

Università degli Studi di Verona
Dipartimento di Informatica
Corso di Laurea in Informatica



TERAPIA INTENSIVA

DOCUMENTAZIONE

A.A. 2017/2018

Pegoraro Marco

Righetti Lorenzo

INDICE

- Descrizione del prototipo implementato
- Modello di sviluppo del software
- Use Case principali e relative schede di specifica
- Sequence diagram di dettaglio per i principali Use Case
- Activity Diagram
- Class Diagram del software progettato
- Sequence diagram del software progettato
- Principali pattern adottati e la loro applicazione
- Breve descrizione delle attività di test del prototipo

DESCRIZIONE DEL PROTOTIPO IMPLEMENTATO

Il prototipo implementato si basa sulle specifiche definite nel documento [Esercizio1-2018-TerapiaIntensiva.pdf](#). In particolare si focalizza sullo sviluppo dell'accesso tramite login, sulla visualizzazione dei parametri vitali registrati a diverse frequenze dal sistema di monitoraggio, sulla gestione degli allarmi, sull'inserimento dei pazienti, sulla chiusura di ricoveri e la gestione di somministrazioni.

Appena viene eseguito il software viene visualizzata la finestra di login tramite la quale è possibile accedere al sistema come infermieri, primari o medici. In ogni istante è possibile accedere alla visualizzazione in tempo reale dei parametri vitali registrati attraverso il menu a tendina in alto a sinistra.

Dopo l'accesso al sistema viene visualizzata una finestra con i pulsanti necessari a svolgere diverse funzioni in base alla tipologia di utente. Il prototipo si focalizza principalmente sulla figura dell'infermiere per il quale sono stati implementati per intero la registrazione di nuovi pazienti e l'inserimento di somministrazioni. Ogni volta che un nuovo paziente viene registrato il sistema crea la cartella del ricovero corrente identificata dalla data di registrazione e inizializza il monitor di registrazione dei parametri vitali. I diversi segnali vengono letti dal sistema di monitoraggio a diverse frequenze stabilite nel documento delle specifiche. Invece per gli allarmi il monitor attende che il sistema di monitoraggio notifichi la presenza di un nuovo allarme.

Il salvataggio dei dati è stato gestito tramite un sistema di file e cartelle. Per ogni paziente ricoverato in terapia intensiva esiste una cartella con il suo codice sanitario contenente un file con i dati anagrafici e le cartelle dei vari ricoveri identificate dalla data di inizio del ricovero. In questo modo a partire dal codice sanitario è possibile recuperare le cartelle pregresse. Ogni cartella di ricovero contiene i file dei vari dati registrati: diagnosi di ingresso, somministrazioni, prescrizioni e i valori di pressione, temperatura e frequenza cardiaca. Inoltre esistono un file contenente i dati per i login e uno con i percorsi e codici sanitari dei pazienti al momento ricoverati.

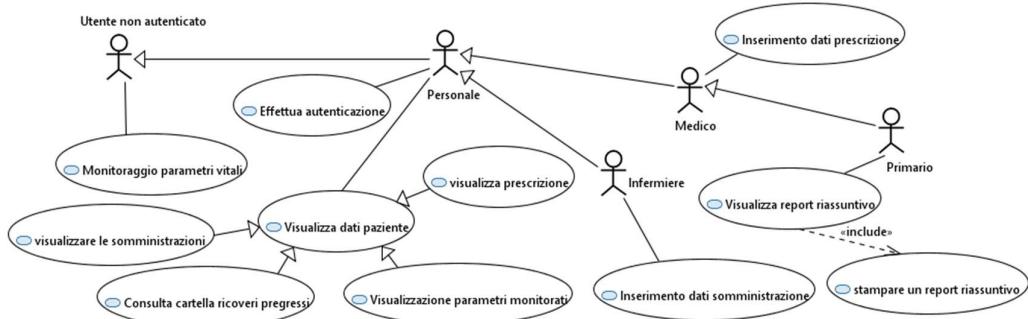
Nel prototipo tutta la parte grafica è stata implementata tramite le librerie *Swing* di java.

MODELLO DI SVILUPPO DEL SOFTWARE

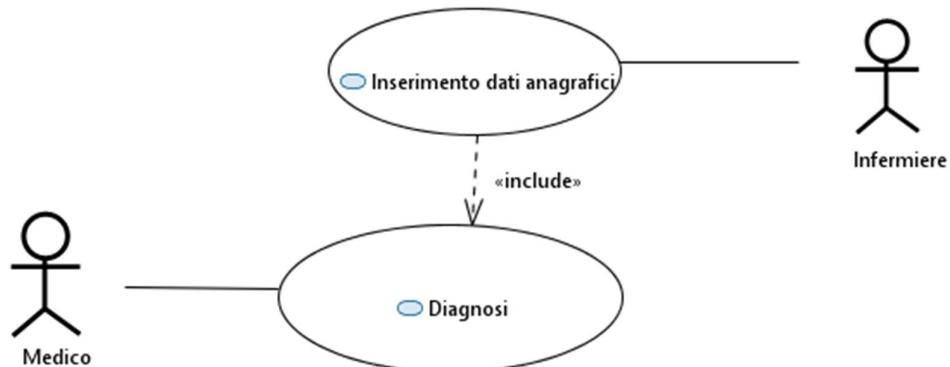
Il processo di sviluppo del software ha seguito un modello agile molto simile allo *scrum*. Dopo un primo incontro per definire in gruppo i principali requisiti da soddisfare, il lavoro è stato suddiviso in sprint intervallati da incontri di aggiornamento. Ogni membro del gruppo aveva dei compiti da portare a termine in ogni sprint in modo da poter lavorare in parallelo. All'incirca una volta a settimana il gruppo si è riunito per confrontarsi sul lavoro svolto nel corso dell'ultimo sprint, presentare eventuali problemi riscontrati e stabilire le attività da portare a termine entro il prossimo sprint. Inoltre per condividere il materiale ed il codice è stata usata la piattaforma di *GitHub*.

USE CASE PRINCIPALI E RELATIVE SCHEDE DI SPECIFICA

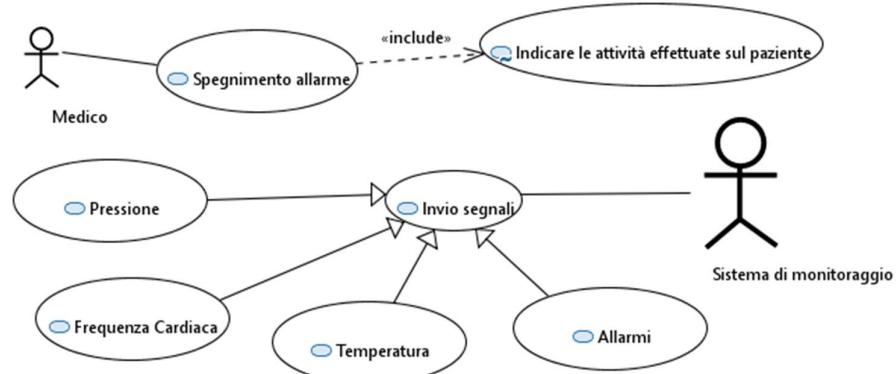
Inserimenti e visualizzazioni



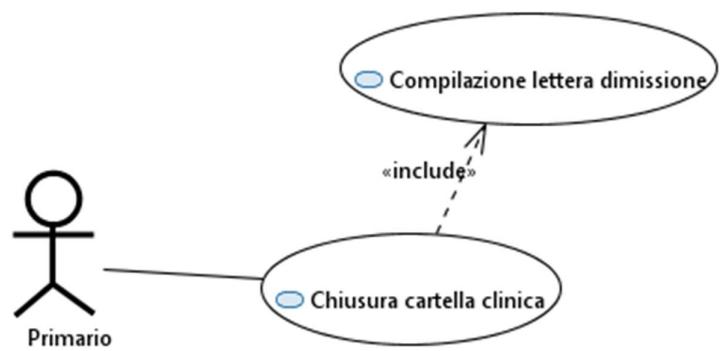
Registrazione paziente



Sistema di monitoraggio

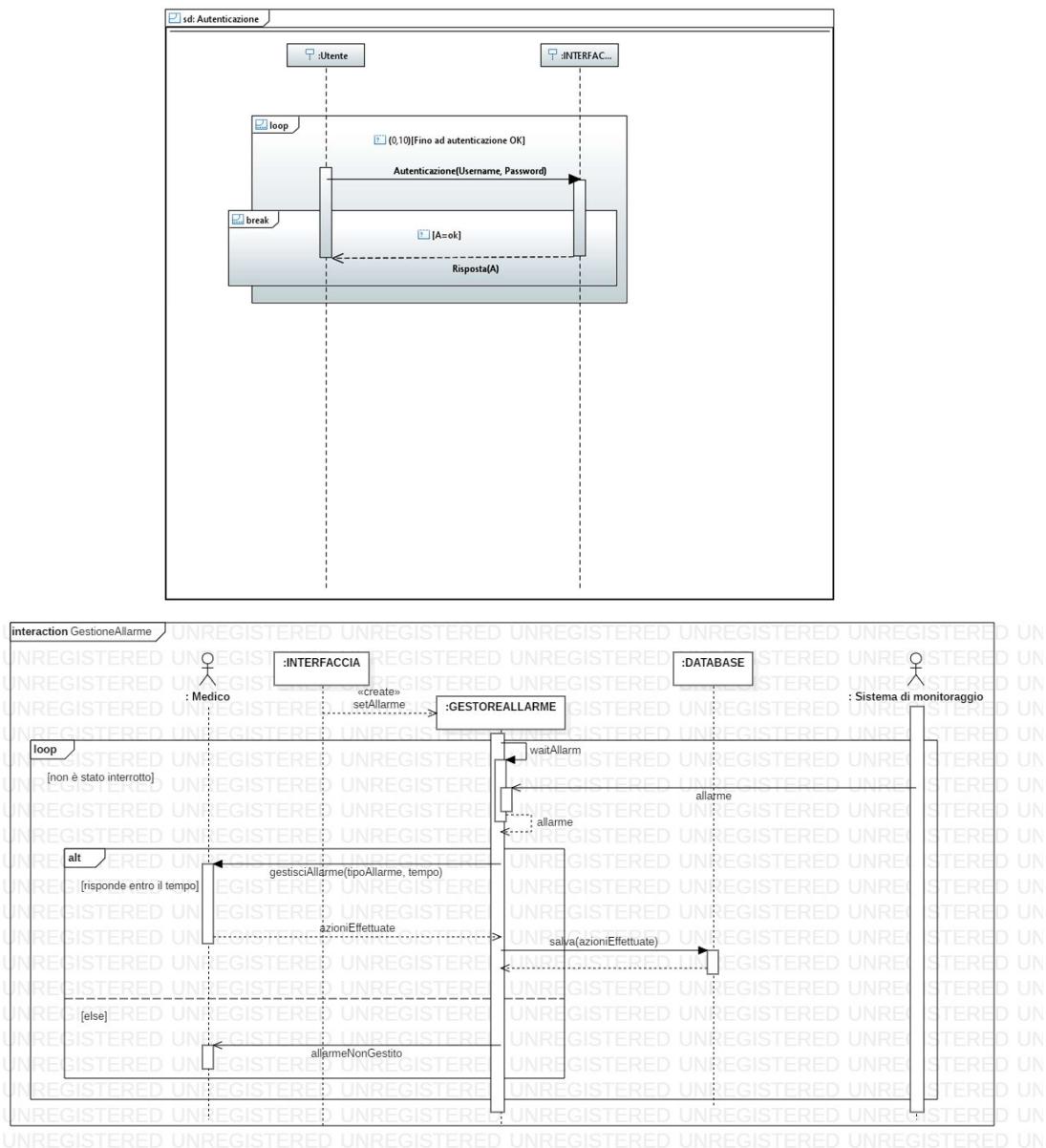


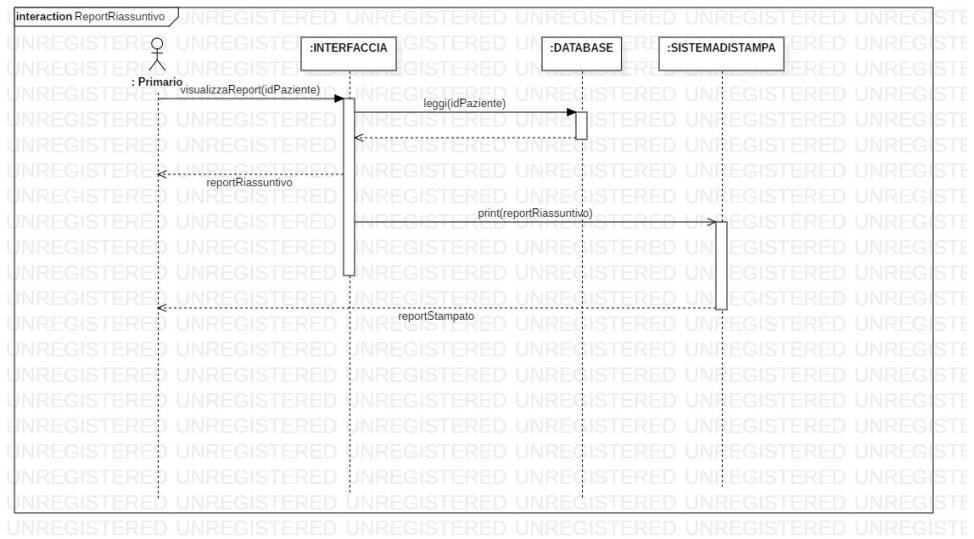
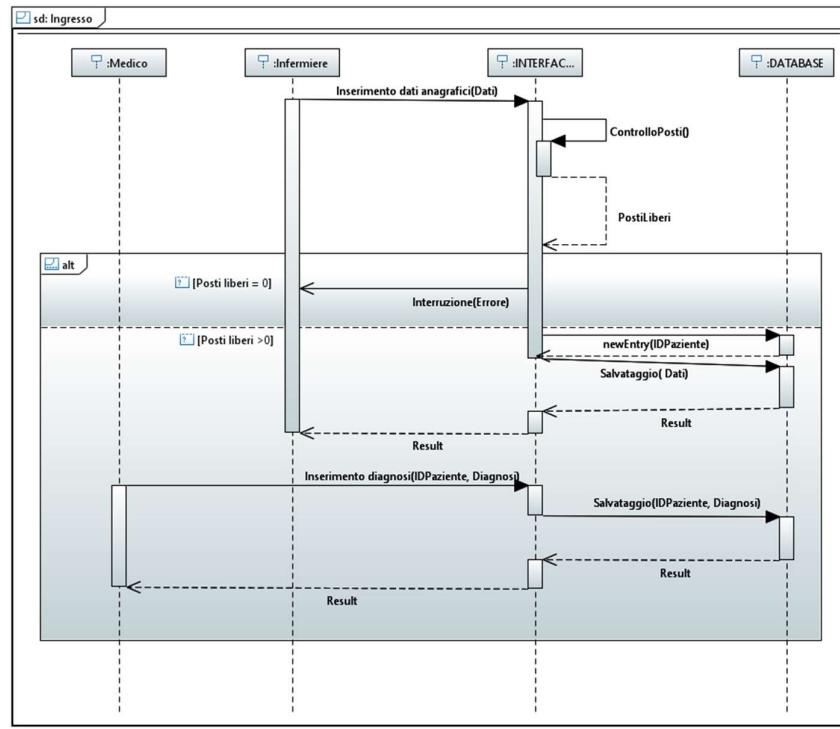
Fine ricovero

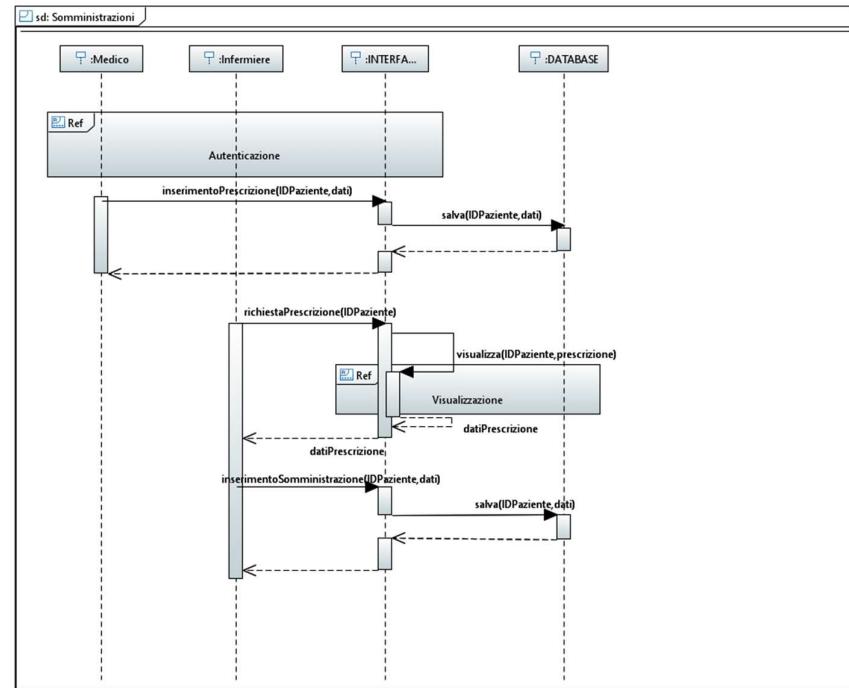
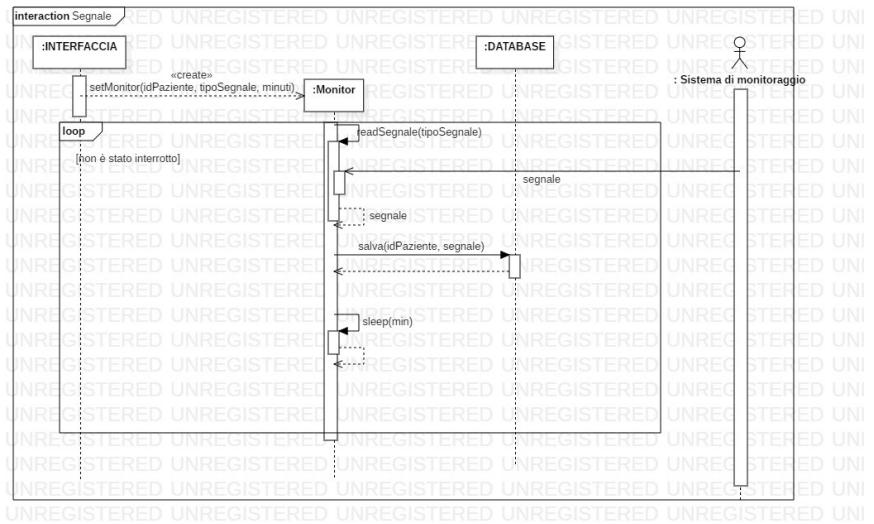


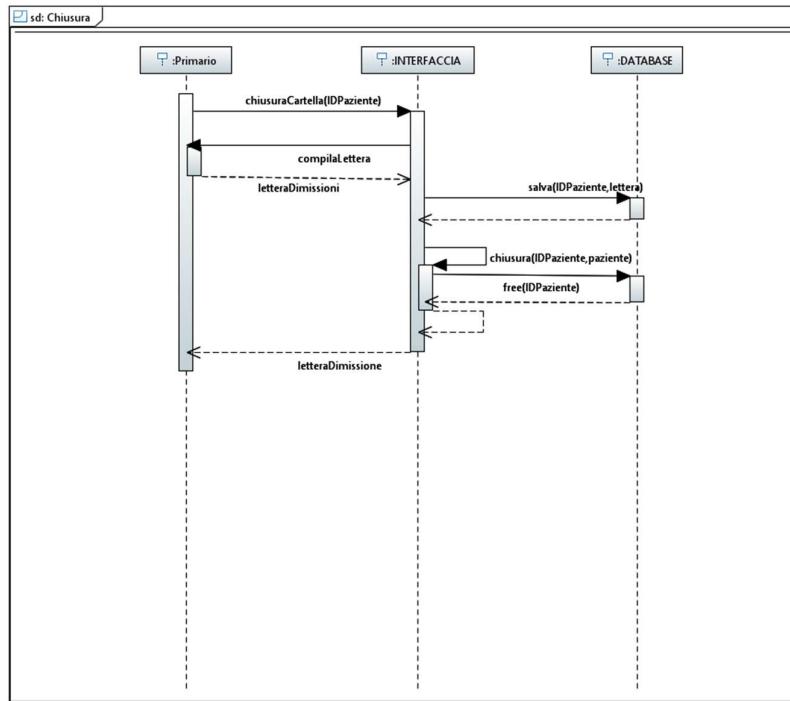
Schede di specifica: [UseCase\Casi d'uso.docx](#)

SEQUENCE DIAGRAM DI DETTAGLIO PER I PRINCIPALI USE CASE



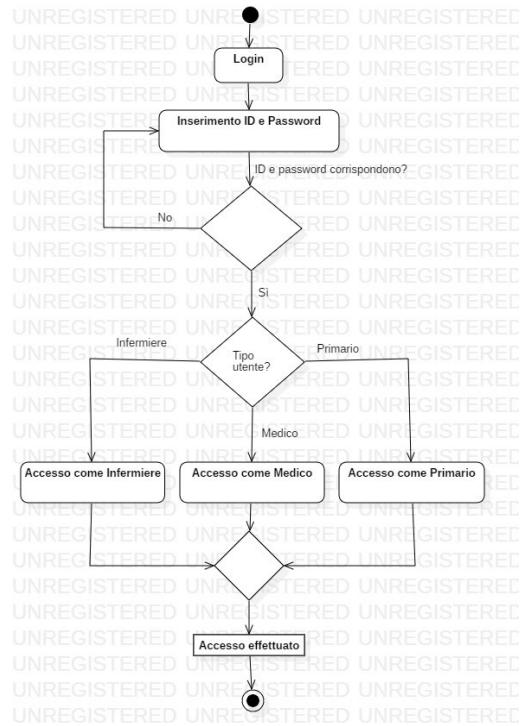




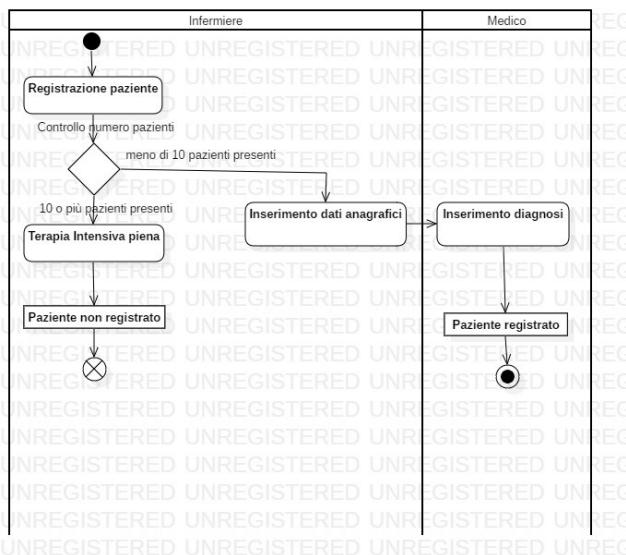


ACTIVITY DIAGRAM

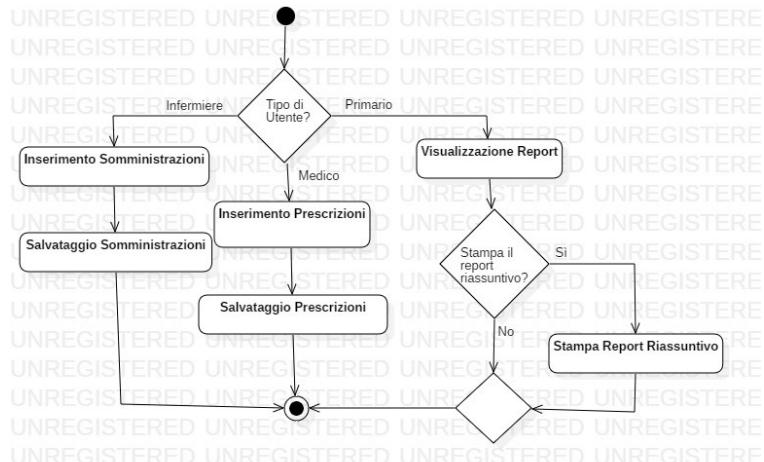
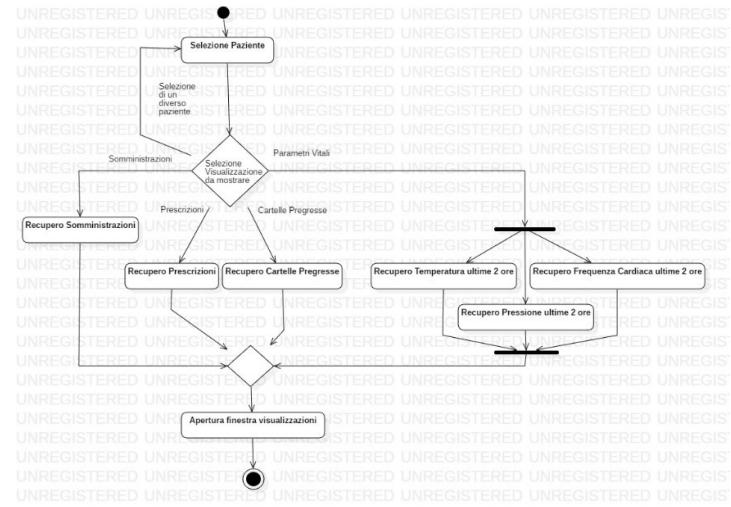
Login



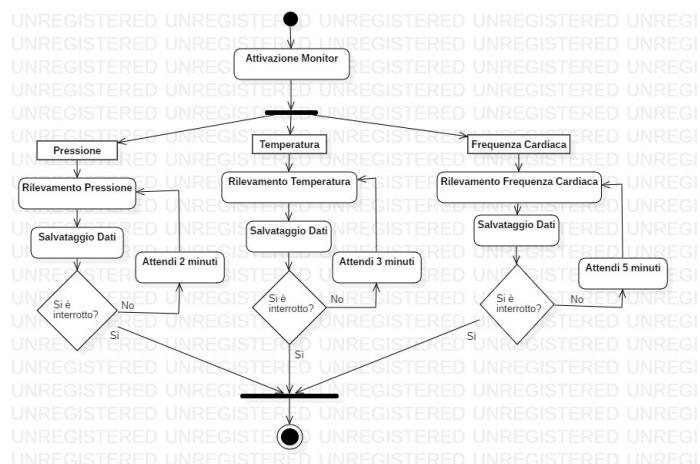
Registrazione paziente



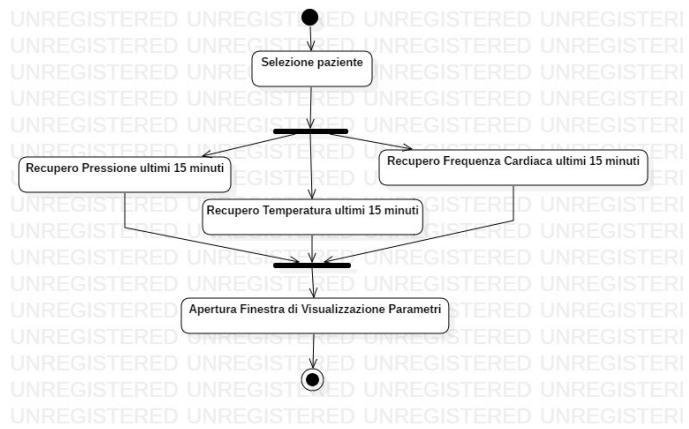
Visualizzazione



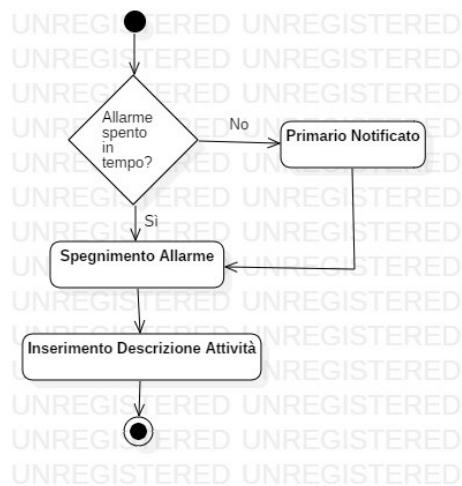
Rilevamento parametri



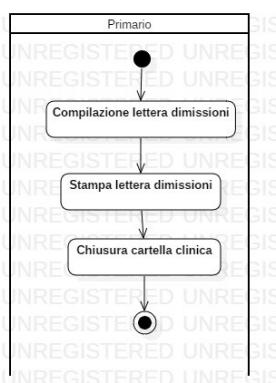
Monitoraggio parametri vitali



Gestione allarme

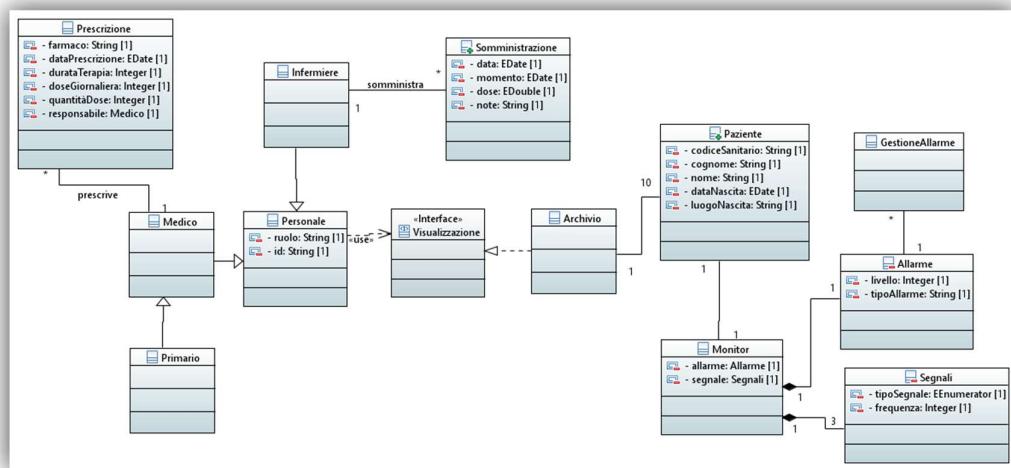


Chiusura Cartella

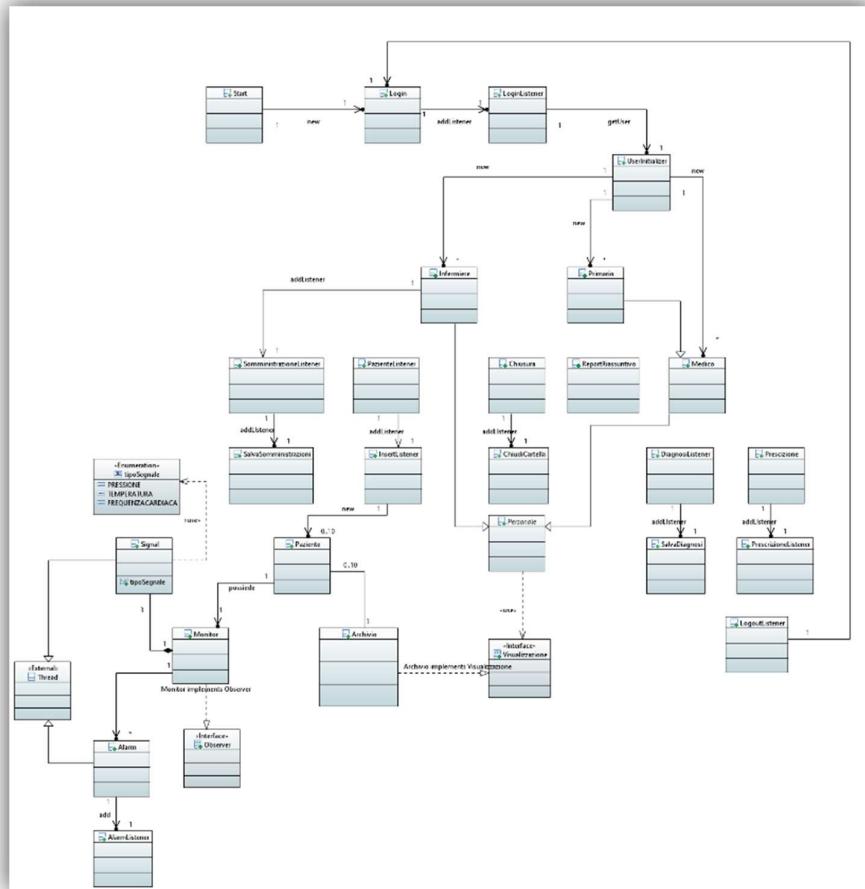


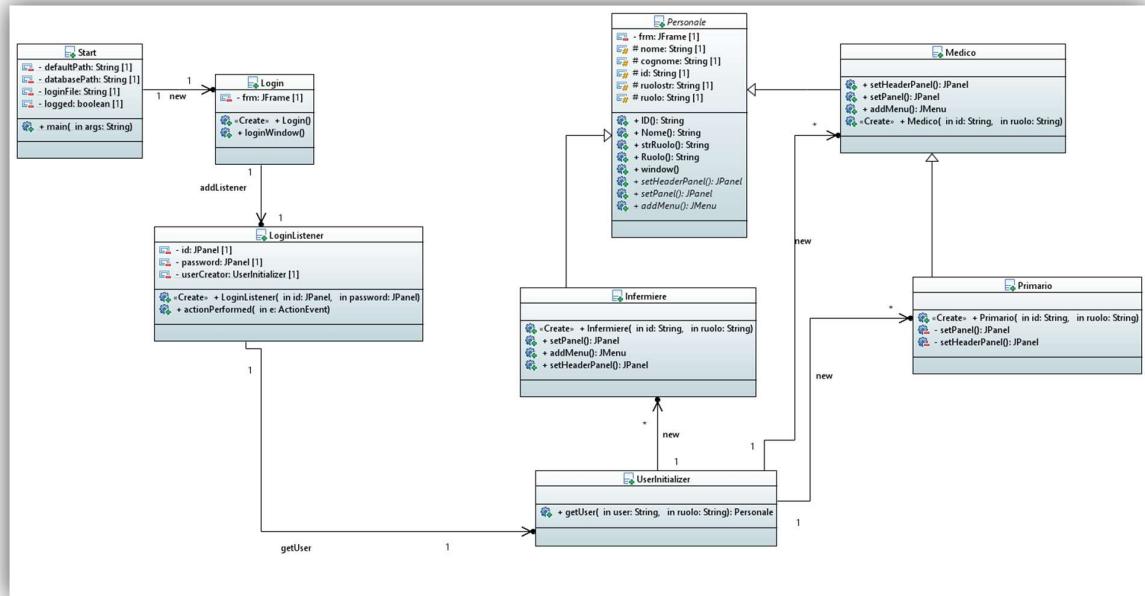
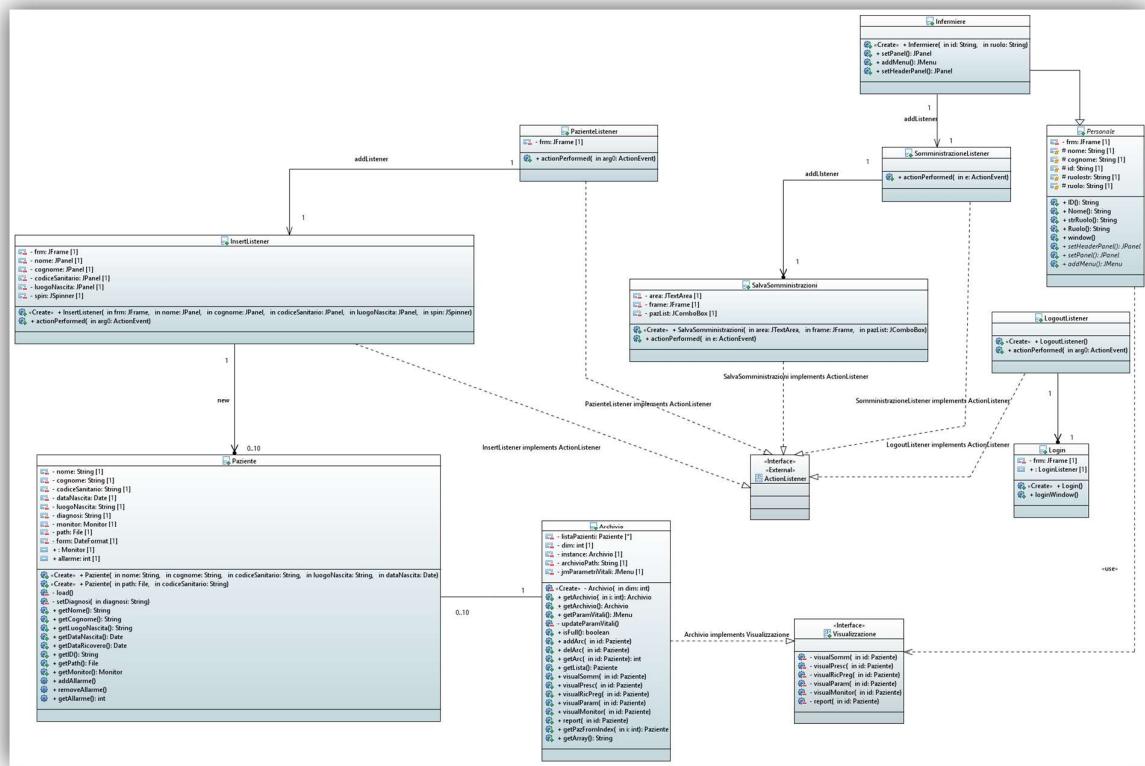
CLASS DIAGRAM DEL SOFTWARE PROGETTATO

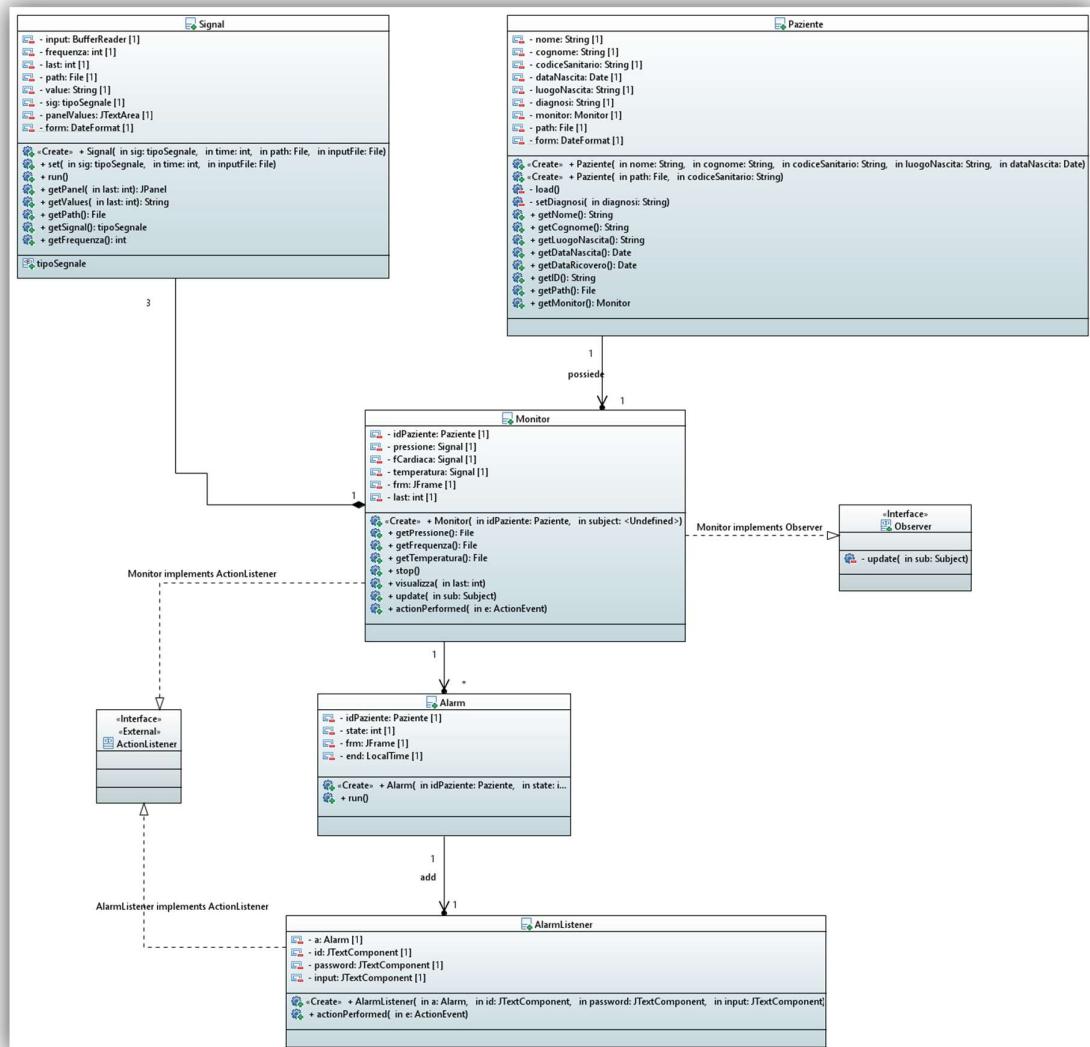
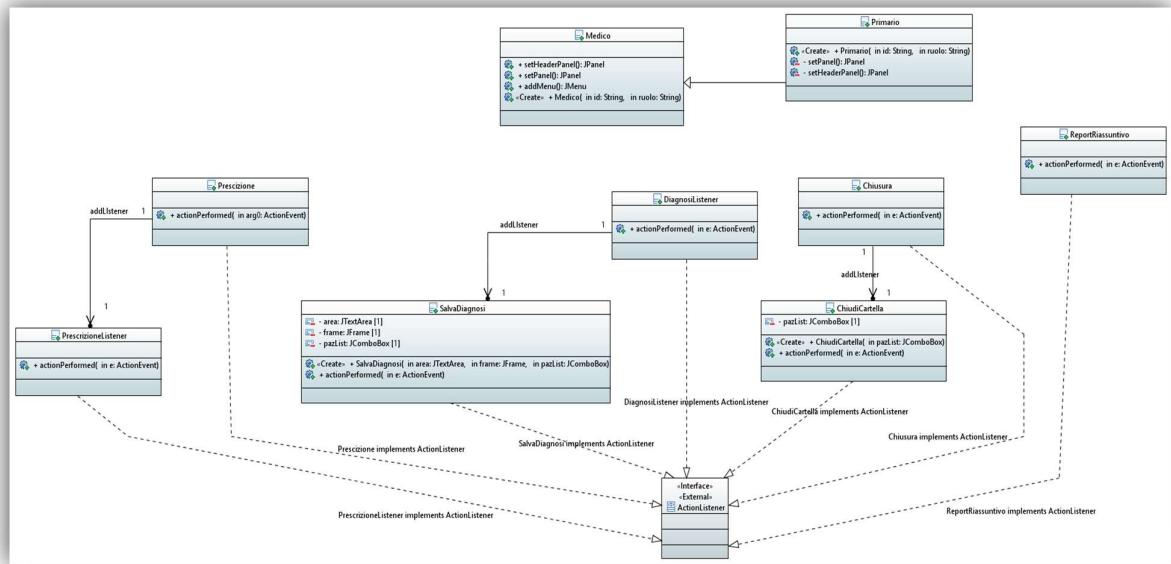
Analisi



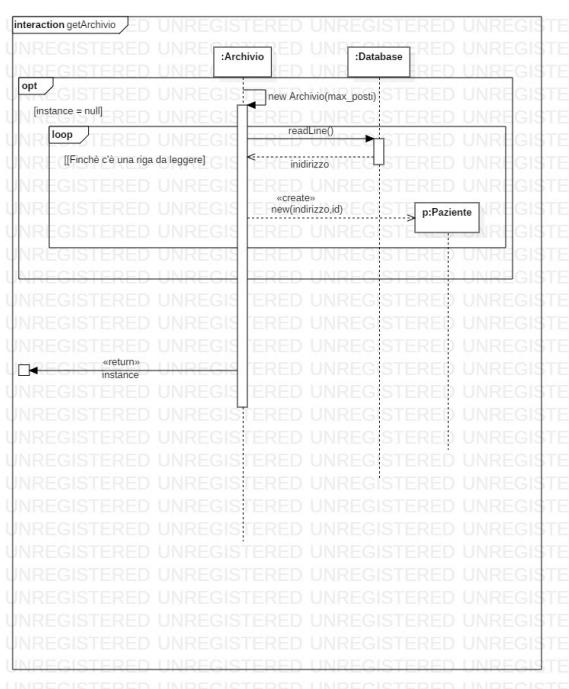
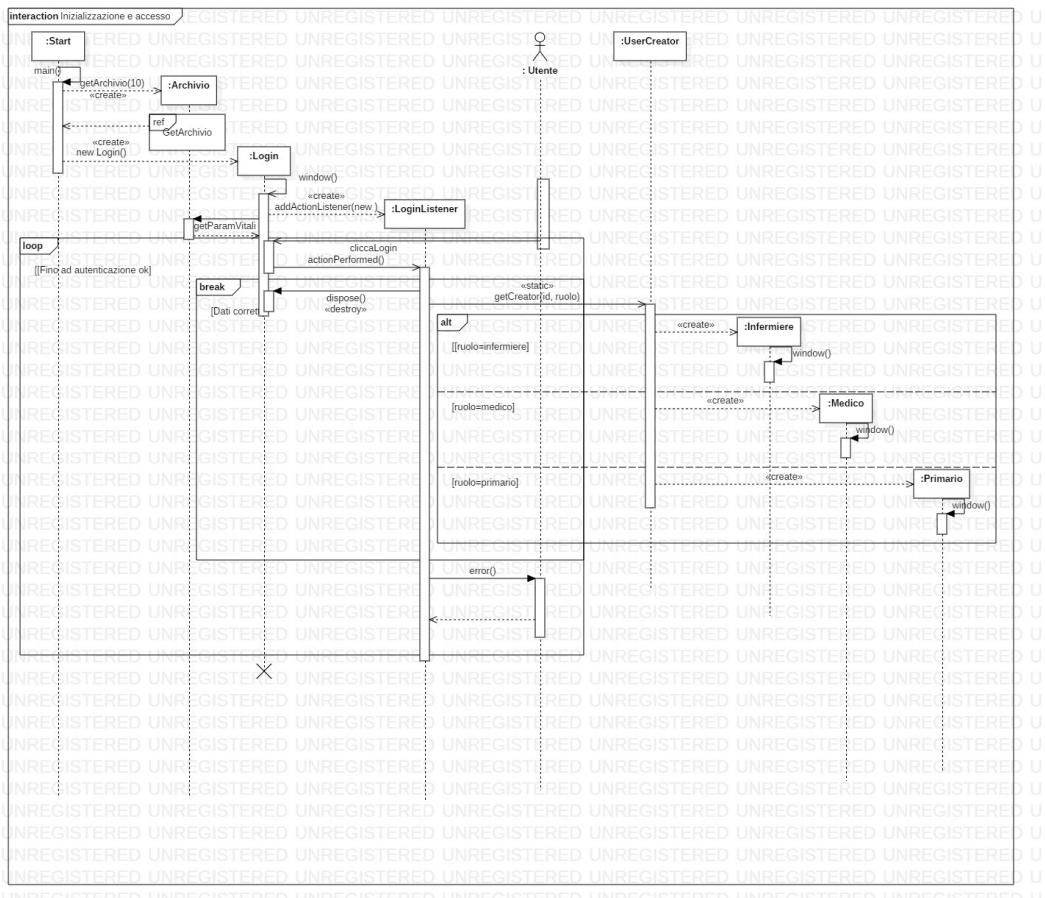
Progettazione

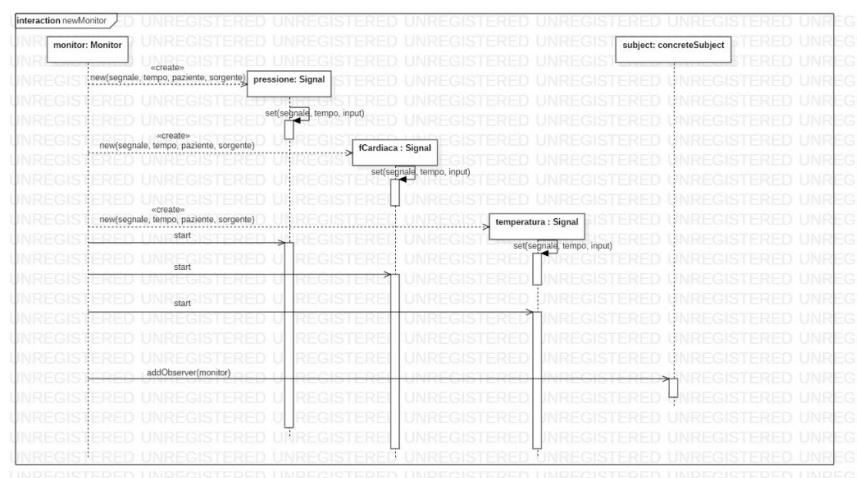
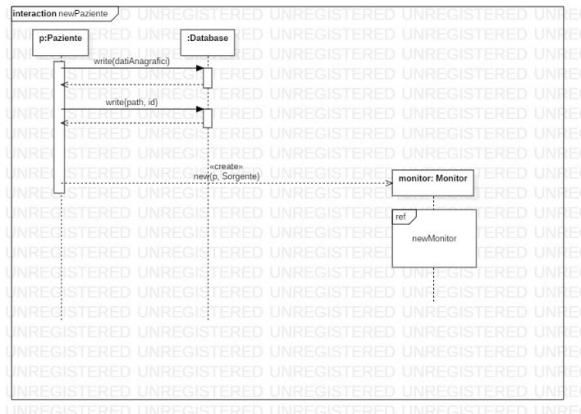
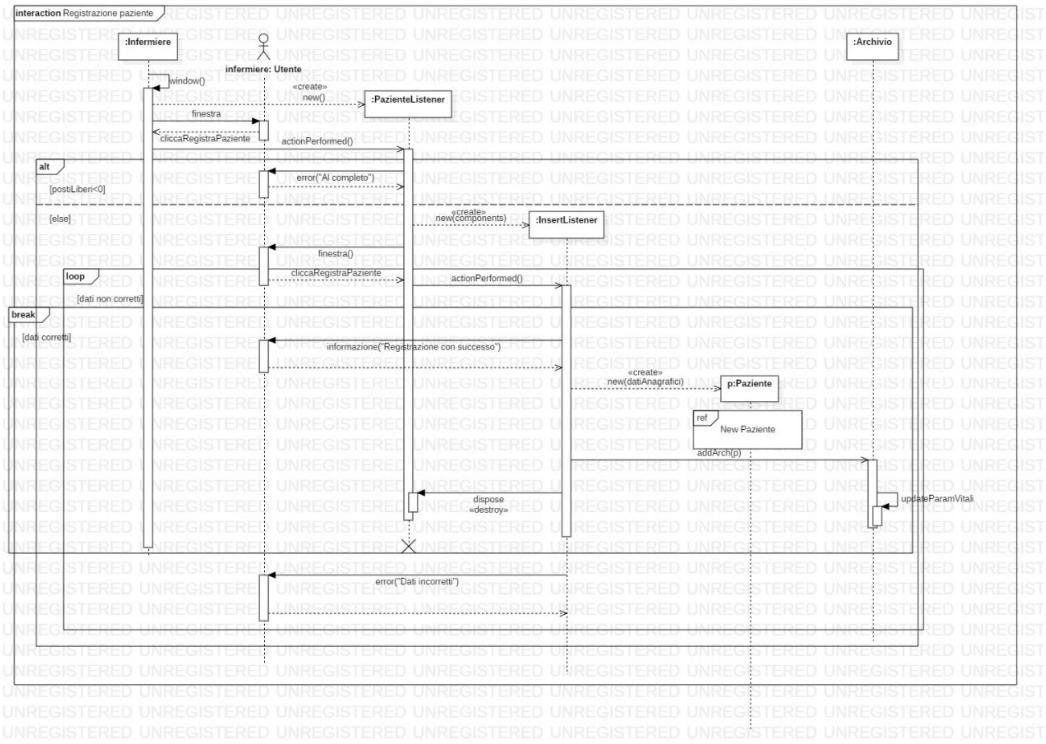


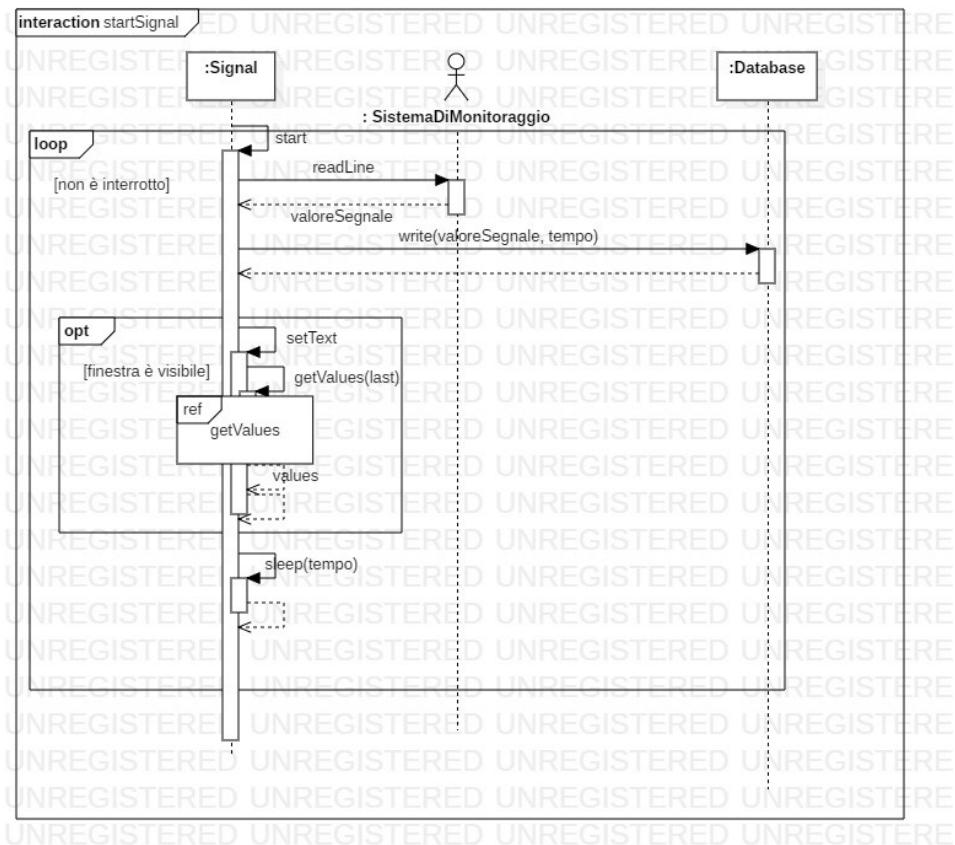
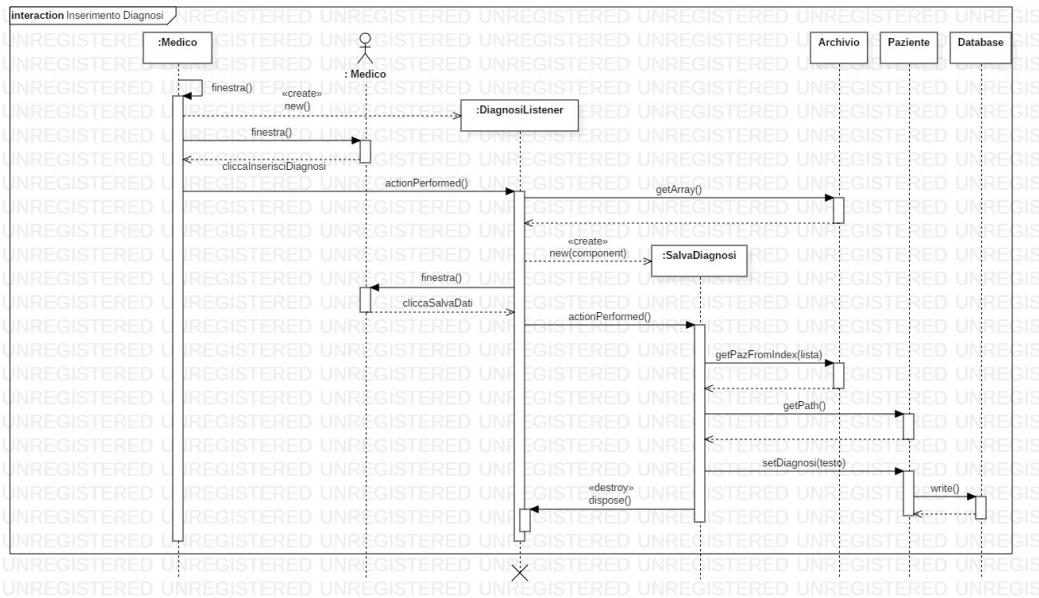


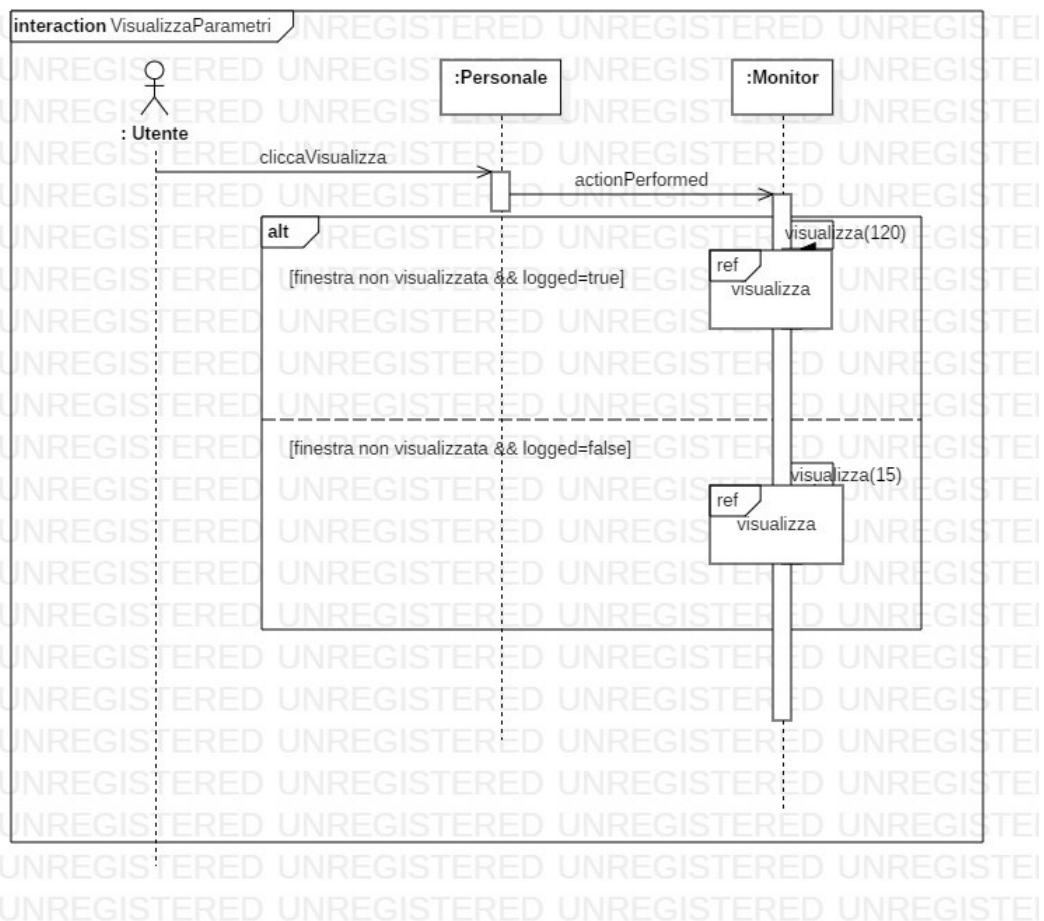
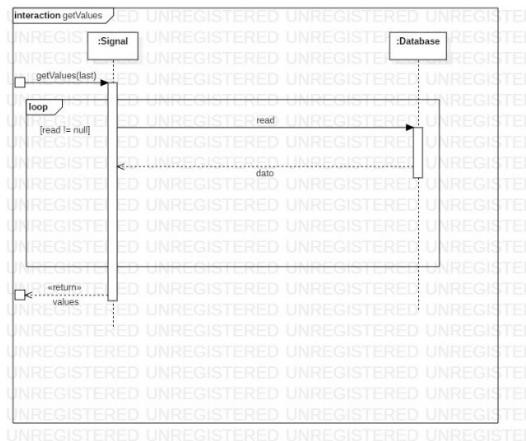


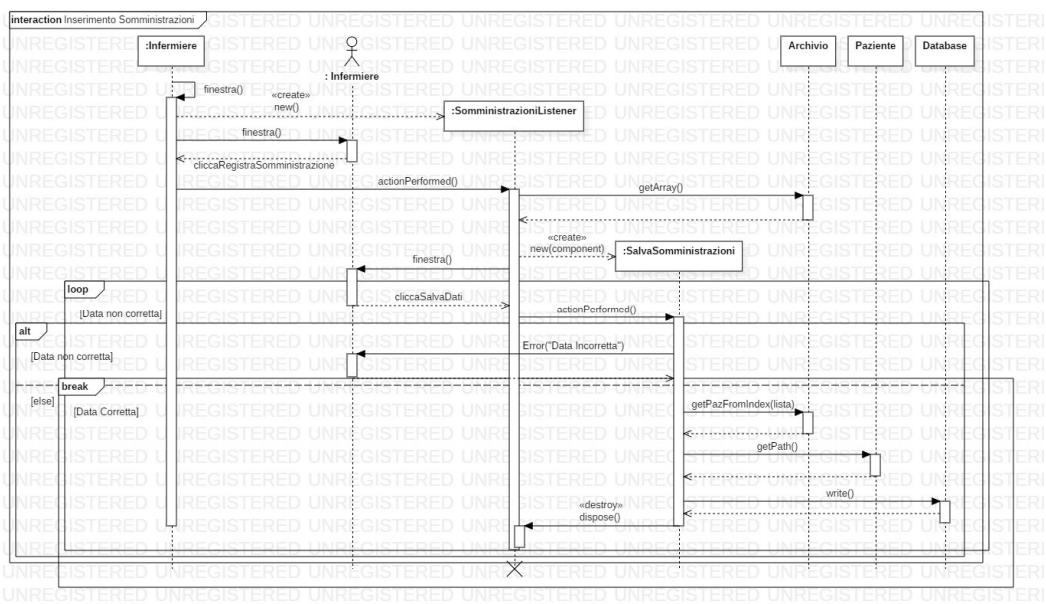
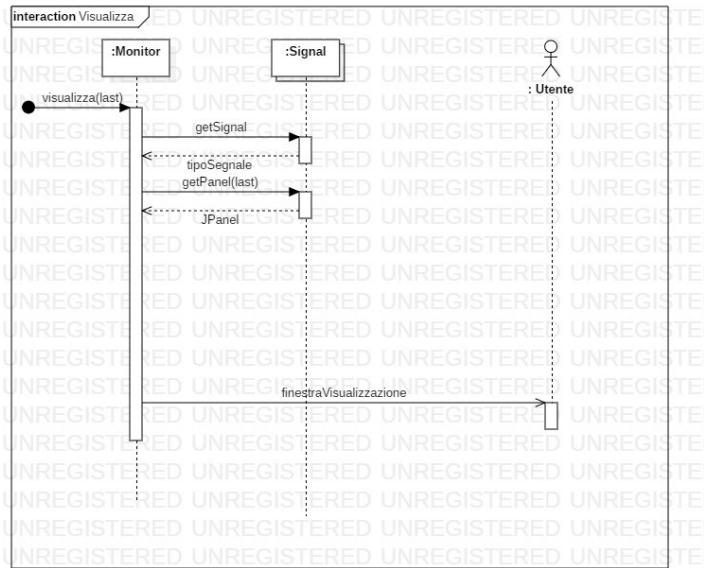
SEQUENCE DIAGRAM DEL SOFTWARE PROGETTATO

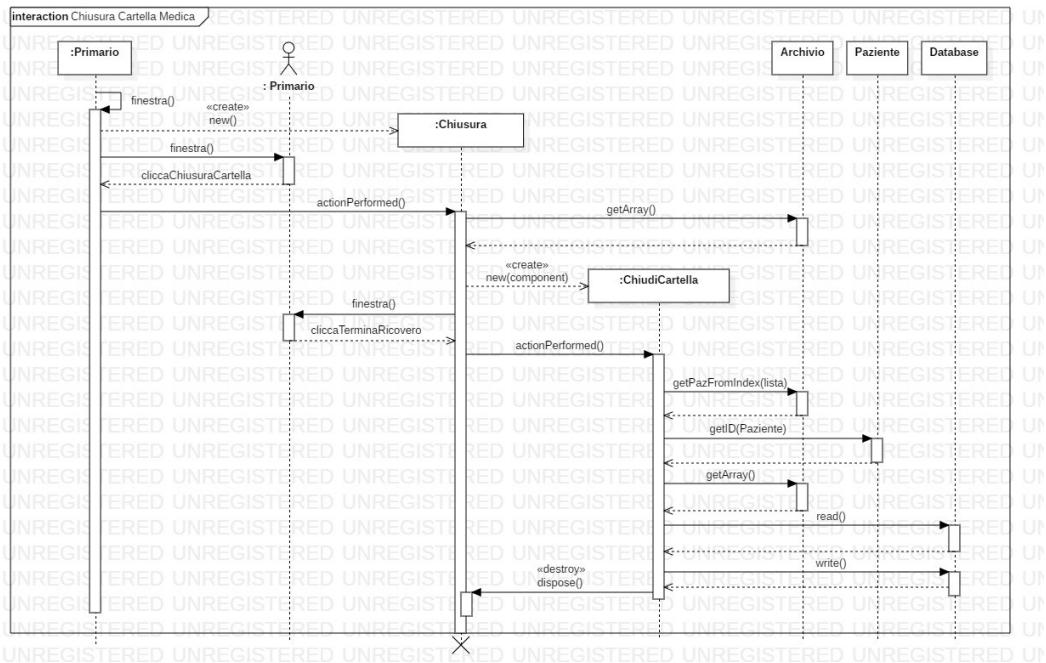
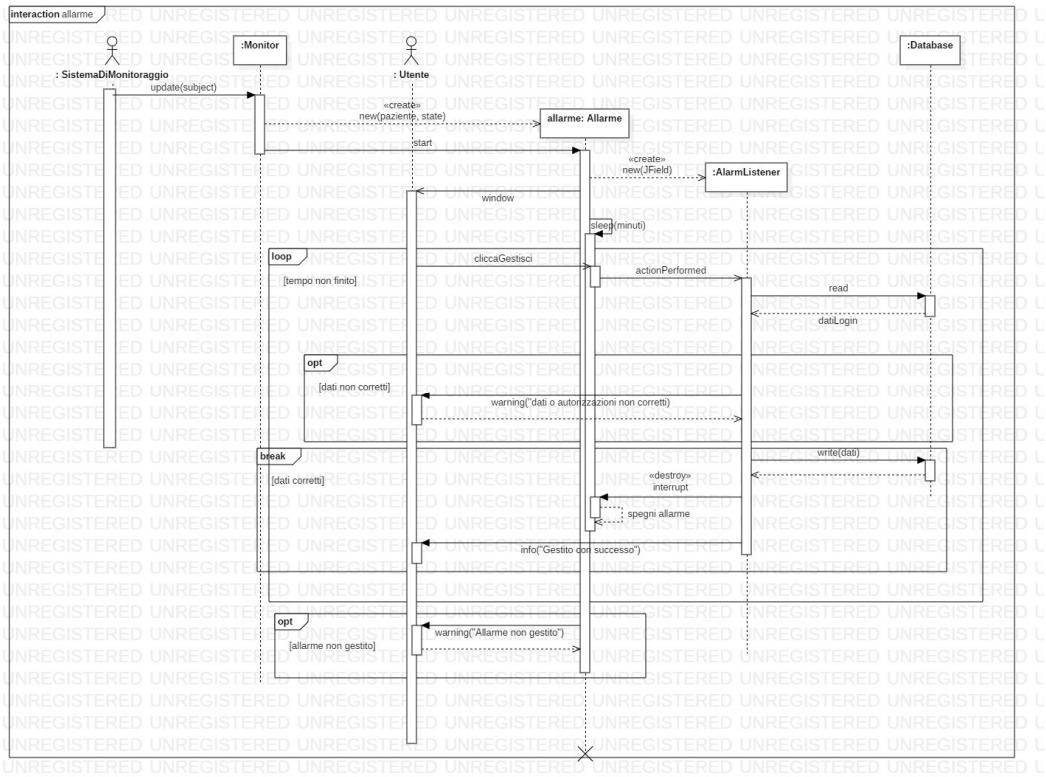












PRINCIPALI PATTERN ADOTTATI E LA LORO APPLICAZIONE

Durante la progettazione sono stati usati alcuni dei design pattern mostrati a lezione.

Tra i principali pattern è stato usato il pattern Singleton per la classe *Archivio* in modo da renderla unica e fornire un unico punto di accesso per tutte le classi. Grazie a questo pattern la gestione dei pazienti attualmente ricoverati è centralizzata in *Archivio*. Inoltre la classe è stata usata come *repository* contenente i pazienti attualmente ricoverati e da cui si può accedere alle loro principali informazioni.

Per la creazione della sessione in base al ruolo dell'utente che accede al sistema è stato usato il pattern creazionale Factory. Ogni volta che un utente accede attraverso il login, il metodo *getUser* in base al ruolo fornito crea una finestra diversa invocando il costruttore adatto. Grazie a questa tipologia di pattern, tutto questo è fatto indipendentemente dal sistema che lo utilizza e la manutenzione del codice risulta molto più facile.

Inoltre per rendere simile la struttura delle finestre create all'inizio della sessione di un utente è stato usato il pattern Template. Il metodo *window*, definito come final nella classe astratta *Personale*, contiene i metodi astratti *setHeaderPanel*, *setPanel* e *addMenu* i quali permettono alle sottoclassi di aggiungere componenti alla finestra della sessione in base al ruolo dell'utente.

Per l'implementazione dei pulsanti è stato adottato il pattern Observer così da attivare una diversa funzionalità ogni volta che l'utente clicca su un pulsante senza creare una forte dipendenza. In particolare si è sfruttato il metodo *addActionListener* della classe *JButton* per aggiungere nuove istanze di classi che implementano *ActionListener* diversamente in base alla funzionalità che dovevano fornire.

Per gestire la registrazione dei parametri vitali e degli allarmi è stato usato il pattern Facade. La classe *Monitor* funge da facciata per il sottosistema formato da 3 classi *Signal*, una per ogni segnale, e dall'observer implementato dentro al monitor stesso. Ogni volta che un'istanza di *Monitor* viene creata, il costruttore istanzia i 3 segnali, li fa partire come thread e si registra alla sorgente di allarmi del sistema di monitoraggio esterno utilizzando il pattern *Observer*. Poi ogni volta che si vuole visualizzare i parametri vitali in tempo reale, il *Monitor* compone la finestra ricavando i componenti con i valori registrati dalle classi *Signal* istanziate e poi restituisce all'utente la finestra completa senza che sappia come viene ricavata. Invece per la gestione degli allarmi è stato adottato il pattern Observer usando il monitor come Observer esterno e la sorgente di allarmi del Sistema di monitoraggio come Subject. In questo modo ogni volta che un allarme è rilevato dal sistema esterno, viene aggiornato il monitor il quale genera una thread che gestisce in modo indipendente l'allarme senza bloccare il funzionamento del resto del sistema interno.

BREVE DESCRIZIONE DELLE ATTIVITÀ DI TEST DEL PROTOTIPO

Le attività di test svolte si possono suddividere in tre fasi: test dei singoli componenti, test del sistema, test su utenti.

Il test dei singoli componenti è stato svolto in autonomo da ogni componente del gruppo ed è relativo alla parte di codice che doveva essere implementata in ogni sprint. In particolare riguardava l'aspetto ed il funzionamento di singole finestre e relativi pulsanti.

Una volta testati i singoli componenti si univano al sistema testandone il suo funzionamento. In questa fase sono stati usati degli utenti inventati per controllare l'accesso al sistema, valori di segnali casuali per il controllo della registrazione dal sistema di monitoraggio e un sorgente di allarmi *ConcreteSubject* il quale genera in modo casuale nell'intervallo di minuti un allarme per ogni paziente prelevando un codice casualmente da un file.

Nell'ultima fase di test, il prototipo completato è stato fatto provare a gente esterna simulando un *beta-test*. Grazie a questa fase sono stati ricevuti feedback sull'aspetto delle finestre, sull'utilizzo delle interfacce e sul funzionamento di alcuni casi particolari. In particolare è stata riscontrata la possibilità di registrare più volte uno stesso paziente già ricoverato in quel momento. Il problema è stato subito risolto aggiungendo un controllo tra il paziente che si vuole registrare e quelli attualmente ricoverati.