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 |  | 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 |  |
| 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. |  |  |
| 2 |  | 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) |  |  |
| 3 |  | 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 |  |  |
| 2 |  | 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) |  |  |
| 1 |  | Estensibilità: Il sistema deve agevolare l’introduzione di nuove funzionalità |  |  |
| 1 |  | Modificabilità: Le funzionalità del sistema devono essere facilmente modificabili |  |  |

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.

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. // Qualcosa su strati, non so se bisogna inserirlo

Il sistema preso in considerazione prevede un controllo degli accessi tramite username e password per poter usufruire di tutte le funzionalità. //Dati sensibili, non so se è applicabile al nostro caso

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.

3.2 Decomposizione in sottosistemi //Da fare insieme e definire meglio insieme e con la teoria

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 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Queue Management //Maybe in Data Access | |
| Servizio | Descrizione |
| Caricamento prenotazione | Il sottosistema permette di caricare una prenotazione nel database |
| Cancellazione prenotazione | Il sottosistema permette di cancellare una prenotazione nel database |

|  |  |
| --- | --- |
| Utenza | |
| Servizio | Descrizione |
| Creazione prenotazione | Il sottosistema permette di caricare una prenotazione nel database |
| Cancellazione prenotazione | Il sottosistema permette di cancellare una prenotazione nel database |