**Introduzione:**

In questo secondo progetto abbiamo dovuto realizzare una piattaforma web che consenta di discutere e condividere contenuti all’interno di gruppi di discussione creabili a piacimento dagli utenti del sistema. Il sistema deve inoltre permettere la visualizzazione dei gruppi “aperti” anche agli utenti che non fanno parte del sistema, e fornire un sistema basilare di moderazione.

**Database:**

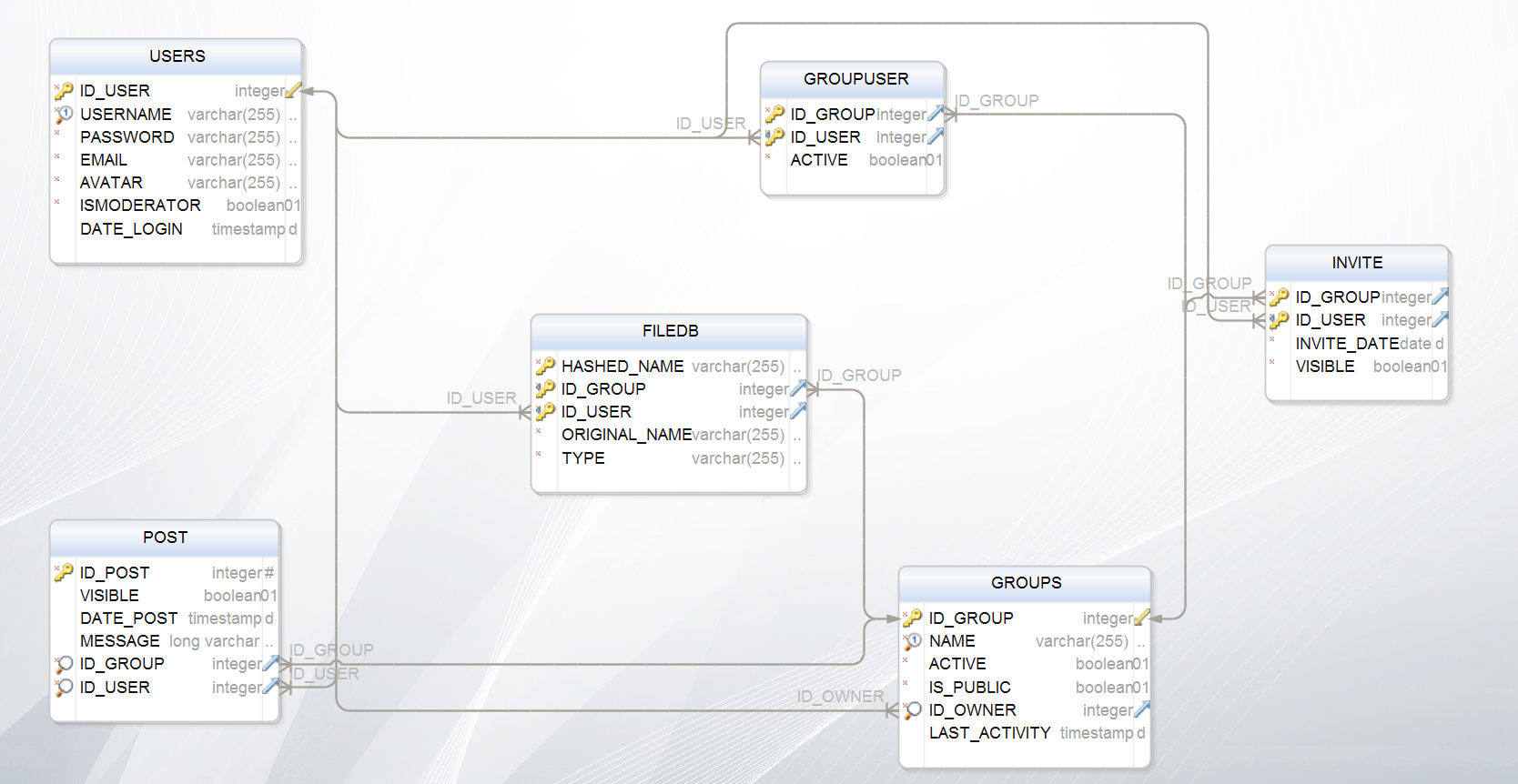
****

Diagramma ER del database

Il database di questo progetto è essenzialmente un adattamento del databasse utilizzato per il primo progetto, con campi aggiuntivi per permettere la chiusura di un gruppo e differenziare gli utenti moderatori dagli utenti normali.

Un campo last\_activity di tipo timestamp è stato aggiunto alla tabella groups e viene aggiornato ad ogni inserimento di un nuovo post per tale gruppo, permettendo una ricerca più facile delle ‘Novità dal tuo ultimo login’ quando si effettua l’accesso.

**Tecnologie:**

La parte applicativa del sito è stata realizzata in linguaggio Java, facendo uso di servlet e pagine JSP: le prime implementano le funzioni necessarie alla logica applicativa ed al trattamento dei dati, mentre le pagine JSP si occupano di generare il codice HTML visualizzato dall’utente.

Il DBMS scelto per il progetto è stato DerbyDB, poiché viene fornito assieme all’ambiente di sviluppo NetBeans e l’entità del progetto non richiedeva database particolarmente performanti o con funzionalità avanzate.

La piattaforma prevede inoltre l’utilizzo di due Filter per regolare l’accesso ai contenuti:

* Il controllo di avvenuto login all’interno delle pagine è effettuato tramite un’apposita Filter che si occupa di verificare la presenza dei dati utente nella sessione browser.
* L’accesso da parte di un utente ad un gruppo, e quindi la visualizzazione di post e il download di file caricati nel gruppo e l’eventuale accesso in modalità anonima sono controllati da un’altra Filter dedicata.

Oltre alla suddetta filter, un ulteriore controllo sui permessi di visualizzazione viene effettuato dalle pagine JSP stesse mediante l’esecuzione di codice JSTL

**Struttura:**

L’applicazione è stata realizzata seguendo in gran parte il pattern MVC con frontcontroller.

Tutte le richieste vengono infatti processate da un’apposita servlet di controllo, chiamata CyberController, la quale si aspetta un parametro passato via GET che indichi il tipo di operazione che si desidera effettuare.

All’interno del controller viene quindi letto il parametro e, in base al suo valore, la richiesta http viene inoltrata alla servlet corrispondente in caso l’operazione richiesta trovi riscontro nella lista di operazioni disponibili, altrimenti si verrà reindirizzati ad una pagina di errore.

Nel caso la richiesta sia stata inoltrata con successo, la servlet ricevente effettua i dovuti controlli e le operazioni richieste, inserisce i risultati nella sessione e reindirizza l’utente verso la pagina JSP che si occupa della visualizzazione dei dati

**Gestione dei File:**

Quando un utente carica un file, questo viene posizionato in una sottocartella dipendente dall’utilizzo del file.

*File caricati all’interno di un gruppo*

Per ogni gruppo esiste una omonima sottocartella dedicata, all’interno della directory ‘uploads’ in cui gli utenti iscritti possono caricare file. I file hanno una dimensione massima di ~10 Mb, sono inoltre univoci perché sono salvati come hash di: nome\_file, nome\_uploader, nome\_gruppo. Tale combinazione fornisce una buona probabilità che non ci siano file con nomi identici, e comunque nel caso viene notificato all’utente che il file con tale nome esiste già.

Una volta caricato il file, i dettagli relativi vengono salvati nel database tenendo traccia anche del nome originale.

Per accedere a un file è necessario passare tramite una servlet apposita che agisce da proxy tra l’utente e il file system, cosi da rendere più sicura e controllata l’interazione tra le parti.

*Immagini del profilo utente (avatar)*

Tutti gli avatar sono contenuti all’interno della directory ‘Avatars’, collocata nella cartella radice dell’applicazione, e sono salvati come hash di nome\_uploader e nome\_file, per evitare (o più precisamente ridurre notevolmente) la possibilità che un utente possa sovrascrivere l’immagine profilo di un altro utente. I file conservano l’estensione e sono accessibili direttamente da url, in modo da permetterne la visualizzazione anche da parte di visualizzatori anonimi non facenti parte del sistema.

**Parsing del testo:**

Per implementare il parsing engine (motore di traduzione) abbiamo usato delle espressioni regolari per identificare tutte le stringhe che fossero nel formato $$string$$. L’espressione regolare usata in questo caso è “\$\$(S+)\$\$”, cioè tutto quello che è tra $$ e $$. Una volta trovate le corrispondenze (in questo caso quelle senza i delimitatori $$) vengono controllate a loro volta per identificarne il tipo e possono essere:

* Link ipertestuali – stringhe che hanno una corrispondenza con l’espressione regolare "((mailto\\:|(news|(ht|f)tp(s?))\\://){1}\\S+)". Si tratta dei protocolli più usati e raggruppa tutti i link che iniziano con ftp, http, https, mailto etc e tutto quello che viene dopo. E’ da notare che www.sito.com non viene riconosciuto.
* Link a file del gruppo – in questo caso viene generato un link che chiama la servlet di download fornendo l’hash del file.
* Stringa casuale – in questo caso viene riportato il testo come è stato fornito.

**Gestione degli inviti:**

Agli utenti è fornita la possibilità di accettare o rifiutare inviti all’iscrizione a gruppi di discussione. L’utente viene presentato con una pagina che visualizza una lista selezionabile di inviti in attesa di risposta e due pulsanti, uno per eliminare gli inviti selezionati e uno per accettarli.

In caso di rifiuto di un invito, questo viene effettivamente rimosso dal database, permettendo al mittente di inviare nuovamente un invito ad un utente. In caso un invito venga accettato, il suo campi VISIBLE viene impostato ad un valore negativo e viene quindi ignorato in fase di visualizzazione

**Recupero password :**

Nella pagina di login è aggiunto un link alla pagina di recupero password. Fornendo il proprio nome utente, il sistema genera una nuova password tramite un algoritmo pseudocasuale ed invia la nuova password all’utente tramite email. L’utente può in qualsiasi momento cambiare la propria password dal menù delle impostazioni utente.

L’invio delle email è gestito utilizzando gli oggetti messi a disposizione dal framework Java, per l’esattezza delle librerie presenti nel namespace javax.mail, e si appoggia ad un server smtp esterno (nella fattispecie al server smtp di google) per inviare i messaggi.

**Login e accesso anonimo:**

Le informazioni relative all’avvenuto login vengono gestite tramite una variabile di sessione che identifica l’utente connesso. Al momento dell’accesso, una variabile contenente i dettagli dell’utente che si è collegato (username, data ultimo login, tipologia utente, indirizzo email) viene inserita nella sessione del browser. In questo modo si può avere facilmente accesso a dati utente che vengono usati spesso minimizzando le richieste alla base dati.

L’accesso anonimo ai gruppi pubblici è implementato tramite una pagina accessibile senza effettuare il login, nulla quale viene visualizzata una lista dei gruppi con valore IS\_PUBLIC impostato a TRUE. Seguendo uno dei link relativi ad un gruppo, viene associato uno specifico utente “Anonimous” non presente nel database alla sessione corrente, che permette di visualizzare correttamente i post dei gruppi pubblici ma impedisce di aggiungervi contenuto