11.

dispositivi all'interno.

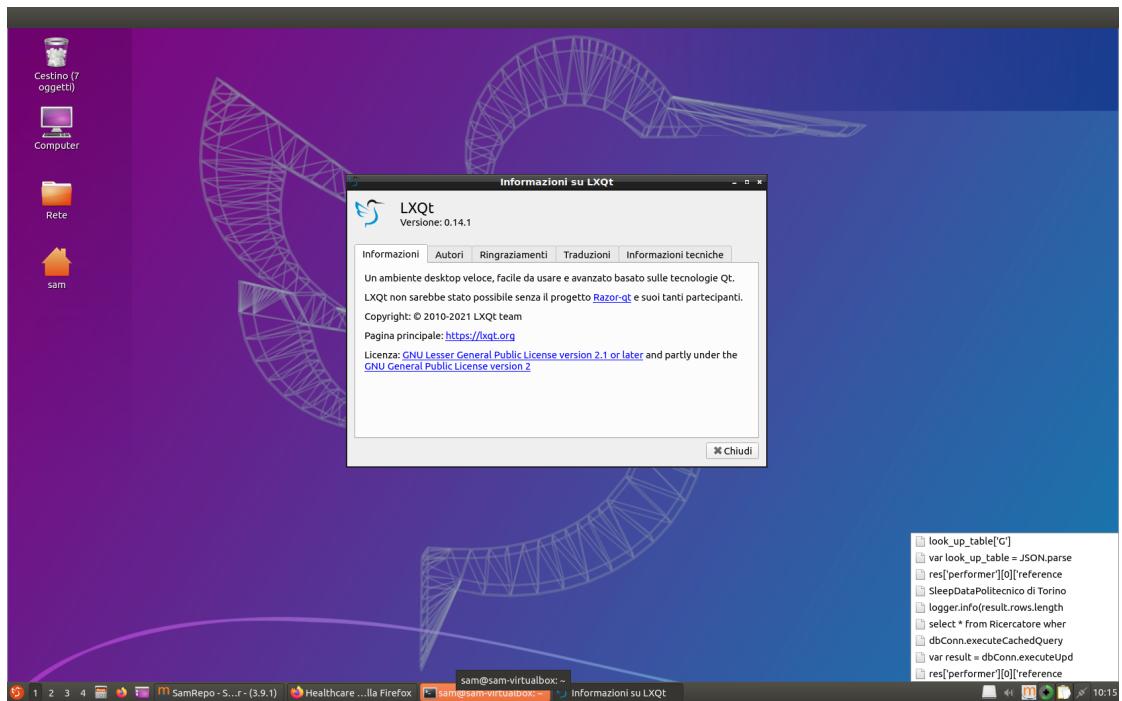


Figura 3.11: Versione di Lubuntu.

Oltre il sistema operativo c'è il Middleware Mirth Connect, alla versione riportata in figura 3.12.



Figura 3.12: Specifiche Mirth Connect.

Infine all'interno della stessa macchina Mirth comunica in *localhost* con il DBMS scelto, ossia MySql.

3.2.2 Sequence diagram

Una volta concettualizzata l'architettura generale si deve ritornare a pensare al funzionamento dei processi. In particolare, si è cominciato a pensare ai messaggi che sono stati effettivamente scambiati tra gli attori. UML, come si è già visto, mette a disposizione uno strumento particolarmente utile, ossia il sequence diagram. Sono stati pensati e modellati due sequence diagram, riducendone il numero rispetto a quanto visto nello Use Case in precedenza. Si è deciso di concentrarsi su questi due perché sono i veri processi che si vogliono modellare e anche perché il caso d'uso del login, essendo più semplice, è stato inserito nei suoi extend. Il caso d'uso della registrazione è molto simile all'upload dei dati dal momento che si manda in ogni caso una risorsa FHIR.

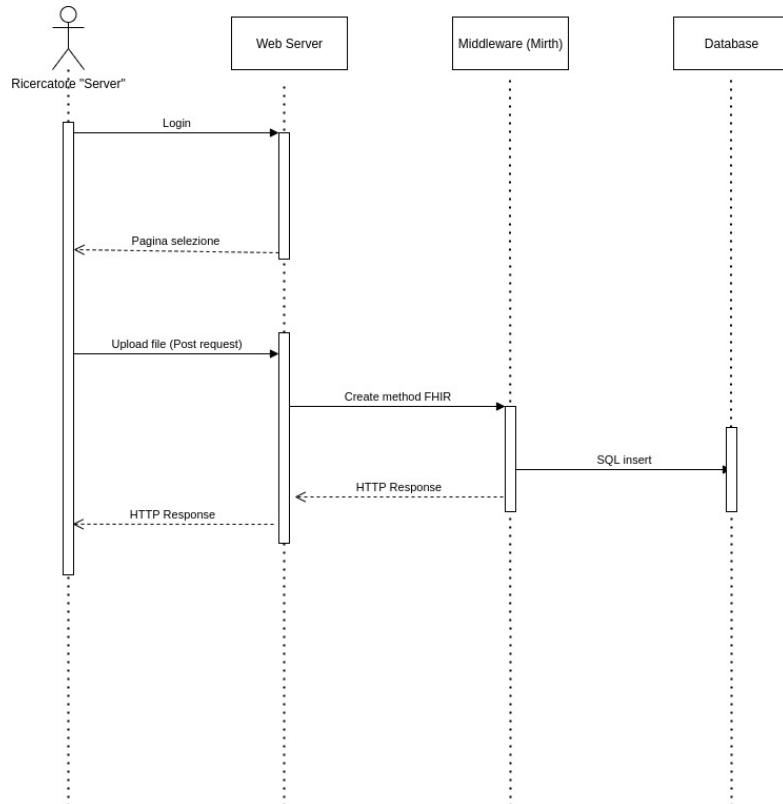


Figura 3.13: Sequence diagram del caso d'uso di upload.

Come si vede nella Figura 3.13 è stata lasciata la vecchia dicitura "Ricercatore server" per rimarcare il ruolo dell'utente nel processo anche se in realtà non c'è

una reale differenza. In questo punto vengono coinvolti tre attori software, ossia il web server il middleware e il database. Il web server non è altro che il codice scritto in Python che si occupa inizialmente dell'autenticazione dell'utente, mentre nell'attore umano è inclusa l'interfaccia web. Una volta autenticato il ricercatore carica i dati tramite una richiesta post. La pagina web a questo punto tenta un post verso Mirth, secondo lo standard FHIR, usando di fatto un CREATE. Si sarebbe potuto utilizzare UPDATE, in realtà, ma siccome si stanno caricando dei dati che si presuppone siano nuovi, non essendo fattibile controllare che quel file già esista identico, è più corretto usare il metodo create, specifico per la creazione di una nuova risorsa. Infine il middleware esegue un comando INSERT sul database. Nel caso di

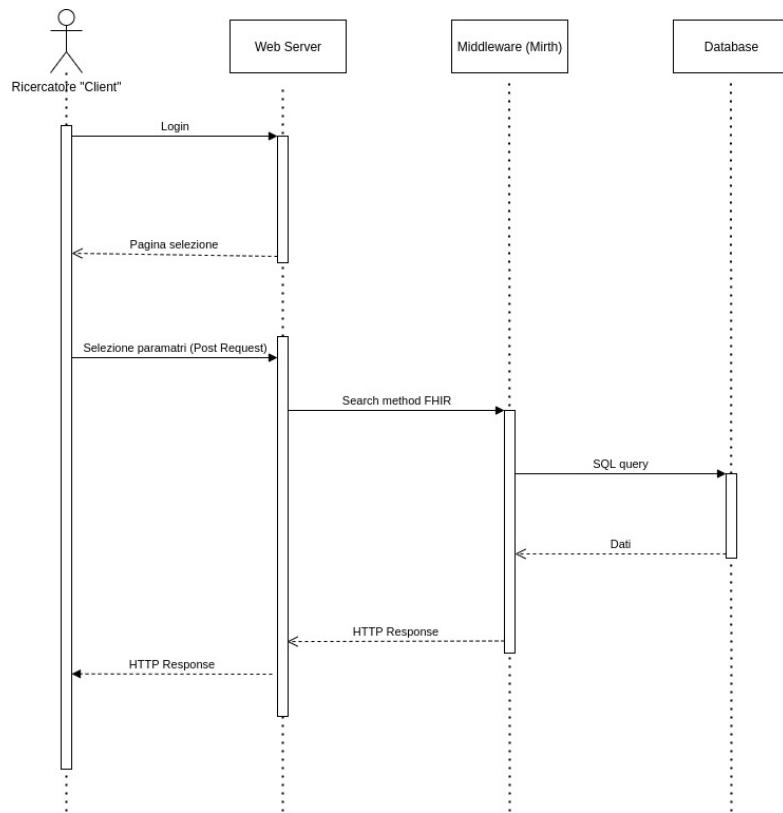


Figura 3.14: Sequence diagram del caso d'uso di download.

download il ricercatore fa login come nel caso d'uso precedente (infatti nello use case download e upload sono suoi extend). Una volta effettuata l'operazione avviene l'invio dei parametri selezionati tramite POST. Questo metodo concettualmente potrebbe sembrare scorretto, infatti sarebbe più appropriato GET, ma è una piccola misura di sicurezza, così come viene consigliato dal W3C e dalla documentazione di FHIR. Se si effettua GET, infatti, i parametri della richiesta appaiono nella barra di ricerca del browser poiché mostrerà tutto l'URL a cui sta facendo la richiesta e può

essere intercettato, oltre al fatto che resta salvato nella cronologia e nei segnalibri del browser. Per mantenere in visione la minor quantità possibile di dati si è preferito utilizzare POST. Il web server a sua volta effettua un SEARCH verso il middleware. In realtà in alcuni casi si sarebbe potuto pensare all'utilizzo di READ, in particolare quando si vuole filtrare usando l'id della risorsa. Come si vedrà più avanti, però, l'uso dell'ID è relativamente complicato, per quanto si è cercato di semplificare al massimo. Di conseguenza il filtraggio con quel parametro è visto come un evento raro, quindi non è stato implementato alcun metodo specifico, perché in ogni caso SEARCH funziona comunque.

3.3 Il database

Una volta che i server possono comunicare, si può pensare a organizzare la base di dati. Come si è già detto si è scelto MySql al fine di avere un database relazionale che supportasse SQL. Una volta installato si è creato il database "SleepingRepo" per poi iniziare a modellare il contesto facendo prima il diagramma E-R completo e poi andandolo a ristrutturare fino ad ottenere le tabelle costitutive del database. Solo a quel punto è stato possibile inserirle nel DBMS.

3.3.1 Il diagramma E-R

Per l'entità-relazione si è ragionato un po' più in grande nell'ottica di espandere il sistema. Nelle possibili espansioni del sistema si può considerare l'aumento del numero di tipi di dati compatibili oppure l'introduzione del sistema anche in un contesto clinico. Nella Figura 3.15 si possono notare quattro entità: ricercatore, medico, paziente e misurazione. È possibile notare due generalizzazioni: una è "operatore" che generalizza il medico e il ricercatore, mentre l'altra è "persona" e generalizza il paziente e l'operatore. Il ricercatore e il medico hanno infatti due attributi in comune, ossia quello chiamato "istituto" che è la struttura ospedaliera o ente di ricerca a cui afferiscono (si consideri che talvolta una azienda ospedaliera può essere anche un centro di ricerca) e lo username, che viene posto come identificatore dell'entità. Inoltre condividono pure una relazione con l'entità misurazione. Tale relazione è chiamata "acquisisce", perché sia il ricercatore che il medico possono acquisire delle misurazioni durante la notte. La cardinalità è "0,N", perché un operatore può non aver ancora acquisito niente e essere momentaneamente solo un frutto del servizio, così come può aver caricato molte misurazioni. La principale differenza consiste nella relazione che il medico ha con il paziente, chiamata "cura": un medico può seguire un paziente, mentre il ricercatore no. La cardinalità di questa relazione per il medico è "0,N" perché un medico può avere più di un paziente, ma può anche non averne nessuno da seguire per il momento. Operatore e paziente sono generalizzati in "persona" che contiene i dati anagrafici. In figura

sono stati inseriti solo nome, cognome e email, ma il numero di dati inseribili come attributi può variare senza alcun problema, magari mettendo la non obbligatorietà del dato (cardinalità "0,1"). Il paziente che ha un solo attributo specifico, ossia il suo identificativo. È stato scritto come "IdPaziente", in modo un po' generico perché questa "espansione" non è stata pensata nelle precedenti fasi di progettazione. L'entità paziente partecipa alla relazione "cura" con il medico con una cardinalità di "1,1", ossia un paziente ha uno e un solo medico nel database. Si è fatta questa assunzione considerando che il paziente faccia riferimento in modo generico al proprio medico di base (che non deve essere per forza chi acquisisce i dati). Si tenga in considerazione comunque che senza una analisi dei requisiti e un contesto stabilito questi dettagli sono soggetti a variazioni, pur non impattando in maniera massiva sul risultato finale del diagramma E-R che resta strutturalmente immutato. La seconda relazione del paziente è con la misurazione e viene chiamata "ha", poiché il paziente *ha* una misurazione. La cardinalità del paziente nella relazione è "0,N" perché un paziente potrebbe non avere ancora misurazioni salvate, così come potrebbe averne pure molte.

Infine si arriva alla misurazione. Questa entità ha molti attributi corrispondenti ai segnali acquisiti:

- Impedenza, ossia l'impedenza transtoracica, come già detto nel capitolo 1 serve ad acquisire la frequenza respiratoria.
- SpO₂, è la saturazione periferica dell'ossigeno. Il principio di funzionamento del saturimetro è stato spiegato nel capitolo 1.
- Piedi. Con questo attributo si intendono i segnali di movimento degli arti inferiori ottenuti tramite estensimetri e sensori inerziali come giroscopi e accelerometri.
- EOG, per valutare il movimento degli occhi.
- ECG, per valutare l'attività elettrica del cuore.
- EEG, per valutare l'attività cerebrale, nonché metodo principe per capire in che fase del sonno è il paziente.
- EMG, principalmente misurato sotto il muscolo milioideo, per valutare, ad esempio, fenomeni di bruxismo. In realtà è stato messo un generico EMG perché nulla vieta di sperimentare acquisendo EMG da altri muscoli al fine di valutare l'atonia muscolare e l'attivazione delle fasce muscolari durante il sonno.
- Flusso Nasale. Con un termistore si va a valutare la frequenza respiratoria, come detto nel capitolo 1.
- Baricentro. Per baricentro si intendono i dati dai sensori inerziali che indicano il movimento del paziente durante la notte, per cui si indica il baricentro.

Infine c'è la data e l'ID, che insieme formano la chiave, in modo da rendere praticamente impossibile avere due ID identici, soprattutto in relazione al metodo di generazione dell'ID all'interno del sistema, come si vedrà in seguito. L'entità misurazione partecipa a due relazioni: la prima è quella con il paziente e ha una cardinalità "0,1" perché un dato può essere pure anonimo, ma non può avere più di un paziente. L'altra è quella con l'operatore e partecipa con una cardinalità "1,1" perché una misurazione deve essere pur fatta da qualcuno.

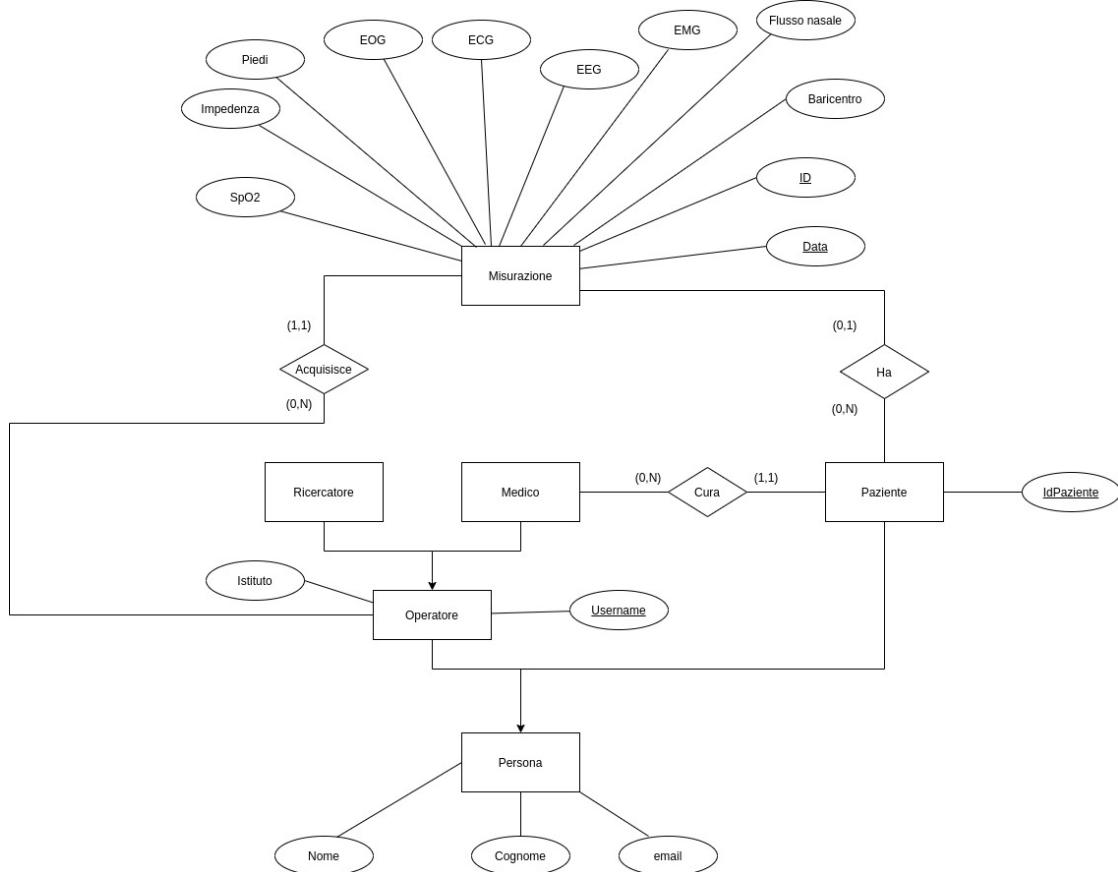


Figura 3.15: Diagramma entità-relazione con l'espansione.

Al termine di questa idealizzazione, in funzione di una possibile espansione, ci si è ricavati il diagramma E-R "reale" come caso particolare di questo più generale. Ad un occhio attento, in ogni caso, il diagramma non risulta corretto, avendo delle generalizzazioni con un'unica entità figlia. Un E-R di questo tipo ha più uno scopo esplicativo per mantenere il parallelismo fino alla ristrutturazione del modello concettuale.

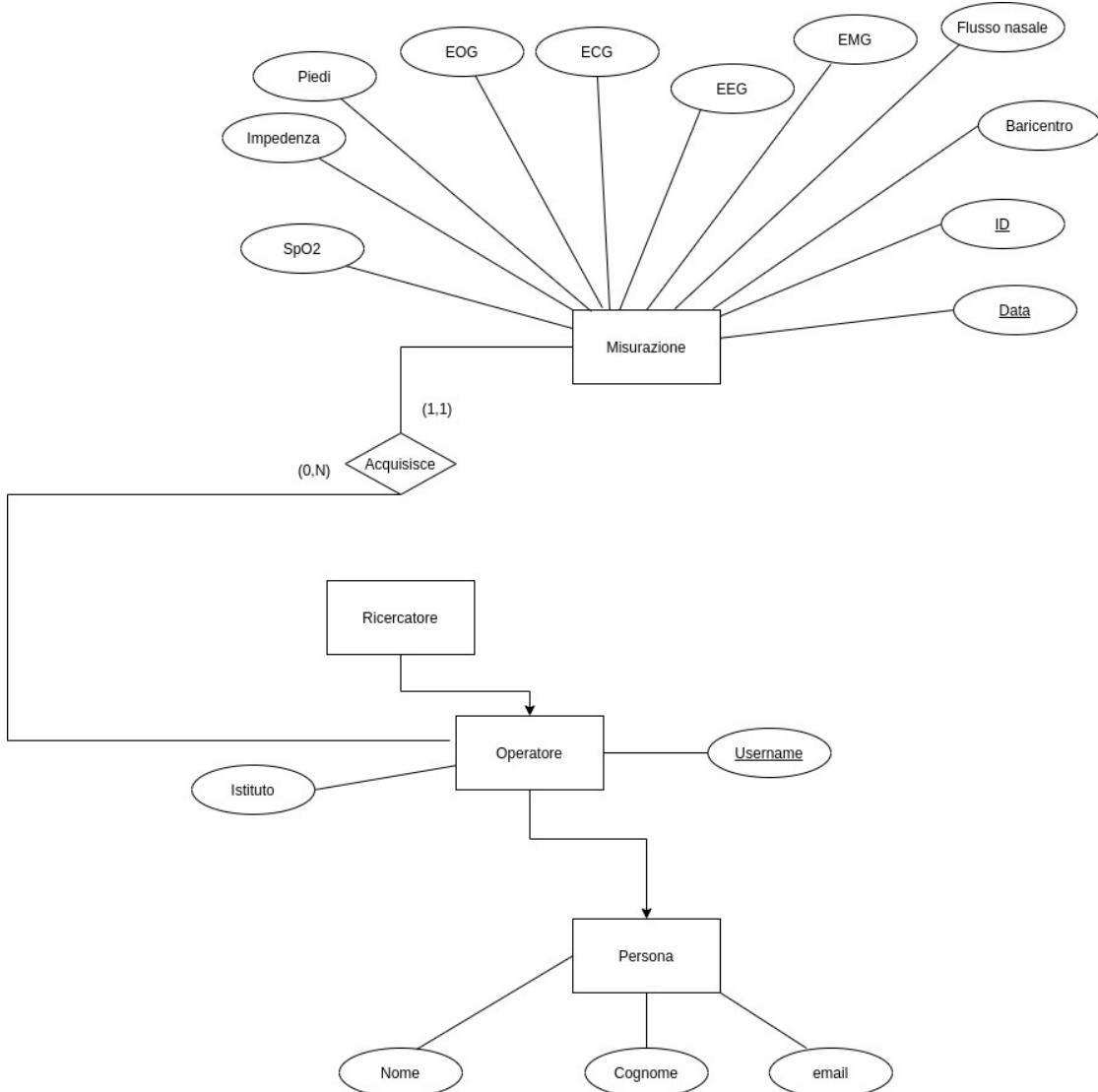


Figura 3.16: Diagramma entità-relazione del progetto SleepingRepo.

3.3.2 Verso il modello logico

Come si è detto nel capitolo 2, la rappresentazione logica di un database non viene rappresentata direttamente nel DBMS. Infatti si è già visto che in DBMS relazionale vanno inserite delle "tabelle" denominate "relazioni". Diagrammi E-R come quelli visti in precedenza non sono inseribili, per cui bisogna ristrutturare il diagramma E-R.

La prima operazione da fare in questo caso è analizzare le ridondanze, ossia cercare tutti quei dati che possono essere derivati da altri. Nel caso in analisi non ce ne

sono. Il secondo passo è eliminare le gerarchie di generalizzazione, che in questo caso sono ben presenti. Ci sono vari modi per eliminarle, infatti si vedranno due soluzioni diverse per le due generalizzazioni.

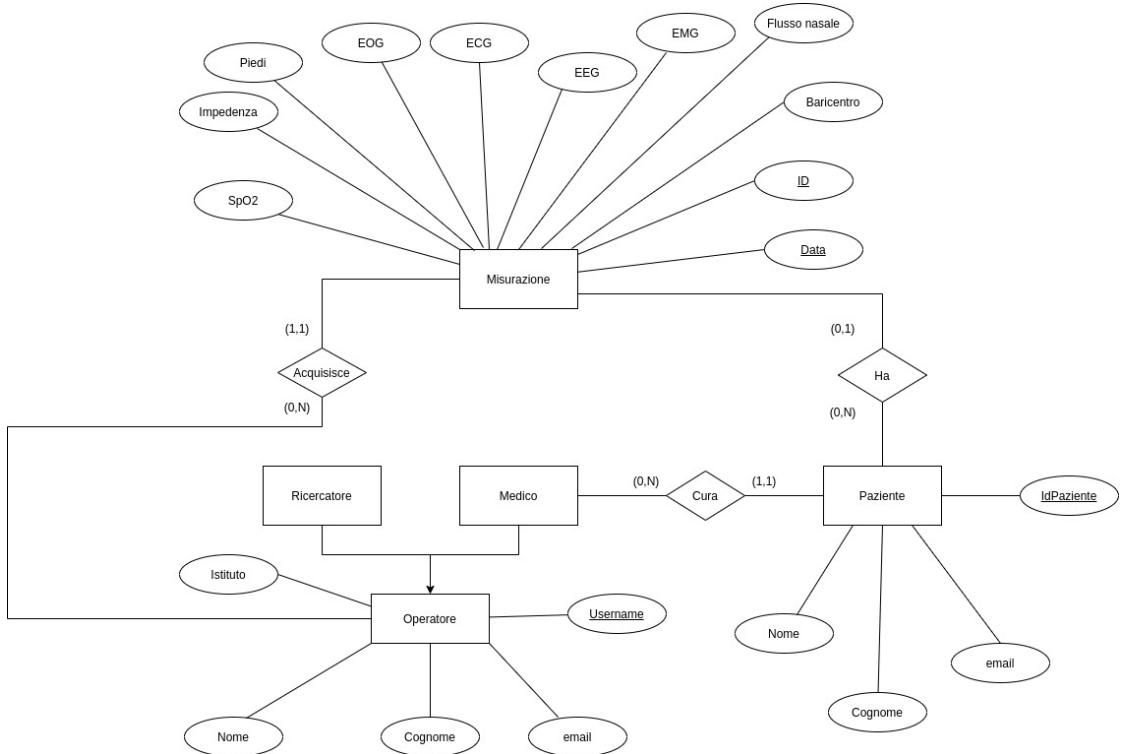


Figura 3.17: Eliminazione della prima generalizzazione.

Come si vede in Figura 3.17 la generalizzazione persona è stata eliminata accorpandola nelle due entità figlie, ossia operatore e paziente. In questo modo si è costretti a dare degli attributi identici a due entità.

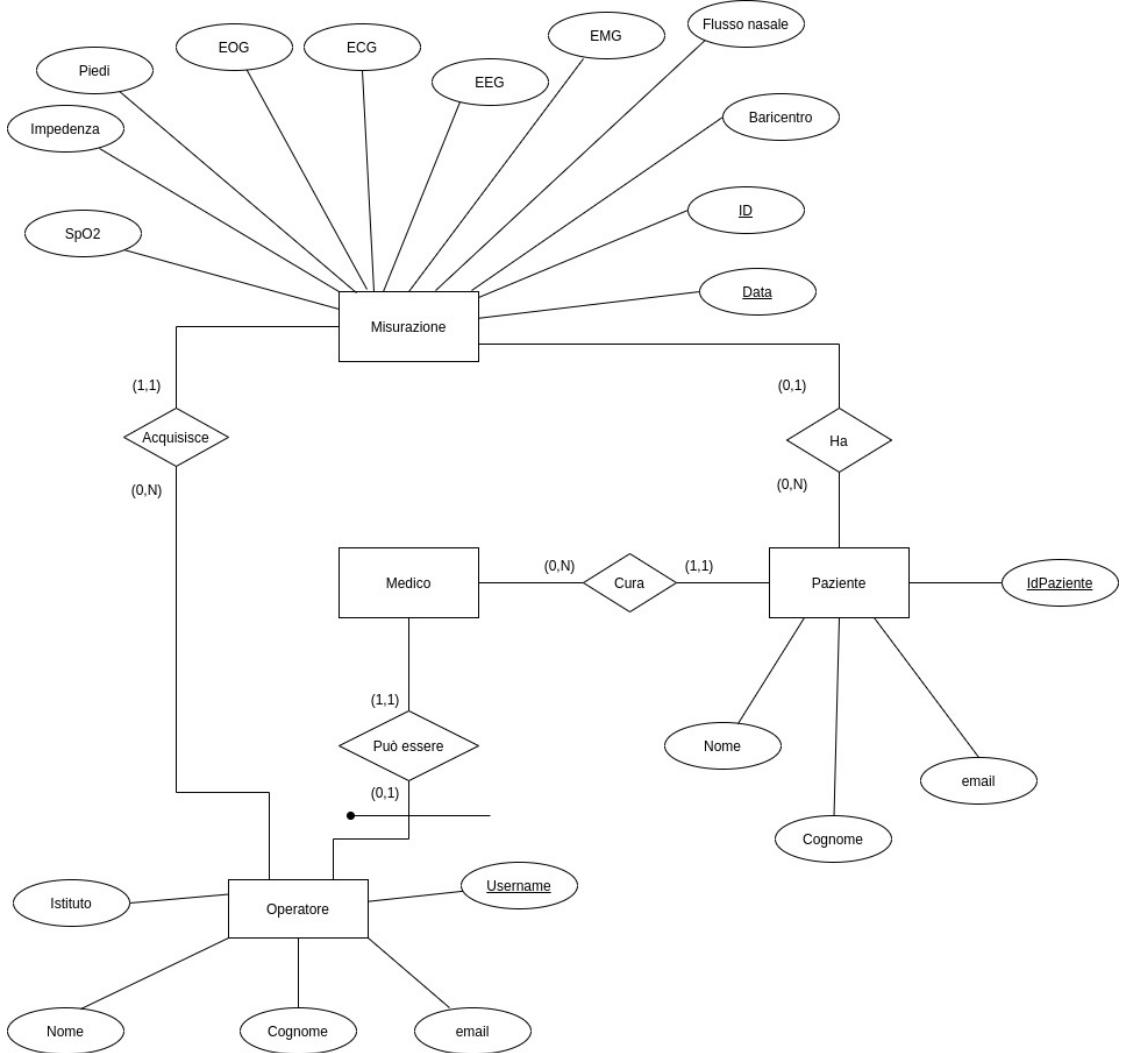


Figura 3.18: Eliminazione della seconda generalizzazione.

Lo step successivo è eliminare la generalizzazione "operatore". Il metodo utilizzato è quello di sostituire la generalizzazione con delle relazioni, dato che dividere Ricercatore e Medico sarebbe deleterio, condividendo una relazione. Accorpate entrambe le entità in operatore sarebbe comunque problematico perché si darebbe la relazione "cura" al ricercatore, portando a una perdita di confini dei ruoli. Di conseguenza la generalizzazione diventa relazione. Il passaggio successivo, già visibile in Figura 3.18, è l'accorpamento delle entità. Se è vero che Ricercatore deve rimanere separato da Medico è anche vero che rispetto alla neonata entità "Operatore" l'entità Ricercatore non ha nulla di diverso, quindi Ricercatore può essere inglobato da Operatore. A questo punto, però, l'entità Medico non ha né attributi, né chiavi, che sono rimasti con Operatore. Di conseguenza per identificare

univocamente un medico bisogna assumere come chiave la relazione "può essere", dal momento che ha cardinalità "1,1". Infatti un medico è anche un operatore, ma, parlando di persone, un medico può essere uno e un solo operatore. Di contro un operatore può essere un medico ma può anche essere un ricercatore, quindi la cardinalità è "0,1".

Paziente		Operatore		Misurazione	
PK	<u>IdPaziente</u>	PK	<u>username</u>	PK	ID
FK	Nome Medico (Operatore) Cognome email	FK	Nome Cognome email Istituto	FK	Data
					Operatore
					Paziente
					ECG
					EEG
					SpO2
					Impedenza
					Baricentro
					EMG
					Piedi
					Flusso Nasale
					EOG

Figura 3.19: Modello logico ampliato.

Ora che il modello E-R è ristrutturato bisogna tradurre verso il modello logico. Sono state individuate tre relazioni (nel senso logico, cioè delle tabelle) e sono: Paziente, Operatore e Misurazione.

La prima è l'entità Paziente con le stesse chiavi e gli stessi attributi con in più una *foreign key*⁷, in riferimento al medico che lo cura, come traduzione della relazione "cura" del modello E-R. L'entità Medico è stata eliminata, perché non ha attributi propri, dato che la sua chiave stava tutta nella relazione. L'entità Operatore non ha foreign key, ed è quindi la traduzione dell'omonima entità. La tabella Misurazione corrisponde alla rispettiva entità ma ha ben due foreign key: Operatore e Paziente. La prima fa riferimento alla relazione E-R "acquisisce", mentre la seconda alla relazione E-R "ha".

A questo punto resta la normalizzazione delle relazioni. Viene definita "prima

⁷Un foreign key, cioè una chiave esterna, è un vincolo di integrità referenziale tra due tabelle, ossia stabilisce un collegamento tra le tuple di due tabelle.

"forma normale" una relazione che non contiene gruppi ripetitivi, ovvero quando è strutturata come una raccolta di tuple e attributi. In pratica una tabella per essere in forma normale deve rispettare i requisiti fondamentali del modello relazionale, cioè:

- Tutte le righe devono contenere lo stesso numero di colonne;
- Gli attributi devono rappresentare informazioni elementari;
- I valori di una colonna devono essere tutti dello stesso tipo;
- Ogni riga deve essere diversa dalle altre;
- L'ordine delle righe non è importante.

Ne segue che la prima forma normale è rispettata.

La seconda forma normale necessita che la prima forma normale sia rispettata, più che ogni attributo che non sia chiave dipenda pienamente dalla chiave. Si immagini, come esempio, di avere una tabella che ha come chiave il modello di smartphone di una persona e il relativo operatore. Un eventuale attributo relativo al costo delle promozione mensile dipenderebbe solo dall'operatore e non dal modello di smartphone, per cui la seconda forma normale non sarebbe rispettata. Nel caso in esame, Misurazione ha due chiavi, ma gli altri suoi attributi sono solo i nomi del paziente e dell'operatore e i segnali che non dipendono da una sola delle chiavi, anzi, i segnali non hanno alcuna dipendenza diretta dalle esse, ma sono quest'ultime a "nominare" il set di segnali che altrimenti non sarebbero identificabili.

La terza forma normale richiede che sia rispettata la seconda forma normale e che ogni attributo non chiave dipenda direttamente dalla chiave e non da altri attributi che a loro volta dipendono dalla chiave (ossia non deve esserci una dipendenza transitiva). Si consideri come esempio una tabella rappresentante un impiegato. Si immagini il cognome come chiave e si immagini di avere, tra gli altri, due attributi: "categoria" e "stipendio". Avendo una sola chiave non ci sono dipendenze parziali, ma l'attributo "stipendio" dipende da "categoria", che non è una chiave, quindi si viola la terza forma normale. Quest'ultima, nel progetto di tesi, invece, è pienamente soddisfatta.

L'ultima forma normale da rispettare è quella di Boyce-Codd che stabilisce che per ogni dipendenza funzionale da X ad A, X è una chiave della relazione. In generale, quindi, se una relazione soddisfa Boyce-Codd soddisfa pure la terza forma normale, ma il contrario non è sempre vero perché Boyce-Codd richiede che la condizione sia soddisfatta per ogni chiave possibile, non solo in funzione di quella scelta.

Questo lavoro di ristrutturazione è stato effettuato allo stesso modo con la versione "reale" del progetto e il modello E-R è quello in Figura 3.20.

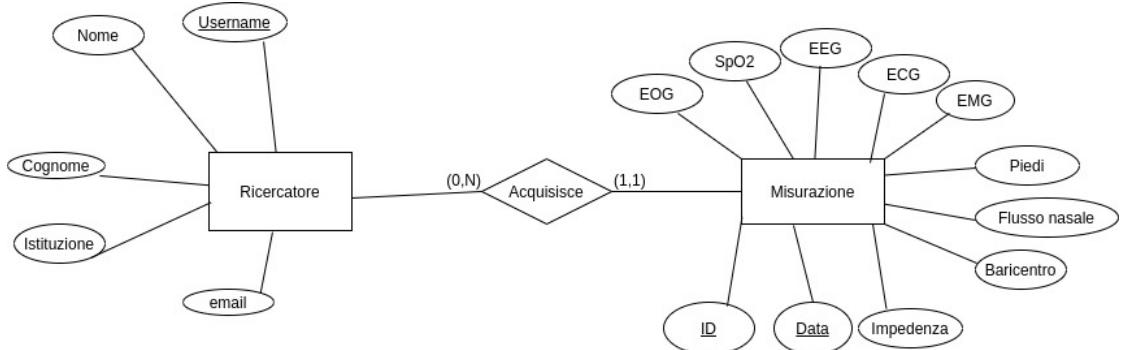


Figura 3.20: Diagramma E-R ristrutturato del progetto SleepingRepo.

Si noti come alla fine il diagramma E-R effettivamente utilizzato non è altro che un caso particolare dell'E-R "espanso", dove al posto del nome "Operatore" è stato usato un termine più significativo in questo contesto, ossia "Ricercatore". Inoltre manca tutto ciò che è correlato al medico e al paziente. Quindi di fatto sono state accorpate tutte le generalizzazioni sull'unica entità figlia.

mysql> Describe Misurazione;						
Field	Type	Null	Key	Default	Extra	
Id	char(40)	NO	PRI	NULL		
Data	date	NO	PRI	NULL		
UserID	char(30)	YES	MUL	NULL		
EEG	text	YES		NULL		
EKG	text	YES		NULL		
EOG	text	YES		NULL		
EMG	text	YES		NULL		
SpO2	text	YES		NULL		
Centre_Of_Gravity	text	YES		NULL		
Feet	text	YES		NULL		
Nasal_Flow	text	YES		NULL		
Impedence	text	YES		NULL		

12 rows in set (0.00 sec)

Figura 3.21: Relazione "Misurazione" inserita sul DBMS mysql.

Come si può vedere dalle Figure 3.21 e 3.22, le tabelle che escono fuori nel caso applicato sono di fatto identiche a quelle che si avevano nel caso completo. L'unica differenza consiste nella mancanza di una foreign key in misurazione, ossia quella con riferimento al paziente. Non esistendo alcun paziente nel database sarebbe sempre "null".

mysql> Describe Ricercatore;						
Field	Type	Null	Key	Default	Extra	
ID	char(30)	NO	PRI	NULL		
Nome	char(30)	YES		NULL		
Cognome	char(30)	YES		NULL		
Istituzione	char(70)	YES		NULL		
Email	char(50)	YES		NULL		

Figura 3.22: Relazione "Ricercatore" inserita sul DBMS mysql.

Bisogna, infine, soffermarsi sull'implementazione nel DBMS. La prima cosa da notare è la dicitura "YES" nella casella "NULL" per l'attributo "UserID", ossia il riferimento al ricercatore. Si è pensato di concedergli la capacità di essere "null" in funzione della possibilità di "innestare" sul middleware qualsiasi tipo di portarle in future configurazioni, per cui, se è vero che con il portale attualmente in uso è impossibile che quel campo sia vuoto, non è altrettanto vero si si dovesse cambiare idea. Per fronteggiare un eventuale "null" e prendere i dati comunque si è deciso di inserire un ricercatore "null" in modo da assegnare a quella tupla jolly le eventuali misurazioni "orfane". Si noti, infine, che i tipi degli attributi sono tutte stringhe, ma nello specifico ci sono attributi che hanno un numero di caratteri limitati, per risparmiare memoria, mentre i segnali sono "text". Il tipo "text" occupa più memoria ma tiene molti più caratteri. È stata presa questa decisione perché il segnale viene salvato nel file system mentre nel database viene lasciato il percorso per recuperarlo, ma, siccome non si sa su quale file system venga poi installato il sistema, non è nota la lunghezza del percorso fino al file.

3.4 Le classi

Dopo il database è stato necessario pensare alle classi che devono essere presenti nel sistema, dal momento che programmare con un paradigma a oggetti può aiutare a rendere il software più mantenibile. Come si è già visto dal deployment diagram, il sistema è composto da due "attori" hardware principali: il server che si occupa dei servizi web e il server che gestisce il database tramite il middleware. Come si è visto nel Capitolo 2, Mirth può essere impostato anche tramite interfaccia grafica e supporta l'utilizzo di script personalizzabili volti a portare a termine determinati compiti. Lato Mirth, non è quindi necessario creare delle classi. D'altro canto sono stati utilizzati degli oggetti presi da librerie Java, ad esempio la classe "bundle" di

FHIR. Nel web server, invece, si è programmato interamente in Python e non c'era nessuna GUI a guidare le impostazioni, quindi è stato utile programmare tutto ad oggetti. Di seguito verranno quindi proposte le classi del webserver scritte in Python e verranno spiegate tramite Class Diagram le risorse FHIR scelte e il loro template JSON.

3.4.1 Le risorse FHIR

Si è già visto come il fulcro di FHIR siano le risorse, che altro non sono che "astrazioni" della realtà. Questo, di fatto, è anche la definizione di classe in un paradigma a oggetti, infatti questo è il paradigma di FHIR. Non risulta quindi strano che le risorse FHIR vengano tratte come vere e proprie classi.

Nel progetto SleepingRepo vengono utilizzate tre risorse: "practitioner", "observation" e "bundle".

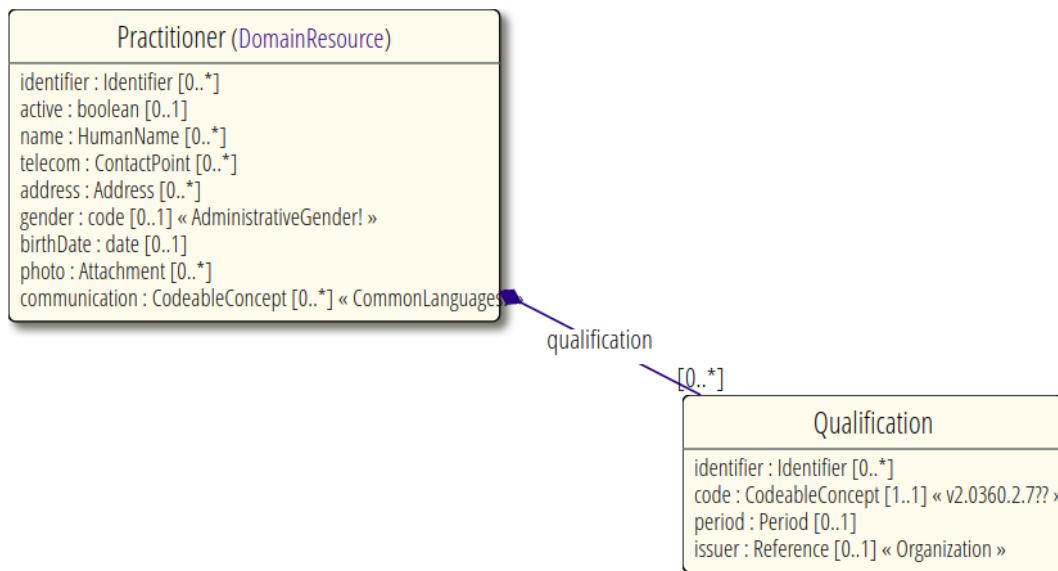


Figura 3.23: Class Diagram della risorsa Practitioner secondo FHIR.

La risorsa Practitioner⁸, rappresenta una persona che partecipa al processo di cura e ai servizi relativi. Un "Practitioner" può essere un medico, ad esempio, un infermiere, un tecnico di radiologia ma anche un impiegato che gestisce i record dei pazienti. Come si può vedere dalla Figura 3.23 la risorsa contiene essenzialmente attributi relativi ai dati anagrafici della persona. C'è da notare la relazione di

⁸In italiano "professionista"

composizione⁹ della risorsa "Qualification" che indica la qualifica professionale della risorsa. Una implementazione di practitioner è visibile nel template costruito per il progetto SleepingRepo.

```
1  {
2    "resourceType" : "Practitioner",
3    "identifier": [
4      {
5        "value":"uname"
6      }
7    ],
8    "name":
9    {
10       "family":"surname",
11       "given": "name"
12     },
13    "telecom":
14    {
15       "system":"email",
16       "value":"mail"
17     },
18    "qualification": []
19    {
20      "identifier": [
21        {
22          "value":"researcher"
23        }
24      ],
25      "code": {
26        "coding": [
27          {
28            "system": "http://terminology.hl7.org/CodeSystem/practitioner-role",
29            "code": "researcher",
30            "display": "Researcher"
31          }
32        ],
33        "issuer": {
34          "display": "Example University"
35        }
36      }
37    }
38  ]
39 }
40 }
41 }
42 }
```

Figura 3.24: Implementazione della risorsa Practitioner nel progetto SleepingRepo.

⁹In UML una composizione indica una classe che è inclusa in un'altra. Ad esempio la coda ha una composizione con il cane.

Come si può notare in Figura 3.24 non sono stati utilizzati tutti gli attributi, in quanto sarebbero risultati sovrabbondanti. Ciò non significa che in futuro non possano essere inseriti, ma al momento non sono necessari. C'è un identificativo che è lo username scelto dall'utente nel portale, c'è l'indirizzo email e c'è la qualifica. In particolare all'interno della chiave "code" nella lista di dizionari di "coding" c'è il ruolo "researcher", proprio perché, secondo la terminologia di FHIR, è tra le possibili scelte, come mostrato in Tabella 3.1[28].

Code	Display	Qualification
doctor	Doctor	Un medico professionista abilitato/registrato
nurse	Nurse	Un professionista con esperienza come infermiere che può essere abilitato/registrato
pharmacist	Pharmacist	Un farmacista abilitato/registrato
researcher	Researcher	Un professionista che può fare ricerche
teacher	Teacher	Qualcuno che può provvedere a servizi educativi
ict	ICT professional	Qualcuno che è qualificato nel mondo ICT

Tabella 3.1: Tabella che riassume i codici utilizzabili.

Un'altra risorsa cardine all'interno del sistema è quella che nel database veniva definita "Misurazione". Traducendo questo concetto in FHIR si incorre nella risorsa "Observation"¹⁰. Una "Observation" viene definita da FHIR come "misure e semplici asserzioni fatte a riguardo di un paziente, un dispositivo o un altro soggetto"[20]. In questa definizione rientra perfettamente la misurazione, anzi il concetto di "observation" è ben più ampio di quanto necessario.

¹⁰In italiano "osservazione"

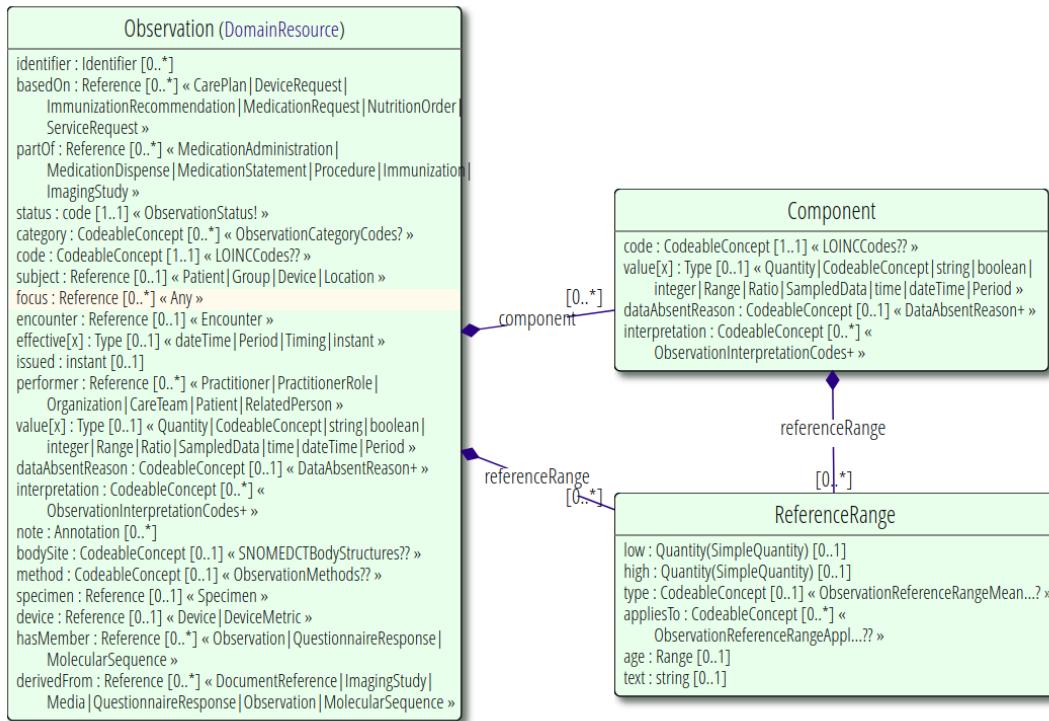


Figura 3.25: Class Diagram della risorsa Observation secondo FHIR.

La risorsa Observation, come si può vedere dalla figura 3.25 è ben più complessa di Practitioner perché contiene moltissimi campi in più opzionali. Inoltre è composta da due risorse: "Component" e "ReferenceRange". La prima contiene uno o più segnali nella misurazione, mentre la seconda il range di riferimento. Tra questi attributi di Observation ne sono stati scelti alcuni, ossia quelli che potessero essere generalmente utili, anche se possono essere null, in relazione alle informazioni che contiene la misurazione.

```
1  {
2      "resourceType": "Observation",
3      "identifier": [
4          {
5              "value": "SleepDataTemplate"
6          }
7      ],
8      "status": "final",
9      "code": {
10         "coding": [
11             {
12                 "system": "https://www.snomed.org/",
13                 "code": "60554003"
14             },
15             {
16                 "text": "Polysomnography"
17             }
18         ]
19     },
20     "effectiveDateTime": "2000-01-01",
21     "performer": [
22         {
23             "reference": "null"
24         }],
25     "component": [
26         {
27             "code": {
28                 "coding": [
29                     {
30                         "system": "https://www.snomed.org/",
31                         "code": "733919004",
32                         "display": "ch"
33                     },
34                     {
35                         "text": "Centre of gravity movement"
36                     }
37                 ]
38             },
39             "valueSampledData": {
40                 "origin": {
41                     "value": null,
42                     "unit": null
43                 },
44                 "period": null,
45                 "dimensions": null,
46                 "data": null
47             }
48         },
49     ],
50 }
```

Figura 3.26: Dettaglio dell'implementazione della risorsa Observation nel progetto SleepingRepo.

Come si vede nella figura 3.26 Vengono mantenuti alcuni elementi come il

riferimento al "Practitioner" (indicato con "performer"), la data, il codice e l'ID. Infine ci sono i vari componenti, che sono i segnali. La chiave "component" ha quindi come valore un array di altri piccoli JSON divisi in "code" e "valueSampledData". Il primo fa riferimento al sistema di codifica usato, mentre il secondo al segnale vero e proprio. Secondo lo standard il segnale deve andare sotto la dicitura "value" e deve essere inserito sotto forma di stringa, quindi, come si vedrà a breve, quando si converte da un formato che non è FHIR bisogna sempre portare il segnale in forma di stringa.

L'ultima risorsa utilizzata è il "Bundle", che non è altro che un contenitore di altre risorse. È utilizzato quando, dopo aver effettuato la query sul database, si devono mandare indietro i risultati ottenuti, per cui un Bundle conterrà più Observation.

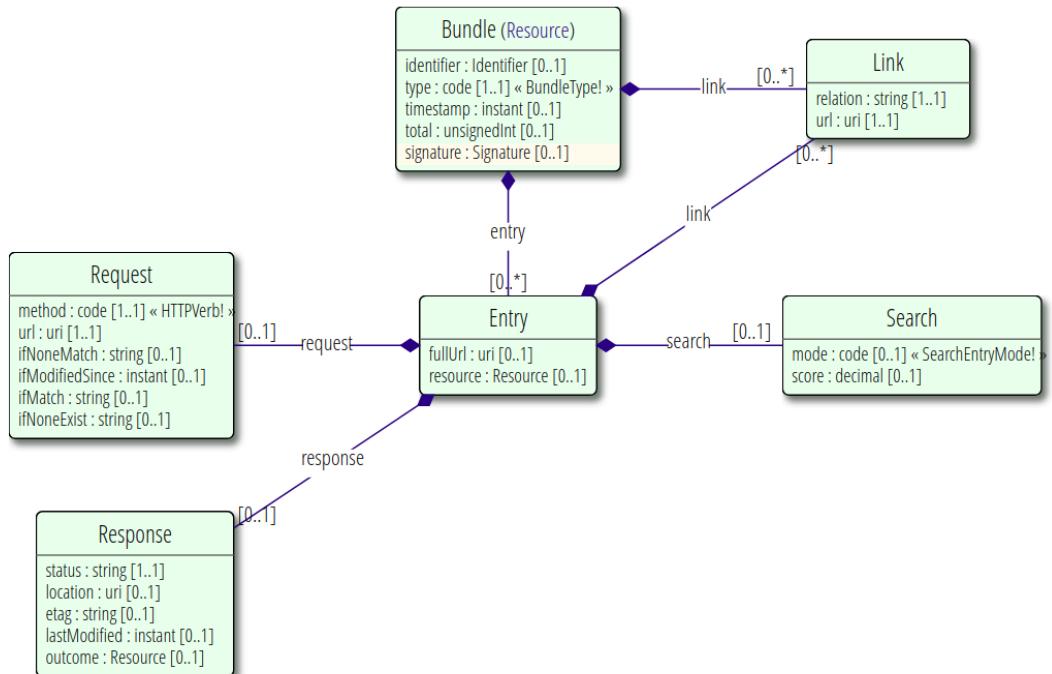


Figura 3.27: Class Diagram della risorsa Bundle secondo FHIR.

Le Observation popolano l'entità "Entry" che è associata al Bundle. All'interno di Entry ci sono i parametri della HTTP request utilizzata e del response in atto. Risulta quindi chiaro che Bundle è una risorsa utilissima quando si devono prendere più risorse, anche di tipo diverso.

3.4.2 Le classi del server

Si è già detto che il server Python è stato costruito a oggetti. Il primo oggetto con cui si ha a che fare scorrendo il codice contenuto in "webpage.py", ossia lo script principale da cui parte tutto, è senza dubbio l'allocazione dell'oggetto "app" di classe "Flask".

```
9  import threading
10 from flask import Flask, url_for, request, render_template, redirect, send_from_directory, make_response
11
12 app = Flask("sleepingrepo")
13 #il browser manderà solo cookie su request in https se il cookie è marcato "SECURE"
14 app.config['SESSION_COOKIE_SECURE'] = True
15 #non permetto l'invio di cookies da altri siti
16 app.config['SESSION_COOKIE_SAMESITE'] = 'Lax'
17
18 app.config['SESSION_COOKIE_HTTPONLY'] = True,
19
20 app.uh = userHandle("dati.json")
21 #qui va sostituito con i file di configurazione
22 app.url1 = 'http://'+get_ip()+":1025"
23 app.url2 = 'http://'+get_ip() + ":1026"
24 app.urlself = 'https://'+get_ip()
25 #app.urlself = 'https://sleepingrepo.ddns.net'
26 app.shared_lock = threading.Lock()
27
```

Figura 3.28: Allocazione dell'oggetto Flask.

Come si vede in Figura 3.28 viene instanziato l'oggetto con il comando:

```
app = Flask("sleepingrepo")
```

dove la stringa "sleepingrepo" è il nome che viene passato alla classe. Questa classe è stata importata dalla libreria di Flask, ma è importante all'interno del Class Diagram (Figura 3.29) poiché, come si vede in Figura 3.28 a riga 20, la classe Flask usa una classe creata appositamente per il progetto, ossia "userHandle".

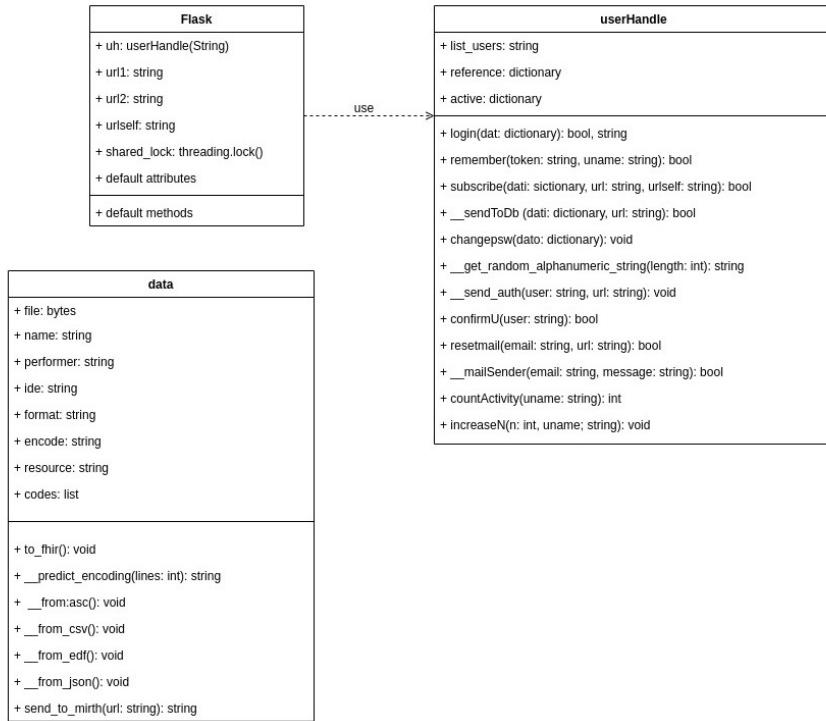


Figura 3.29: Class diagram del web server del progetto SleepingRepo.

Dal Class Diagram è quindi visibile che Flask alloca cinque nuovi attributi della classe: due sono oggetti, di cui il già citato userHandle e un oggetto per la gestione del multithreading denominato "shared_lock". Gli altri attributi sono delle stringhe e, nello specifico, sono i tre url che deve usare Flask: "url1" e "url2" sono gli url per i due canali del middleware, di cui si parlerà nella prossima sezione e "urlself", ossia l'URL del server da inserire quando si manda l'email di conferma.

Dando uno sguardo in generale, oltre a Flask ci sono due classi: "userHandle" e "data". Il primo è l'astrazione di un "gestore utenti" ed entra in ballo quando qualche utente chiede qualcosa, mentre il secondo è l'astrazione generica di un set di dati.

Ritornando al Class Diagram di userHandle si nota come quando si instanzia l'oggetto viene richiesta una stringa, chiamata "list_users" che non è altro che il nome del file json che contiene il nome degli utenti. L'altro attributo usato è

"reference", che è un dizionario ed è, appunto, tale lista di utenti. Infine c'è un dizionario chiamato "active" che contiene il numero di volte che un istituto ha caricato file in giornata. Quando si seguirà il flusso dei dati tramite il prototipo, si capirà meglio il motivo di questo dizionario.

```
 2 import smtplib, ssl
 3 from email.mime.text import MIMEText
 4 from email.mime.multipart import MIMEMultipart
 5 import json
 6 import random
 7 import string
 8 import socket
 9 from datetime import datetime
10 import requests
11
12 #classe che gestisce gli utenti
13 class userHandle(object):
14
15     ##costruttore dell'oggetto: input: string list_users
16     def __init__(self, list_users):
17
18         #tiro fuori la lista degli utenti
19         self.list_users = list_users
20         with open(self.list_users, "r") as f:          # apro il file con i dati degli utenti e inserisco quello che ho in una variabile
21             self.reference = json.load(f)
22         f.close()
23     try:
24         with open('active.json', "r") as f:          # apro il file con i dati degli utenti e inserisco quello che ho in una variabile
25             self.active = json.load(f)
26         f.close()
27     except:
28         with open('active.json', "w") as f:          #se non c'è il file lo creo
29             self.active = {"today": datetime.today().strftime('%Y-%m-%d'), "users":{}}
30             json.dump(self.active,f)
31         f.close()
32
33
34
```

Figura 3.30: Costruttore della classe userHandle.

La maggioranza dei metodi della classe userHandle ha un valore di ritorno booleano. Essendo azioni che vengono svolte sugli utenti si utilizza la variabile booleana come feedback per la riuscita o meno dell'operazione. Esistono quindi metodi per fare login, per impostare i cookies di durata fino ad una settimana, per l'iscrizione degli utenti, per confermare gli utenti, cambiare password o mandare email. Esiste pure una funzione, "sendToDb" che manda i dati dell'utente al database, infatti prende come input un dizionario chiamato "dati" e una stringa contenente l'URL del canale Mirth che si occupa dei ricercatori. All'interno della classe esistono metodi che permettono di generare stringhe alfanumeriche di lunghezza variabile per generare password casuali "token" per il cookie che fa saltare la fase di login. Infine gli ultimi due metodi permettono di gestire la variabile "active", per il conteggio delle attività degli istituti.

La classe "data" viene allocata appena arriva un nuovo file, come si può vedere nella figura 3.30 a riga 247.

```
219
220     elif action == "upload":
221         #chiedo i cookie, voglio lo username
222         #cookies = cherrypy.request.cookie
223         uname = name = request.cookies.get('uname')
224
225         #faccio il nome che ho nel mio json
226         who = 'reposleep' + uname
227         ist = app.uh.reference[who]['institute']
228
229
230         #vedo quanti upload ha fatto l'utente oggi
231         with app.shared_lock :
232             n = app.uh.countActivity(ist)
233             #creo la stringa che indetificherà i file da mandare
234             ide = ist + str(n)
235             #aumento il numero di upload per l'utente
236             app.uh.increaseN(n, ist)
237
238         #leggo il file
239
240         file = request.data
241         #prendo il nome del file
242
243         filename = request.headers['x-filename']
244
245
246         #istanzio un oggetto "dato"
247         dato = dataHandle.data(filename, file, uname, ide, 'observation' )
248         del file
249         del filename
250
251
252         #lo converto in fhir
253         dato.to_fhir()
254         #mando a mirth
255         s = dato.send_to_mirth(app.url1)
256
257
```

Figura 3.31: Gestione dei dati in upload.

La classe richiede in ingresso delle stringhe, ossia il nome del file, lo username dell'utente, l'identificativo del caricamento e il tipo di risorsa che si sta usando e un oggetto in bytes, ossia il file stesso. Sarà compito della classe capire come rappresentare il file a più alto livello.

```
11 #classe che rappresenta il dato
12 class data(object):
13
14     #costruttore della classe data. input: string name, bytes file, string performer, string ide, string resource, string format, string encode
15
16     def __init__(self, name, file, performer, ide, resource):
17         #parametri dell'oggetto
18
19         supported_format=["csv", "edf", "asc", "json"]
20         self.file = file
21         name = name.split('.')
22         self.format = name[-1]
23         self.encode = "unknown"
24         if self.format in supported_format and self.format !='edf' and self.format !='json':
25             self.encode = self.__predict_encoding(1000)
26
27
28         self.performer = performer
29         self.id = ide
30         self.resource = resource
31
32         with open('code_supported.json', "r") as f:           # apro il file con i dati degli utenti e inserisco quello che ho in una variabile
33             self.codes = json.load(f)['code']
34         f.close()
```

Figura 3.32: Costruttore della classe data.

Il costruttore comincia subito a ricavarsi un altro attributo della classe, chiamato "format", a partire dal nome. In questo modo si sa in che formato è il file in ingresso. A questo punto si cerca la codifica tramite il metodo della classe "predict_encoding", tranne che per i JSON, per cui già ci sono delle funzioni di Python utili allo scopo e per l'EDF¹¹, che segue un procedimento particolare. Infine viene aperto un file chiamato "code_supported.json" per controllare che il file sia tra i formati predisposti (al momento c'è solo il codice per la polisonnografia). La classe "data" ha anche un metodo pubblico chiamato "to_fhir". Non prende nulla in ingresso e non ha nessun valore di ritorno in quanto utilizza solo attributi interni. In base al formato nella variabile "format" "to_fhir" decide quale metodo usare per portare il file in FHIR.

```
38     #funzione che inizia la conversione verso fhir. non ha né input né output agendo solo sui parametri interni alla classe
39     def to_fhir(self):
40         #scelgo il formato
41         if self.format == 'asc':
42             self.__from_asc()
43         elif self.format == 'csv':
44             self.__from_csv()
45         elif self.format == "edf":
46             self.__from_edf()
47         elif self.format == "json":
48             self.__from_json()
```

Figura 3.33: Funzione "to_fhir".

Come si vede in Figura 3.33 ci sono quattro conversioni possibili partire da:

- **ASC.** I file ASC solitamente contengono segnali EEG. È un file che una volta decodificato con la funzione di Python "decode" risulta essere una stringa.

¹¹EDF sta per "European Data Format" ed è un formato europeo per lo scambio e il salvataggio di segnali biomedici multicanale

Viene quindi diviso in lista quando ci sono gli "a capo" e la lista viene divisa in due, ossia viene divisa la parte del segnale vero e proprio dalle altre informazioni. Le informazioni vengono sistematate come stringa e così viene fatto per il segnale. Il tutto viene infine inserito nel template.

```
123 |     def __from_asc(self):
124 |         #metodo privato
125 |         #decodifico il file
126 |         file = self.file.decode(self.encode)
127 |         #comincio a dividire in una lista. ogni riga è un elemento della lista
128 |         lines = file.split('\n')
129 |         lines1 = lines[0:10]
130 |         lines2 = lines[11:len(lines)-1]
131 |         file = []
132 |         for i in lines1:
133 |             file.append(i.split(' '))
134 |             #prendo la sampling rate
135 |             sr = file[5][len(file[5])-1]
136 |             sr = int(sr)
137 |             #prendo l'unità di misura
138 |             mis = file[6][len(file[6])-1].replace("'", '')
139 |             mis = mis.replace('[', '')
140 |             mis = mis.replace(']', '')
141 |
142 |             #prendo i nomi dei canali (?)
143 |             n1 = str(lines1[8].split(','))
144 |             n1 = n1.replace('.', '')
145 |             n1 = n1.replace('[', '')
146 |             n1 = n1.replace(']', '')
147 |             n1 = n1.replace("'", '')
148 |             n1 = n1.replace(' ', '')
149 |             n2 = file[9]
150 |             data = []
151 |             for i in lines2:
152 |
153 |                 #prendo gli spazi e ci metto un _ ne ho messi relativamente pochi perché così copro i casi con tante cifre
154 |                 var = i.replace(' ', '_')
155 |                 var = var.replace(',', '_')
156 |                 var = var.replace('_', '_')
157 |                 #do lo stesso comando due volte così elimino la possibilità dei __
158 |                 var = var.replace('__', '_')
159 |                 var = var.replace('_', '_')
160 |                 data.append(var)
161 |
162 |             #qua faccio la trasposta passando per un oggetto di tipo map
163 |             da = list( map(list, zip(*data)))
```

Figura 3.34: Inizio della conversione da ASC.

```
180 |
181 |     with open('observation_temp.json', 'r') as f:
182 |         template = json.load(f)
183 |         f.close()
184 |         #ciclo lungo i componenti del template
185 |         template['effectiveDateTime'] = datetime.today().strftime('%Y-%m-%d')
186 |         template['performer'][0]['reference'] = self.performer
187 |         template['identifier'][0] ['value'] = 'SleepData'+self.id
188 |         for i in range (len(template['component'])):
189 |             if template['component'][i]['code'][0]['coding'][0]['code'] == '18648009':
190 |                 template['component'][i]['code'][0]['coding'][0]['display'] = n1
191 |                 template['component'][i]['valueSampledData'][0]['data'] = stringa
192 |                 template['component'][i]['valueSampledData'][0]['dimensions'] = 8
193 |                 template['component'][i]['valueSampledData'][0]['origin'][0]['unit'] = mis
194 |             self.file = template
195 |             self.format = 'json'
```

Figura 3.35: Fine della conversione da ASC.

- **CSV.** I file CSV sono di solito contengono informazioni provenienti da acceleratori, per esempio dal sensor tile della "STmicroelectronics". Tramite la libreria "pandas" il CSV viene importato, vengono presi i nomi dei canali e viene creata la stringa contenente il segnale. Infine le informazioni vengono messe all'interno del template.

```

69      #funzione per convertire dal csv. non ho né input né output agisce tutto internamente all'oggetto
70  def __from_csv(self):
71      #metodo privato
72      #metto a stringa il file
73      s = str(self.file,self.encode)
74      #e poi lo preparo per essere letto con pandas
75      data = StringIO(s)
76      data = pd.read_csv(data)
77      #pulizia della stringa dei dati dei sensori e delle unità di misura
78      ch = data.columns.array
79      ch = ch[2:11]
80      canali = ch
81
82      ch = str(list(ch))
83      ch = ch[1:-1]
84      ch = ch.replace("", ' ')
85      ch = ch.replace("]", "[")
86      ch = ch.split("[")
87      #faccio a meno uno perché l'ultimo elemento è vuoto
88      ch = ch[:-1]
89      mis = ''
90      cha = ''
91      i = 0
92      #ciclo per scindere le unità di misura dai canali
93      while i < len(ch):
94          mis = mis + ch[i + 1] + ' '
95          cha = cha + ch[i] + ' '
96          i = i + 2
97      stringa = ''
98      #ciclo sulla stringa che comporrà i dati
99      for i in canali:
100         segnali = str(list(data[i]))
101         segnali = segnali.replace(", ''")
102         segnali = segnali.replace("]", ""))
103         stringa = stringa + segnali + '\n'
104
105
106     with open('observation_temp.json', 'r') as f:
107         template = json.load(f)
108         f.close()
109     #ciclo lungo i componenti del template
110     template['effectiveDateTime'] = datetime.today().strftime('%Y-%m-%d')
111     template['performer'][0]['reference'] = self.performer
112     template['identifier'][0][0]['value'] = 'SleepData'+self.id
113     for i in range(len(template['component'])):
114         if template['component'][i]['code'][0]['code']=='733919004':
115             template['component'][i]['code'][0]['coding']=cha
116             template['component'][i]['code'][0]['display']=cha
117             template['component'][i]['valueSampledData']['data']=stringa
118             template['component'][i]['valueSampledData']['dimensions']=len(canali)
119             template['component'][i]['valueSampledData']['origin']['unit']=mis
120             self.file = template
121             self.format = 'json'

```

Figura 3.36: Conversione da CSV.

- **EDF.** Il formato EDF è molto usato per le polisonnografie. È anche stato il formato più difficile da convertire in quanto contiene molti segnali ad alta frequenza di campionamento per molto tempo. Questo ha reso molto complicato inserire tutto in un JSON in quanto veniva occupata una grande quantità di ram. Si noti, inoltre, che vengono continuamente pulite variabili prima che il garbage collector di Python lo faccia per liberare quanta più ram possibile, dato Python non lo fa abbastanza rapidamente. Se si provasse ad usare un solo JSON il computer che su cui funziona il web server arriverebbe quasi a saturare gli 8 GB di memoria RAM per la conversione di un singolo file. Si è ricorsi, quindi, ad un modo semplice ed efficace: si riduce consumo di RAM "appoggiandosi" al file system del computer salvando i vari segnali via via che vengono aperti e mandandoli in sequenza al middleware. Inoltre risulta problematico passare da byte a edf, perché la funzione "EdfReader" non prende in ingresso bytes, quindi si salva temporaneamente il file nel file system e si procede.

```

198 #funzione per la conversione da edf. L'idea di base è quella che prendo il file edf
199 #e lo divido in più fhir che mando con più requests.
200 #Devo fare così perché a livello computazionale è pesante tenere tutto in ram
201 def __from_edf(self):
202
203     s = str(self.file, self.enconde)
204     #e poi lo preparo per essere letto con pandas
205     #data = StringIO(s)
206     #apro il file edf di appoggio che ho creato
207     with open("tmp.edf", "wb") as f:
208         f.write(self.file)
209         f.close()
210     del f
211     del self.file
212     raw = pyedflib.EdfReader("tmp.edf")
213
214     #liste con i nomi dei canali che possono capitare in un edf
215     eeg_name=['Fp2-F4', 'F4-C4', 'C4-P4', 'P4-O2', 'C4-A1']
216     eog_name=['ROC-LOC']
217     emg_name=['EMG1-EMG2']
218     ecg_name=['ECG1-ECG2']
219     imp_name=['ADDDOME', 'ADDOME']
220
221
222
223     with open('observation_temp.json', 'r') as f:
224         template = json.load(f)
225         f.close()
226     #compilo i campi del json
227     template['effectiveDateTime'] = datetime.today().strftime('%Y-%m-%d')
228     template['performer'][0]['reference'] = self.performer
229     template['identifier'][0] ['value'] = 'SleepData'+self.id
230
231
232     os.mkdir(path)

```

Figura 3.37: Inizio della conversione da EDF.

```

255     #in questo ciclo si va a cercare il segnale. Poi si controlla che già non esista già un altro segnale per questa risorsa. Se c'è si aggiunge
256     #variabile per capire se non ho trovato il segnale
257     notfound=True
258
259     if raw.getLabel(i) in eeg_name:
260         fname='SleepData'+self.id+'eeg.json'
261     elif raw.getLabel(i) in eog_name:
262         fname='SleepData'+self.id+'eog.json'
263
264     elif raw.getLabel(i) in ecg_name:
265         fname='SleepData'+self.id+'ecg.json'
266
267     elif raw.getLabel(i) in emg_name:
268         fname='SleepData'+self.id+'emg.json'
269
270     elif raw.getLabel(i) in imp_name:
271         fname='SleepData'+self.id+'imp.json'
272     #metti un else per bypassare tutto easy
273     else:
274         notfound=False
275
276     if notfound==True:
277
278         try:
279             with open(self.id + "/" + fname, 'r') as f:
280                 js = json.load(f)
281             except:
282                 js = template
283                 #prendo -8 perché voglio solo gli ultimi 8 caratteri del nome
284
285                 with open("component/" + fname[-8:], 'r') as f:
286                     js ['component'] = [json.load(f)]
287                     f.close()
288                 #metto questa operazione qua perché non ha senso farla sempre
289                 js['identifier'][0]['value'] = self.id
290                 js['component'][0]['code']['coding'][0]['display'] = js['component'][0]['code']['coding'][0]['display'] + raw.getLabel(i) + " "
291                 js['component'][0]['valueSampledData'][0]['data'] = js['component'][0]['valueSampledData'][0]['data'] + str(raw.readSignal(i).tolist())
292
293                 with open(self.id + "/" + fname, 'w') as f:
294                     json.dump(js, f)
295                     f.close()
296                 del js
297                 del fname
298
299         self.format = "path"
300         self.file = self.id
301
302     del f

```

Figura 3.38: Fine della conversione da EDF.

- **JSON.** Quando ci si trova davanti ad un json non si fa una vera e propria "conversione" ma una validazione. Vengono quindi inseriti i parametri generali e si controlla che il json sia quello giusto, altrimenti viene impostato un formato "None"¹², cioè nullo.

¹²None è la stringa che usa Python per indicare l'assenza di valore.

```
289 #funzione per la validazione dei json. Non ha né input né output
290     def __from_json(self):
291         #banalmente carico il file
292         s = json.loads(self.file)
293         #devo fare così perché se è un json totalmente a caso mi torna errore il codice
294         try:
295             #check se è una observation supportata e se ha delle componenti non vuote (se sono tutte nulle lo vedo su mirth)
296             if s['code'][0]['code'] in self.codes and len(s['component'])!=0 and s["resourceType"]=="Observation":
297                 #solito gioco che metto la data e l'id personalizzato
298                 s['effectiveDateTime'] = datetime.today().strftime('%Y-%m-%d')
299                 s['performer'][0]['reference'] = self.performer
300                 s['identifier'][0] ['value'] = 'SleepData'+self.id
301                 self.file = s
302             #se non va metto il formato "null" così esce l'errore del formato sbagliato
303             else:
304                 self.format = None
305             except:
306                 self.format = None
```

Figura 3.39: Validazione di un file JSON.

Dopo aver convertito il file la classe fornisce un metodo per mandare i file al database. Esso richiede in ingresso solo una stringa chiamata "url", che consiste nell'URL da contattare.

3.5 Il prototipo

Nella presente sotto-sezione si è arrivati al prototipo vero e proprio: verrà quindi seguito il flusso di lavoro del sistema nei suoi tre casi d'uso mostrati nello Use Case, aiutandosi con le interfacce, gli Activity Diagram e il codice che è stato scritto. Come si è già detto erano stati individuati tre "macro" casi d'uso: registrazione, upload e download. Gli ultimi due, in realtà, sono extend del caso d'uso login. Dal momento che il login diventa marginale rispetto agli altri due viene inglobato all'interno di upload e download, così come è stato fatto nel Sequence Diagram. L'ordine con cui verranno presentati vuole seguire una "inizializzazione" del sistema: verrà quindi mostrata la registrazione, per poi passare all'upload, in modo da spiegare come vengono caricati i dati e infine ci si volgerà al download. Quest'ordine, in realtà, non è solo l'ordine con cui vogliono essere presentati i casi d'uso in fase di scrittura della tesi ma è anche l'ordine in cui sono state effettivamente sviluppate la parti di codice. I motivi sono molti e il primo è sicuramente quello di natura logica sopra citato, ma non solo: è stato fatto un lavoro crescente in difficoltà e complessità. Si prenda come esempio la registrazione, in una prima fase non venivano inviati i dati al database, per cui programmare era più semplice non coinvolgendo Mirth. L'upload ha aggiunto la difficoltà di inserire Mirth, ma è risultato comunque più facile rispetto al download perché il comando era estremamente più semplice. Il download è, infatti, risultato essere il più complicato perché ogni request viene creata a partire da diversi "mattoncini" di stringhe, come si vedrà in seguito, per permettere all'utente una grossa fetta di libertà decisionale. Discorso totalmente

analogo sussiste per la creazione delle query in SQL, che è ben più complessa delle situazioni precedenti.

3.5.1 Il primo impatto: la registrazione

La prima volta che ci si connette al portale appare una schermata di login come quella in Digura 3.40.

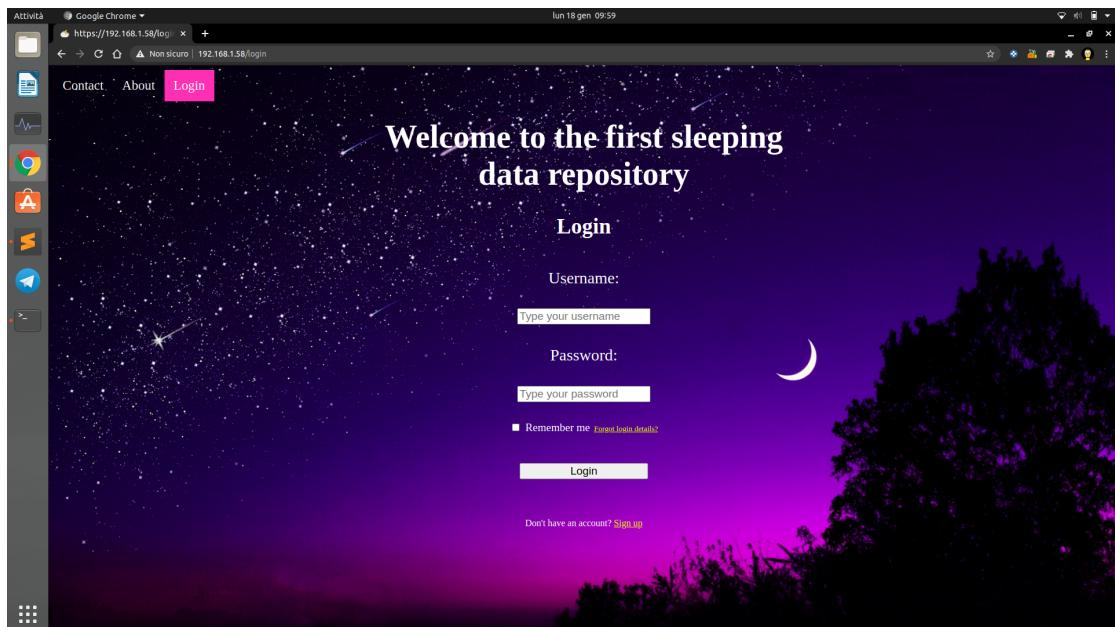


Figura 3.40: Interfaccia di Login.

In alto c'è la barra di navigazione che porta a due pagine di supporto e al centro c'è il form di login. In giallo, sono evidenziati due parti di testo che hanno un link: una permette di recuperare la password e l'altra permette di registrarsi, come mostrato nella Figura 3.41.

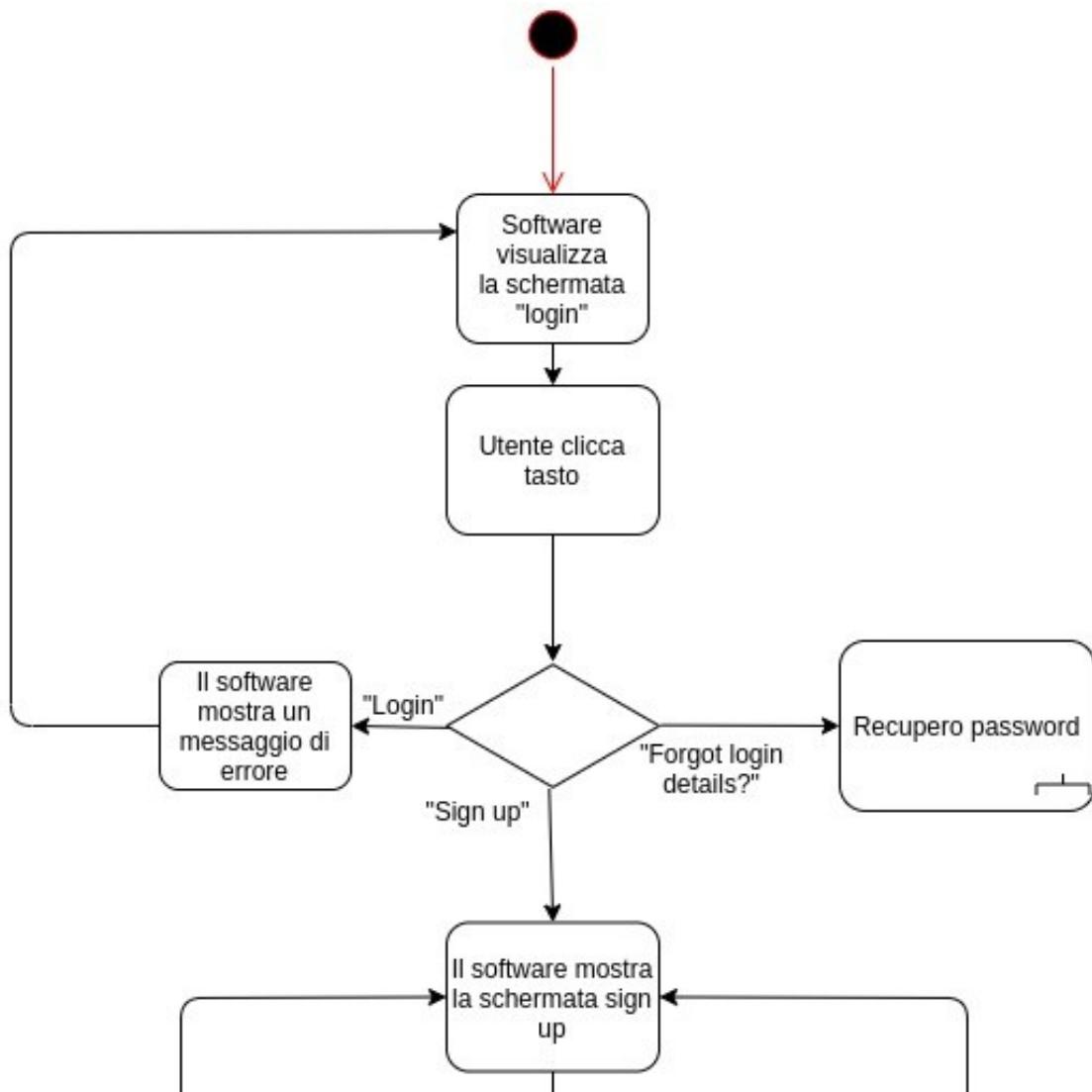


Figura 3.41: Dettaglio dell'Activity Diagram del caso d'uso di registrazione.

Prima di continuare con la registrazione è utile far notare cosa succede se si sono dimenticate le credenziali. Una volta cliccato su "Forgot login details?" appare una pagina per l'inserimento dell'email, come in Figura 3.42.

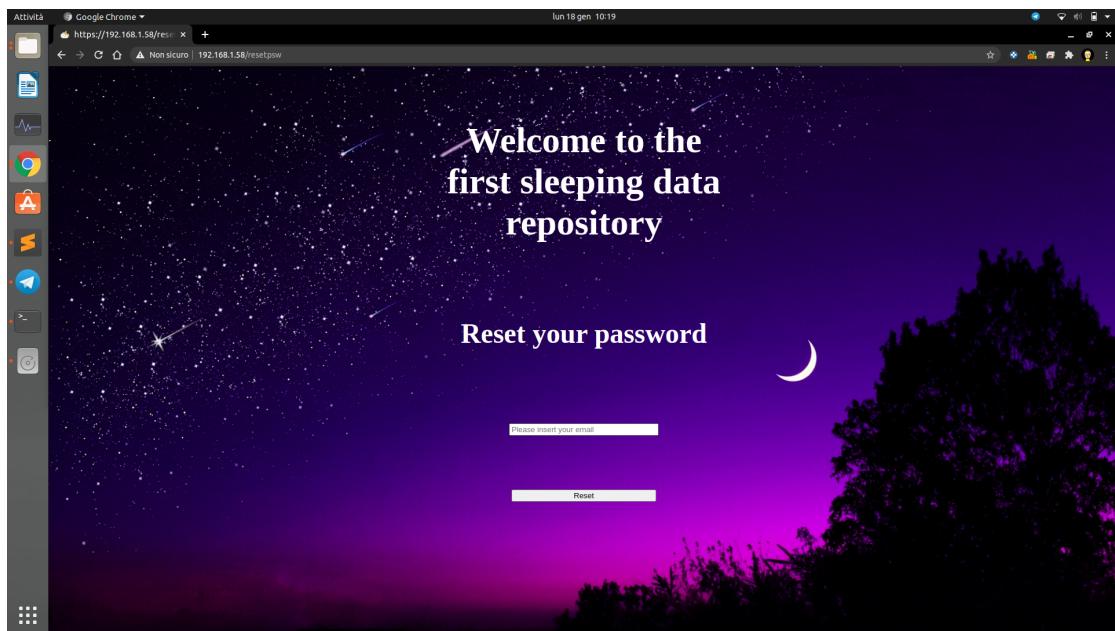


Figura 3.42: Interfaccia di reset della password

Una volta cliccato su reset viene mandato il segnale di invio dell'email al web server. A quel punto sulla propria casella di posta arriverà un messaggio come in Figura 3.43.

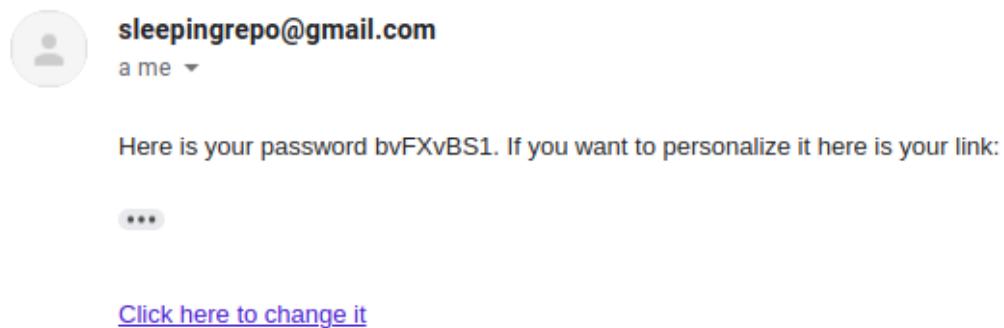


Figura 3.43: Email con password casuale e con il link per personalizzarla.

L'email darà all'utente la possibilità di tenere la password creata casualmente con la classe userHandle oppure di cambiarla seguendo il link che viene mandato. Se si clicca sul link si arriverà ad un form molto simile a quello della registrazione.

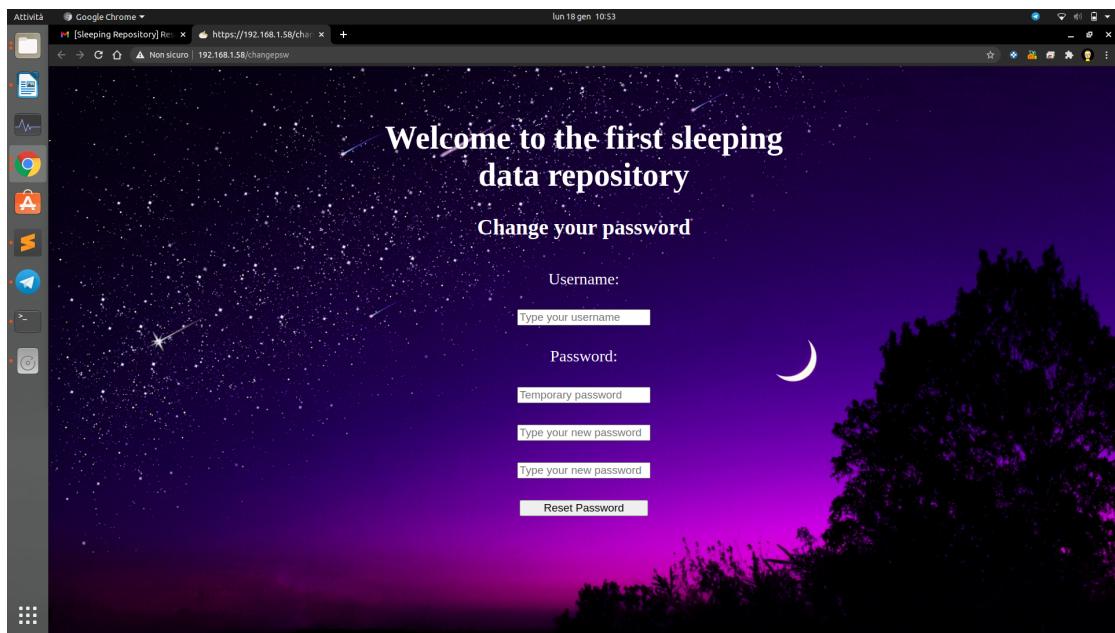


Figura 3.44: Form per cambiare password.

Tornando alla registrazione, quando c'è un nuovo utente l'unica cosa che può fare è cliccare "sign up". A quel punto, come mostrato in Figura 3.41, viene visualizzata la schermata di sign up, che richiede i dati che servono al sistema.

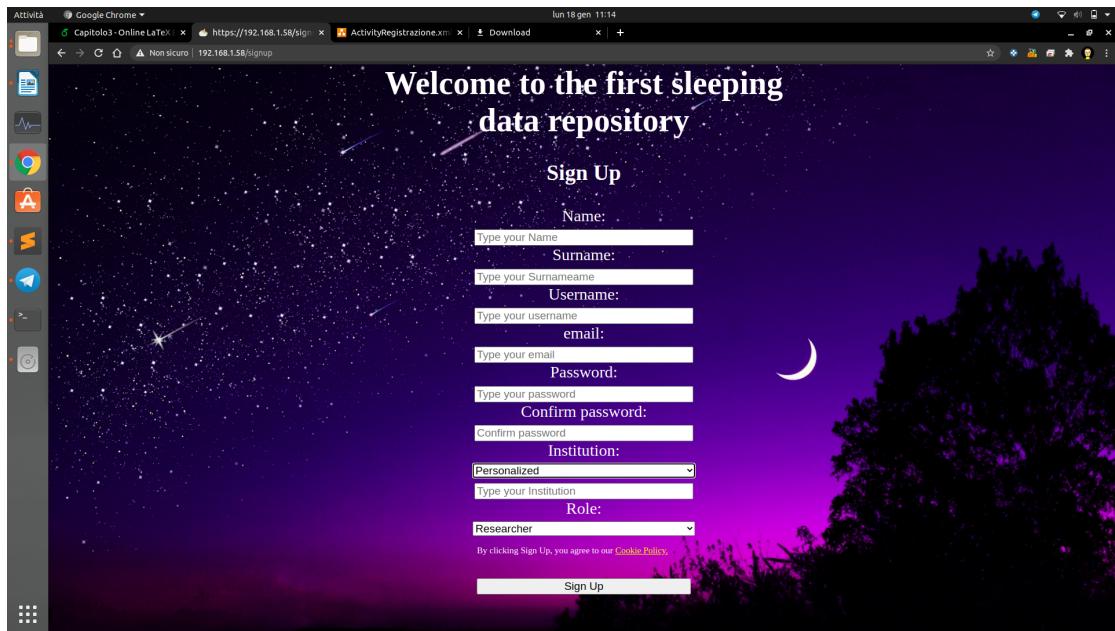


Figura 3.45: Form per registrarsi.

Si noti dalla Figura 3.45 che è possibile scegliere l'istituzione sia scrivendo il nome, sia scegliendo tra quelli già presenti, perché aggiunte da qualche utente. È stato inserito un menù a tendina pure per il ruolo, in ottica di espansione futura, come si diceva nella sezione dedicata ai database, anche se ora contiene solo quello di ricercatore.

Continuando a seguire l'Activity Diagram, che per questioni di leggibilità viene presentato diviso in dettagli, si incappa subito nell'invio dei dati al database.

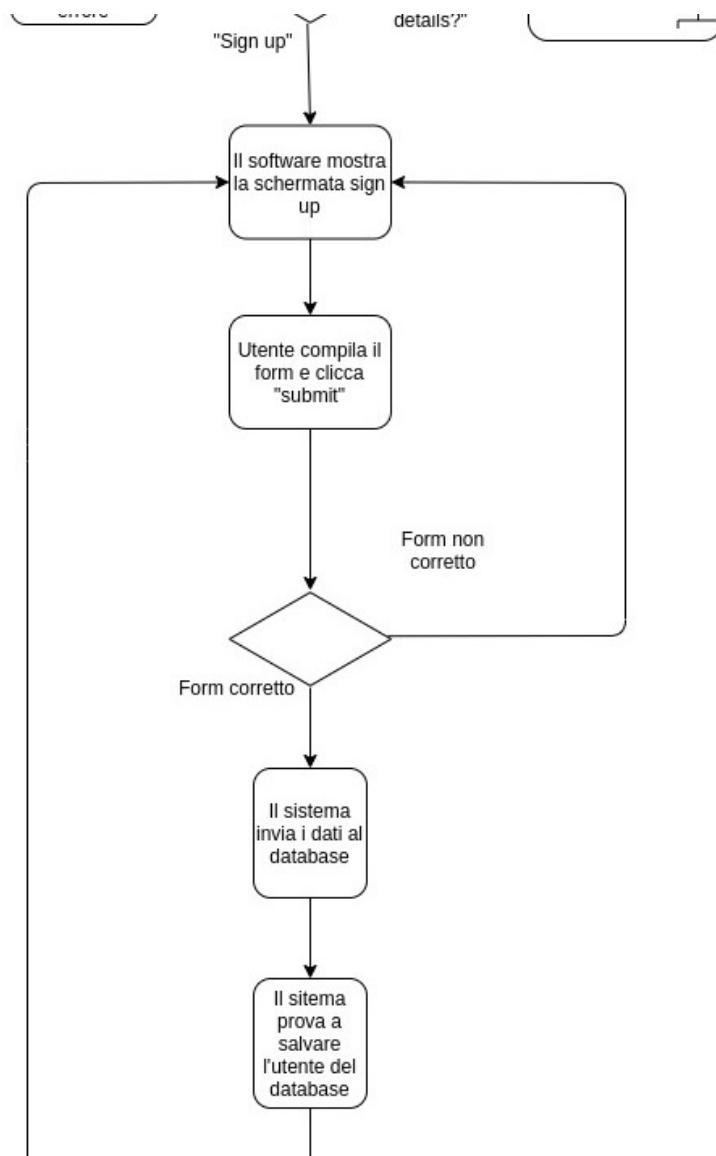
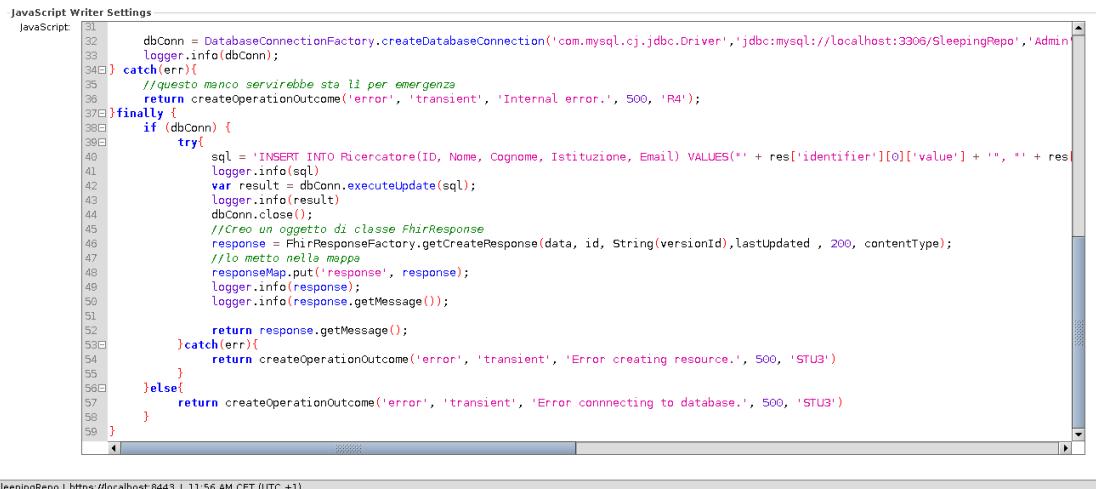


Figura 3.46: Secondo dettaglio dell'Activity Diagram del caso d'uso di registrazione.

In realtà incluso nella frase "validazione form", c'è prima un controllo sul JSON che contiene gli utenti nel web server a fungere da "primo filtro". Dopo il primo filtro i dati dell'utente vengono presi ed elaborati per creare un "practitioner" secondo FHIR in JSON, viene quindi aperto il template e compilato. Tutto ciò avviene nei metodi "subscribe" e "__sendToDB" della classe userHandle che venivano citati in precedenza. Il web server manda quindi una richiesta POST al middleware, anzi per meglio dire utilizza la sintassi di CREATE di FHIR. Una volta che il messaggio arriva a destinazione viene preso in carico dalla "Source" di Mirth, "FHIR Listener". Una volta eseguite le prime validazioni e conversioni passa alla destination dove c'è uno script in Javascript che effettua controlli più specifici, prende i dati e costruisce il comando in SQL.



```

JavaScript Writer Settings
JavaScript:
31
32     dbConn = DatabaseConnectionFactory.createDatabaseConnection('com.mysql.cj.jdbc.Driver','jdbc:mysql://localhost:3306/SleepingRepo','Admin');
33     logger.info(dbConn);
34 } catch(err){
35     //questo manca servirebbe sta li per emergenza
36     return createOperationOutcome('error', 'transient', 'Internal error.', 500, 'R4');
37 }finally {
38     if (dbConn) {
39         try{
40             sql = 'INSERT INTO Ricercatore(ID, Nome, Cognome, Istituzione, Email) VALUES(' + res['identifier'][0]['value'] + "", "" + res['name']['given'][0] + " " + res['name']['family'][0] + ", " + res['address'][0]['city'] + ", " + res['address'][0]['country']);
41             logger.info(sql);
42             var result = dbConn.executeUpdate(sql);
43             logger.info(result);
44             dbConn.close();
45             //creo un oggetto di classe FhirResponse
46             response = FhirResponseFactory.getCreateResponse(data, id, String(versionId),lastUpdated , 200, contentType);
47             //lo metto nella mappa
48             responseMap.put('response', response);
49             logger.info(response);
50             logger.info(response.getMessage());
51
52             return response.getMessage();
53         }catch(err){
54             return createOperationOutcome('error', 'transient', 'Error creating resource.', 500, 'STU3')
55         }
56     }else{
57         return createOperationOutcome('error', 'transient', 'Error connecting to database.', 500, 'STU3')
58     }
59 }

```

Figura 3.47: Estratto dello script in Javascript all'interno di Mirth dove si inserisce il ricercatore nel database.

Lo script è strutturato in modo che possa intercettare quanti più errori possibili così che possa mandare indietro un response. L'obiettivo è impedire che si blocchi per non lasciare senza risposta il web server. Se, però, va tutto a buon fine il web server manda l'email finale per confermare l'identità dell'utente. È stato scelto di mandare l'email dopo l'inserimento perché, qualora il database non fosse raggiungibile per un qualsiasi motivo, l'inserimento del nuovo utente dovrebbe fermarsi. Se non si facesse così un utente risulterebbe registrato e confermato sul portale ma non potrebbe caricare i dati perché la procedura di inserimento fallirebbe in quanto verrebbe a mancare la foreign key della misurazione.

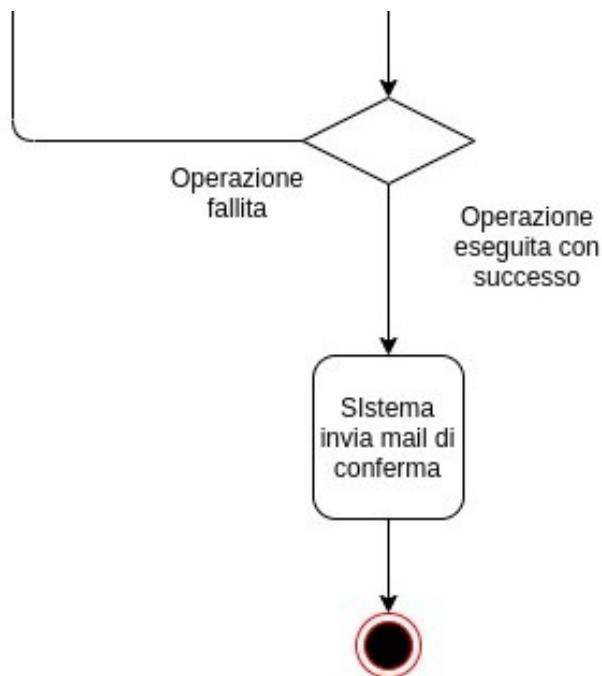


Figura 3.48: Terzo dettaglio dell'Activity Diagram del caso d'uso di registrazione.

3.5.2 Upload dei dati

Una volta che l'utente si è registrato si trova di fronte la schermata del download, ma prima di scaricare qualcosa è bene vedere come caricare dei dati, ipotizzando che il sistema sia appena stato avviato e che quindi non abbia nulla. Cliccando sulla barra in alto si andrà alla schermata di upload.

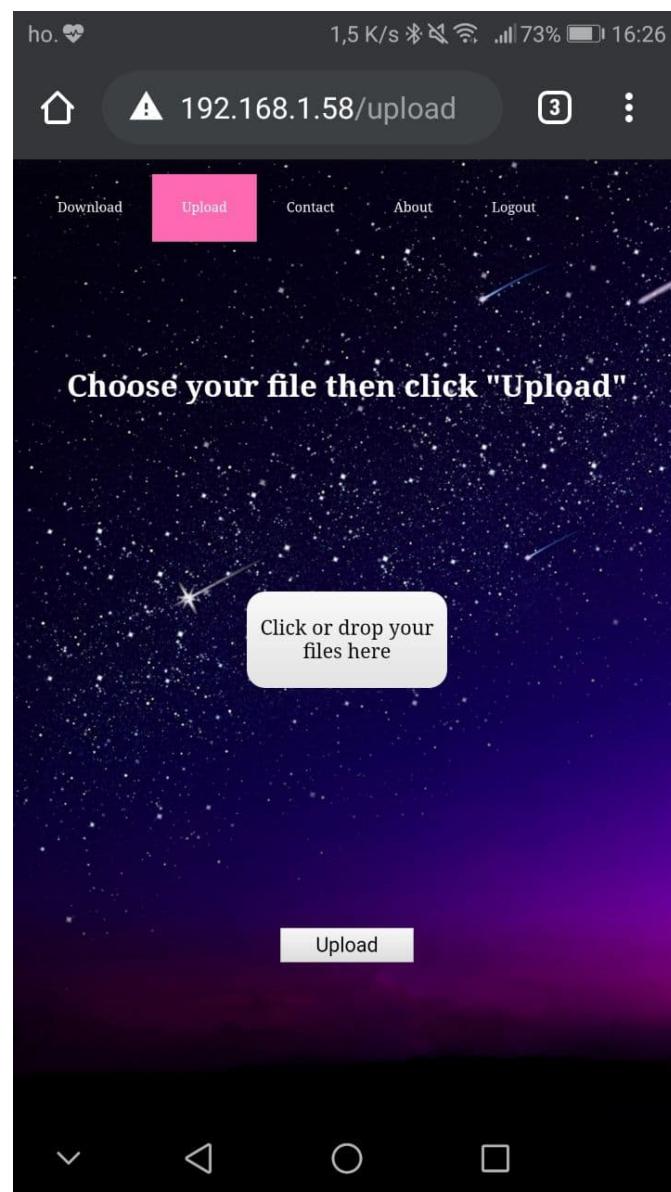


Figura 3.49: Interfaccia di upload da smartphone.

L’interfaccia si presenta in modo molto semplice, con un grande tasto centrale per caricare i file nel browser e un tasto più in basso per mandarli al web server.

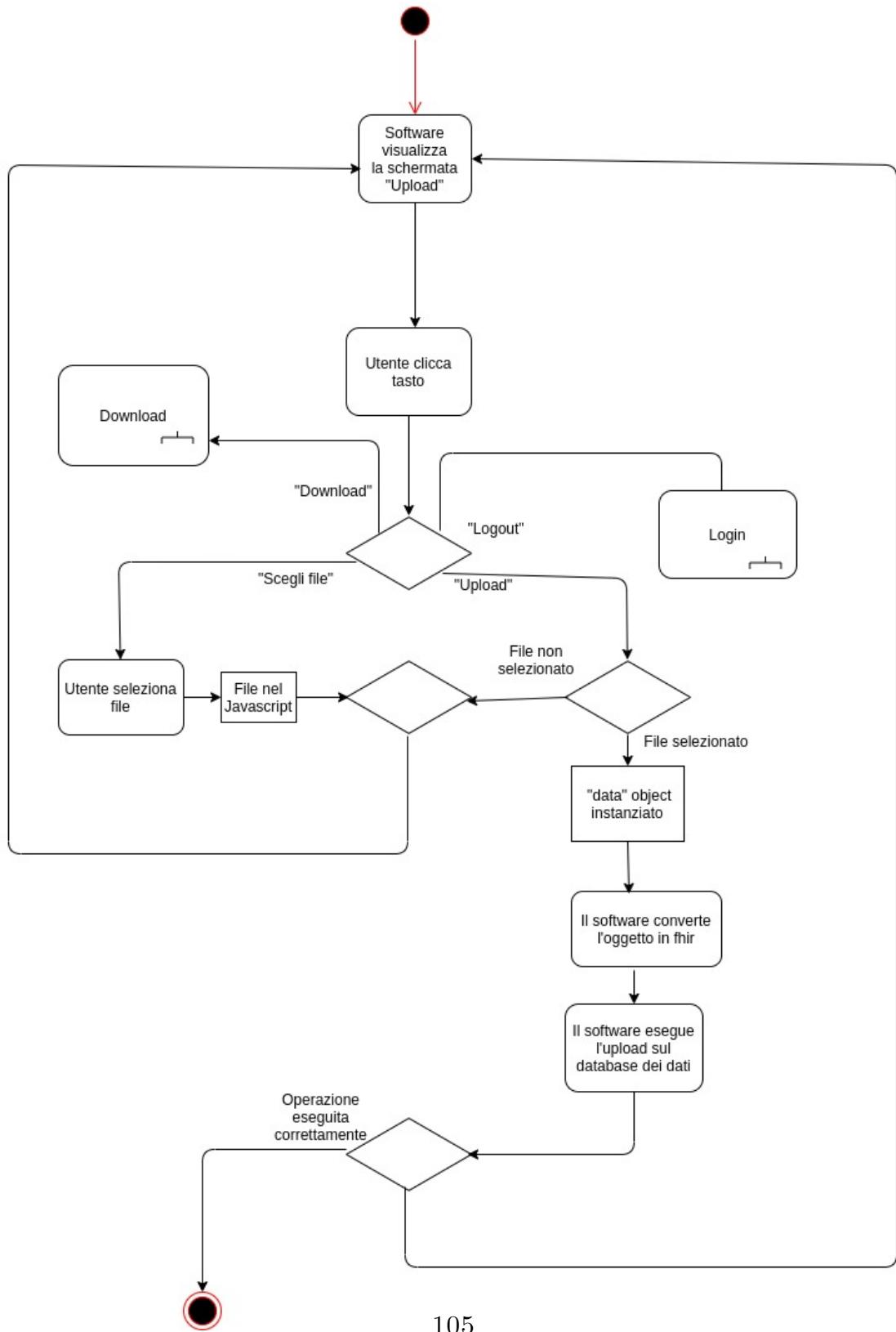


Figura 3.50: Activity Diagram del caso d'uso di upload.

Appena prima di mandare il file la pagina effettua un controllo: controlla nei cookies se l'utente ha verificato l'indirizzo email, altrimenti non si può né caricare né scaricare alcunché. È stato inserito questo meccanismo per ragioni di sicurezza, perché lasciare accesso ai dati senza conferma risulterebbe rischioso.

Appena effettuato l'invio il sistema si preoccupa di generare quello che nel database veniva chiamato "ID", ossia l'identificativo della risorsa che funge da chiave insieme alla data. Questo perché l'ID è il nome dell'istituto a cui appartiene il ricercatore che ha caricato i dati, seguito da un numero, che è il numero di file caricati quel giorno da quell'istituto di ricerca. Si è scelto di utilizzare questo metodo, che può sembrare astruso, perché in realtà semplifica notevolmente l'individuazione delle risorse. Identificare una risorsa con un codice alfanumerico, seppur lungo, avrebbe portato due problematiche: la prima è la probabilità, seppur remota, che ci siano due codici uguali, dal momento che l'assegnazione del codice sarebbe casuale, la seconda è che l'ID della risorsa è difficile da capire proprio perché non è deterministico. Sarebbe pure difficile da ricordare se fosse una semplice enumerazione crescente e non darebbe nessuna informazione sulla risorsa. In questo modo, invece, dall'ID si sa con che metodologia può essere stata acquisita la misurazione. Se per esempio la risorsa si chiama "Astel01", si sa che è stata acquisita da qualcuno della Astel Electronic Engineering, quindi questo operatore avrà quasi sicuramente utilizzato il set di sensori indossabili per testa, torace e piedi progettato in azienda. Introducendo la modalità batch, ossia la possibilità di inviare più file contemporaneamente, ci si è trovati di fronte ad un problema, principalmente ad un limite del sistema di creazione degli ID. La modalità batch, infatti, è stata implementata per essere il più veloce possibile, tenendo conto che alcuni file possono raggiungere le centinaia di MB.

```
4 //funzione per la preparazione a mandare i dati. Non ha né input né output perché prende le informazioni direttamente dal'html
5 function sendData()
6 {
7     perm = dataPermission();
8     if (perm == true)
9     {
10         //prendo i file
11         files = document.getElementById("file").files;
12         //prendo al variabile di appoggio per il conteggio dall'elemento html
13         //è una stringa ma la divido in un array
14         total = document.getElementById("count").value.split(".");
15         //all'inizio metto il totale dei file
16         total[0] = files.length.toString();
17         document.getElementById("count").value = total.join(".");
18         //ciclo sui file ed eseguo l'upload
19         for (var i = 0; i < files.length; i++)
20         {
21
22             var fil = files[i]
23             console.log(fil);
24             upload(fil, files.length);
25
26         }
27
28
29     }
30 }
```

Figura 3.51: Parte di codice dove viene gestito il caricamento batch in Javascript.

Di conseguenza i dati vengono inviati con diverse request che flask gestisce in modo separato utilizzando il multithread. Se è vero che il browser manda i file uno dietro l'altro, è anche vero che si possono accumulare ritardi durante il percorso lungo la rete, dovuti magari ai controlli di integrità fatti dal sottostante protocollo TCP su cui HTTP si basa. Di conseguenza, ci si è trovati a dover fronteggiare request praticamente in contemporanea, con ritardi minori del ms. Con delta temporali così stretti è diventato un problema aggiornare la stessa variabile da più thread, perché capitava che entrambi andassero a utilizzare la stessa copia della variabile di conteggio andando a creare ID identici. Questo spiega la creazione dell'oggetto "threading.lock" chiamato "shared_lock": esso non è altro che un sistema per far accedere e modificare la variabile dei conteggi un thread per volta.

```
---  
220 elif action == "upload":  
221     #chiedo i cookie, voglio lo username  
222     #cookies = cherrypy.request.cookie  
223     uname = name = request.cookies.get('uname')  
224  
225     #faccio il nome che ho nel mio json  
226     who = 'reposleep' + uname  
227     ist = app.uh.reference[who]['institute']  
228  
229  
230     #vedo quanti upload ha fatto l'utente oggi  
231     with app.shared_lock :  
232         n = app.uh.countActivity(ist)  
233         #creo la stringa che indetificherà i file da mandare  
234         ide = ist + str(n)  
235         #aumento il numero di upload per l'utente  
236         app.uh.increaseN(n, ist)  
237
```

Figura 3.52: Parte di codice dove viene gestito il caricamento batch in Python.

Una volta creato l'ID viene instanziato un oggetto "data", che viene convertito in FHIR e mandato al database. Prima di arrivare al database, però, viene elaborato e controllato da Mirth, che in questo caso è fornito di uno script di preprocess dove il file viene aperto, vengono presi i segnali e vengono salvati in una apposita cartella. Di conseguenza al termine di questa fase il valore della chiave "component", anziché essere una lista di JSON, sarà una lista di stringhe, contenenti i percorsi dei segnali.

```

//parse il messaggio
message = JSON.parse(message)

try{
    logger.info(message['resourceType'])
    //check se stanno mandando la risorsa giusta
    if (message['resourceType']=='Observation'){
        {
            var res=message
            var signals=res['component'];
            //creo la parte del nome comune
            p='/home/sam/resources/' +res['identifier'][0]['value'] + res['effectiveDateTime'];
            //ciclo sui segnali
            for (i=0; i<signals.length; i++){
                if (res['component'][i]['valueSampledData']['data']==null){
                    res['component'][i]=null;
                }
                else{
                    code= res['component'][i]['code']['coding'][0]['code'];
                    path=p+'_'+code+'.'json' //metto solo qua perché così recuperò più facilmente la variabile mentre sto sul db
                    var signal = res['component'][i];
                    res['component'][i]=path
                    FileUtil.write(path, false, JSON.stringify(signal));
                }
            }
            message = res;
            message=JSON.stringify(message);
            logger.info(message)
        }
    }
    return message;
} catch(err){
    return FhirUtil.createOperationOutcome('error', 'invalid', "Invalid resource", 'R4');
}

```

Figura 3.53: Script di preprocess.

Il messaggio così modificato arriva alla destination dove viene diviso, in modo da preparare il comando INSERT in modo automatico analogamente a quanto succede quando si aggiunge un utente. L'unica, piccola, differenza con lo script che inserisce un nuovo utente, oltre chiaramente alle differenze dovute al diverso tipo di risorsa, è l'inserimento di una piccola query sulla tabella ricercatore. Viene fatta per una questione di mantenibilità, dato che ora che c'è il portale è impossibile che un utente non registrato carichi dei dati, ma per rendere più modulare il sistema è necessario prendere qualche piccola precauzione, ossia è stato inserito un record nella tabella ricercatore con username "null" e tutti i campi null. A questo record vengono collegate tutte le misurazioni effettuate da ricercatori non presenti nel database, dato che altrimenti non sarebbe possibile inserire questi dati per via della foreign key della tabella Misurazione. L'idea viene dalla necessità di rendere il sistema flessibile. Non è quindi possibile escludere che in futuro il database possa aspettarsi dei dati che abbiano il campo "practitioner" ignoto. Ora questo campo viene forzato dal portale al momento dell'upload, ma non è una cosa scontata in futuro.

Una volta inseriti i dati ritorna il response al web server e infine arriva al browser dell'utente. In caso di batch viene elencato il risultato di tutti i file caricati, specificando se l'upload ha incontrato problemi o meno.

3.5.3 Download dei dati

Quando si vogliono scaricare dei dati ci si trova di fronte alla schermata di download, come in figura 3.54.

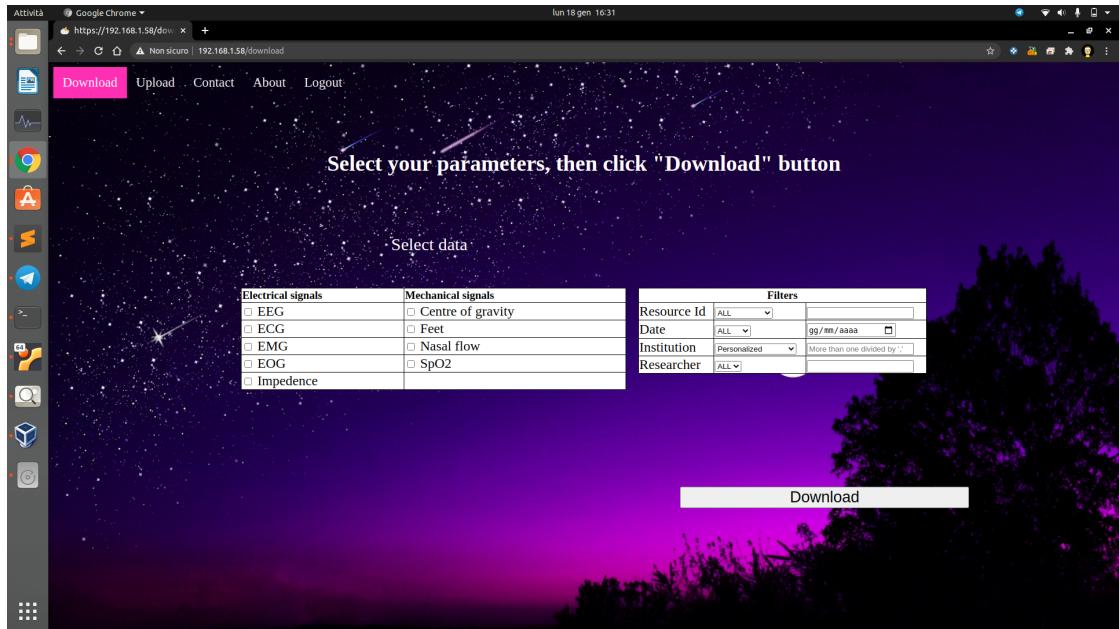


Figura 3.54: Interfaccia di download

Come si può notare, è costituita da due tabelle e un pulsante. La prima è la tabella che permette di selezionare i segnali desiderati divisi, come nel capitolo 1, in segnali elettrici e meccanici, mentre la seconda tabella contiene i filtri, ossia ciò che va a ridurre veramente il campo di ricerca. Una volta selezionati si clicca download e viene mandata la richiesta POST, come si diceva a proposito del sequence diagram, al web server.

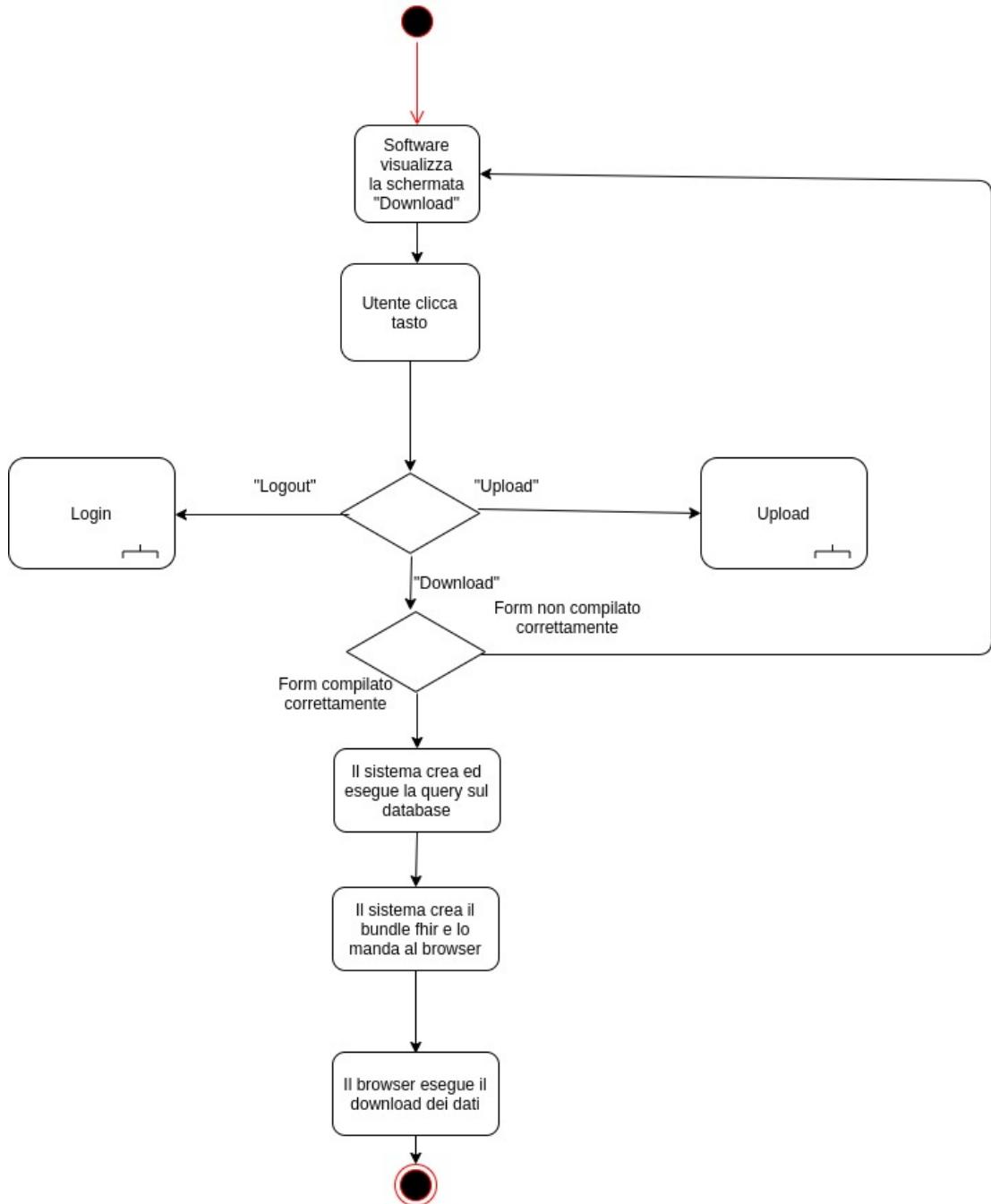


Figura 3.55: Activity Diagram del caso d'uso di download.

Il web server, una volta ricevuti i parametri costruisce in automatico la SEARCH request secondo lo standard FHIR e manda i dati a Mirth. Una volta che il messaggio arriva nel middleware ci si prepara a costruire la query in SQL. Di tutti i comandi

costruiti automaticamente durante il presente progetto di tesi questo è senza dubbio il più complesso. Per costruire la query si è quindi partiti dalla sua costituzione "base":

```
SELECT <attributi da selezionare>
FROM <tabelle da selezionare>
WHERE <discriminazioni sui parametri>
```

Partendo da questa assunzione si è ragionato cercando di creare le tre parti in modo indipendente, per poi unirle in un'unica stringa. Nella parte SELECT si procede andando a valutare quanti segnali sono stati richiesti. Se sono tutti si può benissimo porre "*" e andare al punto successivo. Se non se n'è scelto nemmeno uno si ritorna un errore. Esiste un terzo caso in cui non sono stati selezionati tutti i segnali, quindi bisogna aggiungere alla impostare la stringa del comando SELECT nel seguente modo:

```
SELECT Id, Data, UserId
```

A questa stringa vanno aggiunti i nomi dei segnali desiderati. Per fare ciò bisogna andare a scorrere quelli presenti per "tradurre" i codici nei nomi degli attributi usati nel database. Con questo ciclo *for* si coglie l'occasione per inizializzare anche una stringa da inserire nella parte "WHERE" e che stabilisca che almeno uno dei segnali esista, ossia che non sia null. Successivamente si costruisce la parte del "FROM", che è relativamente semplice avendo due tavelle: se viene richiesto un filtro sull'istituzione, allora bisogna inserire pure la tabella Ricercatore, perché si possa fare un join e prendere il valore. Dopo aver fatto ciò ci sono dei controlli che vengono effettuati per costruire la parte WHERE e infine si uniscono le tre stringhe.

Una volta effettuata la query si ricostruiscono le risorse andando a riprendere i segnali con i rispettivi percorsi e si costruisce la risorsa Bundle, quella che poi viene effettivamente mandata indietro. Una volta nel web server c'è una funzione per contare quanti segnali sono stati trovati. Se, per esempio, viene effettuata una query per avere tutti gli ECG ed EEG di uno specifico giorno, non è detto che ogni record contenuto nel database abbia entrambi, magari c'è una risorsa che ha solo EEG e un'altra che ha solo ECG. Quindi si conta quanti segnali effettivamente sono stati ottenuti, per poi mostrarli all'utente. Infine si avvia il download.

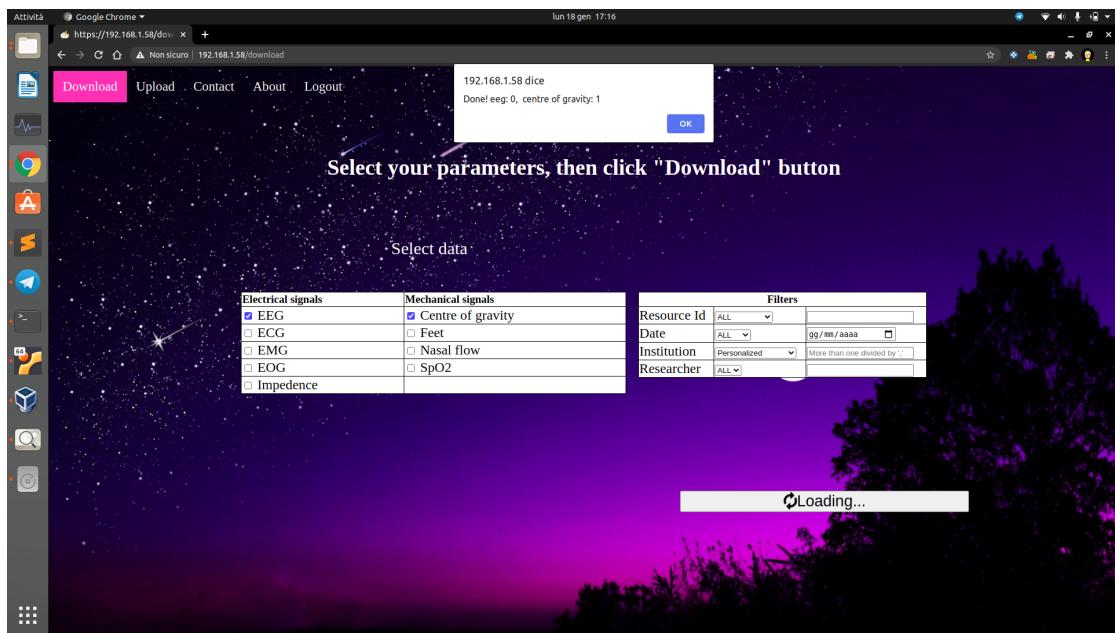


Figura 3.56: Interfaccia in fase di download.

Conclusioni e sviluppi futuri

L'obiettivo della tesi è permettere la raccolta e condivisione di dati sul sonno, al fine di aumentare le informazioni in possesso dei ricercatori per fare ricerca. Il progetto ambisce anche a essere abbastanza flessibile da permettere futuri ampliamenti, anche includendo l'interfacciamento verso altri sistemi informativi sanitari. A questo scopo si è prima cominciato a studiare il contesto, ossia l'obiettivo delle misurazioni, il tipo di misurazioni con i segnali coinvolti e infine si è studiato quali cambiamenti fisiologici porta con sé il sonno.

Capito ciò, è stato importante studiare gli strumenti ICT a disposizione, dai linguaggi di programmazione, di interrogazione e di markup, fino ai linguaggi per modellare i sistemi software. Si è parlato dei protocolli di comunicazione, dei modelli dei database e dei middleware, per tentare di mettere insieme tutti gli attori che possano essere utili. Per concludere gli strumenti ICT non si può non citare uno strumento squisitamente biomedico, ossia lo standard FHIR, con la nomenclatura SNOMED. Studiare gli standard dell'informatica medica è molto importante in un progetto di ingegneria biomedica, proprio perché permette di cogliere il problema principale del campo, ossia l'interoperabilità. Se un sistema informativo sanitario non può comunicare in modo standard diventa più difficile condividere le informazioni, venendo meno ad una parte dello scopo per cui il sistema è stato creato.

Vista l'importanza dell'interoperabilità uno degli sviluppi futuri più interessanti per il progetto è senza dubbio l'interoperabilità con altri sistemi informativi sanitari, come l'HIS¹³, permettendo l'introduzione delle figure "paziente" e "medico", come si è già pensato in fase di progettazione del database. In quest'ottica si potrebbe pensare di ampliare le possibilità di interazione con il web server tramite REST inviando in modo automatico i dati dai dispositivi. Un esempio pratico può essere la raccolta automatica di dati da un dispositivo con Wear OS¹⁴, dal momento che è

¹³"HIS" sta per "Hospital Information System", ossia "sistema informativo ospedaliero", ed è il sistema che coordina le funzioni amministrative, sanitarie e cliniche all'interno di una struttura ospedaliera comunicando con gli altri sistemi attraverso dei middleware.

¹⁴Wear OS è il nome che Google ha dato ad Android per smartwatch.

possibile esistono delle API per interfacciarsi con questi oggetti. Queste applicazioni possono essere sia per smartphone, che per smartwatch, qualora quest'ultimo sia in grado di connettersi ad internet. In realtà si potrebbe pensare anche alla realizzazione di altri software che funzionino su altri dispositivi. Un'altra idea, in un contesto del genere, potrebbe essere quella di integrare anche degli algoritmi di classificazione in cloud, magari utilizzando un terzo server su cui il web server scarichi il costo computazionale della classificazione. In un'ottica di questo tipo, diventa particolarmente interessante introdurre una interfaccia che permetta di far vedere al medico o al paziente stesso i risultati della computazione. Un'altra modifica può derivare dall'implementazione di più codici per la nomenclatura oltre a SNOMED, magari introducendo LOINC e ICD. Si possono aggiungere anche più formati in input, sia per il web server, sia per il middleware. Il primo potrebbe prendere in ingresso informazioni in HL7v2, HL7v3 ed FHIR in XML. Il secondo dovrebbe introdurre il supporto a FHIR in XML per raggiungere una buona *compliance* con lo standard. I formati potrebbero essere scelti in download andando, di fatto, a fare l'inverso di quando si fa l'upload, restituendo in input un EDF. Un altro sviluppo è senza dubbio l'introduzione di ulteriori metodi REST di FHIR. Per esempio si possono introdurre i metodi UPDATE e PATCH, per modificare delle risorse, o il metodo DELETE, per cancellarle. SleepingRepo nasce, come dice il nome, per contenere dati polisonnografici, per cui si potrebbe pensare di aumentare la tipologia di dati utilizzata, per esempio andando a integrare delle registrazioni video dei movimenti durante la notte. Infine si dovrebbe aumentare la sicurezza del sistema, dal momento che, trattandosi di dati biomedici, si sta comunque parlando di informazioni preziose. Sono già state implementate delle accortezze che possono garantire una discreta sicurezza, in particolare al web server. Un esempio può essere il blocco di codici Javascript non approvati a priori, oppure l'utilizzo di una connessione crittografata tramite SSL. Senza alcun dubbio tutto questo non basta, perché i file di configurazione non sono crittografati, non c'è un controllo sulla complessità della password e non si possono escludere a priori attacchi massivi DDoS¹⁵ o metodi di data breach più sofisticati. Sul versante sicurezza bisogna anche considerare che il middleware e il database sono al sicuro se vengono implementati tramite una rete propria LAN o una VPN dove l'unico "accesso" è il web server ma in un'ottica di ampliamento questo potrebbe non essere abbastanza, per cui potrebbe risultare necessario mettere in sicurezza anche quella parte.

¹⁵"DDoS" sta per "Distributed Denial of Service", ossia "Negazione distribuita del servizio", ed è un tipo di attacco informatico volto a sovraccaricare il server riempendolo di richieste che non può gestire.

Bibliografia

- [1] A.Bessani, J.Brandt, *BiobankCloud: a Platform for the Secure Storage, Sharing, and Processing of Large Biomedical Data Sets*, disponibile: https://www.researchgate.net/publication/280317196_BiobankCloud_a_Platform_for_the_Secure_Storage_Sharing_and_Processing_of_Large_Biomedical_Data_Sets, 2015
- [2] J.Freund, D.Comaniciu, *Health-e-Child: An Integrated Biomedical Platform for Grid-Based Paediatric Applications*, disponibile: https://www.researchgate.net/publication/6962204_Health-e-Child_An_Integrated_Biomedical_Platform_for_Grid-Based_Pediatrics, 2006
- [3] S.Kim, K.Lee, *Wearable Multi-Biosignal Analysis Integrated Interface With Direct Sleep-Stage Classification*, IEEE, 2019
- [4] J.Zhang, R.Yao, *Orthogonal convolutional neural networks for automatic sleep stage classification based on single-channel EEG*, disponibile: <http://www.ScienceDirect.com>, 2019
- [5] I.Ferrer-Lluis, Y.Castillo-Escario, *Analysis of Smartphone Triaxial Accelerometry for Monitoring Sleep-Disordered Breathing and Sleep Position at Home*, IEEE, 2017
- [6] L.Mesin, *Introduction to biomedical signal processing*, Politecnico di Torino, 2017
- [7] K.Mandrick, *Near-infrared Spectroscopy application for discriminating the mental workload in humans.*, disponibile: https://www.researchgate.net/publication/267507690_Near-infrared_Spectroscopy_application_for_discriminating_the_mental_workload_in_humans, 2013
- [8] F.Carpi, D.De Rossi, *Fenomeni Bioelettrici*, cap 18 disponibile: <http://www.centropiaggio.unipi.it/sites/default/files/course/material/12.EOG%20ENG%20ERG.pdf>, 2019
- [9] C.Redmond, *Transthoracic Impedance Measurements in Patient Monitoring.*, Analog Devices, 2013
- [10] Y.Jawahar, *Design Of An Infrared Based Blood Oxygen Saturation And Heart Rate Monitoring Device*, disponibile: <https://www.semanticscholar.org/paper/>

- Design-of-an-Infrared-based-Blood-Oxygen-Saturation-Jawahar/
aac1bbe96944018d040fec89dd64885d7cba9c0, 2009.
- [11] I.Imeri, F.Del Gallo, *Sonno e Malattie Neurodegenerative In Modelli Animali*, Università degli studi di Milano, 2013.
 - [12] J.Aschoff, R.Wever, *Human circadian rhythms: a multioscillatory system*. Fed. Proc. 35, 1976.
 - [13] W.C.Dement, N.Kleitman, *The relation of eye movements during sleep to dream activity: an objective method for the study of dreaming*. J. Exp. Psychol. 53, 1957.
 - [14] J.A.Hobson, *REM sleep and dreaming: towards a theory of protoconsciousness*, Nat. Rev. Neurosci. 10, 2009
 - [15] M.A.Carskadon, W.C.Dement, *Monitoring and staging human sleep*, Principles and practice of sleep medicine 5th edition, St. Louis: Elsevier Saunders, 2011
 - [16] L.Ghio, *Appunti di reti di calcolatori*, Politecnico di Torino, 2012.
 - [17] Wikipedia the free Encyclopedia, <http://it.wikipedia.org/wiki/Wiki>
 - [18] Flask Documentation, <https://flask.palletsprojects.com/en/1.1.x/>
 - [19] K.Fakhruddinov <https://www.uml-diagrams.org/>
 - [20] Documentazione ufficiale FHIR <https://www.hl7.org/fhir/>
 - [21] G.Sparks, *Enterprise Architect User Guide*, Sparx Systems, 2014.
 - [22] A.Ravarini, D.Sciuto, *Sistemi per la gestione dell'informazione*, Apogeo, 2008.
 - [23] M.Devanna, *Progetto SOLE Sanità Online*, Regione Emilia-Romagna, 2015.
 - [24] A.Silberschatz, P.B. Galvin, G.Gagne, *Operating System Concepts*, John Wiley & Sons Inc, 2008.
 - [25] A.Umar, *Object-Oriented Client/Server Internet Environments*, Pearson College Div, 1997.
 - [26] NextGen Connect 38 User Guide, NextGen Healthcare, 2019.
 - [27] G.Balestra, C.Castagneri, *Modelling and Analysis of Four Telemedicine Italian Experiences*, IEEE, 2017.
 - [28] Terminologia HL7 <https://terminology.hl7.org/>
 - [29] J.Kempfner, G.L.Sorensen, *Automatic REM Sleep Detection Associated with Idiopathic REM Sleep Behavior Disorder*, IEEE, 2011.
 - [30] B.Sheng, F.Chen, *Design of a Dual Quantization Electromechanical Sigma-Delta Modulator MEMS Vibratory Wheel Gyroscope*, disponibile: https://www.researchgate.net/publication/322348316_Design_of_a_Dual_Quantization_Electromechanical_Sigma-Delta_Modulator_MEMS_Vibratory_Wheel_Gyroscope, 2018.
 - [31] A.Tognetti, corso di biosensori, *Sensori Fisici*, disponibile: <https://www.centropiaggio.unipi.it/course/material/sensori-fisici-accelerometri-e-giroscopi>, 2019.

- [32] M.Liverani, *corso di Sistemi per l'elaborazione delle informazioni, Principi di ingegneria del software*, disponibile: http://www.mat.uniroma3.it/users/liverani/doc/IN530_9_IngegneriaSoftware.pdf, 2019.
- [33] K.Asher, L.Finn, *corso di Sistemi per l'elaborazione delle informazioni, Principi di ingegneria del software*, disponibile: http://www.mat.uniroma3.it/users/liverani/doc/IN530_9_IngegneriaSoftware.pdf, 2019.
- [34] F.Ferrero, A.Lopez *An Affordable Method for Evaluation of Ataxic Disorders Based on Electrooculography*, MDPI, 2019.
- [35] *Mirth Appliance Deployment Guide*, NextGen Healthcare, 2013.
- [36] H.Bahrstaedt, *PyEDFlib Documentation*, disponibile: <https://buildmedia.readthedocs.org/media/pdf/pyedflib/latest/pyedflib.pdf>, 2020.
- [37] N.Vanello, *corso Segnali e Trasduttori, Sistemi di Misura Inerziali*, disponibile: http://www.iet.unipi.it/n.vanello/SM0/sistemi_inerziali_2019.pdf, 2019.
- [38] M.J.Hayes, P.R.Smith, *A New Method for Pulse Oximetry Possessing Inherent Insensitivity to Artifact*, IEEE, 2001.
- [39] F.Reali, *SECURE SOCKET LAYER*, disponibile: <http://www.dmi.unipg.it/~bista/didattica/sicurezza-pg/seminari2009-10/SSL.pdf>, 2009.
- [40] G.Destri, *corso di Ingegneria del Software, Il linguaggio UML*, disponibile: http://www.giuliodestri.it/doc/ingsw/S03_UML.pdf, 2006.
- [41] *UML Models*, Sparx Systems, 2017.
- [42] P.Diez, *Smart Wheelchairs and Brain-computer Interfaces, cap 16*, Academic Press, 2018.
- [43] B.Bergh, M.Birkle, *Architecture of a consent management suite and integration into IHE-based regional health information networks*, disponibile: <https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-11-58>, 2011.
- [44] J.Karlsson, O.Trelles, *MAPI: a software framework for distributed biomedical applications*, disponibile: <http://www.jbiomedsem.com/content/4/1/4>, 2013.
- [45] T.Ryan Burchfield, S.Venkatesan, *Accelerometer-Based Human Abnormal Movement Detection in Wireless Sensor Networks*, disponibile: <https://dl.acm.org/doi/abs/10.1145/1248054.1248073>, 2007.
- [46] M.Chen, C.Tsai, *Web-Based Experience Sharing Platform on Medical Device Incidents for Clinical Engineers in Hospitals*, disponibile: <https://link.springer.com/article/10.1007/s40846-018-0441-7>, 2018.
- [47] J.Clarkson, G..T.Jun, *Health care process modelling: which method when?*, International Journal for Quality in Health Care, 2009.
- [48] J.Clark, P.C.van Oorschot, *SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements*, IEEE, 2013.

- [49] D.Bender,K.Sartipi, *HL7 FHIR: An Agile and RESTful Approach to Healthcare Information Exchange*, IEEE, 2013.
- [50] G.Canal, G.Sparancino, *Interoperabilità in Sistemi Informativi Sanitari realizzata attraverso piattaforma Mirth Connect*, disponibile: <http://tesi.cab.unipd.it/49617/>, 2015.
- [51] M.Knaflitz, *Bioingegneria Elettronica e Sicurezza*, LEVROTTO & BELLA, 2018.
- [52] V.T.van Hees, S.Sabia, *Estimating sleep parameters using an accelerometer without sleep diary*, Nature, 2018.
- [53] Z.Beattie, Y.Oyang, *Estimation of sleep stages in a healthy adult population from optical plethysmography and accelerometer signals*, Physiol. Meas. 38 1968, 2017.
- [54] M.Cesari, J.A.E.Christensen, *A data-driven system to identify REM sleep behavior disorder and to predict its progression from the prodromal stage in Parkinson's disease*, disponibile: <https://doi.org/10.1016/j.sleep.2020.04.010>, 2019.
- [55] N.Cooray, F.Andreotti *Detection of REM sleep behaviour disorder by automated polysomnography analysis*, disponibile: <https://www.sciencedirect.com/science/article/pii/S1388245719300306?via%3Dihub>, 2019.
- [56] <https://www.python.org/>
- [57] *Securing CherryPy*, <https://docs.cherrypy.org/en/3.3.0/progguide/security.html>
- [58] *Documenti Robot Sphero* <https://sphero.docsapp.io/docs/get-started>

Ringraziamenti

Questo lavoro di tesi è stato il mio primo progetto fatto interamente da me, nessuno che nessuno mi dicesse cosa dovessi fare. Anche se mi sono dovuto confrontare con ogni singola parte dell'intero lavoro, non posso assolutamente dire che sono stato solo. Devo dire grazie a molte persone, perché gliene sono debitore. Il mio primo ringraziamento va alla prof.ssa Gabriella Olmo, all'ing Irene Rechichi e alla Astel, in particolare non posso non ringraziare il dott.Paolo Astengo e l'ing Camilla Cervetto, che ha riletto questa tesi infinite volte, per renderla il migliore possibile. Questo lavoro di tesi ha visto la luce grazie a tutti che avete creduto in me e mi avete supportato sempre.

Esteriormente ai miei relatori/correlatori non posso non ringraziare l'ing Mauro Tiberi, che mi segue un po' come una guida spirituale sin dalla laurea triennale. È grazie a lui se ho avuto alcune conoscenze di base per poter portare a termine la tesi. Devo dire grazie alla mia ragazza, Elena, che mi ha sopportato e supportato quando non pensavo di potercela fare continuando tutt'ora a farlo. Grazie veramente, per rendere ogni difficoltà semplice.

In realtà un'altra persona che ha dovuto sopportarmi in questo periodo c'è: si tratta di Igor. A lui, com'è stato anche per Elena, va un ringraziamento che trascende il lavoro di tesi, perché abbiamo condiviso ogni momento dell'esperienza della laurea magistrale e buona parte di quella triennale.

Non posso non ringraziare Nicola a cui ho chiesto consulenze informatiche quando le mie capacità erano traballanti. Durante la tesi una persona che ha vissuto con me si è rivelata importante: Valerio. Quindi almeno un grazie gli è dovuto. Dopotutto Valerio e Camilla mi hanno accompagnato in questo periodo di tesi a Ivrea, quindi grazie ancora ragazzi.

Spostandosi indietro nel tempo devo assolutamente ringraziare i miei coinvolti: oltre al già citato Igor, la mia gratitudine va a Pierpaolo, Antonio e Nicole, che è come se fosse una nostra coinvoltina. Grazie a voi la nostra vita a casa era sempre piena di colori di vita e così familiare da farmi adorare il tempo passato insieme. Tutta la "gang" di Ancona ha la mia gratitudine, per tutti i bei momenti condivisi insieme. Un grazie enorme a Leonardo, perché, oltre a essere estremamente legati personalmente, pure i nostri percorsi sono legati da anni e questo per me conta

moltissimo, perché mette sempre un punto fermo nella mia vita: spero vivamente che continui a essere così. Ringrazio tutti quelli con cui io e Igor (e successivamente Elena) abbiamo collaborato nei laboratori durante questi due anni: c'è sempre qualcosa da imparare. In particolare senza Giorgia e Sofia i primi periodi a Torino io e Igor saremmo stati persi.

Un grazie enorme va alla mia famiglia, alle mie salde radici che mi permettono di andare sempre avanti. Dopotutto, come una pianta, senza radici non è possibile far nascere le foglie e far maturare i frutti. Un grazie gigantesco ai miei genitori che mi supportano sempre, anche se sono lontano, e che mi hanno reso quello che sono. Grazie ai cugini che sono come fratelli per me e grazie agli zii e ai nonni che ci sono sempre. Vi voglio bene.

Grazie a Martin, che è sempre stata la prima persona che ho rivisto appena tornavo a casa dopo i miei genitori e grazie a tutti gli amici di Gualdo Tadino. È sempre bello tornare a casa, perché so che voi ci siete. Grazie alla mia squadra di nuoto di quando ero adolescente. Grazie ad ogni insegnante che ho avuto. Un ulteriore grazie a chi, tra questi, continua ancora a supportarmi. Grazie a tutte le persone che ho conosciuto, non è importante che le abbiate amate o odiate, perché qualsiasi persona che si incontra ci lascia una parte di sé, come insegnamento.

Infine grazie anche a te, chiunque tu sia, che stai leggendo queste righe. Grazie per la pazienza per aver letto il frutto del mio lavoro che corona uno dei periodi più belli della mia vita. Questi anni rimarranno per sempre scolpiti nella mia memoria.