



UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Laurea Magistrale in Ingegneria Informatica

Documentazione progetto per il corso di PROGETTAZIONE, ALGORITMI E COMPUTABILITÀ

Prof.ssa:

Patrizia Scandurra

Studenti:

Giorgia BRESSANELLI

Matricola n. 1053903

Lorenzo FERRARI

Matricola n. 1053161

ANNO ACCADEMICO 2023 / 2024

Indice

| | | |
|----------|--|-----------|
| 0 | Iterazione 0 | 4 |
| 0.1 | Introduzione del sistema e funzionamento | 4 |
| 0.2 | Requisiti | 6 |
| 0.2.1 | Analisi dei requisiti | 6 |
| 0.2.2 | Requisiti funzionali | 9 |
| 0.2.3 | Requisiti non funzionali | 12 |
| 0.3 | Architettura del sistema | 13 |
| 0.3.1 | Deployment diagram - informale | 13 |
| 0.3.2 | Deployment diagram - formale | 13 |
| 0.3.3 | Design pattern MVC | 14 |
| 0.4 | Tecnologie utilizzate | 16 |
| 1 | Iterazione 1 | 17 |
| 1.1 | Diagramma dei tipi di dato | 17 |
| 1.2 | Modellazione database | 18 |
| 1.3 | Diagramma interfacce | 19 |
| 1.4 | Component diagram e Deployment diagram | 20 |
| 1.5 | Diagrammi di flusso | 23 |
| 2 | Iterazione 2 | 25 |
| 2.1 | Introduzione | 25 |
| 2.2 | UC12: Login | 27 |
| 2.3 | UC21: Logout | 27 |
| 2.4 | UC13: Compilazione modulo emergenza | 27 |
| 2.5 | UC14: Invio allarme emergenza | 27 |
| 2.6 | UC15: Comunicazione con capo servizio | 28 |
| 2.7 | UC16: Visualizzazione stato mezzi SOREU | 28 |
| 2.8 | UC18 e UC31: Scelta ospedale di destinazione | 28 |
| 2.9 | UC20: Visualizzazione report fine emergenza | 28 |
| 2.10 | UC28: Memorizzazione dati | 28 |
| 2.11 | UC29: Ricerca AAT più vicina | 28 |
| 2.12 | Analisi statica | 29 |
| 2.13 | Analisi dinamica | 30 |
| 3 | Iterazione 3 | 32 |
| 3.1 | Introduzione | 32 |
| 3.2 | UC1: Login | 33 |

| | | |
|-----|---|----|
| 3.3 | UC9: Logout | 33 |
| 3.4 | UC3: Accettazione chiamata emergenza | 33 |
| 3.5 | UC4 e UC5: Comunicazione con centrale | 33 |
| 3.6 | Analisi statica | 34 |
| 3.7 | Analisi dinamica | 34 |

0 Iterazione 0

0.1 Introduzione del sistema e funzionamento

Il progetto di studio vuole cercare di esaminare ed esporre una differente e più centralizzata gestione delle emergenze che vengono ricevute dal 118.

In Lombardia la gestione delle chiamate d'emergenza è affidata all'ente AREU (Agenzia Regionale Emergenza Urgenza) che coordina quattro centrali operative dislocate in altrettante macroaree chiamate SOREU (Sale Operative Regionali dell'Emergenza Urgenza):

- SOREU Area Metropolitana Milano
- SOREU Area dei Laghi a Como
- SOREU Area Alpina a Bergamo
- SOREU Area della Pianura a Pavia

Le SOREU elaborano le chiamate di soccorso sanitario con l'invio dei mezzi necessari e dei volontari per garantire il completamento dell'urgenza. Le sale operative sono affiancate dalle AAT (Articolazioni Aziendali Territoriali) che si occupano dell'organizzazione logistica del personale sanitario e della distribuzione, utilizzo e manutenzione dei mezzi di soccorso.

L'attenzione del progetto di studio verrà rivolta in particolare alla gestione delle chiamate relative alle SOREU della Pianura a Pavia.

Attualmente l'iter è così strutturato:

- Una persona chiama il 112 per segnalare un'emergenza;
- Il 112 classifica l'emergenza e inoltra la chiamata al 118 nel caso in cui sia un'emergenza medica;
- L'infermiere o medico del 118 fa alcune domande per capire cos'è successo, quali sono i problemi del paziente e i suoi dati;
- Il 118 inoltra a sua volta la chiamata all'ambulanza più vicina al luogo dell'evento. L'allarme arriva sul computer di bordo dell'ambulanza, sul telefono/palmare dell'equipaggio e sul computer della sede;
- Il capo squadra conferma la lettura dell'avviso, prepara la squadra e l'ambulanza, con volontari e attrezzatura;
- L'ambulanza parte per affrontare l'emergenza;

- Si presta soccorso e si fa la valutazione dei parametri se possibile;
- Durante il soccorso, il capo squadra è in continua comunicazione con la sede di Pavia tramite palmare;
- La centrale avvisa l'ambulanza in quale ospedale portare il paziente;
- L'ambulanza consegna il paziente alla struttura ospedaliera indicata e rientra in sede;
- In sede viene chiusa la scheda del paziente e la si carica sul portale del PC (viene rimandata a Pavia).

Lo scopo di tale progetto è aggregare la ricezione delle chiamate e la selezione delle modalità adatte alla risoluzione dell'intervento relativo all'emergenza in entrata.

0.2 Requisiti

0.2.1 Analisi dei requisiti

L'analisi dei requisiti consiste nell'identificazione degli obiettivi primari dell'applicazione partendo dallo schema tutt'ora utilizzato. Per fare ciò è necessario considerare gli utenti finali del sistema e i comportamenti desiderati da tali attori.

Deve essere gestito il processo di comunicazione che avviene tra gli operatori sul posto e i medici/infermieri presenti in sede operativa.

In ogni turno è presente una squadra composta da:

- **Capo servizio o Capo squadra:** spiega in cosa consiste la chiamata che è stata ricevuta, organizza la squadra e comunica per tutta la durata del servizio con l'operatore del 118
- **Autista:** riceve l'indirizzo e il percorso da seguire per recarsi nel luogo dell'emergenza, interagisce con la sede operativa tramite apposita interfaccia presente sull'ambulanza per comunicare lo stato del soccorso
- **Operatore DAE:** volontario abilitato al soccorso, può interagire con la sede operativa tramite interfaccia operativa sull'ambulanza, come l'autista

Gli attori esterni alla squadra che presta soccorso sono:

- **Operatore 118:** è un medico o infermiere che gestisce a distanza tutta la parte logistica e decisionale dell'emergenza, è l'operatore in continua comunicazione con la squadra sul posto
- **Medico automedica:** non è sempre presente, dipende dalla gravità dell'emergenza. Comunica direttamente con la centrale operativa tramite radio o telefono/palmare
- **Infermiere automedica o autoinfermieristica:** come il medico, viene richiamato solo quando necessario

Durante tutta la comunicazione le principali informazioni che vengono scambiate sono le seguenti:

- *Luogo* (indirizzo) in cui è avvenuta l'emergenza;
- *Quando* è successo: non necessariamente è un'emergenza appena avvenuta, può essere un evento accaduto precedentemente ma che ha ripercussioni attuali
- *Nominativo* del paziente o pazienti
- *Sesso* del paziente o pazienti
- *Età* del paziente o pazienti
- *Motivo* della chiamata; ad esempio se è una caduta, se è un incidente stradale e quindi quanti sono i coinvolti o di che tipo di incidente si tratta, se è un problema respiratorio, se è un trauma e in quale parte del corpo si trova, se è un'aggressione, se è un'intossicazione, ecc.
- *Cosciente* o non cosciente o se ha avuto perdite di coscienza
- *Respira* o non respira o respira a fatica
- *Patologie*: ad esempio se il paziente soffre di ipertensione o ipotensione
- *Codice gravità*: codice assegnato dalla centrale operativa, può essere rosso, giallo o verde. Viene utilizzato anche per decidere se è necessario introdurre mezzi di supporto avanzato (MSA)
- *Numero missione*
- *Orario di attivazione*: dato utile alla centrale per il calcolo dei tempi di attesa
- *Presenza di mezzi di supporto*: eventualmente possono essere attivati e inviati in un secondo momento
- *Stati della squadra*

Gli stati che possono essere comunicati sia tramite interfaccia presente sull'ambulanza sia tramite telefono/palmare sono:

- CONFERMA: conferma la presa visione della chiamata
- PARTITO: indica che la squadra esce dalla sede
- SUL POSTO: avvisa dell'arrivo dell'equipaggio sul luogo dell'evento
- DESTINAZIONE OSPEDALE: richiesta dell'ospedale in cui portare il paziente; oltre all'ospedale di destinazione viene inviato anche il codice di gravità
- ARRIVO: indica l'arrivo in ospedale dell'ambulanza, prima che venga scaricato il paziente
- LIBERO OSPEDALE: avvisa che l'ambulanza non ha più il paziente a carico; il mezzo è in sosta presso l'ospedale ed è disponibile per rispondere ad un'altra emergenza
- RIENTRO IN SEDE: avvisa che l'ambulanza è in tragitto verso la sede; in questo stato l'equipaggio è sempre disponibile per rispondere ad un'altra possibile emergenza e quindi la centrale può dirottare il mezzo verso l'emergenza successiva senza rientrare in sede
- IN SEDE: indica il rientro dell'ambulanza in sede
- NON DISPONIBILE: avvisa che l'ambulanza non è pronta per rispondere ad un'altra chiamata. Viene attivato questo stato quando è necessario ripristinare l'attrezzatura, la pulizia e l'ordine del mezzo
- RIENTRO IN SEDE LIBERO NON TRASPORTA REGOLARE: avvisa che dopo aver prestato soccorso l'ambulanza non ha nessun paziente a carico. semplicemente non è necessario un trasporto in ospedale, accade per chiamate semplici o di poca importanza

0.2.2 Requisiti funzionali

Casi d'uso Capo Servizio Un capo squadra può utilizzare il sistema nei seguenti casi

| Codice | Nome | Priorità |
|--------|------------------------------------|----------|
| UC1 | Login | A |
| UC2 | Comunicazione dati turno | A |
| UC3 | Accettazione chiamata emergenza | A |
| UC4 | Comunicazione con centrale | A |
| UC5 | Comunicazione dati paziente | A |
| UC6 | Richiesta MSA | B |
| UC7 | Visualizzazione arrivo MSA | B |
| UC8 | Compilazione report fine emergenza | B |
| UC9 | Logout | A |

Tabella 1: Casi d'uso capo servizio

Casi d'uso Operatore ambulanza L'operatore in ambulanza si occupa solamente della comunicazione degli stati

| Codice | Nome | Priorità |
|--------|--|----------|
| UC10 | Comunicazione degli stati | B |
| UC11 | Visualizzazione ospedale di destinazione | B |

Tabella 2: Casi d'uso operatore ambulanza

Casi d'uso Operatore 118 L'operatore 118 si occupa di tutta la gestione di supporto ai volontari che rispondono all'emergenza

| Codice | Nome | Priorità |
|--------|---------------------------------------|----------|
| UC12 | Login | A |
| UC13 | Compilazione modulo emergenza | A |
| UC14 | Invio allarme emergenza | A |
| UC15 | Comunicazione con capo servizio | A |
| UC16 | Visualizzazione stato mezzi SOREU | A |
| UC17 | Richiesta dati aggiuntivi emergenza | B |
| UC18 | Scelta ospedale di destinazione | A |
| UC19 | Attivazione MSA | B |
| UC20 | Visualizzazione report fine emergenza | B |
| UC21 | Logout | A |

Tabella 3: Casi d'uso operatore centrale

Casi d'uso Operatore MSA Gli operatori di supporto possono interagire nei seguenti casi

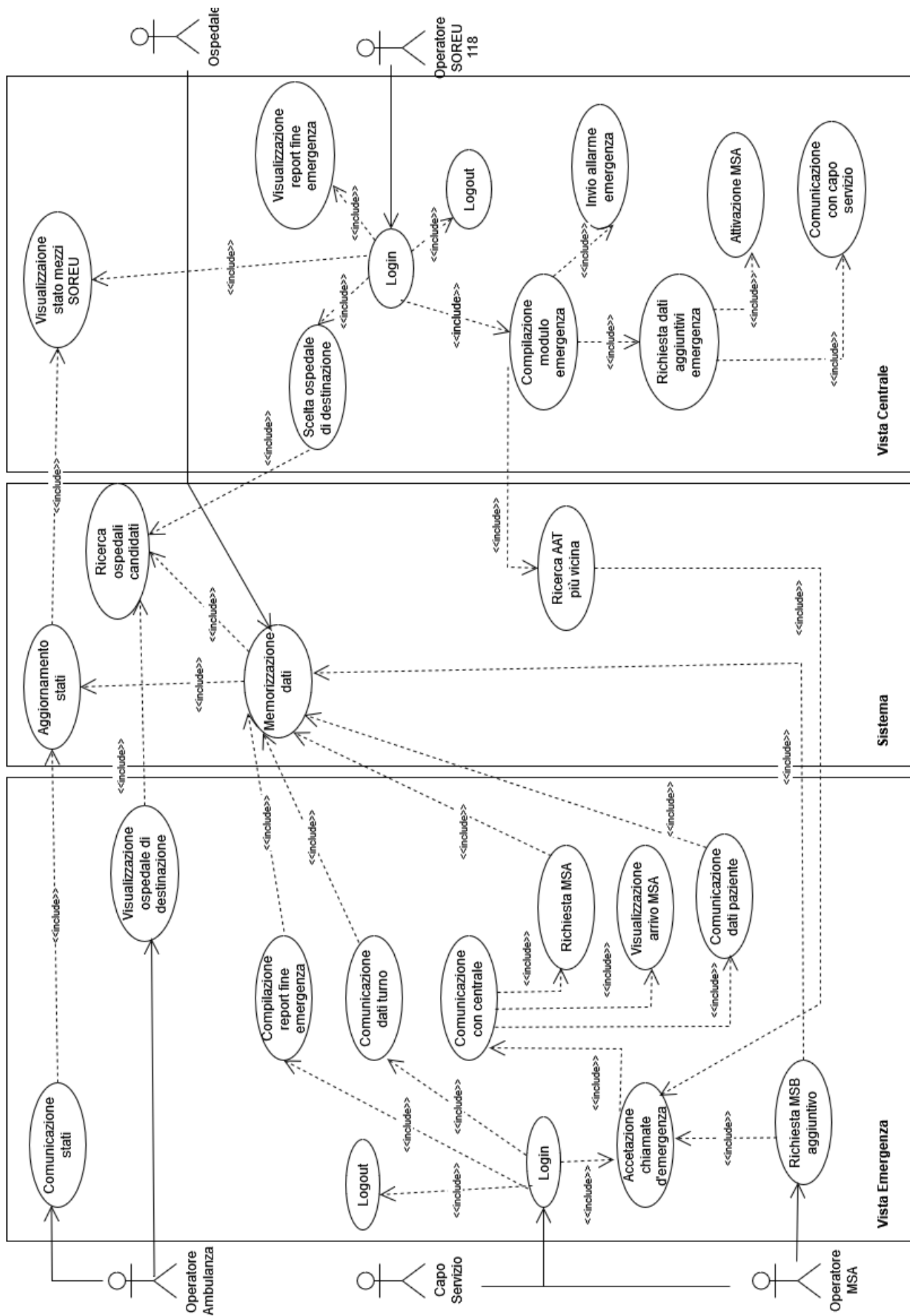
| Codice | Nome | Priorità |
|--------|---------------------------------|----------|
| UC22 | Login | B |
| UC23 | Comunicazione dati paziente | B |
| UC24 | Comunicazione con centrale | B |
| UC25 | Richiesta MSB aggiuntivo | B |
| UC26 | Accettazione chiamata emergenza | B |
| UC27 | Logout | B |

Tabella 4: Casi d'uso operatore MSA

Casi d'uso sistema Il sistema deve fornire le seguenti funzionalità

| Codice | Nome | Priorità |
|--------|----------------------------|----------|
| UC28 | Memorizzazione dati | A |
| UC29 | Ricerca AAT più vicina | A |
| UC30 | Aggiornamento stati | A |
| UC31 | Ricerca ospedali candidati | A |

Tabella 5: Casi d'uso operatore MSA



0.2.3 Requisiti non funzionali

Il progetto verrà sviluppato tenendo conto anche di alcuni requisiti non funzionali quali efficienza, usabilità, rapidità e manutenibilità.

Efficienza Uno dei requisiti più importanti per il progetto, viene garantito dalla standardizzazione delle informazioni inviate nella comunicazione. Le comunicazioni seguono dei protocolli stabiliti per evitare informazioni superficiali o non richieste e quindi l'ipotetica perdita di tempo/efficienza.

Rapidità Un altro aspetto importante per il progetto, è necessario che la ricerca di una struttura vicina sia veloce. Deve essere veloce anche la comunicazione che avviene tra le parti durante un'emergenza.

Usabilità Tale requisito viene garantito attraverso la scelta di fornire una piena visione del documento relativo all'emergenza ad ogni membro del personale incaricato di gestirla. Questo può essere visto osservando la topologia del sistema dove il database viene allocato su un hosting online al quale ogni operatore potrà accedervi sempre mediante un dispositivo, come ad esempio un palmare o un portatile.

Manutenibilità Il requisito della manutenibilità verrà garantito mediante la definizione e allocazione delle risorse persistenti necessarie, ovvero tutte le informazioni relative al personale medico e dei mezzi delle relative AAT. Questo va a favore del fatto che, in caso di modifiche al sistema di emergenza della SOREU, quali aggiunta o chiusura di AAT, queste possano essere diffuse all'interno del sistema.

0.3 Architettura del sistema

Per meglio comprendere l'idea dell'architettura alla base della soluzione proposta, sono stati realizzati due deployment diagram: il primo diagramma fornisce una visione generale del sistema che viene proposto (Figura 1), mentre il secondo (Figura 2) mette in evidenza l'allocazione effettiva delle componenti Hardware e Software dell'architettura.

0.3.1 Deployment diagram - informale

Il Deployment diagram informale evidenzia gli attori e i dispositivi informatici installati sui rispettivi mezzi di intervento, come ambulanze e mezzi di supporto avanzato, dando particolare enfasi all'organizzazione e interazione tra essi. L'architettura posiziona un server centrale che interagisce con un database, nel quale sono memorizzate tutti i dati e le informazioni necessarie al corretto funzionamento del sistema, e con componenti hardware e software necessari per comunicare con il server stesso:

- Interfaccia utilizzata dall'operatore SOREU 112
- Palmare utilizzato dal capo servizio della AAT
- Monitor presente sull'ambulanza della AAT
- Interfaccia utilizzata dall'operatore sanitario AAT
- Monitor presente sulla macchina medica AAT
- Navigatore presente sull'ambulanza della AAT

0.3.2 Deployment diagram - formale

Il deployment diagram formale (Figura 2) mette in evidenza le stesse informazioni presentate precedentemente, fornisce una visione incentrata sulla tecnologia utilizzata per descrivere i diversi componenti riservando uno sguardo alla tipologia di sviluppo architetturale utilizzata per il sistema.

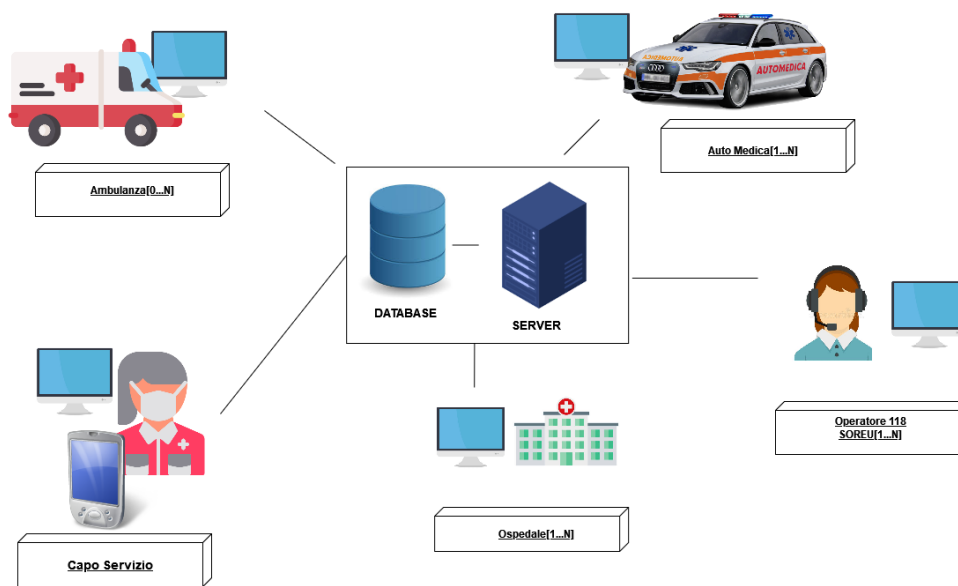


Figura 1: Deployment Diagram informale

0.3.3 Design pattern MVC

Il sistema di gestione delle emergenze e i suoi componenti verranno sviluppati sfruttando il Design Pattern chiamato Model View Controller (MVC), composto da tre layer:

- **Model:** definisce la struttura, l'organizzazione dei dati dell'applicazione, fornendo inoltre i metodi per accedere a quest'ultimi. In questo layer avremo il database che memorizza tutti i dati relativi alle emergenze.
- **View:** visualizza i dati contenuti nel Model presentandoli all'utente, occupandosi inoltre dell'interazione tra l'utente e la logica applicativa del sistema. In questo layer vengono raccolte le informazioni scambiate nella comunicazione e vengono poi elaborate.
- **Controller:** sulla base delle informazioni ricevute dall'operatore, tipicamente per mezzo della view, vengono inserite e aggiornate, mantenendo aggiornati anche i dati resi disponibili negli altri layer.

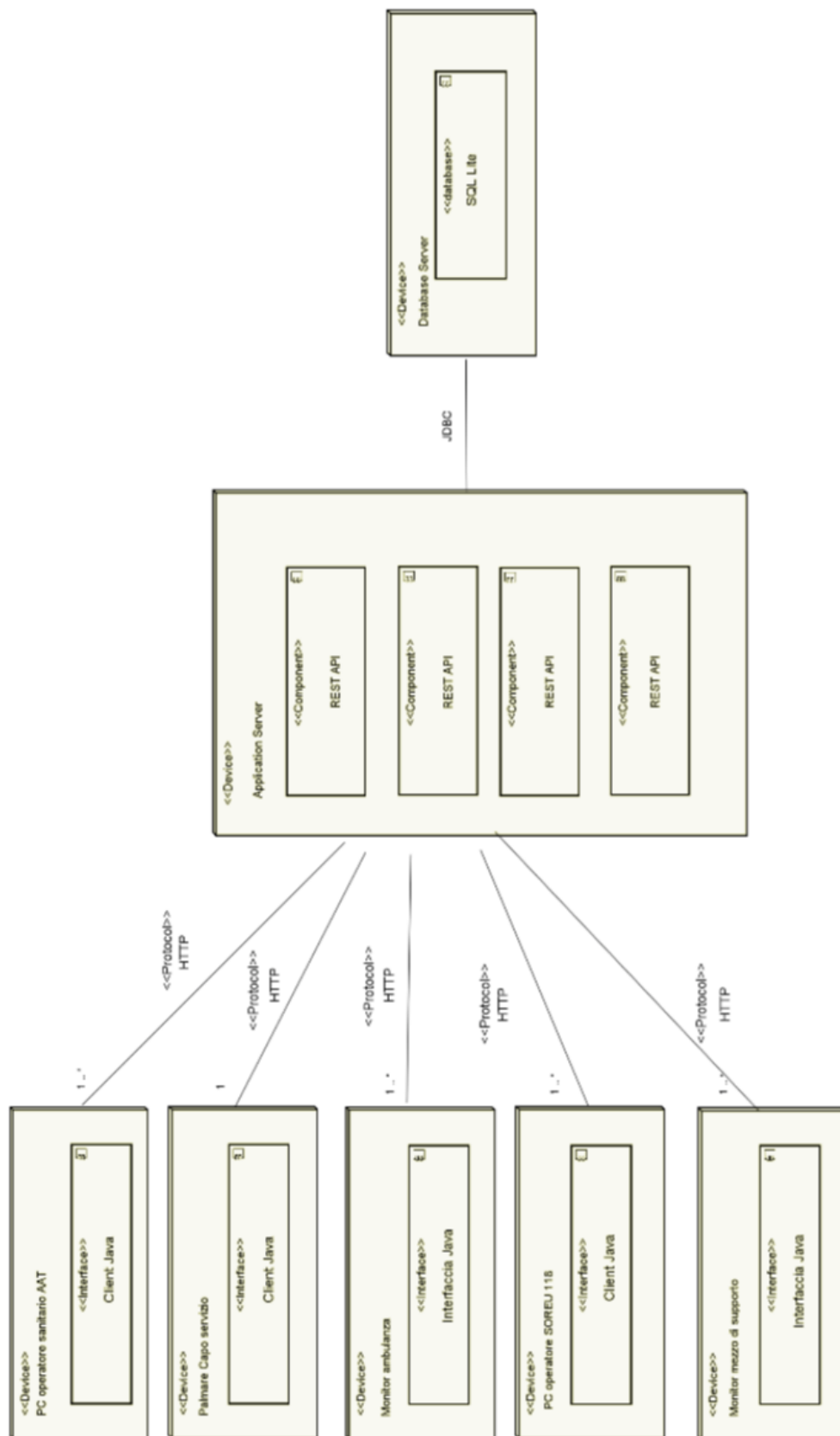


Figura 2: Deployment Diagram formale

0.4 Tecnologie utilizzate

| Tool/Tecnologia | Utilità |
|-------------------------|--|
| Draw.io, PlantUML | Software open source per la realizzazione di grafici e diagrammi UML |
| Google Drive | Servizio di storage per condivisione documenti e immagini |
| Github | Servizio di versioning per condivisione del codice e documentazione |
| TeXnic center, OverLeaf | Software per la scrittura della documentazione in \LaTeX |
| JUnit | Plugin per analisi dinamica del codice |
| Qodana | Plugin per analisi statica del codice |
| IntelliJ | IDE, ambiente di sviluppo per linguaggio Java |
| MySQL Workbench | Servizio di storage di dati |

1 Iterazione 1

In questa iterazione, il focus del progetto verte sul completamento della fase di progettazione. I tipi di dato, le interfacce e le funzionalità ideate sono una prima concretizzazione dell'applicazione, pertanto subiranno delle modifiche nel corso delle successive iterazioni.

1.1 Diagramma dei tipi di dato

Il diagramma riportato in figura 3 mostra i tipi di dato che verranno utilizzati nelle classi per lo sviluppo dell'applicazione.

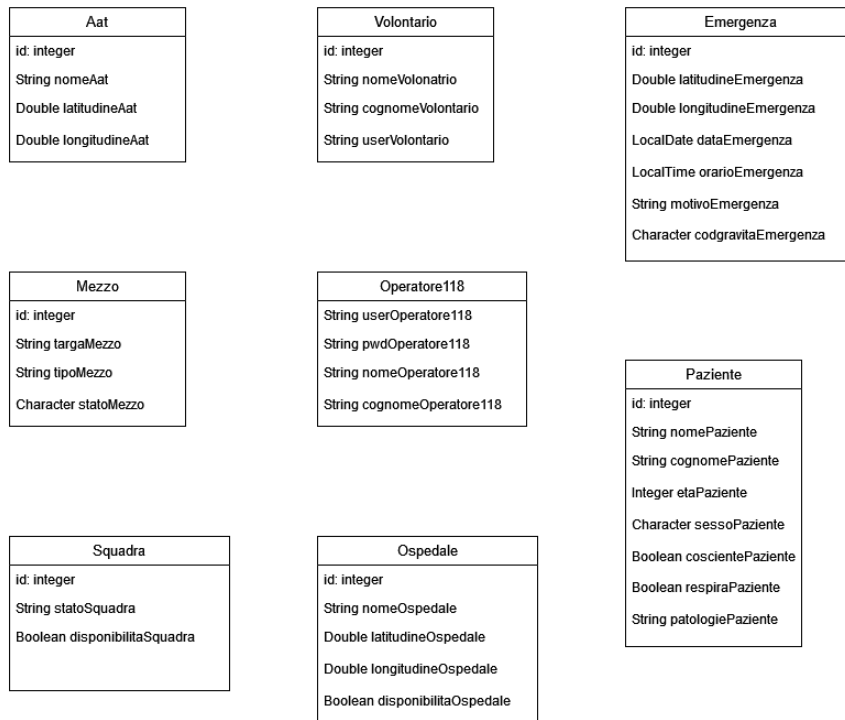


Figura 3: Diagramma delle classi per tipi di dato

1.2 Modellazione database

Per il funzionamento del software è stato modellato un database adatto ai tipi di dato definiti e per sostenere le interazioni necessarie tra i vari componenti.

Di seguito, la figura 4 riporta una rappresentazione del database progettato.

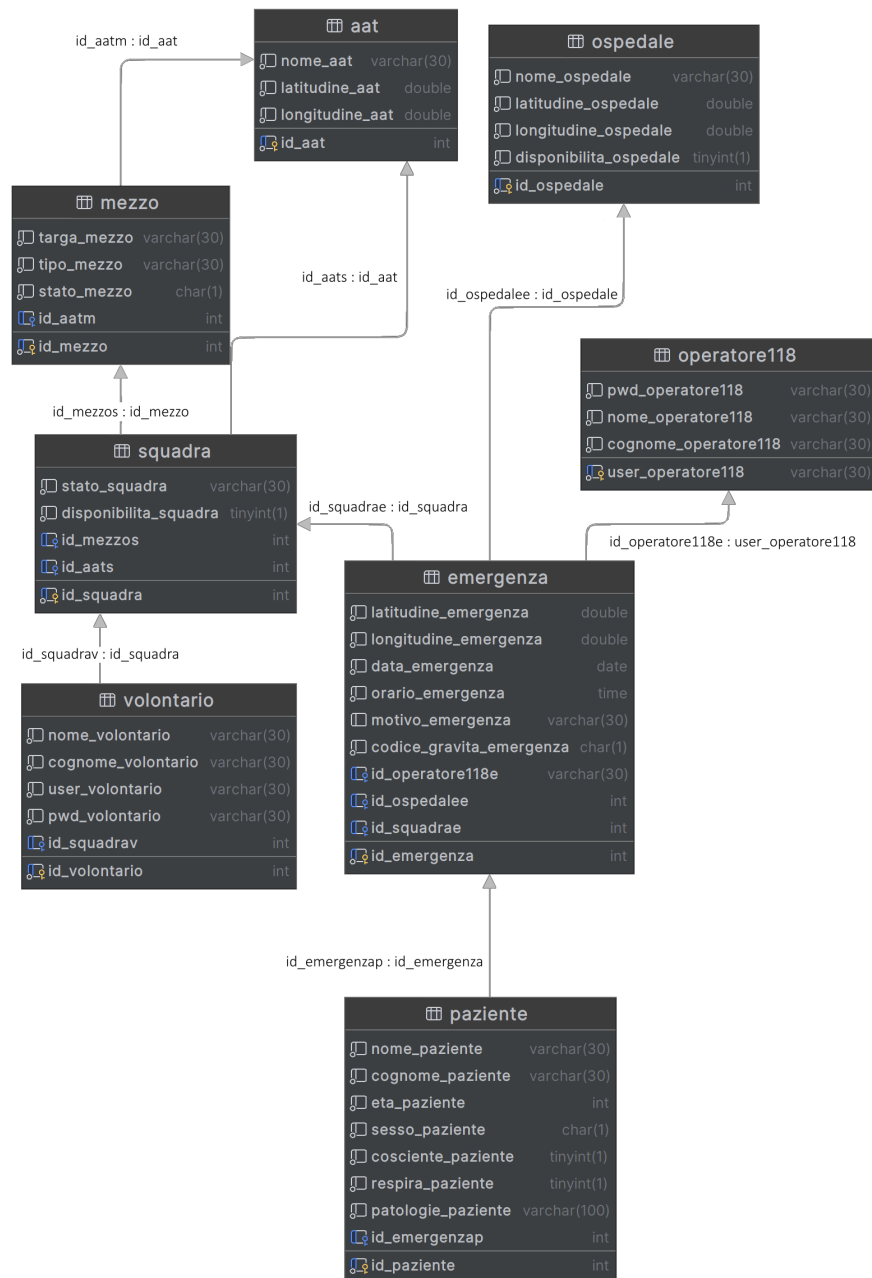


Figura 4: Schema logico database progettato

1.3 Diagramma interfacce

Infine, in figura 5 viene mostrato il diagramma contenente le interfacce proposte, con i relativi metodi e la tipologia di dati scambiati in input e output. Grazie a questo diagramma vengono visualizzati quali saranno i servizi pubblicati nell'applicazione.

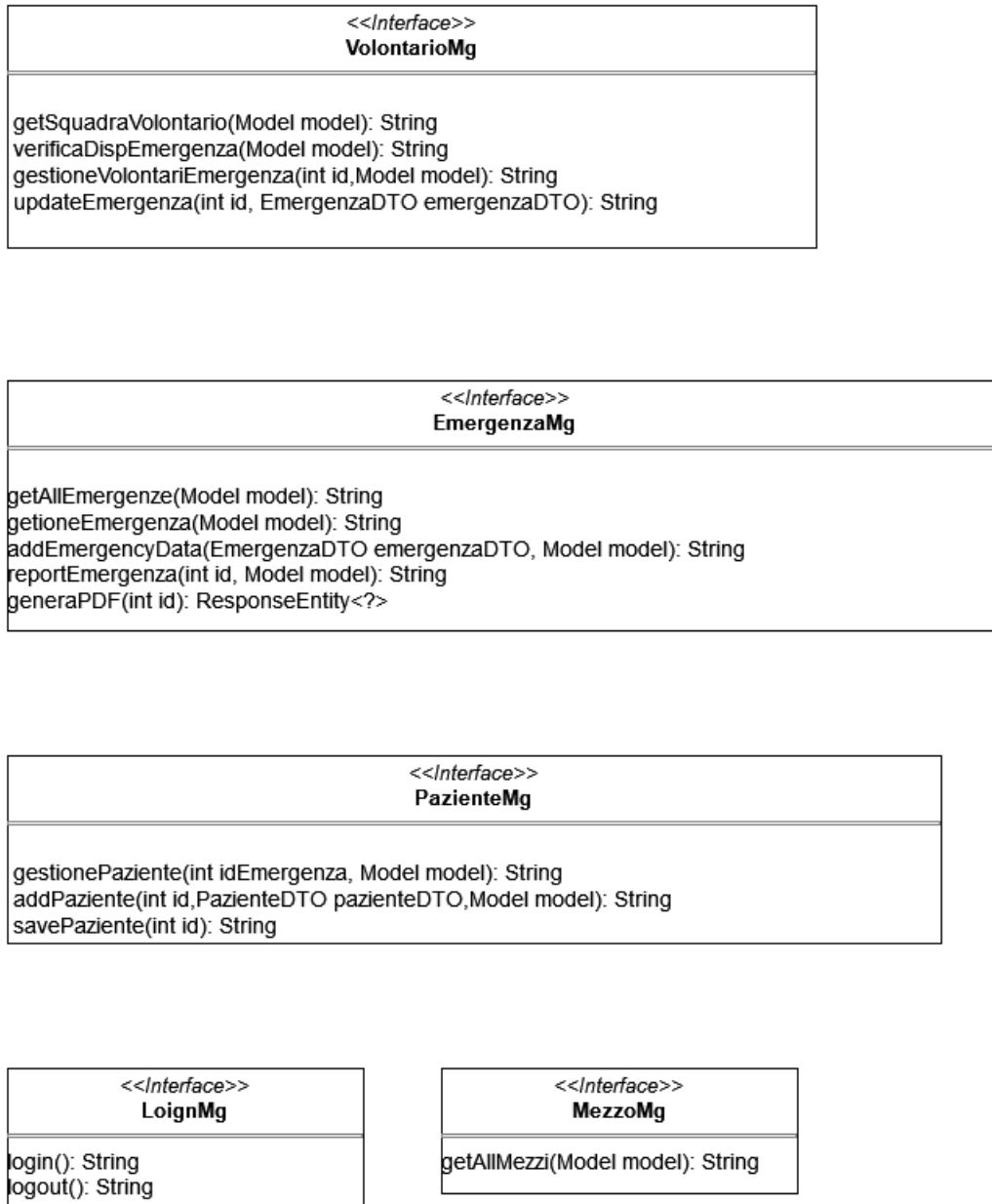


Figura 5: Diagramma delle interfacce

1.4 Component diagram e Deployment diagram

Partendo dalla progettazione effettuata fino ad ora, è stato ricavato il relativo Component Diagram (figura 6) in cui sono stati introdotti i seguenti componenti:

- *control*: componente back-end che permette di gestire la logica di funzionamento del sistema e di interagire con utente e database grazie all'esposizione delle API;
- *data*: componente utilizzato per la gestione del database;
- *boundary*: componente front-end che permette di gestire le interazioni con gli attori attivi durante un'emergenza.

Infine, tramite il Deployment Diagram presente in figura 7 è possibile visualizzare la struttura software e hardware del sistema in esame.

I componenti mostrati sono:

- il database che funge da risorsa persistente in cui vengono memorizzati i dati generati nel corso dell'emergenza;
- il web-server grazie al quale vengono pubblicate le API lato front-end;
- i palmari in possesso degli operatori, grazie ai quali vengono trasmesse le informazioni inerenti all'emergenza in corso.

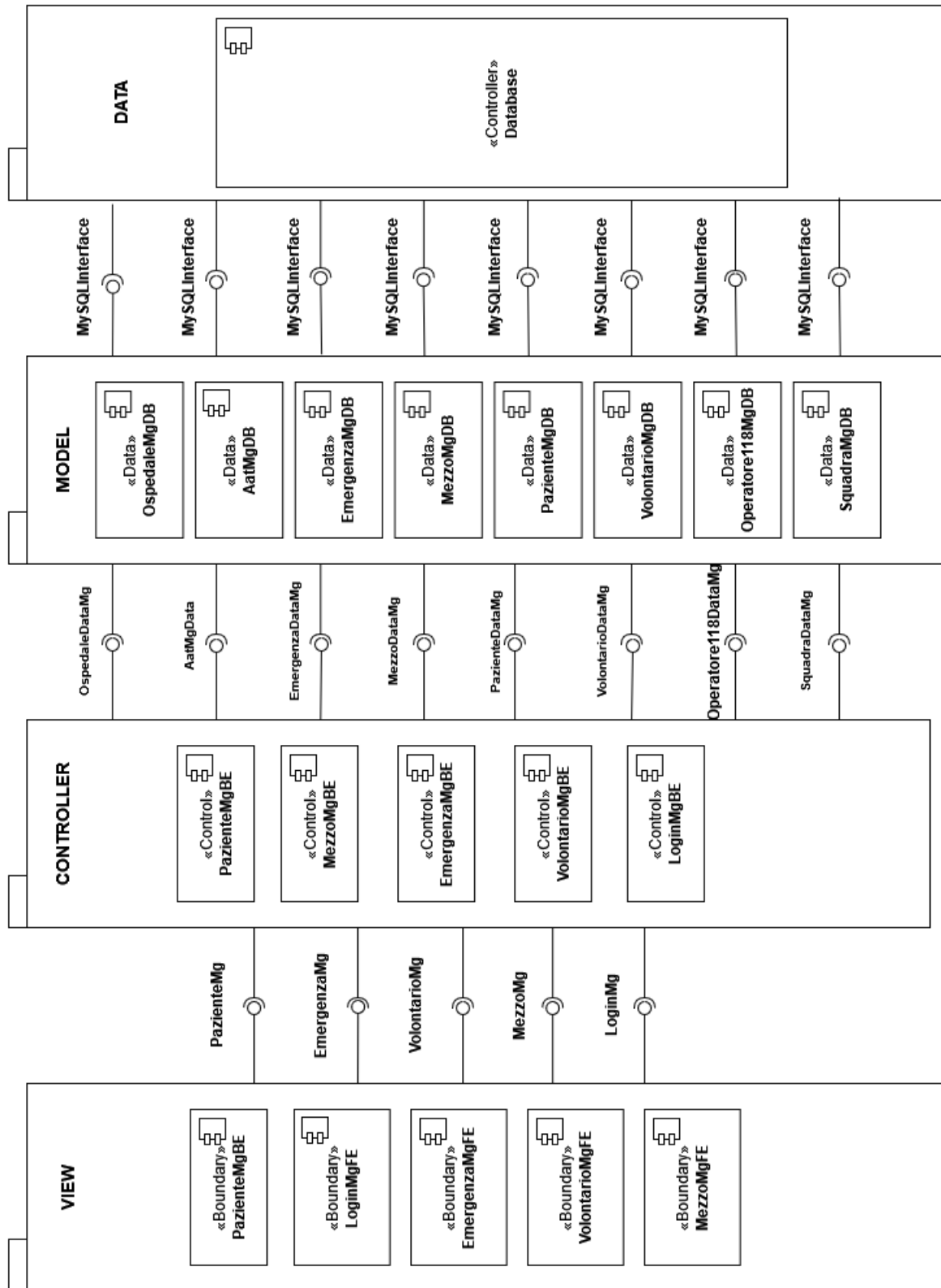


Figura 6: Component Diagram

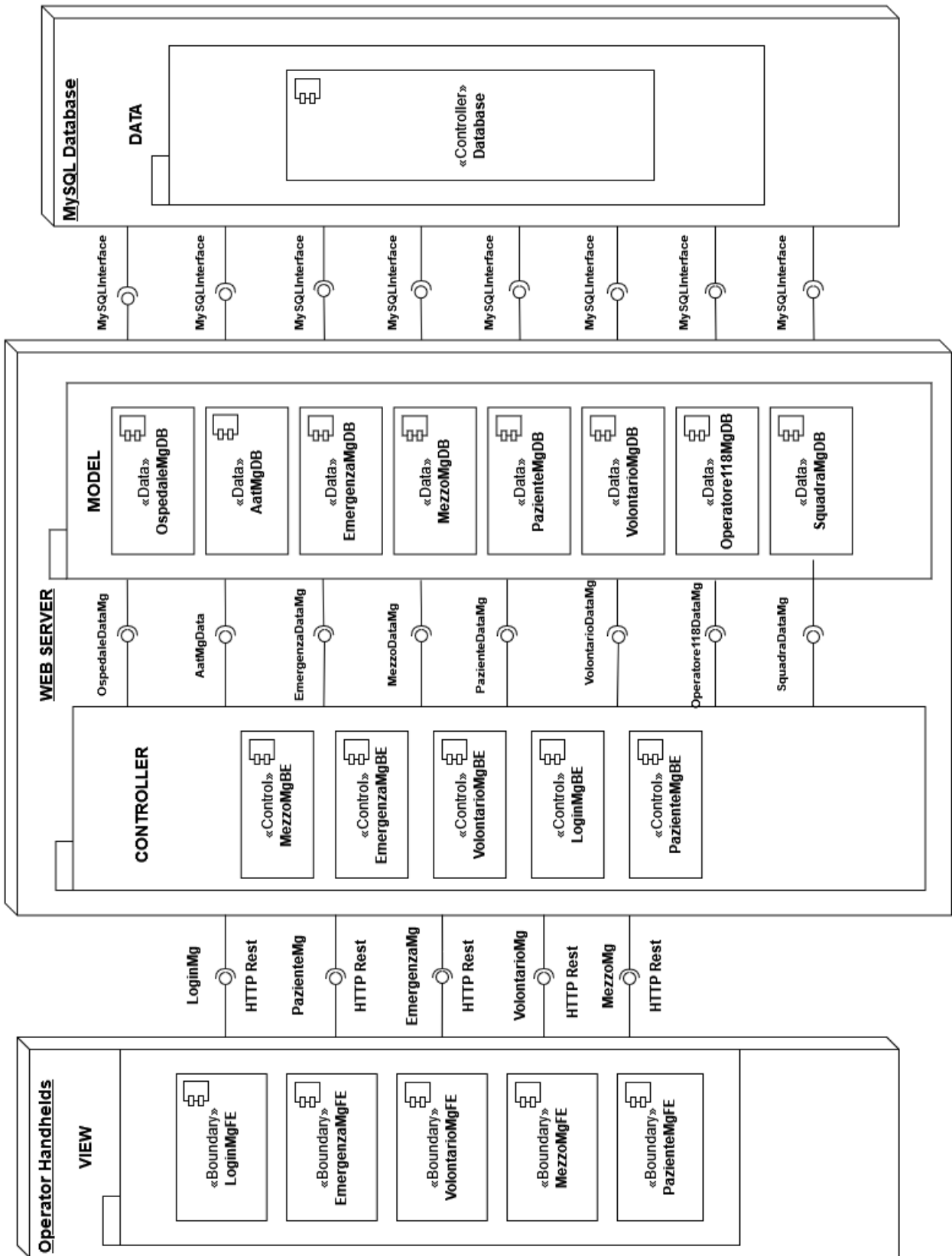


Figura 7: Deployment Diagram

1.5 Diagrammi di flusso

Di seguito vengono presentati i diagrammi di flusso per evidenziare la sequenzialità delle operazioni svolte dai due attori principali.

Bisogna considerare inoltre che i due flussi sono sincroni e interagiscono tra di loro tramite lo scambio di informazioni relative all'emergenza in atto.

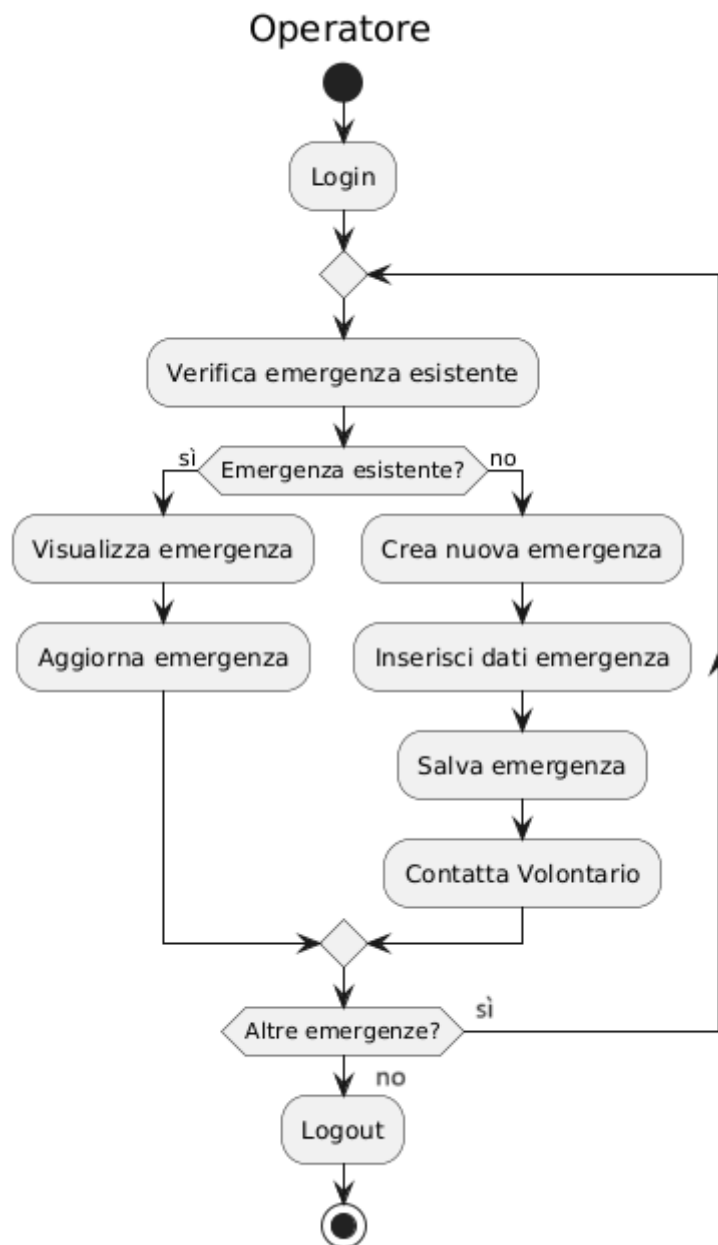


Figura 8: Diagramma di flusso dell'operatore 118

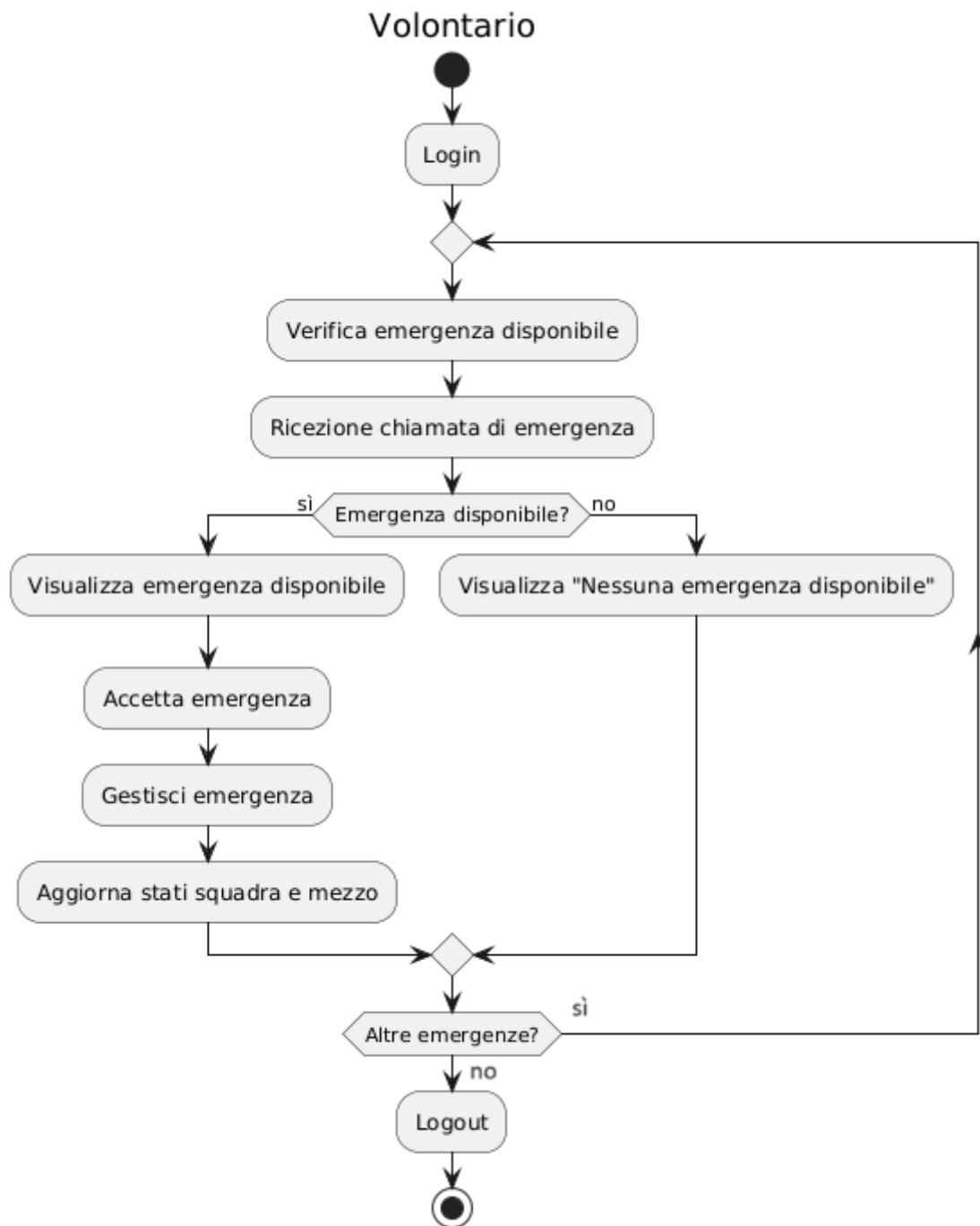


Figura 9: Diagramma di flusso del volontario (capo servizio)

2 Iterazione 2

2.1 Introduzione

In questa iterazione si procede con l'implementazione dei casi d'uso essenziali al funzionamento del software (priorità alta).

In particolare, i casi d'uso selezionati per lo sviluppo sono quelli relativi all'**operatore 118**, essendo uno dei due attori essenziali alla comunicazione durante un'emergenza.

Per proseguire con l'implementazione inoltre è risultato necessario aggiornare le classi relative all'operatore 118 e al volontario, aggiungendo una variabile che ne specificasse il ruolo (Figura 10).

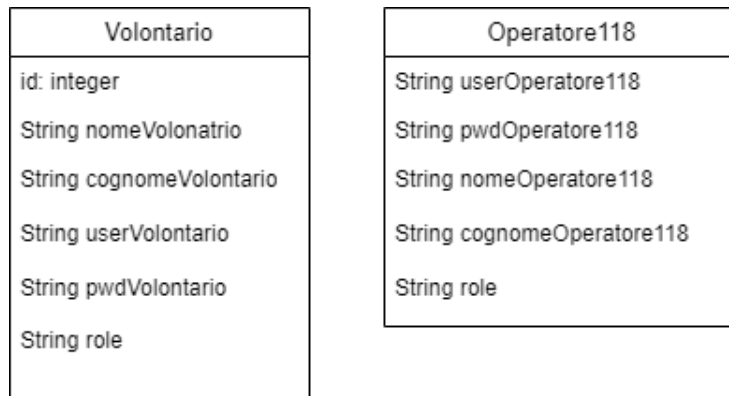


Figura 10: Classi aggiornate

Nello specifico, i casi d'uso scelti sono i seguenti:

- UC12 Login
- UC13 Compilazione modulo emergenza
- UC14 Invio allarme emergenza
- UC15 Comunicazione con capo servizio
- UC16 Visualizzazione stato mezzi SOREU
- UC18 Scelta ospedale di destinazione
- UC20 Visualizzazione report fine emergenza
- UC21 Logout

Inoltre, per poter proseguire successivamente con lo sviluppo dei casi d'uso relativi al capo servizio e per poter integrare i casi d'uso dell'operatore 118, sono stati implementati anche i casi d'uso del sistema:

- UC28 Memorizzazione dati
- UC29 Ricerca AAT più vicina
- UC31 Ricerca ospedali candidati

Di seguito verranno analizzate nel dettaglio le fasi dello sviluppo e i test effettuati.

2.2 UC12: Login

Descrizione Classica fase di login riservata agli operatori 118 e ai capi servizio.

Ogni operatore possiede un proprio username e una password che gli permetteranno di accedere alla schermata di gestione emergenza, differente da quella riservata al volontario. Dopo aver effettuato il login con successo, l'operatore 118 visualizza un'interfaccia che gli permette di

- Rispondere ad una chiamata e contattare una sede AAT da inviare sul posto (azione simulata tramite bottone "*Crea Emergenza*")
- Visualizzare tutte le emergenze concluse
- Visualizzare i report delle emergenze precedenti ed eventualmente generarne altri
- Visualizzare l'emergenza appena inserita dopo aver confermato i dati

2.3 UC21: Logout

Descrizione Classica fase di logout.

2.4 UC13: Compilazione modulo emergenza

Descrizione Dopo aver risposto ad una chiamata al 118, l'operatore visualizza un'interfaccia che gli permette di

- Inserire tramite apposito form i dati ottenuti dalla chiamata
- Visualizzare l'ospedale selezionato dal software ed eventualmente anche gli ospedali candidati per ricevere i pazienti coinvolti nell'emergenza
- Chiudere l'emergenza

2.5 UC14: Invio allarme emergenza

Descrizione Coincide con la chiamata alla sede AAT la quale potrà prendere in carico l'emergenza oppure rifiutarla.

Per semplicità, sono state considerate solamente le squadre aventi lo stato "IN SEDE" come valide per la risposta alle chiamate d'emergenza.

In seguito all'inserimento dei dati ottenuti dalla richiesta di soccorso, l'operatore conferma i dati memorizzati e visualizza lo stato attuale dell'emergenza appena inserita.

L'associazione dell'ultima emergenza con la squadra a cui ne è stata assegnata la gestione,

risulta come "*in attesa*" poiché ancora non sono stati gestiti gli aggiornamenti on-line. In futuro possono essere utilizzati dei Web Socket per fornire questa funzionalità.

2.6 UC15: Comunicazione con capo servizio

Descrizione Questa fase è rappresentata dalle informazioni passate tramite form che vengono caricati per entrambe le parti. Le informazioni sono le medesime citate nell'Iterazione 0.

Per poter sviluppare questo caso d'uso è stato necessario aggiungere uno step intermedio che permetta di memorizzare i dati prima che vengano inviati ai rispettivi form.

2.7 UC16: Visualizzazione stato mezzi SOREU

Descrizione Visualizzazione dell'elenco dei mezzi disponibili per il dipartimento.

2.8 UC18 e UC31: Scelta ospedale di destinazione

Descrizione Visualizzazione degli ospedali disponibili in ordine di vicinanza, calcolata sfruttando le coordinate dell'emergenza e delle strutture stesse. L'operatore ad ora non può forzare la scelta di una struttura che non coincide con la prima suggerita dal software. Si prevede questa implementazione in eventuali sviluppi futuri.

2.9 UC20: Visualizzazione report fine emergenza

Descrizione Visualizzazione delle emergenze memorizzate nel sistema con anche la possibilità di scaricare il documento pdf per ciascuna delle emergenze inserite.

2.10 UC28: Memorizzazione dati

Descrizione Memorizzazione nella base di dati di tutte le informazioni campionate durante la chiamata. Questa fase è essenziale per il momento in cui verrà generato il report di emergenza ma anche per la fase di "comunicazione" tra operatore e capo servizio.

2.11 UC29: Ricerca AAT più vicina

Descrizione Ricerca della sede da contattare più vicina al punto in cui ha avuto luogo l'emergenza. Il calcolo viene eseguito sfruttando la distanza tra i due punti, grazie alla conoscenza delle coordinate di entrambi.

L'operatore 118 può visualizzare direttamente i dati della squadra a cui l'emergenza è stata associata dal programma.

2.12 Analisi statica

Come citato nell'iterazione 0, per l'analisi statica del codice è stato utilizzato il plugin *Qodana*, integrato con *IntelliJ*, software utilizzato per la scrittura del codice.

Qodana permette di verificare l'esistenza di problemi come bugs, dichiarazioni confuse o non utilizzate, violazioni relative a convenzioni adottate per la codifica.

In particolare, in seguito all'analisi effettuata sul codice, sono stati ridefiniti dei metodi mentre altri sono stati completamente rimossi.

Le ulteriori problematiche evidenziate dal tool, visibili in figura 11, sono relative ad un uso non convenzionale di alcune classi per il trasferimento dei dati.

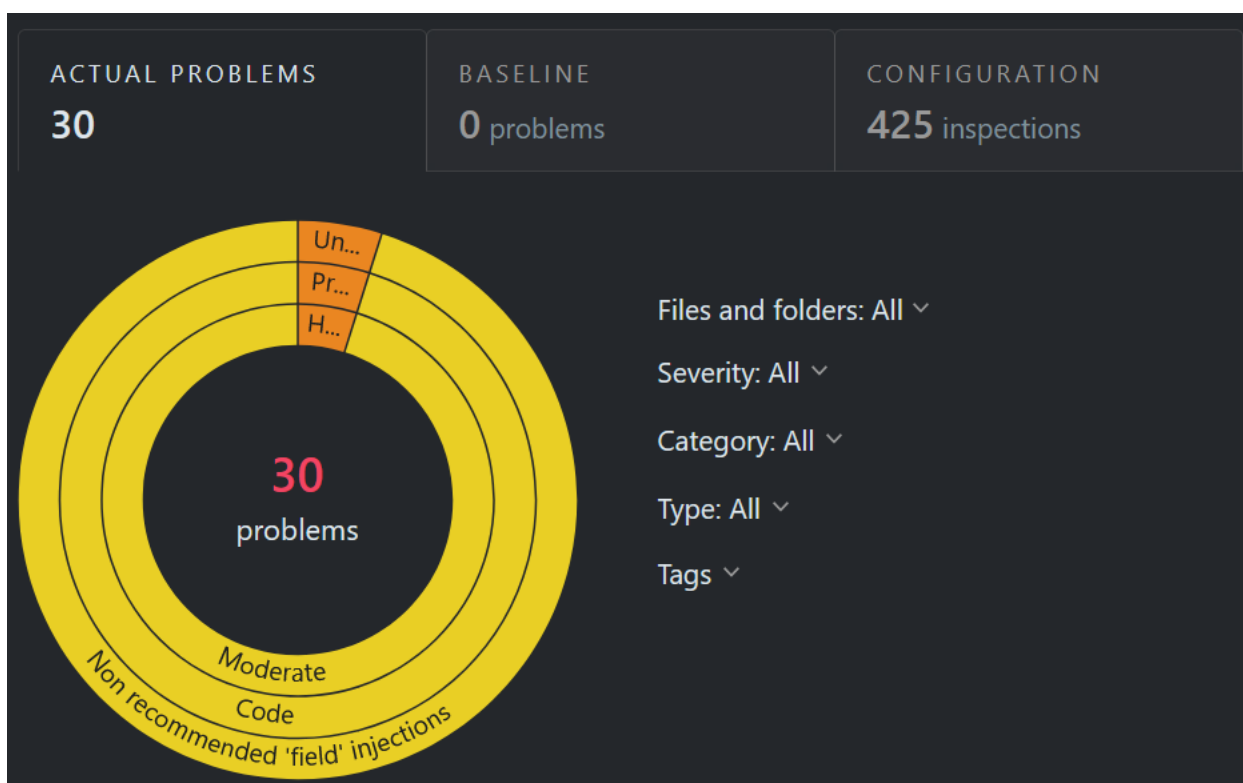


Figura 11: Risultato grafico analisi statica

2.13 Analisi dinamica

Per quanto riguarda la sezione relativa all'operatore sono state testate le seguenti API tramite IntelliJ:

Elenco di tutti i mezzi

GET http://localhost:8080/api/mezzo/all

Content-Type: application/json

X-XSRF-TOKEN: 8de86742-f827-4dd2-8685-759f975b538b

Report dettagliato dell'emergenza numero 1

GET http://localhost:8080/api/emergenza/reportEmergenza/1

Content-Type: application/json

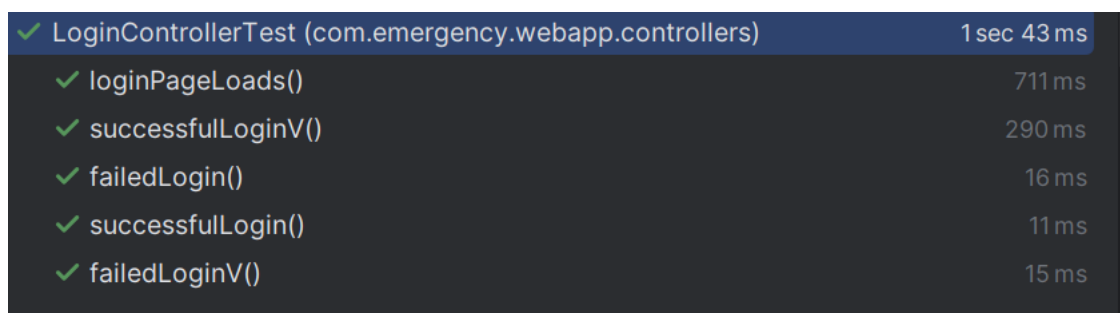
Generazione report dell'emergenza numero 1

GET http://localhost:8080/api/emergenza/generaPDF/1

Content-Type: application/json

Sfruttando *JUnit* i test eseguiti per la fase di login hanno avuto esito positivo, come si può vedere nella figura 12.

In merito ai test per le funzionalità riservate all'operatore 118 si rimanda alla dimostrazione pratica dell'esecuzione dell'applicativo. Non è stato possibile effettuare i test tramite JUnit a causa della presenza del componente SpringBoot Security che limita l'accesso al login.

A screenshot of a JUnit test runner interface showing the results of tests for the LoginControllerTest class. The interface has a dark background with green checkmarks indicating successful tests. The tests listed are loginPageLoads(), successfulLoginV(), failedLogin(), successfulLogin(), and failedLoginV(). The total execution time for the class is 1 sec 43 ms, and individual test times are shown in milliseconds.

| | |
|--|-------------|
| ✓ LoginControllerTest (com.emergency.webapp.controllers) | 1 sec 43 ms |
| ✓ loginPageLoads() | 711 ms |
| ✓ successfulLoginV() | 290 ms |
| ✓ failedLogin() | 16 ms |
| ✓ successfulLogin() | 11 ms |
| ✓ failedLoginV() | 15 ms |

Figura 12: Esiti esecuzione test login con JUnit

In particolare, i metodi implementati nella classe di test relativa ai login sono:

```
public class LoginControllerTest{
    //declarations
    @Test
    public void loginPageLoads() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.get("/login"))
                .andExpect(MockMvcResultMatchers.status().isOk());
    }

    @Test
    public void successfulLogin() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/login")
                .param("username", "11802")
                .param("password", "albuca"))
                .andExpect(MockMvcResultMatchers.status()
                        .is3xxRedirection())
                .andExpect(MockMvcResultMatchers.redirectedUrl
                        ("/api/emergenza/all"));
    }

    @Test
    public void failedLogin() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/login")
                .param("username", "11802")
                .param("password", "aaaaa"))
                .andExpect(MockMvcResultMatchers.status()
                        .is3xxRedirection())
                .andExpect(MockMvcResultMatchers.redirectedUrl
                        ("/login?error=true"));
    }
}
```

3 Iterazione 3

3.1 Introduzione

Per completare il funzionamento base dell'applicativo, per questa iterazione l'implementazione riguarda l'altro capo della comunicazione che avviene durante un'emergenza, ovvero l'attore **capo servizio**.

In particolare, i casi d'uso implementati sono i seguenti:

- UC1 Login
- UC3 Accettazione chiamata emergenza
- UC4 Comunicazione con centrale
- UC5 Comunicazione dati paziente
- UC9 Logout

Di seguito verranno analizzate nel dettaglio le fasi dello sviluppo e i test effettuati.

3.2 UC1: Login

Descrizione Classica fase di login riservata ai capi di servizio e agli operatori 118.

Dopo aver effettuato l'accesso, il capo servizio visualizza i dati forniti per le presenze del turno e in una sezione dedicata è possibile controllare se ci sono emergenze attivate di cui si attende la risposta per intervenire.

3.3 UC9: Logout

Descrizione Classica fase di logout.

3.4 UC3: Accettazione chiamata emergenza

Descrizione Il capo turno ha la possibilità di accettare o rifiutare la chiamata nel caso in cui venga contattato da un operatore 118 per rispondere ad un'emergenza nelle vicinanze.

3.5 UC4 e UC5: Comunicazione con centrale

Descrizione Comunicazione dei dati raccolti durante l'emergenza tramite form adibito per l'inserimento dei dati del/i paziente/i coinvolti nell'emergenza accettata.

3.6 Analisi statica

Anche nel corso di questa iterazione l'analisi statica, effettuata tramite plugin *Qodana*, ha prodotto gli stessi risultati ottenuti nell'iterazione precedente.

3.7 Analisi dinamica

Per quanto riguarda la sezione relativa al volontario è stata testata la seguente API:

```
### Logout dalla applicazione  
GET http://localhost:8080/logout  
Content-Type: application/json
```

Come per l'analisi dinamica relativa all'operatore 118, i test effettuati sul login, con l'ausilio di *JUnit* hanno avuto esito positivo (figura 12).

In particolare, per i test inerenti ai volontari la classe LoginControllerTest viene estesa con i seguenti metodi:

```
public class LoginControllerTest{
    //declarations
    @Test
    public void loginPageLoads() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.get("/login"))
            .andExpect(MockMvcResultMatchers.status().isOk());
    }
    @Test
    public void successfulLoginV() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/login")
            .param("username", "V009")
            .param("password", "stellaalpina"))
            .andExpect(MockMvcResultMatchers.status()
                .is3xxRedirection())
            .andExpect(MockMvcResultMatchers.redirectedUrl
                ("/api/volontari/infoVolontarioSquadra"));
    }

    @Test
    public void failedLoginV() throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.post("/login")
            .param("username", "V009")
            .param("password", "aaaaa"))
            .andExpect(MockMvcResultMatchers.status()
                .is3xxRedirection())
            .andExpect(MockMvcResultMatchers.redirectedUrl
                ("/login?error=true"));
    }
}
```