

# Sviluppo Condiviso

---

*Progetto Tecnologie Web 2014/2015*

Docente: Corazza Anna

Gruppo: Borrelli Angelo(N86/0791) – Panzuto Gianluca (N86/1099)

# **CAPITOLO 1: Proposta di Soluzione**

## ▪ ***Descrizione del problema***

L'applicazione prevede lo sviluppo condiviso, con successiva pubblicazione, di documenti. I documenti sono inizialmente condivisi solo tra gli editori fin quando non saranno poi pubblicati e quindi visibili a tutti. L'applicazione prevede uno storico delle varie versioni del documento precedenti alla pubblicazione visibile solo agli editori di quest'ultimo, con la possibilità di ripristinare una precedente versione. Ogni documento è diviso in varie parti in modo da poterne modificare solo una di esse; la modifica contemporanea di una stessa parte non è resa possibile (tra due editori o più) per evitare problemi di inconsistenza delle versioni. La successiva pubblicazione(dell'ultima versione) sarà possibile quando tutti gli editori avranno dato il loro consenso e per far ciò l'applicazione invierà una notifica di conferma a tutti gli editori.

## ▪ ***Tipi di utenti dell'applicazione ed operazioni che possono svolgere***

### Utente non Loggato:

- 1) Cercare un documento attraverso dei parametri di scelta oppure senza parametri (li visualizza tutti).
- 2) Registrarsi come editore ( Il controllore è già presente e non è possibile registrarsi come tale).
- 3) Effettuare il Login.

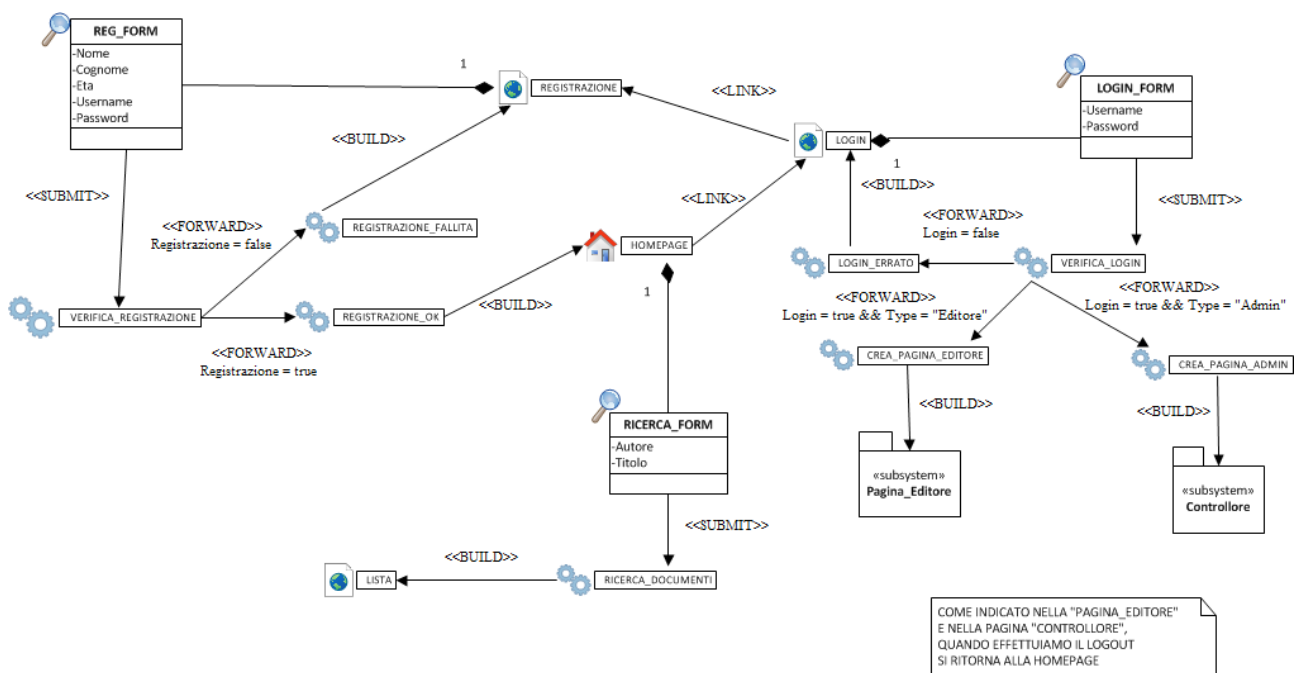
### Utente Loggato come Editore:

- 1) Creare un nuovo documento.
- 2) Modificare un documento esistente.
- 3) Inviare richieste di condivisione di un documento.
- 4) Inviare richieste di pubblicazione di un documento.
- 5) Accettare e/o Rifiutare notifiche di pubblicazione o condivisione.
- 6) Leggere messaggi di notifica di operazioni avvenute( documenti pubblicati, documenti non più pubblicati, accettazione di un invito da parte di un editore, etc..).
- 7) Visualizzare lo storico di un documento non ancora pubblicato ed eventualmente ristabilire una delle versioni precedenti.
- 8) Cercare documenti che esso ha scritto o che condivide.
- 9) Logout

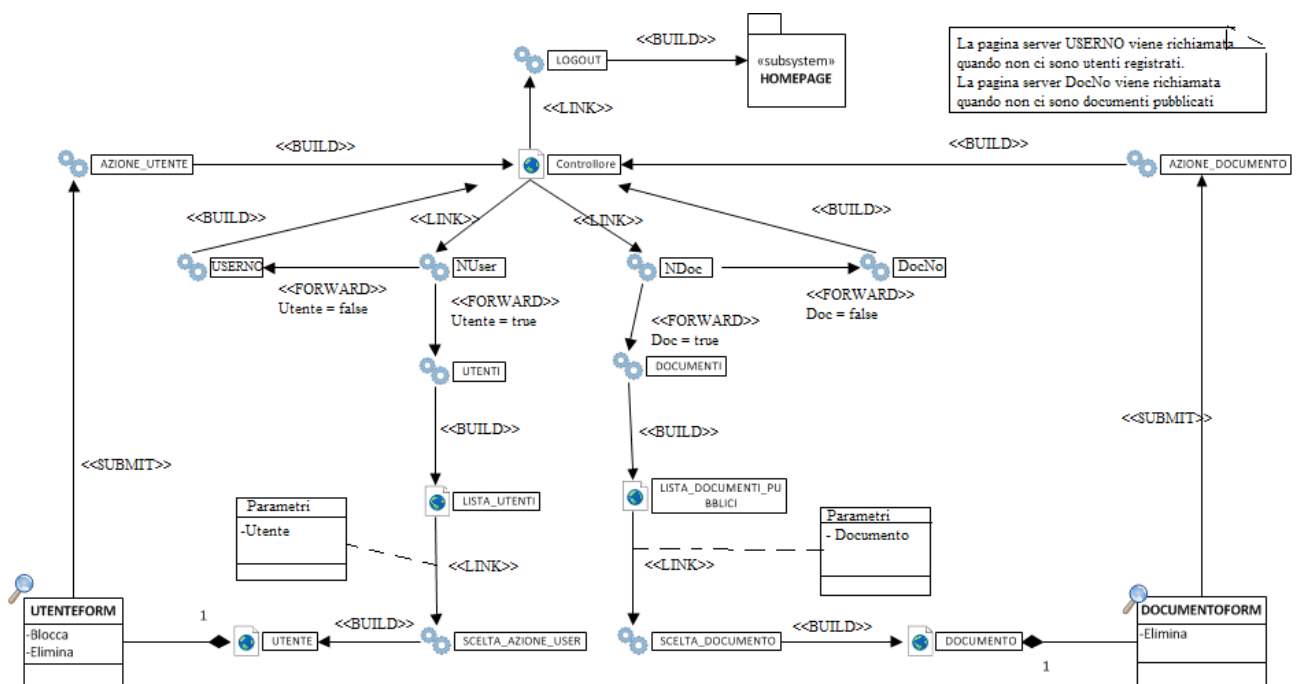
### Utente Loggato come Controllore:

- 1) Eliminare un editore.
- 2) Eliminare un documento pubblicato.
- 3) Bloccare un editore impedendogli il login per un fissato arco temporale.
- 4) Logout.

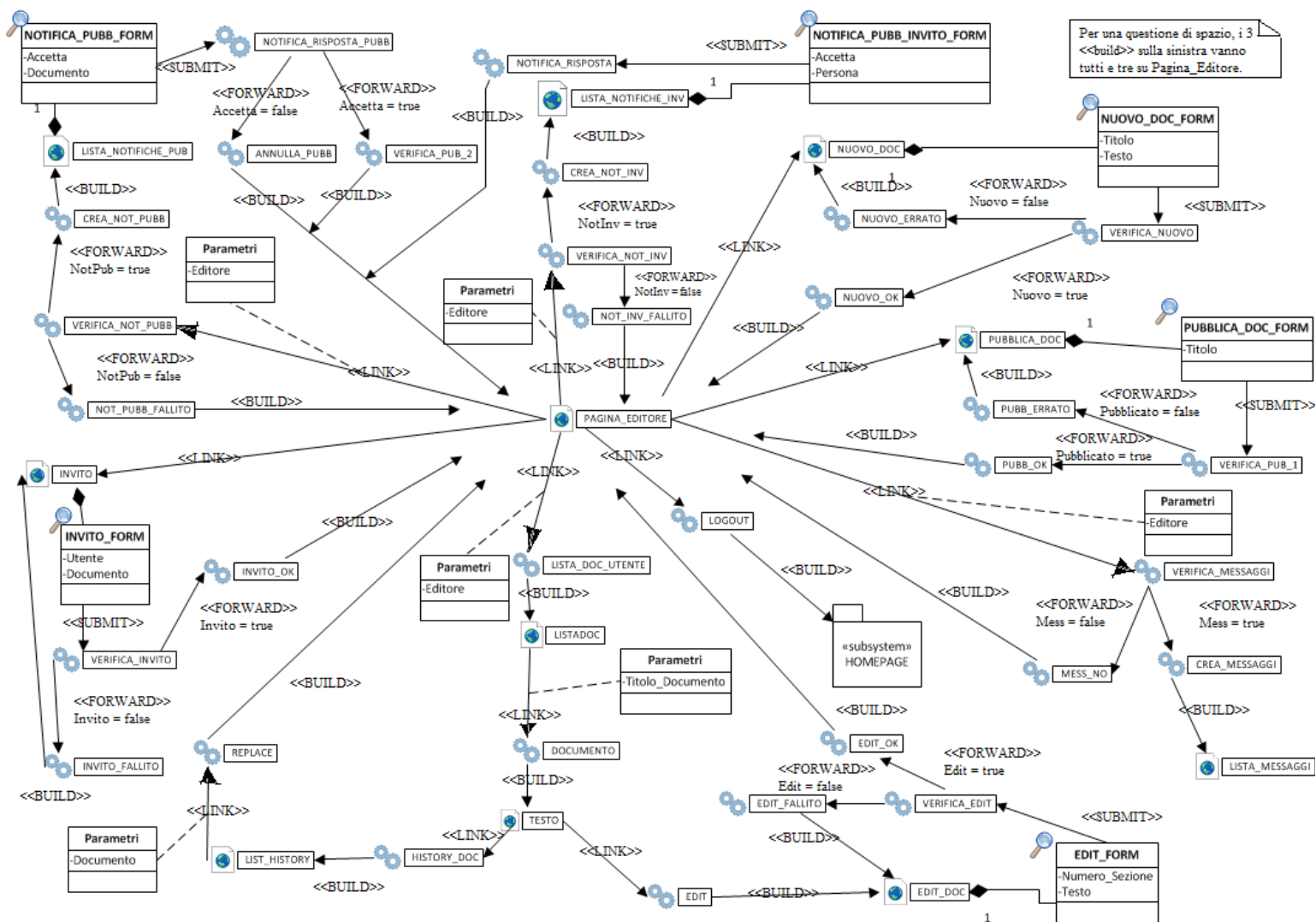
## ▪ Site Map: HomePage



## ▪ Site Map: Controllore



# Site Map Pagina\_Editore



## ▪ Alcune Specifiche della SiteMap

Una volta effettuato il Login si viene indirizzati alla pagina Editore. Analizziamo di seguito le azioni più salienti: Nel caso scegliamo di pubblicare un documento, saremo indirizzati ad una pagina client “Pubblica\_Doc” dove dovremo indicare il titolo del documento da pubblicare; la pagina server “Verifica\_Pub 1” che sarà richiamata dalla form controllerà se effettivamente esiste quel documento e se l’editore è uno degli editori effettivi del documento; nel caso in cui il documento non è condiviso da nessuno lo pubblica anche, altrimenti sarà inviata una notifica di pubblicazione a tutti gli editori che condividono il documento. Se l’operazione termina con insuccesso “Verifica\_Pubb” richiama la pagina server “Pubb\_Errato” che fa la <<build>> sulla pagina client “Pubblica\_Doc” per far ripetere l’operazione, altrimenti richiama la pagina server “Pubb\_Ok” che farà la <<build>> sulla pagina client “Pagina\_Editore”.

Nel caso in cui scegliamo di controllare le notifiche di pubblicazioni, la pagina server “Verifica\_Not\_Pubb” alla quale sarà passata il nome dell’editore, verificherà eventuali notifiche e in caso affermativo richiamerà una seconda pagina server “CREA\_NOT\_PUBB” che genererà una pagina client con la lista delle notifiche che indicano il Nome del documento e la possibilità di accettare o rifiutare. Una volta effettuata la decisione la form richiamerà una pagina server “NOTIFICA\_RISPOSTA\_PUBB” che in caso di risposta negativa richiamerà la pagina server “ANNULLA\_PUBB” che eliminerà le notifiche riguardo a quel documento e di conseguenza la pubblicazione; in caso affermativo richiamerà la pagina server “VERIFICA\_PUB\_2” che verificherà se il numero di consensi è uguale al numero di persone che condividono il documento allora lo pubblicherà, altrimenti aumenterà di uno i consensi per il documento. Sia la pagina server “Annulla\_Pubb” sia quella “Verifica\_Pub\_2” faranno una <<build>> sulla pagina client “PAGINA\_EDITORE”. Nel caso in cui scegliamo di ristabilire una versione precedente del documento, arriveremo al documento stampato Pagina client “TESTO” dove ci sarà un link apposito; sarà così richiamata la pagina server “HISTORY\_DOC” che costruisce la pagina client, “LIST\_HISTORY” che sarebbe lo storico del documento, con annessi link che in caso di click riporterebbero il documento corrente ( ricordiamo che questo si trova in fase di editing ) allo stato precedente scelto, operazione effettuata dalla pagina server “REPLACE” che successivamente farà una <<build>> della “PAGINA\_EDITORE”.

La pagina Server “LISTA\_DOC\_UTENTE” costruirà una pagina “LISTADOC” con tutti i documenti editati fino a quel momento dell’editore; eventualmente sarà vuota nel caso in cui non ha ancora partecipato alla condivisione di qualche documento o scritto documenti.

La pagina server “VERIFICA\_EDIT” inoltrerà la modifica aggiornando la sezione scelta dalla form precedente ,a patto che solo quella sezione non è già in fase di modifica da un altro editore. In Caso di errore richiama la pagina server “Edit\_Fallito” che fa la <<build>> di “Edit\_Doc” altrimenti richiama la pagina server “Edit\_Ok” che farà la <<build>> della pagina client “PAGINA\_EDITORE”.

La pagina server “VERIFICA\_NUOVO” verifica se esiste già un documento con quel titolo ; in caso positivo richiama una pagina server “Nuovo\_Ok” che effettua la registrazione del documento come documento in fase di modifica altrimenti richiama una pagina server “Nuovo\_Errato” che farà una <<build>> della pagina client “NUOVO\_DOC” per ripetere l’operazione. I documenti in fase di modifica sono quelli che memorizziamo nella tabella “ElementiPubblicati” del database e fanno sempre riferimento alla versione più aggiornata del documento.

## ▪ **Tecnologie Utilizzate**

Le tecnologie per la realizzazione sono state prevalentemente **HTML**, **JSP**, **Servlet** . Le pagine in formato HTML sono state scelte per visualizzare il contenuto delle risposte del server al client dopo una ricerca sul database, per la HomePage o per eventuali notifiche. L'utilizzo della tecnologia client **Javascript** è utilizzata per i controlli dei dati da passare in input al server.

Lato Server abbiamo preferito la tecnologia Java (JSP e Servlet) e non PHP, per la maggior efficienza che offre per l'interrogazione alle basi di dati, essendo l'operazione principale su cui si basa l'applicazione, per le continue pagine HTML generate che avrebbero rallentato l'esecuzione in PHP e per la migliore gestione delle eccezioni in fase di runtime (tra cui quelle di caduta di connessione con il Database , errate Query , etc..).

Infine un **Database** sarà necessario per memorizzare i dati relativi agli utenti e ai documenti , potendo così effettuare ricerche e controlli ai fini delle operazioni dell'applicativo.



## ▪ Database

Il Database che utilizzeremo sarà composto da 6 tabelle.

### 1) Tabella “Utente”

id\_user: number  
nome: varchar(20)  
cognome: varchar(20)  
password: varchar(50)  
tipo: varchar(10)  
Bloccato: Boolean

### 2) Tabella “ElementiDaPubblicare”

id\_doc: number  
titolo: varchar(100)  
parte1: varchar(2000)  
parte2: varchar(2000)  
parte3: varchar(2000)  
parte4: varchar(2000)  
parte5: varchar(2000)  
utente\_scrive: number  
utente2: number  
utente3: number  
utente4: number  
utente5: number  
occupato1: Boolean  
occupato2: Boolean  
occupato3: Boolean  
occupato4: Boolean  
occupato5: Boolean

### 3) Tabella “Storico”

titolo: varchar(100)  
data: date  
ora: time  
id\_doc: number  
utente\_modifica: number

parte1: varchar(2000)  
parte2: varchar(2000)  
parte3: varchar(2000)  
parte4: varchar(2000)  
parte5: varchar(2000)

#### 4) Tabella “DocumentiPubblicati”

id\_doc: number  
titolo: varchar  
data: date  
id\_editore: number

#### 5) Tabella “Notifiche”

ID \_Doc: Number  
utente\_mitt: Number  
utente\_dest2: Number  
utente\_dest3: Number  
utente\_dest4: Number  
utente\_dest5: Number  
tipo: varchar(20)  
risposta2: Number  
risposta3: Number  
risposta4: Number  
risposta5: Number

#### 6) Tabella “Messaggi”

utente\_dest: Number  
Testo: varchar(200)

# Capitolo 2: Relazione di fine progetto

## ▪ Modifiche alla proposta di soluzione

- 1) Eliminazione della tabella “ParteInModifica”. Essa la utilizzavamo per memorizzare in maniera temporanea la parte di un determinato documento che era in fase di modifica; così se un utente voleva effettuare una modifica su una determinata parte di documento e questa parte era presente nella tabella, allora non poteva modificare nulla. Però questa soluzione non andava bene se chiudevamo il browser o tornavamo indietro. Pertanto abbiamo scelto di utilizzare una variabile di sessione che memorizza la parte in modifica, in modo che se il browser viene chiuso o si torna indietro nella pagina, si può comunque modificare quella parte; utilizziamo la variabile di sessione come un semaforo mutex che permette ad un solo utente alla volta di effettuare un’operazione atomica (in questo caso la modifica di una parte del documento);
- 2) Eliminazione di campi ridondanti in alcune tabelle del Database come i vari ID e le variabili booleane “occupato”;
- 3) Aggiunta di JavaScript per i controlli di inserimento di caratteri accentati nel Titolo del documento, nell’username e nella password (le varie tabelle del database si trovano nella cartella “Dump20150123”);
- 4) Aggiunta di qualche JSP di verifica. Ad alcune JSP abbiamo preferito un’unica pagina JSP di risposta che contiene una variabile di sessione che memorizza il messaggio di risposta di un’azione effettuata da un editore.

Esempio: se un utente accetta o rifiuta una pubblicazione di un documento, quattro sono i casi: se un utente accetta e tutti gli altri utenti hanno accettato, allora il documento viene pubblicato; se un utente accetta e non tutti hanno ancora risposto, allora non viene pubblicato il documento ma si notifica all'utente della riuscita dell'operazione; se un utente ha rifiutato la pubblicazione, allora il documento non viene pubblicato (come indicato nel proposal, la pubblicazione di un documento condiviso da più utenti deve avvenire in maniera unanime: se almeno uno non accetta, allora il documento non verrà pubblicato).

5) In tutte le JSP abbiamo aggiunto queste due istruzioni:

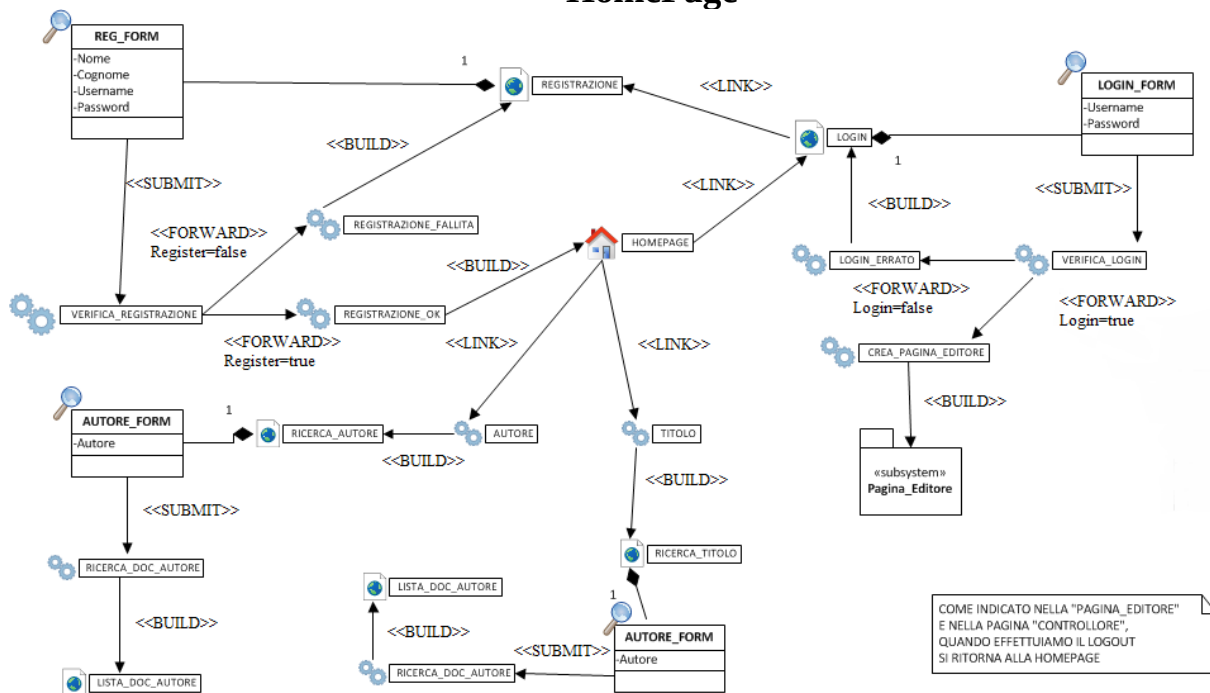
```
<%@ page errorPage="ERRORE.jsp" %>  
<%@ page isThreadSafe="true" %>
```

errorPage è un attributo della direttiva page che permette di specificare una pagina di errore da mostrare in caso avvenga un'eccezione non gestita.

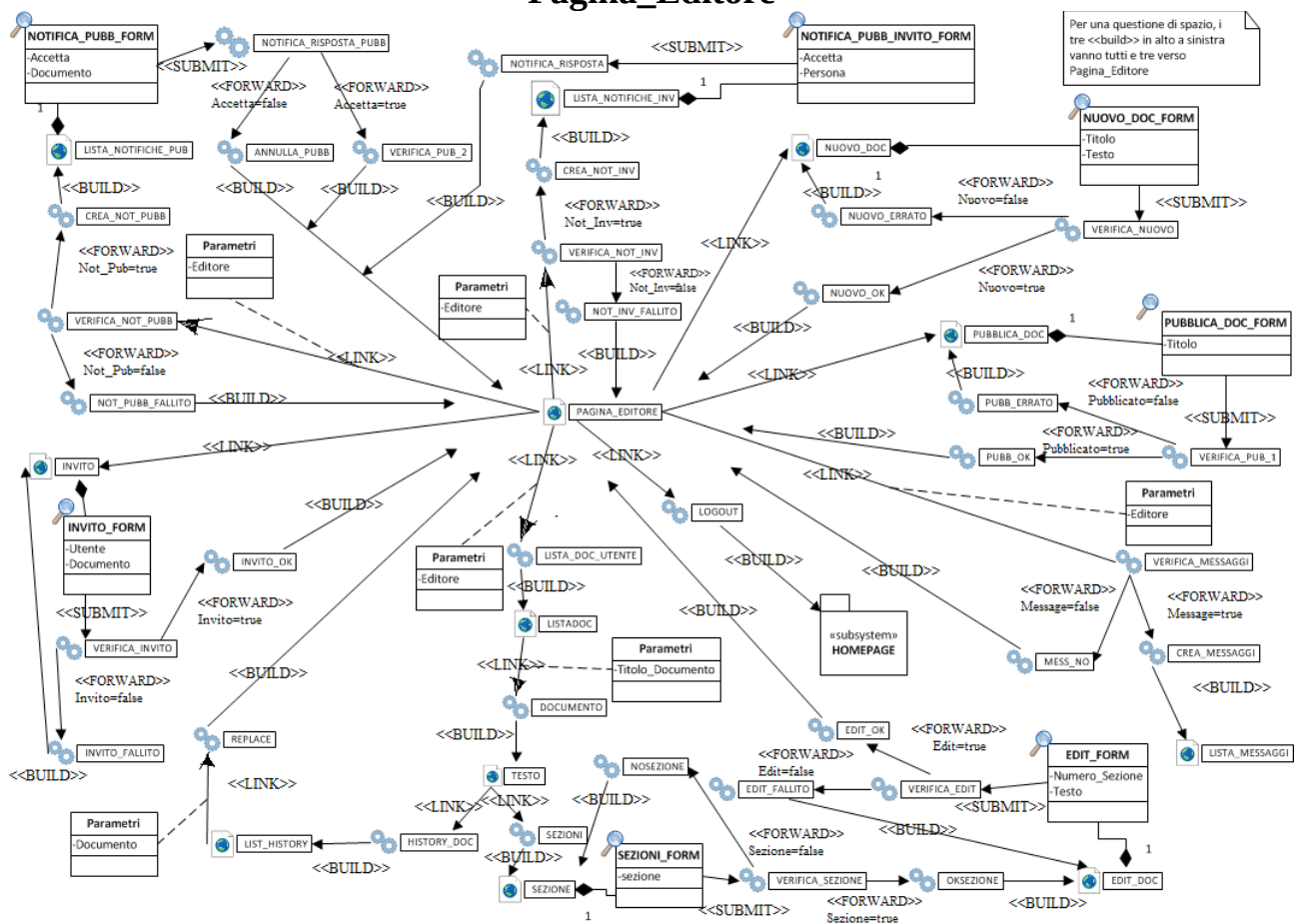
isThreadSafe è usato per indicare se una volta convertita in una Servlet la pagina JSP è in grado di supportare più richieste contemporaneamente: se settato a false il container jsp accoderà le richieste e le elaborerà singolarmente.

## ▪ Modifiche Site Map

## HomePage



## Pagina\_Editore



## ▪ **Funzionalità da implementare**

Nell'incontro con il docente, è stato deciso di implementare solo la parte della HomePage e della Pagina\_Editore.

## ▪ **Architettura del sistema**

L'applicazione e' stata sviluppata con l'Ide NETBEANS 8.0.2 , testata su Mac OSX e Ubuntu, su browser Safari, Google Chrome e Firefox. L'architettura scelta parte dalla considerazione del linguaggio ad oggetti Java che ha spinto all'utilizzo della tecnologia JEE per lo sviluppo di applicazioni web.

JEE (Java Enterprise Edition) rappresenta una base solida per lo sviluppo di un'applicazione ed è stata la scelta fatta per il nostro progetto.

I principali vantaggi derivanti dall'uso delle specifiche JEE per lo sviluppo di applicazioni web sono fondamentalmente quattro:

- **Scalabilità:** è possibile aumentare le funzionalità dell'applicazione software, grazie anche alla peculiare proprietà di distribuibilità della tecnologia Java.
- **Portabilità:** esigenza fondamentale dell'attività di sviluppo software, permette di utilizzare una stessa applicazione JEE in Application Server diversi purché questi implementino le specifiche JEE.
- **Efficienza:** una strutturazione così intensa facilita la gestione di progetti anche molto complessi. Inoltre, grazie all'utilizzo del multi-threading di Java è possibile ottenere elevate performance per l'inte-

razione tra Client e Server

- Sicurezza: assicurata dall'elevata stratificazione dell'architettura e dalle proprietà intrinseche al linguaggio Java.

In quanto applicazione JEE, essa è una applicazione Multitier, nello specifico possiede i seguenti elementi:

- il *Client Tier*, il quale presenta all'utente finale i risultati dell'elaborazione del server, nel nostro caso il browser.
- il *Web Tier*, comprende una serie di componenti che riguardano il lato front-end dell'applicazione, mediando richieste e risposte del protocollo HTTP tra client e server; Nel nostro caso in quanto siamo in presenza di una semplice applicazione web funge anche da interfaccia con il DBMS.

Infine abbiamo il Dmbs per l'accesso ai dati, che nel nostro caso è stato MYSQL.

Il Server utilizzato, necessario per la comunicazione tra le varie componenti dell'applicazione, è stato *Sun GlassFish Enterprise Server*, che essendo gestito direttamente da Sun implementa nel modo più completo e fedele le specifiche dello standard JEE. Non c'è stato un uso specifico di un pattern di sviluppo, dove si è preferita una logica bivalente di elaborazione richieste ed accesso dei dati, per il Server, e fruizione dei contenuti e invio di richieste per il client.

Del Client Tier figurano le tecnologie Client come Html, Javascript e CSS, mentre per il Web-Tier figurano le Servlet, Jsp e JDBC.

## ▪ **Discussione e motivazioni delle tecnologie**

HTML è la scelta fatta per le pagine client, con i fogli di stile CSS che ne caratterizzano il Layout.

Le Servlet hanno il compito prevalente di accesso ai dati e di elaborazione delle richieste;

Le JSP hanno il compito di fornire al client le risposte delle richieste che aveva inviato, costruendo le pagine Html che saranno necessarie per l'interazione con esso.

JDBC sono l'insieme di Api necessarie per l'accesso al database Relazionale scelto, che nel nostro caso è MYSQL.

JavaScript utilizzato per i controlli dei dati inseriti dall'utente, evitando quindi errori lato Server che avrebbero rallentato l'esecuzione complessiva. Javascript dopo aver verificato che i dati immessi sono corretti, grazie all'uso delle Espressioni Regolari o a semplici controlli di campi vuoti, accede grazie alla Dom alle varie Form sulle quali effettua i controlli, abbinandogli la action corrispondente alla pagina server richiesta.



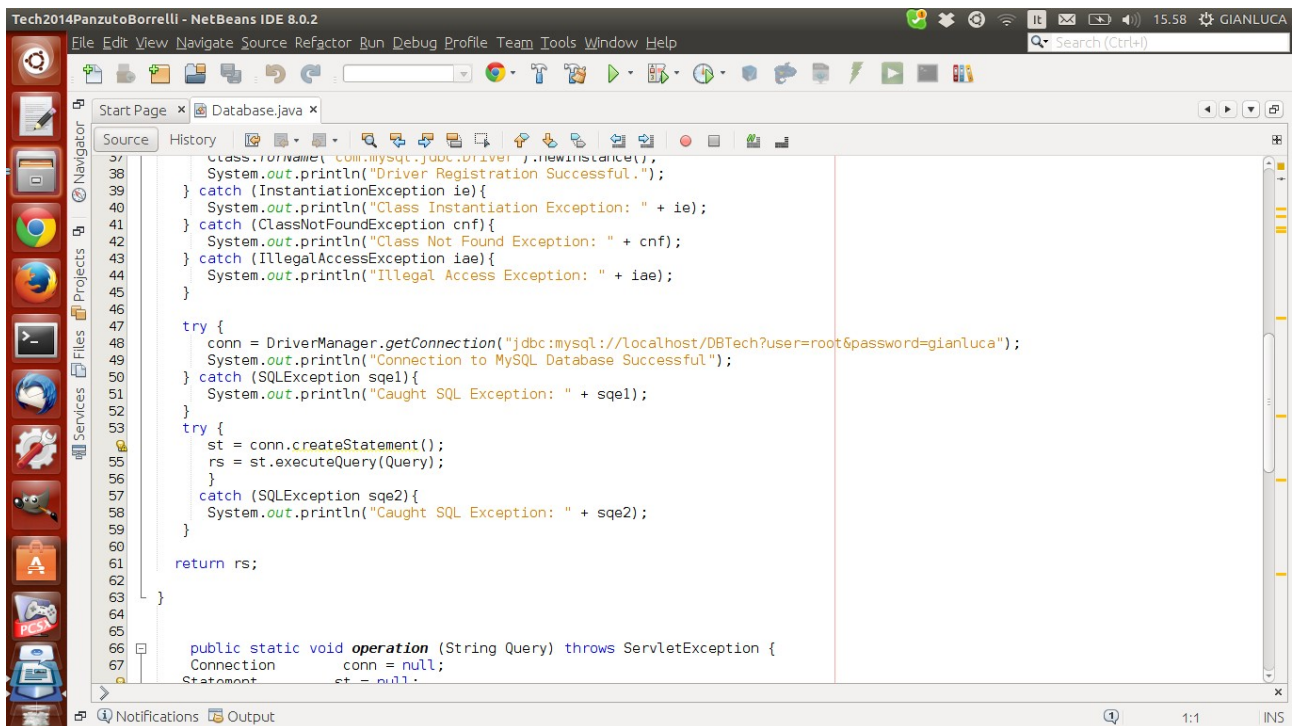
## ▪ Tecniche di session tracking utilizzate

Le tecniche utilizzate sono state le variabili di sessione e i campi Hidden. Con le variabili di sessione si sono memorizzate informazioni prevalentemente sullo username dell'utente che accede, necessario sia per un controllo di sicurezza ad operazioni permesse solo a chi fa il Login, sia per le funzionalità di richieste al database. Altre variabili di sessione che memorizzano informazioni per l'elaborazioni di richieste saranno quelle di modifica, per evitare collisioni di modifica contemporanea sulla stessa parte del documento, titolo del documento, messaggi di errore.

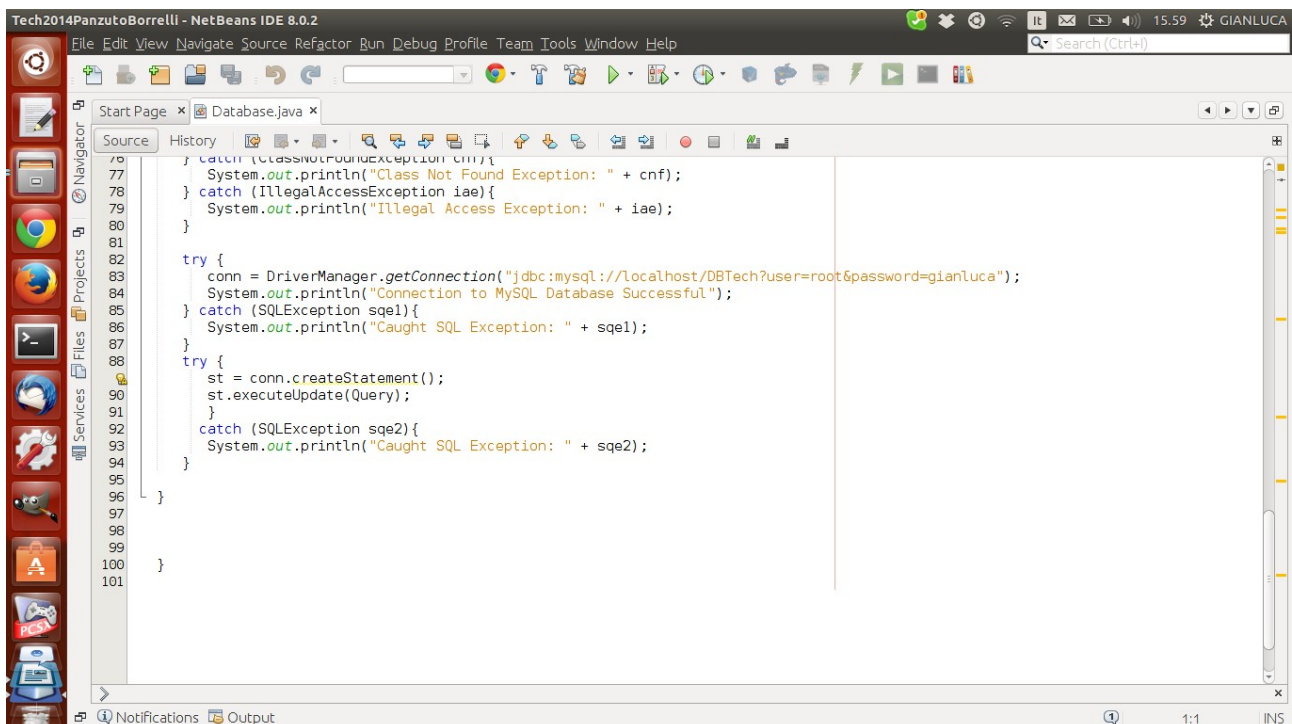
I campi hidden invece sono stati necessari nei pochi casi nei quali le form avevano l'opzione di fare accettare o rifiutare una azione, con un radio nelle form, per cui per inviare al server quale documento aveva avuto una risposta abbiamo scelto di non farlo selezionare dall'utente, poiché quest'ultimo lo legge in un avviso sopra l'opzione di scelta.

## ▪ Discussione Eventuali ipotesi aggiuntive

1) Prima di effettuare una prova, occorre cambiare il nome del DBMS, del database, dell' Username e della password nella classe “Database.java”. La modifica va eseguita nei seguenti punti:



```
37 import java.sql.*;
38
39 class.forName("com.mysql.jdbc.Driver").newInstance();
40 System.out.println("Driver Registration Successful.");
41 } catch (InstantiationException ie) {
42     System.out.println("Class Instantiation Exception: " + ie);
43 } catch (ClassNotFoundException cnf) {
44     System.out.println("Class Not Found Exception: " + cnf);
45 } catch (IllegalAccessException iae) {
46     System.out.println("Illegal Access Exception: " + iae);
47 }
48
49 try {
50     conn = DriverManager.getConnection("jdbc:mysql://localhost/DBTech?user=root&password=gianluca");
51     System.out.println("Connection to MySQL Database Successful");
52 } catch (SQLException sqe1) {
53     System.out.println("Caught SQL Exception: " + sqe1);
54 }
55
56 try {
57     st = conn.createStatement();
58     rs = st.executeQuery(Query);
59 } catch (SQLException sqe2) {
60     System.out.println("Caught SQL Exception: " + sqe2);
61 }
62
63 return rs;
64 }
65
66 public static void operation (String Query) throws ServletException {
67     Connection conn = null;
68     Statement st = null;
```



```
70 } catch (ClassNotFoundException cnf) {
71     System.out.println("Class Not Found Exception: " + cnf);
72 } catch (IllegalAccessException iae) {
73     System.out.println("Illegal Access Exception: " + iae);
74 }
75
76 try {
77     conn = DriverManager.getConnection("jdbc:mysql://localhost/DBTech?user=root&password=gianluca");
78     System.out.println("Connection to MySQL Database Successful");
79 } catch (SQLException sqe1) {
80     System.out.println("Caught SQL Exception: " + sqe1);
81 }
82
83 try {
84     st = conn.createStatement();
85     rs = st.executeQuery(Query);
86 } catch (SQLException sqe2) {
87     System.out.println("Caught SQL Exception: " + sqe2);
88 }
89
90 return rs;
91 }
92
93 public static void operation (String Query) throws ServletException {
94     Connection conn = null;
95     Statement st = null;
```

Modificare nella riga 48 e nella riga 83 nel punto:

`"jdbc:nome_dbms://localhost/Nome_Db?user=nuovo_utente&password=nuova_password"`

2) Abbiamo scelto che il titolo del documento, l'username e la password non possono contenere caratteri accentati. La scelta è motivata dalla difficoltà di ricerca nel database di questi caratteri.

3) Per effettuare una modifica ad un documento, si va in “Visualizza Documento” (Nota bene che fa riferimento a documenti in fase di pubblicazione e non quelli già pubblicati): si sceglie il documento da visualizzare e, una volta visualizzato il testo, puoi visualizzare lo storico o modificare una parte di documento.