Universita' degli Studi di Verona

Ingegneria del Software

Drug Supervision

Documentazione al prototipo

Andrea Soglieri, Mattia Zorzan

29 gennaio 2019

Indice

1 Introduzione

L'obiettivo dell'elaborato e' lo sviluppo di un prototipo software per la gestione delle terapie farmacologiche da parte dei medici di base e segnalare eventuali problemi relativi ai farmaci.

In questa relazione ci proponiamo di raccolgiere la documentazione sviluppata e di fornire spiegazioni dettagliate sulle scelte progettuali ed implementative.

2 Requisiti

Di seguito viene riportata la consegna dell'elaborato.

> *Si vuole progettare un sistema software per gestire le segnalazioni di reazioni avverse (ad esempio, asma, dermatiti, insufficienza renale, ...) da farmaci.*
> *Ogni segnalazione e' caratterizzata da un codice univoco, dall'indicazione del paziente a cui fa riferimento, dall'indicazione della reazione avversa, dalla data della reazione avversa, dalla data di segnalazione, e dalle terapie farmacologiche in atto al momento della reazione avversa.*
> *Per ogni paziente sono memorizzati: un codice univoco, l'anno di nascita, la provincia di residenza e la professione. Per ogni paziente e' possibile memorizzare gli eventuali fattori di rischio presenti (paziente fumatore, iperteso, sovrappeso, ...), anche piu' d'uno. Ogni fattore di rischio e' caratterizzato da un nome univoco, unadescrizione e il livello di rischio associato.*
> *Ogni terapia farmacologica e' caratterizzata da: paziente a cui si riferisce, segnalazioni a cui e' legata, farmaco somministrato, dose, frequenza giornaliera, data di inizio e data di fine della terapia stessa. Per ogni reazione avversa sono memorizzati un nome univoco, un livello di gravita' (da 1 a 5) e una descrizione generale, espressa in linguaggio naturale. Una reazione avversa puo' essere legata a molte segnalazioni. Per ogni paziente sono memorizzati per ogni anno il numero di reazioni avverse segnalate ed il numero di terapie farmacologiche relative.*
> *Il sistema deve supportare i medici che effettuano la segnalazione. Dopo opportuna autenticazione, il medico viene introdotto ad una interfaccia che permette l'inserimento dei dati delle reazioni avverse e dei pazienti. Il codice univoco dei pazienti e' gestito dal sistema, che tiene traccia dei pazienti indicati da ogni medico. Ogni medico vede solo i codici identificativi dei pazienti, dei quali ha gia' segnalato qualche reazione avversa. Ad ogni fine settimana o quando il numero di segnalazioni raggiunge la soglia di 50, il sistema manda un avviso ad uno dei farmacologi responsabili della gestione delle segnalazioni di reazioni avverse. Il farmacologo, dopo autenticazione, accede alle segnalazioni e puo' effettuare alcune analisi di base (quante segnalazioni per farmaco, quante segnalazioni gravi in settimana, ...). Il sistema, inoltre, avvisa il farmacologo quando un farmaco ha accumulato nell'anno oltre 10 segnalazioni di gravita' superiore a 3.*
> *In base alle segnalazioni e agli avvisi del sistema, il farmacologo puo' proporre di ritirare il farmaco dal commercio immediatamente, di attivare una fase di controllo del farmaco, di mettere il farmaco fra quelli che richiedono un monitoraggio piu' atten-*

to. Tali proposte vengono registrate dal sistema, che tiene traccia di tutte le proposte relative ai farmaci segnalati.

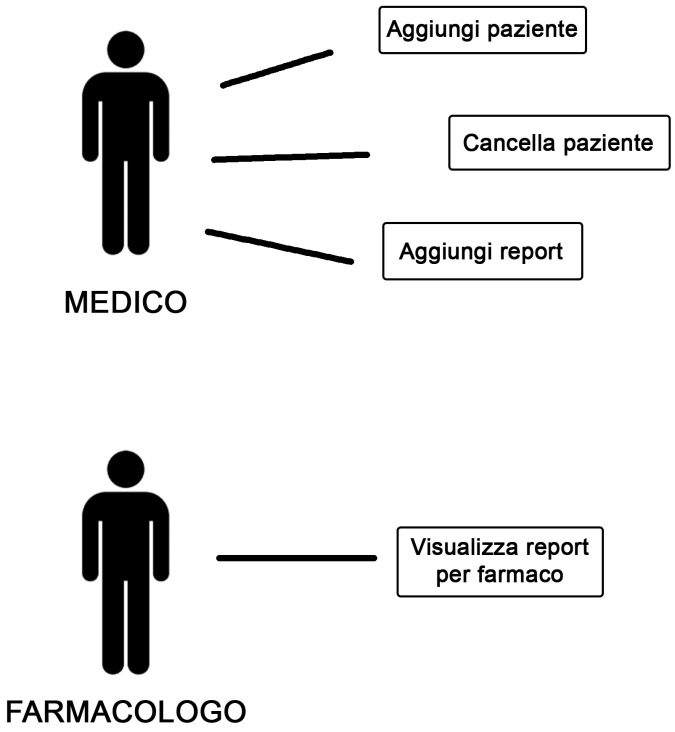
3 UML

In questa sezione sono presentati i diagrammi richiesti per la documentazione del prototipo.

3.1 Use Cases

Gli use case sono stati suddivisi in due macroaree definite dalla tipologia di utente che effettua il login.
Esse possono essere suddivise in base alla GUI:

1. Vista del medico di base.
2. Vista del farmacologo.



Di seguito alcune schede che descrittive degli use case sopracitati.

ID	UC1: Inserimento paziente.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico apre il menù "File", sposta il cursore sopra la voce "New" scegliendo la voce "Patient". Viene visualizzato il dialog corrispondente. 2. Inserisce i dati relativi alla persone e gli eventuali Risk Factor. 3. Se il medico preme " New" in corrispondenza della riga "Risk Factor": 3.1. Inserisce i dati relativi al nuovo Risk Factor. 3.2. Se il medico preme "Add": 3.2.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 3.2.1.1. Il Risk Factor viene aggiunto al Database. 3.2.1.2. Il dialog viene chiuso. 4. Se il medico preme "Add": 4.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 4.1.1. Il paziente viene aggiunto al Database. 4.1.2. Il dialog viene chiuso.
Postcondizioni	Nuovo paziente aggiunto correttamente.

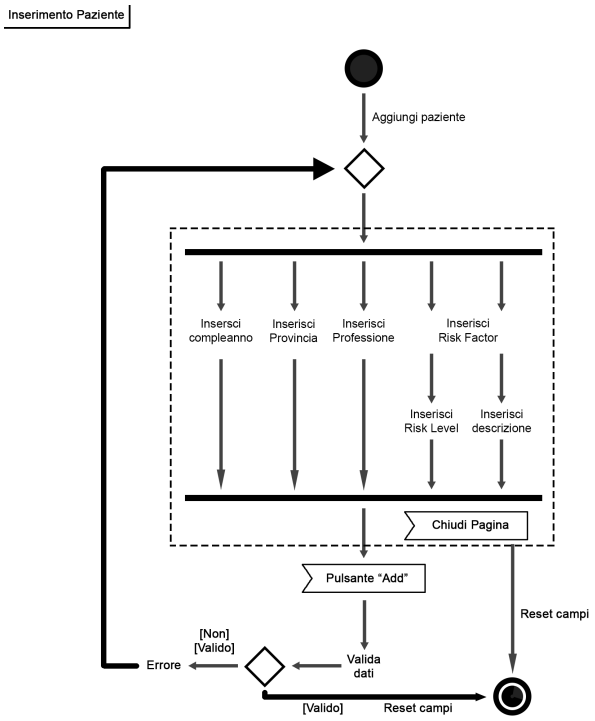
ID	UC2: Inserimento report.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico apre il menù "File", sposta il cursore sopra la voce "New" scegliendo la voce "Report". Viene visualizzato il dialog corrispondente. 2. Inserisce i dati relativi al report, alla reaction e alla terapia farmacologica. 3. Se il medico preme "New" in corrispondenza della riga Reaction: 3.1. Inserisce i dati relativi alla nuova reaction. 3.2. Se il medico preme "Add": 3.2.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 3.2.1.1. La reaction viene aggiunta al Database. 3.2.1.2. Il dialog viene chiuso. 4. Se il medico preme "New" in corrispondenza della riga Therapy: 4.1. Inserisce i dati relativi alla nuova terapia farmacologica. 4.2. Se il medico premo "New" in corrispondenza della riga Drug: 4.2.1. Inserisce il nome del nuovo farmaco. 4.2.2. Se il medico preme "Add": 4.2.2.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 4.2.2.2. Il farmaco viene aggiunto al Database. 4.2.2.3. Il dialog viene chiuso. 4.3. Se il medico preme "Add": 4.3.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 4.3.1.1. La terapia farmacologica viene aggiunta al Database. 4.3.1.2. Il dialog viene chiuso. 5. Se il medico preme "Add": 5.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 5.1.1. Il report viene aggiunto al Database. 5.1.2. Il dialog viene chiuso.
Postcondizioni	Nuovo report aggiunto correttamente

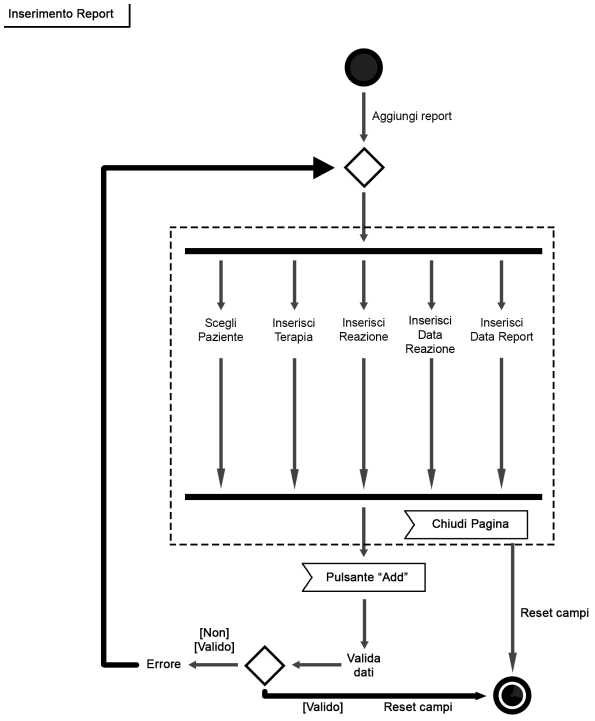
ID	UC3: Cancella paziente.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico apre il menù "File" e sceglie la voce "Delete" 2. Viene mostrato un dialog che chiede di confermare la scelta. 3. Se il medico preme "Ok": 3.1. Il paziente viene eliminato dal Database. 3.2. Il dialog viene chiuso.
Postcondizioni	Paziente eliminato correttamente.

ID	UC4: Visualizzare report relativi ad un certo farmaco.
Attori	Farmacologo
Precondizioni	Il farmacologo deve aver effettuato il login
Sequenza	1. Il farmacologo può scegliere un determinato farmaco attraverso la ChoiceBox. 2. Vengono mostrati i report relativi a quel farmaco nella TableView 3. Se il farmacologo sceglie "Remove": 3.1. Viene settato a "true" l'opzione di rimozione nel Database. 4. Se il farmacologo sceglie "Inspect": 4.1. Viene settato a "true" l'opzione d'ispezione nel Database. 5. Se il farmacologo sceglie "Close Monitor": 5.1. Viene settato a "true" l'opzione di chiusura del monitoraggio nel Database.
Postcondizioni	I farmaci possono essere rimossi, ispezionati o ne può essere chiuso il monitoraggio.

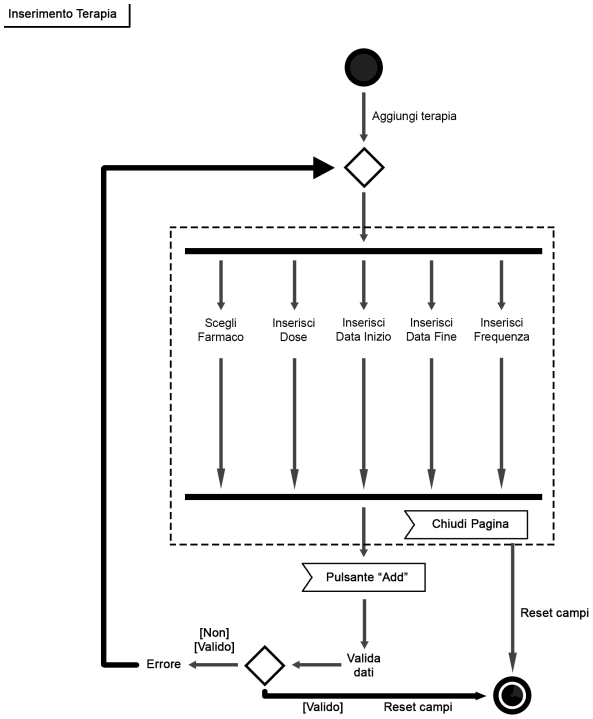
3.2 Activity Diagram

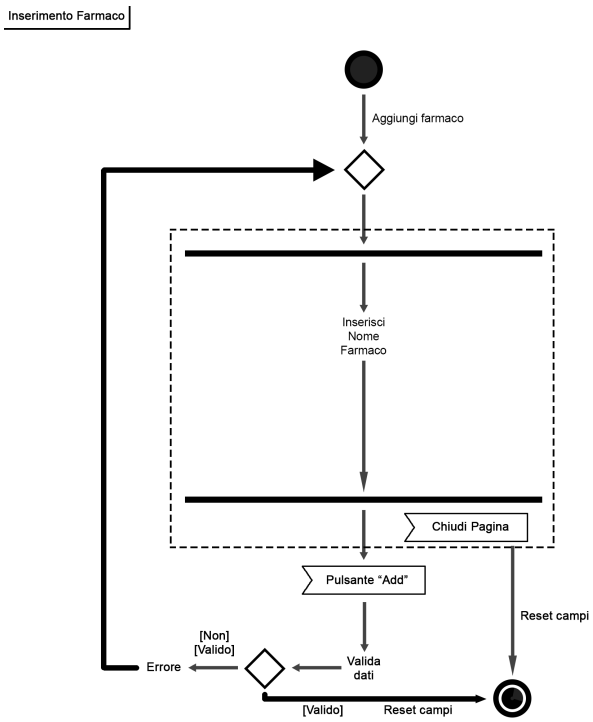
Gli activity diagram di seguito servono a esplicitare meglio il flusso d'esecuzione degli use case descritti in precedenza.

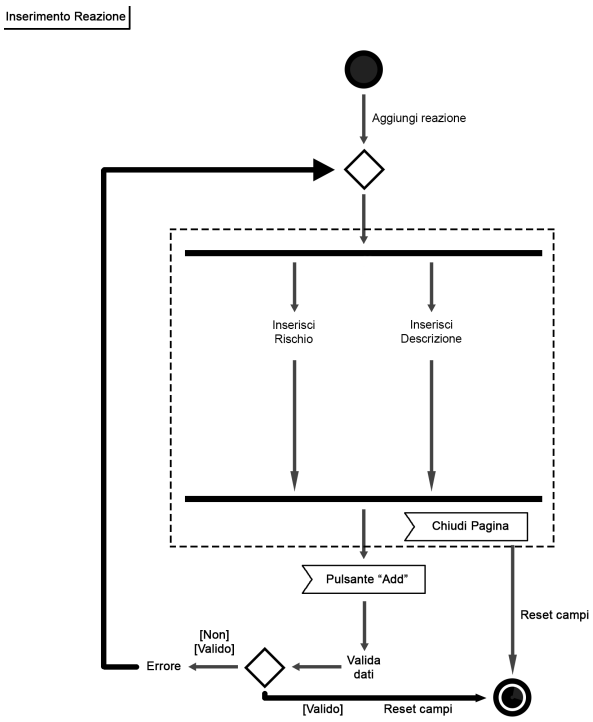




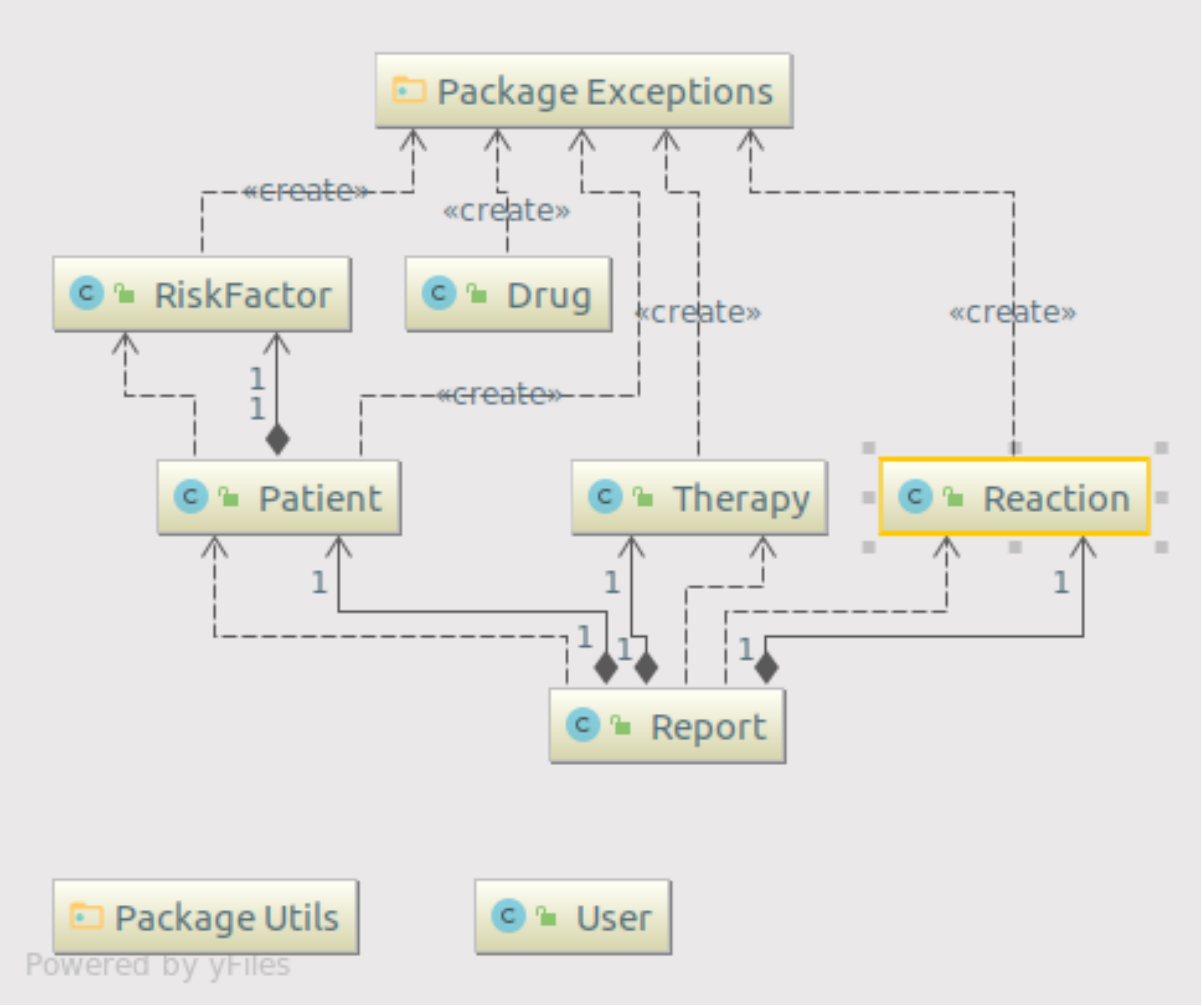
Come su può vedere nella corrispondente tabella, lo UC2 incorpora l'aggiunta di una nova terapia che a sua volta contiente la possibilità di aggiungere un nuovo farmaco e l'aggiunta di una nuova reazione. Nonostante siano parte dello stesso Use Case abbiamo deciso di dargli degli Activity Diagram.

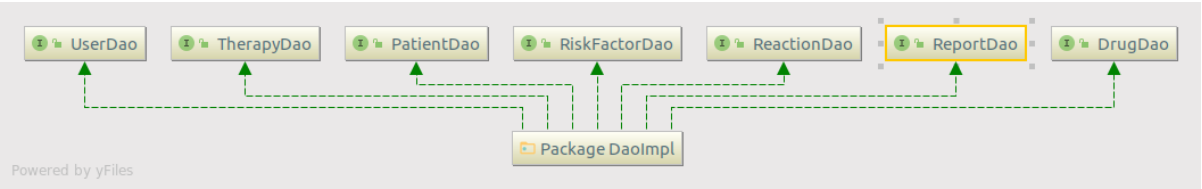


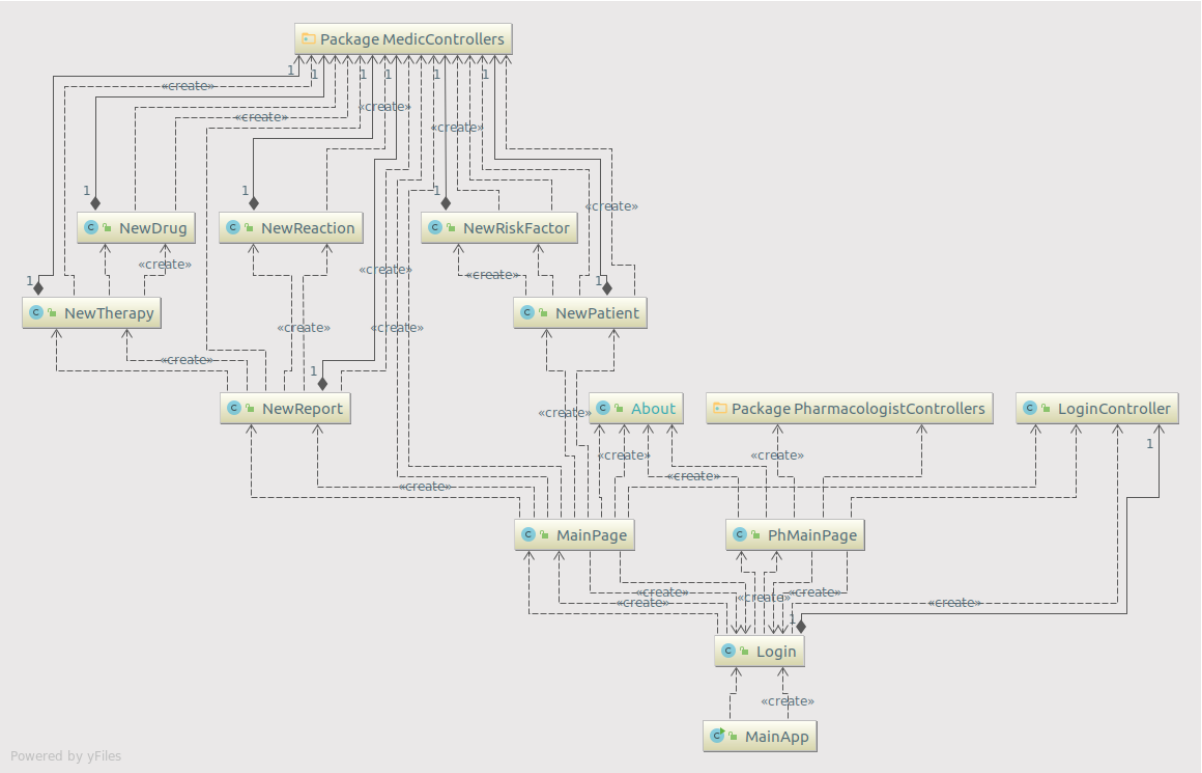



Inserimento Reazione
Aggiungi reazione
Inserisci Rischio
Inserisci Descrizione
Chiudi Pagina
Pulsante "Add"
Reset campi
[Non] [Valido]
Errore
Valida dati
[Valido]
Reset campi

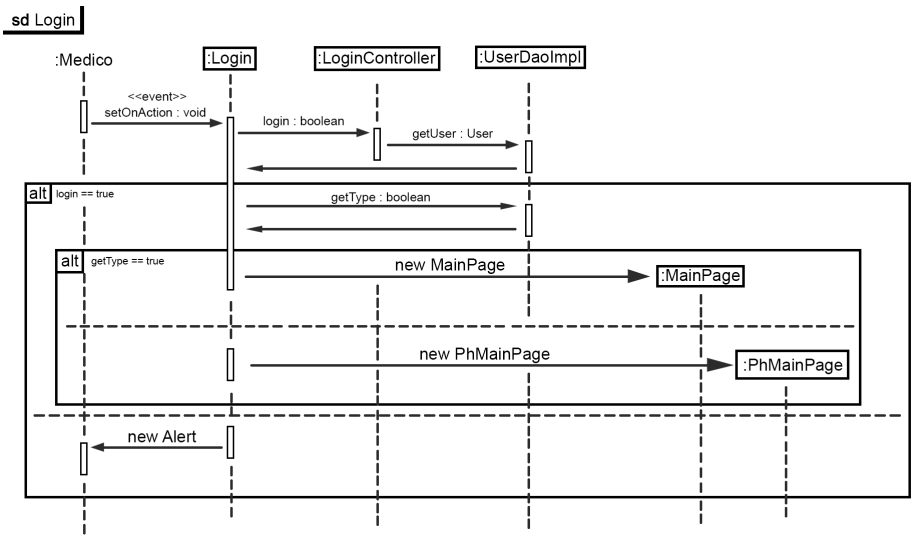
3.3 Class Diagram

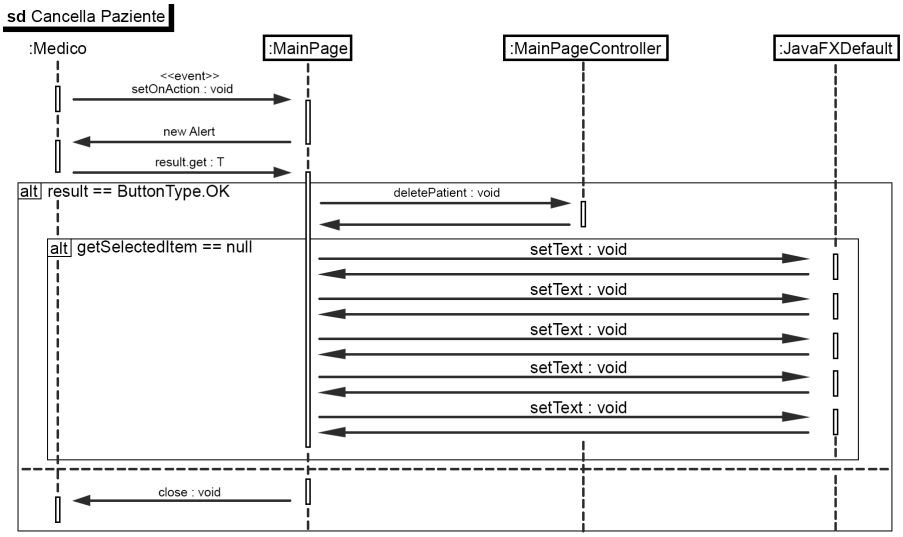


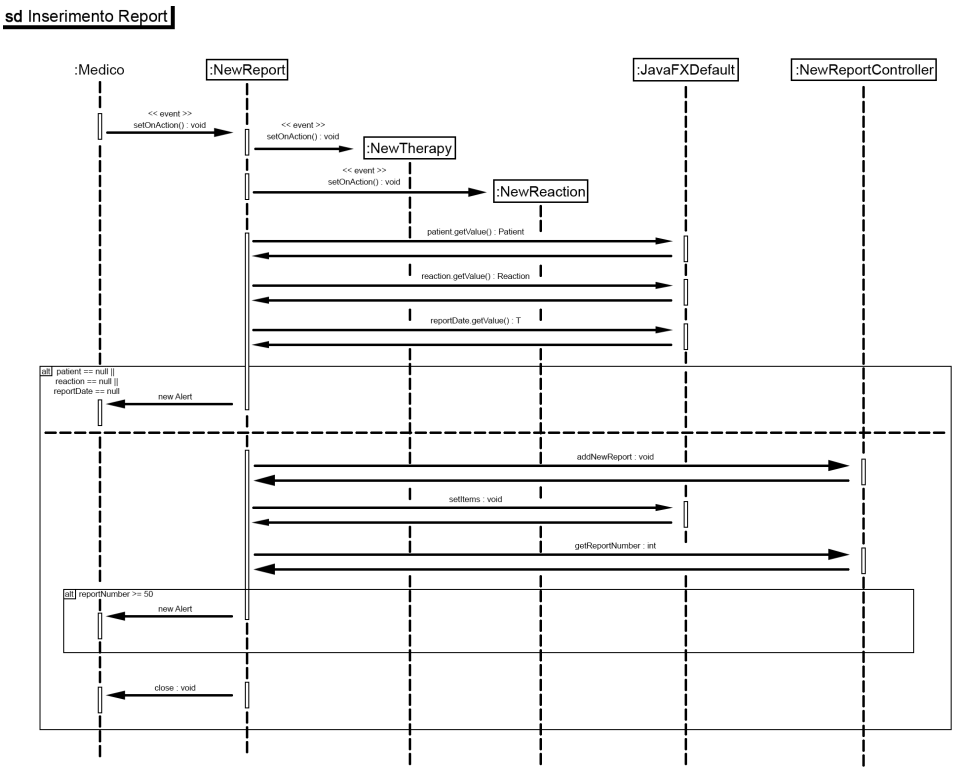


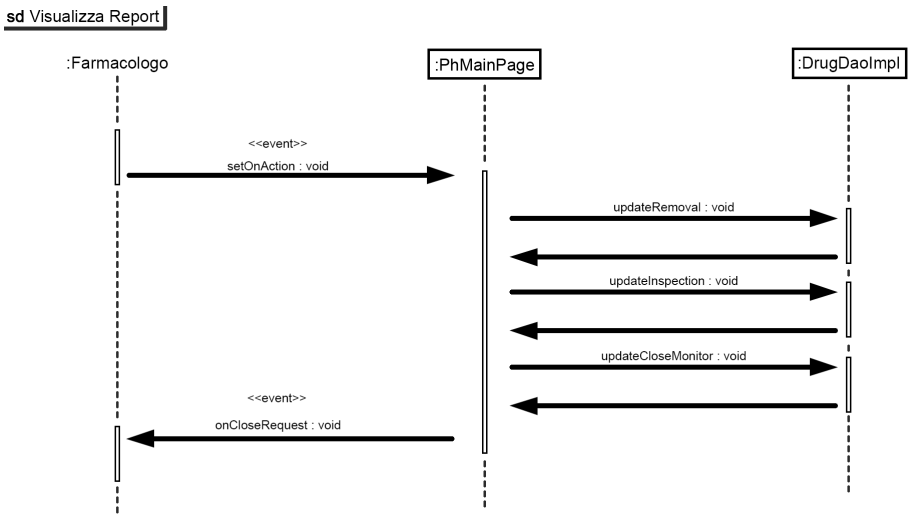


3.4 Sequence Diagram




sd Cancella Paziente
:Medico
:MainPage
:MainPageController
:JavaFXDefault
<<event>>
setOnAction : void
new Alert
result.get : T
alt result == ButtonType.OK
deletePatient : void
alt getSelectedItem == null
setText : void
setText : void
setText : void
setText : void
setText : void
close : void


sd Inserimento Report
:Medico
:NewReport
:JavaFXDefault
:NewReportController
<< event >>
setOnAction() : void
:NewTherapy
:NewReaction
patient.getValue() : Patient
reaction.getValue() : Reaction
reportDate.getValue() : T
alt patient == null ||
reaction == null ||
reportDate == null
new Alert
addNewReport : void
setItems : void
getReportNumber : int
alt reportNumber >= 50
new Alert
close : void



4 Scelte Progettuali

4.1 Sviluppo

Abbiamo scelto di sviluppare il prototipo utilizzando il linguaggio Java vista la sua grande flessibilità e la sua estesa diffusione in quello che è il mercato odierno. Nonostante questo, vista la nostra voglia di ampliare le nostre conoscenze del linguaggio al di fuori degli argomenti trattati a lezione, abbiamo deciso di utilizzare la libreria grafica JavaFX invece di Swing consapevoli del fatto che non avremmo potuto far uso di un designer di interfacce grafiche.
La libreria supporta completamente gli stili CSS, da noi poco utilizzati in quanto soddisfatti del risultato ottenuto con lo stile standard dei componenti.

4.2 Metodologia di Sviluppo

Il progetto è stato interamente sviluppato utilizzando Git come sistema di code versioning e GitHub per l'hosting del repository. Abbiamo seguito la metodologia di sviluppo Agile: il lavoro è stato diviso in due macroaree (area logica e GUI).
A sua volta ogni area è stata divisa in task più piccoli in modo da alleggerire il carico di lavoro e testati man mano che venivano inseriti nel codice. I design pattern sono stati scelti nella fase di sviluppo in base a quelle che erano le nostre necessità al fine di ottenere un codice performante ma allo stesso tempo leggibile.

4.3 Database

Nel corso dello sviluppo del prototipo si è manifestata la necessità di mantenere i dati nel tempo, per questo abbiamo implementato un basilare Database appoggiandoci ad SQLite3. Purtroppo al momento dello sviluppo le nostre competenze in basi di dati si limitavano a quanto appreso durante gli anni di liceo, per questo più che essere utilizzato come un vero e proprio RDBMS è stato implementato quasi come se i dati fossero organizzati in file.
È stata creata una classe DBConnection per aprire e chiudere la connessione con il database e

per ogni classe nel Model è stata creata un'apposita classe DAO al fine di accedere alla base di dati e creare liste che rappresentassero l'intero recordset di una determinata table.

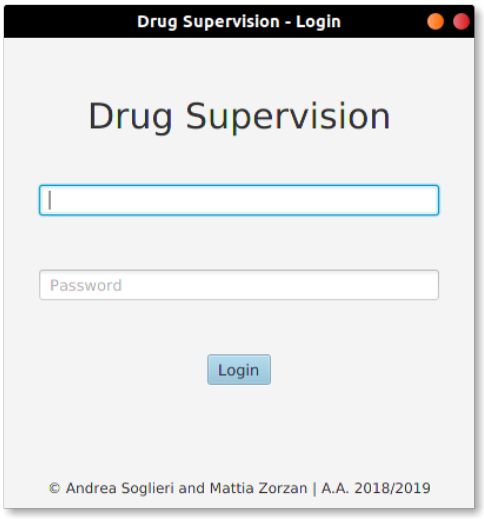
4.4 Organizzazione della GUI

È possibile trovare tre sezioni principali della GUI del prototipo:

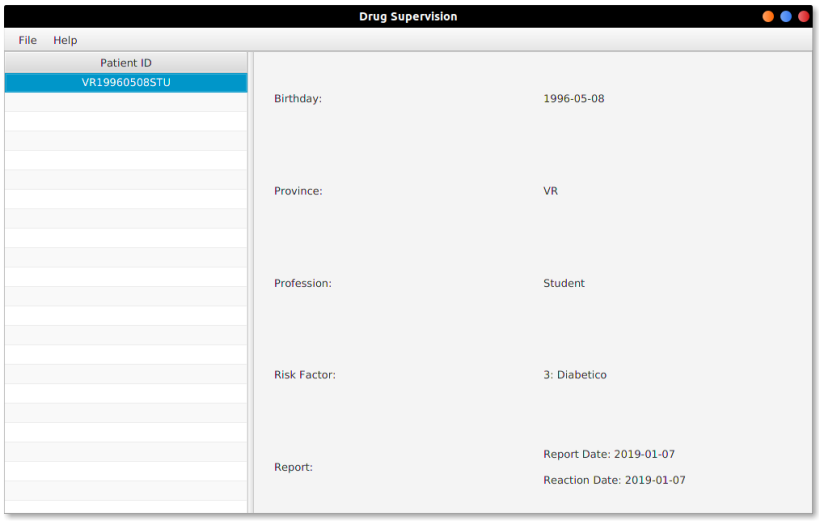
(a). Login: permette l'autenticazione del medico o del farmacologo, prerequisito per ogni Use-Case.

(b). View Medico: il focus è sui pazienti legati al medico autenticato. In questa view è possibile aggiungere, eliminare o visualizzare i dati relativi ad un paziente.

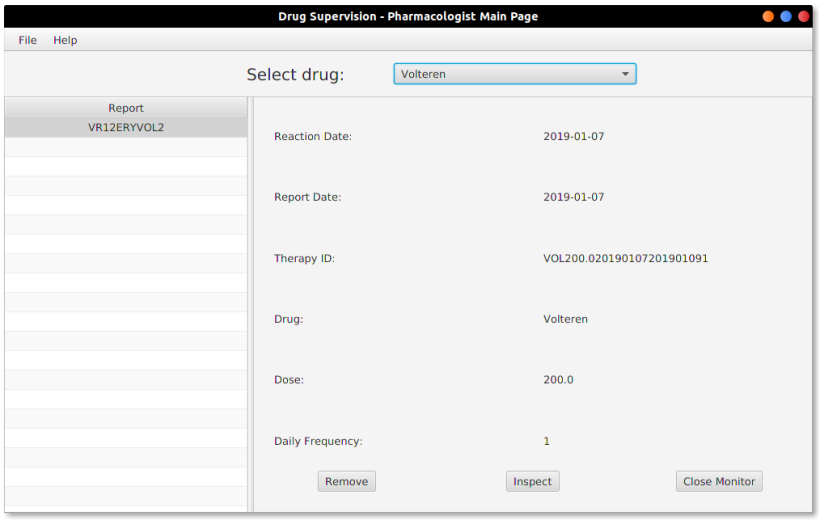
(c). View Farmacologo: il focus è sui report relativi alle terapie farmacologiche. Il farmacologo autenticato avrà la possibilità di monitorare gli effetti di un farmaco e, nel caso fosse dannoso, potrà rimuoverlo, ispezionarlo o chiuderne il monitoraggio.



(a) Schermata di Login



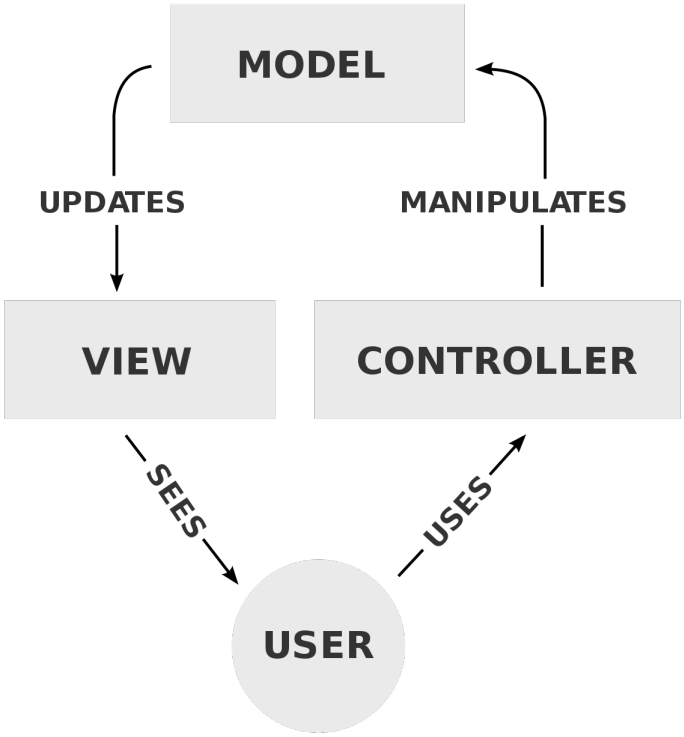
(b) Vista del Medico



(c) Vista del Farmacologo

4.5 MVC Pattern

L'applicazione del pattern prevede la suddivisione logica delle classi in tre componenti: Model, View e Controller.



4.6 Singleton Pattern

Per validare l'istanza dell'applicazione, secondo la nostra visione dell'elaborato, il pattern che meglio si addiceva alle nostre esigenze è il Singleton. Esso è stato utilizzato nella classe *LoginController*. Tramite un istanza da noi chiamata *loginInstance*, che memorizza lo username come stringa, permette di verificare l'avvenuto login tramite un metodo *private* chiamato *isLogged*. Questo restituisce un booleano, *true* se loginInstance ha un valore diverso da *null*, *false* altrimenti.

4.7 Observer Pattern

All'interno del codice non è stato implementato il modello teorico dell'Observer Pattern. Nonostante questo, il rapporto *Listener-Controller* esegue lo stesso tipo di operazioni. Nella classe *MainPage* viene creato un Listener che "ascolta" le variazioni all'inerno della TabelView, se ne rileva una provvede a modificare le Label all'interno della GridPane. Anche se non implementato direttamente da noi, l'uso fatto del Listener corrisponde a quello del metodo *update()* nello schema teorico dell'Observer Pattern.

4.8 Data Access Object Pattern

Per la gestione dei dati a database abbiamo ritenuto opportuno sfruttare il Data Access Object Pattern. Questo ci ha permesso di trattare ogni dato come fosse nativo di Java, senza dover passare a conversioni all'interno dei Controller. Il tutto è stato gestito tramite interfacce chiamate *DAO* e implementazioni denominate *DAOImpl*, i cui metodi definiscono oggetti ottenuti tramite l'uso di basiche query SQL.

5 Validazione

Per la validazione dei dati inseriti ci siamo appoggiati al sistema di Alert fornito da JavaFX. Vengono effettuati controlli tramite *if statements* e *try/catch* al fine di poter garantire una corretta esecuzione.
Per quanto riguarda la GUI non ci è stato possibile testarla automaticamente, abbiamo quindi provveduto ad eseguire un test manuale delle funzionalità. Ci siamo inoltre appoggiati a colleghi ed amici al fine di raccogliere un feedback veritiero sulle prestazioni del prototipo.

6 Conclusioni

Obiettivo del progetto non era quello di sviluppare un prodotto finito e completo ma di realizzare un prototipo rispettando le linee guida. Per questo non sono state implementate alcune funzionalità e l'interfaccia è rimasta "grezza". l progetto completo può essere trovato alla pagina di Github.