ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Dipartimento di Informatica - Scienza e Ingegneria
Corso di Laurea in Ingegneria e Scienze Informatiche

PROGETTAZIONE E SVILUPPO DI UN SISTEMA PER LA RACCOLTA E TRACCIAMENTO DI DATI IN AMBITO OSTETRICO

Elaborato in
SISTEMI EMBEDDED E INTERNET-OF-THINGS

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Dott.ssa SARA MONTAGNA

Presentata da
ANTONIO CHRISTIAN TOSCANO

Terza Sessione di Laurea
Anno Accademico 2017 – 2018

PAROLE CHIAVE

CERERE

Ospedale 4.0

Tracciamento

REST

Android

Alla mia famiglia, senza la quale questo risultato non sarebbe stato possibile.
A mio nonno, che avrebbe voluto essere presente in questo momento.

Indice

Introduzione

L'enorme progresso tecnologico degli ultimi anni ha portato a una vera e propria quarta rivoluzione industriale con grandi cambiamenti nella vita quotidiana di miliardi di persone in tutto il mondo che hanno visto migliorate le proprie condizioni di vita e di lavoro.

In particolar modo tecnologie informatiche come Intelligenza Artificiale, Realtà Aumentata e Internet of Things hanno portato importanti cambiamenti in molti settori industriali attraverso l'automatizzazione nella risoluzione di problemi, la raccolta e la gestione delle informazioni, attraverso la creazione di sistemi informatici pervasivi che assistono l'utente nelle proprie mansioni e interagiscono attivamente con esso.

Fra i settori che potrebbero trarre maggior beneficio dalle nuove evoluzioni tecnologiche vi è quello sanitario che potrebbe aumentare la propria efficienza nella cura del paziente, diminuendo allo stesso tempo i costi, gli sprechi e gli errori. Per questo motivo è nata negli ultimi anni la visione di Ospedale 4.0 che cerca di applicare i progressi informatici nel campo sanitario e in particolar modo nell'assistenza ospedaliera. In quest'ottica si inserisce il progetto CERERE, nato dalla collaborazione tra un gruppo di ricerca del Dipartimento di Informatica - Scienza e Ingegneria (DISI) e l'Ospedale Bufalini di Cesena, con il coinvolgimento delle Unità Operative di Analgesia e Rianimazione, e Ostetricia e Ginecologia. Il progetto si pone l'obbiettivo di fare chiarezza sulla correlazione tra l'utilizzo di trattamenti di analgesia epidurale ed eventuali complicanze di parto delle pazienti, con lo scopo clinico di offrire dei percorsi di prevenzione dedicati a ogni paziente che possano agevolare l'assistenza ed evitare le complicanze di parto. Per arrivare a questo risultato il progetto propone il progetto di un'infrastruttura tecnologica che supporti quotidianamente il personale medico nel tracciamento dei valori clinici della donna al fine di raccogliere il maggior numero di informazioni possibili. Il progetto propone l'adozione di algoritmi di Machine Learning per eseguire un'analisi retrospettiva dei dati raccolti. Allo scopo verranno utilizzati diversi dispositivi, mobile e non, ponendo il progetto in un contesto più ampio di Ospedale 4.0 che potrà evolvere in futuro con nuove e più pervasive funzionalità.

Il lavoro di tesi si pone come obbiettivo la progettazione e lo sviluppo di un

prototipo del sistema che rispetti i requisiti posti dal progetto CERERE e che sia il primo passo verso un percorso di completa digitalizzazione nella gestione delle informazioni delle pazienti relativamente al travaglio di parto. Questo ha comportato un accurato studio del dominio e dei suoi vincoli così come l'interazione diretta con i medici che dovranno utilizzare realmente il sistema al fine di comprende a pieno il contesto sanitario ma anche tecnologico in cui ci si trova. Nel seguito della tesi spiegherò il mio contributo nella realizzazione del sistema discutendo i requisiti del dominio e illustrando quali soluzioni ho adottato per rispettarli, attraverso le diverse fasi della progettazione del software. Pur trattandosi di un prototipo, che dovrà in seguito essere completato per entrare realmente in funziona nel contesto ospedaliero, sono convinto che gli impatti potrebbero essere notevoli: il coinvolgimento dei medici potrebbe essere un punto di svolta nel loro approccio alla tecnologia, mostrandogli quali siano le possibilità di assistenza messe in campo dall'informatica e incentivando quindi una più veloce adozione delle tecnologie moderne.

Capitolo 1

Ospedale 4.0 e il Progetto CERERE

L'enorme sviluppo della così detta Industria 4.0, generata da una vera e propria quarta rivoluzione industriale, ha portato all'affermazione di nuove tecnologie e nuovi approcci per la risoluzione di problemi reali fino a qualche anno fa impensabili. Attraverso la tendenza all'automazione, alla previsione dei comportamenti e dei risultati ci si pone l'obbiettivo di migliorare la qualità della vita e la produttività di una grossa fetta di popolazione.

In questo capitolo verranno spiegati gli strumenti messi a disposizione dall'Industria 4.0 e in particolar modo le prospettive della loro applicazione nel settore sanitario per poi concentrarsi meglio sul caso specifico del progetto CERERE.

1.1 Industria 4.0

Il concetto di Industria 4.0 è davvero molto amplio e lo si può capire da come questa rivoluzione tecnologica abbia cambiato e stia cambiando la nostra vita nella quotidianità.

Partendo dalle smart house fino alle automobili a guida autonoma le potenzialità sono enormi e forse ancora da scoprire.

I capi saldi di questo sviluppo tecnologico sono:

- Realtà aumentata
- Big Data
- Intelligenza artificiale
- Pervasive computing & Internet of Things

1.1.1 Big Data

Con il termine Big Data (letteralmente "grandi quantità di dati") si fa riferimento alla capacità della scienza dei dati di estrapolare, analizzare e mettere in relazione tra loro enormi quantità di dati eterogenei e non, strutturati e non, allo scopo di mettere in luce i legami tra fenomeni diversi tanto da poterli prevedere in futuro.

La definizione stessa di Big Data è soggetta a diverse opinioni da parte della letteratura: ci si chiede infatti oltre quale quantità un insieme di dati possa essere definito "big". Uno dei punti di riferimento a riguardo è il modello delle "3V" spiegato dall'analista Doug Laney che indica in Volume, Varietà e Velocità le tre caratteristiche che definiscono i Big Data. Secondo un'opinione più generale invece si può definire Big Data qualsiasi insieme di dati così grande da non poter più essere gestito da un database relazionale ma che richieda piuttosto l'impiego di tecnologie di tipo NoSQL.

Questa enorme mole di dati viene generata ogni giorno dalle piattaforme di social network e di blogging, dai siti di social news e di condivisione di file multimediali così come dai siti di recensioni e portali di e-commerce.

E' chiaro quindi l'interesse da parte delle aziende di tutto il mondo nello studiare le abitudini dei propri clienti per arrivare a prevederne i comportamenti con lo scopo di rispondere ad un mercato sempre più fluido e mutevole.

1.1.2 Intelligenza Artificiale

Con il termine Intelligenza Artificiale, spesso abbreviata in AI, si indica l'abilità di un sistema tecnologico di risolvere problemi o svolgere attività imitando i meccanismi di pensiero razionale e logico caratteristici della mente umana.

Lo sviluppo di questa branca dell'informatica ha subito alti e bassi dal momento della sua nascita datata all'anno 1943 quando fu creato il primo neurone artificiale. Tuttavia negli ultimi anni i progressi nel campo hardware e il sempre decrescente costo di questi componenti hanno permesso di avere gli strumenti adatti per emulare le funzioni sensoriali e cognitive del cervello umano attraverso un'elevata potenza computazionale e algoritmi appositi.

L'Intelligenza Artificiale si fonda su quattro diversi livelli funzionali:

- **comprensione**: il riconoscimento di testi, immagini, video e voce attraverso la simulazione di capacità cognitive di correlazione dei dati

- **ragionamento**: il collegamento di molteplici informazioni ottenuto applicando ragionamenti logici

- **apprendimento**: la capacità di apprendere di una AI e quindi di specializzarsi in un determinato campo di analisi di un input per la corretta restituzione di un output attraverso algoritmi di Machine Learning

- **interazione**: la capacità dell'AI di interagire con l'uomo attraverso il linguaggio naturale utilizzando sistemi di Natural Language Processing

1.1.3 Pervasive Computing & Internet of Things

Il Pervasive Computing si configura come l'opposto della tradizionale concezione di computer "desktop", ovvero da scrivania, in cui l'utente utilizza il singolo dispositivo per un compito specifico. Con il termine di Pervasive Computing si indica infatti un nuovo approccio in cui gli oggetti intorno a noi svolgono delle attività, in maniera pervasiva, che supportano e migliorano la nostra vita per creare un ambiente intelligente che ci circonda e agisce attivamente. Da qui la possibilità di creare sistemi di calcolo distribuito, intelligenza ambientale, mobile computing e *Internet of Things* (IoT).

Con il termine Internet of Things (letteralmente"internet delle cose") si indica l'approccio del rendere "smart" tutto ciò che ci circonda, creando degli **smart objects** che possono comunicare tra di loro attraverso una connessione, conservare i dati e assisterci più o meno attivamente. Tutto questo non riguarda solo i dispositivi tecnologici di ultima generazione ma anche quelli già esistenti: utilizzando sensori, moduli hardware per la connessione internet, sistemi con Arduino[1] e Raspberry Pi[2] si possono attribuire nuove funzionalità a dispositivi già esistenti per creare un ecosistema tecnologico.

Nel complesso questo approccio apre le porte per la creazione di:

- **Smart city**: città che interagiscono e semplificano la vita dei propri abitanti

- **Smart house e domotica**: case intelligenti e controllabili dallo smartphone

- **Smart factory**: fabbriche e luoghi di lavoro intelligenti che migliorano la qualità della vita e la sicurezza dei propri lavoratori

- **Ospedale 4.0**: ospedali migliori per cittadini e medici che vi lavorano

[1] **Arduino** è una piattaforma hardware composta da una serie di schede elettroniche dotate di un microcontrollore.

[2] **Raspberry Pi** è un single-board utilizzato in ambiti embedded e IoT.

In ognuno di questi contesti è necessario che ogni dispositivo sia identificabile in maniera univoca e abbia la capacità di inviare/scambiare dati, per questo risultano ampiamente utilizzata tecnologie wireless[3] quali:

- **Tag RFID**, ovvero Radio-Frequency IDentificatio, sono dei piccolo chip dotati di antenna con cui si può comunicare attraverso le onde radio. Il tag può contenere delle informazioni che permettono di identificarlo univocamente e di associarlo ad un oggetto o luogo e funziona in maniera passiva, aspettando quindi di essere riconosciuto da un lettore. Esistono molti tipi di tag RFID che si distinguono per velocità di trasmissione, capacità di memoria e distanza nella ricezione delle onde radio.

- **Tag NFC**, ovvero Near Field Communication, sono dei piccoli chip che nascono come evoluzione dei tag RFID e sono perciò utilizzati per le comunicazioni senza fili a corto raggio. La caratteristica del protocollo NFC è di creare una comunicazione bidirezionale tra il tag e il lettore che possono quindi scambiarsi dati. Esistono molti tipi di tag NFC che si distinguono principalmente per la memoria a disposizione e per la distanza della comunicazione, che può arrivare a massimo 10 cm nei tag più costosi.

- **Bluetooth** è lo standard nelle comunicazioni senza fili nel contesto consumer ma anche in alcuni contesti industriali. Il protocollo è caratterizzato da una architettura master/slave[4] e utilizza una frequenza radio sicura e a corto raggio per permettere a due dispositivi di comunicare scambiandosi anche grandi quantità di dati. Permette di identificare ogni dispositivo in maniera univoca attraverso l'UUID[5] ad esso associato.

- **iBeacon** si basa sul Bluetooth Low Energy (BLE), è una tecnologie utilizzata per la localizzazione in contesti indoor, infatti attraverso l'UUID del protocollo Bluetooth è possibile riconoscere ogni dispositivo e a che distanza esso si trova.

- **Wi-Fi** è una tecnologia dedicata ai dispositivi basati sullo standard **IEEE 802.11** per la comunicazione senza fili . Grazie al Wi-Fi è possibile creare un contesto di interoperabilità tra svariati tipi di dispositivi

[3]Il termine **wireless**, dall'inglese senza fili, indica una comunicazione tra dispositivi elettronici che non fa uso di cavi.

[4]L'architettura **master/slave** è un modello di comunicazione dove un dispositivo (master) ha un controllo unidirezionale su uno o più dispositivi (slave) , equiparabile al modello client-server di internet.

[5]Con **UUID** si indica l'identificativo unico bluetooth caratteristico che permette di identificare univocamente ogni dispositivo che utilizza il protocollo Bluetooth.

di ogni genere come smartphone, computer ma dispositivi embedded. Il protocollo Wi-Fi utilizza delle onde radio con un raggio di funzionamento e una velocità di trasmissione maggiori rispetto al Bluetooth e per questo è più esoso in termini di utilizzo di batteria.

1.2 Benefici in ambito sanitario

Parlando di Industria 4.0 applicata all'ambito sanitario, e quindi ospedaliero, ci si riferisce al concetto di Ospedale 4.0: una nuova visione di ospedale che applica gli ultimi approcci tecnologici con lo scopo di minimizzare i costi ma allo stesso tempo aumentare l'efficienza e la qualità dei servizi forniti ai cittadini.

Nonostante in Italia non vi sia un piano nazionale di ammodernamento dell'assistenza sanitaria attraverso l'applicazione delle nuove tecnologie, si può notare come negli ultimi anni si sia diffusa l'esigenza di andare in questa direzione e di conseguenza molte strutture ospedaliere si stiano singolarmente adeguando ai canoni tecnologi moderni per offrire servizi migliori ai pazienti e un ambiente lavorativo più semplice ai medici.

Ne è un esempio l'impegno di molte regioni, tra cui appunto la regione Emilia-Romagna che risulta essere una delle più attive in questo campo, di adottare a pieno titolo il Fascicolo Sanitario Elettronico (FSE) e la Cartella Clinica Elettronica (CCE) muovendosi verso una visione di informazioni cliniche dematerializzate e costruendo un punto unico di condivisione e aggregazione delle informazioni rilevanti e di tutti i documenti sanitari e socio-sanitari relativi al cittadino. Notevoli anche gli sforzi della regione Trentino-Alto Adige che punta a centralizzare il workflow sanitario del paziente come spiegato sul proprio sito di riferimento[1].

A cercare di fare chiarezza nello sviluppo di una sanità digitalizzata è la stessa *ENISA*[2] (European Network and Information Security Agency) che nel 2016 individua i seguenti punti chiave:

- **Cyber Resilience:** la capacità dell'ospedale di adattarsi ai cambiamenti digitali sfruttando al meglio le tecnologie del momento ma non dipendendo esclusivamente da esse.
- **Affidabilità:** l'ospedale deve utilizzare le sue capacità digitale per fornire maggiore affidabilità al cittadino, sia nella gestione dei dati clinici che nel supporto al paziente.
- **Maggiore sicurezza del paziente:** la digitalizzazione dell'ospedale deve portare a una migliore assistenza, sfruttando ad esempio il supporto delle macchine per il monitoraggio e la raccolta dei dati del paziente.

- **Assistenza medica remota:** la capacità di un ospedale di fornire un supporto continuo ai pazienti anche fuori dalla propria struttura attraverso l'utilizzo di tecnologie IoT impiantabili, indossabili e mobile.

- **Continuità dell'assistenza e tempi di attesa migliori:** l'utilizzo di avanzate tecnologie e tracciamento dei dati dovrebbe andare verso la creazione di un workflow sanitario unico, riduzione degli errori e miglioramento l'efficienza delle cure fornite.

Come si evince da questo e da molti altri documenti il contesto di Ospedale 4.0 o Smart Hospitals (in una visione più internazionale) è molto ampio e le possibili strade da intraprendere sono numerose e diverse tra loro.

Big Data e Intelligenza Artificiale. Il tracciamento e la raccolta delle informazioni cliniche dei pazienti permettono di creare collezioni di dati enormi, entrando a pieno titolo nella definizione di Big Data. Tuttavia la raccolta dei dati è soltanto il primo passo verso una nuova concezione di assistenza sanitaria: i dati devono essere analizzati affinché abbiano davvero valore e impatto sulla vita dei futuri pazienti.

A questo compito si è dedicato negli ultimi anni un ampio settore dell'Intelligenza Artificiale applicata proprio al contesto dell'assistenza sanitaria grazie agli ultimi importanti risultati ottenuti dalle grandi aziende come Google, IBM, Microsoft per quanto concerne l'individuazione precoce delle malattie e in particolar modo delle cellule tumorali, la diagnosi e il trattamento delle malattie.

Fra i principali progetti riconosciamo:

- **DeepMind**[4], creata da Google, è una AI che si occupa di pianificare il trattamento radioterapico dei pazienti con risultati persino migliori di quelli ottenuti dai medici del settore

- **LYNA**[5], creata da Google ma meno conosciuta, è una AI specializzata nel riconoscimento precoce del cancro al seno con risultati persino migliori del 99% rispetto allo stesso esame condotto da soli medici

- **Watson**[3], creata da IBM, è una AI che permette di migliorare le diagnosi, le possibilità di cura e ridurre i falsi positivi. Può essere inoltre applicata a campi diversi, tra cui l'interpretazione e la comunicazione in linguaggio naturale.

Esistono migliaia di altre aziende in tutto il mondo che si occupano di sviluppare nuove forme di Intelligenza Artificiale in questo ambiente. Quella che fino a qualche anno fa sembrava una visione futuristica sta lentamente divenendo una

realtà tanto che secondo alcuni studi a riguardo nei prossimi anni l'AI sarà una componente fondamentale nella cura di alcune patologie o trattamenti sanitari.

Internet of Things e Realtà Aumentata Il concetto di Internet of Things applicato al contesto ospedaliero ha un impatto così grande da modificare le visione stessa di ospedale inteso come "edificio" e modificando del tutto l'insieme di tecnologie e processi interni ad esso.

Questa visione non riguarda soltanto gli ospedali in costruzione o ancora in fase di progettazione che devono, per forza di cose, avere una visione più ampia dell'ambiente ospedaliero futuro ma anche gli ospedali già esistenti che guardano con entusiasmo a queste innovazioni tecnologiche.

L'idea è infatti di creare un ecosistema intelligente che coinvolga anche le macchine ospedaliere e all'interno del quale i dispositivi possano comunicare per scambiarsi informazioni.

Un possibile utilizzo dell'IoT unito alla Realtà Aumentata riguarda l'impiego di **dispositivi wearable**, come gli smart glass, durante gli interventi chirurgici sul paziente.

Il più celebre esempio di smart glass sono sicuramente i Google Glass, un modello di occhiali dotati di realtà aumentata sviluppati da Google e che proprio in Italia sono stati utilizzati da Humanitas[6] per testare empiricamente la reale utilità che questo genere di tecnologia può offrire al campo sanitario.

L'interazione con gli occhiali avviene attraverso i comandi vocali e permette ai medici di monitorare i valori vitali e di avere altre informazioni utili senza disturbare il loro campo visivo durante l'operazione. Nel caso dei Google Glass è anche possibile scattare delle foto o registrare dei video che visualizzino esattamente ciò che vede il medico al fine di documentare in maniera completa tutto l'intervento per analizzare a posteriori i dati e fornire una migliore assistenza medica al paziente.

Un altro possibile sviluppo in quest'ambito riguarda il tracciamento e la raccolta delle informazioni del paziente, in maniera automatica, coinvolgendo tutti i dispositivi con cui il medico e il paziente stesso interagiscono all'interno della struttura. Questo implica anche il monitoraggio dei valori clinici del paziente che può così ricevere una migliore assistenza in real-time e una semplificazione nelle mansioni dei medici che vengono aiutati da dispositivi intelligenti. Questo approccio risulta particolarmente utile in contesti di mobilità del personale medico ma anche in situazioni di emergenza e di vincoli di tempo stringenti in cui il monitoraggio continuo del paziente è di vitale importanza. La raccolta dei dati si rivela un'importante risorsa per la seguente fase di analisi, attraverso l'utilizzo dell'Intelligenza Artificiale, che permette di effettuare un'indagine manageriale sui processi ospedalieri ma anche un'indagine clinica sulle meccaniche di assistenza e sul decorso dei pazienti. In questo contesto di

raccolta e successiva analisi dei dati clinici si pone il progetto CERERE, come verrà spiegato in seguito.

1.3 CERERE

Fra le strutture ospedaliere che stanno seguendo la visione di Ospedale 4.0 si contraddistingue l'Ospedale Bufalini di Cesena che da alcuni anni e con diversi progetti sta puntando a migliorare la propria assistenza al paziente.

Ne è un esempio il progetto *CERERE*, acronimo di **maChinE leaRning in travaglio di parto con e senza analgEsia spino-periduRalE**, che si sviluppa in un contesto di collaborazione tra l'ospedale, in particolare le Unità Operative di Analgesia e Rianimazione, Ostetricia e Ginecologia, e l'Università di Bologna attraverso il gruppo di ricerca del Dipartimento di Informatica - Ingegneria E Scienze Informatiche (DISI) della sede di Cesena.

1.3.1 Motivazioni del progetto

Il periodo di travaglio è un momento molto delicato per le pazienti perciò l'impostazione di un piano di assistenza appropriato e la precoce individuazione delle complicanze possono fare la differenza.

Tuttavia il 14esimo Rapporto CedAP (Certificato di Assistenza al Parto) redatto dalla Regione Emilia-Romagna risulta in questo contesto discordante con i pareri della letteratura per quanto riguarda la correlazione tra l'utilizzo di trattamenti di analgesia epidurale e un outcome materno negativo, inteso come:

- espletamento di un parto operativo (Ventosa)
- espletamento di un taglio cesareo (TC)
- parto complicato da emorragia (perdite ematiche maggio di 500 ml per parto vaginale e maggiori di 1000 ml per TC)
- parto con altre complicanze immediate materno-fetali non attese

Nasce quindi l'esigenza di fare chiarezza sulla relazione causa-effetto esistente tra questi dati per effettuare una migliore **valutazione del rischio** della donna in occasione del parto con lo scopo di offrire percorsi assistenziali specifici e differenziati a seconda del profilo della paziente che deve partorire.

1.3.2 Obiettivi del progetto

L'obiettivo del progetto è quello di cercare una correlazione tra un outcome materno negativo e alcuni valori clinici e trattamenti a cui è sottoposta la paziente durante il periodo di travaglio. In seguito, grazie ai risultati di questa analisi dei dati sarà possibile creare percorsi di prevenzione e controllo del rischio diversi per ogni paziente.

Collocandosi in una visione di Ospedale 4.0 il precedente obiettivo clinico passa dal più ampio obiettivo ingegneristico di realizzare un database condiviso tra anestesisti, ostetrici e ginecologi in cui raccogliere il numero maggiore di fattori multidisciplinari (materni, ostetrici, anestesiologici, fetali) da analizzare a posteriori con innovativi algoritmi di Intelligenza Artificiale.

A questo scopo il progetto si propone di esplorare l'uso di tecnologie mobili supportate da opportuni servizi lato server all'interno delle infrastrutture di rete dell'ospedale.

In una prima fase sarà necessario lo sviluppo di un sistema per la raccolta dei dati e delle rilevazioni real-time legati al travaglio di parto attraverso tecnologie relative all'IoT e al pervasive computing.

La seconda fase comporterà l'analisi dei dati raccolti grazie all'utilizzo di algoritmi nati nell'ambito dell'Intelligenza Artificiale e che fanno riferimento al Machine Learning.

Capitolo 2

Analisi dei requisiti

L'analisi dei requisiti è una fase fondamentale per la progettazione di ogni tipo di infrastruttura o servizio informatico. In questo caso l'analisi è il risultato dell'interazione diretta con i medici dell'Unità Operativa di Analgesia e Rianimazion attraverso diversi incontri in cui sono stati raccolti i dati descrittivi del sistema e delle esigenze reali del personale che dovrà in seguito utilizzarlo.

La raccolta dei dati ha permesso la piena comprensione del problema su cui si intende lavorare e la generazione di diversi documenti riassuntivi che sono stati utilizzati nelle successive fasi di progettazione.

2.1 Requisiti

Il luogo in cui si svolgono le meccaniche del dominio include piani distinti dell'ospedale Bufalini di Cesena, in particolare il quarto e il quinto su cui operano le rispettive Unità Operative di Analgesia e Rianimazione, Ostetricia e Ginecologia. Nello specifico, il reparto di ostetricia ospita quattro sale parto in ciascuna delle quali viene ospitata una paziente per l'espletamento del parto naturale, più una sala operatoria per l'eventuale esecuzione del taglio cesareo.

Gli utenti. Il dominio è composto da diversi tipi di utenti:

- n. 1 ostetrica per sala parto, quindi per un massimo di 4 ostetriche
- n. 1 ginecologo (opzionale)
- n. 1 anestesista (opzionale)

Il numero fa riferimento al personale attivo nello stesso momento per l'assistenza alla donna.

Un'ostetrica può seguire una sola paziente e nel caso in cui il travaglio dovesse durare troppo a lungo sarebbe sostituita dalla collega del cambio turno.

La presenza del ginecologo è opzionale ed è legata al decorso del travaglio, così come la presenza del personale anestesista. Quest'ultima però può essere esplicitamente richiesta dalla paziente in un momento antecedente al ricovero, come verrà spiegato in seguito nella sezione 2.1.1. Si stima infatti, secondo i dati in possesso dell'ospedale, che il numero di pazienti che utilizza i servizi di analgesia sia circa il 30%.

Il personale ostetrico opera esclusivamente sul proprio piano e in particolare all'interno delle sale parto. Il personale anestesista deve invece muoversi tra diverse sale e piani dell'ospedale, ad esempio recandosi periodicamente nelle sale parto per effettuare delle rilevazioni e somministrare dei farmaci alle pazienti.

2.1.1 Partoanalgesia

La parte di maggiore interesse riguarda l'area di partoanalgesia e in particolar modo il report cartaceo ad essa correlato poiché raccoglie tutte le informazioni cliniche riguardo il ricovero della paziente, tra cui i trattamenti di analgesia epidurale ricevuti da quest'ultima.

Il report è diviso in più schede che riprendono le due fasi operative:

Prima fase: Prenotazione. La paziente che intende ricevere i trattamenti di analgesia epidurale deve recarsi in ospedale alla 36esima settimana di gravidanza per manifestare la propria volontà di ricevere i trattamenti.

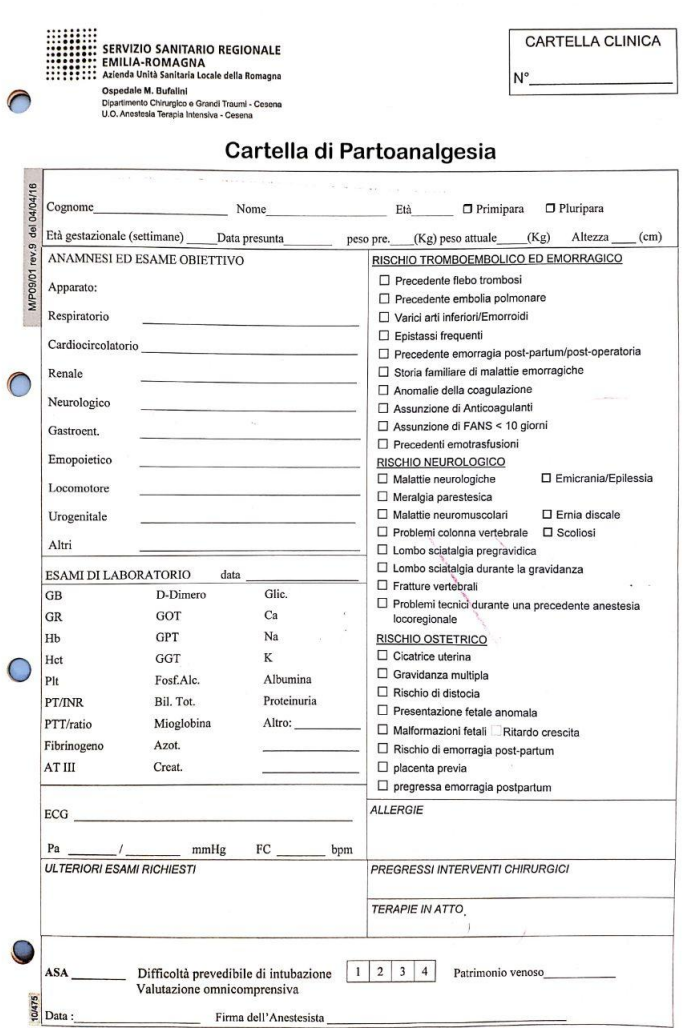
In questa fase vengono registrati i dati anagrafici della paziente, viene eseguita una anamnesi clinica dello stato della paziente e vengono registrate le principali informazioni riguardo lo stato della gravidanza, ceom visibile in figura 2.1. Dopo aver compilato questa scheda la paziente può tornare a casa e il documento rimane in ospedale in attesa dell'effettivo ricovero.

Seconda fase: Ricovero. La seconda fase risulta essere più articolata: inizia nel momento in cui la paziente si reca in ospedale per essere ricoverata nel reparto di ostetricia e termina quando viene dimessa.

Durante questo periodo di permanenza in ospedale la paziente è affidata alle cure dei medici elencati in precedenza, i quali compilano di volta in volta la scheda di partoanalgesia contente tutte le rilevazioni, variazioni e somministrazioni di farmaci a cui è sottoposta la paziente, i suoi valori vitali, l'esito del parto e le informazioni del neonato, così come mostrato in figura 2.2.

I trattamenti di analgesia epidurale sono riservati alle pazienti che hanno già manifestato la volontà di riceverli attraverso la precedente fase di prenota-

zione. Tuttavia il trattamento può essere richiesto anche durante il periodo di travaglio dalla paziente o dai medici che ne seguono il decorso.



SERVIZIO SANITARIO REGIONALE
EMILIA-ROMAGNA
Azienda Unità Sanitaria Locale della Romagna
Ospedale M. Bufalini
Dipartimento Chirurgico e Grandi Traumi - Cesena
U.O. Anestesia Terapia Intensiva - Cesena

CARTELLA CLINICA
N°_____

Cartella di Partoanalgesia

MP09/01 rev.9 del 04/04/16

Cognome_____ Nome_____ Età_____ ☐ Primipara ☐ Pluripara

Età gestazionale (settimane) ____Data presunta_____ peso pre. ___(Kg) peso attuale____(Kg) Altezza ___ (cm)

ANAMNESI ED ESAME OBIETTIVO

Apparato:

Respiratorio _____

Cardiocircolatorio _____

Renale _____

Neurologico _____

Gastroent. _____

Emopoietico _____

Locomotore _____

Urogenitale _____

Altri _____

ESAMI DI LABORATORIO data _____

GB	D-Dimero	Glic.
GR	GOT	Ca
Hb	GPT	Na
Hct	GGT	K
Plt	Fosf.Alc.	Albumina
PT/INR	Bil. Tot.	Proteinuria
PTT/ratio	Mioglobina	Altro: _____
Fibrinogeno	Azot.	_____
AT III	Creat.	_____

ECG _____

Pa _____/_____ mmHg FC _____ bpm

ULTERIORI ESAMI RICHIESTI

RISCHIO TROMBOEMBOLICO ED EMORRAGICO

☐ Precedente flebo trombosi
☐ Precedente embolia polmonare
☐ Varici arti inferiori/Emorroidi
☐ Epistassi frequenti
☐ Precedente emorragia post-partum/post-operatoria
☐ Storia familiare di malattie emorragiche
☐ Anomalie della coagulazione
☐ Assunzione di Anticoagulanti
☐ Assunzione di FANS < 10 giorni
☐ Precedenti emotrasfusioni

RISCHIO NEUROLOGICO

☐ Malattie neurologiche ☐ Emicrania/Epilessia
☐ Meralgia parestesica
☐ Malattie neuromuscolari ☐ Ernia discale
☐ Problemi colonna vertebrale ☐ Scoliosi
☐ Lombo sciatalgia pregravidica
☐ Lombo sciatalgia durante la gravidanza
☐ Fratture vertebrali
☐ Problemi tecnici durante una precedente anestesia locoregionale

RISCHIO OSTETRICO

☐ Cicatrice uterina
☐ Gravidanza multipla
☐ Rischio di distocia
☐ Presentazione fetale anomala
☐ Malformazioni fetali ☐ Ritardo crescita
☐ Rischio di emorragia post-partum
☐ placenta previa
☐ pregressa emorragia postpartum

ALLERGIE

PREGRESSI INTERVENTI CHIRURGICI

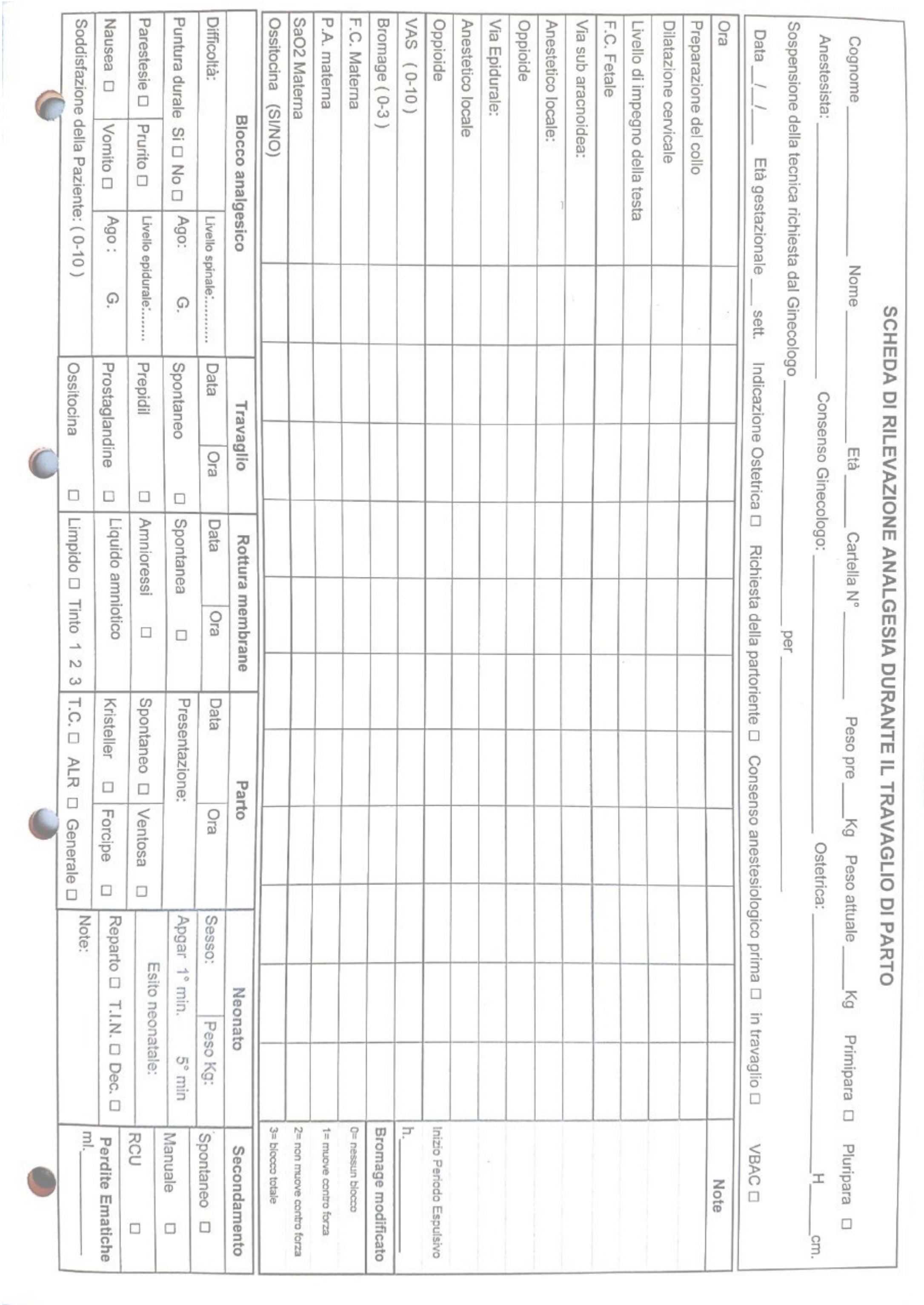
TERAPIE IN ATTO

ASA _____ Difficoltà prevedibile di intubazione Valutazione omnicomprensiva | 1 | 2 | 3 | 4 | Patrimonio venoso_____

Data :_____ Firma dell'Anestesista _____

10475

Figura 2.1: Scheda cartacea per la prenotazione dei trattamenti di analgesia epidurale.



SCHEDA DI RILEVAZIONE ANALGESIA DURANTE IL TRAVAGLIO DI PARTO

Cognome ______ Nome ______ Età ____ Cartella N° ______ Peso pre ___Kg Peso attuale ___Kg Primipara □ Pluripara □

Anestesista: ______ Consenso Ginecologo: ______ Ostetrica: ______ H ___ cm.

Sospensione della tecnica richiesta dal Ginecologo ______ per ______

Data __/__/__ Età gestazionale __ sett. Indicazione Ostetrica □ Richiesta della partoriente □ Consenso anestesiologico prima □ in travaglio □ VBAC □

Ora												Note
Preparazione del collo												
Dilatazione cervicale												
Livello di impegno della testa												
F.C. Fetale												
Via sub aracnoidea:												
Anestetico locale:												
Oppioide												
Via Epidurale:												
Anestetico locale												
Oppioide												Inizio Periodo Espulsivo
VAS (0-10)												h.____
Bromage (0-3)												Bromage modificato
F.C. Materna												0= nessun blocco
P.A. materna												1= muove contro forza
SaO2 Materna												2= non muove contro forza
Ossitocina (SI/NO)												3= blocco totale

Blocco analgesico		Travaglio		Rottura membrane		Parto		Neonato		Secondamento
Difficoltà:	Livello spinale:.........	Data	Ora	Data	Ora	Data	Ora	Sesso:	Peso Kg:	Spontaneo □
Puntura durale Si □ No □	Ago: G.	Spontaneo □		Spontanea □		Presentazione:		Apgar 1° min. 5° min		Manuale □
Parestesie □ Prurito □	Livello epidurale:.......	Prepidil □		Amnioressi □		Spontaneo □	Ventosa □	Esito neonatale:		RCU □
Nausea □ Vomito □	Ago: G.	Prostaglandine □		Liquido amniotico		Kristeller □	Forcipe □	Reparto □ T.I.N. □ Dec. □		Perdite Ematiche
Soddisfazione della Paziente: (0-10)		Ossitocina □		Limpido □ Tinto 1 2 3		T.C. □ ALR □ Generale □		Note:		ml.____

Figura 2.2: Scheda cartacea per la raccolta dei dati di partoanalgesia.

2.1.2 Il report

Come mostrato nelle precedenti figure e in particolar modo nella figura 2.2 i dati da raccogliere sono di diversa natura e comprendono un insieme non eterogeneo di parametri clinici il cui valore può essere:

- sì/no (booleano)[1]
- una o più parole (stringa)[2] scelta da un insieme predefinito di valori possibili
- una o più parole scritte manualmente dagli utenti
- un valore numerico intero
- un valore numerico decimale

Molti parametri possono avere solo un valore facente parte di un range, ad esempio **Bromage** può avere solo un valore intero compreso tra 0 e 3. Altri valori sono invece legati ai vincoli del conteso o del dominio. Ad esempio il parametro **Parto** può assumere il valore "spontaneo", facendo quindi in modo che non siano accettati altri valori, oppure può avere uno **o più** dei valori seguenti valori contemporaneamente:

- Ventosa
- Kristeller
- Forcipe

Alcuni parametri dipendono invece da altri valori, ad esempio se il **Travaglio** fosse di tipo "spontaneo" non sarebbe necessario definire altri dettagli, qualora invece fosse di tipo "indotto" sarebbe necessario esplicitarne la tipologia con uno dei valori tra:

- CRB
- Propess
- Cyotec
- Ossitocina

[1]Il termine **booleano** in informatica indica una variabile che può assumere solo valori vero/falso che possono essere intesi anche come sì/no.

[2]Il termine **stringa** indica una serie di caratteri alfanumerici che possono comporre una o più parole.

2.1.3 Tecnologie disponibili.

Nella struttura ospedaliera e in particolar modo nelle aree interessate dal progetto è presente una connessione Wi-Fi privata a cui ha accesso solo il personale ospedaliero e che viene utilizzata per gestire i propri sistemi informatici e le apparecchiature biomedicali.

Attualmente il personale ostetrico ha a disposizione due computer che si trovano a fianco delle quattro sale parto. Il personale anestesista ha a disposizione un solo computer nel presidio medico di partoanalgesia.

2.2 Analisi

Come si evince dalla precedente raccolta dei requisiti gli utenti finali del sistema sono molteplici e con la possibilità di scrivere alcuni tipi di dati legati alle proprie competenze e altri in condivisione con tutti i medici. I dati presenti in un report possono però essere letti indiscriminatamente da tutti i medici.

Si è scelto di non modellare la prima fase di interazione fra la paziente e l'ospedale nel sistema informatico in quanto non prioritaria per le esigenze degli operatori. La seconda fase è il fulcro delle meccaniche del dominio perciò sarà al centro della seguente analisi e della successiva progettazione.

Sistema distribuito. Dalla raccolta dei dati si deduce che il sistema in questione debba essere un **sistema distribuito**[3] e che i dispositivi che fanno parte del sistema debbano comunicare tra loro attraverso l'ausilio della connessione internet presente nella struttura ospedaliera. I dati dovrebbero essere gestiti in maniera centralizzata poiché ogni informazione deve essere memorizzata in uno spazio di memoria "condiviso" e accessibile a tutti i dispositivi affinché possano essere propagate nel sistema.

Mobilità e interazione Real-time. Da parte del personale anestesista vi è la necessità di un'interazione col sistema in mobilità durante lo svolgimento delle proprie mansioni. Un'interazione in mobilità porta anche ad un'evoluzione real-time dei dati poiché, avendo a disposizione un dispositivo di supporto adeguato, gli anestesisti potrebbero compilare il report durante lo svolgimento di una rilevazione o di un trattamento sulla paziente e le informazioni sarebbero subito visibili agli altri utenti connessi al sistema.

[3]Con il termine **sistema distribuito** si indica una tipologia di sistema informatico costituito da un insieme di processi o dispositivi interconnessi e che comunicano tra loro attraverso lo scambio di messaggi.

Affidabilità e accuratezza dei dati. L'affidabilità riguarda sia la qualità dei dati, che devono essere accurati e rispettare tutti i vincoli imposti dalle meccaniche del dominio, sia il sistema nel suo funzionamento.

Ricordando infatti che i dati dovranno essere analizzati a posteriori da algoritmi di Machine Learning è necessario che essi siano di qualità e seguano una struttura ben definita. Inoltre, pur essendo queste aree coperte dalla connessione Wi-Fi non vi è certezza riguardo la stabilità della connessione e la predisposizione ad imprevisti che facciano rimanere i dispositivi senza connessione. In virtù di quest'evenienza sarà necessario pensare ad un sistema di salvataggio dei dati sulla memoria del dispositivo durante l'assenza di connessione, dando invece priorità al servizio presente sulla rete come fonte primaria dei dati durante la presenza della connessione internet.

Sicurezza. Il sistema informatico dovrebbe garantire un certo livello di sicurezza nella trasmissione e memorizzazione dei dati ma anche fare in modo che i dati siano accessibili solo agli utenti che ne hanno diritto, in particolar modo tenendo presente che i dati in questione sono di carattere clinico e quindi soggetti a rigide leggi sulla privacy.

Il fatto che la connessione dell'ospedale sia privata permette di fare delle assunzioni a priori riguardo la sua sicurezza e la protezione da utenti non autorizzati che andranno comunque migliorate con un sistema di riconoscimento e autorizzazione degli utenti.

2.2.1 Requisiti funzionali

Affinché il sistema adempia agli obiettivi del progetto CERERE è necessario che rispetti i seguenti requisiti funzionali:

- Il sistema deve tracciare ogni tipo di informazione relativa alla paziente per creare un vero e proprio report digitale equiparabile a tutti gli effetti al report cartaceo.
- Il sistema deve identificare gli utenti che vi accedono e tenere traccia delle operazioni da essi eseguite.
- Il sistema deve permettere l'accesso ai soli utenti che hanno il diritto e, qualora necessario, facendo distinzione sui ruoli e sull'accesso ai dati.
- Il personale anestesista deve poter inserire i dati in movimento durante la propria attività lavorativa
- Ogni informazione raccolta dal sistema deve comprendere un dato temporale (data e ora).

- Il software deve permettere l'inserimento e la visualizzazione dei dati da tutti i dispositivi facenti parte del sistema e autorizzati all'accesso.
- I dispositivi facenti parte del sistema devono poter cooperare e scambiarsi informazioni utili alla raccolta dei dati.
- Le informazioni raccolte dal sistema devono essere coerenti e rispettare i vincoli dei dati e le meccaniche del dominio.
- Il sistema deve permettere l'esportazione dei dati in un formato che consenta la loro rappresentazione in forma tabulare, ad esempio CSV[4], affinché possano essere facilmente elaborati e verificati anche da un utente umano.

2.2.2 Requisiti non funzionali

I requisiti non funzionali comprendono:

- **Usabilità**: Il sistema dovrà poter essere utilizzato in modo semplice e immediato. L'interfaccia utente dovrà permettere un'interazione rapida e intuitiva, in modo da facilitare i medici impegnati nelle operazioni di soccorso ed evitare di creare loro ulteriori difficoltà. Allo stesso tempo per gli utenti dovrà essere facile ambientarsi nell'organizzazione dei dati e nell'interazione con l'applicazione.
- **Efficienza**: I software sui diversi dispositivi dovrà essere efficiente in termini di risorse utilizzate e veloce nell'esecuzione delle operazioni richieste.
- **Manutenibilità**: Il sistema dovrà utilizzare tecnologie affidabili che permettano una facile manutenzione nel tempo.
- **Affidabilità**: Il sistema dovrà creare e memorizzare dati corretti e di alta qualità, segnalando quindi eventuali errori agli utenti che stanno inserendo i dati ed evitando di salvare informazioni errate.
- **Sicurezza**: E' necessario identificare e autorizzare ogni utente eventualmente ponendo anche dei limiti ai dati che esso può visualizzare e modificare.

[4]**CSV:** Comma Separated Values è un formato di file basa su file di testo utilizzato per l'importazione e l'esportazione di dati sotto forma di tabella.

Capitolo 3

Progetto del sistema

Come visto nell'analisi dei requisiti i dati clinici della paziente vengono raccolti nelle due fasi di prenotazione e di ricovero, quest'ultima è infatti il fulcro delle meccaniche del dominio e contiene i dati clinici di principale interesse del progetto CERERE, ovvero le informazioni dei trattamenti di analgesia epidurale ricevuti dalla paziente.

Il lavoro di tesi si pone come obiettivo la progettazione e lo sviluppo di un prototipo che rispetti i requisiti esposti nel capitolo precedente e che sia incentrato sulla fase di ricovero della paziente.

3.1 Architettura del Sistema

3.1.1 Backend

Il termine backend identifica il fornitore di servizi, composto da uno o più server, su cui si basa l'effettivo funzionamento del sistema. Inoltre un normale utente non può interagire direttamente con esso senza passare dal frontend.

Server. La gestione centralizzata dei dati necessita della presenza di un server che fornisca questo servizio di memorizzazione e di interazione con i dati e che sia posizionato all'interno della intranet ospedaliera. Il servizio in questione dovrà permettere l'esecuzione delle operazioni esposte nell'analisi dei requisiti.

3.1.2 Frontend

Il termine frontend identifica la parte del sistema visibile all'utente e attraverso cui esso può interagire per l'esecuzione delle operazioni.

Mobile app. La necessità del personale anestesista di spostarsi tra le diverse sale e in piani diversi dell'ospedale non è ben corrisposta dall'utilizzo del computer presente nel corrispettivo presidio medico. Si è scelto perciò di dotare il personale anestesista di un **tablet con sistema operativo Android** per permettere ai medici di compilare il report digitale in maniera dinamica e in real-time mentre si effettua il trattamento e le rilevazioni sulla paziente.

La scelta del tablet, piuttosto che di uno smartphone, è derivata dalla volontà di unire il requisito di portabilità ad una user experience adeguata alla compilazione di documenti digitali poiché le maggiori dimensioni dello schermo permettono una migliore visione dei dati presenti e una maggiore semplicità nell'inserirne di nuovi.

Dashboard. L'interazione col sistema diventa completa con l'utilizzo di una dashboard[1] raggiungibile tramite la intranet ospedaliera attraverso i computer dedicati al personale di anestesia e ostetricia e situati nei rispettivi reparti.

Lo scopo della dashboard è di fornire un ulteriore punto di accesso al sistema per la visualizzazione e la modifica dei dati al pari di come avverrebbe tramite dispositivo mobile ma permettendo inoltre l'esportazione dei dati strutturati in forma di tabella in formato CSV.

L'attività di progettazione si concentrerà tuttavia sulla parte relativa al backend e al tablet con sistema operativo Android poiché ritenuti fondamentali per la creazione di un prototipo del sistema.

3.2 Modello dei dati

Il modello dei dati ha lo scopo di descrivere l'organizzazione dei dati scaturita dalla precedente analisi riprendendo la struttura del report cartaceo ma applicando e rendendo espliciti i vincoli e le dipendenze presenti tra i dati. Per rispettare il criterio di consistenza il modello è utilizzato da tutti i dispositivi facenti parte del sistema.

L'entità principale che si vuole modellare è il report stesso che è composto, a sua volta, da altre entità intese come blocchi di informazioni semanticamente correlate. Ne è un esempio il blocco di informazioni relative alla paziente, così come quelle relative al parto, al neonato eccetera.

Alcuni campi di notevole importanza sono:

- **id**: identificare univocamente ogni report

[1] Con il termine **dashboard** in informatica si intende una schermata che permette di monitorare in tempo reale l'andamento e lo stato delle informazioni interessate.

- **versione**: identificare il numero di versione di un report e consente quindi di capire se esso contiene nuove informazioni
- **status**: permette di distinguere i report attualmente attivi e dai report completati e che quindi non possono essere più modificati.

La prima operazione è stata quella di identificare questi blocchi di informazioni utilizzando come base di partenza il report cartaceo ma applicando anche i suggerimenti e le osservazioni raccolte negli antecedenti colloqui coi medici interessati.

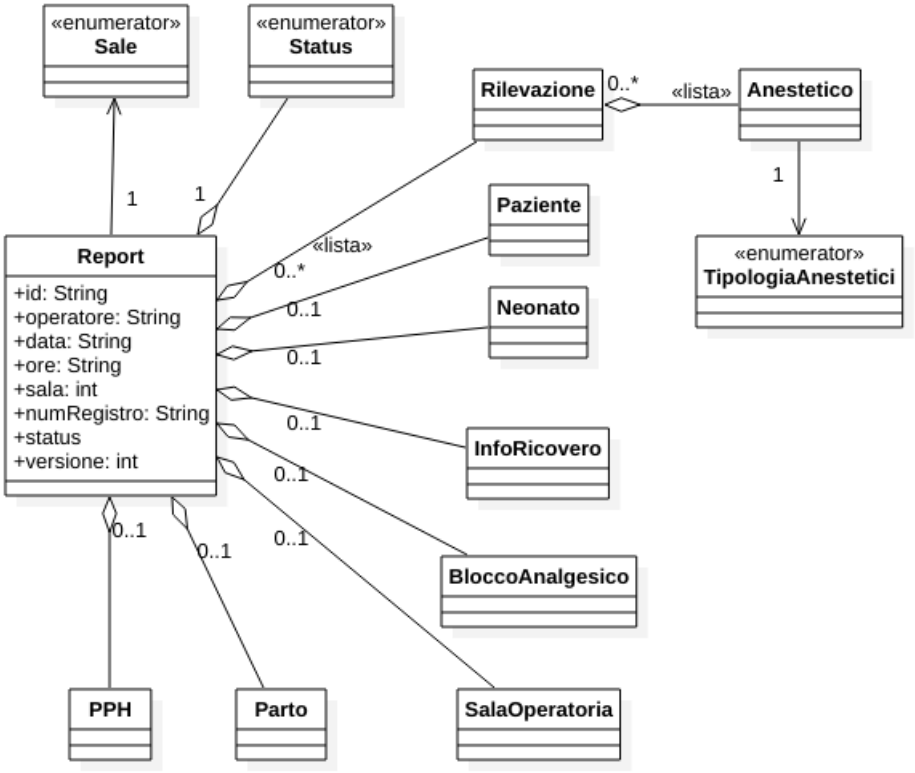


Figura 3.1: Grafico UML semplificato del modello dei dati.

Ogni entità può arrivare ad avere decine di parametri, ognuno dei quali soggetto a vincoli di valore, temporali o relativi al contesto. Una entità molto articolata è quella rappresentante la singola Rilevazione che comprende dati non eterogenei come mostrato in figura 3.1.

```
rilevazione: {
   position: ordine temporale della rilevazione
   time: ora della rilevazione
   collo: a scelta tra
      -Raccorciato
      -Appianato
      -Scomparso
   dilatazione cervicale: valore intero >= 1 e <= 10
   livello impegno testa: valore intero >= -3 e <= 3
   fc fetale: valore intero
   ctg: a scelta tra
      -Tipo1 (normale)
      -Tipo2 (indeterminato)
      -Tipo3 (patologico)
   anestetico: {
   via subaracnoidea: {
      se locale:    a scelta
         -ropivacaina
         -bupivacaina
      se oppioide: sufentanil
   }
   via epidurale : {
      se locale: ropivacaina
      se oppioide: sufentanil
   }
   vas: valore intero >= 0 e <= 10
   bromage: valore intero >= 0 e <= 3
   fc materna: valore intero
   pa materna: {
      paMaternalA: valore intero
      paMaternalB: valore intero
   }
   spo2Materna: valore interno
   temperatura: valore decimale
   ossitocina: valore booleano si / no
    velocit  ossitocina: numero intero, da compilare solo se campo
       ossitocina  attivato
}
```

Listato 3.1: Struttura dell'entità Rilevazione.

Occorre inoltre modellare la figura dell'utente, con le informazioni necessarie al suo riconoscimento, alla differenziazione nell'accesso ai dati e al

tracciamento di quelli inseriti.

L'entità appare così strutturata:

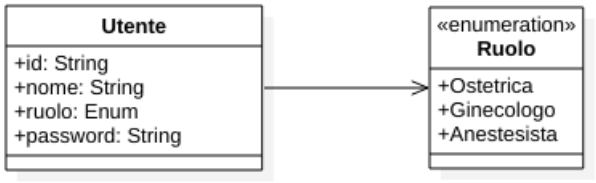


Figura 3.2: Modello delle informazioni di un utente.

3.3 Architettura di comunicazione

E' chiaro che il sistema debba riprendere il modello client-server in cui vi è un solo server di riferimento che permette la centralizzazione dei dati e la propagazione delle modifiche fra tutti i client connessi. Il modello client-server si basa su una sostanziale differenza di poteri fra chi utilizza e richiede i dati (il client) e chi invece fornisce i dati e gestisce le autorizzazioni di accesso (il server). Le due figure devono avere dei protocolli di comunicazione in comune e interagiscono attraverso una connessione internet che può essere privata o pubblica. Per rispondere a queste esigenze verrà utilizzata l'architettura *REpresentational State Transfer* (REST).

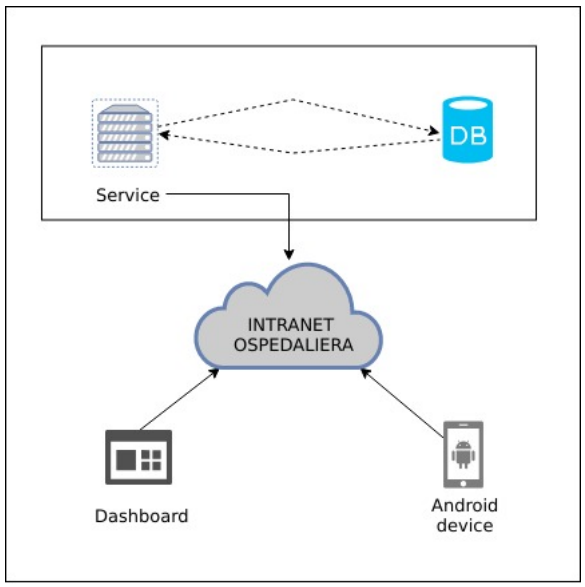


Figura 3.3: Architettura del sistema.

REST Con il termine REST non si intende una tecnologia precisa quanto piuttosto un concetto che implica il rispetto di una serie di linee guida e di approcci che definiscono, ad un alto livello di astrazione, lo stile con cui i dati fra due dispositivi vengono trasmessi.

Queste linee guida definiscono infatti che il server debba:

- identificare ogni risorsa in maniera univoca e attraverso uno URI[2]
- avere un insieme di API[3] uniformi che seguano rigorosamente la stessa struttura. Applicando il concetto secondo cui:

  > "Once a developer becomes familiar with one of your API, he should be able to follow the similar approach for other APIs."

- essere **stateless** ovvero non debba avere una nozione di stato corrente e che le operazioni eseguite non debbano essere collegate alla sessione di autenticazione dell'utente.
- utilizzare una architettura client-server affinché le due entità possano evolvere in maniera indipendente e comunicare attraverso le API.
- eseguire ogni operazione in maniera indipendente affinché non vi sia alcuna correlazione tra le operazioni eseguite.

Tutti questi concetti risultano del tutto slegati dall'effettiva implementazione del server o dal linguaggio scelto, lasciando quindi allo sviluppatore la possibilità di scegliere il linguaggio di programmazione migliore nel proprio contesto, inoltre l'architettura REST è un concetto ampiamente affermato nell'ambito delle comunicazioni di rete e si presta bene con l'idea di "offrire un servizio ad un cliente".

Così facendo non vi è un legame tra chi utilizza il servizio e chi lo mette a disposizione poiché quest'ultimo dovrà semplicemente rispondere a delle richieste senza curarsi del fatto che il richiedente sia uno smartphone, un computer o un tablet.

Il sistema risulta quindi facilmente espandibile con nuovi tipi di utenti che possono utilizzare qualsiasi tipo di dispositivo capace di comunicare col servizio e per l'implementazione di nuove funzionalità basterà aggiungere delle nuove URI alle API disponibili.

[2]In informatica **URI**, ovvero Uniform Resource Identifier, è una sequenza di caratteri che identifica univocamente una risorsa.

[3]**API:** Application Programming Interface, indica un insieme di procedure atte al completamento di un dato compito.

3.4 Backend

Architettura La costruzione del backend si basa sull'architettura REST, esposta in precedenza, in un'ottica orientata ai **micro-servizi**, individuata come migliore soluzione per rispondere alle esigenze del dominio.

L'organizzazione in **micro-servizi** implica che il server sia in realtà diviso in tanti piccoli servizi indipendenti tra loro che cooperano per rispondere alle richieste dei client. Così facendo il sistema risulta facilmente espandibile poiché per aggiungere nuove funzionalità nel server sarà sufficiente creare un nuovo servizio senza modificare quelli già esistenti.

Visto lo stato prototipale dell'infrastruttura sarà sufficiente l'utilizzo di un solo servizio che permetta l'autenticazione degli utenti e l'interazione con i dati.

Memorizzazione delle informazioni La memorizzazione delle informazioni raccolte è un punto fondamentale nel sistema da progettare. I possibili approcci nella scelta di un DBMS (DataBase Management System) sono due: affidarsi ai tradizionali database relazionali o puntare su qualcosa di più flessibile come i database di tipo NoSQL, ovvero non relazionali.

La sostanziale differenza fra queste due macro-tipologie è che i database relazionali organizzano le informazioni in maniera molto strutturata, fissa e predefinita in seguito a diverse fasi di semplificazione e progettazione della struttura dei dati in tabelle che portano ad un compromesso tra prestazioni e ridondanza.

I database di tipo NoSQL si contraddistinguono invece per la facilità di sviluppo, le funzionalità più avanzate e le prestazioni migliori su grandi quantità di dati. In questi database i dati risultano organizzati in schemi altamente flessibili e che possono evolvere agilmente nel tempo per adattarsi a condizioni del dominio o esigenze mutevoli. Tuttavia questi database possono essere molto diversi tra loro per quanto riguarda il linguaggio e la sintassi da utilizzare, la strutturazione dei dati e il funzionamento. In questo caso verrà utilizzato un database di tipo NoSQL per le caratteristiche appena descritte, che permettono una rapida organizzazione dei dati e un'agile evoluzione in futuro.

3.5 Mobile app

Architettura Il pattern architetturale utilizzato è il celebre **MVC** (Model View Controller) che implica una separazione e un elevato grado di indipendenza tra la parte di visualizzazione dei dati (view), il modello del dominio di riferimento (model) e la logica applicativa che li coordina (controller).

Funzionalità Le funzionalità messe a disposizione dall'applicazione hanno lo scopo di facilitare il compito dei medici nella compilazione del report digitale.

In particolare l'applicazione permette di:

- eseguire l'accesso e la disconnessione dell'utente
- creare un nuovo report per una paziente ricoverata
- modificare i dati anagrafici e clinici legati alla paziente
- propagare le modifiche effettuate nel sistema in presenza di una connessione
- assistere i medici nell'inserimento dei dati attraverso la segnalazione degli errori e l'applicazione dei vincoli imposti dal dominio

Uno dei punti cruciali nel funzionamento dell'applicazione è la gestione dei dati in relazione alla presenza o all'assenza di connessione. Il problema, non banale, va gestito affinché i dati scritti dagli utenti non vengano persi se la connessione è assente, allo stesso tempo occorre far sì che ogni utente stia visualizzando e scrivendo sull'ultima versione dei dati.

Networking Il networking, ovvero la comunicazione attraverso la rete internet, è alla base del funzionamento di tutto il sistema. Attraverso lo scambio di messaggi sulla rete Wi-Fi il tablet Android deve poter ricevere e inviare le informazioni tracciate. Tuttavia i dati possono essere inseriti anche da altri dispositivi, occorre quindi gestire l'accesso in concorrenza e soprattutto la propagazione delle modifiche all'interno di tutto il sistema.

In questo contesto i possibili approcci sono due:

- **Polling:** il client controlla periodicamente che i dati in suo possesso siano aggiornati all'ultima versione disponibile. Il controllo avviene confrontando la versione dei dati presente sul server con quella salvata sul dispositivo: se i dati presenti sul server sono più recenti occorrerà scaricarli.
- **Push:** il server notifica il client quando una nuova versione dei dati è disponibile quindi il client scarica la nuova versione.

Pur essendo più "semplice", l'approccio a polling implica un inutile consumo di batteria e di risorse poichè costringe la CPU del dispositivo a risvegliarsi periodicamente per effettuare delle richieste al server.

In questo caso verrà utilizzato l'approccio push, permettendo al client di evolvere il proprio stato in maniera indipendente e di venire notificato dal

server qualora i dati dovessero subire una modifica. E' necessario però fare in modo che il client sia sempre identificabile in maniera univoca e pronto a ricevere un messaggio dal serve, ad esempio creando un collegamento che resti attivo nel tempo tra i due.

Memorizzazione dei dati Il dispositivo deve salvare i dati sulla propria memoria locale per minimizzare la dipendenza dalla rete internet, a causa dei fattori già esposti di incertezza della connessione, e per una serie di motivi prestazionali.

Capitolo 4

Prototipo del sistema

4.1 Backend Service

Il servizio di backend su cui si basa l'intero funzionamento del sistema è implementato grazie all'utilizzo di tecnologie quali **Vert.x** e **MongoDB** attraverso un insieme di **RESTful API**[1] che fanno da interfaccia di interazione con i client presenti nel sistema.

4.1.1 RESTful API

Per l'implementazione delle richieste si è scelto di utilizzare il protocollo HTTP, facendo ampio uso degli status-code[2] per segnalare il successo o il fallimento di un'operazione. Attraverso le richieste HTTP di tipo PUT, POST, DELETE e GET si possono eseguire delle operazioni sui dati memorizzati nel database in maniera trasparente, affinché il client che esegue la richiesta conosca solo il risultato dell'operazione.

I diversi tipi di richieste HTTP servono a distinguere i tipi di operazioni da compiere:

- **GET** consente di ottenere dei dati in base all'URI della richiesta. Per alcune operazioni potrebbe essere necessario mandare dei dati all'interno della richiesta, ad esempio per applicare un filtro nei dati richiesti.
- **POST** consente di aggiungere dei dati solo se ancora **non** presenti, ad esempio la creazione di un nuovo documento. In questo caso è sempre necessario inserire nel corpo della richiesta i dati da memorizzare.

[1] Con il termine **RESTful** si indica un contesto che rispetta l'architettura REST e quindi le linee guida di cui si è discusso nel capitolo precedente.

[2] Gli **status code** sono codici numerici a tre cifre che indicano il risultato di una richiesta HTTP.

- **PUT** consente di modificare i dati già presenti. Anche in questo caso è necessario inserire nel corpo nella richiesta i dati da modificare.
- **DELETE** consente di eliminare i dati selezionati. Non necessita di ulteriori dati nella richiesta

4.1.2 Vert.x

Vert.x è un toolkit event-driven utilizzato per creare applicazioni reattive in molteplici linguaggi di programmazione. In questo caso il linguaggio scelto è stato Java in quanto altamente prestante, affidabile e ampiamente supportato.

La potenza di Vert.x sta nella sua logica asincrona che garantisce un'elevata concorrenza e negli strumenti che mette a disposizione per l'implementazione di API, event bus, elaborazione di file e oggetti JSON, gestione del database.

Il suo funzionamento si basa su dei componenti, chiamati **verticle** che agiscono in maniera concorrente svolgendo ognuno il proprio compito ma con la possibilità di comunicare tra loro attraverso un event bus o degli oggetti condivisi.

Nella fase di implementazione sono stati utilizzati i verticle per creare il servizio di gestione delle informazioni su cui si basa il funzionamento del sistema. Nello specifico i verticle utilizzati sono tre, rispettivamente:

- un verticle per il servizio che riceve le richieste HTTP
- un verticle dedicato all'interazione con il database
- un verticle per il servizio di WebSocket

Routing HTTP. La gestione delle richieste HTTP si basa sul verticle che implementa un Router HTTP per lo "smaltimento" delle richieste in arrivo.

Il funzionamento è semplice: prima si crea il ruoter, in seguito si associa un tipo richiesta HTTP e un indirizzo URI ad una classe o funzione che avrà il compito di rispondere alla richiesta ricevuta, così come mostrato in figura 4.1

```
// get the api status
router.get("/cerere/api/status").handler(new
    ServiceStatusHandler(...));

// add a new report into the database
router.post(API_PATH + "/reports/add").handler(new
    AddReportHandler(...));

// update the report data
```

```
router.put(API_PATH + "/reports/update/:id").handler(new
    UpdateReportHandler(...));

// get the report with this unique id
router.get(API_PATH + "/reports/:id").handler(new
    GetReportHandler(...));
```

Listato 4.1: Esempio di alcune API.

WebSocket. La WebSocket è una tecnologia che fornisce un canale di comunicazione duraturo nel tempo tra un client e un server con lo scopo di permettere una comunicazione continua nel tempo tra i dispositivi collegati. Per questo motivo è stata scelta come mezzo di notifica tra client e server per la propagazione delle modifiche ai dati nel sistema.

Ciò avviene attraverso un canale di comunicazione TCP che, a differenza delle normali richieste HTTP, continua ad essere attivo nel tempo e permette al client e al server di scambiarsi dei dati senza dover inizializzare nuovamente il collegamento. In genere non vi è un limite al numero di WebSocket che un determinato client può tenere attive con un determinato server, tuttavia in questo caso si richiede che il client abbia una sola connessione attiva con il server.

```
private Set<ServerWebSocket> clients = new HashSet<>();

[ ... ]

public void handle(final ServerWebSocket ws) {
    if (ws.path().equals("/ws/subscribe") && !clients.contains(ws)) {
        clients.add(ws);
        ws.closeHandler(h -> clients.remove(ws));
    } else {
        ws.reject();
    }
}
```

Listato 4.2: Registrazione di un client sulla WebSocket.

Così come mostrato in figura 4.2 la WebSocket rimane in ascolto sulla URI predefinita e quando un client chiede di poter inizializzare una connessione controlla che esso non sia già presente nella lista dei client collegati. Se il client è già collegato la richiesta viene rifiutata, altrimenti viene creato il collegamento e il client viene aggiunto alla lista degli utenti collegati.

Questo canale di comunicazione viene utilizzato solo per notificare i client ,nel caso i dati abbiano subito una modifica, seguendo l'approccio di tipo push spiegato nel paragrafo 3.5.

Se, ad esempio, viene creato un nuovo report nel sistema allora tutti gli utenti ricevono un messaggio attraverso la WebSocket che li notifica riguardo la presenza di un nuovo report e chiede loro di effettuare la sincronizzazione attraverso una nuova richiesta alle API del servizio.

4.1.3 MongoDB

Tra i molti tipi di database NoSQL è stato scelto **MongoDB** che si distingue per un'organizzazione delle informazioni in forma di documenti dove i dati hanno una struttura a dizionario, ovvero formata da un insieme di coppie nome-valore, che risulta estremamente comoda per l'elaborazione da parte dei software. Il ruolo dei documenti è assimilabile alle classiche tabelle dei database relazionali e il formato utilizzato per il salvataggio dei dati è il BSON ovvero ovvero Binary JSON.

Questo insieme di caratteristiche permette di rendere universale il modello dei dati poiché JSON è ampiamente supportato da tutti i principali linguaggi di programmazione e risulta essere uno standard **de-facto** nelle comunicazioni client-server in contesti RESTful.

L'istanza del database comprende quindi una collezione contenente tutti i report e una contenente tutti gli utenti autorizzati ad accedere alle informazioni la cui struttura riprendere il modello dei dati spiegato in precedenza nella sezione 3.2.

4.2 Applicazione Android

L'applicazione permette di interagire con il sistema informatico e ha lo scopo di fornire uno strumento di monitoraggio real-time delle informazioni delle pazienti presenti nelle sale parto. Consente inoltre l'inserimento di nuovi dati e la modifica di quelli già esistenti, in accordo con i vincoli del dominio.

Nonostante la scrittura di applicazioni Android native avvenga in linguaggio Java (e più recentemente anche in Kotlin) questo ambiente presenta delle best-practices molto diverse rispetto a una programmazione pura in linguaggio Java. Si è cercato quindi di rispettare le linee guida[7] suggerite da Google[3] riuscendo tuttavia ad applicare i principi del paradigma ad oggetti. Le tecnologie utilizzate per lo sviluppo dell'applicazione sono in prevalenza quelle messe a disposizione dal Software Development Kit[4] (SDK) nativo di Android, in particolare nella sua versione n. 28.

Il pattern architetturale MVC è stato adattato allo stile di programmazione in Android in cui le Activity sono allo stesso tempo View e Controller, il modello dei dati rimane invece isolato nel Model.

4.2.1 Networking

L'applicazione utilizza ampiamente le RESTful API messe a disposizione dal backend sulla rete locale dell'ospedale. La comunicazione di rete permette inoltre la sincronizzazione dei dati attraverso alcuni accorgimenti ingegneristici che combinano l'interazione client-server all'utilizzo di un canale di comunicazione mantenuto attivo e basato sulla tecnologia WebSocket.

[3] Google è l'azienda proprietaria del sistema operativo Android

[4] Con il termine Software Development Kit si indica l'insieme insieme degli strumenti utilizzati per lo sviluppo e la documentazione di un software.

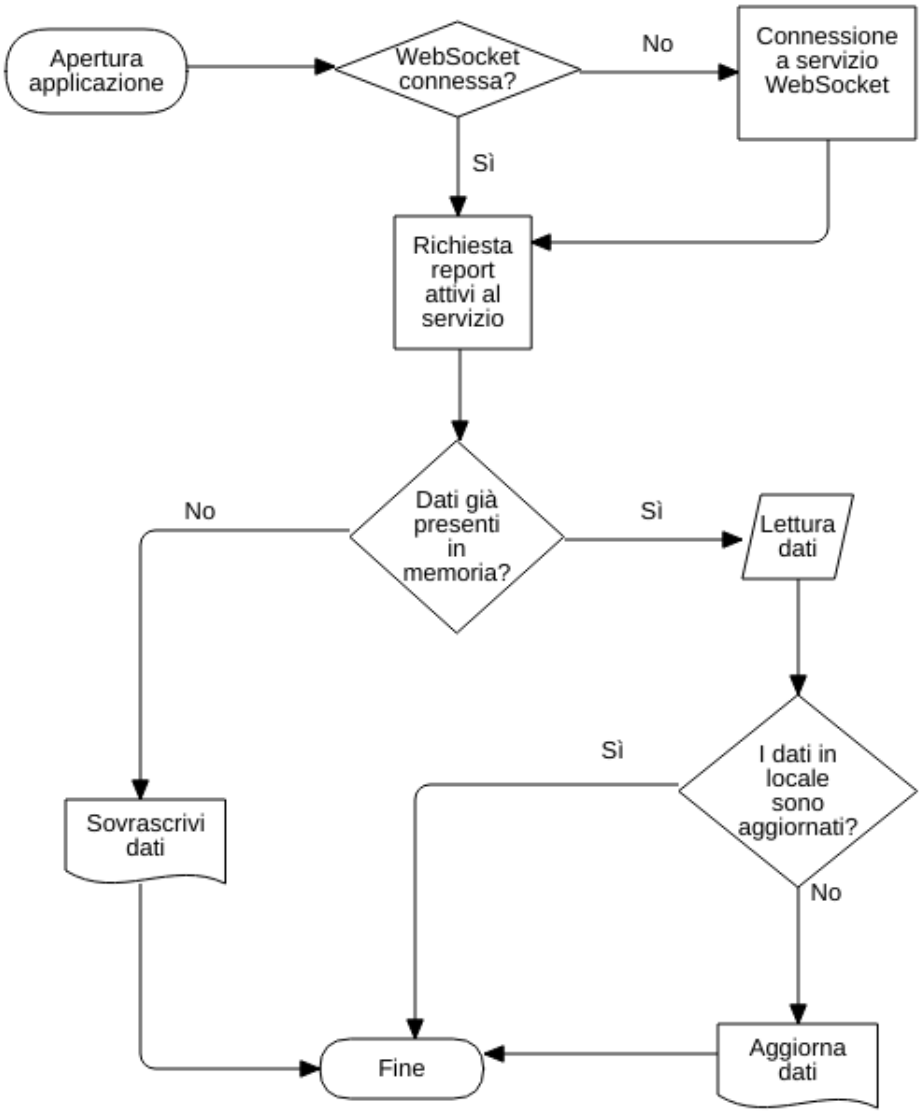


Figura 4.1: Flow-chart delle operazioni di sincronizzazione dei dati eseguite all'apertura dell'applicazione.

Come mostrato in figura 4.1 le operazioni di sincronizzazione dei dati presenti sul dispositivo con quelli presenti sul servizio remoto comprendono più fasi. Questa gestione dei dati dà maggiore priorità a quelli presenti in remoto piuttosto che a quelli in locale al fine di diminuire le possibilità di conflitti ed evitare che un utente con una versione dei dati meno recente possa sovrascrivere quelli più recenti che si trovano sul database del servizio.

Il collegamento tramite WebSocket è utilizzato in maniera passiva al solo scopo di ricevere delle "notifiche" dal servizio in backend quando i dati vengono modificati. E' importante quindi che questo collegamento sia sempre attivo.

Volley. Le richieste sono implementate grazie all'impiego della libreria *Volley* che si propone di semplificare le comunicazioni di rete HTTP e il consumo di risorse remote. La libreria è stata creata direttamente da Google e permette l'utilizzo di alcune funzioni avanzate ma con una quantità di codice minimale, inoltre è particolarmente consigliata in contesti di tipo RESTful.

La scelta di utilizzare Volley è dovuta anche alla volontà di ottimizzare la gestione dei thread, in maniera trasparente, per evitare di bloccare il processo principale (main thread) dell'applicazione.

Un celebre problema di Android è infatti legato alla gestione delle operazioni sul main thread che può generare l'errore Application Not Responding (spesso abbreviato in **ANR**) nel caso in cui una qualsiasi operazione svolta su di esso duri per più di 5 secondi. Questo è dovuto al fatto che, essendoci un solo thread, le operazioni sono svolte in maniera completamente sequenziale e il ritardo nel completamento di un'operazione blocca letteralmente tutta l'applicazione.

E' facile immaginare quanto possa essere rischioso eseguire delle richieste di rete sul main thread non avendo certezza riguardo i tempi di risposta dell'interlocutore (solitamente un server) ed infatti è una operazione fortemente sconsigliata, così come le operazioni di input/output (I/O) su grandi quantità di dati.

L'alternativa tuttavia è una gestione molto articolata della creazione e distruzione di thread appositi che gestiscano le richieste. Volley cerca di sopperire a queste problematiche mettendo a disposizione del programmatore una gestione trasparente dei thread relativi alle richieste di rete e automatizzando alcuni passaggi della loro gestione.

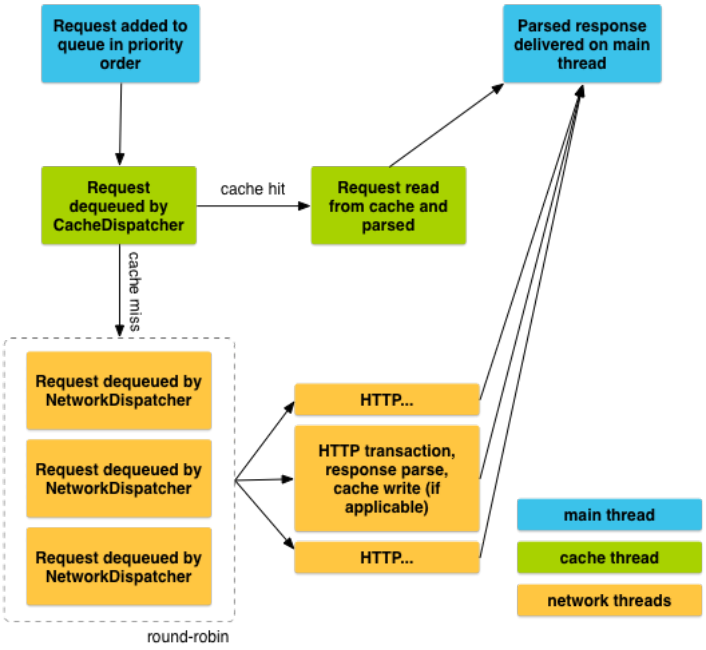


Figura 4.2: Ciclo di vita di una richiesta HTTP in Volley.

Come mostrato in figura 4.2 il ciclo di vita di una richiesta HTTP è soggetto a diversi passaggi ma l'aspetto fondamentale è come venga tutto gestito da thread creati ad-hoc per poi far tornare la risposta relativa alla richiesta di rete sul main thread dove può essere elaborata secondo le esigenze del contesto. A questo punto è responsabilità del programmatore creare un thread secondario qualora si debbano svolgere operazioni prolungate o esose in termini di risorse.

Nel caso dell'applicazione CERERE i dati inviati e ricevuti sono in formato JSON perciò si è utilizzato esclusivamente richieste del tipo:

- JsonObjectRequest qualora la risposta HTTP debba contenere un solo elemento
- JsonArrayRequest qualora la risposta HTTP debba contenere una lista di elementi

Non è stato necessario l'utilizzo di ulteriori thread per compiti specifici poichè il contenuto delle risposte viene salvato sulla memoria del dispositivo senza subire elaborazioni particolari.

WebSocket La tecnologia WebSocket è implementata attraverso un servizio dedicato all'interno dell'applicazione che si occupa di creare il collegamento con

il server non appena l'applicazione viene aperta per la prima volta e successivamente di mantenerlo attivo, agendo in background, anche quando l'utente utilizza altre applicazioni o blocca lo schermo del dispositivo.

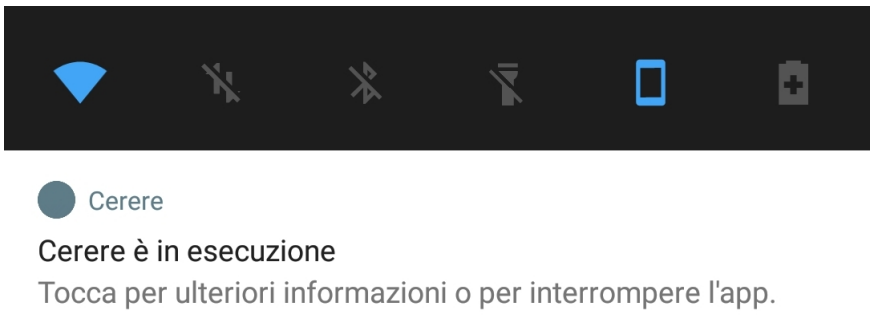


Figura 4.3: Notifica che segnala il funzionamento in background.

La notifica in figura 4.3 è dovuta ad alcuni cambiamenti applicati da Google a partire da Android Oreo (versione SDK 26) che portano delle limitazioni nell'esecuzione di operazioni in background con lo scopo di migliorare la durata della batteria dei dispositivi.

Non conformandosi a questi limiti si riceve infatti un errore dal sistema e l'applicazione cessa di funzionare come previsto. La soluzione è stata quella di utilizzare un Service che crea un Thread dentro cui è contenuta la WebSocket e che cerca di mantenere attivo il collegamento riconnettendosi non appena è rilevata la presenza della connessione Wi-Fi.

4.2.2 Caching dei dati

Il salvataggio dei dati (caching) è implementato in due fasi:

1. Salvataggio sulla memoria locale del dispositivo
2. Salvataggio in variabili a runtime[5]

Il salvataggio dei dati dati sulla memoria del dispositivo avviene attraverso le SharedPreferences, ovvero uno spazio di memoria messo a disposizione da Android ad ogni applicazione installata per salvare piccole quantità di dati. Il salvataggio non richiede particolari accorgimenti e avviene a partire dai dati in formato JSON, convertiti in **stringa** e associati ad una **key** che sarà poi utilizzata per identificarli.

Il salvataggio in variabili a runtime è invece più articolato e implica la lettura dei dati appena salvati nelle SharedPreferences e la loro successiva

[5]**Runtime:** indica il momento in cui un software viene eseguito.

conversione in *Plain Old Java Object* (POJO) ovvero un oggetto Java con costruttore vuoto e con una serie di getter e setter che viene usato nella parte di Model dell'applicazione. Spesso questi oggetti mappano con un rapporto uno ad uno altri elementi come tabelle di database o strutture JSON che sarebbero altrimenti difficili da elaborare o gestire. In questo caso l'oggetto POJO è la rappresentazione completa di un elemento JSON, perciò si può effettuare una conversione in maniera bidirezionale.

Nel caso di un singolo elemento l'elaborazione avviene in maniera sincrona sul main thread poichè non risulta particolarmente esosa. Nel caso di una lista di elementi viene invece utilizzato un ulteriore thread esclusivamente dedito a questo compito, implementato attraverso un AsyncTask, che converte un JsonArray in una lista di oggetti Java.

4.2.3 User Experience

L'interfaccia grafica dell'applicazione, in particolare la disposizione e organizzazione degli elementi visivi, è stata sviluppata immaginando una User Experience semplice ed efficace. Il punto di accesso all'applicazione è la finestra di login in cui l'utente deve autenticarsi per poter proseguire nell'utilizzo dell'applicazione. Una volta autenticato non sarà più necessario effettuare il login fino a che non si esegue la disconnessione.

Dopo essersi autenticati è necessario scegliere la sala di cui si vogliono vedere le informazioni attraverso uno dei quattro bottoni, uno per ogni sala, presenti sulla schermata dell'applicazione. Ogni bottone è caratterizzato dal nome della sala a cui si riferisce. Qualora la sala sia attualmente vuota viene chiesto all'utente di inizializzarla registrando una nuova paziente. Se invece nella sala risulta ricoverata una paziente si potrà accedere alla successiva schermata con le informazioni presenti.

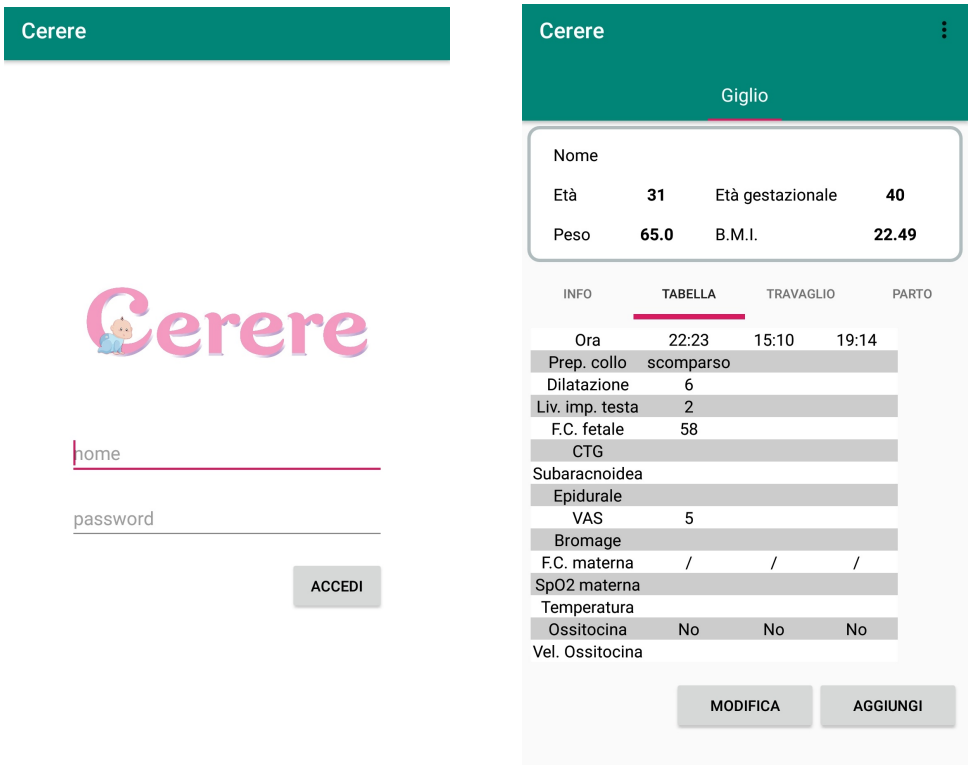
Quest'ultima sezione è il cuore del funzionamento dell'applicazione. Attraverso la navigazione in tab l'utente può scegliere prima la sala di riferimento e quindi la macro-categoria di dati ai cui accedere. In seguito attraverso i bottoni si può accedere ai diversi blocchi di informazioni visualizzate attraverso dei pop-up in cui è possibile anche modificarle.

Questi blocchi di informazioni sono organizzate riprendendo il concetto di blocco di dati semanticamente correlati spiegato nella sezione 3.2. Ne deriva che i dati relativi al paziente si troveranno in un unico pop-up, quelli relativi all'entrata in sala parto si troveranno tutti insieme in un altro pop-up, eccetera. Questa disposizione cerca di rendere più agevole agli utenti il passaggio da cartaceo a digitale poiché le informazioni seguono la divisione in blocchi del report cartaceo. Per questo motivo si è scelto anche di mostrare le rilevazioni

analgesiche attraverso una rappresentazione tabulare, figura 4.4b, che riprende quasi esattamente la struttura cartacea mostrata in figura 2.2.

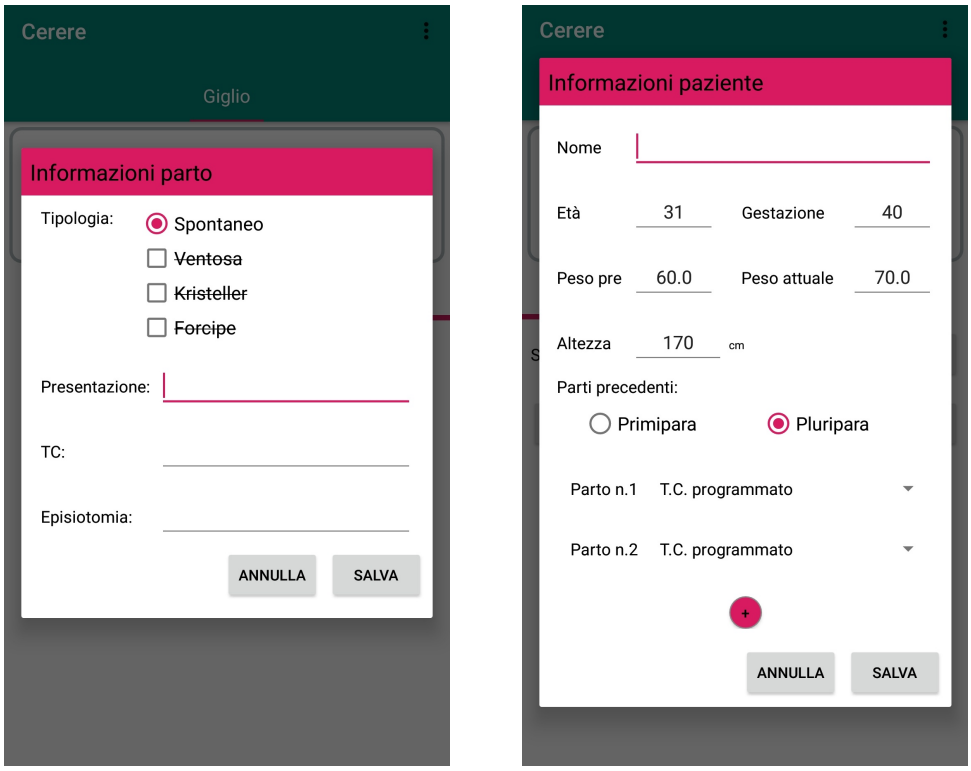
Dal punto di vista ingegneristico questa schermata è costruita attraverso l'utilizzo di Tabbed Activity e **Fragment**, entrambi componenti di Graphical User Interface (GUI) messi a disposizione dal sistema operativo.

In particolare i Fragment sono componenti con un ciclo di vita proprio e indipendente da quello dell'Activity che li contiene. A sua volta una Activity può contenere diversi fragment per creare interfacce dinamiche che si adattino a dispositivi con dimensioni differenti. Occorre tuttavia prestare attenzione al carico prestazionale legato all'elevato utilizzo di questi componenti, alla complessa gestione del loro ciclo di vita e alla comunicazione con gli altri componenti.



(a) Schermata di login. (b) Tabella rilevazioni analgesiche.

Figura 4.4: Alcune schermate dell'applicazione.



(a) Inserimento dati parto. (b) Inserimento dati paziente.

Figura 4.5: Alcuni pop-up per l'inserimento dei dati.

4.3 Sicurezza

Il sistema garantisce un livello di sicurezza idoneo al contesto del dominio attraverso diversi accorgimenti adottati sia sul servizio in backend che sull'applicazione Android:

- Criptazione delle password
- Salvataggio dei dati in modo che siano inaccessibili dall'esterno

Il primo punto è realizzato attraverso l'applicazione di funzioni di hash sulle password.

Queste funzioni hanno due proprietà fondamentali:

- sono unidirezionali, ovvero difficili da invertire, e permettono perciò di criptare un valore senza che questo possa essere decriptato
- identificano univocamente un valore affinché due valori diversi non possano essere equivalenti in seguito all'applicazione della stessa funzione hash

Infatti la richiesta effettuata dall'applicazione al servizio per eseguire l'autenticazione dell'utente contiene il valore della password ottenuto in seguito all'esecuzione di una funzione di hash sulla stessa e il database non contiene le password degli utenti in chiaro bensì memorizza il loro equivalente ottenuto della funzione hash. L'autenticazione di un utente avviene quindi confrontando il valore hash memorizzato nel database con il valore hash ricevuto dal client: se i valori corrispondono vuol dire che sono stati generati dalla stessa password e quindi l'autenticazione avviene con successo, altrimenti si ottiene un esito negativo.

La sicurezza dei dati riguarda in particolar modo l'applicazione Android: in questo ambiente i dati dei report ricevuti ed elaborati dal dispositivo sono salvati sulla memoria locale messa a disposizione dalle SharedPreferences in modalità PRIVATE. Si può immaginare questo spazio di memoria come un container inaccessibile da altre applicazioni o dall'utente neanche con stratagemmi come il rooting (sblocco del dispositivo). Il risultato è un isolamento completo che permette di memorizzare in sicurezza le informazioni sul dispositivo.

La password dell'utente invece non viene salvata sul dispositivo, facendo invece in modo che venga immediatamente criptata tramite funzione hash ed inserita esclusivamente nella richiesta di autenticazione.

Capitolo 5

Validazione e discussione del lavoro svolto

Feedback ricevuti dagli utenti I feedback ricevuti dagli utenti durante lo sviluppo dell'applicazione riguardano principalmente l'interfaccia grafica: è emerso come un workflow di utilizzo basato su molte schermate sia dispersivo e confusionario per l'utente. In questo caso è stata apprezzata maggiormente una disposizione degli elementi facendo ampio uso di tab, che è stata in seguito effettivamente implementata.

Allo stesso modo è stato evidenziato come l'inserimento di valori booleani (sì/no) sia più chiaro utilizzando due bottoni radio piuttosto che un checkbox o un pulsante switch.

Altri feedback hanno riguardato la gestione delle informazioni che, dovendo essere digitalizzate, hanno richiesto alcuni accorgimenti particolari sia nell'applicazione dei vincoli del dominio sia nella definizione dei loro valori ammissibili. Ne è un esempio il parametro che tiene traccia delle precedenti gravidanze portate avanti dalla paziente e che può assumere i valori "nullipara", "primipara", "pluripara" a seguito del quale occorre specificare la tipologia di ogni singolo parto. Questa operazione risulta piuttosto articolata e si è discusso a lungo di come poterla rendere più intuitiva. In fine vi è stata la richiesta di aggiungere un campo **Note** anche nei blocchi di informazioni in cui non era previsto con lo scopo di fornire una certa libertà ai medici nel tracciare eventi insoliti e non conoscibili a priori.

Test e scenari di utilizzo I singoli componenti del sistema hanno subito dei test durante tutta la fase di implementazione per testare in particolar modo la corretta interazione client-server e che il workflow dei dati fosse gestito correttamente. I test sono stati effettuati attraverso i framework dedicati a questo scopo come **JUnit** e **AndroidJUnit**.

Al termine della fase di sviluppo sono stati individuati tre scenari utilizzati per simulare alcuni casi d'uso dell'applicazione nel contesto reale:

1. **L'utente inserisce nuove informazioni mentre il tablet Android è offline**
   In questo caso si è constatato come le informazioni vengano salvate correttamente sulla memoria locale del dispositivo. L'applicazione viene notificata non appena è rilevata una nuova connessione di tipo Wi-Fi e in quel momento manda i dati precedentemente inseriti al server.

2. **L'utente utilizza per la prima volta l'applicazione**
   Si è potuto verificare come in questo caso l'applicazione richieda al server i dati di tutte le paziente attualmente presenti nelle sale parto e li salvi correttamente sulla memoria locale del dispositivo per poi mostrarle all'utente.

3. **Un utente inserisce nuove informazioni tramite dashboard**
   In questo caso l'inserimento dei dati tramite dashboard è stato simulato attraverso delle richieste HTTP manuali alle API del servizio in backend. Il server memorizza i dati ricevuti sul proprio database e in seguito "notifica" il tablet connesso tramite collegamento WebSocket.

Conclusioni e sviluppi futuri

La tesi si colloca nell'ambito del progetto CERERE e ha avuto come obbiettivo il progetto e lo sviluppo di un sistema informatico per la raccolta e il tracciamento delle informazioni delle pazienti in travaglio di parto. Attualmente, infatti, la raccolta di questi dati avviene in forma cartacea a seguito della comunicazione orale tra i medici coinvolti e spesso dopo diverso tempo dall'effettivo verificarsi di un evento o rilevazione clinica.

Le diverse fasi della progettazione del software hanno portato allo realizzazione di un sistema informatico, comprendente diverse tipologie di dispositivi, che ha lo scopo di semplificare le operazioni di gestione dei dati clinici delle pazienti e di fornire ai medici un valido strumento tecnologico di tracciamento e di raccolta dei dati. Nella pratica questo ha comportato lo sviluppo di un backend, lato server, che fornisce il servizio di memorizzazione e gestione dei dati utilizzando tecnologie moderne come Vert.x e MongoDB e lo sviluppo di un'applicazione Android nativa che permette agli utenti di interagire col sistema.

In questo contesto è stata particolarmente interessante l'interazione con i medici poiché ha messo in luce come sia visto l'utilizzo di tecnologie pervasive da parte di quei settori che non sono nativamente tecnologici e che spesso vedono la digitalizzazione come un ostacolo piuttosto che come una risorsa. A mio avviso i medici si sono dimostrati molto entusiasti delle funzionalità messe a disposizione dal sistema informatico e hanno collaborato attivamente dandomi dei feedback per migliorare la user experience e la gestione delle informazioni. E' stata un'esperienza utile poiché è stato necessario lavorare in un contesto reale con dei requisiti da rispettare e degli utenti con cui discutere per migliorare il software.

Dal punto di vista tecnologico questo progetto mi ha dato l'opportunità di studiare e utilizzare numerose nuove tecnologie e approcciarmi alla progettazione del sistema sfruttando a pieno le conoscenze acquisite durante il percorso universitario. In questo contesto lo sviluppo di un'applicazione Android è stata un'esperienza stimolante poiché mi ha permesso di capire quali sono i pregi e i difetti di questa piattaforma e del conseguente sviluppo di applicazioni native su di essa. Allo stesso modo è risultato particolarmente entusiasmante lo svi-

luppo del backend attraverso l'utilizzo di tecnologie quali Vert.x e MongoDB che si sono rivelate delle tecnologie molto piacevoli da utilizzare e con delle ottime potenzialità per il futuro. L'idea che il sistema sia in uno stato prototipale lascia aperte le porte ad ulteriori e più efficaci sviluppi futuri, come:

- **Implementazione di una dashboard**
  Per essere completo il sistema dovrebbe fornire agli utenti un modo per accedere alle informazioni anche tramite computer. In questo contesto l'utilizzo di una pagina web visionabile da browser, che non dipenda quindi dal sistema operativo utilizzato, e con un'interfaccia user friendly sarebbe un'ottima soluzione al problema.

- **Implementazione protocollo HTTPS**
  L'implementazione del protocollo HTTPS, ovvero HyperText Transfer Protocol over Secure Socket Layer, dovrebbe essere implementata nelle prossime fasi di ampliamento del sistema e di test sul dominio reale per offrire una maggiore sicurezza nella trasmissione dei dati.

- **Esportazione digitale delle informazioni raccolte**
  L'esportazione delle informazioni raccolte risulta molto utile per una prima fase di transizione da cartaceo a digitale, con lo scopo di fornire ai medici un pratico mezzo di verifica dei dati in forma tabulare e rendere meno netta la transizione alla forma digitale.

- **Sistema di notifiche** Considerando che il personale anestesista opera su un piano differente da quello in cui si trovano le sale parto, sarebbe molto utile implementare un sistema di notifiche con cui il personale ostetrico, tramite computer, possa chiedere al medico anestesista di turno di recarsi al piano superiore oppure notificarlo dell'occorrenza di un particolare evento clinico.

- **Interazione con le apparecchiature biomedicali**
  Mantenendo una visione molto ampia riguardo i possibili sviluppi del sistema progettato sarebbe senza dubbio utile un'iterazione con le apparecchiature biomedicali già presenti nei reparti interessati dal progetto: la creazione di un ambiente intelligente ridurrebbe al minimo le operazioni di inserimento e gestione dei dati da parte degli utenti umani

Ringraziamenti

Vorrei ringraziare il Prof. Ricci per avermi dato l'opportunità di partecipare a questo progetto, la Dott.ssa Sara Montagna e l'Ing. Angelo Croatti per avermi seguito con simpatia e pazienza durante il lavoro di tesi.

Ringrazio più in generale l'Università di Bologna per avermi accolto a braccia aperte in seguito al mio trasferimento di Università.

Infine un ringraziamento speciale va a Giada per avermi sostenuto, con un sempre immancabile sorriso, negli ultimi due anni del mio percorso universitario.

Bibliografia

[1] Regione Trentino-Alto Adige, *https://trentinosalutedigitale.it/.*

[2] ENISA, *Smart Hospitals: Security and Resilience for Smart Health Service and Infrastructures*, 2016

[3] IBM, *https://www.ibm.com/watson.*

[4] Google DeepMind, The story of AlphaGo so far, *www.deepmind.com.*

[5] Google LYNA, *https://ai.googleblog.com/2018/10/applying-deep-learning-to-metastatic.html.*

[6] Humanitas, *https://www.humanitas.it/news/18050-google-glass-esordio-sala-operatoria.*

[7] Google, Android, *https://developer.android.com/.*