RELAZIONE PROGETTO

29 GENNAIO

LATEOTT

Alessandro Oniarti

Gianluca Mazzucchi

LATEOTT

Documentazione prog. Sistema Sanitario

Introduzione alla Programmazione Web – a.a. 2019/20

Gruppo LATEOTT

Alessandro Oniarti	Gianluca Mazzucchi
185585	*186379*

Descrizione generale del progetto

Il progetto consiste nella creazione di un sito web responsive che simuli il funzionamento di un Sistema Sanitario. Il sistema deve permettere agli utenti di poter visionare i propri appuntamenti in programma e il proprio storico per quanto riguarda le visite, gli esami, le ricette e i ticket pagati.
Per quanto riguarda la sezione dedicata ai medici di base, il sito permette loro di vedere gli appuntamenti in programma, quelli lasciati in sospeso e quelli già completati. Permette inoltre di avere una lista di tutti i propri pazienti, di poterne vedere le informazioni e lo storico, di erogare una visita potendo prescrivere esami e/o ricette.
È presente anche una sezione dedicata all'Ssp, il quale deve poter visualizzare le ricette prescritte per ogni giorno con relative informazioni.

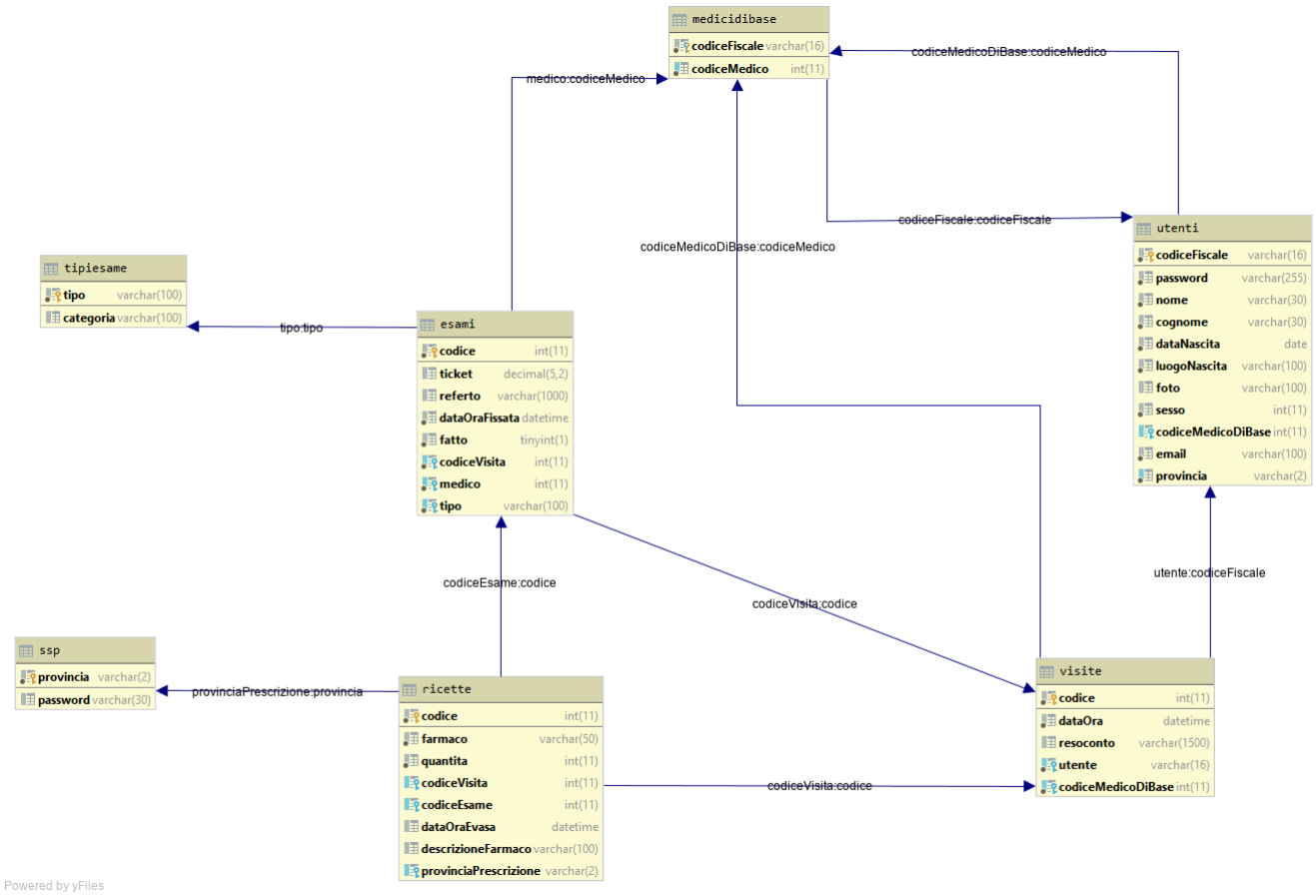
Architettura

Il nostro programma è strutturato in modo da rispettare l'MVC, che abbiamo implementato in questa maniera:

- **DAOs** (Data Access Object): abbiamo implementato un pattern per astrarre l'accesso al database. Il DAO contiene dei metodi che si occupano di eseguire query sul database, in questo modo la business logic si limiterà a chiamare i metodi offerti senza preoccuparsi della loro implementazione, nascondendo il layer di persistenza sottostante.
  I dati restituiti dal database vengono poi incapsulati nei JavaBean
- **Sevlet**: utilizzati per la gestione delle chiamate utente. Si occuperanno della modifica dei dati dello stato dell'applicazione (es: quando l'utente vuole effettuare il login verrà chiamato il servlet che con l'ausilio dei DAOs potrà verificare le proprie credenziali, una volta loggato salverà in sessione un flag ad indicando che l'utente è loggato).
- **Filtri**: i filtri hanno 2 scopi, il primo è quello di verificare che l'utente abbia i permessi per accedere a una risorsa, il secondo è quello di inserire i dati da visualizzare nelle pagine nella richiesta, in modo che questi siano accessibili dalla view.
- **Pagine JSP**: permettono di creare dinamicamente la pagina che visualizzerà l'utente. Con l'ausilio dei tag JSTL e servendosi anche di ajax, che si occupa di chiamare asincronamente altre sottopagine, in modo da non dover ricaricare tutti gli elementi ad ogni azione, rendendo il sito più veloce e reattivo.

Database

Come DBMS abbiamo scelto MySQL, in quanto un database relazionale ci è sembrata la tecnologia più adatta a questo progetto. Il database è strutturato come il diagramma seguente:



Alcune considerazioni sul nostro modello:

- Abbiamo pensato che il medico oltre alle funzioni da medico è anche utente, e per questo avrà anche lui un medico associato. La tabella medicoDiBase infatti ha il compito di associare un codice medico agli utenti che sono effettivamente medici.
- Il modello attraverso dei constraints consente la consistenza dei dati, ad esempio un esame può essere prescritto solo in seguito ad una visita.

Front End

La nostra applicazione si divide in 4 macro-pagine.

- La prima è quella del login, che permette all'utente di validarsi secondo 3 categorie, utente normale, medico o ssp
- La seconda è la sezione utente, che permette la visualizzazione e l'aggiornamento delle proprie informazioni personali. La pagina è strutturata su 4 colonne, una con i prossimi appuntamenti, una con gli appuntamenti passati, una per le ricette evase e una per i ticket pagati.
- La terza è la sezione dedicata ai medici, la pagina caricherà altre sottopagine tramite ajax che permetteranno di vedere gli impegni e la lista dei propri pazienti. Da quest'ultima il medico potrà vedere lo storico e compilare visite per i propri pazienti.
- L'ultima è quella dedicata all'Ssp, che permette semplicemente di scaricare il report in formato xls con le ricette prescritte nel tempo in una determinata provincia.

Per realizzare queste pagine in maniera responsive abbiamo utilizzato Bootstrap, javascript e jQuery. Lo stile è principalmente gestito da Boostrap nella sua versione base, in modo da renderlo più uniforme e gestire la disposizione degli elementi più comodamente, non sono mancate comunque delle modifiche tramite Css e javascript/jQuery. Le varie icone presenti sul sito sono prese dal "catalogo" di font-awesome.

Scelte implementative

Abbiamo utilizzato Maven come framework per la gestione delle dipendenze, in modo da scaricarle agevolmente anche dopo un pull di git, anche attraverso diverse IDE.
Per realizzare alcune funzioni richieste abbiamo utilizzato delle librerie esterne, importate appunto con maven:

- **PDF**: per realizzare i PDF richiesti, per la singola ricetta e per la lista di ticket pagati abbiamo utilizzato la libreria PDFbox, che permette di scrivere sulle pagine di un documento PDF, il quale verrà poi restituito all'utente attraverso un servlet.
- **QR**: per realizzare il QR univoco per ogni ricetta abbiamo utilizzato le librerie qrgen e zxing. I dati vengono inseriti nel QR nel formato json, in modo che l'ipotetico lettore della farmacia possa ottenere tutti i dati relativi alla ricetta tramite la lettura dello stesso.

- **XLS**: per la creazione del file XLS abbiamo usato la libreria OpenXLS, che ne permette la creazione in modo semplice.
- **Mail**: per quanto riguarda l'invio delle mail di notifica relative alle nuove prescrizioni abbiamo utilizzato la libreria Javamail. Questa libreria utilizza il protocollo SMTP per l'invio. Al momento dell'invio viene avviato un nuovo thread che crea un collegamento in SMTP ad un account gmail dedicato (senderprogweb@gmail.com), il quale in seguito effettua l'invio dei messaggi. Abbiamo scelto di utilizzare un thread per l'invio in quanto la connessione al server gmail impiegava un po' di tempo, ritardando il caricamento della pagina per l'utente mentre in questo modo quest'attesa non si verifica.
- **Immagini profilo**: per quanto riguarda le immagini di profilo abbiamo deciso di salvare foto vere e proprie. Per realizzare ciò abbiamo implementato nel form relativo un campo per l'upload di un'immagine. Successivamente abbiamo implementato un algoritmo per rinominare in maniera univoca la foto appena caricata, questo nome univoco verrà salvato nel database nel record relativo all'utente che l'ha caricata. Le foto vengono salvate in una cartella apposita che viene indicata nel web.xml come parametro con un indirizzo assoluto.

Sicurezza

Durante la progettazione e l'implementazione del progetto abbiamo prestato attenzione anche alla sicurezza dove potevamo.

- **Autenticazione dell'utente e accesso alle aree protette**: Alla chiamata di login vengono verificate le credenziali dell'utente: se sono corrette, nella sessione di JavaEE viene conservato l'identificativo univoco dell'utente (non modificabile in alcun modo, nemmeno dall'amministratore del sito). A ogni richiesta a un'area protetta del sito, il ruolo dell'utente è verificato nel database.
- **Dati sensibili**: I soli dati sensibili che abbiamo individuato sono le password degli utenti. Le password sono salvate nel database crittografate utilizzando l'algoritmo md5.

Lavoro in Team

Per svolgere questo progetto abbiamo utilizzato git e github. Essendo il nostro un gruppo da 2 persone abbiamo utilizzato questo strumento principalmente come version control system. Le funzioni di condivisione e di branching sono state utilizzate poco in quanto abbiamo preferito incontrarci e lavorare assieme al progetto.