Hilo

Everyone wanna be a Hilo

System Design Document

Riferimento	
Versione	1.1
Data	06/12/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	**H**ermann Senatore, **I**van Carmine Adamo, **L**orenzo Criscuolo, **O**razio Cesarano
Approvato da	

Revision History

DATA	Versione	Cambiamenti	Autori
01/12/2020	0.1	Prima realizzazione SDD	[tutti]
02/12/2020	0.2	Aggiunta capitoli 1,2,3	[Orazio, Ivan]
03/12/2020	0.3	Aggiunti sottosistemi	[Ivan]
04/12/2020	0.4	Raffinamenti al capitolo 3	[tutti]
06/12/2020	1.0	Revisione	[tutti]
21/12/2020	1.1	Modificati Sottosistemi e Servizi	[Lorenzo, Orazio]

Sommario

1. Introduzione

1.1 Obiettivi del sistema

Il sistema da realizzare ha come obiettivo la gestione dei dati dell'Azienda Sanitaria Locale, relativi alla pandemia da COVID-19.
Si vuole realizzare un sistema che consenta una facile comunicazione medico-paziente sia che quest'ultimo si trovi ricoverato in una struttura sanitaria, sia che si trovi in quarantena domiciliare. Ulteriore obiettivo del sistema proposto è quello di rendere l'analisi dei tamponi più efficiente sfruttando un modulo di IA che consente di creare una coda di tamponi da analizzare più efficiente e più coerente all'obiettivo finale dell'analisi, cioè capire effettivamente quanti sono i positivi al SARS-CoV-2.
Infine, il sistema sarà in grado di fornire una statistica giornaliera dell'andamento dell'epidemia sul territorio.

L'attuale sistema adottato risulta lento, macchinoso e non suscita fiducia nella popolazione per via della scarsa, e a volte inesistente, comunicazione tra ASL e paziente.

Il sistema proposto è una web application a cui avranno accesso i seguenti utenti definiti in fase di analisi:

- **Pazienti:** sia ricoverati che in quarantena domiciliare.
- **Operatori Sanitari:** medici, analisti ed infermieri.
- **Amministratori del sistema**.

Principalmente, il sistema si occuperà della gestione persistente dei dati del paziente e dei tamponi.
Il sistema dovrà consentire il login per tutte e tre le figure individuate, a seconda dei dati immessi.

Con un meccanismo di comunicazione (e-mail e/o SMS), il paziente verrà informato della fine dell'analisi del suo tampone e potrà prendere atto dei risultati facendo il login sulla piattaforma e visualizzando il suo profilo.
Il sistema dovrà consentire l'immissione, da parte degli operatori sanitari, di:

- Risultati dei tamponi;
- Terapia del paziente;
- Misurazioni della febbre;
- Note dell'operatore;
- Sintomi;

Il sistema dovrà consentire l'immissione, da parte dei pazienti domiciliati, di:

- Sintomi;
- Misurazioni della febbre;
- Note del paziente;
- Stato mentale del paziente;

Infine, il sistema dovrà essere in grado di presentare giornalmente i dati riguardanti l'andamento dell'epidemia sul territorio.

1.2 Design Goals

I Design Goals sono organizzati in cinque categorie: Performance, Dependability, Cost, Maintenance, End User Criteria. I Design Goals identificati nel nostro sistema sono i seguenti:

1.2.1. Criteri di performance

- Tempo di risposta:

  Il sistema deve elaborare in tempi minimi una qualsiasi sottomissione effettuata dal paziente, in qualsiasi momento della giornata essa venga effettuata.

- Memoria:

  Per il corretto funzionamento, è richiesto uno spazio di archiviazione minimo sia in termini di dimensioni della piattaforma, sia per le dimensioni della base di dati, in quanto non è previsto un utilizzo massiccio di una memoria di massa; gli allegati delle radiografie o altri possibili file caricati, di fatto, non hanno dimensioni considerevoli.

1.2.2. Criteri di dependability

- Robustezza:

  Il sistema dovrà esser in grado di sopravvivere anche ad input non validi immessi dall'utente, o a condizioni precarie dell'environment.

- Affidabilità:

Il sistema deve eseguire le funzioni richieste sotto determinate condizioni per un periodo di tempo prestabilito.

- Disponibilità:

  Dopo aver effettuato il deploy del server, il sistema dovrà essere disponibile per un periodo significativo di tempo, che comprende qualsiasi periodo della giornata, salvo eventuali manutenzioni del server.

- Tolleranza ai guasti:

  Il sistema deve garantire la capacità di operare anche sotto condizioni di errore dovute, ad esempio, ad un sovraccarico di informazioni nel database.

- Sicurezza:

  L'accesso al sistema è previsto mediante utilizzo di username e password, di cui dovrà essere garantita la resistenza ad attacchi di malintenzionati. Per ogni tipologia di utente dovranno essere gestiti controlli massicci, onde evitare utilizzi da parte di utenti non autorizzati.

1.2.3. Criteri di costo

- Costo di sviluppo:

  È stimato un costo complessivo di 200 ore per la progettazione e lo sviluppo del sistema (50 per ogni project member).

1.2.4. Criteri di manutenzione

- Estensibilità:

  Il sistema dovrà consentire l'aggiunta di nuove funzionalità o nuove classi a sviluppatori futuri, in base alle esigenze del cliente.

- Modificabilità:

  Il sistema dovrà consentire, mediante consultazione della documentazione, la modifica di funzionalità già presenti.

- Adattabilità:

  È previsto lo sviluppo del sistema esclusivamente per l'attuale pandemia del Covid-19, ma il software non include vere e proprie

specifiche di tale ambito, per cui sarà possibile adattarlo a differenti domini applicativi.

- Portabilità:

  Il sistema verrà sviluppato come web application, testato principalmente sul browser Google Chrome. Tuttavia, gli standard moderni consentono al sistema di poter funzionare correttamente anche su browser differenti.

- Tracciabilità dei requisiti:

  Una matrice di tracciabilità consentirà il mapping semplificato del codice nei requisiti specifici.

1.2.5. Criteri di usabilità

- Usabilità:

  Il sistema sarà facile da comprendere, in modo tale da consentire all'utente di imparare ad operare agilmente, grazie ad un'interfaccia user-friendly.

- Utilità:

  Il sistema sarà estremamente utile, in quanto velocizza considerevolmente il processo di gestione dei tamponi e dei pazienti, consentendo l'intervento immediato da parte del personale medico

1.3. Design Trade-off

Functionality vs Usability: Il sistema deve consentire una maggiore usabilità a discapito delle funzionalità previste nel RAD. Chiaramente, in base alle esigenze del cliente verranno mantenute ed implementate le funzionalità in base alle priorità assegnatagli.

Cost vs Robustness: Il sistema verrà sviluppato favorendo la robustezza a discapito dei costi. Alla base di tale decisione vi è l'esigenza che il sistema sia estremamente robusto, in quanto il corretto funzionamento del sistema (anche se soggetto ad input errati o a condizioni precarie) ha un impatto concreto sia sui pazienti che sui medici che devono operare in prima persona con i dati forniti da esso.

Efficiency vs Portability: L'efficienza dovrà essere un punto chiave dello sviluppo del sistema. Sebbene il sistema sia "basato" sulla portabilità, nel caso in cui sarà necessario effettuarne un trade-off, verrà valutata maggiormente l'efficienza, in quanto è più importante che il sistema risponda correttamente alle richieste dell'utente anziché visualizzare, ad esempio, correttamente le pagine.

Rapid development vs Functionality: In base al criterio di *costo di sviluppo* ed alla deadline prestabilita (ed alle priorità assegnate ai requisiti funzionali del sistema), verrà considerato maggiormente il rapido sviluppo della piattaforma a discapito di alcune funzionalità ritenute non essenziali dal cliente.

Cost vs Reusability: Il sistema da sviluppare essenzialmente non dispone di una versione precedente, per cui non esistono vere e proprie componenti legacy riutilizzabili. Si cercherà di mantenere i costi di produzione entro un certo limite e di riutilizzare in futuro il più possibile le componenti ad-hoc realizzate.

Response Time vs Reliability: Il sistema sarà implementato in modo tale da preferire l'affidabilità al tempo di risposta, onde garantire un controllo più accurato dei dati in input a discapito del tempo di risposta del sistema.

1.4. Definizioni, acronimi e abbreviazioni

RAD: Requirements Analysis Document.

SDD: System Design Document.

USER-FRIENDLY: Letteralmente "amichevole per l'utente", di facile utilizzo anche per chi non è esperto.

DB: DataBase, ovvero "Base di Dati".

MySQL: È un RDBMS Open Source basato sul linguaggio SQL, composto da un client a riga di comando e un server.

DBMS: DataBase Management System.

SQL: Structured Query Language; linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per: creare e modificare schemi di database.

RESPONSIVE: Tecnica di web design per la realizzazione di siti in grado di adattarsi graficamente in modo automatico al dispositivo coi quali vengono visualizzati, riducendo al minimo la necessità dell'utente di ridimensionare e scorrere i contenuti.

1.5. Riferimenti

- *Requirements Analysis Document*: NC_04 RAD_V_1.0
- Slide del corso, presenti sulla piattaforma e-learning.

- *Libro di testo*: Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition.

Autori: Bernd Bruegge & Allen H. Dutoit.

1.6. Panoramica

Capitolo 1: Contiene una panoramica del sistema software, dei suoi design goals. Sono presenti definizioni, acronimi e abbreviazioni e riferimenti ad altri documenti al fine di produrre una documentazione che possa essere chiara.

Capitolo 2: Vengono rese esplicite le funzionalità del sistema corrente, elencandone le criticità che hanno portato alla creazione di un nuovo sistema attualmente in progettazione.

Capitolo 3: Viene presentata l'architettura del sistema software proposto. Più dettagliatamente, viene descritta la decomposizione del sistema in sottosistemi e le responsabilità di ognuno di essi; inoltre, viene presentato il mapping hardware/software, la gestione dei dati persistenti, il controllo degli accessi e sicurezza, il controllo del flusso globale del sistema, ed infine le condizioni limite.

Capitolo 4: Vengono presentati i servizi dei sottosistemi.

2. Architettura del Sistema corrente

Attualmente non esiste una vera e propria piattaforma informatica digitalizzata atta alla gestione della pandemia, e in particolare della gestione dei tamponi e delle condizioni psicologiche dei pazienti, sebbene una piattaforma di proprietà della Regione Campania "SINFONIA" sia stata realizzata ma è limitata solo alle comunicazioni dei risultati. La gestione dei tamponi risulta essere del tutto analogica per cui soggetta a rallentamenti ed errori umani che possono essere risolti in gran parte o addirittura eliminati con una buona piattaforma ad-hoc, che permetta di semplificare la situazione drammatica attualmente in corso, sia per il personale medico che per i pazienti.

3. Architettura del Sistema proposto

3.1 Panoramica

Il sistema da noi proposto è una web application che permetterà la facile interazione tra il personale medico e i loro pazienti positivi al covid-19. Gli utenti della piattaforma sono sostanzialmente classificati in due categorie: Pazienti e personale medico. Il paziente accede alla propria area utente ed ha la possibilità di sottomettere dati o altri file che potrebbero interessare il personale medico, può fornire informazioni riguardanti il proprio stato emotivo e come sta reagendo alla terapia proposta; il

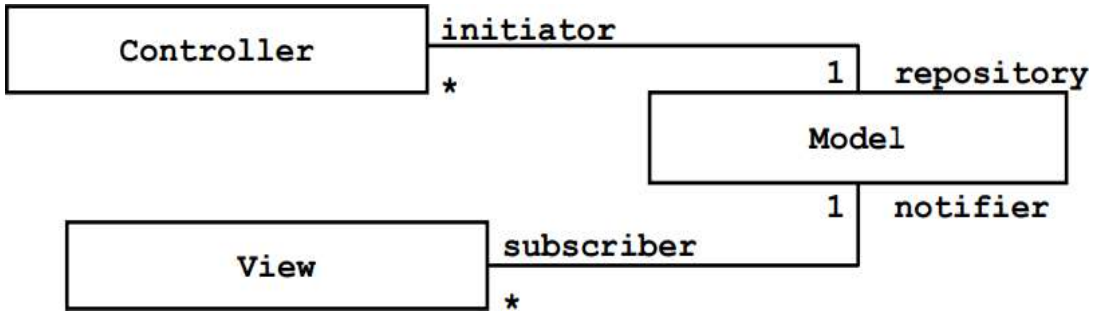
personale medico accede anch'esso dalla pagina adibita al login utente e può visualizzare il profilo dei propri pazienti, aggiornare la cartella clinica, consultare i file da loro caricati, aggiungere radiografie al loro profilo, decidere se fornire supporto psicologico nel caso in cui il paziente risulti particolarmente demotivato o scombussolato dagli eventi clinici. Lo stile architetturale scelto per il sistema è di tipo *repository*, in quanto sono adatti per applicazioni con task di elaborazione dati che cambiano di frequente: nello specifico è un sistema *MVC* (Model-View-Controller). Quest'ultimo è un pattern architetturale molto diffuso nello sviluppo di interfacce grafiche di sistemi software object-oriented, in grado di separare logica di presentazione dei dati dalla logica di business. È un'architettura multi-tier: le varie funzionalità del sistema sono logicamente separate e suddivise su più strati o livelli software differenti in comunicazione tra loro.

3.2 Decomposizione in sottosistemi

3.2.1 Decomposizione in Layer

Come specificato nella precedente overview, il sistema verrà decomposto in tre layer che si occupano di gestirne aspetti e funzionalità differenti:

- **Model**: mantiene la conoscenza del dominio di applicazione fornendo, al contempo, le operazioni per accedere ai dati utili all'applicazione;
- **View**: permette all'utente di visualizzare gli oggetti del dominio dell'applicazione (cioè, i dati contenuti nel model);
- **Controller**: è lo strato responsabile della sequenza di interazioni con l'utente (riceve i comandi dell'utente - in genere attraverso il View – e li attua modificando lo stato degli altri due componenti).



3.2.2 Decomposizione in sottosistemi

Il sistema si compone di 17 sottosistemi collocati nei diversi layer Model, View e Controller.

Il **Model** prevede la gestione di un sottosistema:

- "Archive", responsabile dell'interfacciamento diretto con la base di dati.

- "Admin Management", per la realizzazione delle operazioni permesse ad un un admin nel sistema;
- "Patient Management", per la realizzazione delle operazioni permesse ad un un paziente nel sistema;
- "Health Worker Management", per la realizzazione delle operazioni permesse ad un un operatore sanitario nel sistema;
- "Validation Management", per eseguire la validazione delle richieste in input al sistema;
- "Error Management", per la gestione degli errori di qualunque genere durante elaborazioni e interazioni nel sistema.
- "Statistics Management", per realizzare il calcolo delle statistiche.
- "Swab Management", per la gestione dei tamponi, notifica e scheduling (IA component).

La **View** prevede la gestione di cinque sottosistemi:

- "GUI Patient", l'interfaccia grafica per il paziente;
- "GUI Health Worker", l'interfaccia grafica per l'operatore sanitario;
- "GUI Admin", l'interfaccia grafica per l'admin;
- "GUI Statistics", l'interfaccia che presenta le statistiche periodiche.
- "GUI Error", un'interfaccia grafica comune agli attori per la visualizzazione di errori.

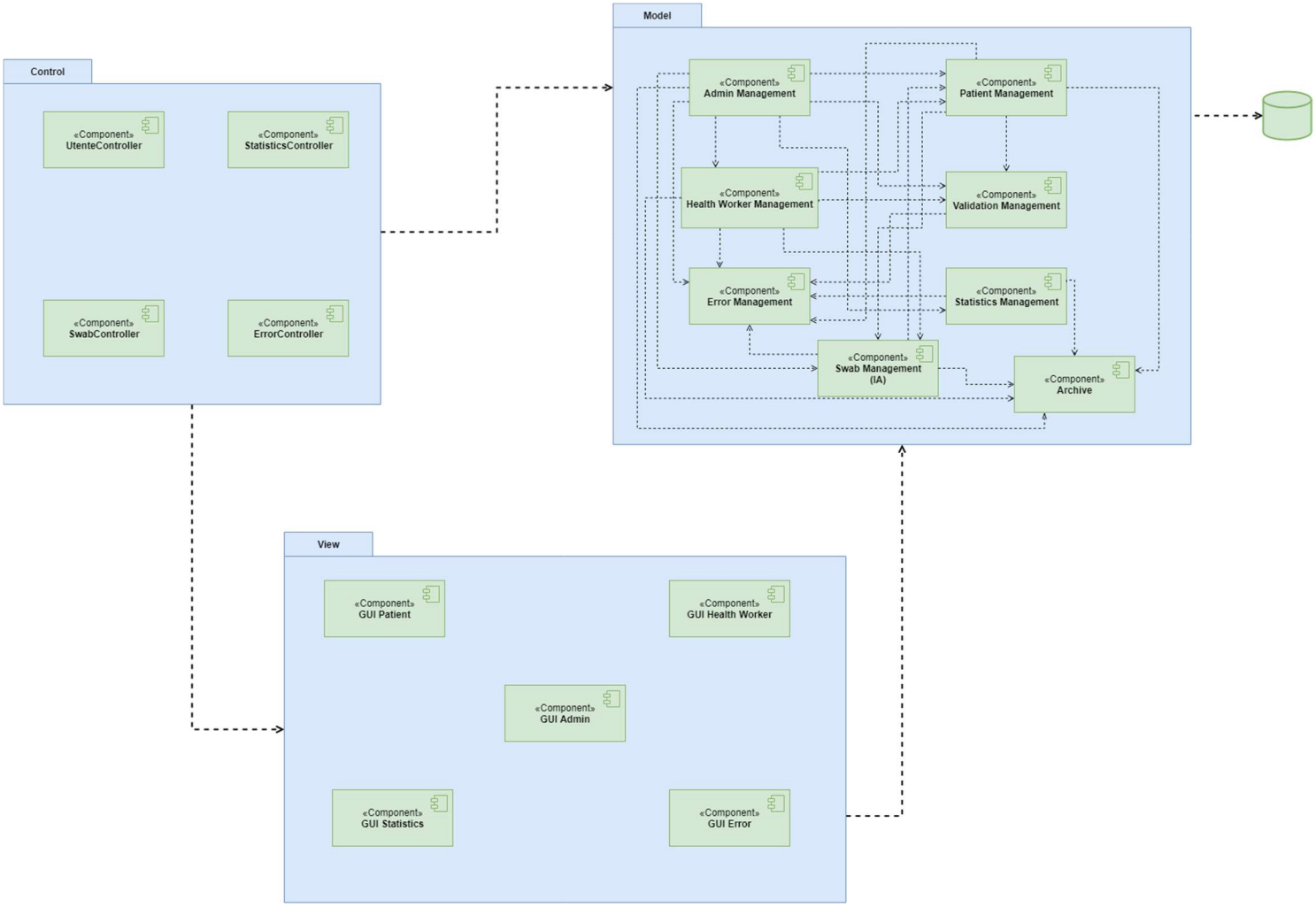
Il **Control** prevede la gestione di sette sottosistemi:

- "UtenteController", per la gestione delle interazioni tra utenti e il sistema;
- "ErrorController", per la gestione degli errori durante l'interazione con il sistema.
- "StatisticsController", per la gestione del calcolo delle statistiche.
- "SwabController", per la gestione dei tamponi.

Il Component Diagram è così suddiviso:

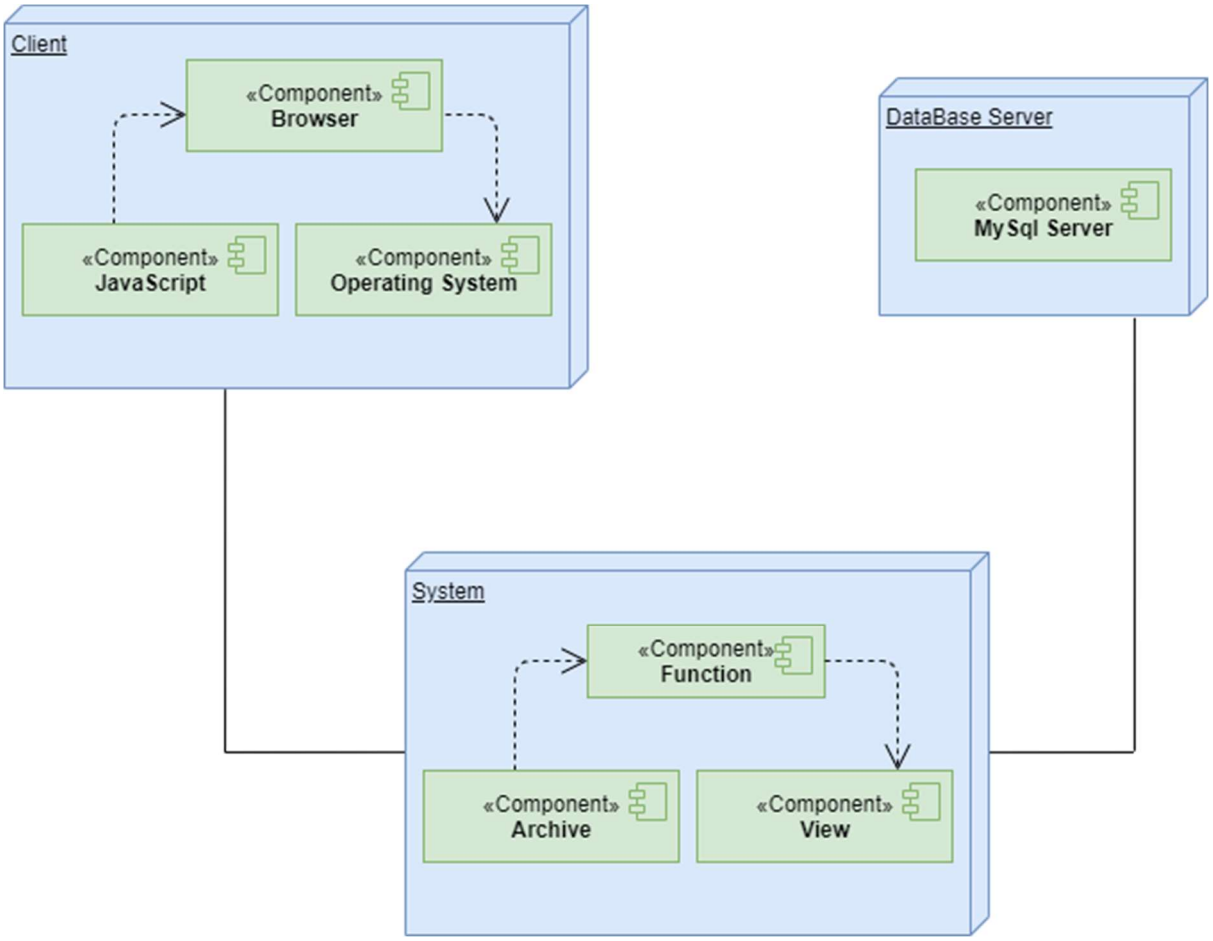
- Model, View e Control in azzurro;

- Componenti e DB in Verde;



3.2.3 Deployment Diagram

L'utente (LAN Client) richiede le funzionalità tramite l'interfaccia messa a disposizione dal sistema. L'unico requisito essenziale per il corretto funzionamento delle operazioni è l'utilizzo di un browser capace di interpretare codice JavaScript. Il tier del LAN Client connette lo strato di View del System sul quale vengono eseguite le funzioni apposite al completamento degli obiettivi del LAN Client. La parte Server racchiude e gestisce la persistenza dei dati. L'intera architettura non richiede l'ausilio di componenti hardware/software esterni.



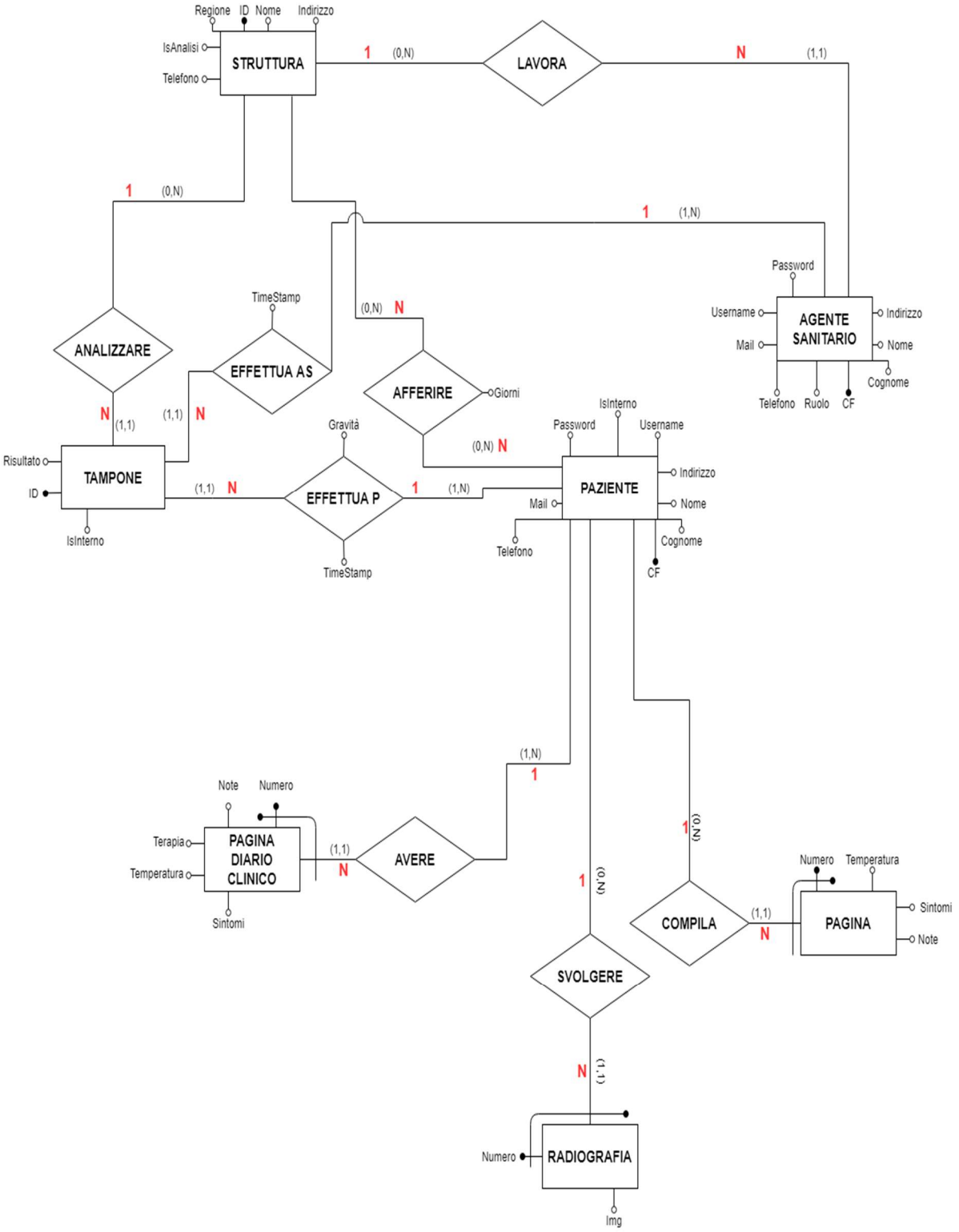
3.3 Mapping hardware/software

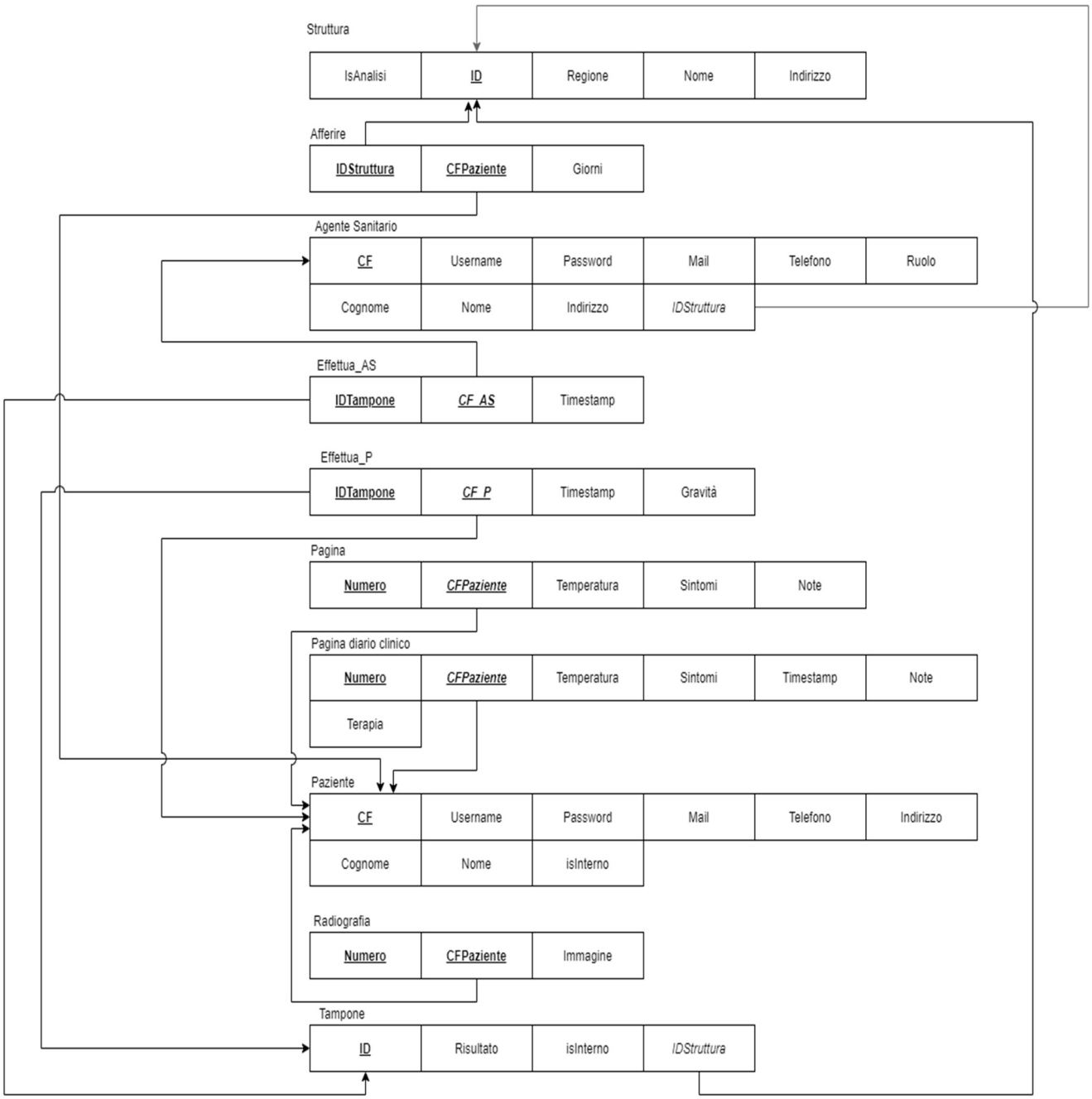
Di seguito il mapping hardware/software utilizzato per "Hilo": "Il sistema che si desidera sviluppare utilizzerà una struttura hardware costituita da un Server che risponderà ai servizi richiesti dai client. Il client è una qualsiasi macchina attraverso il quale un utente può collegarsi, utilizzando una connessione internet, per accedere al sistema mentre la macchina server gestisce la logica e i dati persistenti contenuti nel database. Il client e il server saranno connessi tramite il protocollo HTTPS, con il quale il client inoltra delle richieste al server e quest'ultimo provvederà a fornire i servizi richiesti. Le componenti hardware e software necessarie per il client sono un computer dotato di connessione internet e di un web browser installato su di esso. Per il server, invece, c'è necessità di una macchina con connessione ad Internet e con la capacità di immagazzinare una notevole quantità di dati. La componente software necessaria è dunque un DBMS, per consentire la comunicazione con più client."

3.4 Gestione dati persistenti

Per la memorizzazione dei dati persistenti si è deciso di utilizzare un RDBMS (Relational Database Management System) dal momento che fornisce un accesso ai dati veloce e permette di collegare entità differenti in modo semplice. Fornisce un accesso concorrente ai dati mantenendo la coerenza dei dati anche in condizione di multiutenza e soprattutto, incorpora un meccanismo di permessi, che rende l'accesso a dati sensibili protetto e quindi utenti con operazioni diverse possono accedere a

sezioni diverse della base di dati. Di seguito si allega lo schema concettuale e il mapping logico del database che si intende utilizzare per la realizzazione del sistema.





3.5 Controllo degli accessi e sicurezza

Il sistema prevede un controllo degli accessi basato su autenticazione tramite credenziali per l'interazione con esso. Di seguito viene costruita una matrice degli accessi che rappresenta le operazioni concesse per ogni utente (per cui ovviamente è prevista l'autenticazione).

Oggetto / Attore	Patient Management	Admin Management	Health Worker Management	Statistics Management	Swab Management
Ospite	-	-	-	Visualizzare statistiche	-
Paziente	Vedere i propri dati Inserire o modificare i propri dati Vedere i propri tamponi	-	-	Visualizzare statistiche	-
Operatore Sanitario	Vedere i dati paziente Inserire o modificare i dati paziente	-	Vedere i propri dati Inserire o modificare i propri dati	Visualizzare statistiche	Inserire e rimuovere tamponi Modifica tampone
Admin	-	CRUD	-	Visualizzare statistiche	-

3.6 Controllo flusso globale del sistema

Il sistema in realizzazione richiede per sua natura un livello sostanzialmente molto elevato di interazione con l'utente (sia esso un paziente, un operatore sanitario o l'amministratore della piattaforma). Non prevediamo inoltre un numero elevato di operazioni che non richiedono supervisione. Si può affermare che il sistema possa essere definito come "event-driven" in modo tale da semplificare l'approccio con l'utilizzatore. Ad esempio, saranno inviate e-mail ai pazienti se e solo se sono disponibili aggiornamenti su pratiche che gli riguardano (nella stragrande maggioranza dei casi, risultati di tamponi) e mai in altre situazioni. Si prevede, inoltre, data la particolarità del dominio del problema, di dover gestire in maniera concorrente interazioni del sistema con più utenti.

3.7 Condizione limite

Avvio del sistema. Al primo avvio, il sistema necessita di un web server che fornisca il servizio di accesso ad un database MySQL per la gestione dei dati persistenti. Quando un utente accede al sistema, gli verrà presentata una pagina di login, dove egli potrà effettuare l'accesso mediante le credenziali in suo possesso e sarà reindirizzato su una pagina di benvenuto che gli mostrerà le operazioni che gli è consentito svolgere a seconda del suo ruolo all'interno del sistema.

Terminazione. Alla chiusura dell'applicazione, il sistema termina con un logout automatico derivato dalla terminazione della sessione utente. Il server dovrà essere terminato manualmente dall'amministratore del sistema, dopodiché nessun client potrà connettersi alla piattaforma.

Fallimento. Si possono individuare diverse situazioni di fallimento:

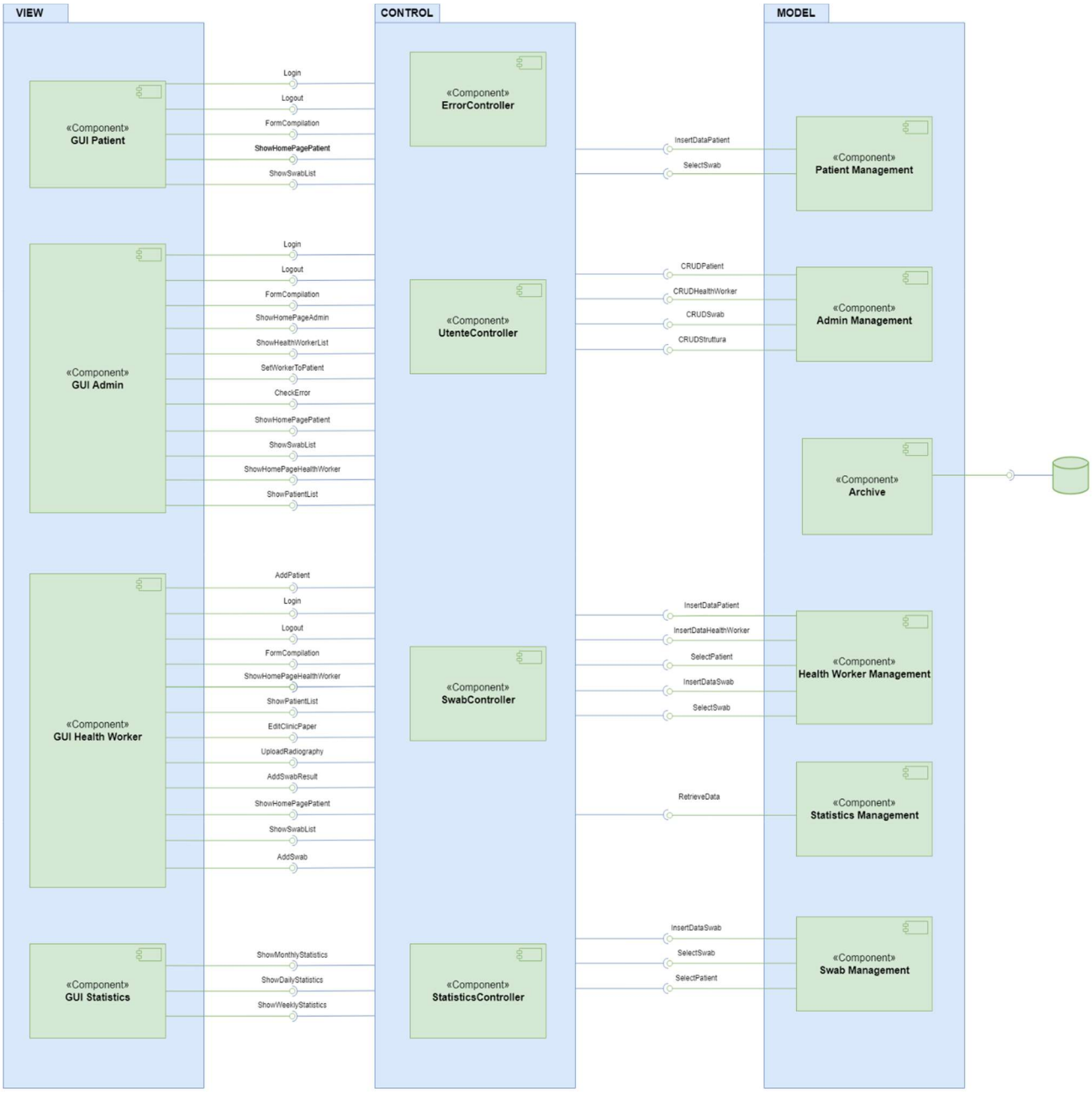
Nel caso di guasti dovuti al sovraccarico del database con successivo fallimento dello stesso è prevista come procedura preventiva il salvataggio periodico dei dati sotto forma di codice SQL per la successiva ricostruzione del DB.

Nel caso in cui si verifichi un'interruzione inaspettata dell'alimentazione del server, sarà previsto un generatore di emergenza che si attiverà nell'istante in cui l'alimentazione verrà a mancare e se per i successivi cinque minuti l'alimentazione non sarà ripristinata allora si procederà con un salvataggio del DB in formato SQL con conseguente arresto di emergenza del server per evitare danni ai dati e all'hardware.

Un altro caso di fallimento potrebbe derivare dal software stesso che causa una chiusura inaspettata dovuta ad errori commessi durante la fase di implementazione. In questo caso non si prevedono politiche di recupero specifiche, se non il riavvio dell'intero sistema nel caso di errori fatali che ne compromettano il normale utilizzo.

Un'ulteriore situazione di errore potrebbe essere il fallimento di una componente hardware come ad es. un dispositivo di archiviazione di massa. In questo caso non si prevedono contromisure software.

4. Servizi dei Sottosistemi



Le **View** rappresentano le interfacce che gestiscono l'interfaccia grafica e gli eventi generati dall'interazione dell'utente con il sistema.

GUI Patient offre cinque servizi all'interfaccia Control:

- Login;
- Logout;
- FormCompilation;
- ShowHomePagePatient;
- ShowSwabList;

GUI Admin offre undici servizi all'interfaccia Control:

- Login;
- Logout;
- FormCompilation;
- ShowHomePageAdmin;
- ShowHomePagePatient;
- ShowHomePageHealthWorker;
- ShowPatientList;
- ShowHealthWorkerList;
- SetWorkerToPatient;
- ShowSwabList;
- CheckError;

GUI Health Worker offre dodici servizi all'interfaccia di Control:

- Login;
- Logout;
- FormCompilation;
- ShowHomePageHealthWorker;
- ShowPatientList;
- ShowHomePagePatient;
- EditClinicPaper;
- UploadRadiography;
- ShowSwabList;
- AddSwabResult;
- AddSwab;
- AddPatient;

GUI Statistics offre tre servizi all'interfaccia di Control:

- ShowMonthlyStatistics;
- ShowDailyStatistics;
- ShowWeeklyStatistics;

Patient Management offre due servizi all'interfaccia di Control:

- InsertDataPatient;
- SelectSwab;

Admin Management offre quattro servizi all'interfaccia di Control:

- CRUDPatient;
- CRUDHealthWorker;
- CRUDSwab;
- CRUDStruttura;

Health Worker Management offre cinque servizi all'interfaccia di Control:

- InsertDataHealthWorker;
- InsertDataPatient;
- SelectPatient;
- InsertDataSwab;
- SelectSwab;

Statistics Management offre un servizio all'interfaccia di Control:

- RetrieveData;

Swab Management offre tre servizi all'interfaccia di Control:

- InsertDataSwab;
- SelectSwab;
- SelectPatient;