**Introduzione:**

Nel corso di questo progetto è stata realizzata un’applicazione web tramite la quale più utenti possono creare dei gruppi di discussione dove poter aggiungere post testuali e caricare file.

Durante l’implementazione del progetto ci siamo imbattuti in diversi problemi da gestire, principalmente legati alla gestione delle autorizzazioni e dei permessi di visualizzazione.

**Database:**

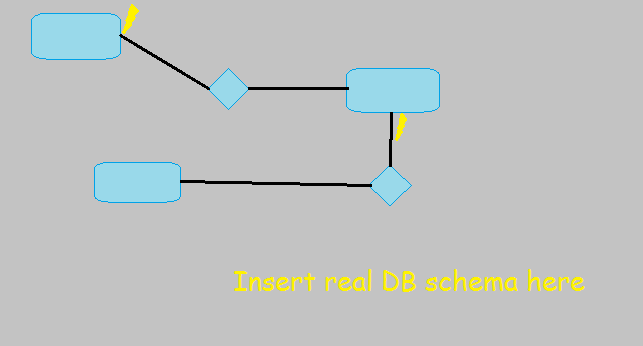
****

Diagramma ER del database

Il database è stato progettato per minimizzare la quantità di colonne NULL in ogni tabella, motivo per il quale è stata creata la tabella FILEDB: poiché l’utilizzo principale di un servizio di questo tipo è quello di postare messaggi, l’inclusione delle informazioni relative ad un file nella tabella POST avrebbe portato ad avere numerose entry nulle.

La presenza dei campi visible permetterà, in caso di cambiamento delle richieste, di implementare la cancellazione di un gruppo o di un post (moderazione), senza effettivamente eliminare dati dal database e mantenendo quindi integri i dati per le funzioni di conteggio post e di storico delle conversazioni

[SOSTITUIRE FUFFA GIÀ SCRITTA NEL PRIMO PROGETTO CON FUFFA SUI TIMESTAMP E SCELTE PER GLI AVATAR]

**Tecnologie:**

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

La parte applicativa del sito è stata realizzata in linguaggio Java, più precisamente tramite l’implementazione di servlet che si occupano di processare le richieste e 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 diverse 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 o di un cookie di sessione o della 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 sono controllati da un’altra Filter dedicata.
* L’accesso al menù di amministrazione di un gruppo è garantito al solo amministratore del gruppo stesso tramite un’altra Filter.

**Struttura:**

[PARLARE DI MVC E DEL CYBERCONTROLLER QUI]

La struttura dell’applicazione è suddivisa in tre parti, che coincidono a grandi linee con le restrizioni eseguite dalle Filter presenti; si divide principalmente in:

* /user/ le servlet che rispondono a questa sezione sono quelle che si occupano della visualizzazione dati relativi all’utente stesso.
* /group/ le servlet che rispondono a questa sezione sono quelle che si occupano della visualizzazione dei post di un gruppo e dell’aggiunta di nuovi post, e che richiedono quindi il controllo di appartenenza al gruppo che si sta visualizzando.
* /admin/ le servlet che rispondono a questa sezione sono quelle che forniscono gli strumenti di amministrazione e creazione di gruppi, e che richiedono quindi la verifica di proprietà di un gruppo prima di permetterne la modifica.

Tutte le servlet sono comunque controllate da un filtro che controlla se l’utente è loggato o altrimenti lo redireziona alla pagina di Login.

**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).