**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:**

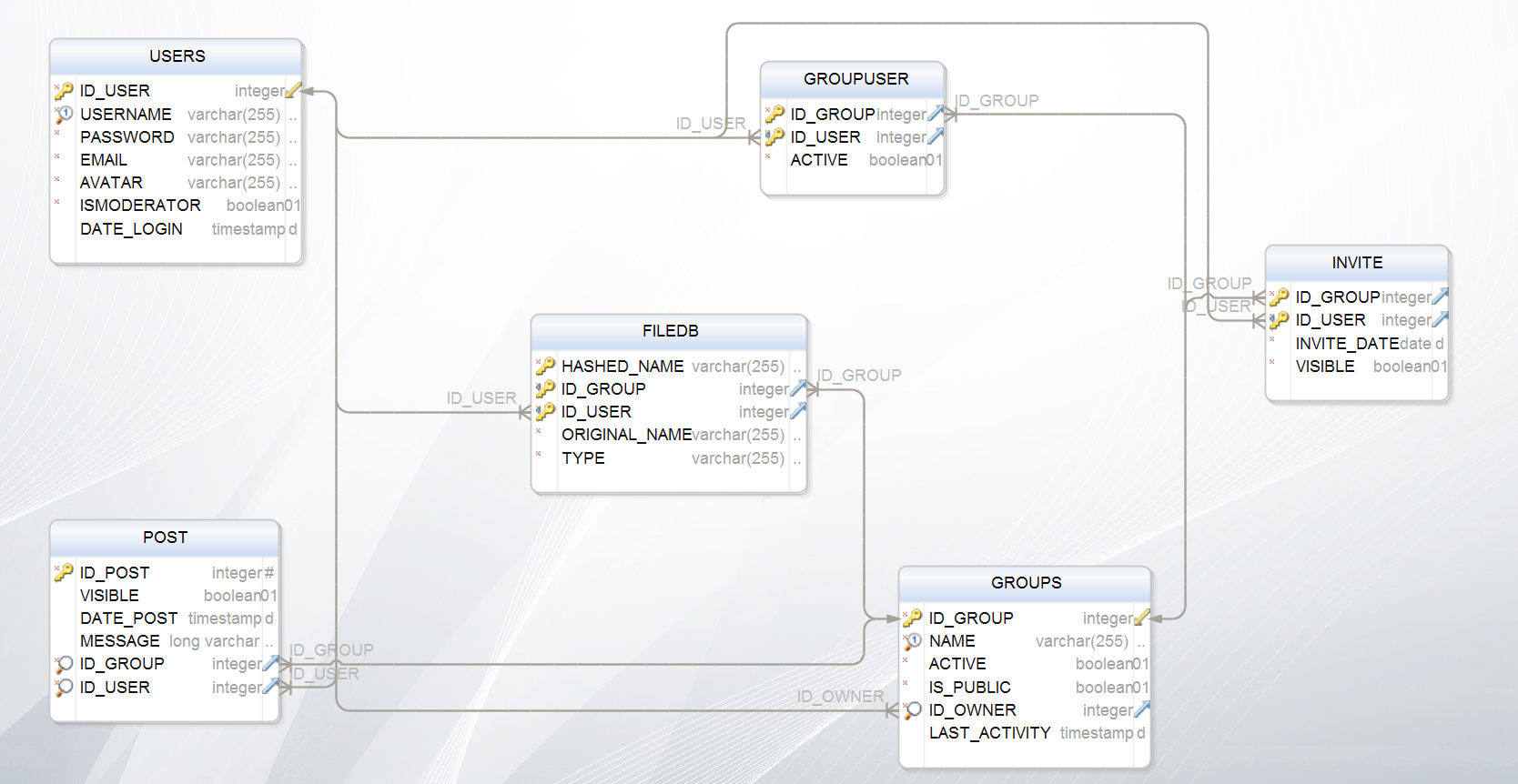
****

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:**

[PARLARE DI JAVA SERVER PAGES, TAZZAFAGIOLI E ACCENNARE ALLE SERVLET]

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:**

[PARLARE DI MVC E DEL CYBERCONTROLLER QUI]

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:**

[SCELTE FATTE PER AVATAR E FILE E PERCHÉ]

I file sono tutti contenuti in una cartella uploads e ogni gruppo ha una cartella omonima dedicata, 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 probabillità 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.

**Parsing del testo:**

[QUESTO NON SEMBRA DEBBA CAMBIARE, POTREMMO TENERLO PARI PARI]

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: [ESPANDERE CON RECUPERO PASSWORD]**

[PARLARE DI COME ABBIAMO IMPLEMENTATO IL SERVIZIO EMAIL]

Agli utenti è fornita la possibilità di accettare o rifiutare inviti all’iscrizione a gruppi di discussione. Abbiamo pensato di implementare in modo differente il fatto che un utente possa accettare più gruppi in una sola volta mentre per declinare l’invito è necessario farlo singolarmente. Si parte dal presupposto che capita spesso di sbagliare, e abbiamo voluto porre maggiore attenzione sul fatto che un utente voglia esser sicuro di rifiutare un invito facendolo singolarmente. Inoltre, abbiamo tenuto conto che un utente possa anche erroneamente cancellare un invito, quindi il sistema concede al mittente di inviare più volta un invito (se cancellato).