SanityManager

DOCUMENTO DI PROGETTO PER INTRODUZIONE ALLA PROGRAMMAZIONE PER IL WEB

a.a 2018/19

Azzolin Steve – Carretta Riccardo – Destro Matteo

In questo testo saranno elencate particolari assunzioni adottate per lo svolgimento del progetto e saranno descritte feature non espressamente richieste ma che per varie ragioni, solitamente elencate assieme alla descrizione della stessa, sono state implementate.

Per la persistenza dei dati abbiamo usato MySQL.

Assunzioni particolari:

1. **Esami specialistici = Visite specialistiche:** Netta separazione tra i compiti del **Medico Specialista** e il **Servizio Sanitario Provinciale** (a.k.a SSP). Tutti gli esami prescrivibili sono completati da **SSP**, eventuali esami specialistici che andrebbe compilati da un **Medico Specialista** sono trattati come visite specialistiche.

2. **Scelta data esame/visita specialistica:** Una volta che il **Medico di base** ha prescritto un esame o una visita specialistica ad un **Paziente** spetta ad esso stabilire la data in cui verrà sostenuto/a. Nel nostro sistema la scelta della data è libera (non prende in considerazioni le disponibilità di **SSP/Medico Spec.**)

3. **Competenza Medico Specialista:** Ad ogni **Medico Specialista** sono assegnate particolare competenza (categorie di visite per cui è abilitato). Il **Medico Spec**. è abilitato a vedere tutti i Pazienti, è però abilitato a compilare visite solo di sua competenza e fissate per il giorno corrente. Non vede visite specialistiche prescritte dal **Medico** di base ma che non sono ancora state fissate dal **Paziente**

Feature implementate:

1. **Utilizzo tag fmt per i testi:** Per supportare eventuali espansioni dell' Applicazione abbiamo deciso di utilizzare fin dall'inizio il supporto all' internazionalizzazione offerta dai tag **JSTL fmt**. E' però comunque disponibile 1 sola lingua.

2. **Integrazione tra tecnologia Servlet e WebService Rest:** L'Applicazione sfrutta la tecnologia dei **WebService Rest** per implementare la ricerca tramite *suggestion box*

degli elenchi di esami/farmaci/visite specialistiche. Lato client abbiamo usato il componente **Select2** con modalità **AJAX.** In questo modo non è necessario scaricare al client ogni volta tutto l'elenco completo.

3. **Prenotazioni per recarsi dal Medico di base:** L'Applicazione permette ad ogni **Paziente** di prenotare il proprio turno dal **Medico di base**, in modo da evitare noiose ed indefinite attese. Permette semplicemente di selezionare uno slot orario preso da un pool fisso. Non prende in considerazione eventuali impegni particolari del **Medico.** Potrebbe essere considerato come punto di partenza per un sistema più complesso di gestione delle prenotazioni.

4. **Sezione statistiche**: Ogni tipologia di utente ha una propria sezione statistiche che mostra una serie di grafici. Il contenuto dei grafici varia a seconda della tipologia di utente e in base a cosa ha senso mostrare come statistica per quell' utente. Abbiamo adottato una struttura generalizzata per la pagina stats.jsp, in modo da averne 1 comune per tutte le tipologie di utente indipendentemente da cosa viene mostrato (disambiguazione tramite tag **JSTL**)

5. **Controllo sui permessi degli utenti**: L' Applicazione è stata costruita in maniera tale da cercare di limitare il più possibile lo spettro di azioni degli utenti, limitandoli alle sole operazioni che sono autorizzati a fare (*last provilage*). Abbiamo implementato una serie di controlli sui filtri (uno specifico per ogni tipo di utente), e non solo. Per esempio il **Medico** può accedere alle sole immagini profilo dei suoi pazienti, oppure il **Paziente** può vedere solo le sue Ricette/Esami ecc... . Se per esempio un attaccante cerca di accedere alla foto profilo di un utente X, ottiene come messaggio di errore *'non autorizzato'* (lo stesso che otterebbe un **Medico** che accede ad un utente che non è suo **Paziente**)*,* in questo modo non riesce a capire se l'utente esiste nel sistema o se semplicemente non ha accesso ai suoi dati

6. **Tentativo di error recovery:** Le pagine per la creazione/prescrizione di nuove Ricette/Esami.. sono state costruite in modo tale che se si verifica un errore nella query di inserimento nel **DB** la pagina ripropone direttamente i dati precedentemente inseriti, così da favorire l' utente che sta inserendo i dati. Attualmente questa copertura è rivolta ai soli errori strettamente dipendenti alle query di inserimento nel **DB**.

7. **Log dei tempi:** Il filtre che segue il controllo sulla sessione attiva (quindi comune a tutte la pagine tranne la landing page) memorizza in una tabella apposita il tempo stimato di processamento della richiesta. Queste info possono essere utilizzate da noi sviluppatori per individuare criticità e punti su cui intervenire per garantire la massima *user-experience.*

8. **Struttura delle pagine modulare:** Durante lo sviluppo di questa piattaforma abbiamo cercato di assimilare varie parti comuni delle pagine web in diversi moduli riutilizzabili. In questo modo la pagina per la ceazione delle Visite per Paziente,Medico,Medico Spec. è la stessa, in cui vengono solo specificate le differenze (solitamente tramite tag **JSTL**). Questa struttura basata sulle differenze permette di risparmiare tempo durante lo sviluppo.

9. **Ridimensionamento immagini profilo:** Quando un utente carica la propria immagine profilo viene creata una copia ridimensionata dell' immagine. L'immagine profilo ridimensionata è quella che viene mostrata nelle tabelle con gli elenchi dei pazienti, in modo da risparmiare banda in download e per rendere più veloce il caricamento delle pagine. L' immagine originale è mostrata nella pagina di dettaglio dei pazienti o nelle impostazioni personali.

Da togliere probabilemnte

Menzioni (secondo noi) meritevoli:

Illustriamo brevemente la struttura modulare delle pagine che abbiamo cercato di mantenere durante lo sviluppo delle pagine con l'obiettivo di ridurre la scrittura di codice simile e quindi velocizzare lo sviluppo e renderlo più facilemente mantenibile. In particolare vorremmo soffermarci sulla gestione delle suggestion_box per gli elenchi di Ricette/Esami/Visite spec.

Ogni pagina che necessita di una **Select2** per la ricerca dinamica contiene il seguente codice:

```
<select <c:if test="${empty i_visita}">id="autocomplete"</c:if> altri_attr..>
    <c:if test="${not empty i_visita}">
        <option selected value="${i_visita.getId_visita_spec()}">
            ${i_visita.getNome_visita()}
        </option>
    </c:if>
</select>
```

I tag <*c:if*> servono per rendere readonly la Select2 nel caso in cui voglia visualizzare i dati di una visita anzichè crearne una nuova (la pagina nei due casi è la stessa, se è settato un oggetto i_visita voglio mostrare i dati di quella visita, settato come attributo di *request*).

Supponiamo che il primo *test* sia *TRUE,* quindi che voglia creare una nuova visita.
Per evitare

Il file head.jsp contiene i link (comuni) a tutte le pagine per Bootstrap,jQuery,DataTable ecc... ma anche :