

SDD

System Design Document

MedQueue

Afeltra Angelo

Amato Adriano

Fucile Andrea

Rapa Giovanni

Sommario

1. **Introduzione**
   1. Obiettivi del sistema
   2. Design Goals & Trade-offs

Tempo di rilascio vs Funzionalità

Prestazioni vs Costi

Prestazioni vs Affidabilità

* 1. Definizioni, acronimi e abbreviazioni
  2. Riferimenti
  3. Panoramica

1. **Architettura di Sistemi simili**
2. **Architettura del Sistema proposto**
   1. Panoramica
   2. Decomposizione in sottosistemi

3.2.3

* 1. Mapping hardware / software
  2. Gestione dati persistenti
  3. Controllo degli accessi e sicurezza
  4. Controllo flusso globale del sistema
  5. Condizione limite
     1. Start-up
     2. Terminazione
     3. Fallimento

1. **Servizi dei Sottosistemi**

**Glossario**

1. **Introduzione**
   1. **Obiettivi di sistema**

L’attesa agli sportelli ospedalieri è una problematica sempre più presente e fastidioso, 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 e sia per, in questi tempi che corrono, 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, ovviamente 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.

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

* 1. **Design Goals & Trade-offs (TUTTO DA RIVEDERE, HO PRESO QUELLO CHE PENSAVO SERVISSE A NOI DAL PROGETTO DI RIFERIMENTO)**

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 password prima di ogni accesso ad informazioni riservate. Le suddette password saranno crittografate | Dependability | RNF-A2 |
| 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 di 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. **Definizioni, acronimi e abbreviazioni**
* Webapp: abbreviazione per “applicazione web”
  1. **Riferimenti**
* Requisiti funzionali: Sezione 3.2 RAD
* Requisiti non funzionali: Sezione 3.3 RAD
* ……. CONTINUARE
  1. **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.

1. **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.

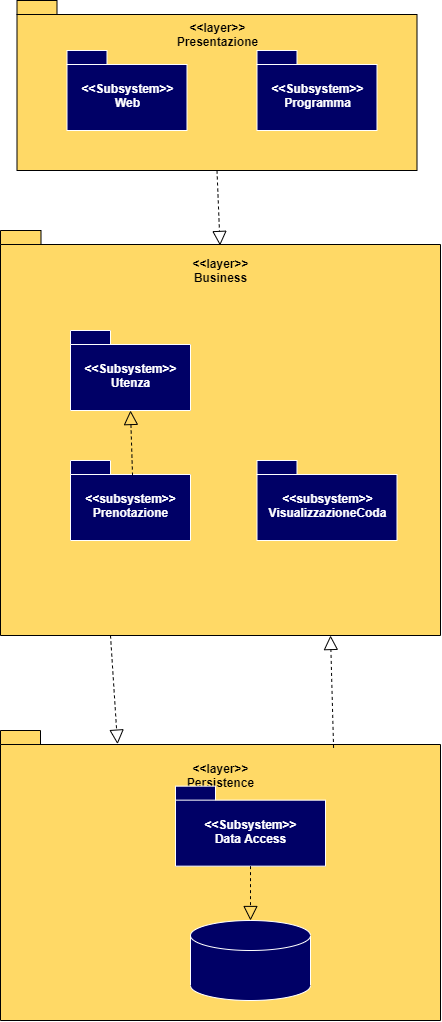
Il sistema preso in considerazione prevede un controllo degli accessi tramite username e password per poter usufruire di tutte le funzionalità.

1. **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.

* 1. **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: definisce l’interfaccia utente e si occupa del controllo del flusso di esecuzione
* App: definisce l’interfaccia dell’impiegato e si occupa del controllo del flusso di esecuzione

Il livello di business è composto da tre sottosistemi:

* Utenza: Definisce l’utente generico del sistema ed offre tutti i servizi relativi all’applicazione
* Prenotazione: Modella tutto ciò che riguarda la prenotazione
* VisualizzazioneCoda: Modella le operazioni di visualizzazione coda

Il livello persistenza è invece composto da un solo sottosistema:

* Data access: Si occupa del reperimento e del salvataggio delle informazioni manipolate dalla webapp e dalla applicazione dal/sul database sottostante

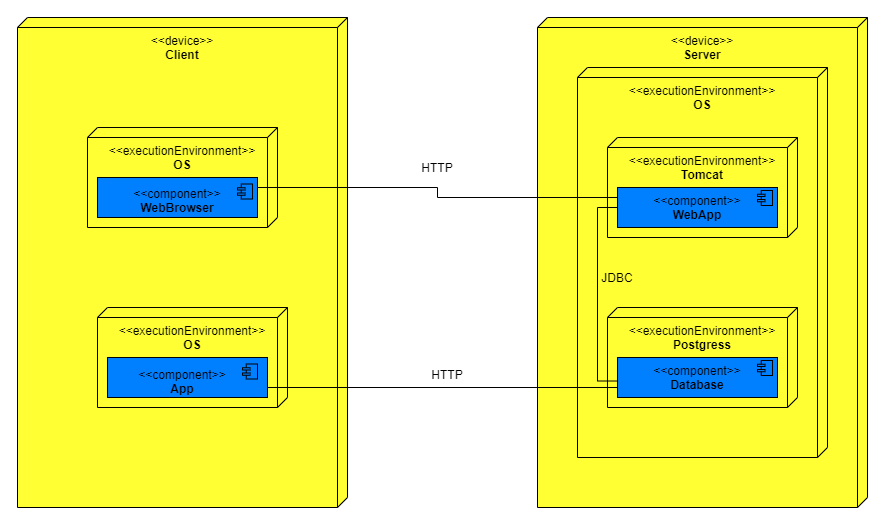
Si noti come la divisione in sottosistemi sia stata realizzata tramite una strutturazione 3-layer al fine di disaccoppiare l’interfaccia dalla logica di business dell’applicazione: il livello di business non è infatti a conoscenza di come l’informazione sarà presentata all’utente e ciò permette, in futuro, 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 quantita di dati da gestire, le suddette componenti sono installate sulla stessa macchina), e di un applicazione installabile su qualsiasi computer del 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



* 1. **Mapping hardware / software**

MedQueue si compone di tre componenti principali:

* Webapp, cui saranno allocati i layer di presentazione e 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 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:

INSERIRE DATABASE

**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:

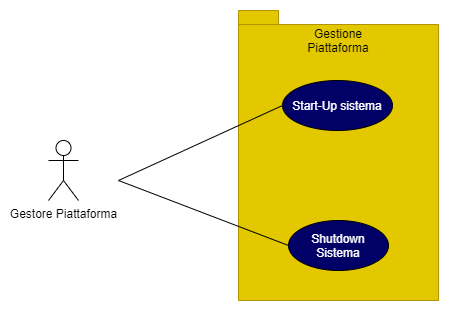
|  |  |  |  |
| --- | --- | --- | --- |
| Oggetto  Attore | Utenza | Prenotazione | Visualizzazione Coda |
| Ospite | Registrazione |  | Visualizzazione CodaPresenza |
| Utente Registrato | Autenticazione  Logout |  |  |
| Utente |  | Richiesta Prenotazione,  Convalida Prenotazione,  Visualizzazione Prenotazioni,  Elimina Prenotazione | Visualizzazione CodaPresenza |
| 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.

**3.7 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 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

1. **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 |
| Caricamento Prenotazione | Il sottosistema permette di caricare una prenotazione nel database |
| Eliminazione Prenotazione | Il sottosistema permette di eliminare una prenotazione dal database |
| Visualizzazione Prenotazioni | Il sistema permette di visualizzare le prenotazioni contenute nel database |
| Visualizzazione Prenotazioni convalidate | Il sistema permette di visualizzare le prenotazioni convalidate presenti del database |
| Informazioni Prenotazione | Il sistema permette di ottenere le informazioni relative alle prenotazioni salvate sul database |
| Informazioni utente | Il sistema permette di ottenere le informazioni relative agli utenti salvati sul database |

|  |  |
| --- | --- |
| Utenza | |
| Servizio | Descrizione |
| Autenticazione | Il sottosistema permette di effettuare un operazione di autenticazione tramite codice fiscale e password |
| Logout | Il sottosistema permette di effettuare un operazione di logout |
| Ottenimento utente autenticato | Il sistema permette di ottenere l’utente autenticato |

|  |  |
| --- | --- |
| Prenotazione | |
| Servizio | Descrizione |
| Richiesta Prenotazione | Il sottosistema permette di effettuare una richiesta di prenotazione scegliendo data, ora, struttura e ambulatorio |
| Elimina Prenotazione | Il sottosistema permette di effettuare una operazione di eliminazione prenotazione tramite l’id prenotazione |
| Convalida Prenotazione | Il sottosistema permette di effettuare una operazione di convalida prenotazione tramite il codice fiscale, data e ora |
| Visualizza Prenotazioni | Il sottosistema permette di effettuare un operazione di visualizzazione prenotazioni tramite il codice fiscale |
| Accetta Prenotazione | Il sottosistema permette di effettuare un operazione di accettazione prenotazione tramite l’id prenotazione |

|  |  |
| --- | --- |
| Visualizza Coda | |
| Servizio | Descrizione |
| Visualizza Coda di Presenza | Il sottosistema permette di visualizzare la coda di presenza tramite data e ora |
| Visualizza Coda Prenotazioni | Il sottosistema permette di visualizzare la coda di prenotazioni tramite data e ora |