Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Scienze di Internet

GESTIONE CARTELLA CLINICA PER UN MEDICO DI BASE

Tesi di Laurea in Programmazione di Internet

Relatore:
Chiar.mo Prof.
Antonio Messina

Presentata da:
Jacopo Tarantini

Sessione I
2009/2010

Indice

Elenco delle figure

Introduzione

L'obiettivo principale del progetto di tesi è quello di implementare un applicativo stand-alone in Java che interagisca con un Database cioè un archivio strutturato di dati che ne consente la loro gestione (inserimento, la ricerca, la cancellazione ed il loro aggiornamento). L'applicazione creata permette la gestione di una cartella clinica per medici di base. L'obiettivo quindi è quello di voler semplificare e soprattutto migliorare il lavoro che quotidianamente svolge un medico all'interno del proprio ambulatorio. I dati contenuti nel database sono i dati di tutti i pazienti, le visite effettuate, i farmaci prescritti, e le patologie avute. Il progetto realizzato è incentrato sull'utilizzo delle tecnologie Java e SQL, due linguaggi innovativi che permettono la realizzazione di qualsiasi tipo di applicazione. Java è un linguaggio orientato agli oggetti, giunto alla versione 6 che è una versione che introduce nuovi miglioramenti e funzioni al fine di offrire un'esperienza utente ottimizzata al consumatore finale. SQL(Structured Query Language)permette, tramite determinate istruzioni di "interrogare" il database in modo da poter leggere, modificare e gestire dati memorizzati in esso. L'utente, cioè il medico che utilizzerà il software, dopo essersi autenticato tramite username e password potrà iniziare ad eseguire diverse operazioni. Potrà effettuare l'inserimento di nuovi pazienti, effettuare la ricerca di quelli contenuti nel database, ed una volta ricercati potrà selezionare quello desiderato per poter visualizzare ed eventualmente modificare i dati anagrafici, oppure accedere alla cartella clinica dove sarà permessa la prescrizione di farmaci, inserire altre informazioni, allegare documenti, o poter visualizzare lo storico, cioè tutte le visite con

le relative informazioni del paziente selezionato precedentemente. Inoltre è presente una funzione che permette al medico titolare di poter abilitare l'accesso al sistema da parte di un collega nel caso in cui per vari motivi possa essere assente. Il seguente lavoro introduce nel primo capitolo una panoramica sull'importanza della cartella clinica, e nei capitoli 2 e 3 le specifiche e le attività di progetto. La progettazione del software è descritta nel capitolo 4. Infine, dopo una breve presentazione dei linguaggi utilizzati (capitolo 5), il capitolo 6 descriverà le funzionalità realizzate.

Capitolo 1

Contesto Generale

1.1 Cartella Clinica Cartacea

Nella maggior parte delle strutture sanitarie italiane, le informazioni vengono solitamente raccolte su documenti cartacei, quali ad esempio, referti, analisi, certificati, ecc. L'insieme di questi documenti costituisce la cartella clinica. L'utilizzo della cartella clinica è divenuto nel tempo di fondamentale importanza per il medico perchè contiene tutte le informazioni relative al singolo paziente sia dal punto di vista anamnestico che clinico, come ad esempio i dati che il medico rileva attraverso l'esame obiettivo. Frequentemente però, i dati sanitari sono poco standardizzati oppure sono strutturati in modo da facilitare la raccolta delle informazioni, ma non sono di facile e semplice lettura. La cartella clinica cartacea inoltre diviene sempre più voluminosa di documenti e risulta quindi sempre più difficile trovare tempestivamente le informazioni necessarie. Queste difficoltà possono essere superate automatizzando la gestione della cartella clinica attraverso l'uso appropriato di un sistema informativo, in modo tale che si possa disporre, in maniera efficiente e sicura, delle corrette informazioni necessarie ad un trattamento adeguato per il paziente.

1.2 Cartella Clinica Elettronica

La cartella clinica elettronica o informatizzata è diventata uno strumento indispensabile di lavoro per il medico di medicina generale per migliorare le sue possibilità assistenziali. La cartella clinica in medicina generale è differente da quella che si utilizza in ambito ospedaliero perchè diverso è il tempo e il campo di utilizzo. Mentre in ospedale devono essere raccolte tutte le informazioni che giustifichino il raggiungimento di una determinata diagnosi, in medicina generale deve essere utilizzata una cartella clinica per problemi specifici, a causa dei quali è stata richiesta la visita, e deve essere possibile distinguere fra problemi attivi, per i quali deve ancora essere trovata una soluzione, e problemi inattivi, ovvero già risolti. La cartella clinica in Italia non è considerata di proprietà della ASL, come in altri Paesi (per esempio la Gran Bretagna), ma del medico curante; questo ha ovviamente facoltà di trasmetterla a specialisti, sostituti, medici ospedalieri coinvolti nella cura del paziente. Solitamente non viene lasciata al paziente per evitare dimenticanze, smarrimenti, incongrue consultazioni mentre è buona norma lasciare al paziente gli originali degli esami e delle visite specialistiche affinchè possano servire in caso di emergenza come fonte di dati per altri medici. Una cartella medica orientata per problemi particolarmente utile è quella ideata da L.L. Weed nel 1969 per la formazione degli studenti di medicina americani, e ispirata a principi di praticità, comprensione, completezza e onestà (dovendo annotare le situazioni che hanno portato a una determinata decisione) e adattabilità alla computerizzazione, nonostante quest'ultima non sia considerata indispensabile. Questo tipo di cartella clinica è cosi suddivisa in:

- Dati di base
- Lista dei problemi attivi e inattivi
- Diario clinico, compilato tenendo conto di dati generali oggettivi (polso, pressione arteriosa, obiettività particolari), dati soggettivi (sintomi), ipotesi diagnostiche con relative strategie diagnostiche e terapeutiche.

- Allegati eventuali per il monitoraggio di disturbi cronici, per attività preventive, per riportare esami strumentali e di laboratorio, visite specialistiche o altro.

Nelle informazioni di base vengono annotati i dati anagrafici del paziente. La lista dei problemi riassume la situazione clinica del paziente mentre il diario clinico è un'ipotesi di lavoro in cui vengono annotati i sintomi riferiti dal paziente, l'esame obiettivo, le ipotesi diagnostiche e le procedure diagnostiche o terapeutiche. Gli allegati sono utili per monitorare determinate pato-logie croniche e ricordarsi dei controlli da effettuare per quella data patologia. Tale tipo di cartella presuppone notevoli capacità sintetiche e talora viene considerata poco flessibile. E' importante comunque utilizzare un linguaggio semplice per evitare fraintendimenti fra diversi medici e facilitare la comunicazione indiretta attraverso la cartella.

1.3 L'ausilio del computer o la vecchia carta

E' errato pensare che la ricerca epidemiologica e clinica debbano necessariamente trarre vantaggio dall'utilizzo dell'informatica. Tutto sta nell'impostazione della ricerca e dei suoi obiettivi; solo in questo caso l'ausilio del computer diventa fondamentale, perchè abbrevia di molto i tempi di lavoro. Oggi i medici che usufruiscono di un computer sono circa il 50 per cento contro il 3-10 per cento del 1986.[1] Solo il 10 per cento utilizza però tale ausilio per la gestione della cartella clinica. I programmi di gestione informatizzati disponibili sono in grado di fornire oltre all'archivio della cartella clinica, la gestione dei dati anamnestici, la possibilità di effettuare celermente ricette e certificati. Bisogna tener presente che non sempre i pazienti gradiscono l'utilizzo del computer durante la visita, bisognerebbe quindi cercare di provvedere all'inserimento dei dati e all'aggiornamento dei problemi clinici, nonchè alla revisione della cartella in momenti differenti dalla visita, nonostante questo richieda ulteriori tempi di lavoro. E' però importante abituare il paziente a questo tipo di cambiamenti, facendogli capire tutti i vantaggi

che si possono avere. E' importante dire che, a meno che non si dispone di uno scanner in grado di inserire automaticamente nel computer i vari referti o relazioni, è utile disporre di un archivio cartaceo da affiancare al computer per poter conservare fotocopie od originali di documenti importanti.

Capitolo 2

Specifiche di progetto

2.1 Analisi dei requisiti

L'obiettivo del software che si vuole realizzare è quello di automatizzare e informatizzare il lavoro che quotidianamente svolge il medico di base.
L'archivio conterrà tutte le informazioni dei singoli pazienti a carico del medico che utilizzerà il software. La cartella clinica è uno strumento fondamentale per il medico. Deve contenere una nitida fotografia dell'attuale stato di salute del paziente. Deve contenere il nome di qualsiasi farmaco prescritto con specificato il dosaggio e la via di somministrazione. Deve contenere nominativamente tutti gli esami di laboratorio e tutti gli esami strumentali richiesti. Devono essere presenti tutte le visite effettuate con le rispettive date, i relativi farmaci assegnati, analisi effettuate, certificati rilasciati. Ad ogni visita effettuata, il medico deve annotare nella cartella i sintomi riferiti dal paziente, le ipotesi diagnostiche. Tutte queste informazioni verranno inserite in un opportuno campo note. L'accesso al sistema da parte del medico deve essere autenticato con username e password. Quando il medico non c'è deve delegare ad un altro medico, quest'ultimo deve poter accedere al sistema. Il medico di base può, utilizzando una determinata funzionalità disabilitare o abilitare l'accesso al sistema da parte di altri medici che in alcuni periodi potrebbero sostituirlo. Nel Database deve essere possibile ef-

fettuare l'inserimento di un nuovo paziente; la cancellazione; la ricerca; la modifica di qualche campo. I dati del paziente che verranno gestiti sono:

- Codice Fiscale
- Cognome
- Nome
- Data di nascita
- Luogo di nascita
- Indirizzo
- Residenza
- Occupazione
- Numero di telefono
- Malattia
- Cura prescritta
- Note

Quando vengono effettuate delle analisi, o altri esami di laboratorio da parte del paziente, il medico deve poter avere i risultati, li può ricevere, tramite allegato ad una email o in modo cartaceo. In entrambi i casi, il documento digitale ricevuto potrà essere inserito all'interno del sistema tramite una procedura di upload dei file, i formati che saranno supportati sono: .jpg, .gif, .png, .bmp, .doc, .docx, .xls, .pdf.
Il software Gestione Cartella Clinica Medico di Base prevede le seguenti funzionalità:

- Inserimento di un paziente
- Ricerca di un paziente

- Visualizzazione dei dati anagrafici del paziente
- Modifica dei dati anagrafici del paziente
- Compilazione della Cartella Clinica
- Stampa della ricetta in formato pdf
- Visualizzazione dello storico di tutte le visite
- Upload e download di file in diversi formati
- Abilitazione di un medico sostituto

L'analisi dei requisiti (talvolta detta semplicemente analisi) rappresenta una delle prime fasi nel ciclo di vita di un prodotto software; scopo generale dell'analisi è stabilire che cosa il sistema in questione deve fare (mentre le decisioni sul come sono rimandate alla successiva fase di progettazione). A questo punto è importante definire i requisiti funzionali e quelli non funzionali.

Requisiti funzionali

- Connessione al database
- Predisposizione per il login
- L'utente può effettuare l'inserimento di un nuovo paziente
- L'utente può effettuare una ricerca di uno o più pazienti
- L'utente può visualizzare e modificare i dati anagrafici
- L'utente può creare una cartella clinica del paziente
- L'utente può visualizzazione lo storico di tutte le visite
- L'utente può stampare le ricette
- L'utente può abilitare l'accesso di un medico sostituto per un determinato periodo di tempo

- L'utente può caricare o scaricare eventuali allegati

Requisiti non funzionali

- Software sviluppato in Java
- Utilizzo di un Database
- Software di tipo stand-alone

Capitolo 3

Attività di progetto

3.1 Piano di processo

Il piano di processo è una fase fondamentale per la buona realizzazione di un software, in quanto specifica le principali regole che il processo dovrà seguire per riuscire a raggiungere l'obiettivo finale. Più specificatamente in questo capitolo viene descritta la scelta del modello, la rappresentazione della Working Breakdown Structure e il diagramma di Gantt che è utile per mostrare le attività svolte nell'arco temporale. Il modello di processo software che si è scelto di utilizzare per lo sviluppo del progetto è il RUP (Rational Unified Process). Si è scelto questo modello perchè si è ritenuto il più adeguato per la progettazione del software perchè è un modello di tipo iterativo e incrementale, permette una valutazione migliore dei rischi e una correzione immediata, risultando così un modello molto flessibile. Il RUP è progettato e documentato tramite l'utilizzo di UML (Unied Modeling Language) e viene rilasciato online e aggiornato circa due volte all'anno per non renderlo obsoleto e per far sì che coloro che lo utilizzano ne possano trarre il massimo vantaggio. La scelta di utilizzare il RUP è inoltre data dal fatto che questo modello è specificatamente object-oriented e quindi si adatta perfettamente allo sviluppo di questa applicazione. Come detto in precedenza, una delle caratteristiche principali del modello è il fatto di essere

iterativo, questo significa che il processo è organizzato in diverse fasi, ognuna delle quali viene ripetuta più volte. Ogni fase produce un incremento e permette di raffinare in modo graduale le analisi del processo e di valutare lo stato di avanzamento ed evoluzione del progetto. Gli eventuali errori solitamente vengono rilevati durante la fase di integrazione, ma grazie al metodo iterativo si può vedere rapidamente se i rischi percepiti sono davvero tali, e si possono individuare nelle fasi iniziali quando ancora sono più facili e meno costosi da correggere. In pratica il processo di sviluppo può essere migliorato efficacemente lungo il percorso. Il suo ciclo di vita è suddiviso in quattro fasi: Inception, Elaboration, Construction, Transition.

1. Inception (concezione): analisi dei requisiti, valutazione dei rischi e del budget a disposizione. Studio della fattibilità dell'idea. Lo scopo principale è quello di comprendere il tipo di mercato al quale il progetto si riferisce ed eventualmente identicare gli elementi che possono condurre a un successo commerciale. Fra gli strumenti utilizzati vi sono un modello dei casi d'uso, la pianicazione iniziale del progetto, la valutazione dei rischi e una definizione iniziale dei requisiti;

2. Elaboration (elaborazione): inizio della progettazione, predisposizione di una adeguata base architetturale e sviluppo di un piano per la realizzazione del progetto. Questa fase comprende l'analisi di dominio e una prima fase di progettazione dell'architettura e deve concludersi con il superamento di una milestone conosciuta come Lifecycle Architecture Milestone. Se il progetto non passa questa milestone, potrebbe ancora essere abbandonato, oppure dovrà essere rivisitato. Al termine di questa fase si transita infatti in una situazione di rischio più elevato, in cui le modiche all'impostazione del progetto saranno più difficili e dannose;

3. Construction (costruzione): fase in cui si realizza una versione del sistema, sviluppo incrementale del prodotto e rilascio della relativa documentazione

4. Transition (transizione): il sistema diventa operativo e può essere validato dagli utenti. Prima però bisogna vericare che il prodotto sia conforme alle aspettative descritte nella fase iniziale. Se questo non è accaduto si ripeterà il ciclo dall'inizio altrimenti si raggiunge il termine dello sviluppo. Viene inoltre scritta la documentazione, ovvero il manuale utente.

Diagrammi di supporto alla gestione del progetto

L'utilizzo dei diagrammi nel progetto è molto importante per facilitare la programmazione del lavoro e quindi per ottenere risultati migliori. I diagrammi realizzati per le stime sono:

- GANTT: Stima temporale
- WBS: Stima delle attività

Diagramma di Gantt

[2]Il Diagramma di Gantt è uno strumento che permette di modellizzare la pianificazione dei compiti necessari alla realizzazione di un progetto. Si tratta di uno strumento inventato nel 1971 da Henry L. Gantt. Il diagramma di Gantt permette di definire un calendario di tutte le singole attività previste dal processo di sviluppo formalizzato. Nel diagramma ogni compito è rappresentato con una linea, mentre le colonne rappresentano i giorni, le settimane o i mesi del calendario secondo la durata del progetto. Il tempo stimato per un'azione è modellizzato su una barra orizzontale la cui estremità sinistra è posizionata sulla data d'inizio prevista e l'estremità destra sulla data prevista per la fine della realizzazione. I compiti possono susseguirsi in sequenza oppure essere eseguiti in parallelo. Tutto ciò è utile al fine di pianificare, coordinare e tracciare specifiche attività in un progetto dando una chiara illustrazione dello stato d'avanzamento di quello rappresentato. Di seguito è riportato il diagramma di Gantt che mostra l'effettivo tempo impiegato per la realizzazione del progetto.

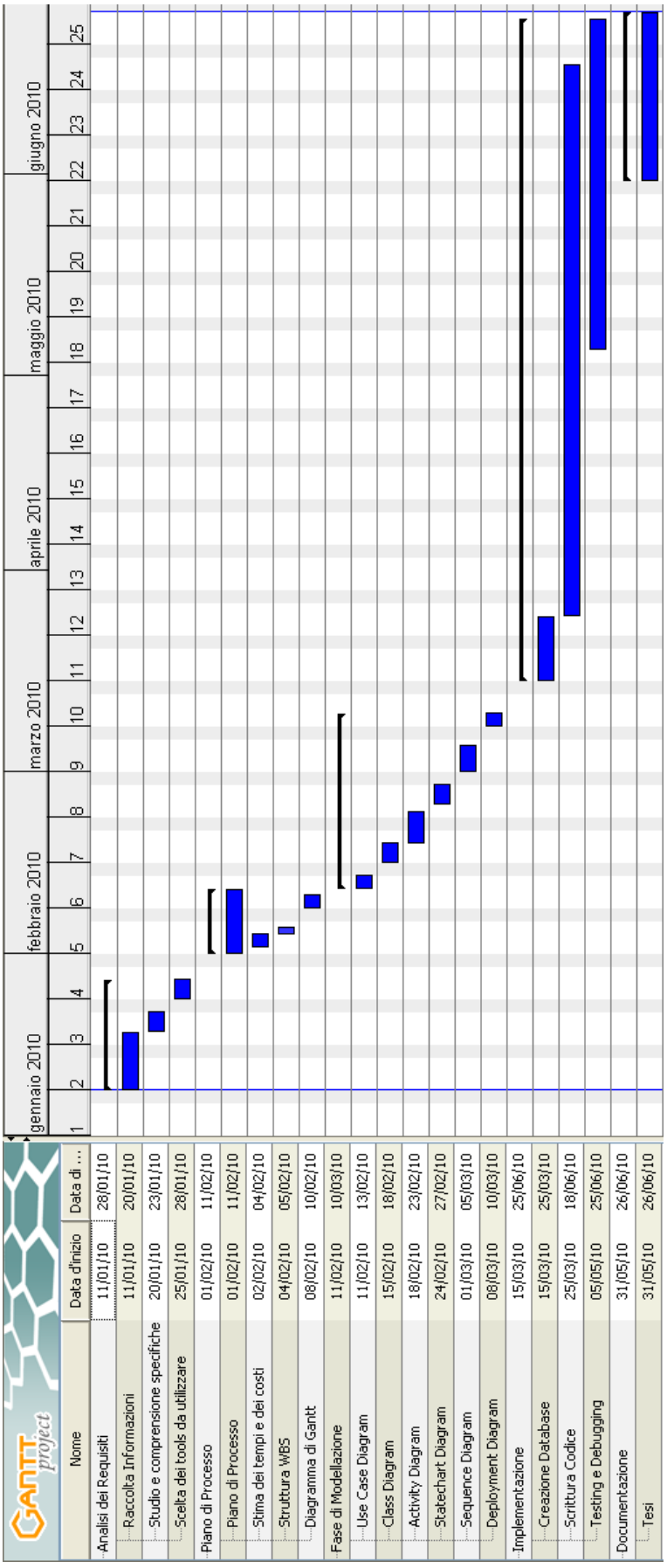


Figura 3.1: Diagramma di Gantt

Work Breakdown Structure

Il WBS (Work Breakdown Structure), è un diagramma ad albero che permette di visualizzare tutte le attività di un progetto che viene suddiviso solitamente nel materiale, nel software, nei servizi, nei dati e nelle attrezzature che lo compongono. L'albero gerarchico al suo interno ha tutte le attività ed ognuna è suddivisa in diversi livelli che descrivono in maniera sempre più dettagliata i compiti necessari per lo sviluppo del progetto.

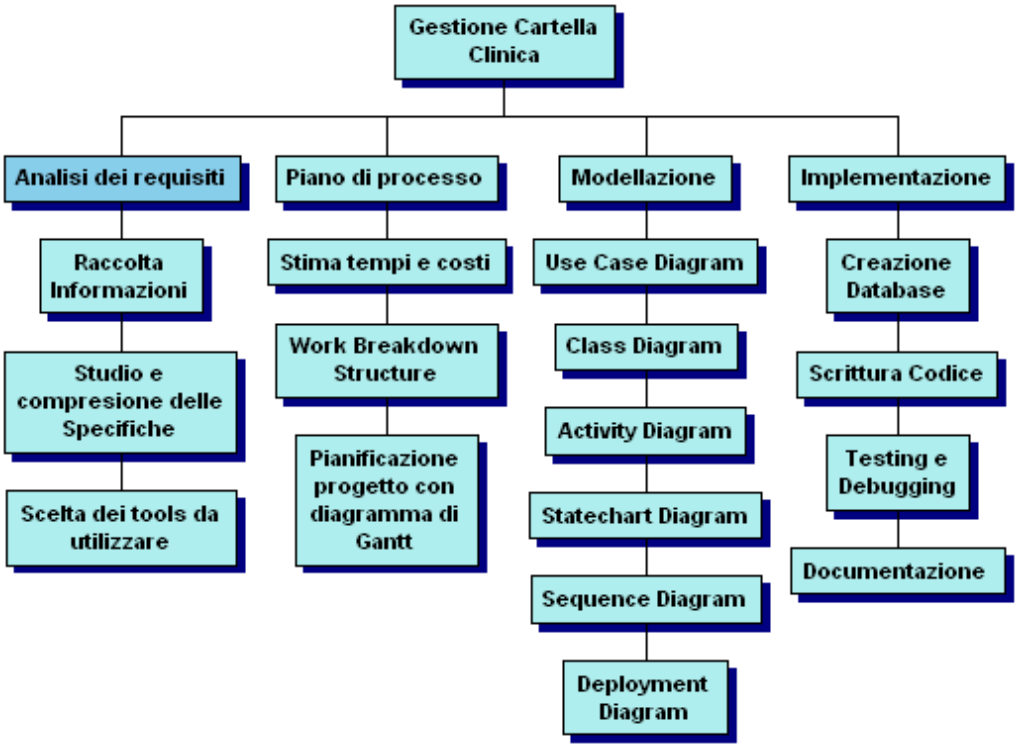


Figura 3.2: Diagramma WBS

Capitolo 4

Progettazione del Software

4.1 Diagrammi UML

UML (Unified Modeling Language) è un linguaggio di modellazione basato sul paradigma object-oriented[3]. Il nucleo del linguaggio fu definito nel 1996 dai "tre amigos", Grady Booch, Jim Rumbaugh e Ivar Jacobson sotto tutela della OMG (Object Management Group), un consorzio fondato con l'obiettivo di creare e gestire standard nel contesto dello sviluppo del software a oggetti. OMG ancora oggi gestisce lo standard di UML. UML consente di costruire modelli object-oriented per rappresentare domini di diverso genere. Nel contesto dell'ingegneria del software, viene usato soprattutto per descrivere il dominio applicativo di un sistema software, il comportamento e la struttura del sistema stesso. Il modello è strutturato secondo un insieme di viste che rappresentano diversi aspetti della realtà modellata (funzionamento, struttura, comportamento, ecc.), sia a scopo di analisi che di progetto. Di per sé, UML è solo un linguaggio di modellazione, e non definisce una specifica metodologica per la creazione di modelli (o alcun processo software). Può quindi essere utilizzato nel contesto di diversi approcci metodologici. La OMG gestisce uno standard metodologico, correlato a UML ma proposto come specifica indipendente, detto RUP (Rational Unified Process). Molto spesso UML viene utilizzato non solo per la modellazione di sistemi

software, ma anche per descrivere domini di altro genere, come sistemi hardware, strutture organizzative aziendali, processi di business. Il punto di forza di questo linguaggio di modellazione consiste nel fatto che il piano di processo di produzione può essere progettato in modo che committenti, clienti, tecnici, analisti e programmatori, possano esaminare con cura e studiare in modo esaustivo il sistema. I tre aspetti principali che permettono all'UML di descrivere un sistema sono il modello funzionale, il modello ad oggetti e il modello dinamico. Tutti utilizzano un insieme di tipi di diagrammi specifici che possono essere messi in relazione tra loro. Modello Funzionale rappresenta il sistema dal punto di vista dell'utente, ovvero ne descrive il suo comportamento così come esso è percepito all'esterno, a prescindere dal suo funzionamento interno. Questo tipo di modellazione corrisponde, in ingegneria del software, all'analisi dei requisiti. La modellazione funzionale utilizza i diagrammi dei casi d'uso (Use case Diagram). Modello ad oggetti rappresenta la struttura e sottostruttura del sistema utilizzando i concetti object-oriented di classe, oggetto, le relazioni fra classi e fra oggetti. In ingegneria del software, questo tipo di modellazione può essere utilizzata sia nella fase di analisi del dominio che nelle varie fasi di progetto a diversi livelli di dettaglio. Questo tipo di modellazione utilizza i diagrammi di classe (class diagram), diagrammi degli oggetti (object diagram) e diagrammi di deployment. Modello dinamico rappresenta il comportamento degli oggetti del sistema, ovvero la loro evoluzione nel tempo e le dinamiche delle loro interazioni. E' strettamente legato al modello ad oggetti e viene impiegato negli stessi casi. Vengono utilizzati i diagrammi di sequenza (sequence diagram), diagrammi delle attività (activity diagram) ed i diagrammi di stato (statechart diagram). Grazie all'utilizzo di questi diagrammi il sistema software viene progettato in maniera professionale prima dell'implementazione. Essendo il progetto antecedente la scrittura del codice, si potrà conoscere in anticipo il risultato finale, prevedere eventuali errori e si avrà una maggiore facilità nella scrittura; da non sottovalutare è anche la chiara visione che i diagrammi forniscono a livello di sfruttamento delle risorse hardware in

termini di memoria ed efficienza.

- **Modello Funzionale:** rappresenta il sistema dal punto di vista dell'utente, ovvero ne descrive il suo comportamento così come esso è percepito all'esterno, a prescindere dal suo funzionamento interno. Questo tipo di modellazione corrisponde, in ingegneria del software, all'analisi dei requisiti. La modellazione funzionale utilizza i diagrammi dei casi d' Uso (Use Case Diagram).

- **Modello ad Oggetti:** rappresenta la struttura e sottostruttura del sistema utilizzando i concetti object-oriented di classe, oggetto, le relazioni fra classi e fra oggetti. In ingegneria del software, questo tipo di modellazione può essere utilizzata sia nella fase di analisi del dominio che nelle varie fasi di progetto a diversi livelli di dettaglio. Questo tipo di modellazione utilizza i diagrammi di classe (Class Diagram), diagrammi degli oggetti (Object Diagram) e diagrammi di deployment.

- **Modello Dinamico:** rappresenta il comportamento degli oggetti del sistema, ovvero la loro evoluzione nel tempo e le dinamiche delle loro interazioni. E' strettamente legato al modello ad oggetti e viene impiegato negli stessi casi. Vengono utilizzati i diagrammi di sequenza (sequence diagram), diagrammi delle attività (Activity Diagram) ed i diagrammi di stato (Statechart Diagram).

Grazie all'utilizzo di questi diagrammi il sistema software viene progettato in maniera professionale prima dell'implementazione.

4.1.1 Use Case Diagram

I diagrammi dei casi d'uso sono diagrammi utilizzati per rappresentare le funzioni o i servizi offerti da un sistema.[4] Solitamente sono i primi diagrammi ad essere disegnati durante il processo di sviluppo, successivamente all'analisi dei requisiti. Più precisamente si può dire che questi diagrammi possono essere considerati come uno strumento di rappresentazione dei requisiti funzionali di un sistema. Gli elementi principali che sono utilizzati nei diagrammi dei casi d'uso sono tre: lo scenario, gli attori e il caso d'uso. Lo scenario viene rappresentato con un rettangolo, dove al suo interno verranno messi i model element che rappresentano caratteristiche del sistema, mentre quelli che rappresentano entità esterne (apparteneti al dominio o al contesto del sistema) verranno posizionati all'esterno. Gli attori sono rappresentati graficamente nel diagramma da un'icona che raffigura un uomo stilizzato. Praticamente l'attore è colui che interagirà con il sistema. Un caso d'uso è raffigurato tramite un'ellisse contenente il nome del caso d'uso in particolare rappresenta una funzione o un servizio offerto dal sistema a uno o più attori. Di seguito è illustrato il diagramma dei casi d'uso del sistema realizzato "Gestione Cartella Clinica" tenendo presente che include si riferisce ad un comportamento necessario mentre extend ad un comportamento opzionale.

Diagramma Caso d' Uso Gestione Cratella Clinica

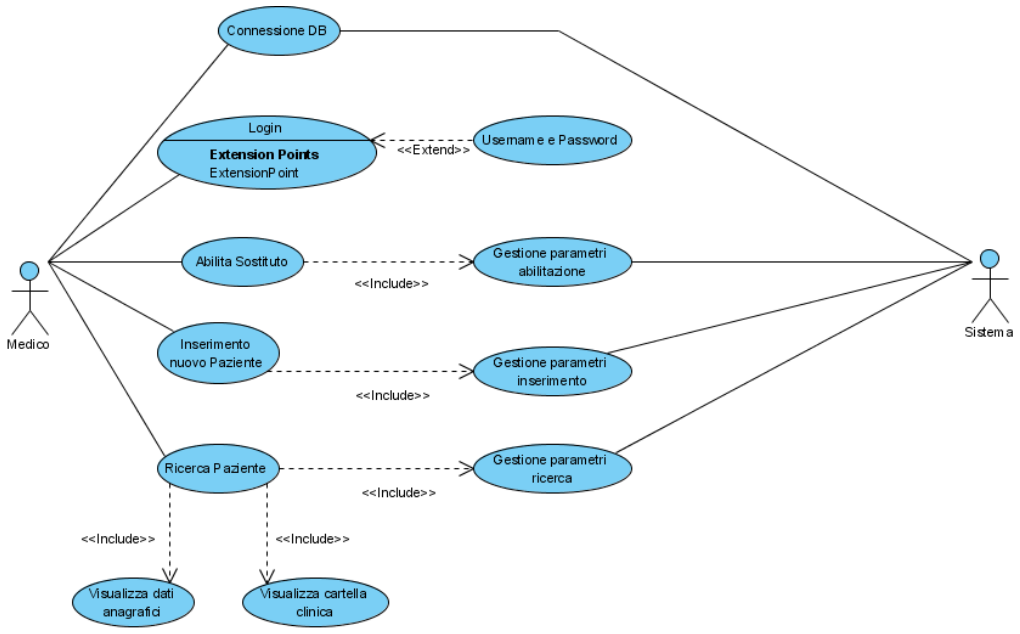


Figura 4.1: Diagramma Caso d' Uso Gestione Cartella Clinica

Il diagramma del caso d'uso rappresentato mette in evidenza tutte le funzionalità con cui il medico potrà interagire, e come il sistema gestirà le varie situazioni. In particolare il medico potrà effettuare il login inserendo la username e la password; potrà abilitare un sostituto nel caso in cui non ci sarà per un determinato periodo di tempo; potrà effettuare l'inserimento di un nuovo paziente; potrà effettuare la ricerca dei pazienti all'interno dell'archivio; successivamente alla ricerca potrà visualizzare i dati anagrafici o la cartella clinica del paziente. Il sistema gestirà tutte le funzioni attraverso una serie di controlli.

Scenari

Gli scenari dei casi d'uso descrivono sequenze di passi che caratterizzano una particolare iterazione tra l'utente e il sistema. Generalmente è una sequenza specifica di azioni e iterazioni tra il sistema e alcuni attori. Uno scenario descrive una particolare storia nell'uso del sistema, o un percorso attraverso il caso d'uso. Poiché all'interno di un caso d'uso ci possono essere diversi scenari, vengono descritti per spiegare le differenti iterazioni che ci possono essere fra più casi d'uso. In particolare vengono descritti i casi d'uso principali: Login, Inserimento nuovo paziente, Ricerca Paziente, Cartella Clinica

Scenario Caso d'Uso: Login
ID: UC1
Attori: Utente, Sistema
Precodizioni: Connessione al database
Sequenza degli eventi: 1. Inserimento username e password 2. Il sistema verifica le credenziali 2.1 Accesso avvenuto 3. Se le credenziali sono errate 3.1 l' utente deve reinserirle
Postcondizione: Vengono attivate tutte le funzionalità del software

Scenario Caso d'Uso:Inserimento nuovo paziente
ID: UC2
Attori: Utente, Sistema
Precodizioni: L'utente deve aver effettuato l'accesso
Sequenza degli eventi: 1. l'utente inserisce i dati 2. il sistema controlla che siano stati inseriti almeno i dati obbligatori 3. l'utente è avvisato se l'inserimento è andato a buon fine o meno
Postcondizione: Nessuna

Scenario Caso d'Uso: Ricerca Paziente
ID: UC3
Attori: Utente, Sistema
Precodizioni: L'utente deve aver effettuato l'accesso
Sequenza degli eventi: 1. L'utente inserisce i parametri di ricerca 1.1 Il sistema visualizza i risultati 2. Se nessun risulato viene visualizzato 2.1 l'utente deve cambiare i parametri di ricerca
Postcondizione: Possibile accesso alla cartella clinica o ai dati anagrafici

Scenario Caso d'Uso: Cartella Clinica
ID: UC4
Attori: Utente, Sistema
Precodizioni: Aver effettuato la ricerca
Sequenza degli eventi: 1. L'utente inserisce tutti i dati che riguardano la visita 2. L'utente può effettuare il salvataggio nel database 3. L'utente può stampare la ricetta 4. L'utente può visualizzare lo storico di tutte le visite effettuate 5. L'utente può effettuare l'upload o il download di eventuali allegati
Postcondizione: Nessuna

Scenario Caso d'Uso: Abilita sostituto
ID: UC5
Attori: Utente, Sistema
Precodizioni: L'utente ha effettuato l'accesso
Sequenza degli eventi: 1. L'utente sceglie tramite una lista il medico che lo dovrà sostituire 2. L'utente inserisce le date del relativo periodo di sostituzione 3. Il sistema controlla che le date inserite siano corrette
Postcondizione: Il sostituto puo effettuare l'accesso in quel determinato periodo

4.1.2 Class Diagram

I diagrammi delle classi consentono di descrivere tipi di entità, con le loro caratteristiche, e le eventuali relazioni fra questi tipi.[4] Questo tipo di diagramma appartiene alla categoria di quelli che vengono definiti di struttura. Con il concetto di classe si vuole rappresentare in modo diretto e intuitivo la realtà, in qualsiasi ambito. UML prevede un loro impiego a livello di analisi e in particolare analisi del dominio, ma anche a livello di progettazione per descrivere la struttura interna del sistema, dei suoi componenti e delle loro relazioni. In questo diagramma una classe rappresenta una categoria di entità (istanze); il nome della classe indica la categoria di entità descritta della classe. Ogni classe ha al suo interno un insieme di attributi e operazioni. Gli attributi descrivono le caratteristiche degli oggetti della classe, mentre le operazioni descrivono il comportamento. Graficamente le classi UML sono rappresentate da un rettangolo suddiviso in tre parti, dove nella prima parte in alto viene inserito il suo nome, in quello al centro vengono inseriti gli attributi e nell'ultimo è dedicato alle operazioni. Due classi possono essere legate da associazioni che rappresentano i collegamenti che ci possono essere tra gli oggetti delle classi associate. Ci sono 4 tipi principali di associazione (aggregazione, composizione, dipendenza, generalizzazione) ed ognuna è rappresentata da una particolare freccia che connette le classi coinvolte. Per potere arrivare alla definizione del diagramma si deve partire conoscendo già i casi d'uso di un sistema, le specifiche dei requisiti, le sequenze degli eventi e il glossario con i termini del progetto. Successivamente all'analisi di questi documenti si possono definire le classi e le loro responsabilità. Di seguito è mostrato il diagramma delle classi del software "Gestione Cartella Clinica".

Diagramma delle Classi Gestione Cartella Clinica

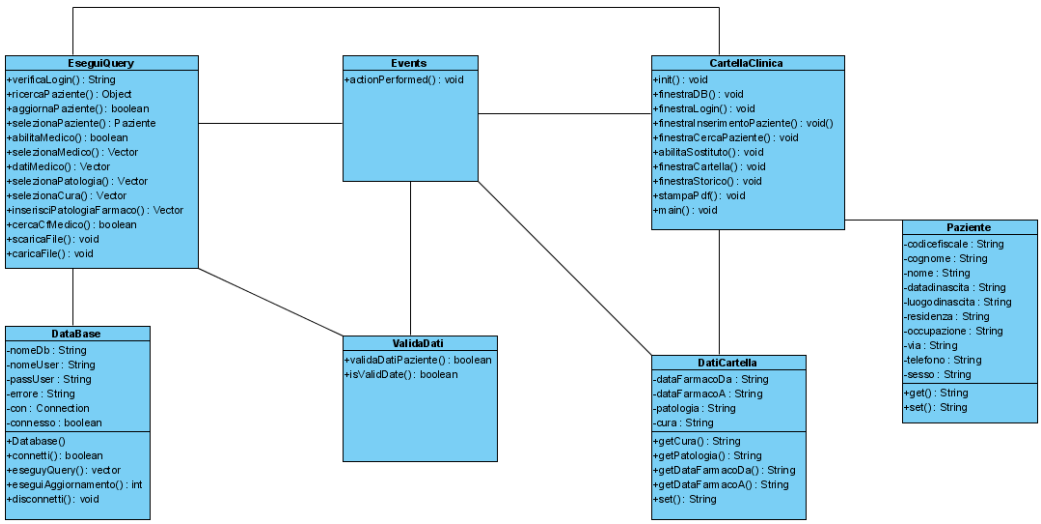


Figura 4.2: Diagramma delle Classi Gestione Cartella Clinica

Nel diagramma delle classi sono rappresentate le sette classi principali che formano il software, in particolare la classe CartellaClinica conterrà il metodo main() e tutti i metodi che comporranno le varie interfacce grafiche con cui interagirà l'utente, più alcuni metodi di funzionalità. La classe EseguiQuery, conterrà tutti i metodi riguardanti le query che serviranno per interagire, aggiornare il database.
La classe DataBase gestirà la connessione con la base di dati e l'esecuzione delle varie query.
La classe Events conterrà un metodo dove al suo interno ci saranno tutte le azioni che avverranno cliccando con il mouse su un pulsante di ogni interfaccia grafica. La classe Paziente contiene tutti i dati dei pazienti.
La classe DatiCartella contiene tutti i dati che serviranno per compilare ogni cartella clinica. La classe ValidaDati serve per controllare la correttezza di alcuni dati che verranno inseriti.

4.1.3 Statechart Diagram

Lo Statechart Diagram è un diagramma previsto dall'UML per descrivere il comportamento di entità o di classi in termini di stato (macchina a stati).[4] Il diagramma mostra gli stati che sono assunti dall'entità o dalla classe in risposta ad eventi esterni. Il concetto di stato è spesso posto in relazione al ciclo di vita; l'insieme completo di stati, che un'entità o una classe può assumere, dallo stato iniziale a quello finale, ne rappresenta il ciclo di vita. Ogni transizione è una relazione tra due stati che indica che quando si verifica un evento, l'oggetto passa dallo stato precedente allo stato successivo. E' rappresentata con una freccia etichettata con il rispettivo evento. Un evento è un avvenimento significativo o degno di nota. Uno stato è la condizione di un oggetto in un certo intervallo di tempo. Nel diagramma è rappresentato tramite un rettangolo con angoli arrotondati.

Diagramma di Stato Gestione Cartella Clinica

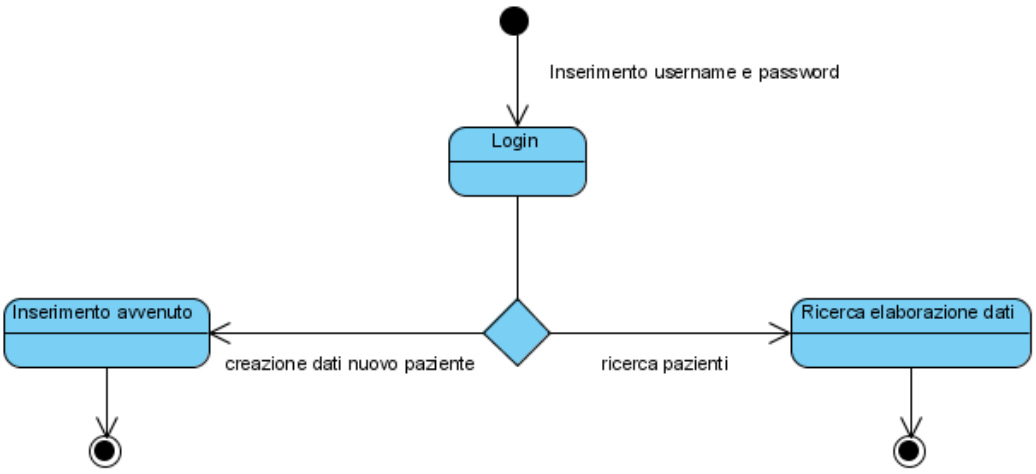


Figura 4.3: Diagramma di Stato Gestione Cartella Clinica

4.1.4 Activity Diagram

Un diagramma delle attività (activity diagram) definisce le attività da svolgere per realizzare una data funzionalità.[4] Un diagramma delle attività è tipicamente associato ad una o più classi UML. In alcuni casi può essere associato ad un metodo. Questi diagrammi sono utili per rappresentare comportamenti sequenziali, concorrenza, sistemi distribuiti, business workflow. Permettono di rappresentare processi paralleli e la loro sincronizzazione, il flusso è rappresentato tramite delle frecce orientate, che indicano la sequenza temporale con cui devono essere effettuate le diverse attività. Tramite un simbolo viene indicato l'inizio del flusso ed un altro ne indica il termine. Alcune attività possono essere svolte in parallelo ed in questo caso il punto di divisione chiamato fork è rappresentato da frecce divergenti rispetto al segmento. Il punto di ricongiungimento detto join è rappresentato tramite un segmento su cui le frecce si ricongiungono. Nel caso in cui le attività siano alternative, cioè il loro svolgimento o meno dipende da una scelta, il punto di decisione è rappresentato da dei rombi da cui partono i flussi alternativi. Di seguito sono sono rappresentati quattro diagrammi di attività i quali rappresentano le azioni principali che verranno fatte dall'utente nell'interazione con il sistema.

Diagramma di Attività Login

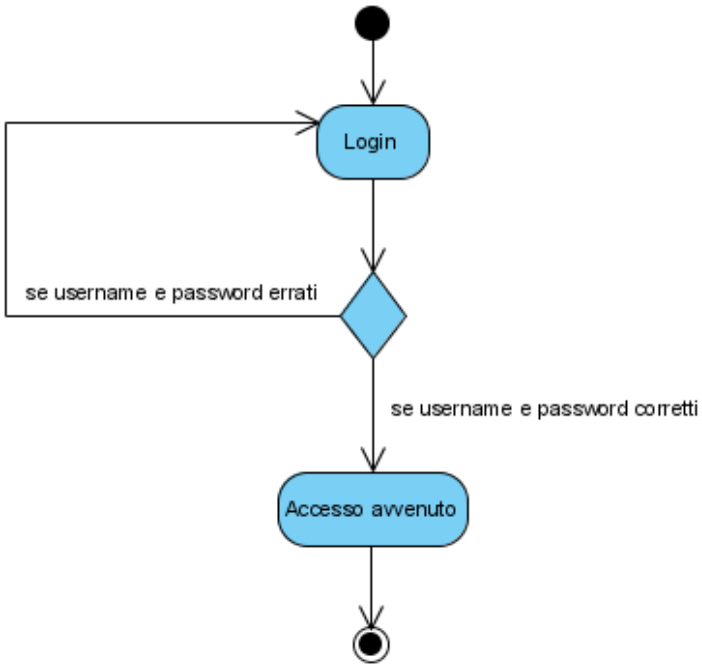


Figura 4.4: Diagramma di Attività Login

In questo diagramma viene rappresentata l'azione del login, l'utente inserirà le credenziali, a questo punto il sistema, effettuerà un controllo, se la username e/o la password sono errate, l'utente dovrà reinserirle di nuovo. Se invece risulteranno corrette, avverrà l'accesso.

Diagramma di Attività Inserimento Paziente

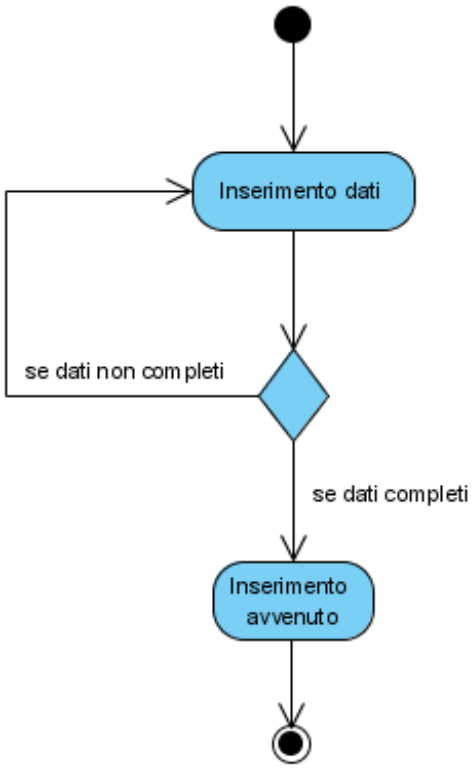


Figura 4.5: Diagramma di Attività Inserimento Paziente

In questo diagramma viene mostrato come avverrà l'inserimento dei dati di un nuovo paziente, anche in questo caso il sistema effettuerà un controllo, e attraverso un decision node avverà un controllo, se i dati non saranno completi l'utente dovrà reinserirli o completarli, se invece risulteranno completi, l'inserimento sarà avvenuto, e il sistema lo comunicherà attraverso un messaggio.

Diagramma di Attività Ricerca Paziente

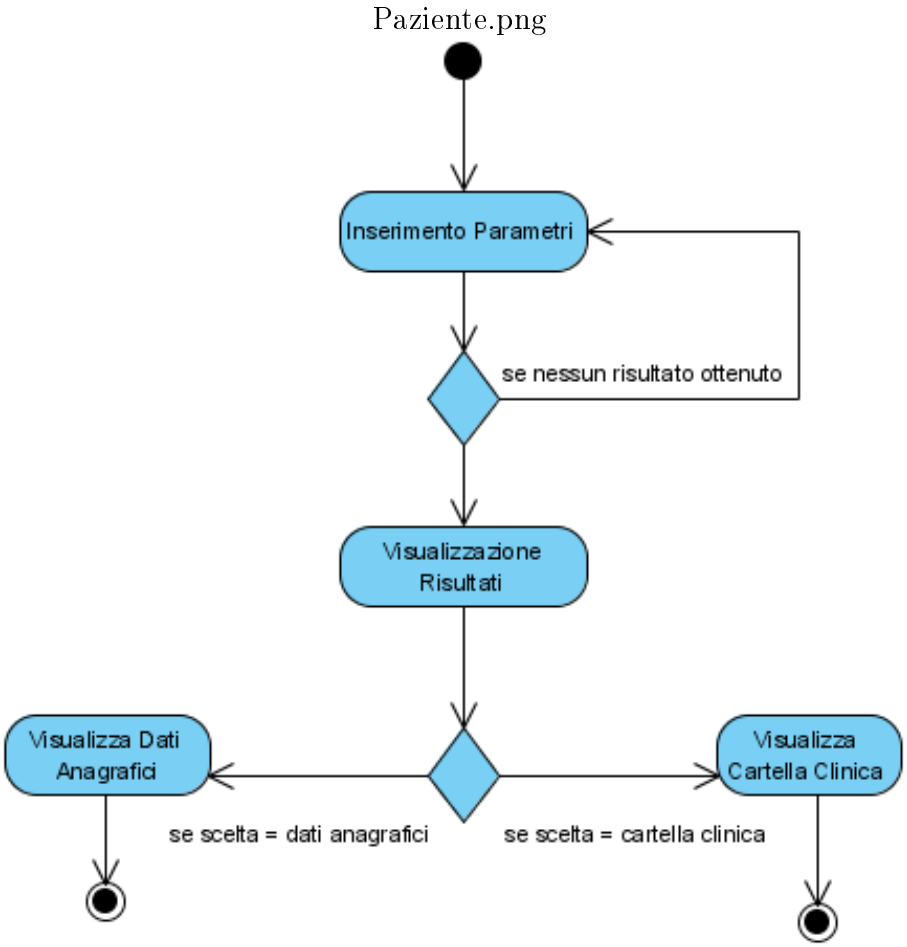


Figura 4.6: Diagramma di Attività Ricerca Paziente

Con questo diagramma si rappresenta come avviene l'azione di ricerca di uno o più pazienti all'interno del database. Per prima cosa verranno inseriti i parametri, se questi parametri non produrranno nessun risultato, si potrà riprovare di nuovo inserendo parametri diversi. Se invece la ricerca avrà prodotto dei risultati, a questo punto si potrà scegliere se visualizzare i dati anagrafici o la cartella clinica del relativo paziente.

Diagramma di Attività Cartella Clinica

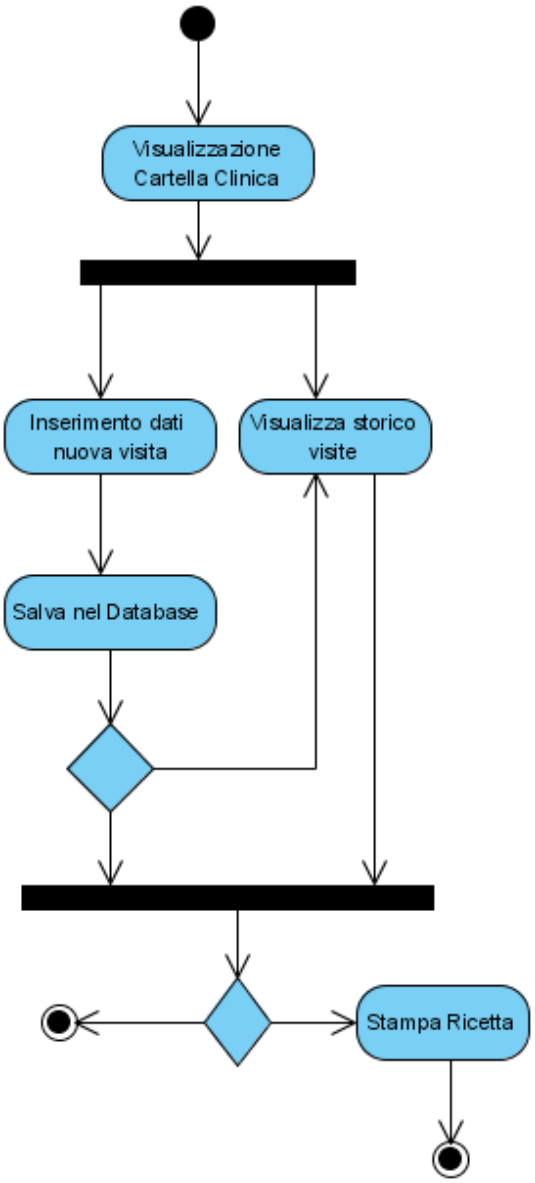


Figura 4.7: Diagramma di AttivitàCartella Clinica

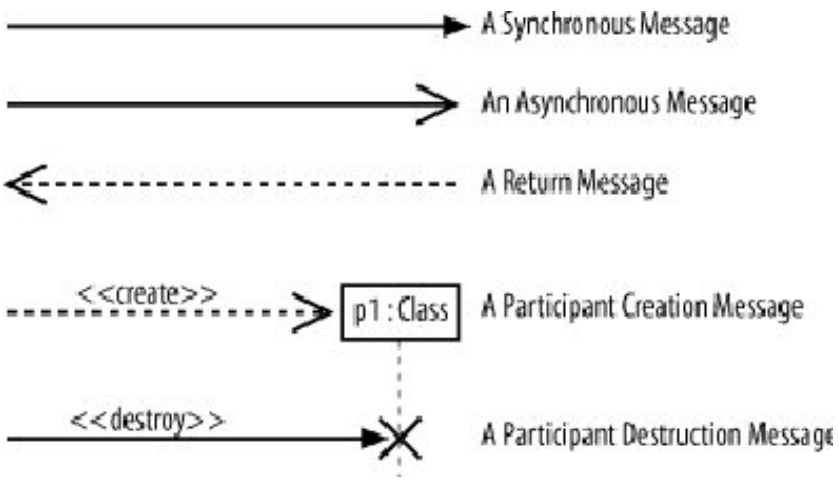
Questo diagramma mostra cosa avverrà quando l'utente si troverà all'interno della cartella clinica di un paziente. Se dovrà effettuare una nuova visita inserirà i dati relativi a quest'ultima e poi potrà salvare all'interno del database.

Se invece non deve effettuare una visita, può visualizzare lo storico di tutte le visite effettuate dal paziente in questione. Infine se vorrà potrà anche stampare la ricetta.

4.1.5 Sequence Diagram

Il diagramma di sequenza è un altro tipo di diagramma previsto dall'UML.[4] Questo tipo di diagramma, permette di modellare la comunicazione tra un oggetto ed un altro, in relazione al trascorrere del tempo. Si capisce subito che in questo tipo di rappresentazione un ruolo cruciale viene svolto appunto dal tempo. L'idea chiave è che le interazioni tra gli oggetti avvengano seguendo un ordine ben preciso e che tale sequenza avvenga, nel tempo, dall'inizio alla fine. Il Sequence diagram è costituito da oggetti rappresentati nel modo ormai usuale, come rettangoli recanti un nome (con il nome sottolineato), messaggi rappresentati da linee continue recanti una freccia alla loro fine e il tempo rappresentato come progressione verticale. Un messaggio che viaggia da un oggetto ad un altro, viene disegnato a partire dalla lifeline dell'oggetto da cui parte il messaggio e arriva sulla lifeline dell'oggetto a cui il messaggio è diretto. Si può anche verificare il caso in cui un oggetto mandi un messaggio a se stesso, cioè un messaggio che parte dalla sua lifeline e arriva alla stessa lifeline. Tale tipo di comportamento viene definito ricorsione. Nella tabella seguente vengono identificati tutti i possibili messaggi definiti in UML per i sequence diagrams:

Synchronous	Se un oggetto invia un messaggio sincrono, allora si attende che gli venga restituita una risposta al messaggio stesso prima di poter continuare con altre operazioni.
Asynchronous	Diversamente dai messaggi sincroni, se un oggetto invia un messaggio asincrono, non attende che gli venga inviata alcuna risposta prima di continuare con altre operazioni.


A Synchronous Message
An Asynchronous Message
A Return Message
<<create>>
p1 : Class
A Participant Creation Message
<<destroy>>
A Participant Destruction Message

Diagrammi di Sequenza Inserimento Paziente

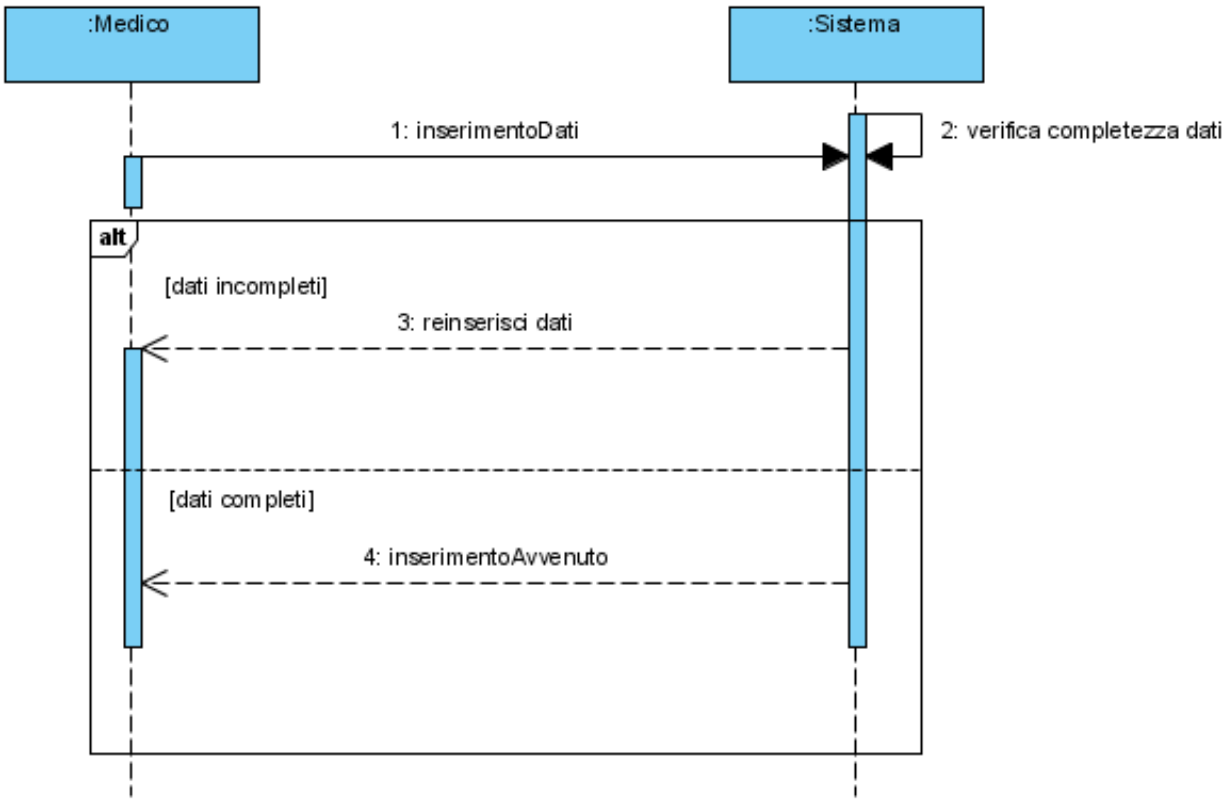


Figura 4.8: Diagramma di Sequenza InserimentoPaziente

In questo diagramma viene rappresentato in particolare l'andamento sequenziale di come avviene l'inserimento di un paziente. Come si può vedere il medico inserisce tutti i dati e li manda al sistema, a questo punto viene effettuato un controllo sulla completezza dei dati. In seguito attraverso la funzione di "alt", vengono analizzati due casi. Nel caso in cui i dati risultassero incompleti, il sistema manderà un messaggio al medico con scritto di reinserire i dati o di completarli. Nel secondo caso, cioè se i dati saranno completi, il sistema comunicherà che l'inserimento è avvenuto con successo.

Diagrammi di Sequenza Ricerca Paziente

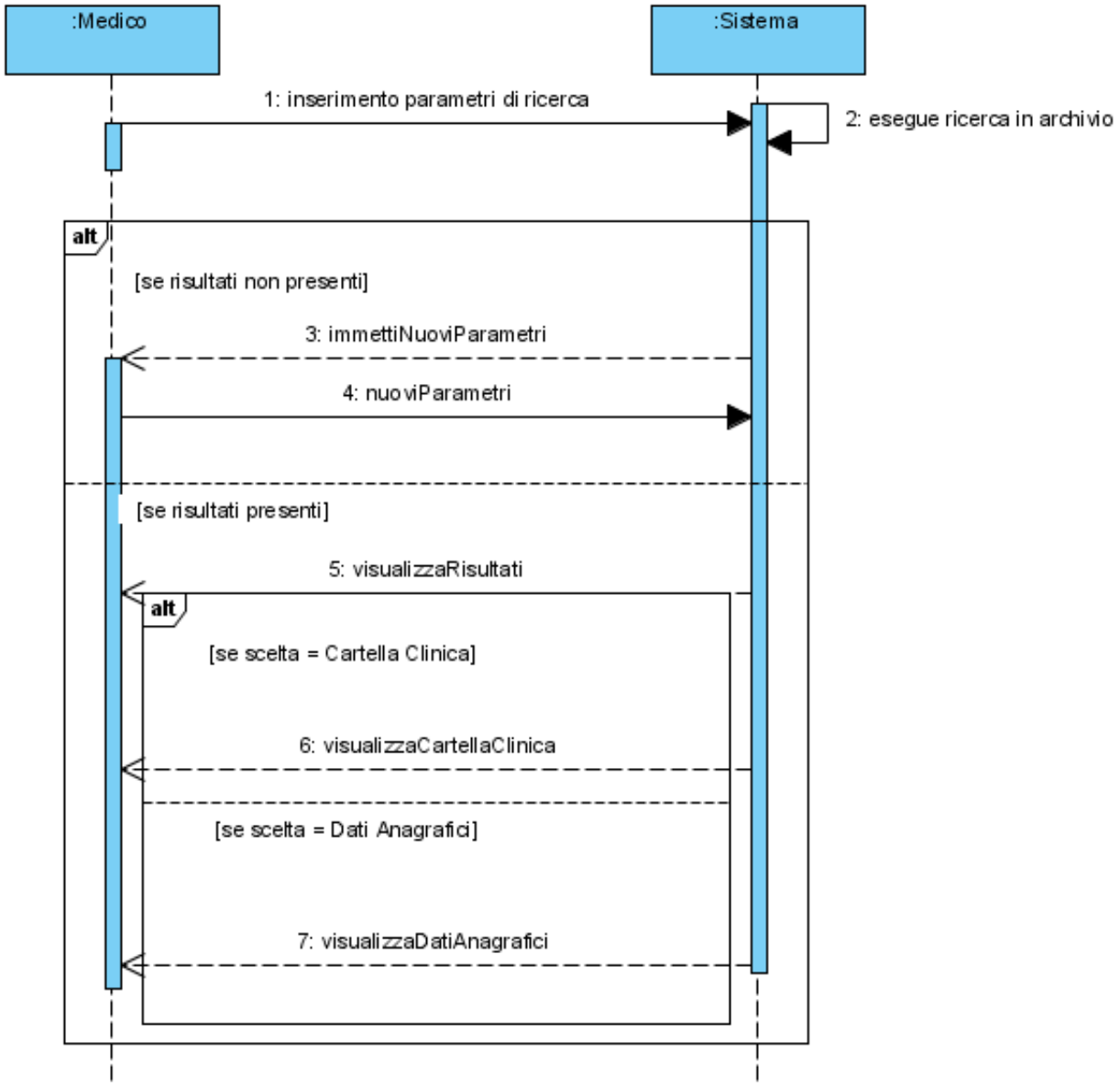


Figura 4.9: Diagramma di Sequenza RicercaPaziente

In questo diagramma viene rappresentato in particolare l'andamento sequenziale di come avviene la ricerca dei pazienti all'interno del database. Il medico

manderà al sistema i parametri di ricerca. Il sistema effettuerà la ricerca in archivio, a questo punto attraverso l'uso di un "alt" vengono analizzati i due casi che si possono presentare. Se la ricerca non ha prodotto alcun risultato, il sistema dirà al medico di immettere nuovi parametri. Se invece la ricerca avrà prodotto dei risultati, il sistema li visualizzerà, e a questo punto ci sarà un altro "alt" per gestire altri due casi. Nel caso in cui il medico deciderà di visualizzare i dati anagrafici lo richiederà al sistema e quest'ultimo li visualizzerà. Mentre l'altro caso è quando il medico decide di entrare all'interno del pannello della cartella clinica.

4.1.6 Deployment Diagram

[4]Il diagramma del deployment mostra la distribuzione degli elementi software all' architettura fisica e la comunicazione tra gli elementi fisici. Gli elementi principali del diagramma si chiamano nodi e sono collegati da path di comunicazione.

Diagramma di Deployment Gestione Cartella Clinica

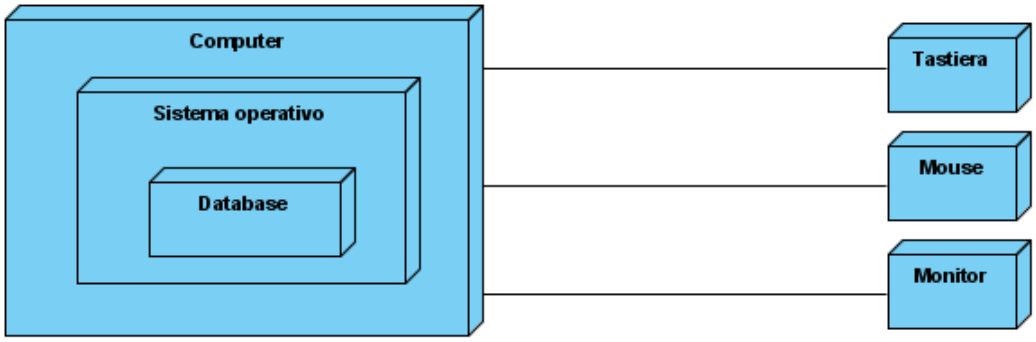


Figura 4.10: Diagramma di Deployment Gestione Cartella Clinica

4.2 Progettazione Concettuale

In questa fase viene studiato l'insieme dei dati del sistema informativo; per formalizzare tale insieme vengono identificate e definite le entità e le relazioni coinvolte. Per scegliere i dati da rappresentare si è adottato il seguente metodo: sono stati presi in considerazione solo i dati realmente necessari allo sviluppo dell'applicazione, escludendo alcuni dati specifici, che comunque potrebbero essere aggiunti facilmente in future espansioni del sistema.

4.2.1 Entità e relazioni

Le entità che sono state analizzate per la realizzazione del software sono: paziente, medico, patologia, farmaco, visita e patologia_farmaco.
Ogni paziente è identificato da un codice fiscale univoco, inoltre devono essere disponibili il nome, il cognome, la data di nascita, la via, la residenza, l'occupazione, il numero di telefono e il sesso. Ad ogni paziente è associata una o più visite e il medico che le sostiene. Il medico è colui che utilizzerà il software e anch'esso è identificato da dati anagrafici, ma inoltre avrà degli attributi per la username, la password, due date che identificheranno il periodo di accesso al sistema, un attributo attivo, e uno responsabile per indicare se è il titolare di utilizzo del software o se è un possibile sostituto. La patologia è identificata da un id, e ha come attributi il nome, la causa e gli effetti. Il farmaco è anch'esso identificato da un id e ha come attributi il nome e il principio attivo. La visita ha un'identificativo della visita, la data, il codice fiscale del paziente, il codice fiscale del medico, un campo note e un allegato. Patologia_farmaco è una tabella che viene identificata da due date (da, a) che servono per indicare la durata di prescrizione del farmaco, il codice fiscale del paziente che sostiene la visita, la patologia, il farmaco e l'idvisita. [5]Di seguito viene tiportato lo schema E-R (Entity-Relations) e tutte le tabelle di ciascuna entità analizzata precedentemente.

Schema E-R

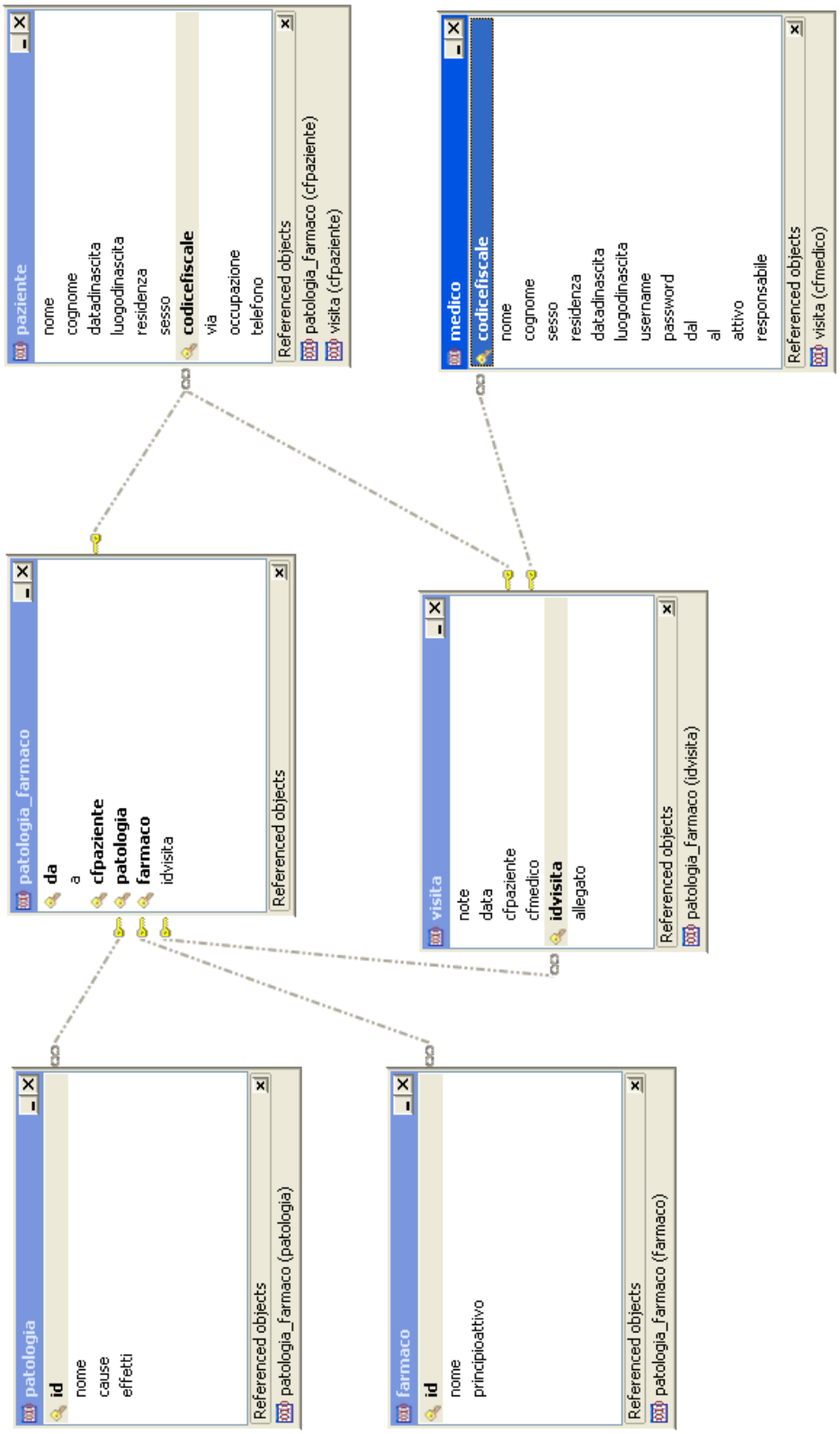


Figura 4.11: Schema E-R del software Gestione Cartella Clinica

Tabella Entità paziente

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
nome	varchar(30)	latin1_swedish_ci	NO				select,insert,update,references	
cognome	varchar(30)	latin1_swedish_ci	NO				select,insert,update,references	
datadinascita	date	{null}	YES		{null}		select,insert,update,references	
luogodinascita	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
residenza	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
sesso	char(1)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
codicefiscale	varchar(16)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
via	varchar(70)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
occupazione	varchar(20)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
telefono	varchar(20)	latin1_swedish_ci	NO				select,insert,update,references	

Figura 4.12: Tabella dell'entità paziente

Tabella Entità medico

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
codicefiscale	varchar(16)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
nome	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
cognome	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
sesso	char(1)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
residenza	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
datadinascita	date	{null}	YES		{null}		select,insert,update,references	
luogodinascita	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
username	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
password	varchar(30)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
dal	date	{null}	YES		{null}		select,insert,update,references	
al	date	{null}	YES		{null}		select,insert,update,references	
attivo	int(1)	{null}	YES		{null}		select,insert,update,references	
responsabile	int(1)	{null}	YES		{null}		select,insert,update,references	

Figura 4.13: Tabella dell'entità medico

Tabella Entità visita

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
note	varchar(4000)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
data	varchar(20)	latin1_swedish_ci	NO				select,insert,update,references	
cfpaziente	varchar(16)	latin1_swedish_ci	NO				select,insert,update,references	
cfmedico	varchar(16)	latin1_swedish_ci	NO				select,insert,update,references	
idvisita	int(9)	{null}	NO	PRI	{null}	auto_increment	select,insert,update,references	
allegato	longblob	{null}	YES		{null}		select,insert,update,references	

Figura 4.14: Tabella dell'entità visita

Tabella Entità patologia

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
id	int(11)	{null}	NO	PRI			select,insert,update,references	
nome	varchar(100)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
cause	varchar(100)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
effetti	varchar(100)	latin1_swedish_ci	YES		{null}		select,insert,update,references	

Figura 4.15: Tabella dell'entità patologia

Tabella Entità farmaco

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
id	int(11)	{null}	NO	PRI			select,insert,update,references	
nome	varchar(100)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
principioattivo	varchar(45)	latin1_swedish_ci	YES		{null}		select,insert,update,references	

Figura 4.16: Tabella dell'entità farmaco

Tabella Entità patologia_cura

Field *	Type *	Collation	Null *	Key *	Default	Extra *	Privileges *	Comment *
da	varchar(45)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
a	varchar(45)	latin1_swedish_ci	YES		{null}		select,insert,update,references	
cfpaziente	varchar(45)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
patologia	varchar(45)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
farmaco	varchar(45)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
idvisita	int(9)	{null}	YES		{null}		select,insert,update,references	

Figura 4.17: Tabella dell'entità patologia_cura

Capitolo 5

Tecnologie utilizzate

In questo capitolo vengono descritte le principali tecnologie utilizzate per la realizzazione del software, in particolare il linguaggio Java, SQL, ed il DBMS MySQL.

5.1 Java

Questo paragrafo è dedicato alla principale tecnologia utilizzata per il progetto di tesi ovvero la tecnologia Java.[6] Questo linguaggio ha diverse caratteristiche fondamentali, ma la principale è quella di essere un linguaggio orientato agli oggetti. L'idea alla base della OOP è di rappresentare, nella progettazione del software, le entità reali o astratte che compongono il problema sotto forma di oggetti istanziati da classi. Gli oggetti sono caratterizzati da proprietà(definite variabili o campi di istanza o di esemplare) e da metodi o funzioni applicabili sugli oggetti stessi, che possono ad esempio modificarne lo stato o estrarne informazioni. I programmi scritti in Java possono essere unicamente orientati agli oggetti, di conseguenza tutto il codice deve essere necessariamente incluso in una classe. Un'altra caratteristica importante del linguaggio è quella di avere una piattaforma indipendente, cioè che il compilatore Java non produce un codice oggetto nativo per una determinata piattaforma, ma piuttosto delle istruzioni byte code da usare con

la JVM (Java Virtual Machine). La piattaforma di programmazione Java è fondata sul linguaggio stesso, sulla Macchina virtuale Java (JVM) e sulle API Java (Application Programming Interface) Interfaccia di Programmazione di un'Applicazione. [7]Il linguaggio Java è derivato da un linguaggio chiamato OAK, che fu sviluppato nei primi anni '90 da un gruppo di ricerca chiamato Green Team che apparteneva all'azienda Sun Microsystem. Questa azienda diede al Green Team il compito di sviluppare un ambiente software platform-indipendent. L'obiettivo fu anche proposto a James Gosling che si associò da subito al team. Il Green Team partì quindi dal presupposto di non conoscere la piattaforma (CPU e sistema operativo) sulla quale il nuovo software doveva essere eseguito. Con questi presupposti, il team di Gosling sviluppò un sistema derivante dalla estensione del compilatore C++ che però ben presto non rispettò le aspettative. Fu a questo punto che Gosling fece partire il Green Project, ovvero un progetto totalmente nuovo sulla base di specifiche che aveva in mente. Nacque così, intorno alla metà del 1991, Oak, un linguaggio di programmazione object oriented, platform-indipendent, robusto e sicuro. Dopo qualche fallimento nel giugno del 1994, Oak intraprese un'altra direzione, fu sfruttato da Bill Joy (co-fondatore di Sun) per progettare e sviluppare LiveOak "un grande-piccolo sistema operativo". In pochi mesi il progetto si indirizzò esclusivamente ad Internet e nel Gennaio del 1995 fu rinominato Java. Inoltre nell'autunno dello stesso anno Van Hoff implementò il compilatore Java utilizzando Java, mentre quello esi-stente era stato realizzato da Gosling in C++. In questo periodo aumentava l'attenzione sul World Wide Web, sebbene i browser erano alla loro prima generazione (NCSA Mosaic 1.0). Per questo motivo si decise di tentare di sviluppare un browser di seconda generazione utilizzando questo linguaggio. In un week-end, uno dei membri del team, Patrick Naughton sviluppò un prototipo di browser di nuova generazione basato su Java. Il team, compreso Gosling, ancora una volta cambiò rotta, questa volta dirigendo i propri sforzi verso il Web. In pochi mesi Naughton ed un suo collega Jonathan Payne completarono un browser basato su una tecnologia sconosciuta alla prima generazione dei browser e

scritto con il linguaggio Java. Nacque così HotJava 1.0 Il giorno del successo arrivò il 23 maggio 1995 quando la Sun Microsystem introdusse formalmente Java e HotJava in occasione del SunWorld '95. Dopo anni di duro lavoro, sacrifici, problemi, delusioni e perseveranza la visione del Green Team ebbe successo, un nuovo grande prodotto fu lanciato sul mercato. Con Java, Sun dimostrò che era riuscita a realizzare il primo linguaggio di programmazione che era "slegato" da ogni sistema operativo o microprocessore. Nel Gennaio del 1996 Sun fondò JavaSoft, una unità dedicata allo sviluppo di prodotti basati sulla tecnologia Java, all'aggiornamento del linguaggio e alla collaborazione con terze parti per creare applicazioni, tool e servizi per migliorare le capacità del linguaggio. Nello stesso mese JavaSoft rilasciò il Java Development Kit (JDK) 1.0 oggi arrivata alla versione 6.0, un collezione rudimentale di componenti per facilitare lo sviluppo di applicazioni scritte in Java, tra cui un compilatore, un applet viewer, un prototipo di debugger, la Java Virtual Machine (JVM) necessaria per eseguire programmi basati su Java su qualunque piattaforma e numerose librerie di classi per la grafica, l'audio, le animazioni ed il networking. Da questo punto in poi JavaSoft, anche con le segnalazioni dei sempre più numerosi programmatori Java, ha apportato periodicamente aggiornamenti e miglioramenti al JDK. Il 13 novembre 2006 la Sun ha rilasciato la sua implementazione del compilatore Java e della macchina virtuale sotto licenza GPL. L'8 maggio 2007 vennero pubblicate anche le librerie sotto licenza GPL, rendendo Java un linguaggio di programmazione la cui implementazione di riferimento è libera. Dal giorno dell'annuncio ufficiale il linguaggio è stato adottato da tutti i maggiori vendors di software incluse IBM, Hewlett Packard e Microsoft. Nei primi tempi sembrava che Java fosse il linguaggio giusto per la creazione di siti Web, ma in realtà non era così, aveva ed ha molte altre potenzialità. Oggi è un potente linguaggio di programmazione che sta diventando sempre di più la soluzione ideale ai problemi che accomunano aziende operanti in settori diversi (banche, software house, compagnie d'assicurazioni) come la sicurezza. La Sun ha investito molto in questi anni sul progetto Java ottenendo risultati straordinari. Java è stato

creato proprio per superare, i limiti che gli altri linguaggi hanno. In generale si andò nella direzione di un linguaggio potente, moderno, chiaro, ma soprattutto robusto e funzionante. In molti punti chiave del linguaggio è favorita la robustezza piuttosto che la potenza. La piattaforma Java è disponibile in tre configurazioni , la scelta tra una di esse dipende dall'uso che se ne vuole fare. Per il progetto di tesi è stata utilizzata la SE (Standard Edition) che permette di scrivere applicazioni stand-alone, applicazioni client-server, accesso a database, calcolo scientifico e altri tipi. Il 27 gennaio 2010 Oracle Corporation ha acquisito la Sun MicroSystem, ed ha deciso di investire molto nella tecnologia Java. Il vicepresidente di Oracle, Thomas Kurian, ha fornito numerosi dettagli sulle modalità di integrazione tra i prodotti dell'azienda e quelli di Sun e sulle strategie di rilancio ed il miglioramento di Java sia sotto il punto di vista delle performance che sotto quello del supporto ai dispositivi di nuova generazione. Il dirigente ha spiegato che ci sono diversi software di entrambe le aziende che, entro breve tempo, potranno essere fusi o integrati in un singolo prodotto. L'azienda si è impegnata a "rivitalizzare" la Java Community Process (JCP), l'organizzazione multivendor che gestisce e regolamenta lo sviluppo della tecnologia Java, cercando di renderla ancora più aperta e capace di attrarre un maggior numero di vendor e di sviluppatori indipendenti. Oracle inoltre, si è impegnata ad ottimizzare le performance runtime di Java Micro Edition, cioè la piattaforma indirizzata ai dispositivi mobili, migliorandone le funzioni relative alla gestione dei consumi e la possibilità, per gli sviluppatori, di adottare differenti tipi di interfaccia grafica. Il colosso statunitense si è poi detto particolarmente interessato a JavaFX, un linguaggio di scripting concepito per creare web application interattive e ricche di contenuti multimediali, cercando di competere con tecnologie rivali come Adobe Flash/AIR e Microsoft Silverlight . Per rendere l'idea di quanto Java si sia sviluppato, riporto alcune informazioni riprese direttamente dal sito della Sun. "Attualmente, la piattaforma Java è stata adottata da più di 6,5 milioni di sviluppatori software. Viene utilizzata in tutti i principali segmenti del settore ed è presente in un'ampia gamma di dispositivi, computer e

reti. Grazie a versatilità, efficienza, portabilità della piattaforma e sicurezza, la tecnologia Java è ideale per il network computing. Dai portatili ai datacenter, dalle console per videogiochi ai computer altamente scientifici, ai telefoni cellulari e a Internet, Java è onnipresente! Più di 4,5 miliardi di dispositivi sono attualmente basati sulla tecnologia Java, inclusi: Oltre 800 milioni di computer; 2,1 miliardi di telefoni cellulari e altri dispositivi portatili (fonte: Ovum); 3,5 miliardi di smart card; Decoder, stampanti, web cam, giochi, sistemi di navigazione per veicoli, terminali delle lotterie, apparecchiature mediche, parchimetri e così via. Oggi, molte scuole e università offrono corsi di programmazione per la piattaforma Java. Inoltre, gli sviluppatori possono anche perfezionare le proprie conoscenze di programmazione Java leggendo il sitoWeb di Sun java.sun.com, iscrivendosi alle newsletter incentrate sulla tecnologia Java, utilizzando le esercitazioni Java e il Nuovo utente del centro di programmazione Java e iscrivendosi ai corsi Web, virtuali o tenuti da un istruttore".

5.2 SQL

SQL(Structured Query Language) è un linguaggio d'interrogazione per Database ideato per leggere, modificare e gestire dati memorizzati in un sistema basato sul modello relazionale.[8] L'interrogazione e la gestione delle basi di dati, avvengono attraverso costrutti denominati query. Il linguaggio si divide in tre diversi sottoinsiemi:

- Data Definition Language(DDL) - permette di creare e cancellare database o di modificarne la struttura
- Data Manipulation Language(DML) - permette di inserire, cancellare, modificare e leggere i dati
- Data Control Language(DCL) - permette di gestire gli utenti e i permessi

I comandi DDL (CREATE, ALTER, DROP) servono a definire la struttura del database e quindi dei dati da esso contenuti, però non fornisce gli strumenti per poter modificare i dati stessi. Di questa operazione se ne occupa il DML che attraverso specifiche istruzioni (SELECT, INSERT, UPDATE, DELETE) riesce a manipolare i dati. L'utente per poter accedere ed agire sulla struttura dati deve avere determinati permessi, questi permessi sono assegnati tramite il DCL. Gli informatici chiamano questo linguaggio di alto livello perchè permette di svolgere operazioni dichiarando solamente cosa si deve ottenere senza interessarsi del come. I linguaggi di terza generazione o procedurali sono quelli dove bisogna specificare il come si fa, non è sufficiente dichiarare il cosa si deve fare. Non è così per questo linguaggio, che pur limitando le scelte del programmatore e l'efficienza del programma, libera lo sviluppatore dal gravoso compito di scrivere pagine e pagine di codice. Chi usa questo linguaggio può essere usato sia da programmatori, sia da chi si avvicina in maniera marginale all'informatica, come ad esempio gli impiegati, i professionisti, i commessi, i magazzinieri, ecc. insomma chiunque ha la necessita di manipolare o consultare basi di dati. Probabilmente la causa del suo grande successo sta nella sua semplicità di utilizzo. Non bisogna, però farsi ingannare, perchè se da un lato SQL è intuitivo e semplice, da un altro, per essere capito a fondo deve essere studiato attentamente per riuscire a capirne tutte le sfumature e le notevoli potenzialità. Il linguaggio SQL nasce nel 1974 in California, quando la società IBM sviluppa il System R, un applicativo per la gestione dei dati, il cui linguaggio veniva chiamato Sequel. Questo linguaggio rappresentava l'embrione di quello che sarebbe poi diventato l'attuale SQL. Alla fine degli anni 70, l'IBM, sviluppò un altro prodotto chiamato DB2, un Relational Database Management System, cioè un sistema per la gestione di database relazionali. Questo RDBMS utilizzava una primordiale versione di SQL. Da allora si sono succeduti un gran numero di prodotti che implementano questo linguaggio e ogni produttore, aggiungendo delle variazioni e estensioni proprie, ha contribuito alla creazione della miriade di dialetti che oggi vengono chiamati SQL.

Capitolo 6

Funzionamento del software

6.1 Descrizione

All'avvio dell'applicazione si avrà la seguente interfaccia grafica.

Figura 6.1: Interfaccia grafica iniziale

Come si può notare sono disabilitate alcune funzionalità. L'unica funzione di menù abilitata, è quella file che al suo interno ne ha delle altre. Quella fondamentale è quella di login, che permette di effettuare l'accesso, così da poter avere abilitate tutte le funzionalità del software.

Figura 6.2: Interfaccia grafica iniziale con il primo menù

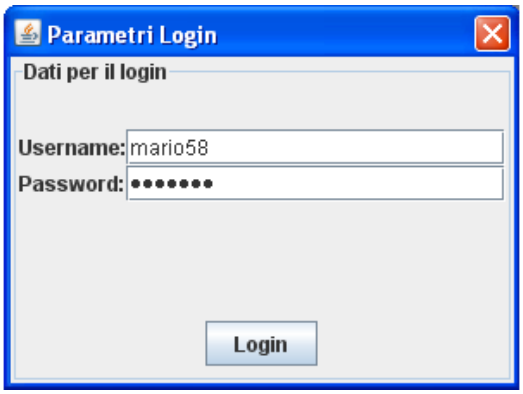


Figura 6.3: Interfaccia grafica Login

Se la username e la password inserite saranno corrette si potrà effettuare l'accesso. Quindi successivamente all'accesso sarà possibile effettuare l'inserimento di un nuovo paziente, effettuare una ricerca oppure abilitare un eventuale medico di sostituzione. Iniziando dall'inserimento, come si può vedere dall'interfaccia, è possibile completare i campi, per poi fare il salvataggio nel database. E' necessario inserire almeno i primi tre campi, altrimenti il sistema genererà un errore. Inoltre la data di nascita dovrà essere inserita nel modo corretto, altrimenti anche in questo caso il sistema non permetterà l'inserimento nel database, ma genererà un messaggio di errore

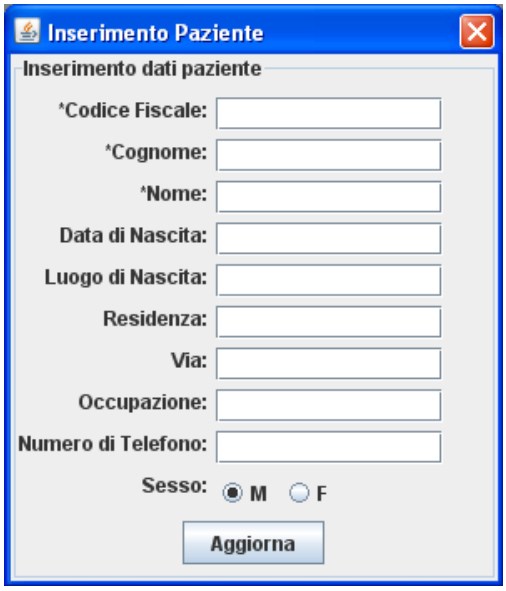


Figura 6.4: Interfaccia grafica inserimento paziente

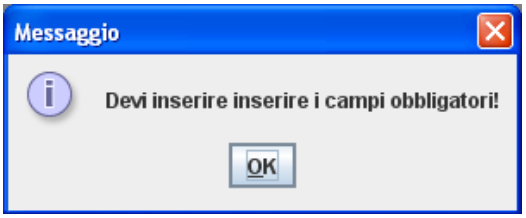
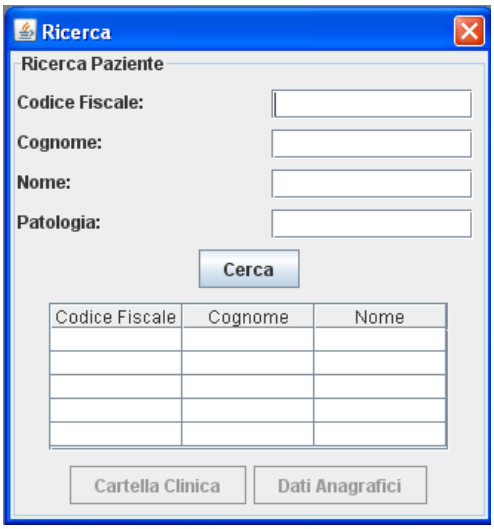
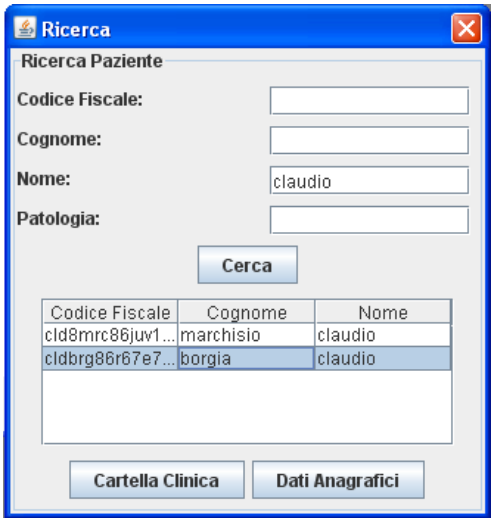


Figura 6.5: Messaggio di errore nell'inserimento

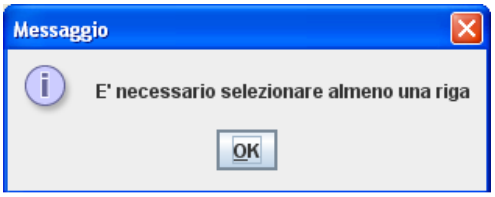
Anche per quanto riguarda la ricerca, osservando l'interfaccia, c'è bisogno di inserire alcuni parametri. Però potrebbe anche essere effettuata senza inserire alcun parametro, in questo caso verranno visualizzati tutti i pazienti all'interno dell'archivio. I risultati della ricerca verranno visualizzati all'interno di una tabella. Successivamente si dovrà selezionare il paziente desiderato, e a quel punto si potrà scegliere se visualizzare i dati anagrafici del relativo paziente o la cartella clinica. Inizialmente i due pulsanti saranno disabilitati, entreranno in funzione solo dopo aver effettuato la ricerca. Se dopo la ricerca non verrà selezionato nessun paziente il sistema manderà un messaggio di avviso.



(a) Interfaccia grafica ricerca paziente



(b) Interfaccia grafica ricerca paziente con risultati



(c) Errore che viene segnalato da una finestra di dialogo

Se si sceglierà di entrare nella sessione dei dati anagrafici si avrà la possibilità di modificare alcuni dati, tranne il codice fiscale.

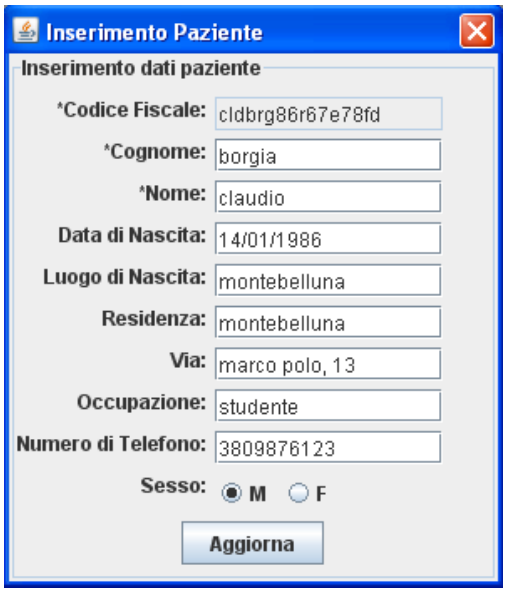


Figura 6.6: Interfaccia dati anagrafici

Se si sceglierà di entrare nella cartella clinica si avrà la possibilità di completare tutti i campi necessari per effettuare una nuova visita. Il sistema farà i controlli sulle varie date. Una volta inseriti tutti i dati necessari sarà possibile fare il salvataggio nel database, oppure stampare e salvare la ricetta.

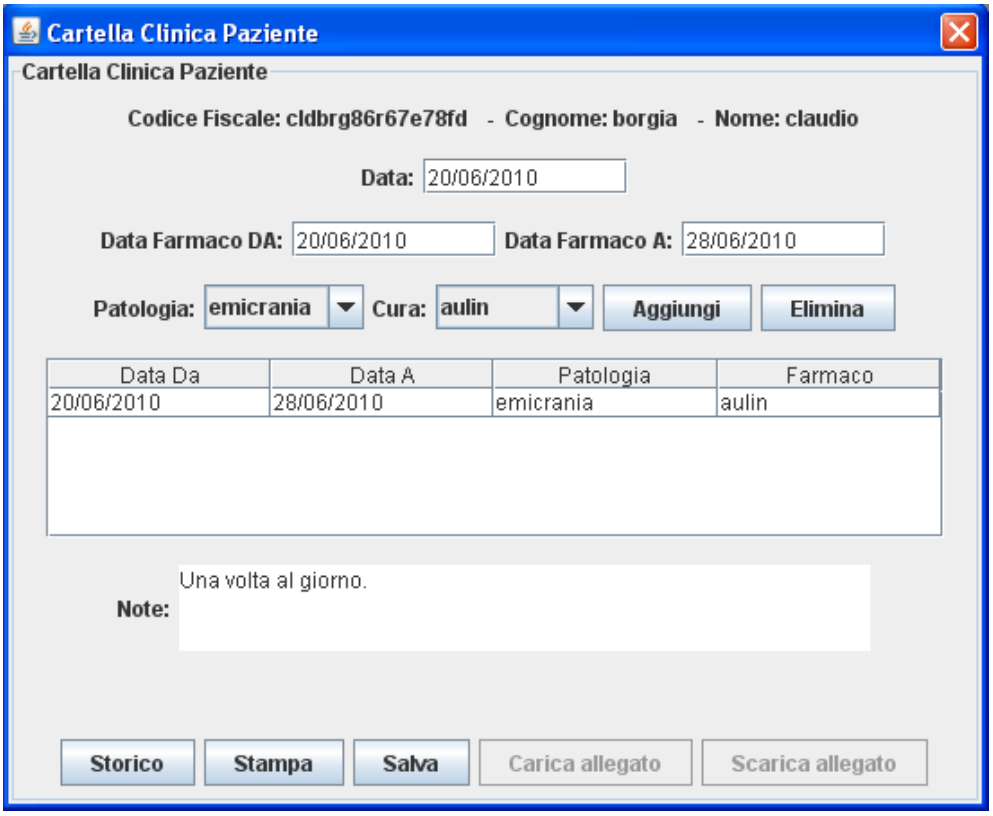


Figura 6.7: Interfaccia cartella clinica

Inoltre in questa sessione è possibile visualizzare lo storico di tutte le visite effettuate dal relativo paziente. Come si può vedere dalla schermata, tutte le visite saranno contenute all'interno di una tabella.

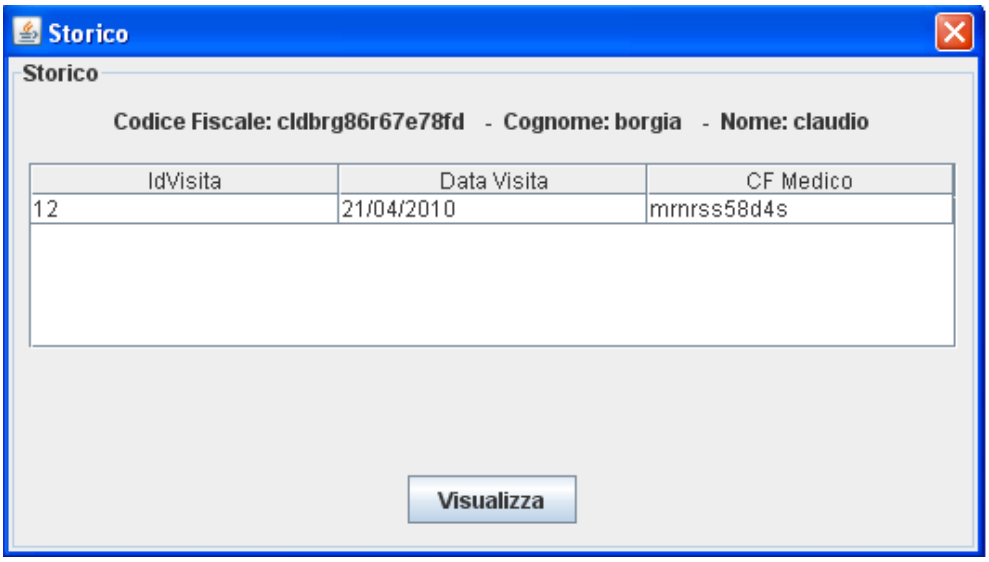
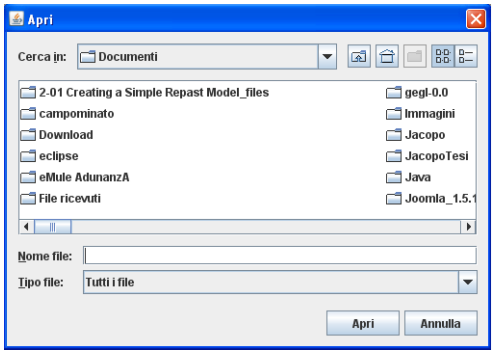
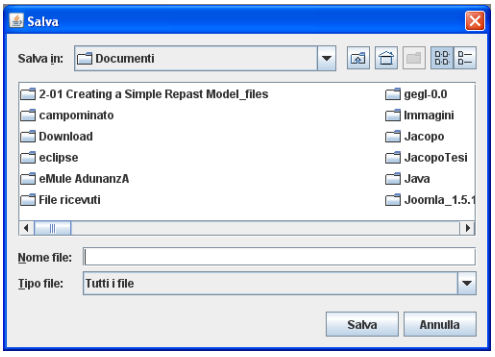


Figura 6.8: Interfaccia dello storico della cartella

Sarà possibile anche caricare o scaricare eventuali allegati, come per esempio certificati, analisi ecc.



(a) Interfaccia per l'upload



(b) Interfaccia per il download

L'ultima funzionalità del software è quella dell'abilitazione di un possibile sostituto. Qui il medico titolare sceglierà attraverso un menù, il medico che lo sostituirà, ed inserirà le date relative al periodo di sostituzione. Il medico sostituto potrà accedere al sistema solamente in quel relativo periodo.

Figura 6.9: Interfaccia per abilitare un sostituto

Il sostituto prima di effettuare il login dovrà scegliere una password che sarà valida solamente per il relativo periodo. Per fare questa operazione dovrà accedere alla funzionalità chiamata "primo accesso". In questa sessione il medico dovrà inserire il suo codice fiscale, se risulterà corretto verrà generata automaticamente una username, e successivamente potrà scegliere la password. Solo dopo aver terminato quest'operazione, potrà fare il login per accedere al sistema.

Figura 6.10: Interfaccia primo accesso sostituto

Conclusioni

In questa tesi si è sviluppato un software dimostratore per la gestione della cartella clinica per un medico di base. Uno sviluppo futuro di tale software, sarebbe quello di ampliarlo e modificarlo sulla base di una qualche forma di collaborazione con uno studio medico, in modo da poterlo poi validare.

Strumenti utilizzati

Per la realizzazione del progetto si sono utilizzati molti tipi di strumenti, tramite i quali si sono potuti creare diversi diagrammi necessari per il piano di processo, per la modellazione, per la progettazione del software, per l'implementazione del codice e per la stesura della documentazione.
Di seguito sono riportati i tools utilizzati:

Gantt Project: è un programma libero per la gestione dei progetti, permette di fare diagrammi di Gantt per pianificare un progetto e gestire le proprie risorse. E' scritto in Java ed è multipiattaforma. E' in grado di stabilire diversi punti ogni giorno, settimana o mese, controllando in modo efficace che ogni lavoro sia portato a termine. Nessuno potrà concludersi senza che prima sia finito quello precedente. GanttProject è uno strumento completo con funzionalità di esportazione in formato PDF o HTML.

WBS ChartPro: è un project management software usato per pianificare e visualizzare i progetti utilizzando un diagramma ad albero conosciuto come un Work Breakdown Structure (WBS) Chart. I grafici WBS visualizzano la struttura di un progetto e mostra la sua suddivisione in livelli più dettagliati. L'uso di WBS Chart Pro serve per abbozzare un piano di progetto sullo schermo utilizzando un approccio Top-Down

Visual Paradigm: è un CASE (Computer Aided Software Engineering) tool che facilita la realizzazione di diagrammi UML (Unified Modeling Lan-

guage). Il tool implementa tutti i tipi di diagrammi definiti nella specifica UML. In particolare, Visual Paradigm offre una comoda interfaccia grafica che permette di creare e salvare i vari diagrammi UML, come ad esempio lo use case diagram e il class diagram. I diagrammi realizzati possono essere salvati in diversi formati grafici, tra cui PNG e JPG.

Eclipse: è un programma scritto in linguaggio Java, e funge da piattaforma di sviluppo per la creazione di altri software di vario genere.

MySQL: è un relational database management system composto da un client con interfaccia a caratteri e un server, entrambi disponibili sia per sistemi Unix come GNU/Linux che per Windows.

Toad for MySQL: è uno strumento freeware di sviluppo che migliora la produttività e il lavoro degli sviluppatori ed amministratori che utilizzano MySQL su un sistema operativo Windows. E' uno strumento molto potente che prevede molte funzionalità tra cui l'import e l'export dei database, crea ed esegue query molto rapidamente e sviluppa il codice sql più efficentemente.

MikTex: Si tratta di un sistema TeX completo, con vari pacchetti aggiuntivi cui sono affiancati diversi tools utili ad esempio per la conversione e creazione di PDF. È uno dei più potenti, evoluti ed affidabili sistemi esistenti di tipo composizione elettronica. La gestione della bibliografia è un altro punto di forza, così come la gestione delle note, dei riferimenti e la composizione automatica degli indici. È stato sviluppato e migliorato diventando lo standard di fatto per tutte le pubblicazioni scientifiche.

OpenOffice: OpenOffice.org è una suite di proprietà Sun Microsystems, con copyright e licenza LGPL, che può essere classificata come software di produttività personale. Il progetto ha come obiettivi quelli di fornire a tutti gli utenti un prodotto libero che possa competere con i prodotti commer-

ciali attualmente dominanti in questo settore. Ha la caratteristica di essere parzialmente compatibile con i formati di file di Microsoft Office, ma dispone anche di formati nativi basati su XML che utilizzano un algoritmo di compressione. Sono supportate ufficialmente versioni per Linux, Microsoft Windows, Solaris e Mac OS X , ma è possibile installarlo anche su altri sistemi operativi.

Bibliografia

[1] http://www.softwaremedico.it/cartellaclinicaelettronica.asp, aprile 2010

[2] http://it.wikipedia.org/wiki/Diagramma_di_Gantt, febbraio 2010

[3] http://it.wikipedia.org/wiki/Unified_Modeling_Language, febbraio 2010

[4] Craig Larman "Applicare UML e i pattern" Pearson 2005

[5] http://www.quest.com/toad-for-mysql/

[6] http://it.wikipedia.org/wiki/Java_(linguaggio)

[7] http://www.mokabyte.it/1998/10/javastory.htm

[8] http://it.wikipedia.org/wiki/Sql

Ringraziamenti

Ringrazio i miei genitori che in questi anni mi hanno sostenuto psicologicamente ed economicamente, permettendomi di raggiungere questo importante obiettivo.