**Università degli Studi di Salerno**

**Corso di Ingegneria del Software**

**CheckUp –**

# SYSTEM DESIGN DOCUMENT


Data 08/01/2021

Partecipanti:

| *Cognome e Nome* | *Matricola* |
|---|---|
| **Tamburello Martina** | 0512105452 |

Revision History:

| *Data* | *Versione* | *Cambiamenti* | *Autori* |
|---|---|---|---|
| 10/01/2021 | 1.0 | Stesura documento | Tamburello Martina |
| 13/01/2021 | 1.1 | Correzione class diagram | Tamburello Martina |
| 20/01/2021 | 1.2 | Modifica dati persistenti | Tamburello Martina |
| 05/02/2021 | 1.3 | Correzione servizi sottosistemi | Tamburello Martina |

# Sommario

# 1. Introduzione

## 1.1 Scopo del sistema

In un momento così particolare della nostra storia, in cui, a causa dell'eccessiva pressione sul sistema sanitario, l'accesso elle strutture sanitarie è notevolmente ridotto, il sistema si propone di agevolare l'interazione a distanza tra medico e paziente.
Un paziente, in cura presso una specialista per una qualsiasi patologia, è costretto periodicamente ad effettuare particolari esami. Nella situazione attuale il paziente è impossibilitato nel recarsi presso il proprio medico per il controllo di routine degli esami, quindi, paziente e medico sono costretti a comunicare via e-mail.
Il sistema si propone, quindi, di fornire una piattaforma in cui il paziente può caricare i risultati dei propri esami e il medico curante può accedere e visionare tali referti. Il paziente, inoltre, può prenotare visite specialistiche e disdirle all'occorrenza.

## 1.2 Obiettivi di progettazione

Gli obiettivi di progettazione (*Design Goals*) sono organizzati in cinque categorie: Performance, Dependability, Cost, Maintenance, End User and Criteria. I Design Goals identificati nel nostro sistema sono i seguenti:

### *Criteri di performance*

- Tempo di risposta:
  Il software deve consentire una navigazione rapida a tutti i tipi di utenti, quindi tempi di risposta minimi (circa 5 secondi) nello svolgimento delle funzionalità da esso offerte.
- Memoria
  La memoria complessiva del sistema dipenderà dalla memoria utilizzata per il mantenimento del Database.

### *Criteri di affidabilità*

- Robustezza:
  Il sistema informerà l'utente di eventuali input errati attraverso messaggi di errore e consentirà all'utente di correggere tali input.

- Affidabilità:
  Il sistema deve garantire l'affidabilità dei servizi proposti. I risultati visualizzati saranno attendibili. Gli orari disponibili per le prenotazioni saranno quelli inseriti dai medici. Il processo di login da parte di tutti gli utenti sarà gestito in modo affidabile, assicurando il corretto funzionamento del sistema.

- Disponibilità:
  Una volta online, il sistema sarà disponibile per tutti i pazienti, medici e segretari registrati almeno 20h/24h. Le ore di inattività, dovute a manutenzione o mantenimento, dovranno essere concentrate durante la notte.

- Tolleranza ai guasti:
  Il sistema potrebbe essere soggetto a fallimenti dovuti a varie cause tra cui un sovraccarico di dati nel database. A tal fine, periodicamente sarà previsto un salvataggio dei dati sotto forma di codice SQL necessario alla rigenerazione del Database.

- Sicurezza:
  L'accesso al sistema sarà garantito mediante email e password per tutti i segretari e i medici, e tramite codice fiscale e password per tutti i pazienti. Le password verranno inserite nel database in forma criptata e non verranno mostrate in chiaro all'interno del sistema. Ogni utente deve avere il permesso di accedere a tutte e sole le funzionalità riservate al proprio ruolo.

### *Criteri di costi*

- Costo di sviluppo:
  È stimato un costo complessivo di 200 ore per la progettazione e lo sviluppo del sistema.

### *Criteri di manutenzione*

- Estensibilità:
  Grazie all'architettura modulare prevista, sarà possibile aggiungere nuove funzionalità al sistema, dettate dalle esigenze del cliente o dall'avvento di nuove tecnologie.

- Adattabilità:
  Il sistema è adattabile a qualsiasi centro medico.

- Portabilità:

  L'interazione con il sistema avviene tramite browser, quindi possiamo definirlo portabile. Poiché il sistema viene sviluppato come una Web App, esso è accessibile da qualunque dispositivo, che sia esso mobile o meno, purché abbia un browser installato. Questa caratteristica garantisce la portabilità dello stesso.

- Tracciabilità dei requisiti:

  La tracciabilità dei requisiti sarà possibile grazie ad una matrice di tracciabilità, attraverso la quale sarà possibile retrocedere al requisito associato ad ogni parte del progetto. La tracciabilità sarà garantita dalla fase di progettazione fino al testing.

***Criteri di usabilità***

- Usabilità:

  Il sistema guiderà l'utente nelle operazioni che vuole effettuare.
  A tal fine i campi saranno pre-formattati e ogni interfaccia utente avrà un numero limitato di icone. Le icone dovranno essere grandi e con colori in contrasto rispetto allo sfondo, al fine di essere più evidenti all'utente.
  Le interfacce dovranno essere responsive.

- Utilità:

  Il lavoro dell'utente verrà supportato nel miglior modo possibile dal sistema, infatti il paziente potrà evitare di recarsi presso il centro solo per far visionare al medico il risultato di un esame o per effettuare una prenotazione; il segretario potrà avere a disposizione i dati di tutti gli utenti iscritti e potrà gestirli all'interno del sistema; e il medico avrà a disposizione tutte le cartelle cliniche dei suoi pazienti.

## 1.2.1 Design Trade-off

***Performance* vs Memoria:**

Il sistema deve garantire risposte rapide a discapito della memoria utilizzata. Ciò significa che verranno introdotte delle ridondanze per evitare interrogazioni costose in termini di performance.

***Tempo di risposta vs Affidabilità***

Il sistema sarà implementato in modo tale da preferire l'affidabilità al tempo di risposta, in modo tale da garantire una risposta del sistema consistente a discapito del tempo impiegato per produrla.

***Disponibilità* vs Tolleranza ai guasti**

Il sistema deve essere sempre disponibile all'utente in caso di errore di una funzionalità a media o bassa priorità, anche a costo di rendere non disponibile quest'ultima per un lasso di tempo. Ovviamente se questa è una funzionalità core, il sistema verrà messo in manutenzione fin quando il guasto non verrà risolto.

***Criteri di manutenzione* vs Criteri di performance**

Il sistema sarà implementato preferendo la manutenibilità alle performance in modo da facilitare agli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

Di seguito è riportata una tabella che mostra i design goal preferiti nei trade off. Il **grassetto** indica la preferenza.

***Trade Off***

| | |
|---|---|
| **Performance** | Memoria |
| **Affidabilità** | Tempo di risposta |
| **Disponibilità** | Tolleranza ai guasti |
| **Criteri di manutenzione** | Criteri di performance |

## 1.3 Definizioni, acronimi e abbreviazioni

### Definizioni

**Referto:** rappresenta, in generale, un referto medico, che si tratti del risultato di un *esame* o di una *diagnosi*.

**Risultato esame**: rappresenta il referto di un esame caricato sulla piattaforma da un *paziente*. Esso comprende le seguenti informazioni:

**Diagnosi**: rappresenta la relazione aggiunta da un *medico* in seguito ad una visita. In alcuni casi verrà usato il termine *relazione* come sinonimo.

**Cartella clinica**: rappresenta l'insieme di *risultati esami* e *diagnosi* inseriti da un *medico* in riferimento ad un *paziente*.

**Paziente**: rappresenta un paziente della struttura, avente un proprio account in cui saranno specificati:

1. Nome
2. Cognome
3. Data di nascita
4. Luogo di nascita
5. Residenza
6. Codice fiscale
7. Numero di telefono
8. E-Mail
9. Password

**Medico**: rappresenta un medico che lavora presso la struttura, avente un proprio account in cui saranno specificati:

1. Nome
2. Cognome
3. Data di nascita
4. Luogo di nascita
5. Residenza
6. Codice fiscale
7. Numero di telefono
8. E-Mail (@checkUp.it)
9. Password
10. Specializzazione

**Segretario**: rappresenta un segretario che lavora presso la struttura, avente un proprio account in cui saranno specificati:

1. Nome
2. Cognome
3. Data di nascita
4. Luogo di nascita
5. Residenza
6. Codice fiscale
7. Numero di telefono
8. E-Mail (@checkUp.it)
9. Password

**Prenotazione**: rappresenta l'azione effettuata da un *paziente* per richiedere di effettuare una visita presso il centro.

**Orario visita:** rappresenta un orario in cui un *medico* è disponibile per una *prenotazione.*

Quando un utente è **"loggato"** vuol dire che ha effettuato l'autenticazione.

### *Acronimi*

RAD= Requirements Analysis Document (Documento di raccolta dei requisiti)

SDD = System Design Document (Documento di Progettazione del Sistema)

DBMS = Database Management System

GUI = Graphical User Interface (Interfaccia utente)

## 1.4 Riferimenti

Libro:

- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition

Autori:

-- Bernd Bruegge

-- Allen H. Dutoit

Documenti:

- CheckUp_RAD_V1.5.docx – Requirements Analysis Document

## 1.5 Panoramica

**Capitolo 1**

Contiene l'introduzione agli obiettivi del sistema, i design goals, i trade-off e un elenco di definizioni, acronimi e abbreviazioni utili alla comprensione dell'intero documento.

**Capitolo 2**

Contiene la descrizione del sistema Corrente, o un sistema similare di riferimento.

**Capitolo 3**

Contiene la descrizione del sistema che verrà realizzato, degli obiettivi che andrà a realizzare, in cui sarà gestita la decomposizione in sottosistemi, il mapping Hardware/Software, la gestione dei dati persistenti, il controllo degli accessi e della sicurezza, il controllo del flusso globale del sistema e le condizioni limite.

**Capitolo 4**

Contiene la rappresentazione dei servizi dei sottosistemi.

# 2. Architettura del sistema corrente

Attualmente non esiste una piattaforma per il caricamento dei referti o per la gestione delle prenotazioni.
Nella situazione corrente il paziente invia i referti al proprio medico tramite e-mail, invece, la gestione delle prenotazioni avviene telefonicamente o in presenza.

# 3. Architettura del sistema proposto

## 3.1 Panoramica

Il sistema che si vuole realizzare rientra nel campo della Green-field Engineering. Il nuovo sistema "CheckUp" è un'applicazione web, in locale per motivi di sicurezza, che verrà sottoposto a re-engineering al fine di aggiungere nuove funzionalità e migliorare quelle già presenti. Il sistema è rivolto a medici, segretari e pazienti della struttura. Tutti gli utenti registrati potranno effettuare login e log-out, visualizzare i propri dati di profilo e modificare la propria password; i pazienti avranno la possibilità di richiedere la registrazione che dovrà essere approvata, mentre medici e segretari dovranno essere registrati da un segretario.
La piattaforma metterà a disposizione diverse funzionalità, a seconda del tipo di utente:

- Il segretario potrà visualizzare la lista degli utenti che hanno richiesto la registrazione e, per ognuno, decidere di confermare o respingere la registrazione;
  Egli, inoltre, potrà visualizzare la lista dei medici, dei segretari e dei pazienti con la possibilità di visualizzare i dettagli di ognuno ed eliminare o aggiungere un segretario o un medico.
  Infine, potrà visualizzare la lista delle prenotazioni effettuate e, contestualmente, visualizzare i dati del paziente relativo alla prenotazione.
- Il medico potrà gestire il suo orario di disponibilità presso la struttura, visualizzare i referti aggiunti dai suoi pazienti, visualizzare la lista dei suoi pazienti e gestire la cartella clinica di questi ultimi.
- Il paziente potrà effettuare prenotazioni per visite specialistiche e, eventualmente effettuarne la disdetta; potrà caricare risultati di esami da far visionare a uno dei suoi medici e visualizzare le proprie cartelle cliniche.

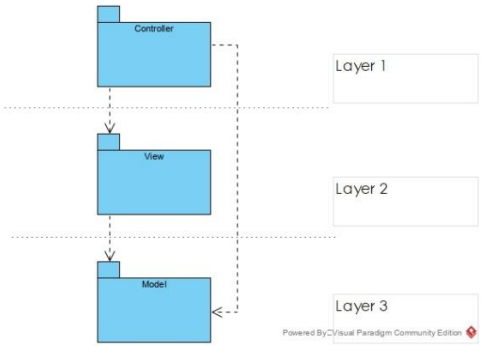
  Lo stile architetturale usato è di tipo repository in quanto i sottosistemi che compongono il software accedono e modificano una singola struttura dati chiamata, appunto, repository: il database. L'architettura permette una gestione centralizzata.

Il sistema implementa un pattern di tipo MVC. Il modello Model View Controller (MVC) è un modello utilizzato per la semplice programmazione basata sugli eventi delle interfacce utente, volto a separare il livello di visualizzazione dalla logica di back-end che gestisce i dati. Si tratta di un'architettura multi-tier, ovvero un'architettura software di tipo client-server per sistemi distribuiti, in cui le varie funzionalità del software sono logicamente separate su più strati o livelli software differenti in comunicazione tra loro (nel caso di applicazioni web questi strati sono la logica di presentazione, l'elaborazione dei processi e la gestione della persistenza dei dati).

## 3.2 Decomposizione del sottosistema

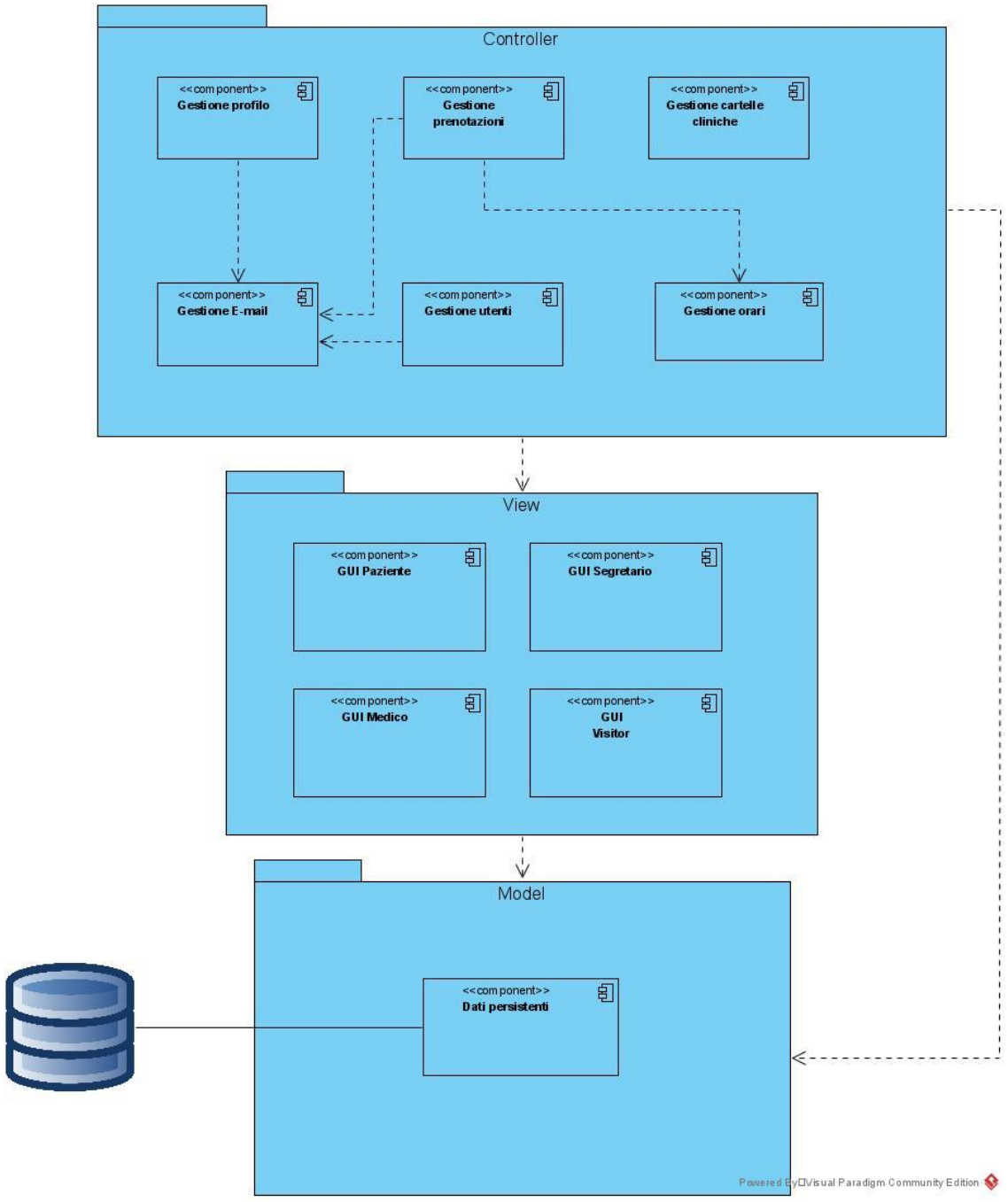
La decomposizione prevista per il sistema è composta da tre layer:

- **Livello di presentazione**
  Questo è il livello più alto dell'applicazione. Il livello di presentazione mostra le informazioni relative a servizi come merce online, acquisti, e i contenuti del carrello della spesa. Comunica con altri livelli attraverso i risultati di output al livello browser/client e tutti gli altri livelli della rete.
- **Livello applicazione (business logic)**
  La logica di primo livello viene tirato fuori dal livello di presentazione e, come suo proprio livello, controlla la funzionalità di un'applicazione eseguendo elaborazioni dettagliate.
- **Livello dati**
  Questo livello è costituito da server database. Qui le informazioni vengono memorizzate e recuperate. Questo livello mantiene i dati neutrali e indipendenti da applicazioni server o da logica di business. Fornendo informazioni del proprio livello inoltre migliora la scalabilità e le prestazioni.



Seguendo le esigenze del sistema, un'*architettura aperta* migliorerebbe l'efficienza, mantenendo comunque un basso accoppiamento tra le componenti. Ciascuna componente inoltre è stata idealizzata per avere un'alta coesione, che permetterà una facile manutenzione all'occorrenza.

## Component Diagram




Il layer **Control** gestisce 5 sottosistemi:

- Gestione Profilo
- Gestione Prenotazione
- Gestione Cartelle cliniche
- Gestione E-mail
- Gestione Utenti
- Gestione Orari

Il layer **View** gestisce 4 sottosistemi:

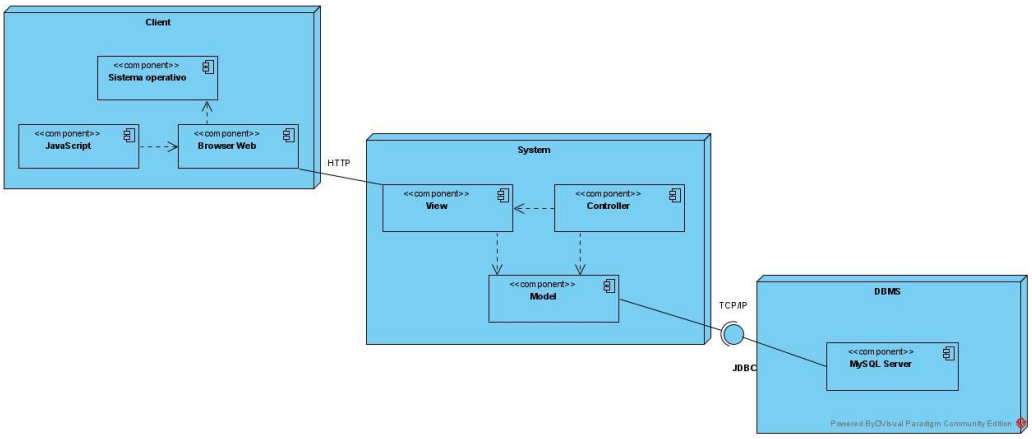
- GUI Paziente
- GUI Medico

- GUI Segretario
- GUI Visitatore

Il layer **Model** gestisce un solo sottosistema: Dati Persistenti, che si occupa della gestione dei dati persistenti del sistema, e si interfaccia con il DBMS.

### Deployment Diagram

L'utente (Client) richiede le funzionalità tramite l'interfaccia che il sistema mette a disposizione a patto che si possieda un browser capace di interpretare JavaScript, in modo che le funzioni definite dal sistema possano essere eseguite in maniera corretta. Il tier del Client connette lo strato di view del Sistema sul quale vengono eseguite le funzioni atte al completamento degli obiettivi del Client. La parte DBMS racchiude e gestisce la persistenza dei dati. Il Sistema viene eseguito sul web server Apache Tomcat.




## 3.3 Mapping Hardware/Software

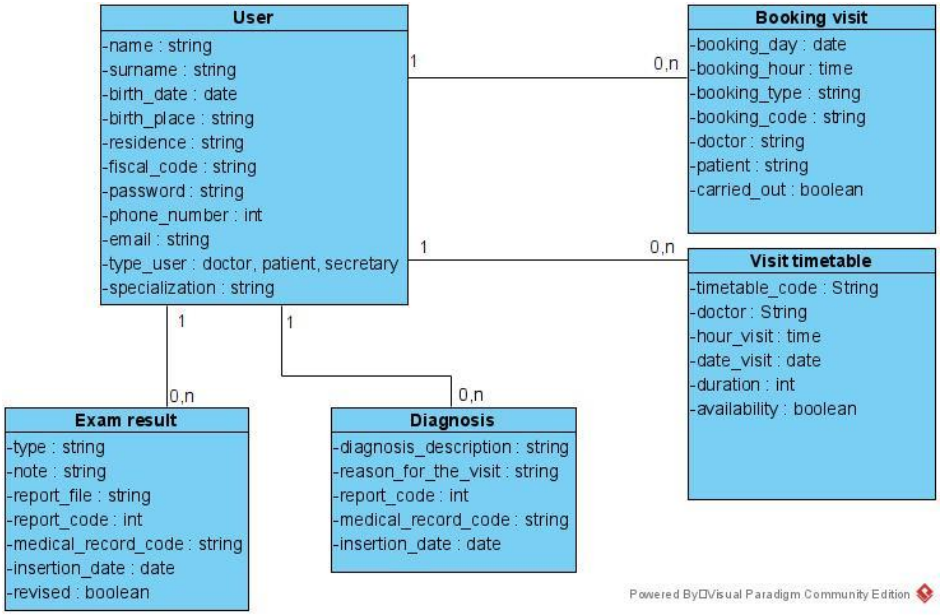
Il sistema che si desidera sviluppare utilizzerà una struttura hardware composta da un Server che risponderà ai servizi richiesti dal client. Il client è una macchina attraverso la quale un utente può collegarsi, usando una connessione a internet, per accedere al sistema mentre la macchina server gestisce la logica e i dati persistenti inseriti nel database. Client e server saranno connessi tramite il protocollo HTTP, con cui il client inoltra richieste al server e provvederà a fornire i servizi richiesti.

Le componenti hardware e software di cui ha bisogno il client sono un computer, un tablet oppure un qualsiasi mobile dotato di connessione internet.

Per quanto riguarda il server, vi è la necessità di avere a disposizione una macchina avente connessione a internet e con capacità di immagazzinare grandi quantità di dati. La componente di cui ci si ha bisogno è un DBMS, in questo caso MySQL, per consentire il salvataggio dei dati in maniera persistente. Inoltre, è necessario usufruire dei servizi offerti da un web server, nello specifico Apache Tomcat, per consentire la comunicazione con più client.

## 3.4 Gestione dati persistenti

Per la memorizzazione dei dati è stato scelto di utilizzare un Database Relazionale, il modello di database più popolare sul mercato, progettato per contenere grandi volumi di dati, per consentire brevi tempi di risposta e ridurre i limiti di spazio di archiviazione. È possibile ripristinare lo stato del database in caso di danni software o hardware attraverso una copia dei dati effettuata periodicamente. I dati presenti nel database sono privatizzati, vale a dire che il DBMS consente un accesso protetto, di conseguenza con operazioni diverse, l'utente, può accedere a diverse sezioni del database.




**User**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **fiscal_code** | VARCHAR(16) | NOT NULL | PRIMARY KEY |
| **user_name** | VARCHAR(20) | NOT NULL | |
| **surname** | VARCHAR(20) | NOT NULL | |
| **birth_date** | DATE | NOT NULL | |
| **birth_place** | VARCHAR(20) | NOT NULL | |
| **residence** | VARCHAR(50) | NOT NULL | |
| **user_password** | VARCHAR(256) | NOT NULL | |
| **phone_number** | BIGINT | NOT NULL | |
| **e-mail** | VARCHAR(50) | NOT NULL | KEY |
| **type_user** | ENUM('doctor', 'patient', 'secretary') | NOT NULL | |
| **specialization** | VARCHAR(50) | | |

**UserNotVerified**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **fiscal_code** | VARCHAR(16) | NOT NULL | PRIMARY KEY |
| **user_name** | VARCHAR(20) | NOT NULL | |
| **surname** | VARCHAR(20) | NOT NULL | |
| **birth_date** | DATE | NOT NULL | |
| **birth_place** | VARCHAR(20) | NOT NULL | |
| **residence** | VARCHAR(50) | NOT NULL | |
| **user_password** | VARCHAR(256) | NOT NULL | |
| **phone_number** | BIGINT | NOT NULL | |
| **e-mail** | VARCHAR(50) | NOT NULL | KEY |

**BookingVisit**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **booking_code** | VARCHAR(100) | NOT NULL | PRIMARY KEY |
| **doctor** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **patient** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **booking_day** | DATE | NOT NULL | |
| **booking_hour** | TIME | NOT NULL | |

| | | | |
|---|---|---|---|
| **booking_type** | VARCHAR(50) | NOT NULL | |
| **carried_out** | BOOLEAN | NOT NULL | |

**ExamResult**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **exam_code** | INT | NOT NULL | PRIMARY KEY |
| **doctor** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **patient** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **revised** | BOOLEAN | NOT NULL | |
| **report_type** | VARCHAR(50) | NOT NULL | |
| **note** | VARCHAR(200) | | |
| **report_file** | VARCHAR(500) | NOT NULL | |
| **medical_record_code** | VARCHAR(100) | NOT NULL | |
| **insertion_date** | DATE | NOT NULL | |

**Diagnosis**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **diagnosis_code** | INT | NOT NULL | PRIMARY KEY |
| **doctor** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **patient** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **medical_record_code** | VARCHAR(100) | NOT NULL | |
| **diagnosis_description** | VARCHAR(500) | NOT NULL | |
| **reason_for_the_visit** | VARCHAR(250) | NOT NULL | |
| **insertion_date** | DATE | NOT NULL | |

**VisitTimetable**

| *NOME* | *TIPO* | *VINCOLI* | *CHIAVE* |
|---|---|---|---|
| **timetable_code** | INT | NOT NULL | PRIMARY KEY |
| **doctor** | VARCHAR(16) | NOT NULL | FOREIGN KEY |
| **hour_visit** | TIME | NOT NULL | |
| **date_visit** | DATE | NOT NULL | |
| **duration** | INT | NOT NULL | |
| **availability** | BOOLEAN | NOT NULL | |

## 3.5 Controllo degli accessi e sicurezza

Diversi attori possono interagire con il sistema, ognuno con ruolo e funzionalità diversi. Si utilizza, quindi, una matrice degli accessi per capire quali attori possono eseguire quali funzioni. Quando un attore prova ad effettuare un'operazione non consentitagli, gli viene mostrata una pagina di errore.

Nella seguente tabella verranno riportate:

- in alto, un'astrazione delle istanze delle classi del sistema;
- sul lato sinistro, gli attori;
- la cella che incrocia attore e istanza rappresenta cioè che l'attore può eseguire su quell'istanza.

| | ***Paziente*** | ***Medico*** | ***Segretario*** | ***Risultato esame*** | ***Diagnosi*** | ***Cartella clinica*** | ***Prenotazione*** | ***Orari*** |
|---|---|---|---|---|---|---|---|---|
| Utente non registrato | 1) Richiesta di registrazione | | | | | | | |
| Paziente | 1) Visualizza dati personali;<br>2) Modifica password; | | | 1) Visualizza;<br>2) Aggiunge; | 1) Visualizza; | 1) Visualizza; | 1) Effettua;<br>2) Visualizza;<br>3) Elimina; | 1)Visualizza; |
| Medico | 1) Visualizza; | 2) Visualizza dati personali;<br>3) Modifica password; | | 1) Visualizza;<br>2) Sposta nella cartella clinica; | 1) Visualizza;<br>2) Aggiunge; | 1) Visualizza;<br>2) Aggiunge referti o diagnosi; | | 1)Aggiunge;<br>2)Visualizza; |
| Segretario | 1) Visualizza; | 1) Visualizza;<br>2) Elimina ;<br>3) Registra; | 1) Visualizza dati personali;<br>2) Modifica password;<br>3) Visualizza;<br>4) Elimina;<br>5) Registra; | | | | 1) Visualizza; | |

## 3.6 Controllo flusso globale del sistema

Il flusso del sistema di "CheckUp" richiede una continua interazione dell'utente, per cui il controllo del flusso globale del sistema è di tipo event-driven, ovvero il flusso del programma è largamente determinato dal verificarsi di input dell'utente.

Per quanto riguarda la concorrenza, le funzionalità offerte dal Web Server, garantiscono un'interazione concorrente con tutti gli utenti connessi al sistema. Infatti, ogni utente connesso al sistema, tramite il suo browser web, avrà un thread dedicato tramite il quale il server interagirà con lui.

## 3.7 Condizioni limite

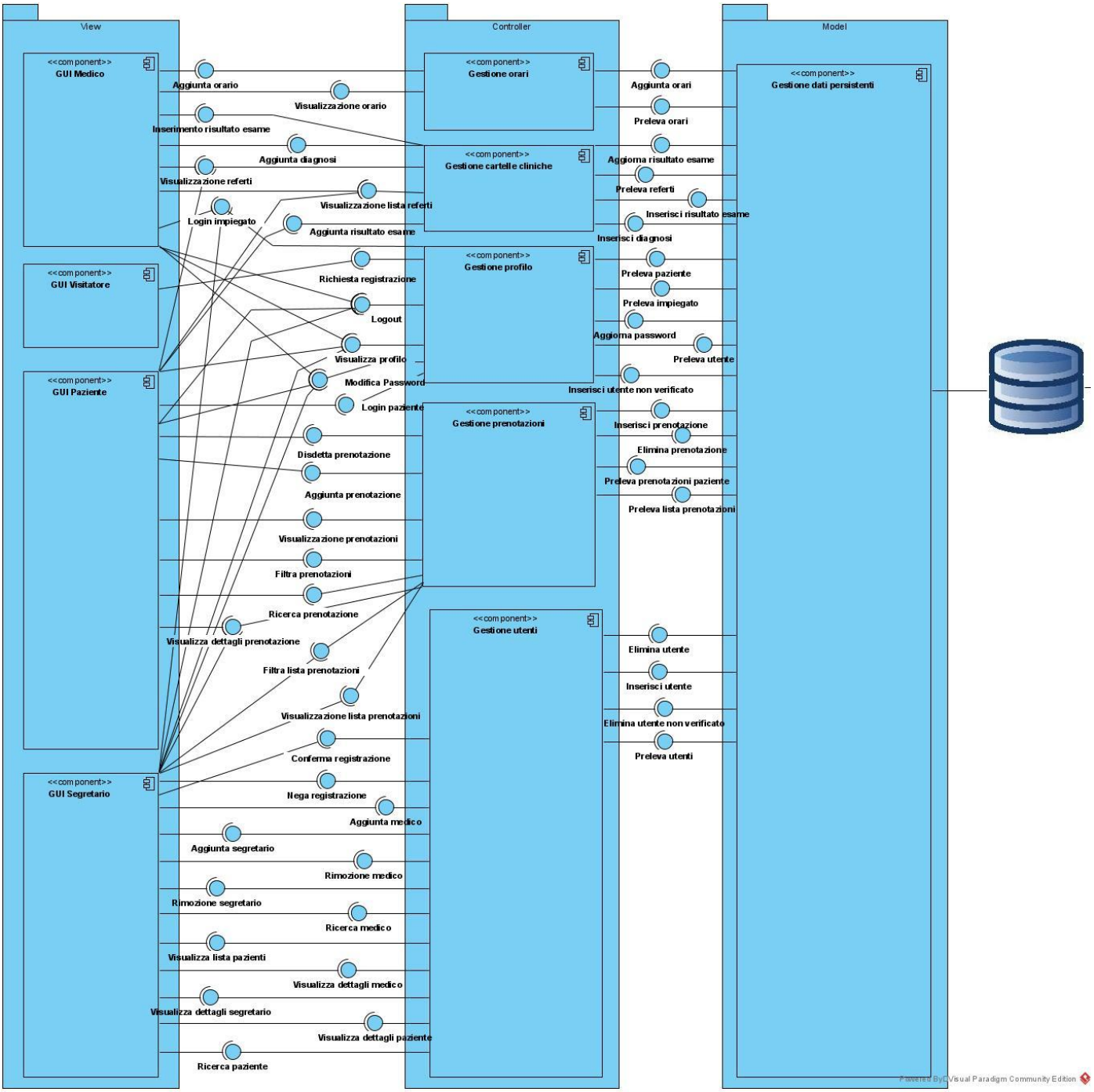
**Start-up**
Lo start-up del sistema prevede l'avvio del web server nel quale il sistema è installato e l'avvio del DBMS per accedere ai dati persistenti memorizzati nel database. Quando sia Web Server che DBMS sono in esecuzione, il sistema carica in memoria centrale le servlet principali attraverso le quali gli utenti possono effettuare le operazioni. Dopo l'avvio del sistema, l'utente può interagire con esso.

**Shut-down**
Quando il sistema deve essere arrestato, il gestore del sistema termina l'esecutivo del web server. Quando ciò avviene tutte le risorse che il sistema utilizza (connessione al database e connessione alla rete) vengono rilasciate, e nessun utente può più connettersi al sistema.

**Fallimento**

1. Nel caso di un'interruzione inaspettata dell'alimentazione, non vi sarebbero metodi che possano ripristinare lo stato del sistema precedente allo spegnimento non voluto. Qualsiasi transazione con il database verrebbe annullata e verrebbe ripristinato lo stato consistente più recente delle informazioni persistenti.
2. In caso di guasti dovuti al sovraccarico di informazioni al database, la rete verrebbe congestionata. Il Web Server percepirebbe questo stato e inviterebbe tutti i client connessi a ri-eseguire le operazioni in un secondo momento.
3. Nel caso di una chiusura inaspettata del software, dovuta a errori avvenuti durante la fase di implementazione, il server risponderebbe con una pagina di errore.
4. Nel caso in cui si verificasse un sovraccarico di richieste al server, questo rimarrà congestionato.
5. Se un utente dovesse inviare al server informazioni errate (volutamente o meno), o che non permettono la corretta esecuzione di un'operazione, il server risponderebbe con una pagina di errore.
6. Nel caso di errore critico dell'hardware, non è prevista una soluzione.

# 4. Servizi dei sottosistemi




## Servizi offerti dal controller per il sottosistema view GUI Medico:

- Aggiunta orario;
- Visualizzazione orario;

## Servizi offerti dal controller per il sottosistema view GUI Medico:

- Login impiegato
- Logout
- Visualizza profilo
- Modifica password
- Aggiunta orario;

- Visualizzazione orario;
- Inserimento risultato esame nella cartella clinica
- Aggiunta diagnosi
- Visualizzazione lista referti
- Visualizzazione referto
- Visualizza lista pazienti

## Servizi offerti dal controller per il sottosistema view GUI Visitatore:

- Richiesta registrazione

## Servizi offerti dal controller per il sottosistema view GUI Paziente:

- Login paziente
- Logout
- Visualizza profilo
- Modifica password
- Effettua prenotazione
- Disdetta prenotazione
- Visualizza lista prenotazioni
- Filtra prenotazioni
- Aggiunta risultato esame
- Visualizzazione lista referti
- Visualizzazione referto
- Visualizza dettagli prenotazione

## Servizi offerti dal controller per il sottosistema view GUI Segretario:

- Login impiegato
- Logout
- Visualizza profilo
- Modifica password
- Aggiunta medico
- Rimozione medico
- Aggiunta segretario
- Rimozione segretario
- Visualizza lista segretari
- Visualizza lista pazienti
- Visualizza lista medici
- Visualizza lista prenotazioni
- Visualizza dettagli segretario
- Visualizza dettagli medico
- Visualizza dettagli paziente
- Filtra prenotazioni
- Conferma registrazione
- Respingi registrazione
- Ricerca medico
- Ricerca paziente

## Servizi offerti dal model per il sottosistema controller Gestione orari:

- Inserisci orari
- Preleva orari

## Servizi offerti dal model per il sottosistema controller Gestione cartelle cliniche:

- Aggiorna risultato esame
- Preleva referti
- Inserisci risultato esame
- Inserisci diagnosi

## Servizi offerti dal model per il sottosistema controller Gestione profilo:

- Preleva paziente
- Preleva impiegato
- Aggiorna password
- Preleva utente

## Servizi offerti dal model per il sottosistema controller Gestione prenotazioni:

- Inserisci prenotazione
- Elimina prenotazione
- Preleva prenotazioni paziente
- Preleva lista completa prenotazioni

## Servizi offerti dal model per il sottosistema controller Gestione utenti:

- Elimina utente
- Inserisci utente
- Elimina utente non verificato
- Preleva utenti