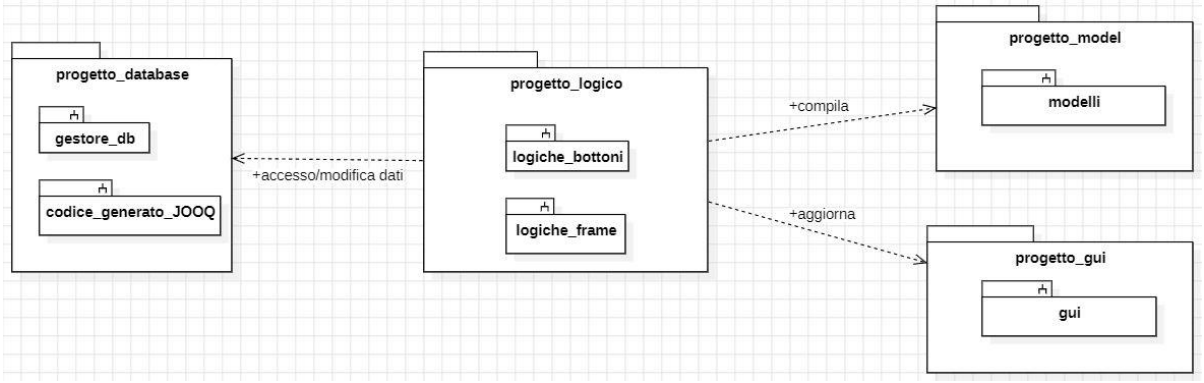
Progetto Ingegneria del Software - Documentazione

Medical
Environment
Database

1. Ciclo di vita del software

Il processo di sviluppo seguito nel corso della progettazione è stato di tipo agile, più precisamente SCRUM.
Lo sviluppo del software è stato suddiviso inizialmente in 3 pacchetti di lavoro principali (database, logica, GUI), ai quali è stato aggiunto successivamente un quarto (modello).
Tutte le modifiche alla baseline definita inizialmente sono state discusse durante le riunioni tenute giornalmente, come previsto dal modello SCRUM.
La definizione dell'architettura dei pacchetti di lavoro è stata dunque riassunta nel seguente diagramma UML:



In seguito si è rivelato necessario aggiungere ulteriori sotto-pacchetti all'interno di alcuni dei progetti, ma la struttura e il funzionamento complessivi sono rimasti invariati.

2. Gestione della configurazione

I sistemi utilizzati per gestire la configurazione del progetto sono stati GitHub e l'applicazione GitHub desktop.
Inizialmente le operazioni di commit sono state eseguite direttamente sul main branch, ma successivamente abbiamo iniziato a gestire gli aggiornamenti del codice attraverso l'apertura di branch e l'utilizzo di pull request.
Nel corso dell'implementazione alcuni errori e problematiche inattese sono state notificate e susseguentemente gestite attraverso l'utilizzo delle issues;
queste sono state aperte dal componente del team che riscontrava il problema e assegnate da parte dello SCRUM master a uno o più membri in grado di completare la richiesta.
Per monitorare lo stato di avanzamento dei pacchetti di lavoro è stata utilizzata una kanban board, aggiornata progressivamente ogni qual volta veniva raggiunta una milestone.

3. Organizzazione del team e gestione del personale

La configurazione del team di sviluppo adotta il paradigma della "Squadra Agile", conformemente al processo di sviluppo software selezionato.
In particolare, i ruoli assegnati ai vari membri del gruppo sono i seguenti:

- **Gotti Daniele**, matricola 1079011 -> principale programmatore dell'interfaccia grafica;
- **Bolis Filippo Antonio**, matricola 1079493 -> product owner, principale programmatore delle funzionalità logiche;
- **Mazzoleni Gabriele**, matricola 1079514 -> SCRUM master, principale programmatore della gestione del database, assistente programmatore delle funzionalità logiche;
- **Masinari Gabriele**, matricola 1079692 -> principale redattore della documentazione UML, assistente programmatore dell'interfaccia grafica.

Per ottimizzare al meglio le tempistiche, abbiamo deciso di suddividere i compiti secondo i principali pacchetti di lavoro previsti inizialmente.
L'organizzazione dello sprint backlog è stata comunque generalmente rispettata, essendo stata ideata per permettere il lavoro parallelo e collaborativo sui differenti componenti del software.

4. Qualità del software

La qualità del prodotto software è stata testata progressivamente e valutata al termine di ogni ciclo di lavorazione (sprint).
In particolare, per quanto riguarda l'utilizzo da parte degli utenti finali, il prodotto finito deve garantire:

- correttezza, permettendo l'accesso ai dati dei pazienti secondo i vari criteri definiti dai requisiti funzionali, senza la ripetizione o l'omissione di dati utili;
- affidabilità, la garanzia che il software risponda entro tempistiche ammissibili (nell'ordine di pochi secondi nei peggiori dei casi) senza riscontrare perdite di dati forniti in input dagli utenti;
- efficienza, riferita all'uso delle risorse della macchina che ospita il software, oltre che a un efficace e rapido accesso alla base di dati;
- integrità, cioè la limitazione dell'accesso ai dati al solo personale registrato e la limitazione delle interazioni possibili con il sistema al solo personale competente in tale specifica attività;
- usabilità, che permetta a un qualsiasi utente autorizzato di operare sui dati in maniera intuitiva, senza la necessità di spiegazioni estese.

Rispetto alla manutenibilità del software, le aspettative di qualità definite sono le seguenti:

- manutenibilità, il codice dovrebbe essere abbastanza chiaro da poter individuare e risolvere eventuali difetti inattesi, senza la necessità di rivedere e riscrivere ampie sezioni del codice;
- testabilità, le operazioni di collaudo del corretto funzionamento del programma dovrebbero risultare rapide, grazie alla disponibilità di classi di test per quanto riguarda il database oppure attraverso l'interazione diretta con l'eseguibile;
- flessibilità, il programma dovrebbe presentare sufficiente flessibilità da permettere futuri miglioramenti di forma o aggiunta di ulteriori funzionalità, senza che questi vadano ad intaccare il corretto funzionamento della versione precedente;

Dato l'ambito di destinazione nel progetto, l'unico requisito che è stato ritenuto opportuno definire riguardo alla portabilità consiste nella necessità di garantire che il programma possa funzionare su macchine differenti da quelle in cui è stato sviluppato.

5. Ingegneria dei requisiti

I requisiti sono stati elicitati tramite intervista diretta con un possibile cliente, un operatore in campo ospedaliero, e attraverso l'analisi delle funzionalità del software ospedaliero attualmente utilizzato in alcune strutture del territorio.

Obiettivo:
Il software gestisce il percorso clinico del paziente dal suo arrivo alla sua dimissione.

Specifica dei requisiti funzionali:

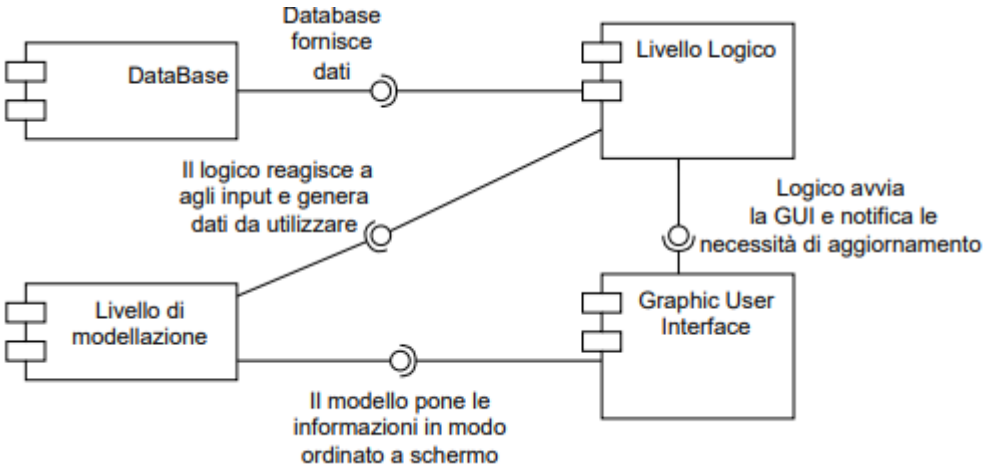
- Per accedere al software è richiesto il log-in del personale (Segreteria, Infermiere, Medico). Il log-in è presentato come una finestra in cui viene chiesto Id personale e password fornite dall'ospedale all'assunzione.
- All'arrivo del paziente viene richiesta la tessera sanitaria collegata al database statale dalla quale il personale di accoglienza (segreteria) inserirà a mano nel software i dati del paziente (Nome, Cognome). La segreteria inserisce anche il grado di urgenza (Verde, Giallo, Rosso).
- I medici assegnati al pronto soccorso possono visualizzare i pazienti in attesa della prima visita all'interno della sezione "in pronto soccorso". I medici possono anche osservare l'urgenza assegnata e prendere a carico i pazienti per ordine di urgenza.
- Durante la prima visita il medico curante compila la diaria medica, non eliminabile, nella quale inserisce il motivo del ricovero, lo storico del paziente, eventuali allergie, il reparto di destinazione, la lista dei farmaci e dei trattamenti che ritiene necessari. Il sistema terrà traccia automaticamente della data di inserimento e della matricola del medico che sta eseguendo l'operazione.
- La modifica della diaria medica da parte del medico è consentita solamente quando il paziente è stato assegnato a un letto in un reparto.
- Al completamento della prima diaria medica, il paziente viene automaticamente spostato nella sezione "da prendere in carico" visibile anche dal personale infermieristico.
- Un operatore o un membro della segreteria si occupa di assegnare al paziente un modulo e un posto letto in un reparto della struttura, in modo che corrisponda a quello consigliato all'interno della diaria medica; il paziente è quindi spostato automaticamente nella sezione "in reparto".
- Ogni reparto è composto da N moduli ognuno con 15 posti letto. Il paziente può essere spostato di posto letto, di modulo e di reparto.
- Il software aiuta gli infermieri consentendo di filtrare i pazienti per reparto. Nel proprio turno di lavoro gli infermieri selezionano un paziente dalla tabella "reparto" e, consultando le informazioni sulla cura prescritta, compilano la diaria inserendo il farmaco somministrato ed eventuali note riguardanti lo stato del paziente, con la possibilità di assegnare un flag di "importanza" alla nota inserita.
- L'infermiere, se richiesto dal medico, può aggiungere i dati delle rilevazioni effettuate sui pazienti (glicemia, temperatura, pressione, frequenza cardiaca, dolore). Inoltre, anche nella diaria infermieristica e nelle rilevazioni verrà automaticamente salvata la matricola dell'infermiere che le compila e l'orario di somministrazione del farmaco o delle rilevazioni.
- I medici osservano periodicamente lo storico delle diarie e delle rilevazioni per tenere traccia dell'andamento della cura, con la possibilità di modificare la diaria medica come sopra citato.
- Alla dimissione il paziente è rimosso dal database; nel caso in cui sia stato dimesso per essere posto in un altro reparto il paziente deve risostenere la registrazione e la visita medica in pronto soccorso.

6. Modellazione UML

La presentazione della modellazione è avvenuta attraverso l'impiego dei seguenti diagrammi UML:

- Casi d'uso
- Attività
- Classe
- Sequenza
- Macchina a stati (che evidenzia gli stati del paziente all'interno del database)
- Componenti
- Diagramma ER
- Package

Nell'esempio qui sotto, viene illustrata la modellazione del diagramma dei componenti, suddiviso in quattro parti per rappresentare l'architettura del software:



7. Architettura del software

La prospettiva iniziale consisteva nello strutturare il progetto utilizzando un'architettura a tre livelli: database, logico, interfaccia; in seguito, il team ha ritenuto più opportuno optare per un altro stile architettonico: una variante dello stile Model-View-Controller.

In questa architettura il ruolo di Controller viene eseguito dal pacchetto "logico", che si appoggia a dati e funzionalità offerti dal pacchetto di gestione del database.

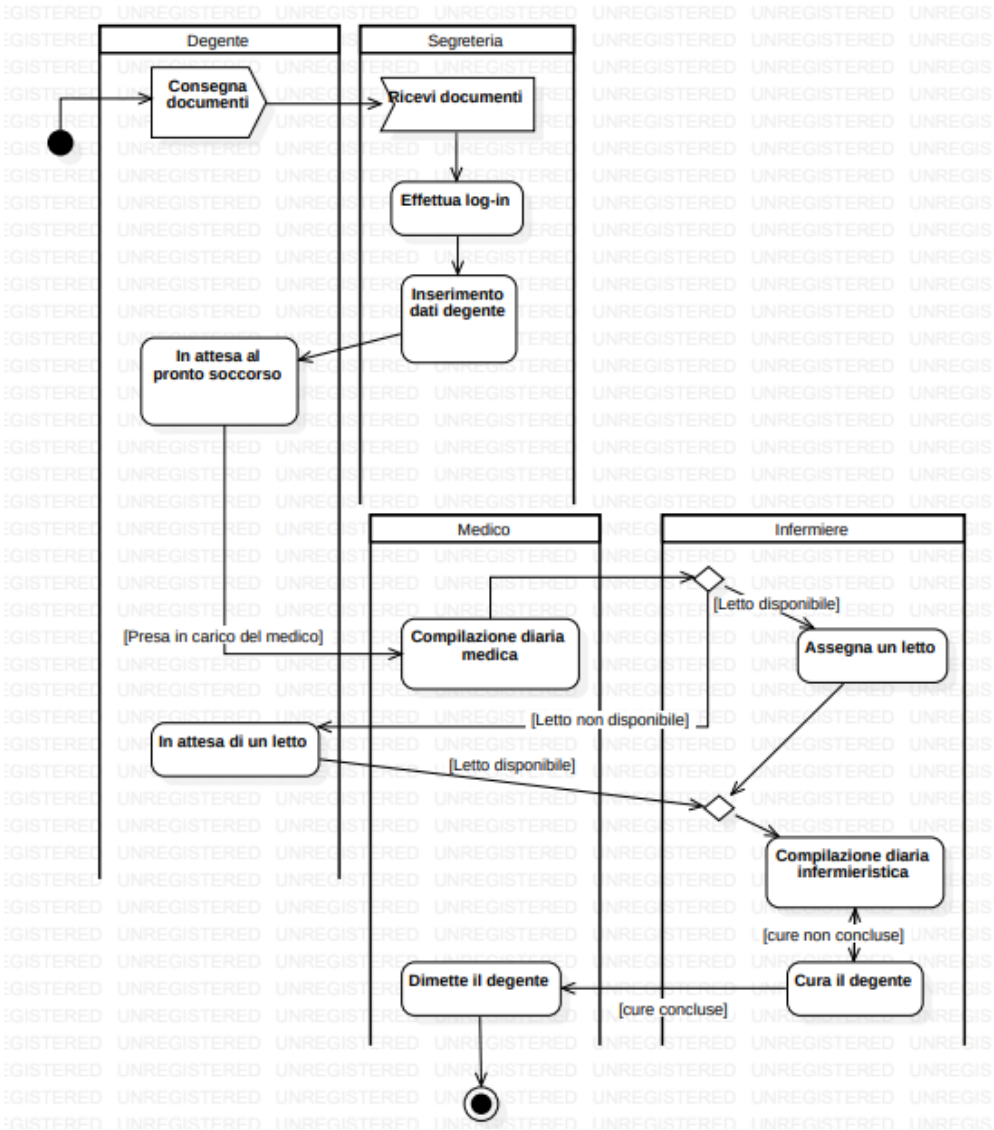
Il controller accetta le chiamate implicite provenienti dal livello View, identificato dal pacchetto "GUI", e attua dunque le necessarie manipolazioni sui dati utilizzando metodi JOOQ e/o i metodi forniti dalle classi del gestore del database (inserimento, rimozione e modifica).

I dati raccolti dal logico sono dunque caricati sul Model, rappresentato per l'appunto dal pacchetto "model", che permette la loro corretta visualizzazione sull'interfaccia utente.

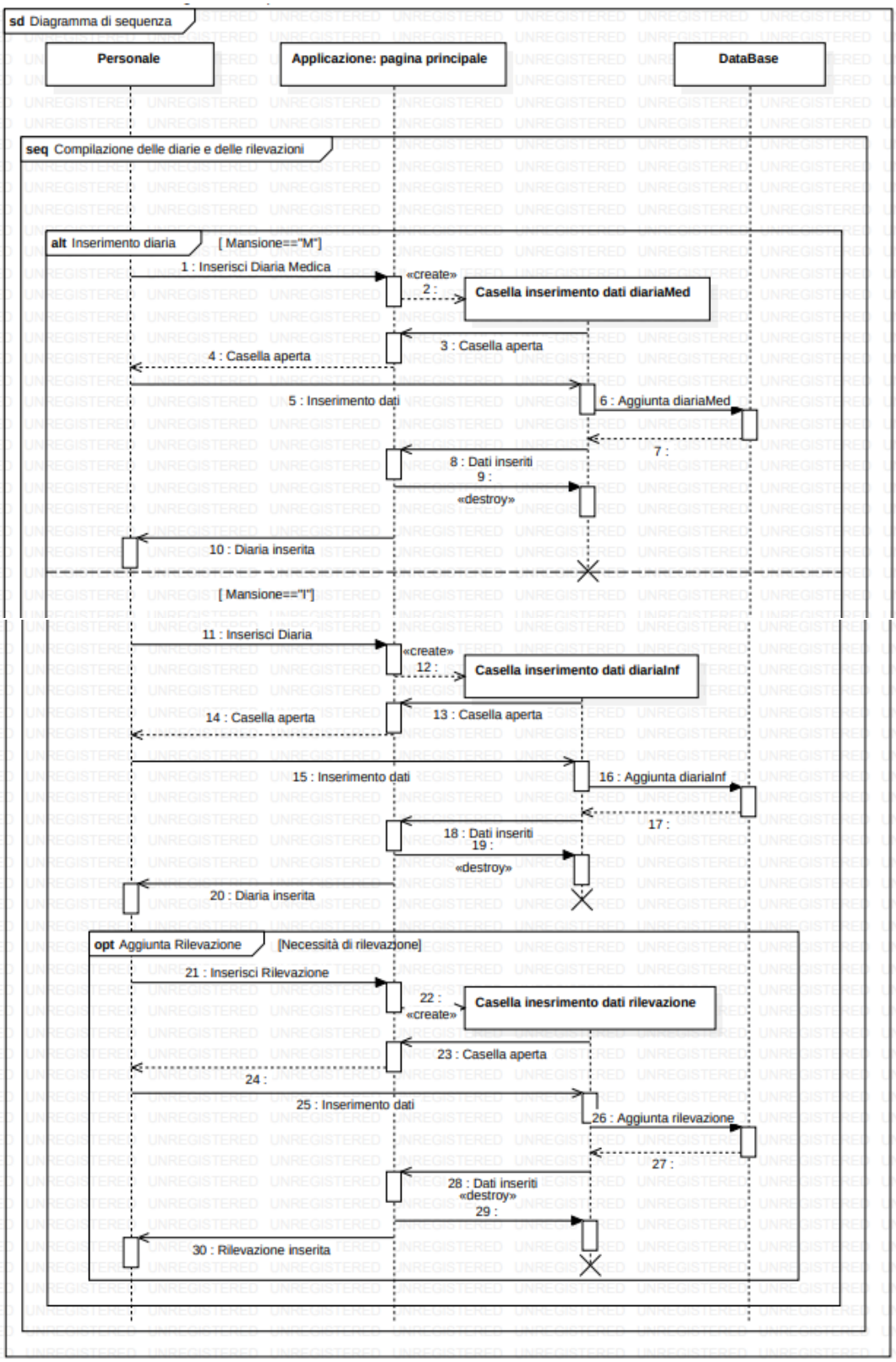
La chiamata di aggiornamento della View con i dati del Modello è eseguita dal Controller.

8. Design del software

La descrizione dei design del software è stata eseguita con UML, qui sotto è mostrato il diagramma delle attività, che rappresenta le attività svolte da chi utilizza il software:



Inoltre, qui mostrato il diagramma di sequenza per evidenziare la comunicazione tra il personale, il software e il database:


sd Diagramma di sequenza
Personale
Applicazione: pagina principale
DataBase
seq Compilazione delle diarie e delle rilevazioni
alt Inserimento diaria
[Mansione=="M"]
1 : Inserisci Diaria Medica
«create»
2 :
Casella inserimento dati diariaMed
3 : Casella aperta
4 : Casella aperta
5 : Inserimento dati
6 : Aggiunta diariaMed
7 :
8 : Dati inseriti
9 :
«destroy»
10 : Diaria inserita
[Mansione=="I"]
11 : Inserisci Diaria
«create»
12 :
Casella inserimento dati diariaInf
13 : Casella aperta
14 : Casella aperta
15 : Inserimento dati
16 : Aggiunta diariaInf
17 :
18 : Dati inseriti
19 :
«destroy»
20 : Diaria inserita
opt Aggiunta Rilevazione
[Necessità di rilevazione]
21 : Inserisci Rilevazione
22 :
«create»
Casella inesrimento dati rilevazione
23 : Casella aperta
24 :
25 : Inserimento dati
26 : Aggiunta rilevazione
27 :
28 : Dati inseriti
«destroy»
29 :
30 : Rilevazione inserita

8.1 Design Pattern

Nel corso dell'implementazione del codice, si è scelto di applicare il pattern singleton per la creazione progetto database, assicurando che la classe possa avere una sola istanza al suo interno. Questo risultato è ottenuto mediante l'uso di un'unica istanza privata, un costruttore privato e, infine, un metodo pubblico che consente di richiamare l'istanza quando necessario.

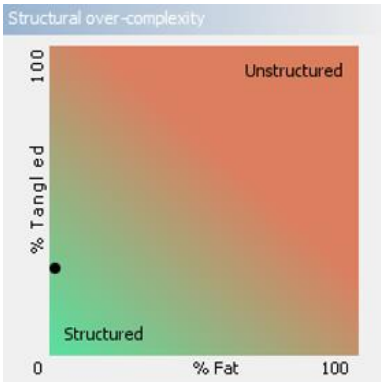
Un altro design pattern presente è l'abstraction occurrence utilizzato per le logiche dei bottoni, per le quali si è creata una superclasse astratta estesa da tutte le occorrenze di logica per un pulsante.

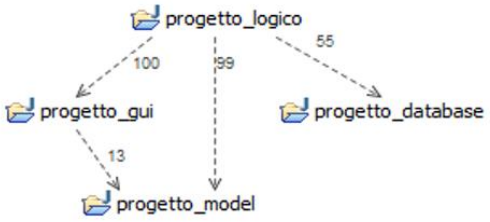
8.2 Metriche

Analizzando il codice con il tool Structure101 Studio si trovano i seguenti risultati:

- Percentuale di tangled (intreccio) bassa. Tangled indica quanto il codice è interconnesso e complesso, se è basso significa che i componenti sono chiaramente separati e con relazioni ben definite.
- Percentuale di fat (grasso) quasi inesistente. Fat indica la complessità dei singoli componenti del codice, se è basso significa che i componenti sono snelli e concentrati su singole responsabilità.

Dall'analisi risulta che il codice è abbastanza ben strutturato e con dipendenze chiare. Questo lo rende più facile da gestire, estendere, manutenere e correggere. Inoltre occorre specificare che il package più intrecciato è jooq.generated, quindi codice generato automaticamente per l'utilizzo di jooq.





Top tangles and fat (1)	#	Size	Problem
progetto_database.med_db.jooq.generated	19	7.929	Tangled

9. Testing del software

Il testing del software è stato effettuato in concomitanza con la scrittura del codice, al fine di assicurarsi che gli elementi integrati progressivamente nel programma operassero correttamente secondo le specifiche funzionali.

Per effettuare il collaudo sono stati adoperati principalmente due metodi:

- il primo consiste nella scrittura di classi di test JUnit; è stato utilizzato per valutare il corretto funzionamento dei metodi che il pacchetto "database" fornisce agli altri componenti del programma, in particolare i test hanno permesso di valutare il rispetto di vincoli d'integrità dei dati rispetto alle chiavi primarie e i vincoli d'integrità referenziale, ad esempio dimostrando la corretta eliminazione di dati di entità "deboli" alla rimozione di dati dalle

relative entità “forti” (esempio, alla dimissione di un degente, sono eliminate le diarie a lui assegnate ed è rimossa l’assegnazione del suo letto);
le classi di test scritte in questo modo coprono ognuna tra il 50 e il 90% del totale delle righe di codice delle rispettive classi, un coverage che è stato ritenuto soddisfacente dal team;

- il secondo consiste nella valutazione del funzionamento direttamente sull’eseguibile; questo metodo è stato adoperato per valutare il corretto funzionamento delle classi del pacchetto “logico”; utilizzando questo metodo è stato possibile accertarsi che ogni pulsante dell’interfaccia eseguisse le corrette ricerche nel database e caricasse i corretti dati all’interno dei modelli, senza omissioni o ripetizioni.
  Questo metodo è stato utile anche per valutare la corretta implementazione dei requisiti di sicurezza, che permettono l’accesso solo a utenti registrati e che limitano alcuni casi d’uso solamente a determinate figure professionali.

10. Manutenzione del software

La manutenzione del software è stata svolta effettuando delle operazioni di refactoring, le principali eseguite sono le seguenti:

- estrazione di una superclasse astratta;
- ridenominazione di campi, variabili, progetti e classi;
- aumento della qualità del codice con l’aggiunta di commenti JavaDoc.

Inoltre, essa è stata effettuata in maggior parte in fase di sviluppo andando ad attuare al meglio le esigenze del cliente e andando a ridurre il più possibile gli errori che si sono presentati.

In una possibile futura manutenzione del software si ipotizza un miglioramento della lunghezza complessiva del codice e un adattamento ai cambiamenti hardware e software più moderni.