Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Cartella Cardiologica Virtuale

Progettazione e Sviluppo di un'applicazione web mediante i framework Spring e Hibernate

Relatore: Prof.ssa Elisabetta Fersini

Correlatore: Dott.ssa Annalisa Marra (Sync Lab Srl)

Relazione della prova finale di:
Manuel Nicoletta
Matricola 806237

Anno Accademico 2020-2021

Sommario

Il progetto "Cartella Cardiologica Virtuale" è stato realizzato durante uno stage curricolare presso l'azienda Sync Lab Srl e ha lo scopo di virtualizzare la cartella cardiologica di un paziente.
Questo elaborato introduce i concetti teorici necessari per l'implementazione del software realizzato, inoltre presenta le fasi di progettazione e sviluppo della Cartella Cardiologica Virtuale.

Ringraziamenti in costruzione

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'elaborato presenta le fasi di progettazione e sviluppo dell'applicazione realizzata durante l'attività di stage svolta presso l'azienda Sync Lab Srl.

1.1 Cartella Cardiologica Virtuale

Il progetto in questione ha come scopo l'informatizzazione della cartella clinica ponendo particolare rilevanza al rischio cardiologico.

1.1.1 Cartella Clinica

La cartella clinica è un documento sanitario il cui fine è quello di verbalizzare l'attività del reparto ospedaliero.

In particolare essa rappresenta il mezzo mediante il quale viene documentato il decorso clinico di ogni degente e contiene l'intera documentazione del ricovero, ovvero tutti i referti medici, e tutti i report delle visite che vengono effettuate al paziente durante la degenza.

1.1.2 Rischio Cardiologico

Il progetto in questione pone particolare attenzione al rischio cardiologico e al calcolo di esso.

Il *rischio cardiovascolare* (o *rischio cardiologico*) è un indicatore che permette di valutare la probabilità che una persona vada incontro ad un rischio cardiovascolare maggiore(ad esempio un ictus o un infarto del miocardio).

Il rischio viene calcolato mediante uno strumento denominato Carta Del Rischio.

Carte del Rischio Cardiovascolare

Le Carte del rischio cardiovascolare sono delle classi di rischio globale implementate in base ad alcuni fattori:

- *Sesso*: definito in Donne e Uomini;
- *Età*: definita negli intervalli: 40-49, 50-59, 60-69 anni di età;
- *Pressione Arteriosa Sistolica*: suddivisa negli intervalli: 90 mmHg - 130 mmHg, 130 mmHg - 150 mmHg, 150mmHg - 170 mmHg, 170 mmHg - 200 mmHg.

- *Colesterolemia Totale*: suddivisa negli intervalli: 130 mg/dl - 174 mg/dl, 174 mg/dl - 213 mg/dl, 213 mg/dl - 252 mg/dl, 252 mg/dl - 291 mg/dl, 291 mg/dl - 320 mg/dl.
- *Abitudine al Fumo*: si considera non fumatore colui che non fuma da almeno 12 mesi;
- *Diabete*: definito nelle due categorie: Paziente Diabetico e Paziente Non Diabetico

È importante precisare che qualora i fattori del paziente non fossero compresi negli intervalli di Età, Colesterolemia Totale e Pressione Sistolica Arteriosa specificati, allora non sarà possibile calcolare il rischio cardiologico.

Le Carte del Rischio Cardiologico vengono suddivise in quattro categorie in base al sesso e alla presenza o meno del diabete:

- Donna Non Diabetica
- Donna Diabetica
- Uomo Non Diabetico
- Uomo Diabetico

Ognuna di queste categorie viene ulteriormente suddivisa in fumatori e non fumatori.

Di seguito vengono riportate le carte del rischio cardiovascolare e il livello di rischio ad esse associato.

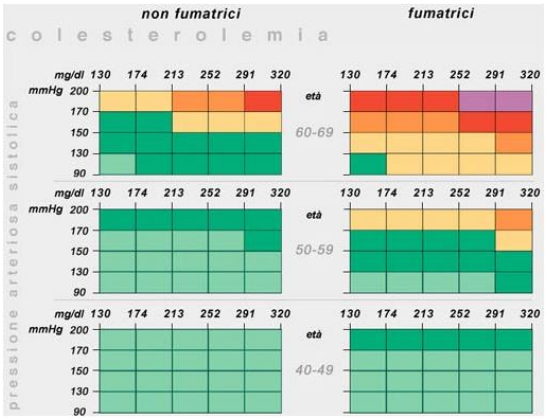

Figura 1.1: Carta Cardiologica delle Donne Diabetiche

Figura 1.2: Carta Cardiologica delle Donne Non Diabetiche

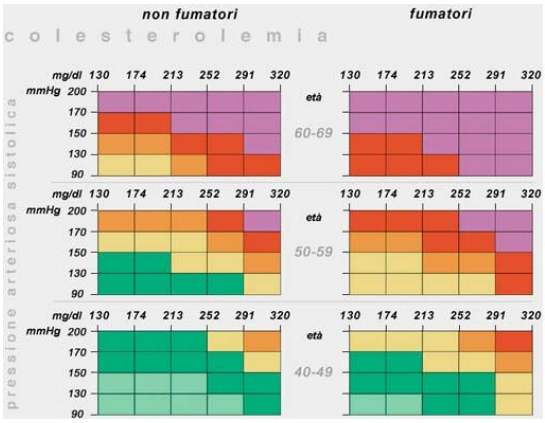

Figura 1.3: Carta Cardiologica degli Uomini Diabetici

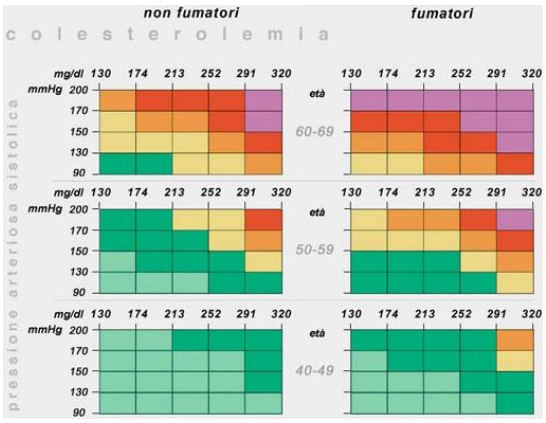

Figura 1.4: Carta Cardiologica degli Uomini Non Diabetici

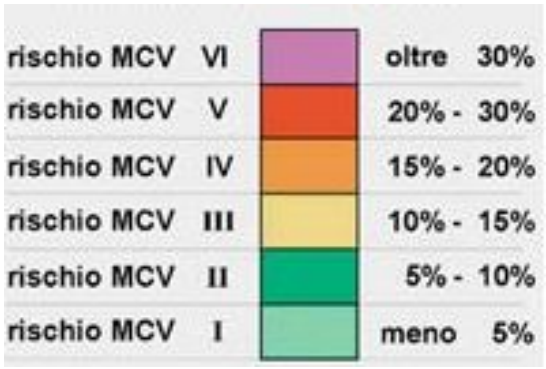

Figura 1.5: Indice del Rischio Cardiologico

La Cartella Cardiologica Virtuale permette quindi di informatizzare la cartella clinica e offre la possibilità di calcolare la percentuale di rischio cardiologico del paziente in base alle Carte del Rischio Cardiologico sopra descritte.

[0]Le Figure 1.1, 1.2, 1.3, 1.4, 1.5 sono tratte dal sito web "www.cuore.iss.it"

1.2 Analisi e Progettazione Del Software

Il ciclo di vita del software si compone di diverse fasi, tra le quali assume una particolare importanza l'Analisi Del Software.
Il fine di quest'ultima è quello di definire le funzioni che il programma deve garantire e di fissare un modello di riferimento per l'elaborazione dell'applicazione.
In particolare l'Analisi Del Software presenta le fasi di:

- **Analisi dei requisiti**

  Per *requisiti* si intendono le funzionalità che il software deve garantire. Essi si dividono in *Requisiti Funzionali*, ovvero i servizi offerti dal sistema e *Requisiti Non Funzionali*, ovvero i vincoli sui servizi offerti dal sistema.

- **Progettazione**

  Per *Progettazione* si intende lo sviluppo di un modello di riferimento per l'implementazione del software.
  In questo capitolo viene presentato il linguaggio UML, adottato per la realizzazione del modello.

- **Implementazione e realizzazione del software**

  Avendo come riferimento le specifiche del progetto, i requisiti e il modello di riferimento, si procede alla *realizzazione* vera e propria del software.

In questo elaborato verranno presentate le fasi di Analisi e Progettazione del software sopra descritte.

1.2.1 UML - Unified Modeling Language

L' UML (Unified Modeling Language) è un linguaggio di progettazione realizzato per facilitare la modellazione dell'architettura dei software basati sulla programmazione ad oggetti.
Questo strumento permette di modellare l'architettura di un software mediante l'utilizzo dei seguenti diagrammi:

- *Diagramma dei Casi D'uso*, ovvero la rappresentazione grafica dei casi d'uso
- *Diagramma di sequenza*, ovvero la rappresentazione grafica della sequenza di interazioni che avvengono tra l'attore e il sistema in un determinato caso d'uso
- *Diagramma delle attività*, ovvero la modellazione del flusso di lavoro di un'attività
- *Diagramma delle Classi*, ovvero la rappresentazione grafica delle relazioni presenti tra le diverse classi

In seguito verranno presentati tutti i diagrammi sopra elencati relativi ai diversi casi d'uso.

1.3 Design Pattern

I Design Pattern sono delle soluzioni progettuali generali applicabili a contesti simili.

Come definito dall'architetto Christopher Alexander:

> "Ogni pattern descrive un problema che si ripete più e più volte nel nostro ambiente, descrive poi il nucleo della soluzione del problema, in modo tale che si possa riusare la soluzione un milione di volte, senza mai applicarla alla stessa maniera."

L'architettura del software presentato in questo elaborato è definita mediante l'utilizzo del design pattern MVC.

1.3.1 Model View Controller

Il Model View Controller è un pattern architetturale molto diffuso che viene utilizzato principalmente nell'ambito della Programmazione orientata agli oggetti e nelle applicazioni web. L'MVC permette di realizzare l'indipendenza tra i principali livelli in cui è composto un sistema.
La figura 1.6 illustra una generica rappresentazione del funzionamento di quanto appena descritto.

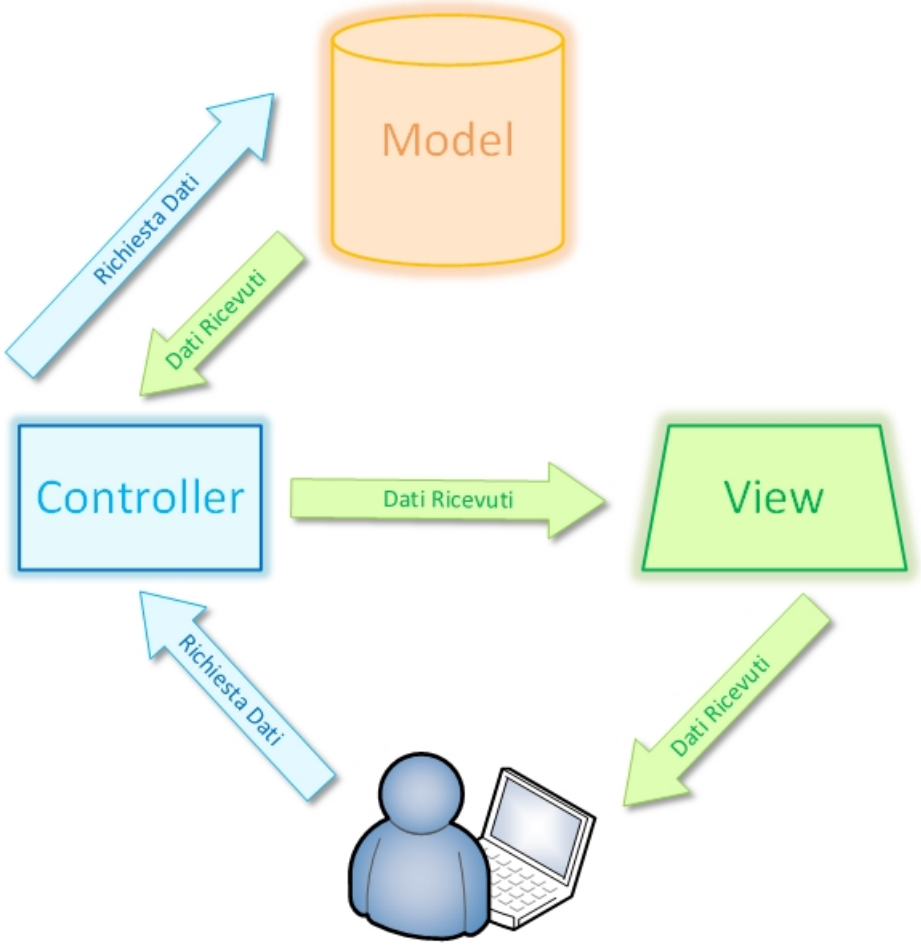

Figura 1.6: Rappresentazione grafica del Model View Controller

Come mostrato in figura 1.6 il Model View Controller è costituito da tre componenti principali:

- **Model**

  Il Model può essere considerato il modello principale dell'intera architettura in quanto ad esso è attribuita la gestione dei dati

- **View**

  Il View, definito anche Livello di presentazione, è costituito da tutte le funzioni che interagiscono direttamente con l'utente. Grazie ad esso è possibile definire il modo in cui l'utente vede e interagisce con l'applicazione.

- **Controller**

  Il Controller è il livello di connessione tra il Model e il View. Esso riceve l'input dall'utente e il livello di presentazione restituisce i dati in base alle "regole" definite nel Model

Capitolo 2

Tecnologie Utilizzate

In questo capitolo vengono riportate le tecnologie utilizzate per la realizzazione del progetto presentato in tale elaborato.

2.1 IDE

Un IDE (Integrated Development Environment) è un ambiente di sviluppo integrato progettato per la realizzazione di applicazioni.
Generalmente un IDE è caratterizzato da un *editor* che agevola la scrittura di codice software evidenziando eventuali errori di sintassi, e da un *debugger* che permette di testare l'applicazione realizzata.

2.1.1 Eclipse JEE

L'IDE utilizzato per la realizzazione dell'applicazione descritta in questo elaborato è Eclipse JEE, particolare la versione Eclipse JEE - 2019.
Tale è IDE è open source e può essere adattato alle diverse esigenze di programmazione aggiungendo dei plug-in.

2.2 Framework

Un Framework è una struttura logica il cui scopo è quello di fornire degli strumenti per velocizzare il lavoro di programmazione.
Più semplicemente un framework, comunemente detto piattaforma, è costituito da una serie di classi che vengono "richiamate" dal programmatore durante la stesura del codice. Tali classi implementano delle funzioni che possono essere utilizzate dallo sviluppatore nella costruzione dell'applicazione.

L'applicazione presentata in questo elaborato è stata sviluppata mediante il linguaggio di programmazione Java, con l'implementazione di due framework: Spring e Hibernate.

2.2.1 Spring

Spring è un modello di programmazione e configurazione per le applicazioni basate sul linguaggio Java.
Questo framework viene rilasciato per la prima volta nel 2003 con licenza Apache, e nasce come alternativa a JavaBeans.

Prima di continuare nella descrizione del framework Spring è opportuno fornire una breve descrizione di JavaBeans.

JavaBeans

Nel linguaggio di programmazione Java, per "*componente*" si intende una classe standard utilizzabile in più applicazioni.
Ciò permette alle componenti di essere definite una sola volta e di essere riutilizzate in contesti differenti, migliorando quindi la produttività.
Un *Java Bean* è una componente di Java, ovvero è una classe che ha delle caratteristiche fondamentali:

- tutti gli attributi sono di tipo *private*
- ha un metodo costruttore privo di argomenti
- per ciascun attributo vengono implementati i metodi **get** e **set** di tipo *public*.

Spring è caratterizzato da una struttura modulare, ovvero è costituito principalmente da cinque livelli:

- il livello*AOP (Aspect Oriented Programming)*: garantisce la Separation of Concerns, ovvero ha l'obiettivo di separare le funzionalità trasversali dell'applicazione dalla logica di business. Questo livello comprende a sua volta i livelli DAO e ORM:
  - livello *DAO (Data Access Object)*: è un pattern architetturale che ha lo scopo di separare le logiche di business da quelle di accesso ai dati
  - livello *ORM (Object Relational Mapping)*: permette la persistenza dei dati, nello specifico, tramite strumenti software (ad esempio Hibernate), offre la possibilità di associare le tabelle presenti nel database ai corrispondenti oggetti in Java.
- livello *JEE (Java Platform Enterprise Edition)*: insieme delle tecnologie che permettono di estendere le funzionalità del linguaggio Java alla programmazione web.
- livello *WEB*: implementa il pattern MVC (Model View Controller) presentato nela capitolo introduttivo, mediante il framework Spring MVC. In particolare il framework Spring MVC permette di realizzare applicazioni web basate sul pattern MVC sfruttando l'AOP e l'Inversion Of Control.

Tali livelli sono costruiti a partire dalla componente "base" di Spring denominata Core Container. Questa componente è costituita dai moduli:

- *Core*: il modulo fondamentale che implementa la Dependency Injection
- *Bean*: il modulo che fornisce le classi factory per l'inizializzazione dei bean;
- *Context*: il modulo che fornisce l'accesso a tutti i bean;
- *SpEL*: il modulo che permette la manipolazione dei bean a runtime

La Figura 2.1 offre una rappresentazione grafica dell'architettura del framework Spring.

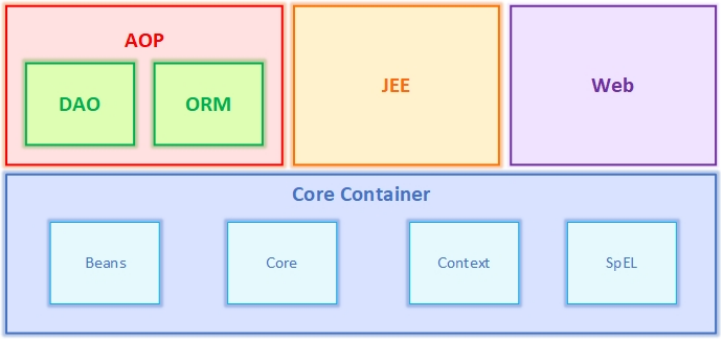

Figura 2.1: Architettura del framework Spring

Spring permette l'implementazione della Dependency Injection che è uno degli aspetti fondamentali dell' Inversion Of Control(IoC);

L'*Inversion Of Contro*l è una caratteristica dei framework che permette di spostare il controllo del flusso di sistema dal programmatore al framework.

La *Dependecy Injection* è un'implementazione del Inversion Of Control la cui idea di base è quella di utilizzare un componente esterno per creare gli oggetti e le loro dipendenze e assemblarli mediante l'Injection.

2.2.2 Hibernate

Hibernate è un framework che offre un servizio di ORM (Object Relational Mapping).
Quest'ultima è una tecnica di programmazione che permette di effettuare il mapping di un qualsiasi oggetto, ad esempio un oggetto Java, su un database relazionale, comunemente denominato RDBMS (Relational Database Management System).
Hibernate si occupa quindi di gestire tutte le operazioni CRUD (Create, Update, Read, Delete) sul database autonomamente, semplificando di molto il lavoro del programmatore.

A livello di codice, il mapping è realizzato mediante le annotazioni @Entity e @Table.

In Figura 2.2 viene riportato un frammento di codice che mostra l'utilizzo delle annotazioni sopra indicate.

```
@Entity
@Table(name = "configurazioneRischio")
public class ConfigurazioneRischio {

    @Id
    @Column(name="idRischio")
    private String idRischio;
    @Column(name="sesso")
    private String sesso;
    @Column(name="etaMin")
    private int etaMin;
    @Column(name="etaMax")
    private int etaMax;
    @Column(name="diabete")
    private int diabete;
    @Column(name="fumo")
    private int fumo;
    @Column(name="pressioneMin")
    private int pressioneMin;
    @Column(name="pressioneMax")
    private int pressioneMax;
    @Column(name="colesteroloMin")
    private int colesteroloMin;
    @Column(name="colesteroloMax")
    private int colesteroloMax;
    @Column(name="percentuale")
    private int percentuale;
```

Figura 2.2: Frammento di codice dell'applicazione Cartella Cardiologica Virtuale

Con l'annotazione @Entity si indica che la classe definita (in questo caso "ConfigurazioneRischio") deve essere trattata come un bean, mentre @Table specifica che quella determinata entità deve essere trattata come una tabella del datatabase.
In Figura 2.2 sono inoltre presenti le annotazioni @Id e @Column. Con @Id si specifica che l'attributo (in questo caso "idRischio") è una chiave primaria.
Mentre con l'annotazione @Column si indicano gli attributi della tabella.

2.2.3 AngularJS

AngularJS è un framework Javascript sviluppato da Google che fa la sua prima comparsa nel 2010. Tale piattaforma è un framework open source per le applicazioni web e adotta il pattern architetturale MVC. AngularJs è stato realizzato con lo scopo di semplificare lo sviluppo del front-end delle web-app e ha la particolarità di estendere le funzionalità delle singole pagine HTML mediante l'utilizzo di determinate direttive.

Il framework descritto in questo capitolo è facilmente implementabile, infatti è sufficiente inserire lo script presentato in Figura 2.3 all'interno della pagina web.

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

Figura 2.3: Script di AngularJS utilizzato nella Cartella Cardiologica Virtuale

Successivamente all'inserimento dello script è possibile usufruire di tutte le funzionalità offerte da questo framework mediante le diverse direttive. Alcune direttive di AngularJS sono:

- *ng-model*: direttiva che associa il valore di controlli HTML per i dati delle applicazioni;
- *ng-app*: direttiva che inizializza un'applicazione AngularJS;
- *ng-init*: direttiva che inizializza i dati delle applicazioni;
- *ng-show* e *ng-hide*: due direttive che consentono rispettivamente di visualizzare o nascondere elementi sulla pagina web in seguito alla valutazione di una espressione;

Nella web-app descritta in questo elaborato è stata utilizzata la direttiva *ng-show*. Le seguenti figure 2.4 e 2.5 , tratte dal codice dell'applicazione in questione, illustrano l'implementazione di tale direttiva.

```
<a href="#" ng-click="login = !login" class="btn btn-outline-primary">Login</a>
```

Figura 2.4

```
<div ng-show="login" class="container">
```

Figura 2.5

In particolare nella figura 2.4 viene mostrato il codice che permette di visualizzare il "contenitore" (definito mediante il codice mostrato in figura 2.5) in seguito al click sul link "Login".

Capitolo 3

Analisi Dei Requisiti

La Cartella Cardiologica Virtuale ha lo scopo di virtualizzare una cartella clinica permettendo ad un qualsiasi paziente di accedere alla propria area personale e visualizzare tutte le visite che gli sono state effettuate.
Inoltre un medico ha la possibilità di accedere in qualsiasi momento alla cartella clinica dei propri pazienti per visualizzare le loro visite, per calcolare il loro rischio cardiologico e caricare eventuali referti medici.

Di seguito verranno presentate le specifiche di tale progetto seguite da un'approfondita analisi dei requisiti.

3.1 Specifiche Generali

Il progetto trattato in questo elaborato presenta tre attori principali:

- Amministratore
- Medico
- Paziente

Il sistema ha le seguenti funzionalità principali:

- Registrazione degli utenti nel sistema con relativa assegnazione di username e password;
- Inserimento delle visite e dei referti da parte del medico nella cartella di un determinato paziente;
- Accesso del paziente alla propria area personale e visualizzazione delle visite effettuate.

3.2 Requisiti Funzionali

In questa sezione vengono analizzate nello specifico le mansioni dei diversi attori del sistema.

L'*amministratore*, dopo aver effettuato il login, dispone delle seguenti funzionalità:

- *Inserimento di un medico*: l'amministratore compila il form inserendo i dati del anagrafici del medico;
- *Eliminazione di un medico*;
- *Inserimento di un nuovo amministratore*

Il *medico*, dopo essersi autenticato, dispone delle seguenti funzionalità:

- *Inserimento di un nuovo paziente*: il medico compila il form inserendo i dati anagrafici del paziente;
- *Visualizzazione della lista dei pazienti* registrati nel sistema;
- *Download della lista dei pazienti* registrati nel sistema: viene generato un file PDF contenete la lista di tutti i pazienti presenti nel sistema con i relativi dati anagrafici;
- *Accesso alla cartella di un paziente*: il medico può accede alla cartella di un determinato paziente e disporre delle seguenti funzioni:
  - *Inserimento di una visita*: il medico compila il form della visita inserendo tutti i dati necessari (ad esempio il peso, l'altezza, e la circonferenza della vita del paziente) e successivamente il sistema memorizza tali informazioni nella cartella del paziente;
  - *Eliminazione di una visita*;
  - *Visualizzazione ed eventuale modifica della visita del paziente*: il medico, dopo avere selezionato una determinata visita all'interno della cartella del paziente può decidere di modificarla compilando il relativo form;
  - *Download della lista delle visite del paziente*: viene restituito un file di tipo PDF con la lista di tutte le visite effettuate dal paziente;
  - *Inserimento di un referto medico*: il medico inserisce il referto (che generalmente può essere un file di tipo immagine o PDF) all'interno della cartella del paziente;
  - *Eliminazione di un referto medico*;
  - *Download di un referto medico* di un paziente: il referto, ovvero il file caricato precedentemente dal medico oppure dal paziente stesso, viene memorizzato sul dispositivo del medico;
  - *Calcolo del rischio cardiologico* di un paziente: il medico compila il relativo form inserendo i dati riguardanti il paziente (ad esempio la pressione e la colesterolemia) e il sistema restituisce la percentuale di rischio per il paziente. Infine il rischio viene inserito nella cartella del paziente. Il medico può calcolare in qualsiasi momento il rischio cardiologico per il paziente e decidere di associarlo o meno alla sua cartella.

Il *paziente*, dopo aver effettuato il login, dispone delle seguenti funzionalità:

- *Registrazione nel sistema*: il paziente può registrarsi autonomamente nel sistema compilando il relativo form e inserendo i propri dati anagrafici;
- *Accesso alla propria cartella*: il paziente, dopo aver effettuato il login, può eccedere alla propria cartella personale e da qui può:
  - *Visualizzare le proprie visite*;
  - *Inserire un referto*: il paziente può caricare nella propria cartella un referto medico (in genere si tratta di un file di tipo immagine o PDF)
  - *Scaricare un referto*: il referto verrà salvato sul dispositivo dell'utente
  - *Scaricare la lista delle visite*: il paziente può effettuare il download di tutte le visite che gli sono state effettuate; verrà generato un file PDF contenente tutte le informazioni inerenti le visite del paziente;
  - *Visualizzare il proprio rischio cardiologico*, se quest'ultimo è già stato calcolato dal medico (in caso contrario il paziente visualizzerà un messaggio di errore)
  - *Calcolare il proprio rischio cardiologico*: il paziente può compilare il form inserendo le opportune informazioni (ad esempio pressione e colesterolemia). Il Rischio calcolato dal paziente però non può essere associato alla sua cartella in quanto si preferisce attendere la supervisione del medico.

3.3 Requisiti non Funzionali

Come è già stato esplicitato in precedenza, il sistema prevede tre attori principali: Amministratore, Medico e Paziente.
Per ognuno di essi vi è un livello di accesso differente.

L'operazione di autenticazione avviene mediante una pagina di login.
Il sistema riconosce il tipo di utente (ovvero amministratore, medico oppure paziente) e lo reindirizza all'opportuna pagina iniziale.

Un medico può essere inserito all'interno del sistema solamente da un amministratore.

Un paziente può essere inserito all'interno del sistema da un medico oppure si può registrare autonomamente.

Le operazioni di inserimento, quali, ad esempio, l'inserimento di un nuovo paziente oppure l'inserimento di una nuova visita, richiedono inizialmente la compilazione di un form in cui inserire i dati opportuni e, successivamente per completare l'operazione è necessario che l'utente faccia click sull'apposito pulsante di "salva" o "invia". Il sistema verifica se i dati inseriti rispettano i vincoli definiti (ad esempio la tipologia di dato, o la mancanza del dato).
Nel caso in cui i vincoli del form non vengono soddisfatti allora il sistema restituisce un messaggio di errore.
In caso contrario il sistema provvede a salvare i dati inseriti all'interno del database.

L'operazione di download dei referti inseriti all'interno del sistema si conclude con il salvataggio del referto desiderato sul dispositivo dell'utente.

Le operazioni di download delle liste delle visite e delle liste dei pazienti generano un file pdf contente tutte le informazioni rispettivamente delle visite e dei pazienti.

3.4 Casi d'uso

In questo capitolo vengono trattati i casi d'uso di: Login, Visualizzazione di una visita, Compilazione del form di una visita, Compilazione del form di inserimento di un medico e Inserimento di un referto.

La Figura 3.1 mostra in modo schematico i casi d'uso che si è scelto di trattare in questo capitolo, evidenziando gli attori di tali casi.

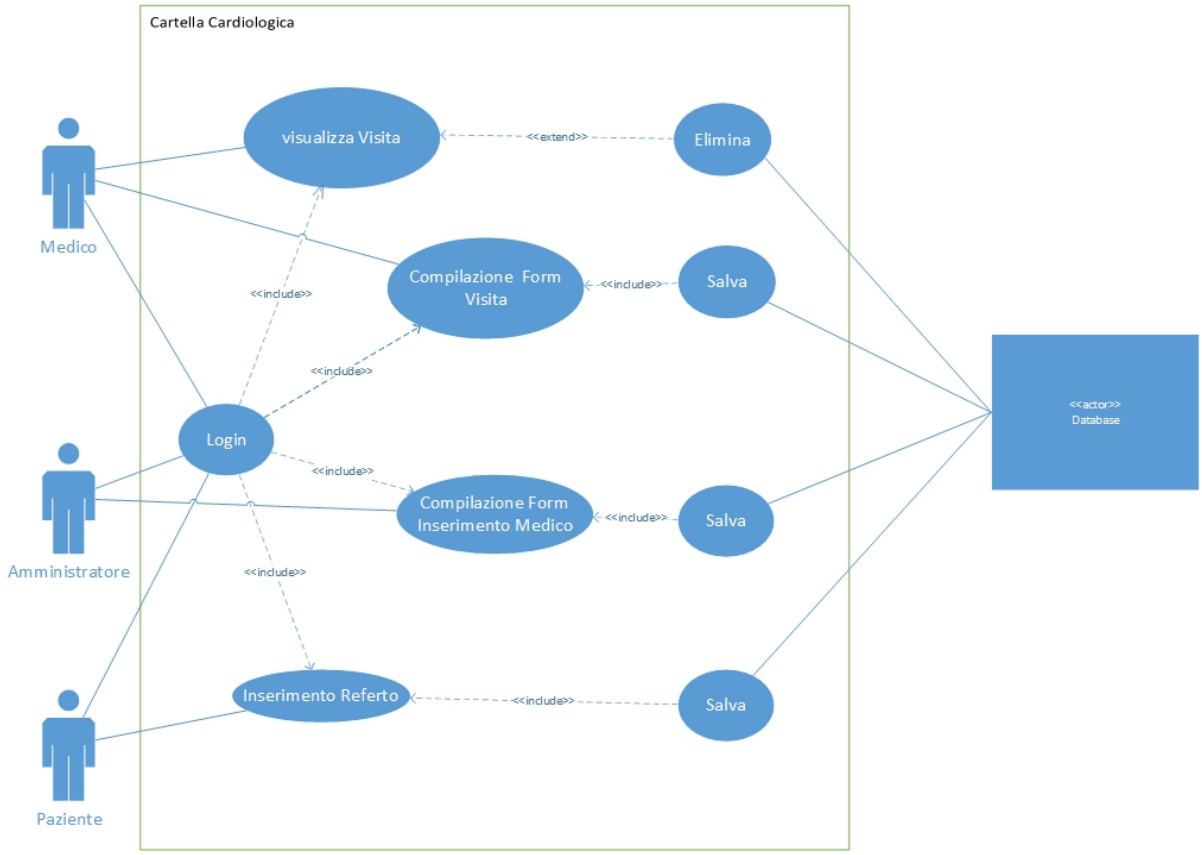

Figura 3.1: Diagramma Dei Casi D'Uso

Di seguito vengono approfonditi i casi d'uso sopra elencati e per ognuno si riportano i relativi diagrammi di sequenza e diagrammi delle attività.

3.4.1 Login

Nome: Login
Portata: Cartella Cardiologica Virtuale
Livello: Obiettivo Utente
Attore Primario: Amministratore, Medico, Paziente
Parti interessate e interessi: Un Utente, ovvero un amministratore, un medico o un paziente, vuole accedere al sistema
Pre-condizioni: L'utente è registrato
Garanzie di Successo: L'utente effettua il login e accede all'applicazione
Scenario principale di successo: L'utente effettua il login e accede alla sua area personale

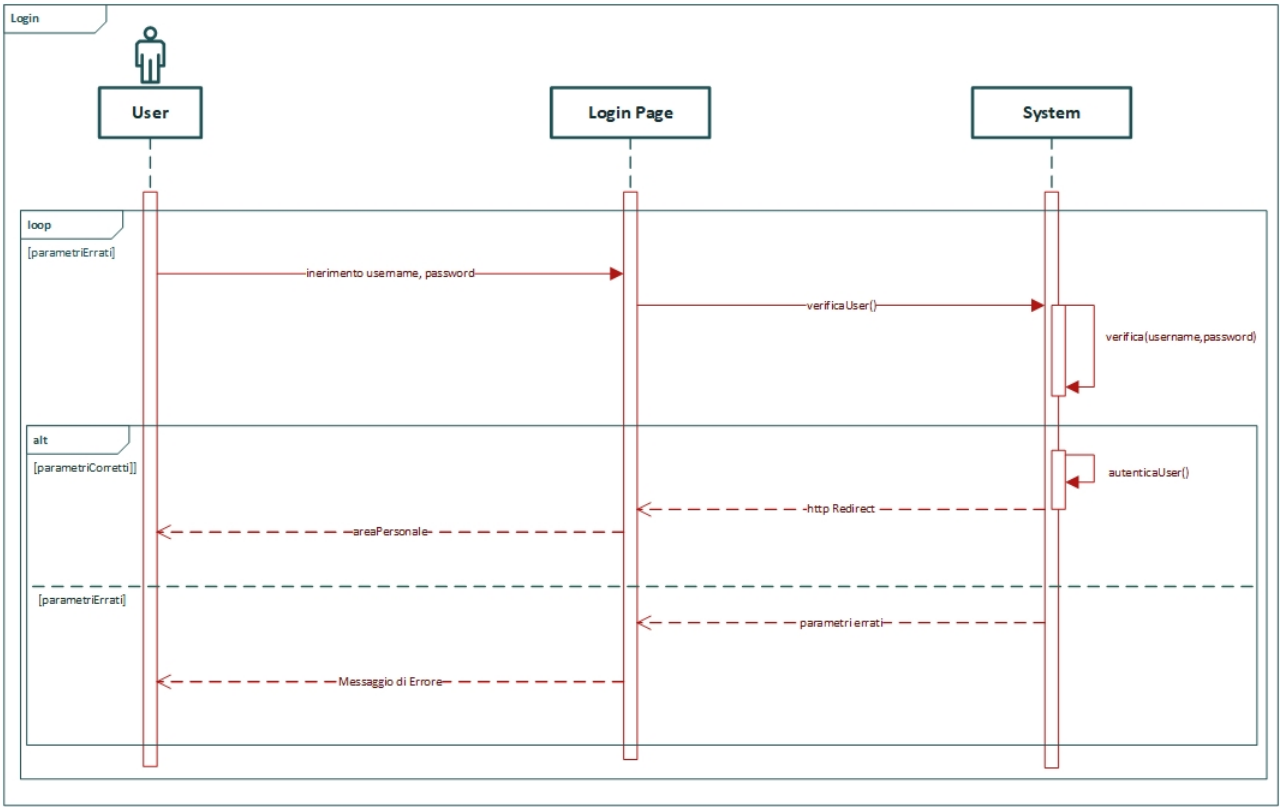

Figura 3.2: Diagramma Di Sequenza del Login

3.4.2 Visualizza Visita

Nome: Visualizza Visita
Portata: Cartella Cardiologica Virtuale
Livello: Obiettivo Utente
Attore Primario: Medico
Parti interessate e interessi: Il medico vuole visualizzare la visita di un paziente
Pre-condizioni: Il medico deve essere registrato nel sistema e deve aver effettuato il login
Garanzie di Successo: Il medico visualizza la visita del paziente
Scenario principale di successo:

- Il medico effettua il login
- Accede alla cartella del paziente
- Visualizza la visita del paziente

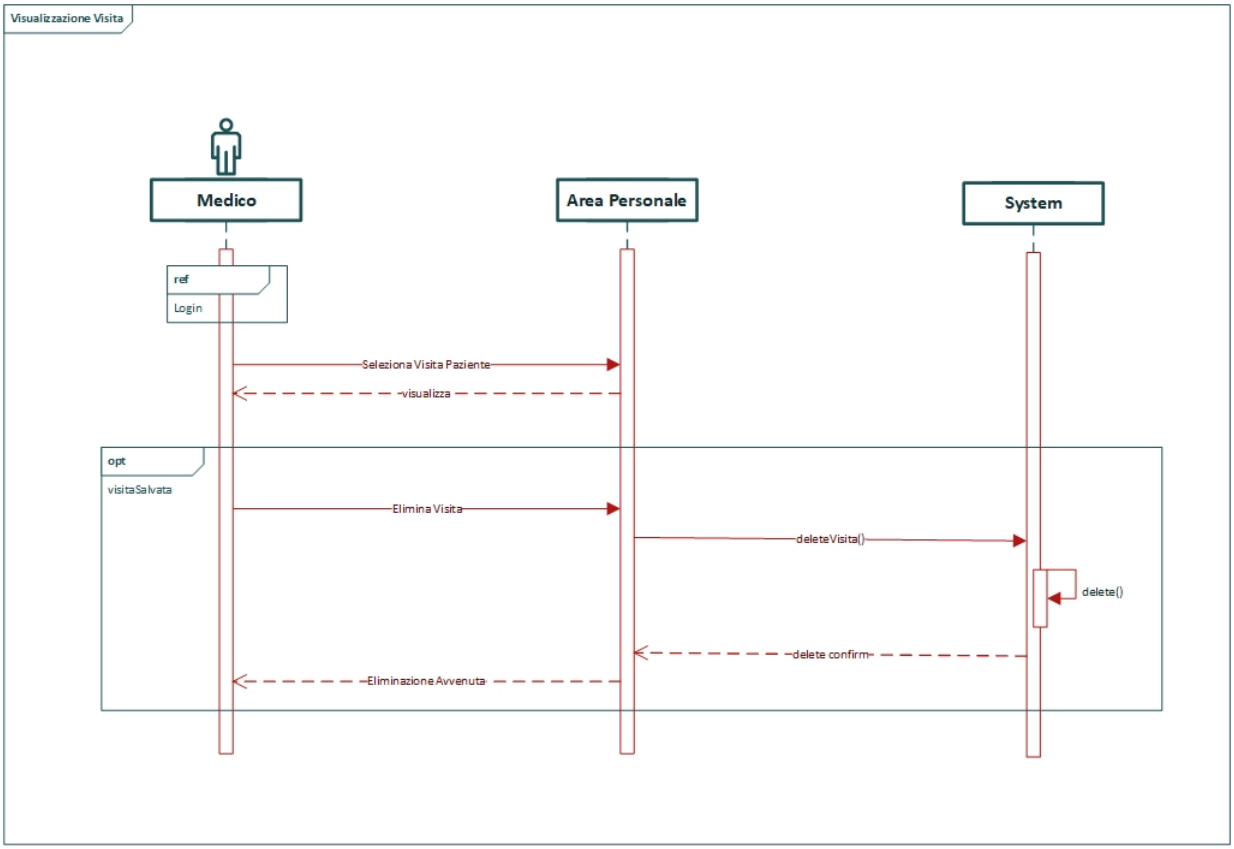

Figura 3.3: Diagramma Di Sequenza del caso d'uso: Visualizza Visita

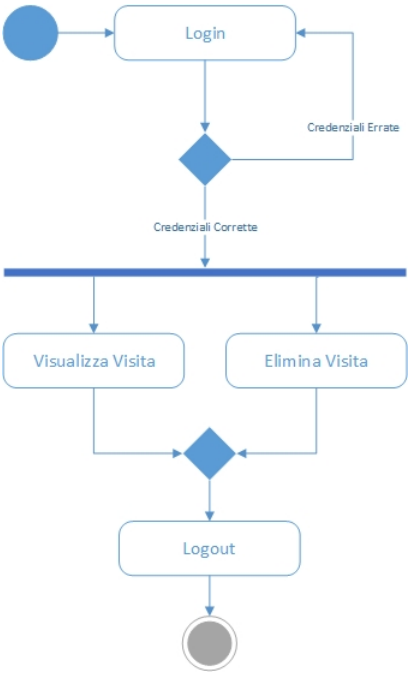

Figura 3.4: Diagramma delle Attività del caso d'uso: Visualizza Visita

3.4.3 Compilazione Form Inserimento Medico

Nome: Compilazione Form Inserimento Medico
Portata: Cartella Cardiologica Virtuale
Livello: Obiettivo Utente
Attore Primario: Amministratore
Parti interessate e interessi: L'amministratore vuole compilare il form per l'inserimento dei dati di un nuovo medico
Pre-condizioni:L'amministratore è registrato nel sistema
Garanzie di Successo: L'amministratore compila il form con i dati del nuovo medico
Scenario principale di successo:

- L'amministratore effettua il login
- compila il form con i dati del paziente
- il medico viene inserito nel sistema

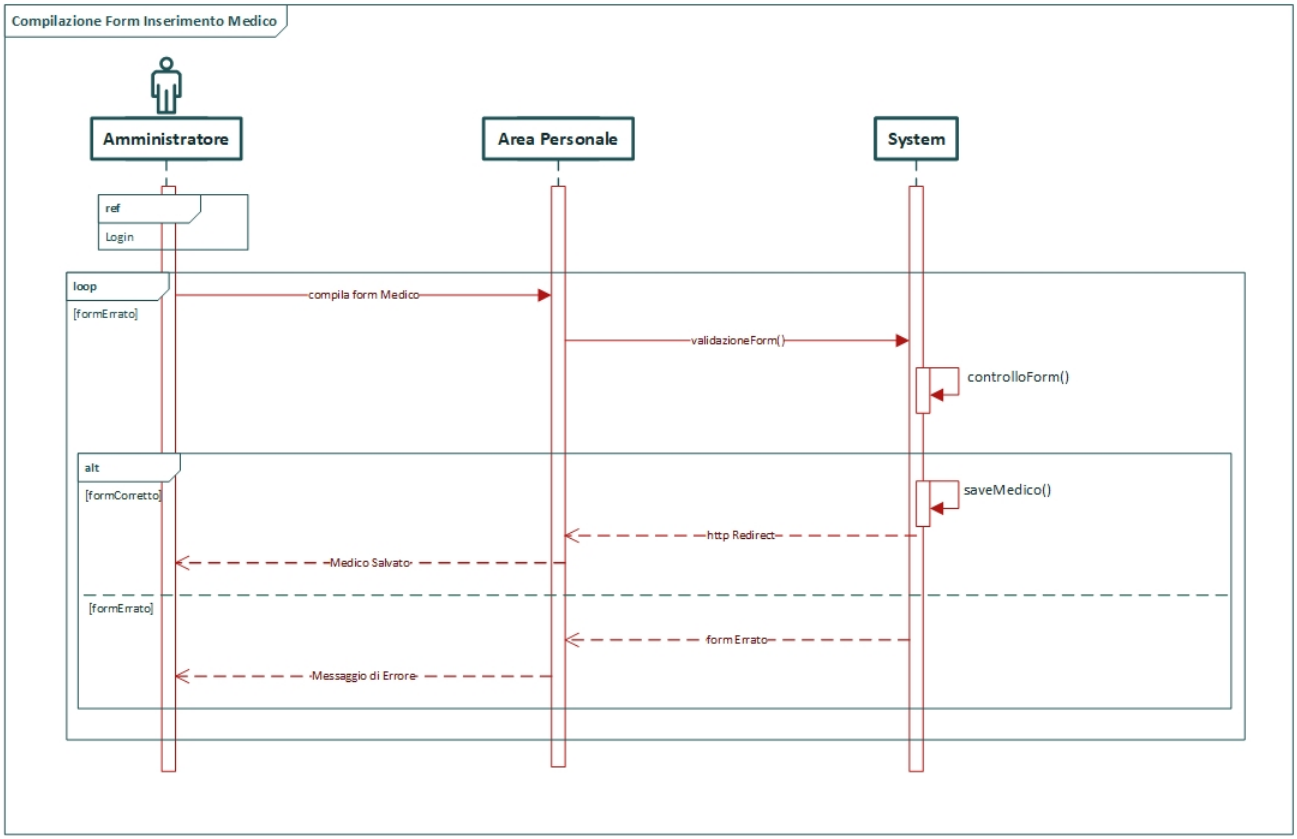

Figura 3.5: Diagramma Di Sequenza del caso d'uso: Compilazione Form Inserimento Medico

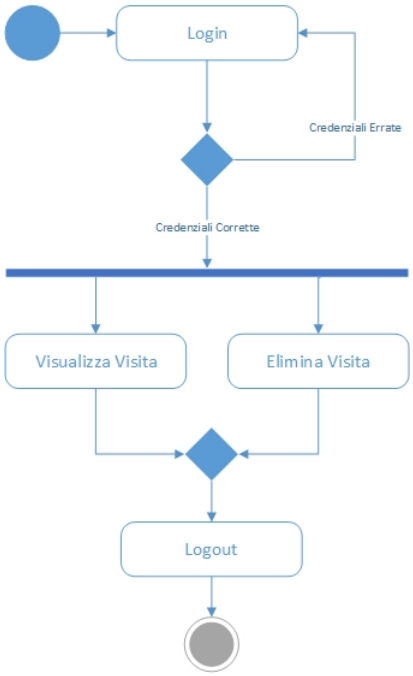

Figura 3.6: Diagramma delle Attività del caso d'uso: Compilazione Form Inserimento Medico

3.4.4 Compilazione Form Visita

Nome: Compilazione Form Visita
Portata: Cartella Cardiologica Virtuale
Livello: Obiettivo Utente
Attore Primario: Medico
Parti interessate e interessi: Il medico vuole compilare il form della visita per il paziente
Pre-condizioni:Il medico è registrato nel sistema
Garanzie di Successo: Il medico compila il form con i dati della visita
Scenario principale di successo:

- Il medico effettua il login
- compila il form della visita con i valori del paziente
- la visita viene inserita nel sistema

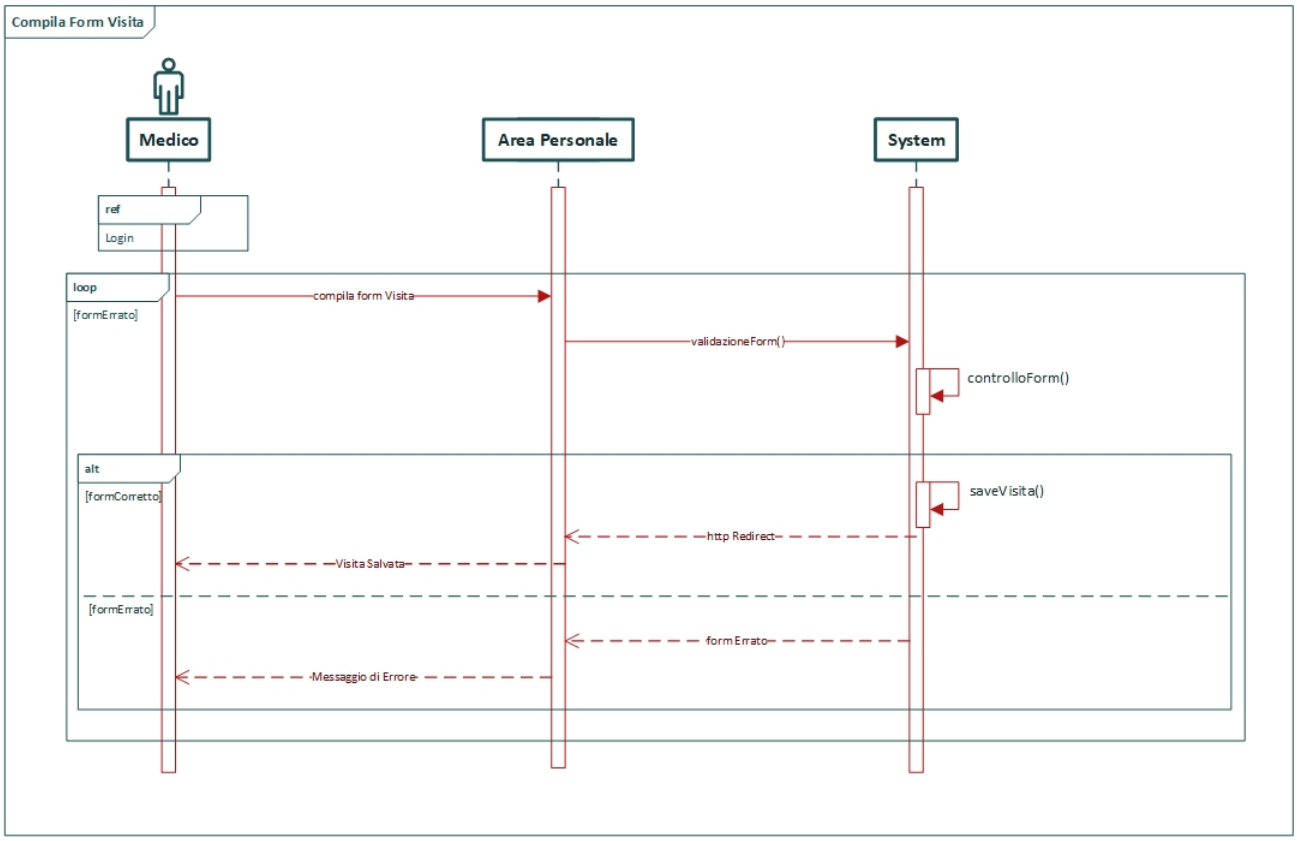

Figura 3.7: Diagramma Di Sequenza del caso d'uso: Compilazione Form Visita

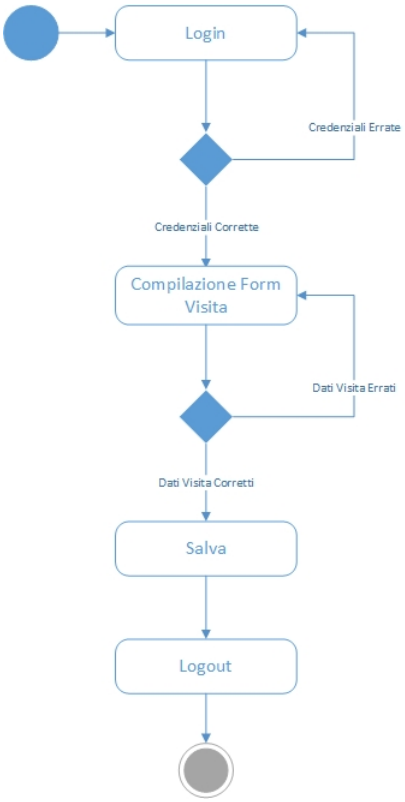

Figura 3.8: Diagramma delle Attività del caso d'uso: Compilazione Form Visita

3.4.5 Inserimento Referto

Nome: Inserimento Referto
Portata: Cartella Cardiologica Virtuale
Livello: Obiettivo Utente
Attore Primario: Paziente
Parti interessate e interessi: Il paziente vuole inserire un referto nella sua cartella clinica
Pre-condizioni:Il paziente è registrato nel sistema
Garanzie di Successo: Il paziente inserisce il referto, ovvero un file, nel apposito form
Scenario principale di successo:

- Il paziente effettua il login
- accede alla sua area personale
- carica il file (referto) nel form
- il referto viene memorizzato nella cartella del paziente

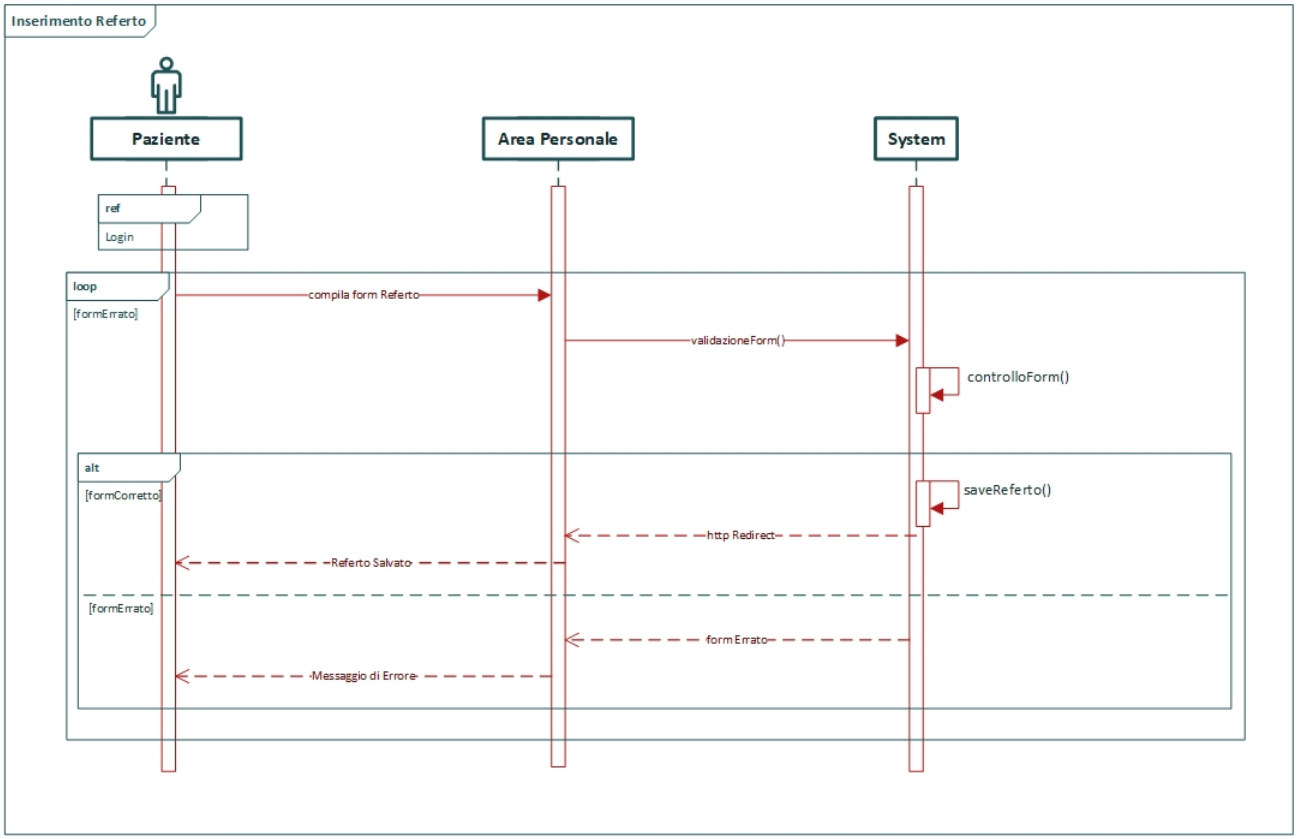

Figura 3.9: Diagramma Di Sequenza del caso d'uso: Inserimento Referto

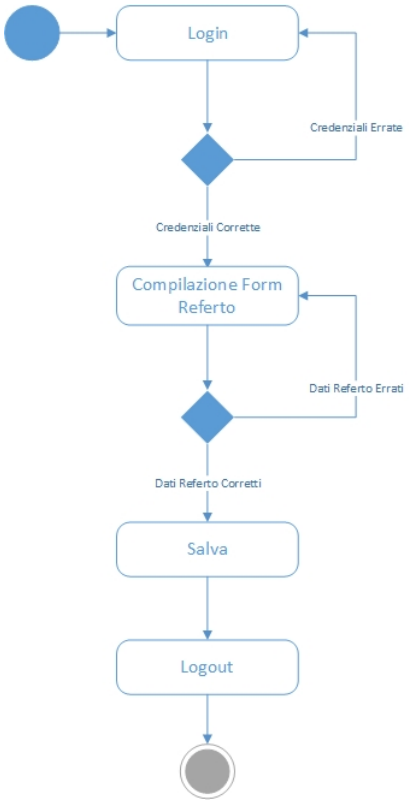

Figura 3.10: Diagramma delle Attività del caso d'uso: Inserimento Referto

Capitolo 4

Progettazione

In questo capitolo vengono presentate tutte le scelte di progettazione adottate per l'implementazione della Cartella Cardiologica Virtuale.

Come già accennato in precedenza lo scopo di questa applicazione è quello di permettere ad un utente di accedere in qualsiasi momento alla propria cartella clinica per visualizzare le informazioni desiderate.
Il software realizzato è una web-app, ovvero un'applicazione progettata per essere accessibile mediante il web da qualsiasi utente.
L'architettura di questo software è stata implementata utilizzando il pattern Model View Controller che, come esplicitato nel Capitolo 1, è il design pattern maggiormente indicato per applicazioni di questa tipologia.

In seguito vengono presentate le implementazioni del lato Client e del lato Server di tale applicazione.

4.1 Client

Il lato Client è stato implementato mediante l'utilizzo del framework Angular JS e dei CSS per definire l'aspetto grafico.
Il front-end segue il paradigma MVC e ciò è reso possibile grazie all'utilizzo di pagine jsp (JavaServer Pages) ovvero una tecnologia di programmazione web in Java che consente di sviluppare la logica di presentazione secondo il pattern MVC.
Le pagine jsp permettono infatti di realizzare l'aspetto grafico, ovvero tutto ciò che viene visualizzato dal client e con il quale egli può interagire.
Ogni pagina jsp viene interfacciata da una classe controller, più precisamente una Servlet, che si occupa di elaborare le richieste del client.

4.1.1 Interfaccia

L'interfaccia è stata progettata ponendo come obiettivo principale la semplicità di interazione.
L'applicazione presentata in questo elaborato è orientata all'ambito medico, nel quale l'utilizzatore ha la necessità di acquisire le informazioni il più velocemente e facilmente possibile.

Per tale ragione si è scelto di realizzare un'interfaccia sufficientemente semplice e intuitiva.

La Figura 4.1 mostra l'Homepage della web-app descritta in questo elaborato

Figura 4.1: Homepage

L'aspetto grafico delle diverse pagine jsp è stato realizzato mediante l'utilizzo dei CSS, e si è scelto di adottare colori tenui che facciano risaltare le importanti informazioni del client.

Interfaccia del Medico

Figura 4.2: Area privata del Medico - Lista dei Pazienti

La Figura 4.2 mostra l'area personale del medico dalla quale egli può visualizzare la lista dei pazienti registrati nell'applicazione e, se ne ha la necessità, può inserire un nuovo utente.

Figura 4.3: Cartella Clinica Del Paziente

Nella propria area personale il medico può visualizzare la lista dei pazienti registrati nell'applicazione (come mostrato in Figura 4.2) e, per ognuno di essi, può scegliere di eliminarlo, effettuare una visita oppure accedere alla sua cartella clinica.
La Figura 4.3 mostra la cartella clinica di un paziente vista dall'area personale del medico. Da qui il medico può visualizzare tutte le visite del paziente, effettuare l'upload o il download di eventuali referti, oppure calcolarne il rischio cardiologico.

Interfaccia del Paziente

Figura 4.4: Area privata del Paziente

La Figura 4.4 mostra l'area personale del paziente, dalla quale egli può visionare le visite che gli sono state effettuate, può effettuare l'upload e il download di eventuali re-

ferti medici e visionare il proprio rischio cardiologico.

Figura 4.5: Rischio Cardiologico del Paziente

Inoltre, come mostrato in Figura 4.5, il paziente può visionare il proprio rischio cardiologico.
L'applicazione infatti restituisce un messaggio che riporta tale rischio associando ad esso un colore significativo come definito nell'indice di Figura 1.5.

4.2 Server

Il lato Server è stato implementato nel linguaggio di programmazione Java adottando i framework Spring e Hibernate.
Come già accennato in precedenza nella fase di progettazione si è scelto di adottare il pattern MVC in quanto più indicato per la realizzazione di applicazioni web.
Al fine di implementare tale architettura è stato utilizzato il framework Spring, o più precisamente, il modulo Spring MVC.
Spring è una piattaforma basata su una struttura modulare, il che lo rende leggero e facilmente integrabile con altri framework, quali Hibernate.
Nella web-app in questione infatti, la persistenza dei dati è stata gestita mediante il framework Hibernate che ha la proprietà di presentare i dati relazionali come oggetti di Java e di mapparli all'interno del database mediante lo strumento ORM (Object Relational Mapping).
Questa sua caratteristica rende Hibernate un framework molto utile in quanto garantisce una gestione semplice e veloce del database.

I framework Spring e Hibernate vengono illustrati in modo accurato nel capitolo sulle Tecnologie Utilizzate.

In seguito vengono presentati: il Diagramma delle classi e la Base Di Dati.

4.2.1 Diagramma delle Classi

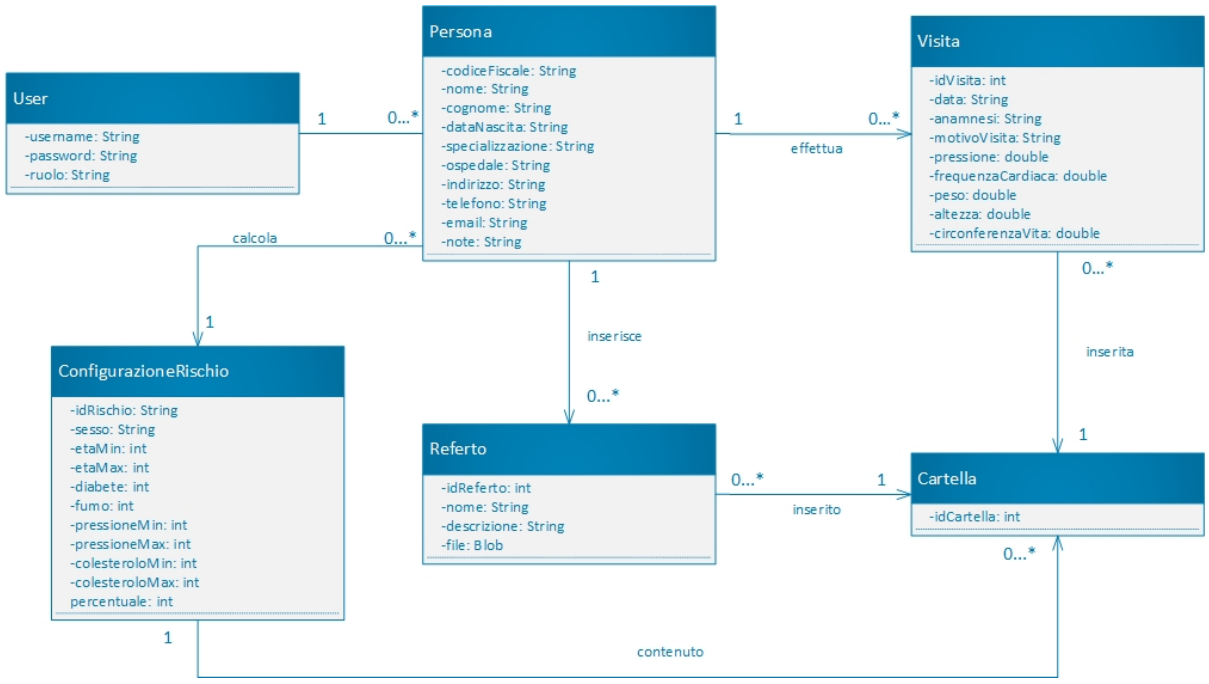

Figura 4.6: Diagramma Delle Classi

4.3 Base Di Dati

Nella fase di progettazione della base di dati sono state inizialmente individuate le entità principali e le relazioni esistenti tra loro.
Questi legami vengono rappresentati in modo schematico mediante il diagramma E-R (Entity Relationship) presentato in Figura 4.7:

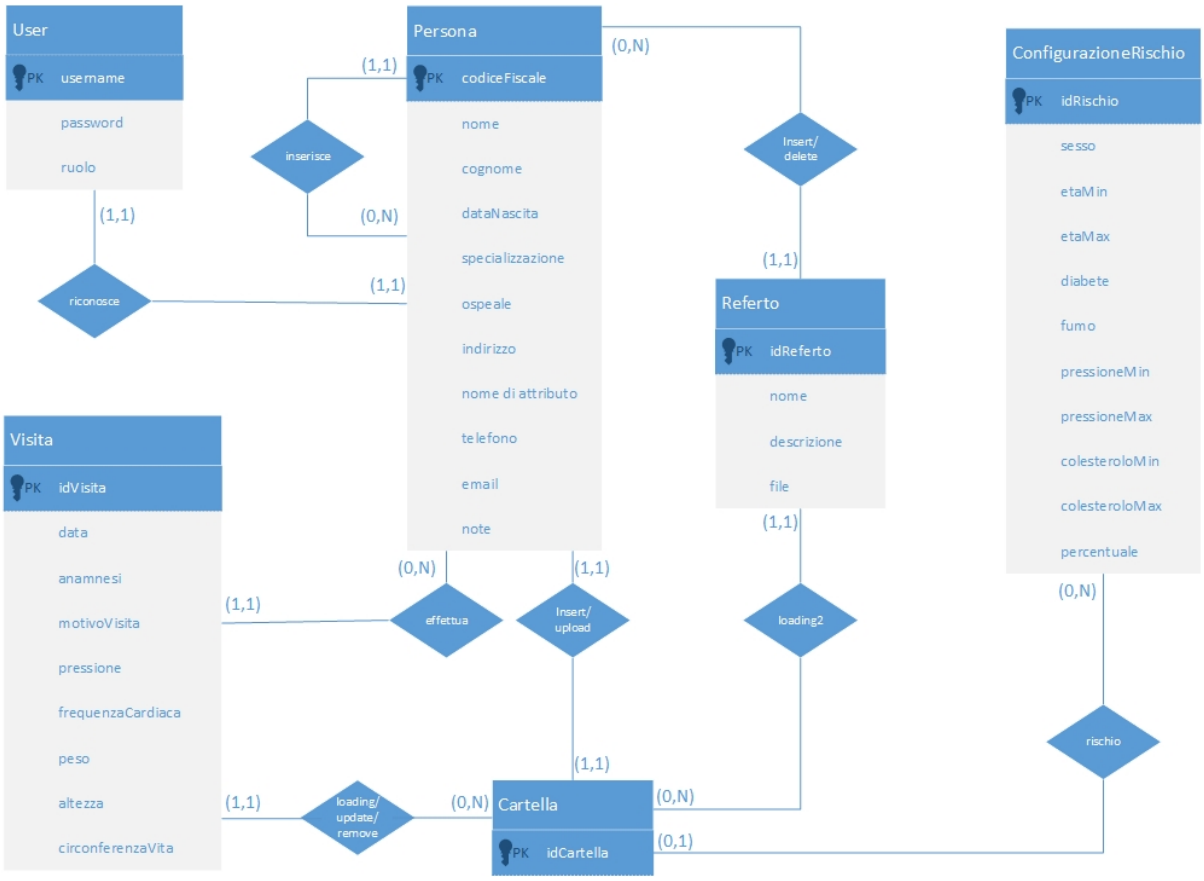

Figura 4.7: Diagramma E-R

In tale modello le entità vengono rappresentate mediante dei rettangoli e ne vengono definiti gli attributi. Gli attributi che rappresentano la chiave primaria vengono evidenziati mediante il simbolo "PK" preceduto da una chiave. Le relazioni esistenti tra le diverse entità vengono rappresentate mediante i rombi.

Successivamente il modello E-R è stato tradotto nel modello relazionale presentato in Figura 4.8.

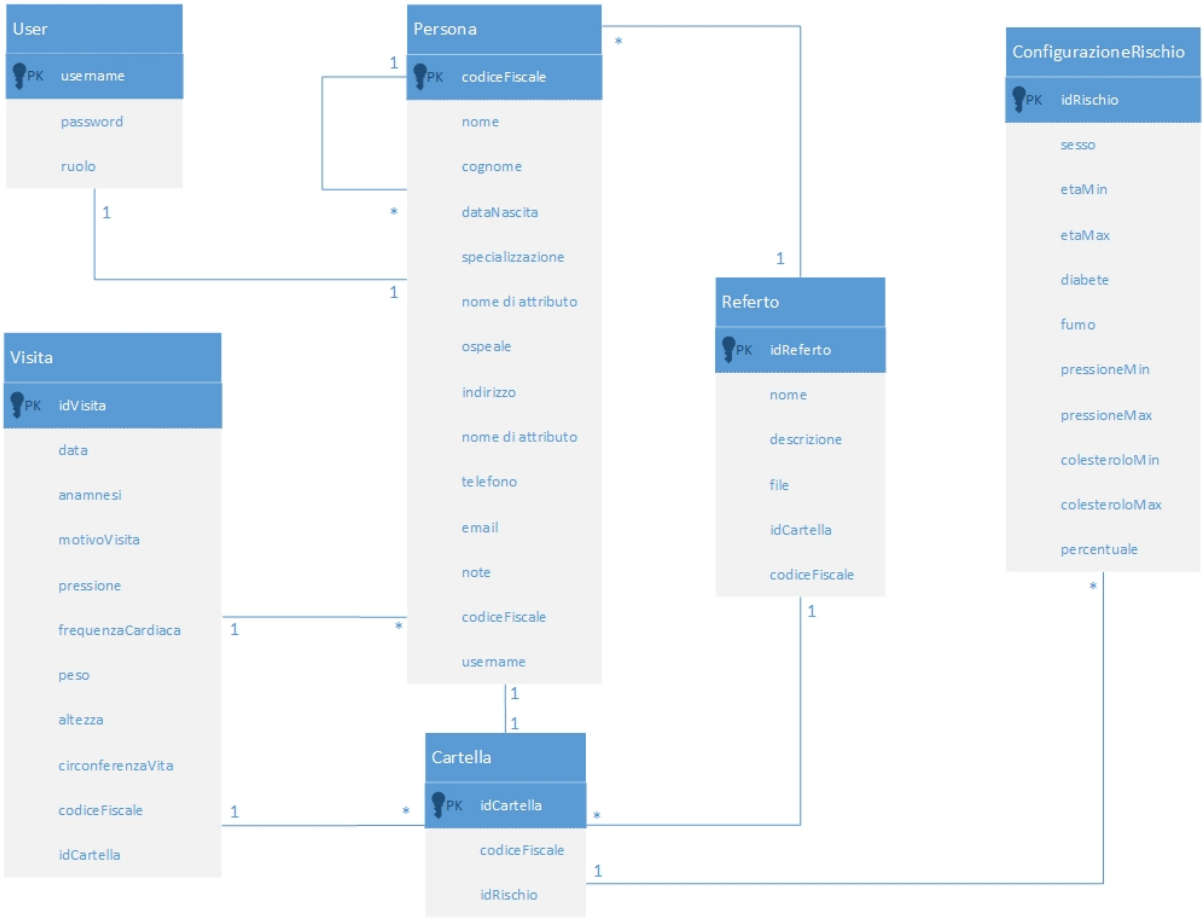

Figura 4.8: Modello Relazionale

Tale modello è una rappresentazione logica della struttura del database.
Tutti i dati sono rappresentati come un insieme di relazioni o tabelle.
Le colonne di una tabella corrispondono agli attributi dell'entità che essa rappresenta, mentre le righe corrispondono alle tuple (o record), ovvero le informazioni contenute.

Per chiare tale concetto si introduce il seguente esempio:

Si consideri la tabella User così definita:

Username	Password	Ruolo
lucaRossi	prova	paziente
user	psw123	medico

Le colonne di tale tabella rappresentano gli *attributi*, ovvero Username, Password e Ruolo.

La seconda e la terza riga identificano invece due *record*: il primo ha come username: "lucaRossi", come password: "prova" e come ruolo: "paziente". Mentre il secondo record ha come username: "user", come password:" psw123 " e come ruolo: "medico".

L'implementazione della base di dati è stata effettuata mediante il linguaggio SQL (Structured Query Language). Quest'ultimo è un linguaggio per la gestione di di database basati sul modello relazionale.

Nello specifico, per l'implementazione della base di dati, è stato utilizzato uno script nel linguaggio SQL, mediante il quale sono state definite le tabelle del database con i relativi attributi e vincoli su di essi.

Capitolo 5

Conclusioni

L'esperienza di tirocinio svolto presso l'azienda Sync Lab Srl è stata molto significativa in quanto ha potenziato le mie competenze in ambito informatico.
Il progetto descritto in questo elaborato mi ha fornito l'opportunità di lavorare, per la prima volta, allo sviluppo Full-Stack di un'applicazione, ovvero alla realizzazione del back-end e del front-end della web-app.
Un ulteriore aspetto positivo di questa esperienza è stata la possibilità di apprendere e implementare nello sviluppo dell'applicazione i due framework Spring e Hibernate.

L'applicazione sviluppata può rappresentare una soluzione efficiente per la virtualizzazione di una cartella clinica.
Grazie a questa applicazione infatti, tutti i pazienti di un ospedale, così come i medici, potrebbero accedere in qualsiasi momento all'applicazione e acquisire le informazioni desiderate.

La web-app sviluppata, inoltre, potrebbe fornire una duplice soluzione ai problemi di inquinamento e di archiviazione dei dati.
Al giorno d'oggi una delle tematiche maggiormente discusse è quella dell'impatto ambientale.
Il consumo di carta, infatti, ha un impatto negativo sull'ambiente in diverse modalità, tra cui la produzione di significanti quantità di rifiuti, l'uso di risorse naturali come alberi, acqua e, combustibili fossili con il conseguente rilascio di sostanze inquinanti nell'atmosfera.
La virtualizzazione dei dati ospedalieri, quali referti medici, documenti e simili, potrebbe rivelarsi significativa per la riduzione di tali problematiche.
Inoltre l'archiviazione fisica dei documenti, come quelli ospedalieri richiede grandi quantità di spazio e tempi di consultazione rilevanti.
La virtualizzazione di tali documenti permetterebbe quindi di ridurre la necessità di avere spazi di dimensioni rilevanti con il solo scopo di conservare documenti, e garantirebbe un accesso ai dati in tempi brevi.