MedQueue

SDD

System Design Document

MedQueue

Afeltra Angelo
Amato Adriano
Fucile Andrea
Rapa Giovanni

Sommario

1. **Introduzione**
   - 1.1 Obiettivi del sistema
   - 1.2 Design Goals & Trade-offs
     - Tempo di rilascio vs Funzionalità
     - Prestazioni vs Costi
     - Prestazioni vs Affidabilità
   - 1.3 Definizioni, acronimi e abbreviazioni
   - 1.4 Riferimenti
   - 1.5 Panoramica
2. **Architettura di Sistemi simili**
3. **Architettura del Sistema proposto**
   - 3.1 Panoramica
   - 3.2 Decomposizione in sottosistemi
   - 3.3 Mapping hardware / software
   - 3.4 Gestione dati persistenti
   - 3.5 Controllo degli accessi e sicurezza
   - 3.6 Controllo flusso globale del sistema
   - 3.7 Condizione limite
     - 3.7.1 Start-up
     - 3.7.2 Terminazione
     - 3.7.3 Aggiunta Impiegato
     - 3.7.4 Eliminazione Impiegato
     - 3.7.5 Aggiunta Struttura
     - 3.7.6 Eliminazione Struttura
     - 3.7.7 Aggiunta Ambulatorio
     - 3.7.8 Eliminazione Ambulatorio
     - 3.7.9 Fallimento
4. **Servizi dei Sottosistemi**
5. **Glossario**

Data	Versione	Cambiamenti	Autore
28/11/2020	v0.1	Prima stesura con obiettivi di sistema	Giovanni Rapa
29/11/2020	v0.2	Aggiunta la suddivisione in sottosistemi	Adriano Amato
29/11/2020	v0.3	Aggiunta architettura client server e Database	Angelo Afeltra
30/11/2020	v0.3.1	Revisione del lavoro fatto fin ora	Angelo Afeltra
02/12/2020	v1	Aggiunta di Design Goals & Trade-offs, Diagramma di deployment	[Tutti]
02/12/2020	v1.1	Modifica Database	Andrea Fucile
03/12/2020	v1.2	Aggiunte Condizioni limite	Angelo Afeltra
03/12/2020	v1.3	Aggiunta Servizi dei Sottosistemi	Adriano Amato

1. Introduzione

1.1 Obiettivi di sistema

L'attesa agli sportelli ospedalieri è una problematica sempre più presente e fastidiosa, sia per i "clienti" sia per il personale che vede una mole importante di persone ad aspettare pazientemente (o meno) il proprio turno. Abbiamo così ideato MedQueue!

Proponiamo un sistema che nasce dalla volontà di voler diminuire i tempi di attesa sempre di più, sia per ottimizzare il tempo di entrambe le parti sia, in questi tempi che corrono, per rispettare le ordinanze anti-Covid-19. Si vuole realizzare una piattaforma web come interfaccia per l'utente che utilizzerà un semplice browser web e un semplice programma per il personale delle strutture.

Il nostro sistema ha la necessità di gestire i dati persistenti: prenotazioni, strutture disponibili e informazioni su di esse e i dati dell'utente. Da tale database attingerà un'applicazione web deputata alla gestione delle interazioni con l'utente ed alla manipolazione dei suddetti dati, e un'applicazione fornita alla struttura aderente a MedQueue da permettere all'impiegato di svolgere la gestione delle prenotazioni.

Viene garantito il controllo degli accessi alla piattaforma tramite l'autenticazione in seguito all'inserimento della propria mail e di una password.

1.2 Design Goals & Trade-offs

Illustriamo nella seguente tabella gli obbiettivi di design per il sistema e le relative priorità (a numeri più bassi corrispondono priorità più elevate). Per ogni obbiettivo riportiamo anche l'origine, facendo riferimento, in particolare, all'identificativo del requisito non funzionale ad esso associato.

Priorità	ID	Descrizione	Categoria	Origine
1	DG_1	**Leggibilità:** Il codice prodotto dev'essere semplice da comprendere. Ogni metodo e campo non banale dev'essere documentato opportunamente al fine di aumentarne la comprensione	Manutenzione	RNF-S1
2	DG_2	**Robustezza:** Vogliamo proporre un sistema che abbia la capacità di sopravvivere ad input non validi immessi dall'utente. Pertanto, il sistema deve garantire il filtraggio dei dati inconsistenti o errati inseriti dall'utente, invitandolo a reinserirli.	Dependability	RNF-A3
3	DG_3	**Affidabilità:** Il sistema dev'essere in grado di riconoscere situazioni anomale e prevenire modifiche ai dati persistenti al fine di garantire la consistenza	Dependability	RNF-A1
3	DG_4	**Sicurezza:** Il sistema prevede l'immissione da parte degli utenti di dati sensibili, si rende necessario fornire uno strumento di autenticazione sicuro, composto dalla richiesta di username e	Dependability	RNF-A2

		password prima di ogni accesso ad informazioni riservate. Le suddette password saranno crittografate		
2	DG_5	**Costi di sviluppo:** Lo sviluppo del prodotto richiederà costi ridotti sia in termini di risorse umane (per cui è fissato un tetto di 75 ore-lavoro), sia in termini economici (per cui si punta a ricorrere a soluzioni off-the-shelf open source)	Costo	Top management
3	DG_6	**Usabilità:** Il sistema deve essere facile da apprendere ed intuitivo da utilizzare senza necessariamente consultare la documentazione. I contenuti dovranno essere fruibili attraverso dispositivi sia desktop che mobile ed accessibili attraverso un numero ridotto di interazioni	End User	RNF-U1 RNF-U2
2	DG_7	**Tempi di risposta:** Il sistema deve elaborare le richieste e produrre output in meno di 2 secondi (al netto di ritardi dovuti alla trasmissione su rete)	Performance	RNF-P1
3	DG_8	**Throughput:** Il sistema deve permettere l'interazione contemporanea di almeno 100 utenti diversi	Performance	RNF-P2
1	DG_9	**Estensibilità:** Il sistema deve agevolare l'introduzione di nuove funzionalità	Manutenibilità	RNF-S3
1	DG_10	**Modificabilità:** Le funzionalità del sistema devono essere facilmente modificabili	Manutenibilità	RNF-S2

Riportiamo ora quelli che sono i compromessi considerati e la posizione del team in relazione ad ognuno di essi.

Tempo di rilascio vs Funzionalità
Sebbene i tempi siano piuttosto proibitivi, preferiamo consegnare con leggero ritardo un prodotto che faccia ciò che promette piuttosto che un prodotto che non possa essere utilizzato a causa della mancanza di funzionalità.
Prestazioni vs Costi
Considerato il budget ridotto a disposizione, si preferisce rientrare nei costi dedicando un numero ridotto di ore-lavoro alla massimizzazione delle prestazioni.
Prestazioni vs Affidabilità
I dati gestiti dal sistema sono piuttosto sensibili, pertanto preferiamo garantire un maggior controllo di input e consistenza a scapito dei tempi di risposta

1.3 Definizioni, acronimi e abbreviazioni

- Webbapp: abbreviazione per "applicazione web"

1.4 Riferimenti

- Requisiti funzionali: Sezione 3.2 del RAD
- Requisiti non funzionali: Sezione 3.3 del RAD
- RabbitQueue

1.5 Panoramica

Nel documento verranno affrontati l'analisi delle architetture di sistemi simili, la decomposizione in sottosistemi del sistema proposto con la definizione della strategia di deploy e le condizioni limite. Verranno quindi definiti i servizi esposti da ciascun sottosistema.

2. Architettura di Sistemi simili

Dopo varie ricerche non abbiamo trovato altri software già esistenti nella realtà in cui vogliamo calarlo.

Prendiamo in considerazione il sistema implementato alle Poste Italiane per la gestione code con la possibilità di prenotarsi online negli uffici che decidono di supportare questa metodologia di gestione.

Dall'analisi si è arrivati alla conclusione che il sistema analizzato ha come base la memorizzazione dei dati persistenti e la gestione dinamica delle code interrogando e aggiornando i dati tramite un'interfaccia web e un semplice programma.

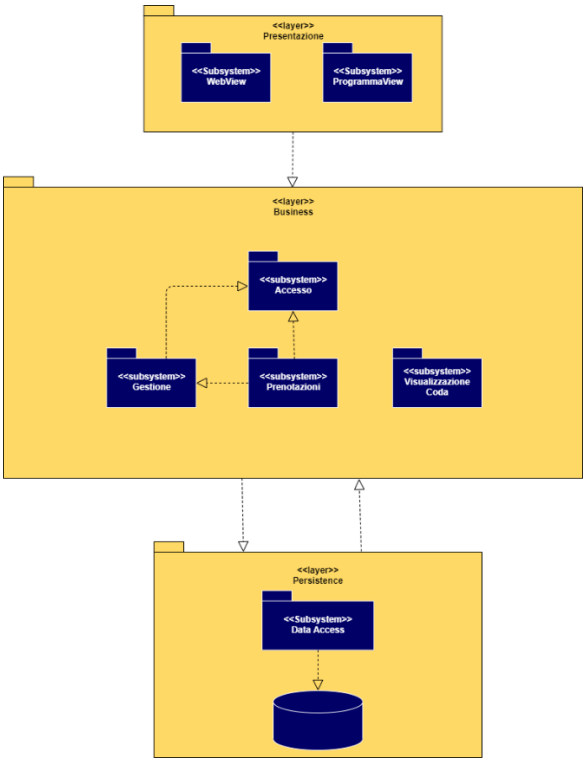
Il sistema preso in considerazione prevede un controllo degli accessi tramite username e password per poter usufruire di tutte le funzionalità.

3. Architettura del Sistema Proposto

3.1 Panoramica

Si può usufruire dei servizi di MedQueue tramite interfaccia web o programma apposito (per gli impiegati). Si ricorre all'utilizzo di un database relazionale per il salvataggio dei dati persistenti.

3.2 Decomposizione in sottosistemi



Il sistema è suddiviso in 3 livelli logici: presentazione, business e persistenza che si occupano rispettivamente di presentazione delle informazioni all'utente, definizione della logica applicativa e gestione dei dati persistenti.

Il livello di presentazione è composto da due sottosistemi:

- Web app: definisce l'interfaccia utente
- Programma: definisce l'interfaccia dell'impiegato

Il livello business è composto da quattro sottosistemi:

- Gestione: modella il lato di gestione delle prenotazioni da parte dell'impiegato
- Prenotazioni: modella il lato di inserimento, eliminazione, visualizzazione e convalida delle prenotazioni da parte degli utenti
- Accesso: Definisce l'utente generico del sistema ed offre tutti i servizi relativi all'applicazione
- Visualizzazione Coda: Modella le operazioni di visualizzazione coda

Il livello di persistenza invece è composto da un solo sottosistema:

- Data access: si occupa del reperimento del salvataggio delle informazioni manipolate da Web app dal/sul database sottostante, quando serve tramite RabbitMQ

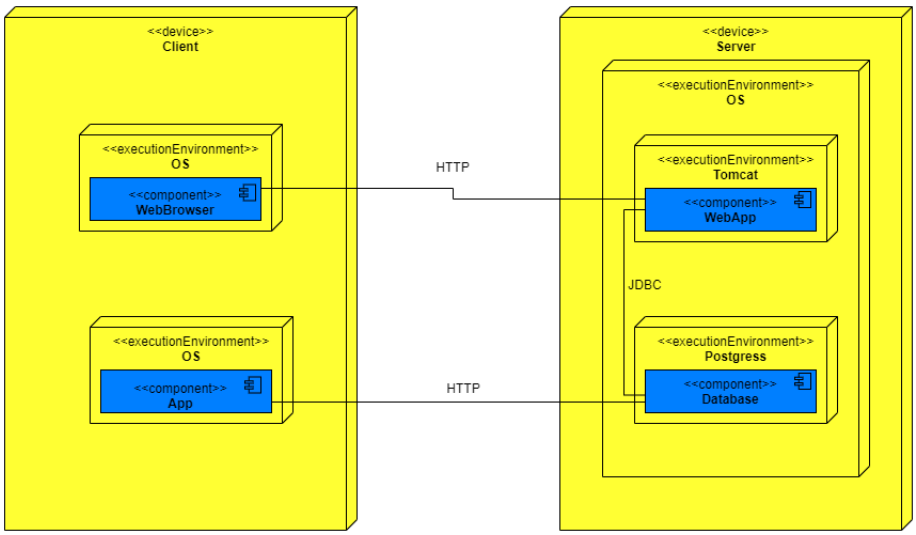
Si noti come la divisione in sottosistemi sia stata realizzata tramite una strutturazione 3-layeral fine di disaccoppiare l'interfaccia dalla logica di business dell'applicazione: il livello di business non è infatti a conoscenza di come l'informazione sarà presentata all'utente e ciò permette, in futuro, di poter realizzare un client mobile oppure un'interfaccia desktop piuttosto che web.

Nella divisione in sottosistemi sono stati utilizzati concetti fondamentali del pattern MVC, con l'allocazione di View e Controller al livello di presentazione e la realizzazione del Model tramite i livelli di Business e Data Access

3.2.3 Diagramma di deployment

MedQueue consiste di un'applicazione distribuita installabile su un qualsiasi server in grado di eseguire Java e Postgres (data la ridotta quantità di dati da gestire, le suddette componenti sono installate sulla stessa macchina), e di un'applicazione installabile su qualsiasi computer dell'ufficio che permetta all'impiegato di autenticarsi e accettare prenotazioni.

Il sistema sarà accessibile tramite comuni browser web installati sui dispositivi a disposizione degli attori e dall'applicazione disponibile per l'impiegato



3.3 Mapping hardware / software

MedQueue si compone di tre componenti principali:

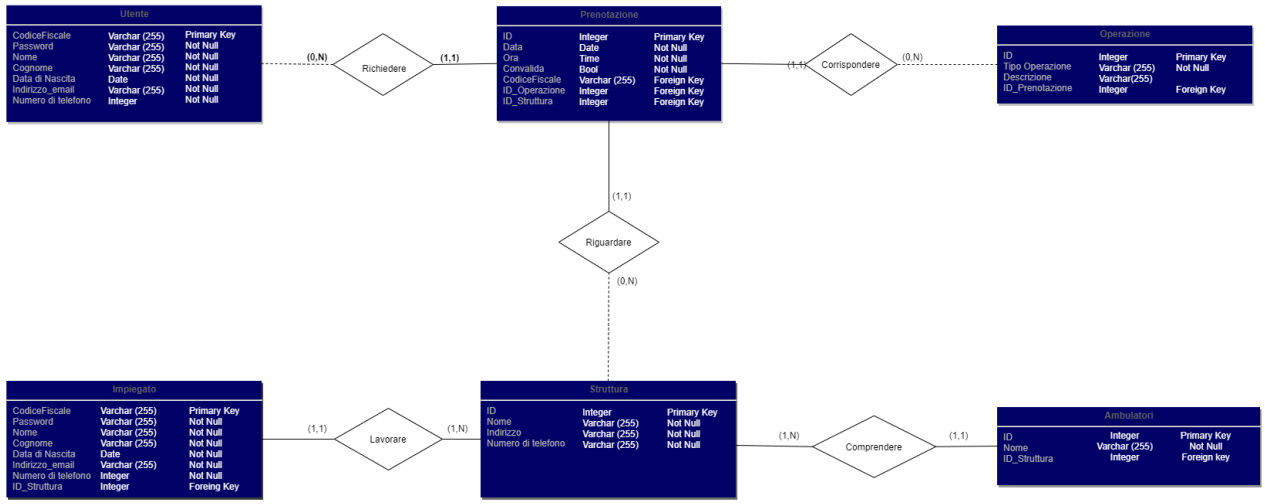
- Webapp, cui saranno allocati i layer di presentazione e di business oltre al sottosistema di data acces
- Database, realizzante i layer di persistenza
- Applicazione (impiegato) cui saranno allocati i layer di presentazione dell'applicazione e di business oltre al sistema di data access

Il sistema necessita di una macchina in grado di supportare Apache Tomcat, al fine di garantire l'operabilità della WebApp, e Postgress per garantire invece l'operabilità del database con cui la WebApp si interfaccia.

WebApp e DBMS saranno quindi installati sullo stesso nodo in modo da ridurre i possibili fallimenti o ritardi di propagazione delle informazioni dovuti a problemi di connettività.

3.4 Gestione dei dati persistenti

Per la gestione dei dati persistenti, MedQueue si affida ad un database relazione gestito tramite Postgress. La struttura dei dati memorizzati segue il seguente schema:



3.5 Controllo degli accessi e sicurezza

Il controllo degli accessi è garantito tramite l'utilizzo di username e password per gli utenti del sistema che hanno possibilità di creare o modificare gli oggetti che modellano entità di dominio, così da prevenire accessi non autorizzati ad informazioni sensibili. Sottolineiamo che il sistema non fornirà un metodo di recupero o modifica delle password, almeno nella sua prima versione.

Si ricorrerà all'utilizzo della sessione del server per tenere traccia dell'utente loggato. Per questioni di efficienza, la sessione sarà attiva per soli 30 minuti dopo l'ultima interazione dell'utente col sistema.

Per questioni legate al budget a disposizione del team, il salvataggio delle password sarà in chiaro su database: non ci sarà alcun tipo di cifratura, almeno nella prima versione del sistema.

Nelle prime versioni non sarà inoltre utilizzato SSL su connessione HTTP tra client e server ma non ne è escluso l'utilizzo in futuro.

Le operazioni che gli utenti dell'applicazione web possono effettuare sugli oggetti sono riportate nella tabella che segue:

Oggetto Attore	Utenza	Prenotazione	Visualizzazione Coda	Gestione
Ospite	Registrazione		Visualizzazione Coda Prenotazione	
Utente Registrato	Autenticazione Logout			
Utente		Richiesta Prenotazione, Convalida Prenotazione, Visualizzazione Prenotazioni, Elimina Prenotazione	Visualizzazione Coda Prenotazione	
Impiegato	Autenticazione Logout			Accettazione Prenotazioni

Dalla tabella si evince come l'utente non abbia interazioni dirette con il sottosistema di accesso al database, cui invece accedono i singoli sottosistemi di business: per questo motivo si è deciso di non riportarlo nella matrice d'accesso.

Questa soluzione permette al sistema di poter rispondere a più utenti contemporaneamente ma richiede che gli accessi in scrittura ai dati persistenti avvengano sequenzialmente, gestendo opportunamente le sezioni critiche.

In generale, ogni richiesta da parte di un utente verrà eseguita in un thread dedicato.

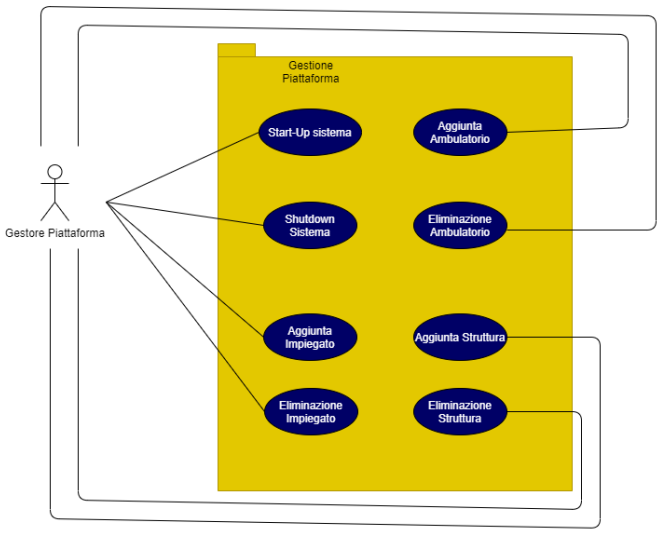
3.6 Controllo flusso globale del sistema

Il sistema adotta un controllo del flusso globale di tipo thread-driven, questo perché il web container (Tomcat) web permette l'interazione concorrente tra le WebApp e più client tramite l'intercettazione di eventi generati proprio da questi ultimi.

Questa soluzione permette al sistema di poter rispondere a più utenti contemporaneamente e avere accessi in scrittura ai dati persistenti in maniera sincrona, gestendo opportunamente le sezioni critiche.

In generale, ogni richiesta da parte di un utente verrà eseguita in thread dedicato.

3.7 Condizione limite



3.7.1 Start-up

Identificativo *UC_GP1*		*Start-up sistema*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di avvio del sistema per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso alla macchina su cui è installato il sistema		
Exit condition On success		Il sistema è avviato correttamente		
Exit condition On failure		Il sistema non è avviato		
Rilevanza/User Priority		Elevata		
Frequenza stimata		1/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Accende il server Lancia il servizi del DBMS Lancia il web container		
2	Sistema:	Comunica al gestore che lo startup si è concluso con successo		
I Scenario/Flusso di eventi Alternativo: Non è possibile avviare il sistema				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.2 Terminazione

Identificativo *UC_GP1*		*Shutdown sistema*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Giovanni Rapa*
Descrizione		*Lo UC fornisce al gestore della piattaforma la possibilità di terminare il sistema*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso alla macchina su cui è installato il sistema		
Exit condition On success		Il sistema è terminato correttamente		
Exit condition On failure		Il sistema resta in esecuzione		
Rilevanza/User Priority		Elevata		
Frequenza stimata		1/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Termina il servizio del web container		
2	Sistema:	Comunica al gestore che il servizio è stato terminato correttamente		
3	Attore:	Termina il servizio del DBMS		
4	Sistema:	Comunica al gestore che il servizio è stato terminato correttamente		
I Scenario/Flusso di eventi Alternativo: Non è possibile terminare il sistema				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.3 Aggiunta Impiegato

Identificativo *UC_GP2*		*Aggiunta Impiegato*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di aggiunta di un impiegato al database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Impiegato aggiunto al database		
Exit condition On failure		L'impiegato non è stato aggiunto al database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		6/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per aggiungere un impiegato al database		
2	Sistema:	Comunica al gestore che l'impiegato è stato aggiunto con successo		
I Scenario/Flusso di eventi Alternativo: Non è possibile aggiungere l'impiegato				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.4 Eliminazione Impiegato

Identificativo *UC_GP3*		*Eliminazione Impiegato*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di eliminare un impiegato dal database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Impiegato eliminato dal database		
Exit condition On failure		L'impiegato non è stato eleminato dal database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		6/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per eliminare un impiegato dal database		
2	Sistema:	Comunica al gestore che l'impiegato è stato eliminato con successo		
I Scenario/Flusso di eventi Alternativo: Non è possibile eliminare l'impiegato				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.5 Aggiunta Struttura

Identificativo *UC_GP4*		*Aggiunta Struttura*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di aggiungere una struttura al database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Struttura inserita nel database		
Exit condition On failure		La struttura non è stata inserita nel database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		4/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per aggiungere una struttura al database		
2	Sistema:	Comunica al gestore che la struttura è stata aggiunta correttamente		
I Scenario/Flusso di eventi Alternativo: Non è possibile aggiungere la struttura				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.6 Eliminazione Struttura

Identificativo *UC_GP5*		*Eliminazione Struttura*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di eliminare una struttura dal database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Struttura eliminata dal database		
Exit condition On failure		La struttura non è stata eliminata dal database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		1/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per eliminare una struttura dal database		
2	Sistema:	Comunica al gestore che la struttura è stata eliminata correttamente		
I Scenario/Flusso di eventi Alternativo: Non è possibile eliminare la struttura				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.7 Aggiunta Ambulatorio

Identificativo *UC_GP6*		*Aggiunta Ambulatorio*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di aggiungere un ambulatorio al database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Ambulatorio aggiunto al database		
Exit condition On failure		L'ambulatorio non è stato aggiunto al database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		2/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per aggiungere un ambulatorio al database		
2	Sistema:	Comunica al gestore che l'ambulatorio è stato aggiunto con successo		
I Scenario/Flusso di eventi Alternativo: Non è possibile aggiungere l'ambulatorio				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.8 Eliminazione Ambulatorio

Identificativo *UC_GP7*		*Eliminazione Ambulatorio*	*Data*	*03/12/2020*
			Vers.	*0.00.001*
			Autore	*Angelo Afeltra*
Descrizione		*Lo use case definisce la funzionalità di eliminare un ambulatori dal database per il gestore della piattaforma*		
Attore Principale		**Gestore piattaforma**		
Attori secondari		**NA**		
Entry Condition		Il gestore ha accesso al database		
Exit condition On success		Ambulatorio eliminato dal database		
Exit condition On failure		L'ambulatorio non è stato eliminato dal database		
Rilevanza/User Priority		Elevata		
Frequenza stimata		1/anno		
Extension point		NA		
Generalization of		NA		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	Il gestore esegue il comando per eliminare una ambulatorio dal database		
2	Sistema:	Comunica al gestore che l'ambulatorio e stato eliminato		
I Scenario/Flusso di eventi Alternativo: Non è possibile eliminare l'ambulatorio				
4.1	Sistema:	Mostra al gestore un messaggio che ne specifica il motivo		

3.7.9 Fallimento

MedQueue può incorrere in diversi casi di fallimento, riguardanti sia l'hardware che il software:

- Fallimenti Hardware
  - Crash del disco su cui i dati persistenti sono salvati: il sistema non prevede alcuna strategia di backup e ripristino dei dati
- Fallimenti nell'ambiente di esecuzione
  - Interruzione della fornitura elettrica al server: il sistema non prevede alcuna strategia che ne garantisca l'operabilità in questo tipo di condizione
- Fallimenti Software
  - Impossibilità di stabilire una connessione col database: il sistema mostra all'utente una schermata che riporta il rilevamento di un errore interno

4. Servizi dei Sottosistemi

Data Access	
Servizio	**Descrizione**
Caricamento account	Il sottosistema permette di caricare un account nel database
Cancellazione account	Il sottosistema permette di cancellare un account nel database
Caricamento struttura	Il sottosistema permette di caricare una struttura nel database
Modifica struttura	Il sottosistema permette di modificare una struttura nel database
Elimina struttura	Il sottosistema permette di eliminare una struttura nel database

Prenotazioni	
Servizio	**Descrizione**
Caricamento prenotazione	Il sottosistema permette di caricare una prenotazione nel database
Cancellazione prenotazione	Il sottosistema permette di cancellare una prenotazione nel database
Convalida prenotazione	Il sottosistema permette di convalidare una prenotazione effettuata, inserendola nella RabbitQueue

Gestione	
Servizio	**Descrizione**
Accettazione prenotazione	Il sottosistema permette di accettare una prenotazione, prelevandola da RabbitQueue ed eliminandola dal database

Accesso	
Servizio	**Descrizione**
Registrazione	Il sottosistema permette di creare un nuovo account e salvarne i dati nel database
Login	Il sottosistema permette di autenticare un account presente nel database
Logout	Il sottosistema permette di abbandonare la sessione di un utente che ha effettuato il login

5. Glossario

Postgres: Database relazionale utilizzato per la gestione dei dati.

Java: linguaggio di programmazione orientato agli oggetti

Web app: programma accessibile tramite browser web ed in grado di elaborare richieste e risposte http

Programma: applicazione sviluppata in linguaggio Java

Application Server: sistema software per la gestione delle richieste/risposte provenienti dai client

Apache Tomcat: specifico application server.