Università degli Studi di Verona

Ingegneria del Software

Med-H

Documentazione al prototipo

Mirko Morati, Noè Murr

20 febbraio 2018

Indice

1 Introduzione

Obiettivo dell'elaborato è la progettazione di un prototipo software per la gestione delle prescrizioni di farmaci da parte dei medici di base di una regione.

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

2 Requisiti

Di seguito viene riportato il testo dell'elaborato. I requisiti vengono identificati con una linea tratteggiata.

> Si vuole progettare un sistema informativo per gestire le prescrizioni di farmaci da parte dei medici di base di una regione. Il medico di base effettua la prescrizione di farmaci su richiesta del paziente (nel caso di farmaci legati a malattie croniche del paziente stesso) o in seguito alla visita. Nel caso di prescrizione su richiesta, il paziente può richiedere la prescrizione alla segreteria del medico che si occuperà di raccogliere le richieste, far preparare le ricette al medico e consegnare le prescrizioni ai pazienti.
>
> I farmaci potranno essere prescritti anche da eventuali medici consorziati con il medico di base di riferimento o in caso di assenza del medico di base, dal medico sostituto designato. Ogni prescrizione è caratterizzata dall'indicazione del paziente a cui fa riferimento, dalla data di emissione, dalla data di scadenza, dai farmaci prescritti. Ogni prescrizione è dotata di un codice identificativo univoco. La prescrizione deve essere presentata in farmacia per poter acquistare i farmaci necessari. Il farmacista utilizza la prescrizione, unitamente alla tessera sanitaria del paziente, per rilasciare lo scontrino parlante. In caso il medico abbia prescritto un farmaco e la farmacia non ne abbia a disposizione, il farmacista provvederà ad ordinare il farmaco per il paziente. Per ogni paziente sono memorizzati: il codice sanitario univoco, il cognome, il nome, la data di nascita, il luogo di nascita, l'indirizzo di residenza (via, civico, cap, città, provincia). Per ogni paziente è possibile memorizzare gli eventuali fattori di rischio presenti (anche più d'uno).
>
> Per ogni medico di base si registrano il codice fiscale, il codice univoco assegnato a livello regionale, il cognome, il nome, la data e il luogo di nascita. Per i medici di base con specializzazione si registrano anche l'ultima specializzazione ottenuta e la data di conseguimento. Ad ogni medico sono legate tutte le prescrizioni effettuate.
>
> Il medico può richiedere al sistema di visualizzare la lista dei farmaci prescritti ad un certo paziente in un certo periodo di tempo (mese, trimestre, semestre o anno), oppure la lista di farmaci prescritta nel complesso su base mensile o annuale. Inoltre può verificare se e quando qualche paziente non ha usato la prescrizione emessa o se un paziente ha preferito il farmaco generico.
>
> In ogni prescrizione possono apparire al più 5 farmaci. Ogni farmaco è caratterizzato da nome commerciale univoco, dal costo, dalla forma farmaceutica, dalle controindicazioni, e dalla sua classificazione secondo il sistema ATC (Anatomico-Terapeutico-Chimico).

Per ogni farmaco si tiene traccia dei principi attivi contenuti: per ogni principio attivo di un farmaco sono indicati la quantità contenuta in ogni unità di farmaco (ad esempio, “500 mg per compressa” o “120 mg per ml”). Per ogni principio attivo sono memorizzati un nome univoco, una sua descrizione e l’effetto principale. Il medico può poi specificare quali prescrizioni contengono farmaci che potrebbero interagire con risultati negativi per il paziente. In tal caso sono specificate le coppie di farmaci della stessa prescrizione che possono interagire.

Il medico può richiedere al sistema di visualizzare la quantità di un certo farmaco prescritta ai suoi pazienti su base mensile, trimestrale, semestrale o annuale. Può anche visualizzare la lista dei pazienti a cui ha prescritto un certo farmaco e la lista delle eventuali reazione avverse comunicate all’agenzia di riferimento.

3 UML

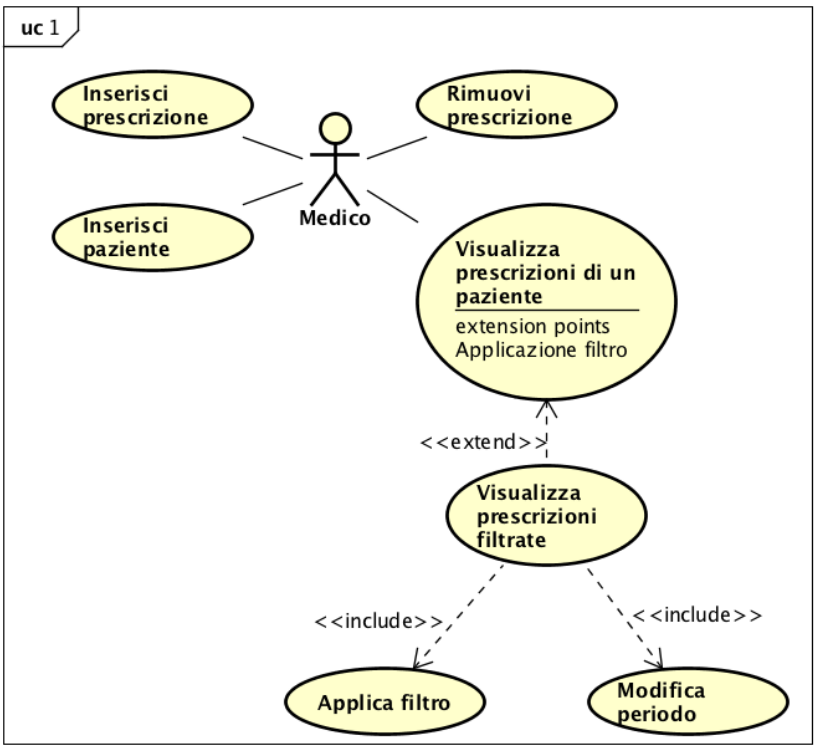
In questa sezione sono presentati i diagrammi richiesti per la documentazione del prototipo.

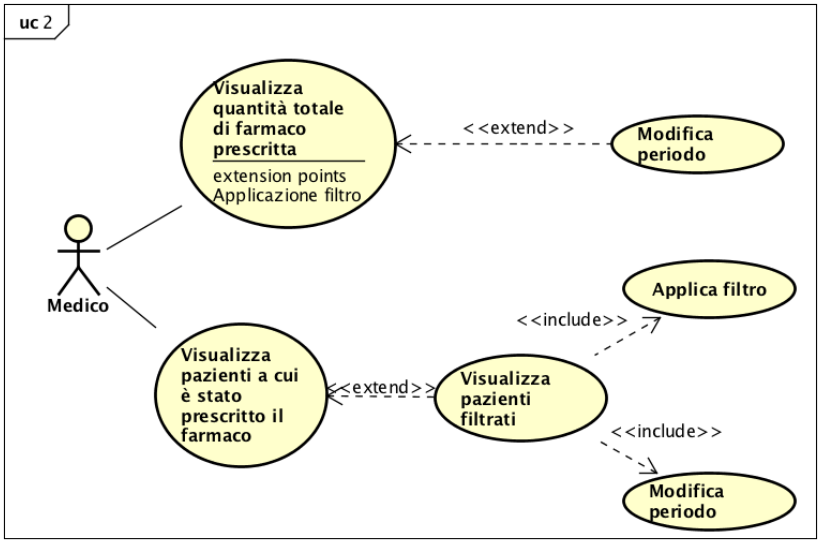
3.1 Use Cases

Gli use case individuati sono stati suddivisi in due macroaree definite dalla scelta di disposizione della GUI:

1. Vista con focus sui pazienti
2. Vista con focus sui farmaci.

Di seguito vengono proposte alcune schede descrittive degli use case principali.





ID	UC1: Inserimento paziente.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico preme il pulsante di aggiunta paziente e viene visualizzato il dialog corrispondente. 2. Inserisce i dati relativi alla persona ed eventuali fattori di rischio. 3. Se il medico preme Ok: 3.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 3.1.1. Il paziente viene inserito a database. 3.1.2. Il dialog viene chiuso. Se non sono validi viene visualizzato un messaggio di errore. 4. Se il medico preme Annulla: 4.1. Il dialog viene chiuso.
Postcondizioni	La lista dei pazienti è aggiornata.

ID	UC2: Inserimento prescrizione.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico seleziona un paziente. 2. Il medico preme il pulsante di aggiunta prescrizione e viene visualizzato il dialog corrispondente. 3. Inserisce i dettagli della prescrizione. 4. Se il medico preme Ok: 4.1. Viene effettuata la validazione dei dati immessi. Se sono validi: 4.1.1. La prescrizione viene aggiunta al database. 4.1.2. Il dialog viene chiuso. Se non sono validi viene visualizzato un messaggio di errore. 5. Se il medico preme Annulla: 5.1. Il dialog viene chiuso.
Postcondizioni	La lista delle prescrizioni è aggiornata.

Il seguente use case è stato pensato per soddisfare il requisito della visualizzazione della lista dei farmaci prescritti ad un certo paziente in un certo periodo di tempo.

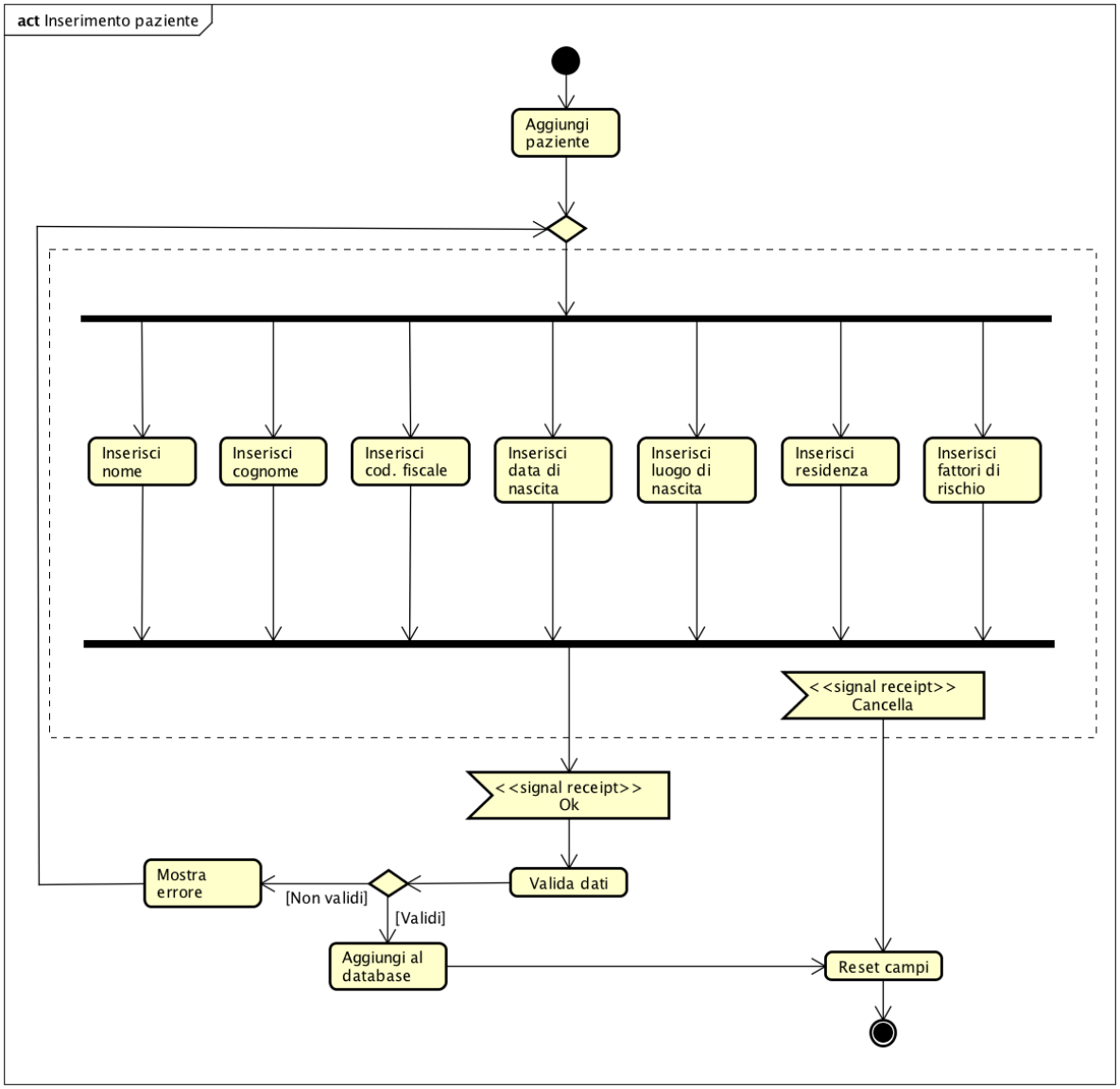
ID	UC3: Visualizza prescrizioni di un paziente.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico deve selezionare un paziente. 2. Il medico può applicare un filtro: 2.1. Il filtro viene attivato. 2.2. Viene selezionato un range tra mese, trimestre, semestre, anno e custom. 2.3. Viene selezionata una data di inizio per modificare il range.
Postcondizioni	La lista delle prescrizioni è aggiornata.

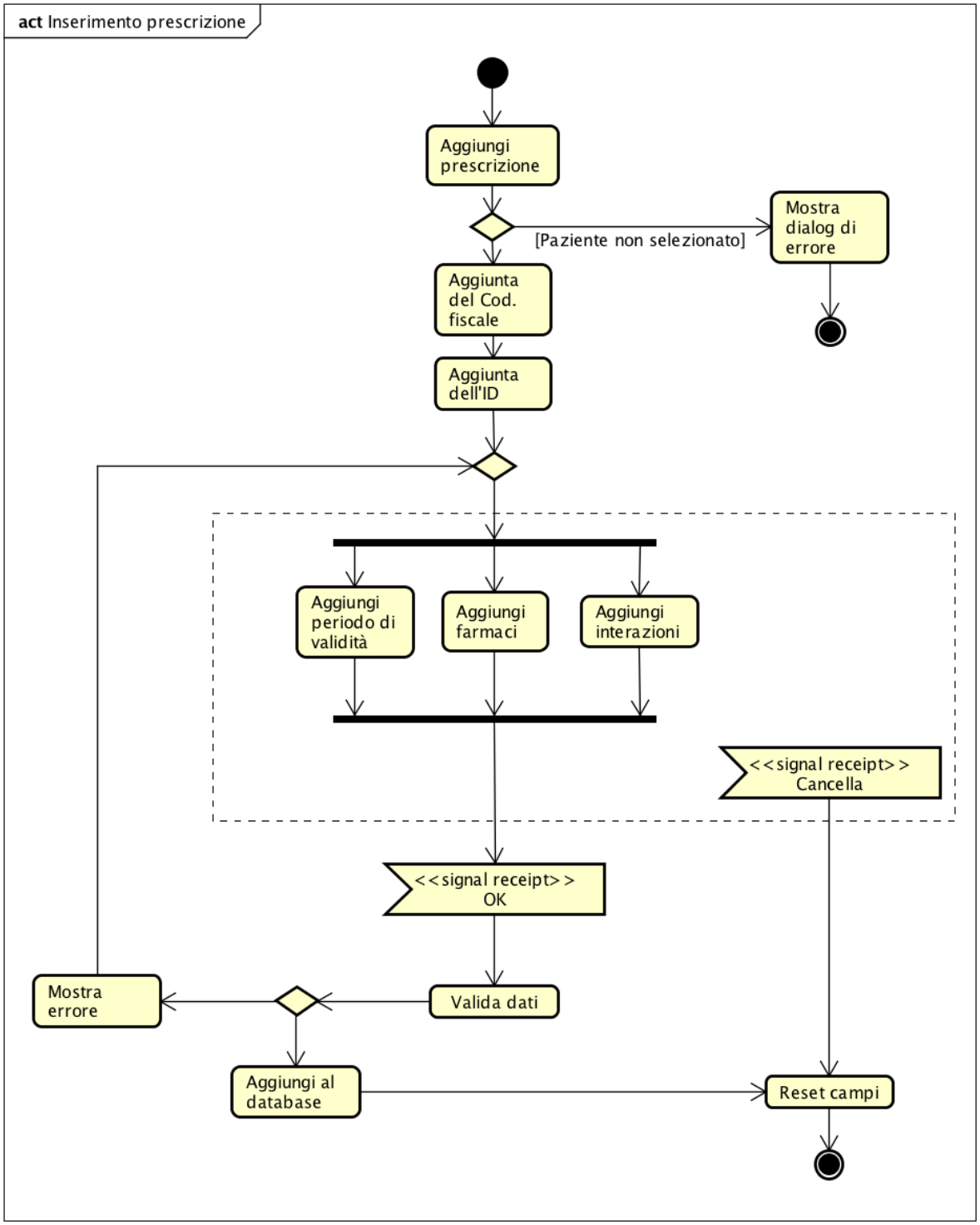
ID	UC4: Visualizza pazienti a cui è stato prescritto il farmaco.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico deve selezionare un farmaco. 2. Il medico può applicare un filtro: 2.1. Il filtro viene attivato. 2.2. Viene selezionato un range tra mese o anno. 2.3. Viene selezionata una data di inizio per modificare il range.
Postcondizioni	La lista dei pazienti è aggiornata.

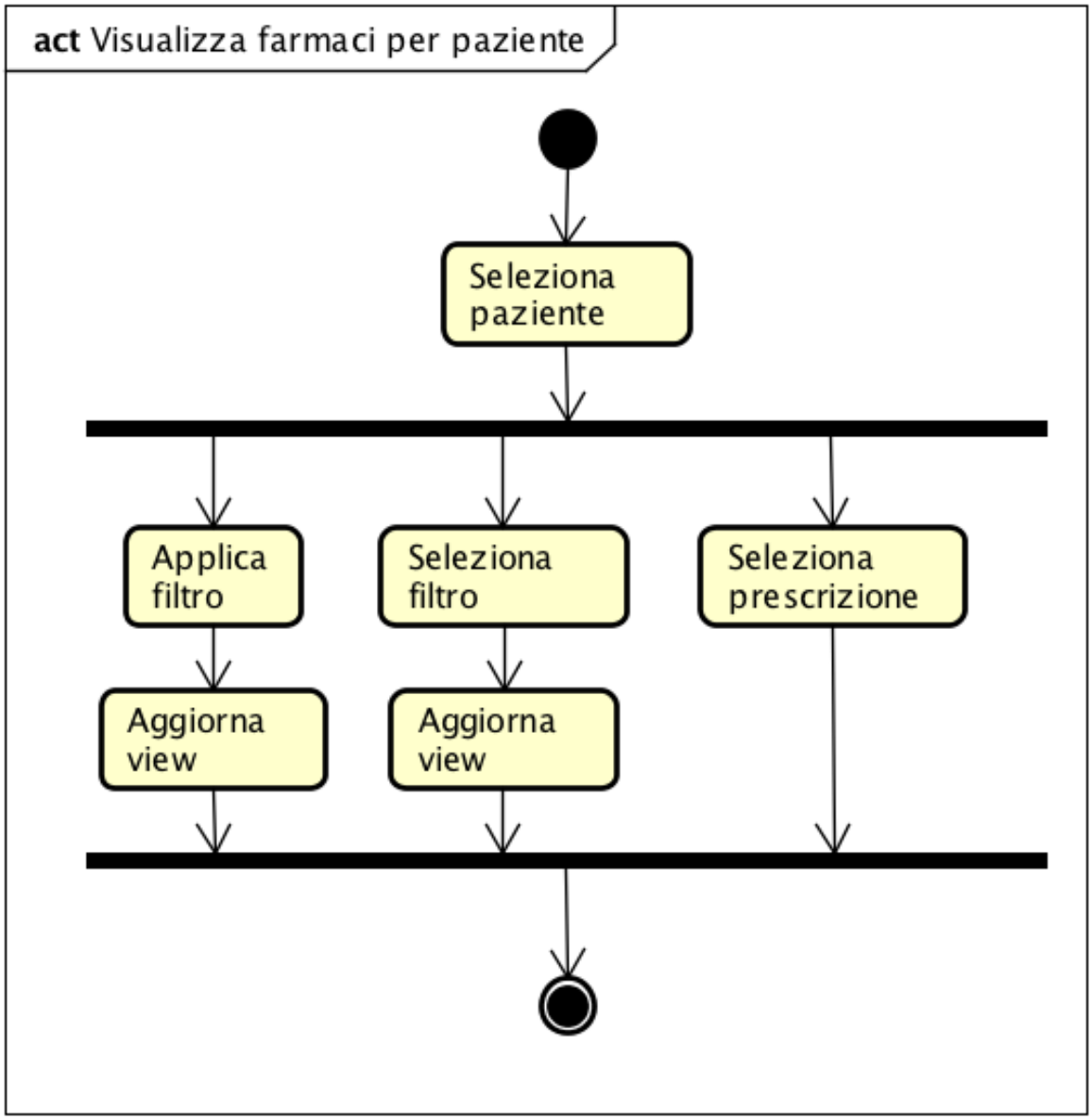
ID	UC5: Visualizza quantità di farmaco totale prescritta.
Attori	Medico
Precondizioni	Il medico deve aver effettuato il login.
Sequenza	1. Il medico deve selezionare un farmaco. 2. Il medico può modificare il filtro relativo alla quantità: 2.1. Viene selezionato un range tra mese, trimestre, semestre e anno. 2.2. Viene selezionata una data di inizio per modificare il range.
Postcondizioni	La label della quantità è aggiornata.

3.2 Activity Diagram

I seguenti activity diagram cercano di modellare in modo più dettagliato il flusso d'esecuzione relativo ad alcuni degli use case precedentemente proposti. Si noti che i diagrammi relativi agli use case `UC4` e `UC5` risulterebbero molto simili al diagramma presentato per lo use case `UC3`.

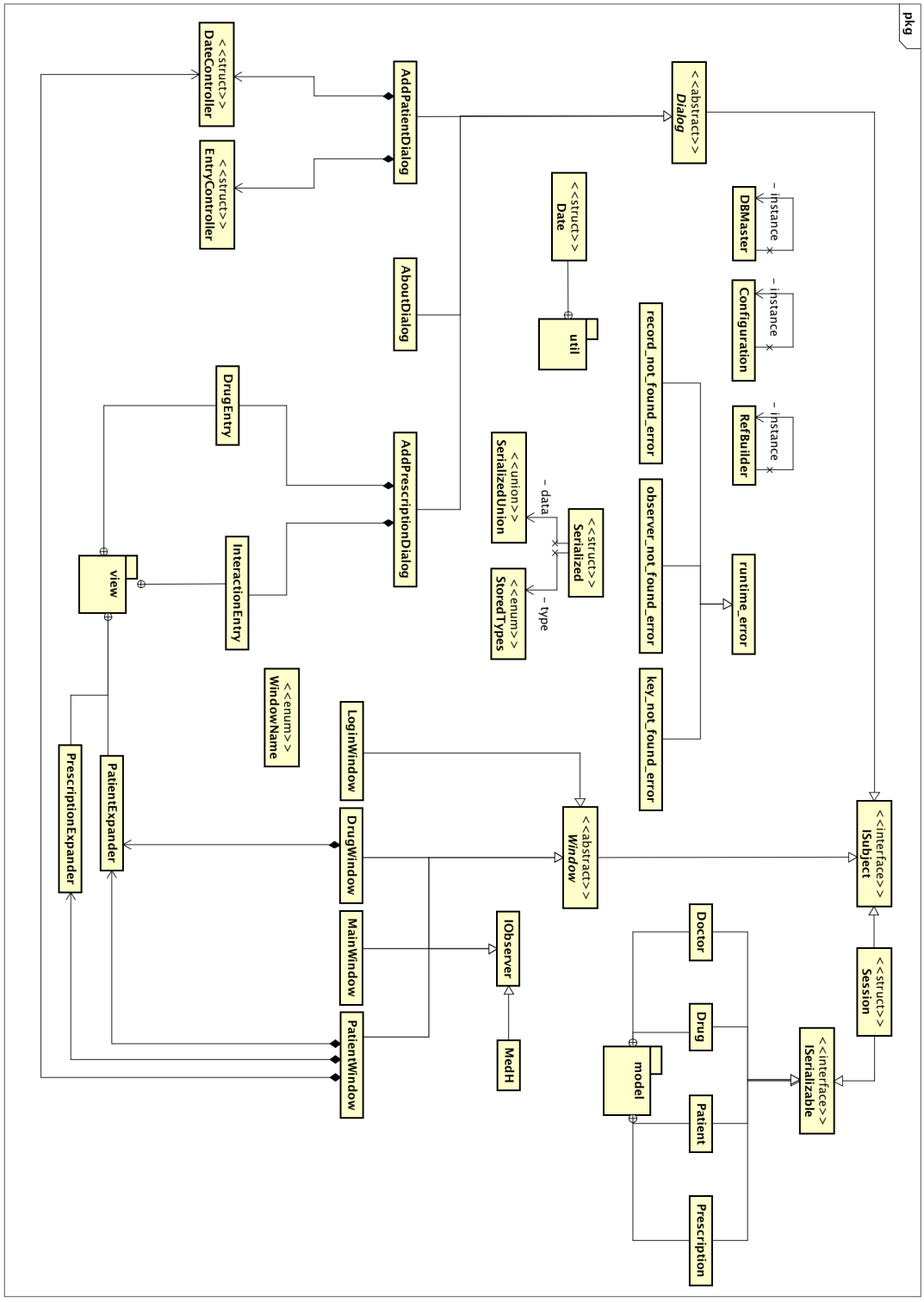



act Inserimento prescrizione
Aggiungi prescrizione
[Paziente non selezionato]
Mostra dialog di errore
Aggiunta del Cod. fiscale
Aggiunta dell'ID
Aggiungi periodo di validità
Aggiungi farmaci
Aggiungi interazioni
<<signal receipt>> Cancella
<<signal receipt>> OK
Mostra errore
Valida dati
Aggiungi al database
Reset campi

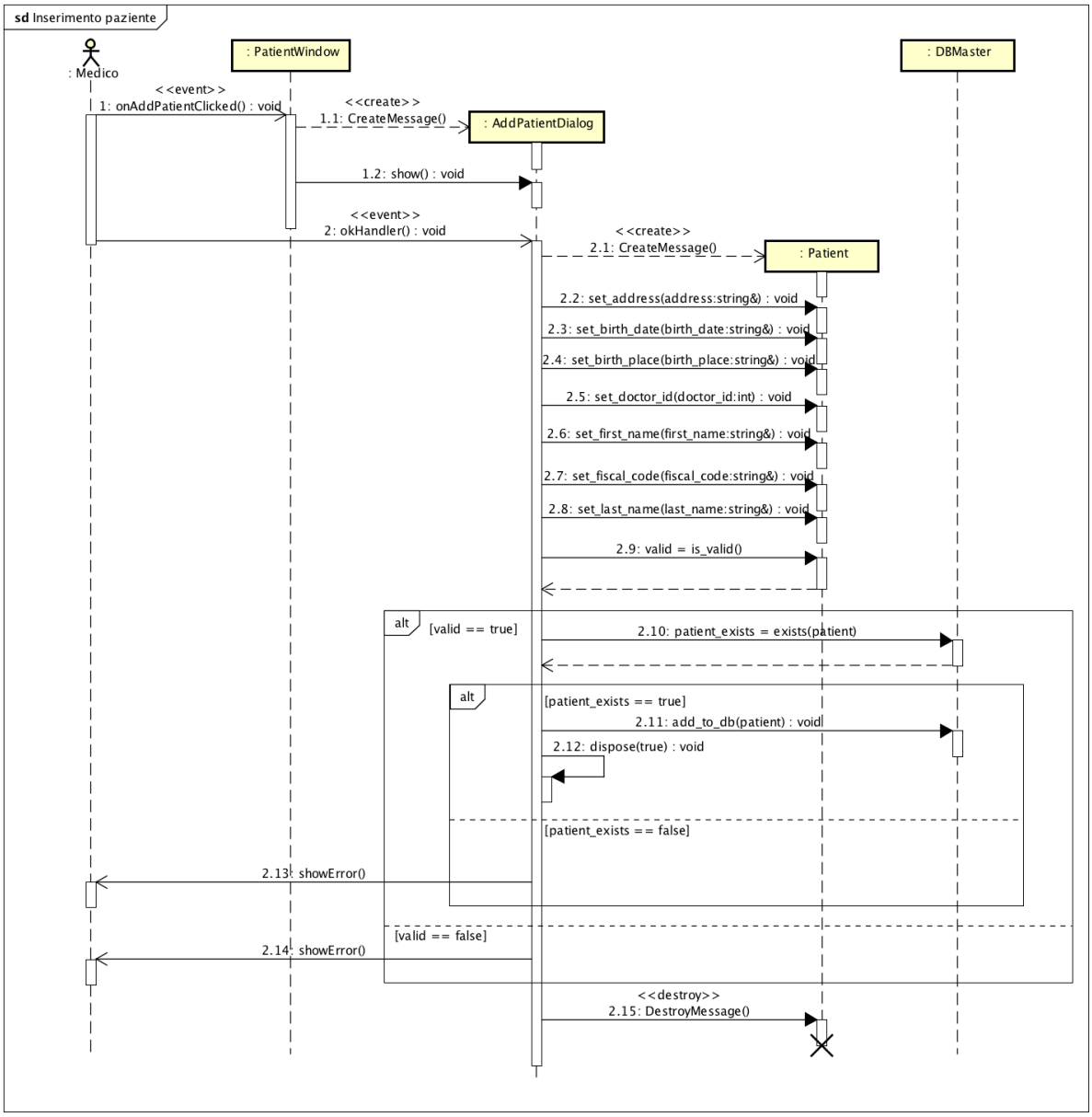

act Visualizza farmaci per paziente
Seleziona paziente
Applica filtro
Seleziona filtro
Seleziona prescrizione
Aggiorna view
Aggiorna view

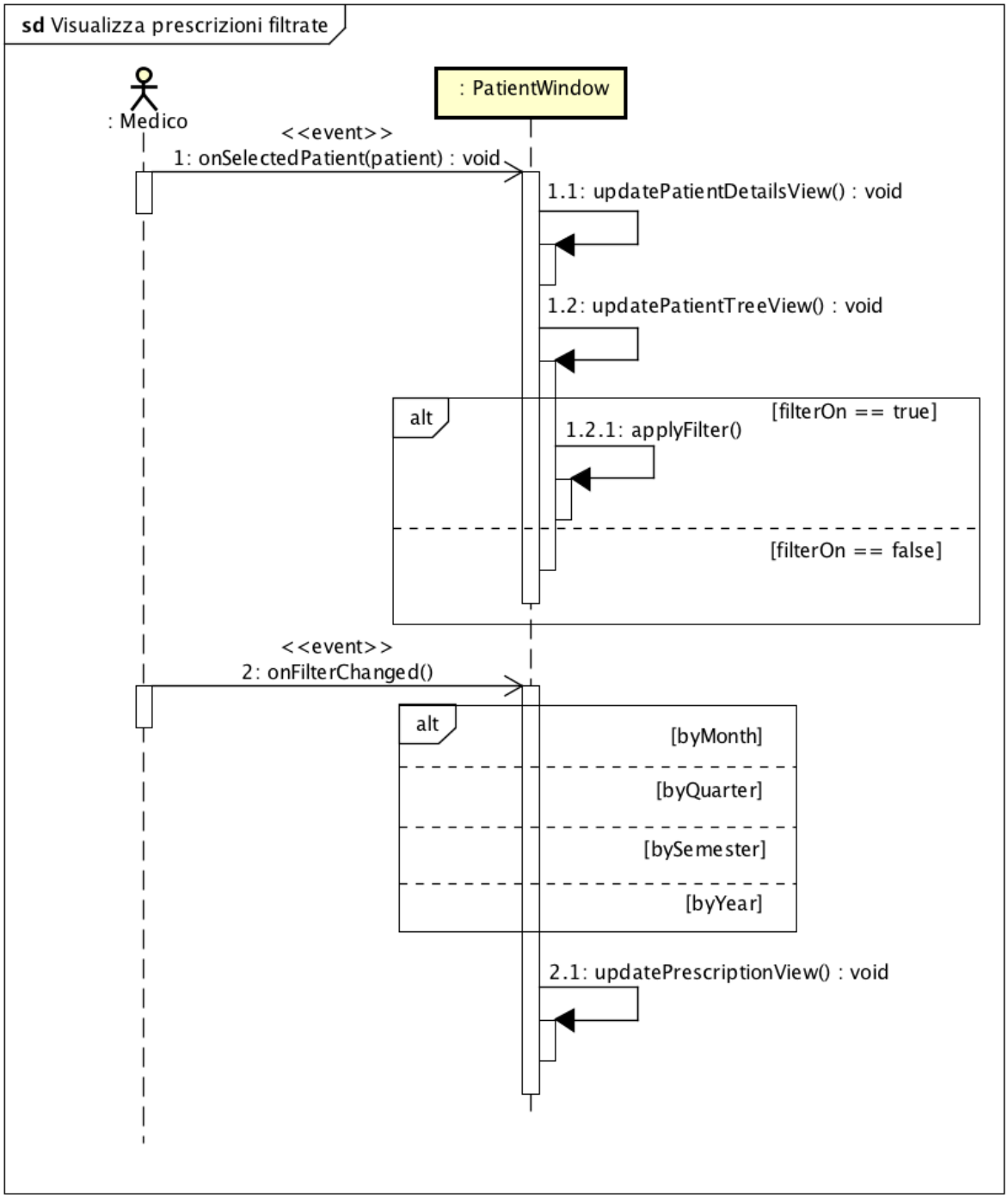
3.3 Class Diagram

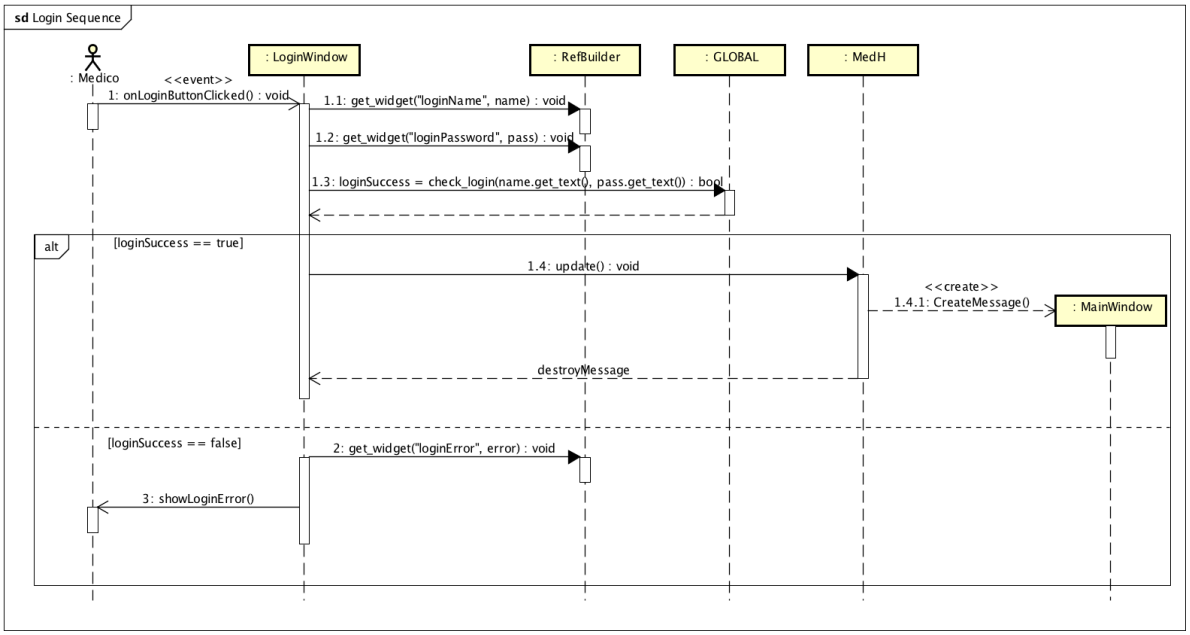
Il class diagram qui proposto non è comprensivo di attributi e proprietà, serve a dare un'idea generale dell'organizzazione del prototipo software. Si rimanda al class diagram completo all'Appendice A.



3.4 Sequence Diagram




sd Visualizza prescrizioni filtrate
: Medico
: PatientWindow
<<event>>
1: onSelectedPatient(patient) : void
1.1: updatePatientDetailsView() : void
1.2: updatePatientTreeView() : void
alt
[filterOn == true]
1.2.1: applyFilter()
[filterOn == false]
<<event>>
2: onFilterChanged()
alt
[byMonth]
[byQuarter]
[bySemester]
[byYear]
2.1: updatePrescriptionView() : void



4 Scelte Progettuali

4.1 Sviluppo

Abbiamo scelto di sviluppare il prototipo usando il linguaggio C++ in modo da poter ampliare la nostra conoscenza applicando i nuovi standard ad un progetto reale. L'obbligo di rispettare dei requisiti applicando dei pattern di programmazione orientata agli oggetti ci ha spinti a ricercare ed implementare soluzioni al di fuori della nostra esperienza.

Uno stimolo in più l'abbiamo ricevuto quando abbiamo deciso di usare la libreria grafica Gtkmm3, un'interfaccia verso la più popolare libreria Gtk+. Alcune delle feature che ci hanno spinto ad operare questa scelta sono state:

- uso dell'ereditarietà per definire nuovi widget
- type-safe signal handlers
- polimorfismo
- management automatico della memoria in C++
- cross-platform
- open source

Oltre a questo abbiamo trovato utile la possibilità di costruire la GUI utilizzando il RAD tool Glade, un designer di interfacce grafiche. La definizione dell'interfaccia risiede in un file XML che viene parsato all'inizio del programma ed i widget costruiti vengono resi disponibili tramite una classe singleton (si veda MVC Pattern). Da notare il supporto completo dei componenti agli stili CSS: la libreria non esclude la possibilità di personalizzare l'interfaccia senza passare dal tool né la creazione di temi da parte di utenti.

4.1.1 Metodologia di Sviluppo

Il progetto è stato interamente sviluppato usando Git come sistema di versionamento del codice e Github per l'hosting del repository. Abbiamo seguito una metodologia di sviluppo Agile: dopo aver analizzato i requisiti del testo proposto abbiamo diviso il lavoro in due macroaree (logica ed user interface). Le aree sono state suddivise in task brevi e di facile implementazione, testando gradualmente il nuovo codice inserito. I design pattern applicati sono stati decisi nella fase di sviluppo valutando di volta in volta le necessità per mantenere il progetto leggibile e scalabile.

In molte occasioni è stato necessario operare un refactoring del codice per migliorare in efficacia il prototipo. Il sistema di versionamento ci ha permesso di dividere logicamente i task in branch e mantenere una cronologia delle modifiche in modo da poterle eventualmente scartare.

4.2 Database

Siccome si è reso necessario mantenere un insieme di dati persistenti abbiamo deciso di appoggiarci al sistema di database Sqlite3. Purtroppo le nostre competenze in basi di dati all'inizio del progetto erano alquanto scarne, quindi più che essere utilizzato come un vero e proprio RDBMS è stato utilizzato quasi come se i dati fossero organizzati in file.

È stata realizzata una classe di interfacciamento che provvede a nascondere la complessità delle transazioni (si veda Facade Pattern).

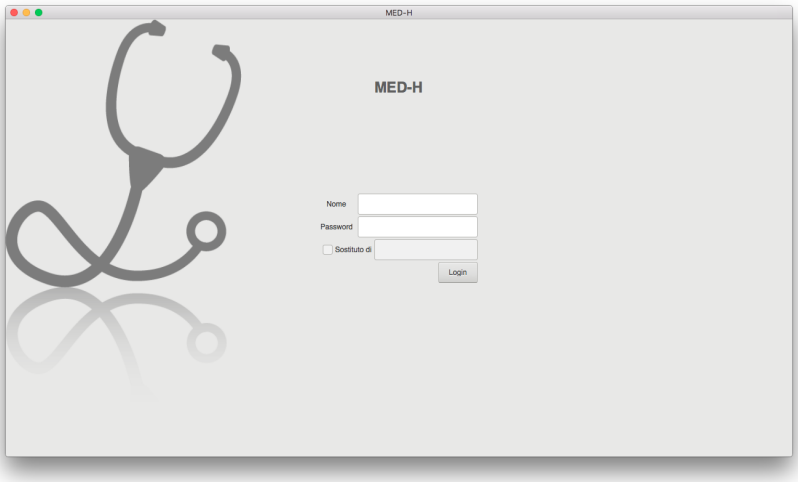
4.3 Organizzazione della GUI

È possibile individuare tre sezioni principali della GUI del prototipo, ognuna delle quali cerca di rispondere ad un insieme di requisiti specifici:

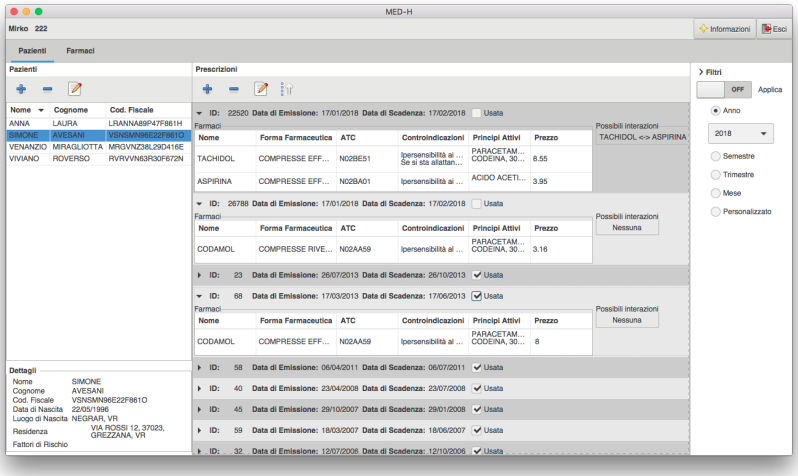
(a). Login: permette l'autenticazione del medico, prerequisito fondamentale per tutti gli use case definiti.

(b). Tab pazienti: il focus è sui pazienti legati al medico autenticato. In questa sezione è possibile vedere, filtrare ed aggiungere le prescrizioni relative ad un paziente (oltre ad altre operazioni), in questo modo è possibile vedere l'elenco di farmaci prescritti ad un paziente in un dato periodo.

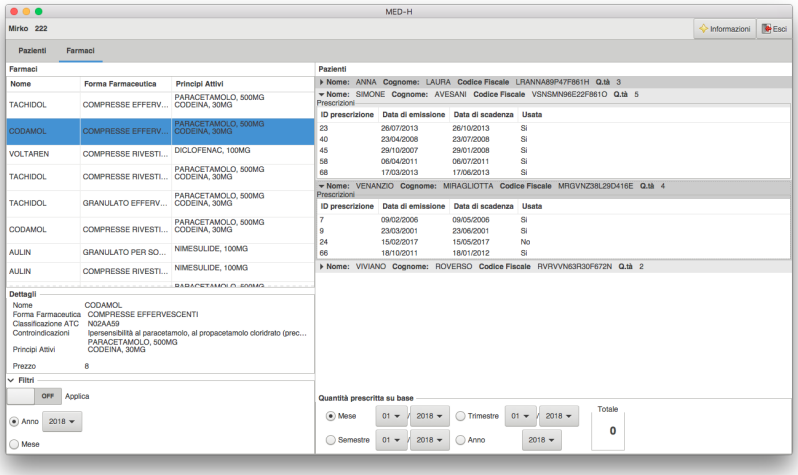
(c). Tab farmaci: il focus è sui farmaci disponibili a database. In questa sezione è possibile per ogni farmaco visualizzare, in un determinato periodo, una lista dei pazienti a cui è stato prescritto con dettagli. È anche possibile visualizzare la quantità di un determinato farmaco prescritta a tutti i pazienti in un determinato periodo.



(a) Schermata di Login



(b) Tab Pazienti

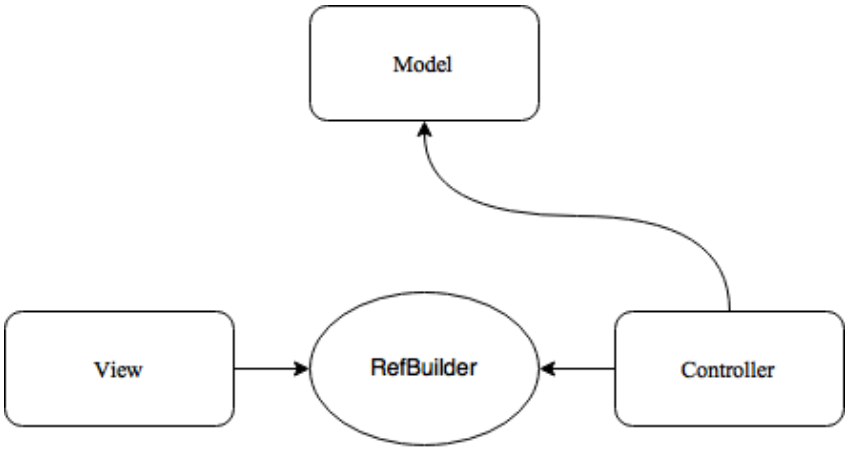


(c) Tab Farmaci

4.4 MVC Pattern

L'applicazione del pattern prevede la suddivisione logica delle classi in tre componenti: model, view, controller. Abbiamo adattato il pattern per poterlo sfruttare al meglio nonostante la realizzazione dell'interfaccia grafica in XML abbia reso complicato organizzare la comunicazione tra view e relativi controller. La libreria Gtkmm3 mette a disposizione un Builder che si occupa della costruzione della view a partire dall'XML; per rendere disponibili i widget ai controller abbiamo realizzato un singleton che si occupa di fornire un'interfaccia di collegamento.

Svantaggio di questa soluzione è la mancanza di un incapsulamento reale che impedisca ad un controller di accedere ad un widget di una view con cui non dovrebbe avere collegamenti. Questo è un compito lasciato al programmatore.



4.5 Singleton Pattern

Per raccogliere insiemi di funzioni che necessitavano di una configurazione, abbiamo ottenuto opportuno utilizzare il singleton pattern. In questo modo abbiamo fornito in tutto il codice la possibilità di utilizzare un insieme di interfacce inizializzate in un punto solo del programma (e "safe"). Le classi che adottano il pattern sono:

- DBMaster
- RefBuilder
- Configuration
- Login.

Da notare che in C++ è possibile definire due implementazioni differenti per il singleton pattern, una lazy e una eager. La nostra implementazione è lazy, ovvero l'istanza della classe viene creata solo alla prima richiesta di utilizzo.

4.6 Observer Pattern

Anche l'observer pattern è stato adattato al prototipo. Viene utilizzato per sincronizzare i dialog con le rispettive view principali in caso di modifiche che rendono necessario un aggiornamento.

A differenza del pattern originale la nostra implementazione permette di specificare un parametro nella notifica in modo da poter differenziare gli update in ogni observer. Nel prototipo non si è reso necessario (ma rimane possibile) specificare più di un observer per ogni subject.

4.7 Facade Pattern

Il facade pattern permette di “mascherare” per mezzo di un’interfaccia di semplice utilizzo la complessità di un’operazione. È stato adottato nella classe DBMaster per fornire dei metodi che coprano l’utilizzo della libreria (sviluppata in C) per l’accesso al database sqlite.

5 Validazione

La parte di codice ritenuta più critica è quella relativa alle transazioni a database. Abbiamo scritto degli unit test utilizzando la libreria open source Catch2 che permette di creare test case in modo semplice e veloce.

Per quanto riguarda l’interfaccia utente abbiamo dovuto testarla manualmente in quanto non abbiamo trovato un metodo automatico. È stato anche richiesto ad alcuni colleghi di provare il prodotto in modo da raccogliere feedback sull’esperienza di utilizzo dello stesso.

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”. Al termine del progetto abbiamo convenuto che la scelta della libreria per l’interfaccia grafica non è stata delle più ottimali poiché purtroppo la documentazione manca di validi esempi e non è sempre intuitivo passare dall’idea al codice.

Il progetto completo può essere trovato alla pagina di Github.