SDD: SYSTEM DESIGN DOCUMENT
Health Care

Riferimento	2022_RAD_C03_ver1.2 2022_MT_C03
Versione	1.4
Data	07/02/2023
Destinatario	Prof.ssa F. Ferrucci, Prof. F. Palomba
Presentato da	Giovanni Borrelli, Gerardo Di Muro, Agostino Andrea Mangia, Giovanni Mercurio, Luca Pastore, Angelo Zuottolo
Approvato da	Giusy Castaldo, Mariarosaria Esposito, Francesca Perillo

REVISION HISTORY

Data	Versione	Descrizione	Autori
25/11/2022	0.1	Design Goals	Tutti
25/11/2022	0.2	Trade Off	Tutti
25/11/2022	0.3	Architettura Sistemi Simili	Gerardo Di Muro, Luca Pastore
25/11/2022	0.4	Obiettivo del Sistema SDD	Agostino Andrea Mangia, Giovanni Borrelli
29/11/2022	0.5	Panoramica Sistema Proposto SDD	Luca Pastore
29/11/2022	0.6	Mapping hardware/software SDD con Modulo FIA	Luca Pastore
02/12/2022	0.7	Architettura del sistema e sottosistemi individuati SDD	Tutti
02/12/2022	0.8	Deployment Diagram SDD	Tutti
02/12/2022	0.9	Gestione Dati Persistenti	Agostino Andrea Mangia, Angelo Zuottolo, Gerardo Di Muro, Giovanni Borrelli, Giovanni Mercurio
02/12/2022	1.0	Controllo degli Accessi e Sicurezza	Agostino Andrea Mangia, Gerardo Di Muro, Giovanni Mercurio

02/12/2022	1.1	Controllo del Flusso Globale del Sistema	Agostino Andrea Mangia, Gerardo Di Muro, Giovanni Mercurio
02/12/2022	1.2	Servizi dei Sottosistemi	Tutti
02/12/2022	1.3	Condizioni Limite	Angelo Zuottolo, Giovanni Borrelli, Luca Pastore
07/02/2023	1.4	Revisione Documentazione	Tutti

Sommario

1. Introduzione

1.1 Obiettivo del sistema

Lo scopo principale del sistema è quello dell'identificazione di eventuali Malattie Rare di cui sono affetti i pazienti dei MMMG.

L'obiettivo che ci poniamo, in quanto team, è quello di creare un sistema che sia di aiuto ai MMMG nel processo di individuazione e diagnosi di Malattie Rare nei pazienti, che permetta loro di creare e commentare dei Form di discussione con cui potersi confrontare con altri MMMG che hanno affrontato situazioni simili e che, inoltre, permetta la consultazione dell'elenco di tutte le Malattie Rare disponibili attraverso la ricerca per nome o per sintomi delle malattie stesse.

Il sistema sarà un'applicazione web che permetterà l'accesso e la registrazione unicamente ai MMMG.

L'analisi del sistema ha messo in evidenza 3 categorie in cui sarà diviso il sistema:

- Gestione dell'utenza;
- Gestione dei sintomi e delle malattie;
- Gestione dei Form.

1.2 Design Goals

Priorità	ID Design Goals	Descrizione Design Goals	Categoria	RNF di Origine
Alta	DG_1 Usabilità	Il sistema deve garantire che tutte le funzionalità messe a disposizione, come ricerca di una malattia ed apertura di un Form avvengano in modo semplice ed efficiente.	End user	RNF_1 RNF_2
Bassa	DG_2 Usabilità	Il sistema deve disporre di un manuale utente disponibile in formato PDF, per permettere all'utente di poter utilizzare l'applicativo.	End User	RNF_3

Medio/Bassa	DG_3 Robustezza	Il sistema deve sapersi comportare in situazioni di input non valido notificando all'utente l'errore sotto forma di messaggio.	Dependability	RNF_4
Medio/Bassa	DG_4 Sicurezza	Il sistema deve garantire la sicurezza dei dati di ogni utente registrato, imponendo delle credenziali d'accesso che impediranno a utenti non autorizzati di accedere a profili altrui.	Dependability	RNF_5
Alta	DG_5 Sicurezza	Il sistema deve crittografare i dati di accesso secondo una funzione hash di crittografia (MD5) standardizzata con la RFC 1321	Dependability	RNF_6
Alta	DG_6 Robustezza	Il sistema deve effettuare una pre-validazione dei dati, tramite apposite funzioni Java, per stabilirne la correttezza.	Dependability	RNF_7
Media	DG_7 Tempi di risposta	Il sistema deve garantire un tempo di risposta non superiore a cinque secondi.	Performance	RNF_8
Alta	DG_8 Leggibilità	Il sistema deve sfruttare la portabilità del linguaggio Java per rendere più semplice e leggibile lo sviluppo del nostro software.	Maintenance	RNF_9

Bassa	DG_ 9 Portabilità	Il software deve essere responsive in modo da poter essere accessibile in ambienti diversi.	Maintenance	RNF_12
Alta	DG_10 Usabilità	Il sistema deve risultare facilmente comprensibile grazie all'uso delle 8 regole d'oro di Shneiderman per il design delle interfacce grafiche.	End user	RNF_13
Alta	DG_11 Declinazione dei costi	Il sistema deve essere attivato e scaricato dal Medico di Medicina Generale tramite il link e il relativo codice di attivazione che verrà recapitato tramite e-mail.	Costo	RNF_14
Alta	DG_12 Portabilità	L'utilizzo di HTML5, CSS3 e JavaScript rende accessibile il sistema su vari tipi di browser.	Maintenance	RNF_10

1.2.1. Design Trade-off

Memoria usata - Tempo di risposta

Uno dei requisiti del sistema è quello di fornire all'utilizzatore tempi di risposta brevi, per raggiungere questo obiettivo potrebbe risultare necessaria l'allocazione di una quantità maggiore di memoria centrale privilegiando così il tempo di risposta rispetto alla memoria utilizzata. Il team di analisi ha deciso quindi di dare priorità maggiore al tempo di risposta nel trade off tra Memoria usata e Tempo di risposta.

Tempo di Consegna – Qualità

Considerata la necessità del cliente di avere un prodotto finito nei tempi di consegna previsti, è stato dato un focus maggiore al Tempo di Consegna piuttosto che alla qualità delle funzionalità la cui priorità rientra in un range medio/basso.

Costo – Robustezza

Considerando che il nostro sistema tratta dati fortemente sensibili e tenendo conto che l'obiettivo generale del sistema è quello di velocizzare la diagnosi da parte del MMG, il nostro team ha deciso di dare una priorità maggiore alla Robustezza del sistema vista la necessità di un meccanismo che si comporti in modo ragionevole in situazioni impreviste. Il team di analisi ha deciso di privilegiare la Robustezza nel trade-off tra costo e robustezza.

1.3 Organizzazione del Documento

Il documento è suddiviso in sezioni organizzate nel modo seguente:

Sezione 1: Presenta l'introduzione al sistema con la definizione dei design goals, dei trade off scelti in fase di analisi e diverse informazioni utili per la comprensione di questo documento.

Sezione 2: Descrive le funzionalità offerte dal sistema corrente.

Sezione 3: Descrive l'architettura del sistema proposto, specialmente la decomposizione in sottosistemi, il controllo degli accessi e sicurezza, i dati persistenti, le condizioni limite, il mapping hardware/software, il controllo del flusso globale del sistema.

Sezione 4: Descrive i servizi dei sottosistemi tramite lo schema UML.

Sezione 5: Elenco dei riferimenti utilizzati.

Sezione 6: Glossario del documento.

2. Architettura del Sistema corrente

Secondo alcune ricerche, attualmente esiste un software che funge da supporto ai MMMG per diagnosticare Malattie Rare come fa il nostro sistema, tuttavia quest'ultimo non prevede l'utilizzo di un ambiente di confronto tra MMMG, cosa che in Health Care avviene grazie all'utilizzo dei Form.

Health Care, oltre ad offrire la possibilità di ricercare una malattia rara attraverso l'inserimento dei sintomi di un paziente, offre la possibilità di poter aprire un luogo di confronto tra i vari Medici di Medicina Generale, al fine di consentire uno scambio di idee sul come poter trattare una determinata patologia rara.

Il sistema che si avvicina di più a Health Care è il seguente:

- Orphanet, un sistema che permette ad un utente generico di poter ricercare, tramite il nome, una malattia rara.

3. Architettura del Sistema Proposto

Panoramica

Il sistema da noi proposto è una Web App che funge da supporto per i Medici di Medicina Generale al fine di diagnosticare Malattie Rare in modo semplice e veloce. Dopo un'attenta analisi del nostro software, abbiamo ritenuto opportuno utilizzare un'architettura di tipo Three-Tier, in cui l'interfaccia utente, i processi logico funzionali, l'archiviazione informatica dei dati e l'accesso ai dati sono sviluppate e mantenute come moduli indipendenti, la maggior parte delle volte su piattaforme separate. L'architettura Three-Tier è un'architettura software ben consolidata che organizza applicazioni in tre tier di calcolo logici e fisici:

- il Tier di presentazione (**Presentation**), o interfaccia utente;
- il Tier dell'applicazione (**Application**), dove vengono elaborati i dati;
- il Tier dei dati (**Storage**), dove vengono archiviati e gestiti i dati associati all'applicazione.

Il vantaggio principale dell'architettura three-tier è che, poiché ciascun tier viene eseguito sulla propria infrastruttura, può essere sviluppato contemporaneamente da un team di sviluppo separato e può essere aggiornato o scalato in base alle necessità senza effetti sugli altri tier. Il tier di presentazione è l'interfaccia utente ed il livello di comunicazione dell'applicazione, tramite il quale cui l'utente finale interagisce con l'applicazione. Lo scopo del layer di presentazione è quello di visualizzare e raccogliere informazioni dall'utente.

Il tier di applicazione, noto anche come tier logico o tier intermedio, è il cuore dell'applicazione. In questo tier, le informazioni raccolte nel tier di presentazione vengono elaborate - a volte rispetto ad altre informazioni nel tier dei dati - utilizzando la logica aziendale, una serie specifica di regole aziendali. Il tier dell'applicazione può anche aggiungere, eliminare o modificare i dati nel tier dei dati.

Il tier dei dati, in alcuni casi chiamato tier del database, tier di accesso ai dati o back-end, è il punto in cui vengono archiviate e gestite le informazioni elaborate dall'applicazione. Nel nostro sistema si prevede l'utilizzo un tipo di database non relazionale (MongoDB).

Abbiamo deciso di utilizzare un'architettura three-tier, inoltre, per i seguenti motivi:

- **Sviluppo più veloce:** poiché ciascun tier può essere sviluppato contemporaneamente da team differenti, un'organizzazione può immettere l'applicazione sul mercato più velocemente e i programmatori possono utilizzare i linguaggi e gli strumenti più recenti e migliori per ogni tier.

- **Scalabilità migliorata:** qualsiasi tier può essere scalato indipendentemente dagli altri in base alle necessità.
- **Massima affidabilità:** un'interruzione in un tier avrà meno probabilità di avere impatto sulla disponibilità o sulle prestazioni degli altri tier.
- **Sicurezza migliorata:** poiché il tier di presentazione e il tier dei dati non riescono a comunicare direttamente, un tier dell'applicazione ben progettato può funzionare come una sorta di firewall interno, impedendo SQL Injection e altri exploit dannosi.

3.1. Decomposizione in sottosistemi

La decomposizione del nostro sistema è basata secondo l'architettura Three Tier, composta da tre layer che definiscono aspetti e funzionalità differenti. Il motivo per cui abbiamo scelto di utilizzare un'architettura Three Tier è per la possibilità di separare la logica di Presentazione, di Business e di Storage.

Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il sottosistema di **Storage** che mantiene la conoscenza del dominio di applicazione e quindi fornisce i metodi per accedere ai dati utili all'applicazione;
- il sottosistema di **Presentation** che visualizza all'utente gli oggetti del dominio di applicazione;
- il sottosistema di **Application** che è responsabile della sequenza di interazioni con l'utente, riceve comandi dall'utente attraverso il tier di Presentation e li attua modificando lo stato degli altri componenti.

Da un'analisi dei vari oggetti (Storage, Presentation e Application) e vista l'esigenza di utilizzare basso accoppiamento ed elevata coesione tra le diverse componenti del sistema, si definiscono i seguenti sottosistemi, divisi per categorie:

Presentation:

- InterfacciaUtente (sottosistema che permette di interfacciarsi con l'utente)

Application:

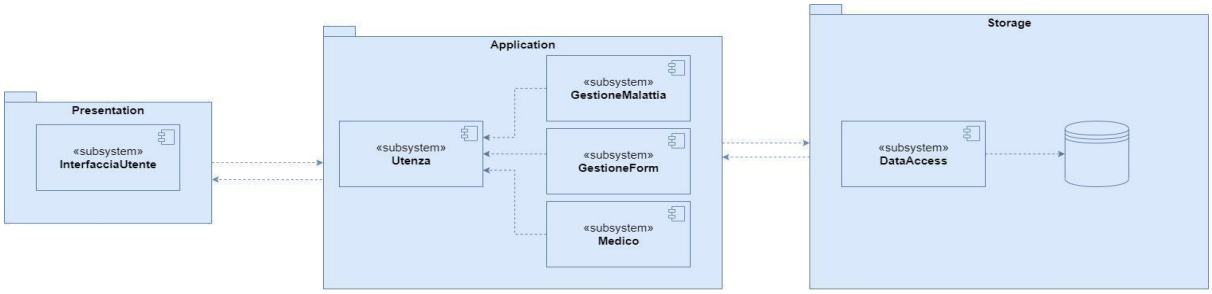
- Utenza (sottosistema per la gestione del login e del logout degli utenti registrati e delle richieste di registrazione);
- GestioneMalattia (sottosistema per la gestione di Malattie Rare)
- Medico (sottosistema per la gestione dei dati del Medico di Medicina Generale)

- GestioneForm (sottosistema per la gestione dei Form)

Storage:

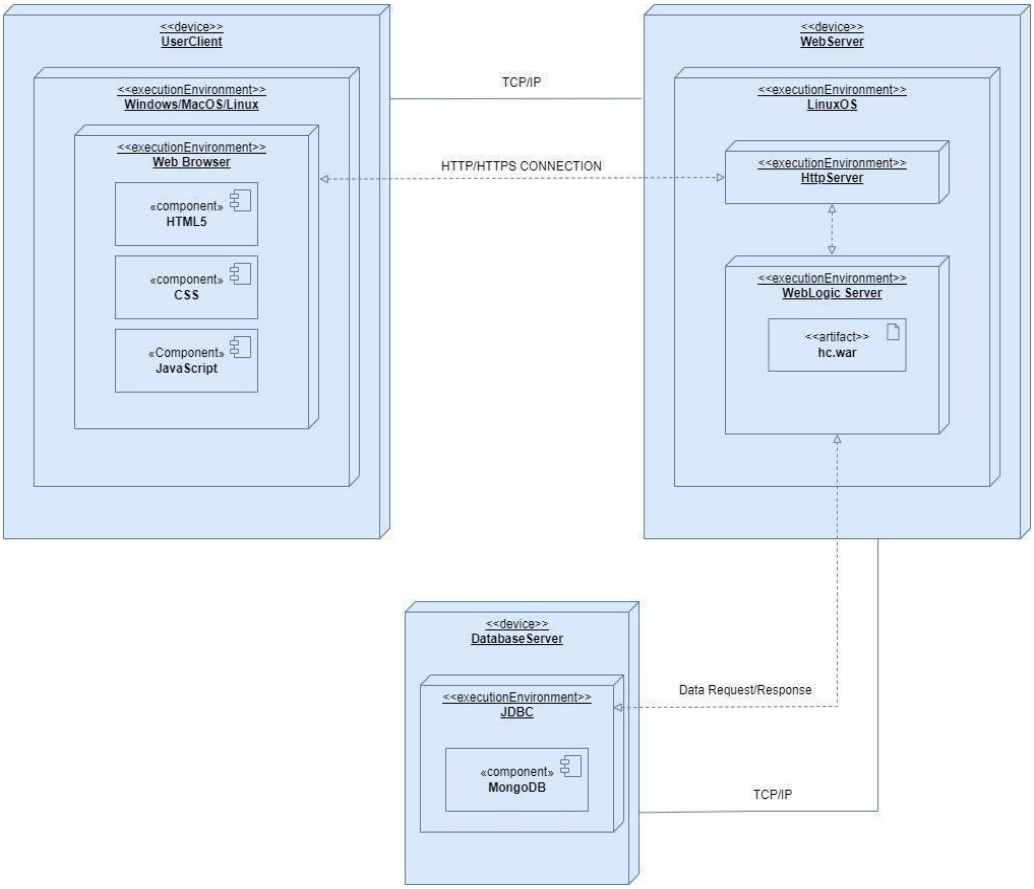
- DataAccess (sottosistema che si interfaccia con il database permettendo l'accesso e la memorizzazione di dati persistenti)

Dunque, il sistema si compone di sei sottosistemi, come mostrato in figura:



3.1.1. Deployment Diagram

Lo User Client, tramite interfaccia, richiede una serie di funzionalità che possono essere messe a disposizione solo se si ha un Web Browser capace di interpretare codice HTML, CSS e JavaScript. Lo User Client si connette al Web Server per servirsi delle diverse funzionalità offerte dal nostro software. Dall'altra parte, vi è una componente, la Web Logic, che soddisferà le richieste dello User Client, fornendo anche la connessione al database.



3.2. Mapping hardware/software

Il sistema sarà basato su una struttura hardware costituita da un client e un Server. Il client è qualsiasi macchina che abbia un accesso ad Internet e un web server installato in modo da potersi interfacciare con il server per poter comunicare. Il server è una qualsiasi macchina che abbia una connessione ad Internet e che possa immagazzinare una grande quantità di dati. Esso è responsabile di gestire i dati persistenti all'interno del database. Lo scambio tra client e server avviene tramite il protocollo HTTP all'interno del quale il client invierà delle richieste al server e il server le soddisferà. La componente di cui necessita il nostro sistema è un DBMS che permette la comunicazione tra più client.

3.3. Gestione dati persistenti

Il sistema che abbiamo intenzione di sviluppare deve gestire la memorizzazione di dati persistenti tramite l'utilizzo di un database non relazionale, tra i vantaggi si notano sicuramente le performance più alte per i tempi di risposta; infatti, nei database non relazionali un elemento contiene tutte le informazioni necessarie e dunque non serve usare i dispendiosi "join" come invece avviene per i database relazionali. Inoltre, la semplicità di questi database è uno degli elementi fondamentali, è proprio questo che permette di scalare in orizzontale in maniera così efficiente, molti DB non relazionali, infatti, permettono di aggiungere nodi a caldo in maniera impercettibile dall'utente finale. Infine, scegliendo un database adatto

alla mappatura più diretta alle object classes del proprio applicativo si possono ridurre di molto i tempi dedicati allo sviluppo del metodo di scambio dati tra il database e l'applicativo stesso (il cosiddetto object-relational mapping che è invece necessario in presenza di database relazionali). Grazie all'utilizzo del database i dati sono protetti e quindi diversi utenti possono accedere a diverse sezioni del database in modo semplice e sicuro.

Di norma per la normale visione di uno schema non relazionale è preferibile utilizzare un formato di visualizzazione come JSON, per semplicità di seguito verrà riportato lo schema non relazionale in un formato tabulare risultando così più leggibile.

_id	nome_ medico	cogno me	email	indirizzo	sesso	eta	provi ncia	comu ne	telefon o	passwo rd

_id	topic	titolo	descrizione	status	Data Apertura	Data Chiusura	autore

_id	codice	nome_malattia	descrizione	sintomo(n)

_id	email	descrizione	id_form

_id	codice	nome_sintomo

3.4. Controllo degli accessi di sicurezza

All'interno del nostro sistema vi è un unico attore che si serve delle diverse funzionalità che il sistema mette a disposizione. Attraverso il meccanismo dell'Access Control List, abbiamo stilato una lista di coppie [Attore, Operazione], in modo che sia più chiaro quali sono le funzionalità di cui il Medico di Medicina Generale si può servire.

Lista di controllo degli accessi per il sottosistema Utenza:

Attore	Operazione
MMG	Login ()
MMG	Logout ()

Lista di controllo degli accessi per il sottosistema GestioneMalattie:

Attore	Operazione
MMG	RicercaPerNome()
MMG	RicercaPerSintomi()
MMG	VisualizzazioneElencoMalattieRare()
MMG	VisualizzazioneDettagliMalattieRare()

Lista di controllo degli accessi per il sottosistema Medico:

Attore	Operazione
MMG	VisualizzazioneDatiProfilo()
MMG	ModificaDatiProfilo()

Lista di controllo degli accessi per il sottosistema GestioneForm:

Attore	Operazione
MMG	CreazioneForm()
MMG	ModificaForm()
MMG	CancellazioneForm()
MMG	VisualizzazioneFormAperti()
MMG	VisualizzazioneFormChiusi()
MMG	ChiusuraForm()
MMG	RiattivazioneForm()
MMG	InserimentoIntervento()
MMG	CancellazioneIntervento()
MMG	ModificaIntervento()
MMG	VisualizzazioneNotificheIntervento()
MMG	VisualizzazioneInterventiEffettuato()

Occorre notare che alcune delle funzionalità non saranno implementate data la priorità Medio-Bassa specificata in precedenza all'interno del RAD.

3.5. Controllo flusso globale del sistema

Il nostro sistema richiede una continua interazione da parte dell'utente, per questo motivo utilizziamo un tipo di controllo del flusso del sistema di tipo event-driven, ovvero guidato dagli eventi.

3.6. Condizioni Limite

Per la prima start-up del nostro sistema è indispensabile che tutto venga gestito da un web server, dotato del servizio di una base di dati che gestisca l'amministrazione dei dati persistenti. Una volta che il web server viene avviato, sarà possibile accedere alla nostra Web-App attraverso l'utilizzo di credenziali, previa

opportuna registrazione. Una volta effettuato l'accesso, il MMG avrà a disposizione una serie di funzionalità di cui potrà servirsi.

Avvio del sistema

Identificativo UC_SU	Avvio del server	Data:	30/11/2022
		Versione:	0.00.001
		Autore:	Luca Pastore, Giovanni Borrelli, Angelo Zuottolo
Descrizione	Lo UC fornirà la funzionalità di avvio del server da parte dell'amministratore del sistema		
Attore principale	Amministratore È interessato ad avviare il server		
Attori secondari	NA		
Entry condition	L'amministratore è interessato ad avviare il server AND Il sistema deve fornire un meccanismo che glielo permetta		
Exit condition On success	L'amministratore avvia il server correttamente AND Il sistema restituirà la pagina di accesso per i futuri utenti		
Exit condition On failure	Il server non viene avviato AND Il sistema mostra all'amministratore un messaggio d'errore		
Rilevanza/	Alta		

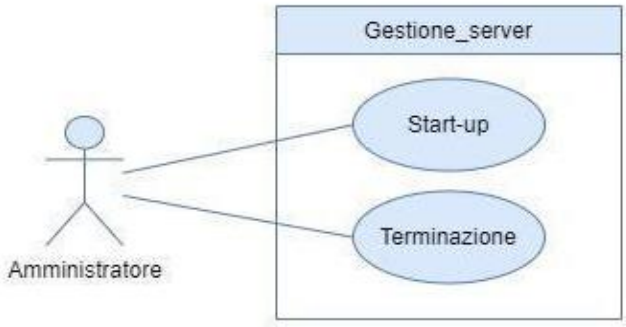
User priority		
Frequenza stimata	6 volte/anno	
Extension Point	NA	
Generalization of	NA	
FLUSSO DI EVENTI PRINCIPALE / MAIN SCENARIO		
1	Amministratore	Stabilisce la connessione col database
2	Amministratore	Avvia uno script di configurazione del database
3	Sistema	Inizializza il database
4	Amministratore	Esegue il deploy del file "hc.war" del sistema sul server
5	Sistema	Se il server è stato spento normalmente, legge la lista dei MMMG. Se il server si è spento per un errore, viene notificato all'amministratore che c'è stato un errore improvviso
I Scenario/Flusso di eventi di ERRORE: Avvio non riuscito		
5.1	Sistema:	Mostra a schermo un messaggio di errore

Terminazione

La terminazione del sistema avviene attraverso la funzionalità di logout. Per consentire che il server termini nel modo corretto, viene disposto un meccanismo secondo il quale, alla chiusura del server, nessun altro client ha possibilità di collegarsi ad esso.

Identificativo UC_SD	Terminazione del server	Data:	30/11/2022
		Versione:	0.00.001
		Autore:	Luca Pastore, Giovanni Borrelli, Angelo Zuottolo
Descrizione	Lo UC fornirà la funzionalità di terminazione del server da parte dell'amministratore del sistema		
Attore principale	Amministratore È interessato a terminare il server		
Attori secondari	NA		
Entry condition	L'amministratore è interessato a terminare il server AND Il sistema deve fornire un meccanismo che glielo permetta		
Exit condition On success	L'amministratore termina il server correttamente AND Il sistema farà in modo da non permettere ad altri utenti di potersi collegare ad esso		
Exit condition On failure	Il server non termina AND Il sistema mostra all'amministratore un messaggio d'errore		
Rilevanza/ User priority	Alta		
Frequenza stimata	6 volte/anno		

Extension Point	NA	
Generalization of	NA	
FLUSSO DI EVENTI PRINCIPALE / MAIN SCENARIO		
2	Amministratore	Avvia uno script per terminare la connessione al database
3	Sistema	Esegue la terminazione del server
I Scenario/Flusso di eventi di ERRORE: Terminazione non riuscita		
5.1	Sistema:	Mostra a schermo un messaggio di errore



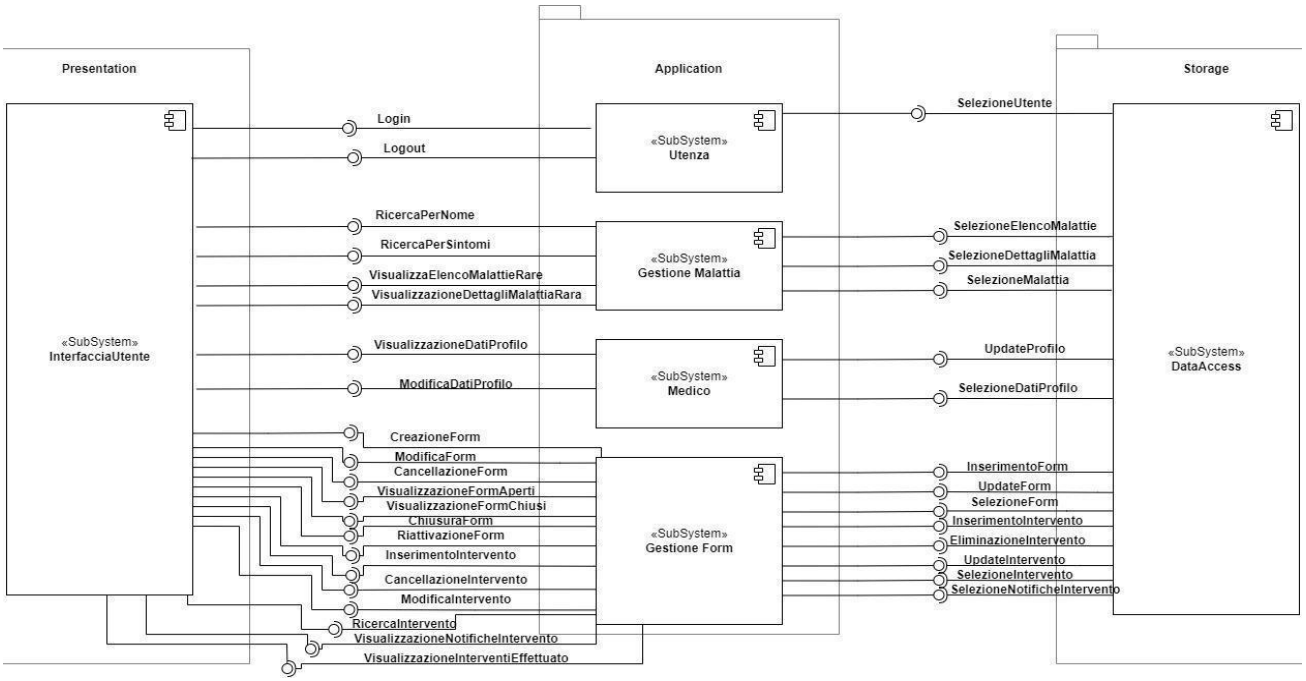
Fallimento

Le possibili cose che potrebbero portare al fallimento del nostro sistema sono:

- difetto critico di una componente per la quale non abbiamo soluzioni.
- Sovraccarico della base di dati che potrebbe portare a una conseguente perdita di dati. Siccome nel nostro caso i dati presenti nel database sono dati sensibili, il nostro sistema provvederà ad effettuare un salvataggio periodico di essi, in modo da ripristinare i dati che sono andati perduti.

- Un altro caso di fallimento potrebbe essere dovuto ad una terminazione inaspettata del server. Non prevediamo particolari contromisure a questo tipo di problema. L'unica operazione che ci precludiamo di fare è la riattivazione di tutto il sistema.

4. Servizi dei sottosistemi



5. Riferimenti

- 2022_RAD_C03
- Slide disponibili sulla piattaforma del corso di Ingegneria del Software 2022/2023
- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition (Bernd Bruegge & Allen H. Dutoit).

6. Glossario

A

Access Control List: metodologia di controllo degli accessi ai sottosistemi, per ogni sottosistema sono specificate coppie (attore, operazione), ognuna di queste coppie indica quali operazioni sono effettuabili da quali attori.

Application Tier: Layer del pattern architetturale Three-Tier responsabile della gestione della logica di business.

B

Boolean: tipi di dato necessarie per la memorizzazione dei dati persistenti nella base dati.

C

Char: tipo di dato necessario per la memorizzazione di dati persistenti nella base dati di tipo "Carattere".

D

Design Goals: obiettivi di design progettati per il sistema.

Design Trade-off: scelte e compromessi tra design goals dissonanti.

Deployment Diagram: diagramma di specifica per le relazioni tra le componenti realizzate e le risorse Hardware e Software necessarie al corretto funzionamento del sistema.

Date: tipo di dato necessario per la memorizzazione di dati persistenti nella base dati di tipo "Data".

E

Event-driven: controllo del flusso del sistema basato sull'uso di eventi.

I

Integer: tipo di dato necessario per la memorizzazione di dati persistenti nella base dati di tipo "Intero".

M

MMG: Medico di Medicina Generale.

MMMG: Medici di Medicina Generale.

MT: Matrice di tracciabilità.

P

Presentation Tier: Layer del pattern architetturale Three-Tier responsabile della visualizzazione delle informazioni all'utente.

R

RAD: Requirement Analysis Document.

S

Storage Tier: Layer del pattern architetturale Three-Tier responsabile della memorizzazione dei dati sul database e dell'accesso ad essi.

T

Three-Tier: Pattern architetturale che divide il sistema in tre Tier: Tier di presentazione, di applicazione e di storage.

U

UC: Use Case.

V

Varchar: tipo di dato necessario per la memorizzazione di dati persistenti nella base dati di tipo "Stringa a lunghezza variabile".