COPERTINA PRESENTAZIONE

Indice

Introduzione

Strumenti utilizzati e linguaggi scelti

L'obiettivo del progetto è implementare un web service che simuli una libreria distribuita. Per fare questo abbiamo utilizzato tre strumenti:

- Apache tomcat (7.0.29);
- Apache axis2 (1.6.2);
- Apache httpd (2.4.1);

Apache tomcat è un servelet container per pagina JSP e fornisce una piattaforma in grado di supportare l'esecuzione di applicazioni sviluppate con il linguaggio Java.
Apache axis2, invece, è l'engine del Web Services e mette a disposizione una struttura per creare, pubblicare e consumare Web service, sia in Java sia in C. Permette, inoltre, lo scambio di messaggi SOAP e l'utilizzo del WSDL.
Apache httpd è un http Web server e realizza le funzioni di trasporto delle informazioni e di collegamento.
Abbiamo scelto di utilizzare il linguaggio Java per la realizzazione del server, mentre PHP e HTML per realizzare la parte client.
Il vantaggio principale che deriva da queste scelte è che l'utente può usufruire e consumare il web service semplicemente attraverso un broswer. Non ha bisogno quindi di applicazioni particolari o software specifici, ma soltanto di un broswer.
Abbiamo, inoltre, utilizzato piccole parti di codice JavaScript - quindi lato client - per controlli, che altrimenti sarebbero stati impossibili con il solo utilizzo di PHP e HTML.

Attori

Chi può consuma re il nostro web service?

Come già detto nell'introduzione, chiunque dotato di un broswer può usufruire del nostro servizio.
Il sito web è consultabile in tre differenti modi:
- da utente registrato;
- da libreria registrata;
- da visitatore;

Nessuna funzionalità è stata sottratta al visitatore non registrato, così da permettere a tutti di utilizzare il servizio.
Ovviamente, ci sono dei limiti; infatti è impossibile per il visitatore poter lasciare una recensione al libro, votare, oppure aggiungere un libro al catalogo. Questo non è stato fatto per limitare le possibilità di questa categoria di utenti, bensì per un fatto puramente tecnico: nel database non avremmo avuti i dati necessari da inserire.

Cosa può fare il visitatore?

Il visitatore può vedere le recensione inserite su ogni libro, consultare tutto il catalogo e vedere ogni dato del libro, effettuare ricerche (anche quelle avanzate) .

Cosa può fare l'utente registrato?

L'utente può inserire recensioni (modificarle o anche cancellarle), consultare il catalogo ed effettuare ogni tipo di ricerca.

Cosa può fare la libreria?

La libreria può inserire nuovi libri al catalogo e alla propria libreria personale (specificando sconto e copie da lei possedute), può modificare i dati del libro inseriti, effettuare ricerche.

Come si può vedere, la registrazione (specialmente quella utente) non offre molte più funzionalità rispetto al visitatore e questo è stato voluto: il nostro obiettivo era un servizio che fosse accessibile, senza limitazioni e con la minima diffusione di dati personali.
Inoltre, è stata prevista (ma non implementata) la possibilità da parte dell'utente registrato di acquistare libri dalle librerie, consultando il catalogo e scegliendo la libreria che ha l'offerta migliore su quel libro.

Apache HTTPD

Non c'è molto da dire riguardo all'utilizzo di Apache.
Soltanto piccole modifiche sono state apportate al file di configurazione *httpd.conf* per permettere il *RewriteRule*.
Invece, c'è qualcosa da dire riguardo all'utilizzo del file di configurazione *htaccess*.

htaccess e RewriteRule

L'utilizzo della riscrittura degli url è stato di fondamentale importanza e ci ha permesso di utilizzare una pagina PHP per un numero infinito di casi.
Che cosa significa?
Faccio un esempio e spiego meglio.
Prendiamo il caso della pagina PHP *libro.php*; come vedremo nel dettaglio in seguito, basti sapere che tale pagina mostra tutte le informazioni relative ad un libro. E' chiaro che - trattandosi di una libreria - potenzialmente i libri sono infiniti ed è impossibile prevedere una pagina specifica per ogni libro; ma allora come fare per realizzare una sola pagina per tutti i libri?
Ecco che allora ci serviamo del *RewriteEngine*; la riscrittura degli url ci consente proprio questo: convertire una richiesta ad un url differente.
Vediamo ora l'esempio per capire meglio.
Mettiamo che vogliamo vedere le informazioni relative ad un libro che ha ISBN '3n4j6jjdje03'.
Il link che l'utente vedrà sarà *www.miosito.org/books/3n4j6jjdje03*.
A questo punto entra in gioco la *RewriteEngine* e converte quel url in l*ibro.php*?ISBN=3n4j6jjdje03.
La pagina PHP *libro.php* quindi legge come url *libro.php?ISBN=3n4j6jjdje03* e non *www.miosito.org/books/3n4j6jjdje03*.
A questo punto si recupera l'isbn dal url in questo modo

```
function getUrl() {
        return
substr($_SERVER["QUERY_STRING"],strrpos($_SERVER["QUERY_STRING"],"=")+1);
}
$isbn=getUrl();
```

e in base a questo si mostra il libro voluto dal client.
La *RewriteRule* necessaria nell'esempio fatto è

```
RewriteRule ^books/(.*) /libro.php?ISBN=$1
```

Le informazioni utili alla riscrittura del url vengono prese dal file *.htaccess*
Ecco il nostro:

```
Options +FollowSymLinks
RewriteEngine on
RewriteRule ^cancella/recensione/(.*) /rimuoviRecensione.php?ISBN=$1
RewriteRule ^modifica/recensione/(.*) /modificaRecensione.php?ISBN=$1
RewriteRule ^aggiungi/recensione/(.*) /aggiungiRecensione.php?ISBN=$1
RewriteRule ^modifica/libro/(.*) /modificaLibro.php?ISBN=$1
```

RewriteRule ^cancella/libro/(.) /rimuoviLibro.php?ISBN=$1*
RewriteRule ^books/(.) /libro.php?ISBN=$1*
RewriteRule ^autore/(.) /autoreLibro.php?autore=$1*
RewriteRule ^libreria/(.) /profiloPubblicoLibreria.php?libreria=$1*
RewriteRule ^modifica/catalogo/(.) /modificaLibroCatalogo.php?ISBN=$1*

Tutto avviene ovviamente lato server e il client non si accorge di nulla.

Database

DBMS scelto

Abbiamo optato per un RDMBS e nello specifico per MySQL.
E' disponibile sia per sistemi operativi GNU/Linux, sia per Windows, sia per Unix, e questo ci permette - insieme a Java, HTML e PHP, di essere più cross-platform possibili.

La base di dati e Java

Per la comunicazione tra Java - quindi server - e il database abbiamo utilizzato il driver JDBC (MySQL Connector/J, Tipo 4).
JDBC è un connettore che consente l'accesso da qualsiasi programma scritto in Java, indipendentemente da DBMS utilizzato.
MySQL Connector/J è il connector specifico per Java e MySQL; Il Tipo 4 è stato scelto perché fornisce più funzionalità e maggiore performance rispetto a tutti gli altri (Tipo 1, 2, 3); è scritto totalmente in Java e quindi - ancora una volta - questo ci garantisce un maggiore cross-platform.
L'unico neo del Tipo 4 è che ogni DBMS ha il suo connector, quindi se si vuole cambiare DBMS occorre cambiare il driver. Nonostante questo, abbiamo ritenuto questa la scelta migliore: se si vuol cambiare DMBS basterà cambiare la sola funzione di connessione al database e l'import. Il resto sono tutti vantaggi rispetto agli altri Tipi.
Saranno soltanto le funzioni scritte in Java ad interfacciarsi al database, quindi PHP non avrà nessun accesso alla base; ogni interrogazione necessaria del client sarà demandata alle funzioni Java che si faranno carico di ricevere la chiamata da PHP, interrogare il database, recuperare i dati necessari e trasmetterli a PHP.

I dettagli implementativi

Vediamo ora come avviene effettivamente la connessione al database e quali sono gli import necessari.
Cominciamo dagli import:
- import *java.sql.Connection*: è la classe che implementa l'oggetto necessario a mantenere la sessione di connessione al database;
- import j*ava.sql.DriverManager*: è la classe che stabilisce la connessione al database;
- import *java.sql.PreparedStatement* è la classe che implementa l'oggetto che ci consente di effettuare query al database evitando possibili complicazioni derivanti da query maligne;
- import *java.sql.ResultSet*: è la classe che implementa l'oggetto necessario ad accogliere i risultati prelevati dalla query al database;
- import *java.sql.SQLException*: è la classe che implementa la cattura dell'eccezione sollevata da qualunque istruzione che fa riferimento al database;

connectDatabase() è la funzione che si fa carico di stabilire e mantenere la connessione al database.
Attraverso l'istruzione

Class.forName("com.mysql.jdbc.Driver").newInstance();

viene caricata la classe del driver JDBC e generata una nuova istanza.

In questo modo, invece, si crea la connessione al database.

con = DriverManager.getConnection(url, username, password);

dove con è un oggetto Connection, dichiarato come variabile globale così da renderlo accessibile a tutti i metodi della classe Server.
url, username e password sono stringhe che contengono i dati per l'accesso al database.
connectDatabase è richiamata da ogni funzione che deve operare sulla base, per controllare se la connessione è stabilita e - in caso negativo - di effettuarla.

Connector/J e Tomcat

Dato che si tratta di un web service, e stiamo utilizzando Apache Tomcat, dobbiamo rendere disponibile a questo il driver.
E' un operazione necessaria, perché se Tomcat non ha il driver, non può far connettere al database.
Per fare questo è necessario inserire in *$CATALINA_HOME/common/lib* il file .jar del Connector/J; in questo modo il driver sarà disponibile a tutte le applicazioni all'intero del container.

Struttura database

Tabelle

Tabella `commenti`:

La tabella tiene conto di tutte le recensioni che gli utenti posso rilasciare su di un libro.

Struttura:

`id` int(11) NOT NULL AUTO_INCREMENT,
`nickname` varchar(30) NOT NULL,
`ISBN` varchar(30) NOT NULL,
`corpo_commento` varchar(1000) NOT NULL,
`data` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (`id`),
UNIQUE KEY `nickISBN` (`nickname`,`ISBN`),
KEY `ISBN` (`ISBN`)

Tabella `libri_table`:

La tabella contiene il catalogo generale di tutti libri presenti.

Struttura:

`titolo` varchar(30) NOT NULL,
`ISBN` varchar(30) NOT NULL,
`autore` varchar(30) NOT NULL,
`data_pubblicazione` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
`voto` double NOT NULL DEFAULT '0',
`casa_editrice` varchar(30) DEFAULT NULL,
`anno` int(4) DEFAULT NULL,
`lingua` varchar(30) DEFAULT NULL,
`prezzo` double NOT NULL DEFAULT '0',
`nome_libreria` varchar(30) NOT NULL,
PRIMARY KEY (`ISBN`),
UNIQUE KEY `titolo` (`titolo`),
KEY `nome_libreria` (`nome_libreria`)

Tabella `libro_venditore`:

La tabella contiene tutti i libri messi a disposizione dalla librerie.

Struttura:
`nome` varchar(30) NOT NULL,
`ISBN` varchar(30) NOT NULL,
`sconto` int(2) NOT NULL,
`copie` int(4) NOT NULL,
PRIMARY KEY (`nome`,`ISBN`),
KEY `ISBN` (`ISBN`)

Tabella `users`:

La tabella contiene tutti gli utenti iscritti al sito web.

Struttura:
`nickname` varchar(30) NOT NULL,
`password` varchar(30) NOT NULL,
`email` varchar(30) NOT NULL,
`location` varchar(50) NOT NULL DEFAULT 'default',
`data_nascita` date NOT NULL,
`nome` varchar(30) NOT NULL,
`cognome` varchar(30) NOT NULL,

`indirizzo` varchar(30) NOT NULL,
`citta` varchar(30) NOT NULL,
`cap` int(8) NOT NULL,
`telefono` varchar(15) NOT NULL,
`telefono2` varchar(15) NOT NULL,
PRIMARY KEY (`nickname`),
UNIQUE KEY `email` (`email`),
UNIQUE KEY `indirizzo` (`indirizzo`,`citta`,`cap`),
UNIQUE KEY `telefono` (`telefono`),
UNIQUE KEY `telefono2` (`telefono2`),
UNIQUE KEY `location` (`location`)

Tabella `vendors`:

La tabella contiene tutte le librerie iscritte al sito web.

Struttura:
`nome` varchar(30) NOT NULL,
`password` varchar(30) NOT NULL,
`email` varchar(30) NOT NULL,
`indirizzo` varchar(30) NOT NULL,
`citta` varchar(30) NOT NULL,
`cap` varchar(10) NOT NULL,
`partita_iva` varchar(15) NOT NULL,
PRIMARY KEY (`nome`),
UNIQUE KEY `email` (`email`,`partita_iva`),
UNIQUE KEY `indirizzo` (`indirizzo`,`citta`,`cap`)

Tabella `voti_libri`:

La tabella contiene tutti i voti rilasciati dagli utenti su i libri.

Struttura:
`ISBN` varchar(30) NOT NULL,
`username` varchar(30) NOT NULL,
`voto` double NOT NULL,
PRIMARY KEY (`ISBN`,`username`),
KEY `username` (`username`)

Chiavi esterne:

`commenti`:

FOREIGN KEY (`ISBN`) REFERENCES `libri_table` (`ISBN`) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY (`nickname`) REFERENCES `users` (`nickname`) ON DELETE NO ACTION ON UPDATE CASCADE;

`libri_table`:

FOREIGN KEY (`nome_libreria`) REFERENCES `vendors` (`nome`) ON DELETE NO ACTION ON UPDATE CASCADE;

`libro_venditore`:

FOREIGN KEY (`nome`) REFERENCES `vendors` (`nome`) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY (`ISBN`) REFERENCES `libri_table` (`ISBN`) ON DELETE CASCADE ON UPDATE CASCADE;

`voti_libri`:

FOREIGN KEY (`username`) REFERENCES `users` (`nickname`) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (`ISBN`) REFERENCES `libri_table` (`ISBN`) ON DELETE CASCADE ON UPDATE CASCADE;

Diagramma Entità-Relazione

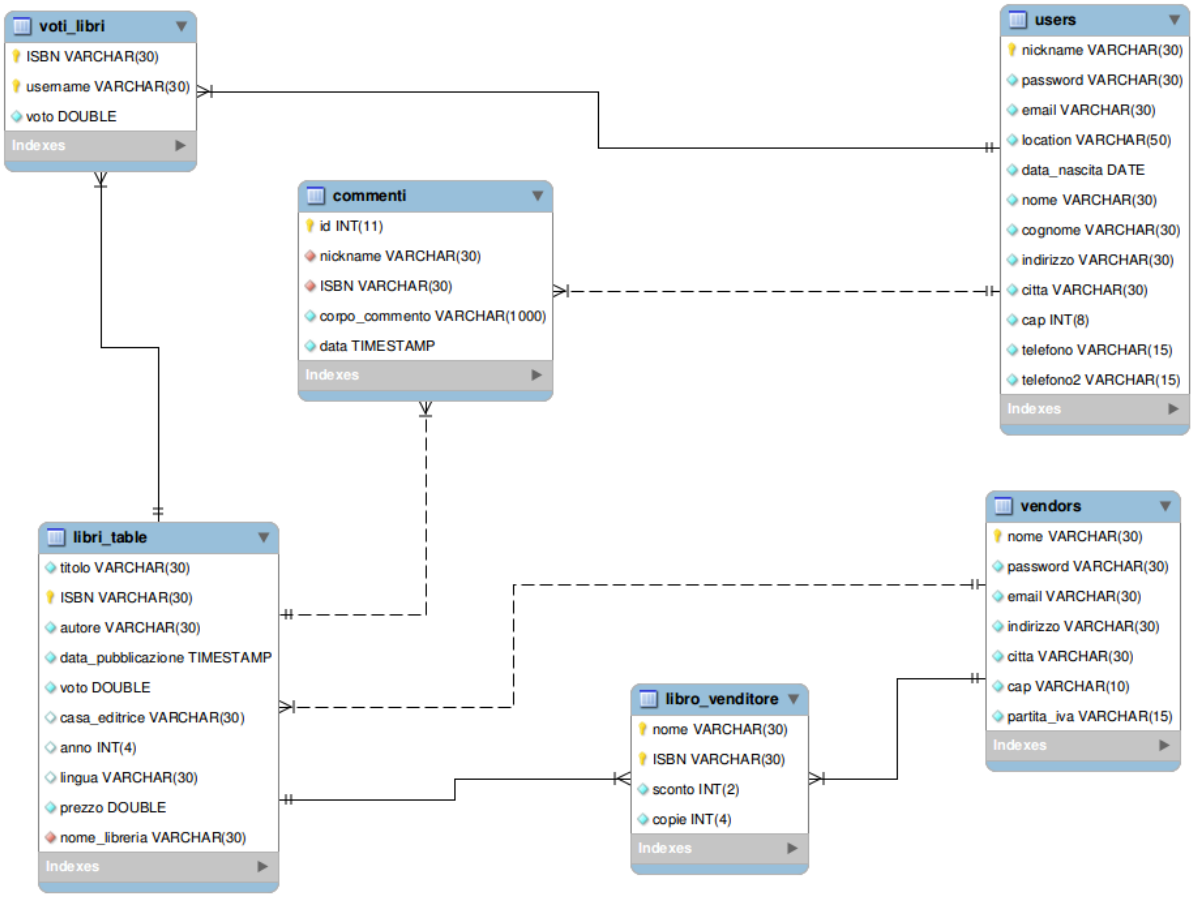


Diagramma ER

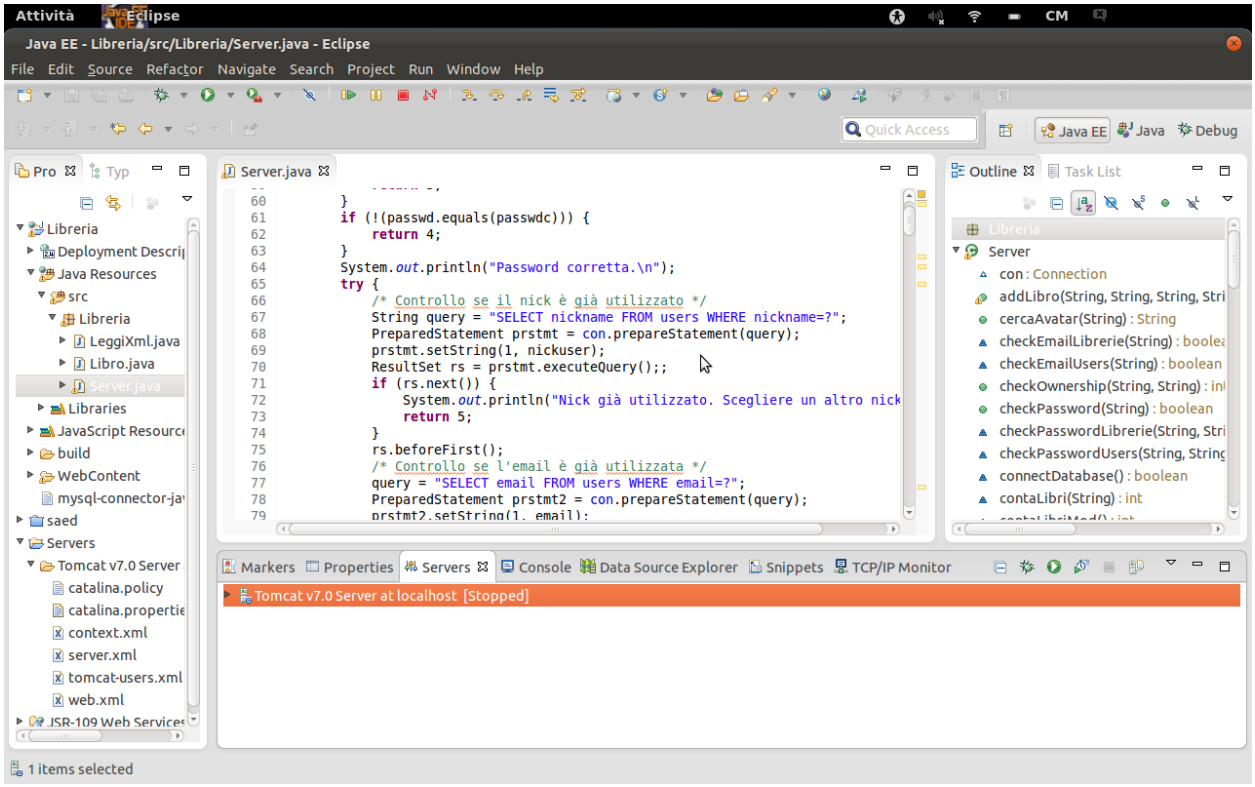
In che modo si è gestito il database?

Per gestire il database ci siamo affidati a *phpmyadmin*.
Fornisce un ottima GUI ed elimina molti dei problemi dovuti all'utilizzo di MySQL da terminale.

Java

In che ambiente è stato sviluppato il codice Java? Come lavorano e a cosa servono le funzioni?

Per sviluppare il codice ci siamo affidati all'ambiente di sviluppo integrato Eclipse.
La versione da noi utilizzata è la Eclipse Java EE IDE for Web Developers, Juno Release.
Inizialmente ci siamo serviti dell'IDE solo per sviluppare il codice, quindi come mero editor di testo; successivamente abbiamo scelto di utilizzare Eclipse per lo sviluppo dell'intero progetto; infatti abbiamo demandato ad Eclipse la gestione del Server Tomcat e sviluppato con i suoi strumenti un progetto web dinamico.
Questa scelta penso si sia rivelata ottima, perché seppur all'inizio ha richiesto un maggior impiego di tempo dovuto alla configurazione dell'IDE e al prendere consapevolezza degli strumenti su cui dovevamo operare, nel seguito ci ha semplificato e velocizzato lo sviluppo dell'applicazione.



Eclipse

Eclipse e Tomcat

Come detto, la gestione del server Tomcat è stata affidata all'IDE e quindi la pubblicazione, l'avvio, il restart e lo stop avvenivano utilizzando gli strumenti messi a disposizione da Eclipse.

Le funzioni Java

In ordine alfabetico sono elencate e descritte le funzioni.
Dove non specificato le funzioni sono da considerarsi public, cioè consumabili dal client.

AddLibro:

```
public int addLibro(String titolo, String isbn, String autore, String
                editore, String anno, String lingua, String prezzo,
                String copie, String sconto, String email);
```

Tale funzione riceve in input tutti i campi che devono essere esplicitati quando si vuol inserire un libro all'interno del nostro database.

La funzione ritorna un intero, che sarà poi valutato all'interno delle pagine PHP per capire se l'inserimento è andato a buon fine, o se, in caso di errore, cosa sia andato male.

La funzione - inizialmente - ricerca all'interno del catalogo (*libri_table*) se esiste già un libro con lo stesso ISBN, in caso affermativo non aggiunge il libro al catalogo, ma solo alla libreria del venditore (*libro_venditore*). Se, invece, non dovesse trovare nel catalogo il libro in questione, lo aggiunge sia al catalogo che alla libreria del venditore.

cercaAvatar:

```
public String cercaAvatar(String username);
```

Tale funzione si occupa di ricercare l'avatar del profilo dell'utente.

Più in dettaglio: la funzione riceve come parametro il nickname dell'utente e ritorna la posizione in cui si trova l'avatar; si cerca all'interno della tabella users il valore dell'attributo position e si ritorna esso come risultato della funzione.

checkEmailLibrerie e checkEmailUsers:

```
boolean checkEmailLibrerie(String email);
boolean checkEmailUsers(String email);
```

Le due funzione sono raggruppate perché svolgono il medesimo compito; entrambe, infatti, si occupano di controllare che l'email utilizzata durante la registrazione di libreria e utente non sia giù utilizzata.

Le due funzioni ricevo come parametro l'email e ritornano un valore booleano che sarà false se l'email non è presente e true se invece è presente.

Inoltre, queste funzioni non sono public, cioè non sono accessibili e consumabili dal client, ma sono funzione di gestione che vengono utilizzata esclusivamente all'interno del server per controlli di sistema.

checkOwenership:

```
public int checkOwnership(String email,String isbn);
```

La funzione è utilizzata per controllare la paternità di un libro all'interno del catalogo, ovvero quale libreria ha inserito il libro in *libri_table*.

Più nello specifico: la funzione riceve due parametri, email e ISBN e cerca all'interno della tabella *libri_table* una corrispondenza nella tupla per email e ISBN.

A seconda del risultato ottenuto dall'interrogazione del database verrà ritornato un valore

intero.

Per chiarire meglio la sua utilità, la funzione è da inserirsi nel contesto della modifica del libro presente nel catalogo. Questa infatti è utilizzata all'interno della pagina PHP *libro.php* per consentire solo all'autore dell'inserimento del libro di poter modificare le caratteristiche del libro.

checkPassword:

```
public boolean checkPassword(String password);
```

La funzione si occupa semplicemente di controllare se la password inserita durante la registrazione (sia dell'utente che della libreria) contenga un numero di caratteri non inferiori 8.

La funzione riceve come parametro la password da controllare e ritorna un valore booleano che sarà true se la password rispetta i criteri, ovvero false se non rispetterà i requisiti richiesti.

checkPasswordLibrerie e checkPasswordUsers:

```
boolean checkPasswordLibrerie(String email, String passwd);
boolean checkPasswordUsers(String nickuser, String passwd);
```

Le due funzioni sono riunite perché assolvono allo stesso compito, cioè controllare la correttezza della password inserita.

Nel caso di checkPasswordLibrerie la funzione riceve come parametri l'email e la password, mentre nel caso di checkPasswordUsers riceve il nickname e la password.

Entrambe ritornano true se il valore della password coincide, e false in caso contrario.

Ovviamente in checkPasswordLibrerie la query è fatta sulla tabella *vendors*, mentre in checkPasswordUsers è fatta su users.

Le due funzioni, inoltre, sono disponibili solo al server, e quindi non consumabili per il client, cioè non sono dichiarate public.

connectDatabase:

```
boolean connectDatabase();
```

Questa funzione realizza e controlla la connessione al database della libreria.

Questa funzione non riceve nessuna parametro e ritorna un valore booleano true se viene effettuata con successo o se già presente, ovvero false se, invece, la connessione non avviene.

Questa funzione è chiamata ogni volta che un altra funzione è chiamata ad operare sul database;

La funzione opera su un oggetto Connection dichiarato globalmente alla classe che è inizializzato a null; *connectDatabase* controlla l'oggetto Connection e se è null prova a creare la connessione, altrimenti (cioè se non è null, il che sta a significare che la connessione con il database è attiva e funzionante) restituisce direttamente true.

Ovviamente se l'oggetto Connection è null e la funzione non riesce a stabilire la connessione il database viene ritornato false, ovvero true.

La funzione non è dichiarata public, quindi è utilizzata solo all'interno del server.

contaLibri:

```
int contaLibri(String query);
```

Questa funzione si occupa di contare il numero dei libri presenti in qualunque tabella del database.

Nel dettaglio: la funzione riceve come parametro una qualsiasi stringa di query; successivamente essa estrapola la parte significativa e utilizzabile (la stringa a partire dalla parola

chiave FROM) e la concatena ad una stringa generale per il conteggio dei libri presenti aventi determinati valori per gli attributi selezionati.

La funzioni ritorna un intero corrispondente al numero dei libri trovati per la query aventi come valori quelli passati nella stringa del parametro.

Un valore pari a 0 sta a significare errore nella query, sia nessun risultato ottenuto per quei valori.

La funzione non è public, quindi non è fruibile da parte del client.

contaLibriMod:

```
int contaLibriMod();
```

La funzione semplicemente conta il numero dei libri presenti all'interno della tabella *libri_table*.

Non riceve nessun parametro e ritorna un intero, corrispondente al numero di tuple selezionate; un numero negativo è ritornato nel caso in cui si presenti un errore all'interno della funzione.

La funzione non è public, quindi non consumabile da parte del client.

contaLibriRicerca:

```
public int contaLibriRicerca(String campo);
```

La funzione conta il numero dei libri presenti all'interno del catalogo (*libri_table*) aventi specifici valori come attributi.

Nel dettaglio: la funzione riceve come parametro una stringa in cui è contenuto la chiave di ricerca con cui contare il numero di tuple. E' utilizzata nella ricerca semplice, cioè quella che controlla ogni campo, per cui il valore passato come parametro viene impostato su ogni attributo nella tabella e tutti gli attributi messi in OR logico. Qualsiasi tupla che contenga come valore ricercato viene presa in considerazione. Attenzione, l'utilizzo del verbo "contenere" non è casuale, infatti per selezionare una tupla è sufficiente che questa abbia la chiave di ricerca come sua sotto-stringa.

Un valore intero è ritornato. Un numero negativo mi identifica un errore all'interno della funzione, un valore pari a 0, invece, identifica "nessun risultato", mentre un numero maggiore di 0 identifica il successo della funzione e anche il numero di tuple ritornate.

contaRecensioni:

```
int contaRecensioni();
```

La funzione ritorna il numero di recensioni presenti all'interno della tabella *commenti*.

Più nello specifico: non riceve nessun parametro, quindi si intuisce che la funzione fa un conteggio totale delle tuple presenti all'interno della tabella commenti (nessun clausola WHERE è specificata). Come ritorno ha un intero: ad un 0 corrisponde correttezza della funzione e nessuna tupla presente, ad valore maggiore di 0 corrisponde correttezza della funzione e il numero di tuple trovate, mentre ad numero negativo corrisponde un errore all'interno della funzione.

La funzione non è public, e viene utilizzata esclusivamente all'interno del server e non resa disponibile al client.

deleteLibro:

```
public int deleteLibro(String email, String ISBN);
```

La funzione elimina un libro presente all'interno della libreria del venditore.

Qualche dettaglio: si occupa di eliminare un libro all'interno di libro_venditore solo se questo è posseduto dalla libreria che sta tentando la rimozione. Riceve come parametro l'ISBN e l'email della libreria.

Opera in questo modo: ricerca l'email all'interno di *vendors*, se non trova corrispondenze restituisce errore, altrimenti preleva il nome della libreria che ha l'email passata per parametro. Ottenuto il nome passa ad individuare il libro all'interno di libro_venditore: cerca una corrispondenza sulle tuple che abbiamo ISBN e nome come richiesti; se non trova nulla restituisce errore, altrimenti elimina il libro.

La funzione ritorna un valore diverso da 0 in caso di errore, mentre ritorna 0 in caso di successo.

editLibreria:

```
public int editLibreria(String[] campi);
```

La funzione realizza l'aggiornamento/editing/modifica dei dati presenti all'interno del profilo della libreria.

Qualcosa in più: viene fatta una prima distinzione tra modifica password e il resto; infatti essendo la password un dato di fondamentale importanza occorre una separazione dal resto per dedicargli maggiore attenzione. Viene per prima cosa controllata se una nuova password e una vecchia password sono state inserite; per fare questo si controlla se la stringa è diversa da quella vuota: se lo è, allora si vuol modificare l'email. A questo punto si controlla la validità della vecchia password, per mettere la modifica di questa. Se anche questo controllo riesce si passa al controllo della validità della nuova email, cioè se rispetta i requisiti richiesti. Se pure questo controllo da esito positivo si può effettuare il cambio di password.

Un'altra distinzione deve essere fatta nel caso si voglia modificare l'email: questa infatti è la chiave primaria della tabella *vendors*, e quindi occorre prestare un maggior controllo alla sua modifica. Viene quindi controllato se le due email coincidono: se coincidono si esclude l'email dalla modifica e si modificano gli altri dati; se invece non coincidono si procede a controllare se la nuova email è già utilizzata all'interno della tabella: se è esistente viene restituito errore, altrimenti si modifica l'email.

La funzione ritorna un valore intero.

editLibroCatalogo:

```
public int editLibroCatalogo(String campi[]);
```

La funzione si occupa di modificare i dati di un libro presente all'interno del catalogo (*libri_table*).

Dettagli: la funzione riceve come parametro un array di stringhe in cui sono presenti i campi del libro.

La funzione procede ad aggiornare tutti i campi con i valori presenti all'interno dell'array senza effettuare controlli, questo perché vengono passati ogni volta tutti i valori, anche quelli che non vengono modificati. Se da un lato questa scelta aumenta il carico perché ogni volta vengono modificati attributi inutilmente, dall'altro evita un carico overhead dovuto al controllo da farsi su ogni valore per verificare se questo sia stato o meno aggiornato.

Ovviamente la modifica di un libro all'interno di un catalogo è garantita soltanto alla libreria che ha inserito quel libro, e non alle altre. Questo controllo è svolto precedentemente sulla pagina PHP che permette la modifica al catalogo.

Tutti i dati del libro sono modificabili, ad eccezione dell'ISBN. Questa esclusione è necessaria per evitare incongruenze che si possono verificare all'interno del database. La modifica dell'ISBN è demandata all'admin, anche su segnalazione delle librerie e degli utenti.

La funzione ritorna ancora una volta un intero che avrà valore negativo in caso di errore, e

valori pari a 0 in caso di successo.

editLibroVenditore:

```
public int editLibroVenditore(String[] campi);
```

La funzione si occupa di modificare i dati di un libro presente all'interno della libreria del venditore (*libro_venditore*).

Dettagli: la funzione riceve come parametro un array di stringhe in cui sono presenti i campi del libro.

La funzione inizialmente ottiene il nome della libreria e passa poi a modificare i campi senza ulteriori controlli.

Anche in questa situazione è stata scelta la strada della modifica di tutti i valori, come nel caso della funzione *editLibroCatalogo*.

La funzione ritorna ancora una volta un intero che avrà valore negativo in caso di errore, e valori pari a 0 in caso di successo.

editUser:

```
public int editUser(String[] campi);
```

La funzione realizza l'aggiornamento dei dati relativi al profilo di utente.

Dettagli: come nel caso di *editLibreria*, un attenzione maggiore viene riservata alla modifica della password; il procedimento è il medesimo_____

inserisciRecensione:

```
public boolean inserisciRecensione(String nickname, String ISBN,
        String corpo);
```

La funzione si occupa di inserire la recensione rilasciata dall'utente all'interno della tabella commenti.

Dettagli: la funzione riceve come parametri il nickname dell'utente che ha sottomesso il commento, l'ISBN del libro che ha ricevuto il commento e il corpo della recensione (o commento). Viene costruita la query prendendo come valori i parametri e la data corrente e viene fatto l'inserimento.

La funzione ritorna un valore booleano che sarà true nel caso in cui l'inserimento vada a buon fine, ovvero false negli altri casi.

iscrizioneLibrerie:

```
public int iscrizioneLibrerie(String nome, String passwd,
              String passwdc,String email, String indirizzo, String
              citta, String cap, String partitaIva);
```

La funzione consente e realizza l'iscrizione di una libreria al sito web.

Dettagli: la funzione riceve come parametri i dati essenziali che una libreria deve necessariamente avere, e sono: nome, password, la conferma della password, l'email, l'indirizzo, la città, il cap e la partita iva.

Prima si procede al controllo dell'email: se rispetta i requisiti e se le due password coincidono. Poi si passa al controllo dell'unicità del nome, dell'email e della partita iva; se tutti i controlli vengono superati si procede all'inserimento nella tabella *vendors* dei nuovi dati ottenuti.

La funzione ritorna un valore intero che sarà 1 in caso di corretto inserimento, altrimenti assumerà un valore intero positivo a seconda dell'errore verificatosi.

iscrizioneUser:

```
public int iscrizioneUser(String nickuser,String passwd,String
                passwdc, String email);
```

La funzione realizza l'iscrizione di un utente al sito web.

Dettagli: la funzione riceve i dati necessari ad un utente per potersi iscrivere, e sono: nickname, password, la conferma della password e l'email.

Come nel caso di *iscrizioneLibrerie*, vengono fatti controlli sulla password (rispetto dei requisiti e coincidenza delle due password) e poi si passa al controllo dell'unicità di nickname ed email; se tutti i controlli vengono superati con successo viene fatto l'inserimento all'interno della tabella *users* dei nuovi dati recepiti.

La funzione ritorna un intero che avrà valore in caso di inserimento riuscito, altrimenti assumerà valori interi positivi a seconda dell'errore verificatosi.

leggiAutore, leggiDataPubblicazione, leggiEditore, leggiLingua, leggiPrezzo, leggiTitolo, leggiVoto:

```
public String leggiAutore(String isbn);
public String leggiDataPubblicazione(String isbn);
public String leggiEditore(String isbn);
public String leggiLingua(String isbn);
public String leggiPrezzo(String isbn);
public String leggiTitolo(String isbn);
public double leggiVoto(String isbn);
```

Tutte queste funzione si occupano di leggere dati che sono riferiti ad un libro.

Dettagli: tutte le funzione ricevono come parametro l'ISBN del libro che sarà utilizzato nella query per selezionare l'attributo richiesto dalla funzione (autore, data pubblicazione, editore, lingua, prezzo, titolo, voto); viene controllato che una tupla sia stata selezionata e in caso positivo si procede all'estrazione del valore richiesto.

Le funzione ritorna in tutti i casi una stringa che sarà il valore richiesto se tutto sarà andato a buon fine, mentre sarà una stringa vuota in caso di errore e di mancata selezione della tupla.

leggiDatiLibreria:

```
public String[] leggiDatiLibreria(String email);
```

La funzione si occupa di prelevare tutti i dati che fanno riferimento ad una particolare libreria.

Dettagli: la funzione riceve per parametro l'email della libreria di cui si vuol recuperare i dati; seleziona la tupla della libreria utilizzato il parametro email e restituisce un array di stringhe contenente i dati della libreria.

leggiDatiUtente:

```
public String[] leggiDatiUtente(String nickuser);
```

La funzione si occupa di prelevare tutti i dati che fanno riferimento ad un particolare utente.

Dettagli: la funzione riceve come parametro il nickname dell'utente di cui si vuol recuperare i dati; seleziona la tupla utilizzando il parametro nickname e restituisce un array di stringhe contenente i dati dell'utente.

leggiISBN:

```
public String leggiISBN(String titolo);
```

La funzione preleva l'ISBN del libro richiesto.

Dettagli: la funzione riceve come parametro il titolo del libro ricercato e lo sfrutta nella query per selezionare la tupla corrispondente; preleva l'ISBN dalla tupla selezionata con la query e lo restituisce.

La funzione ritorna una stringa che conterrà l'ISBN in caso di successo, altrimenti una stringa vuota.

leggiLibro:

```
public String[] leggiLibro(String isbn);
```

La funzione restituisce i valori del libro ricercato (titolo, autore, casa editrice, anno, prezzo e lingua).

Dettagli: la funzione riceve come parametro l'ISBN del libro richiesto e seleziona la tupla corrispondente dalla tabella *libri_table*, cioè dal catalogo; la funzione, infatti, si occupa di leggere i dati del libro presenti nel catalogo; tutti i dati vengono prelevati e inseriti in un array di stringhe che sarà poi il ritorno della funzione in questione.

Un controllo preventivo si rende necessario per verifica l'effettiva presenza del libro richiesto. In caso il libro non sia presente viene ritornato immediatamente un array di stringhe vuoto, in caso positivo si procede con l'estrazione dei dati richiesti.

Come detto la funzione ritorna un array di stringhe.

leggiLibroVenditore:

```
public String[][] leggiLibroVenditore(String isbn);
```

La funzione restituisce i valori del libro ricercato presente nella libreria del venditore.

Dettagli: a differenza di *leggiLibro*, questa funzione legge i dati presenti nella tabella libro_venditore, che corrisponde alla libreria personale di ogni libreria. La funzione riceve come parametro l'ISBN del libro ricercato e procede alla costruzione di una matrice di stringhe contente i libri che hanno l'ISBN richiesto e reso disponibile da ogni libreria.

Inizialmente vengono contate il numero di tuple presenti all'interno della tabella libro_venditore che hanno l'ISBN richiesto; questo viene attraverso una semplice query che avrà come clausola WHERE proprio l'ISBN in questione.

Il numero così ottenuto servirà come parametro per la creazione della matrice che conterrà di dati richiesti; viene quindi creata una matrice di stringhe che avrà un numero di righe pari al numero precedentemente trovato e un numero di colonne pari a 3; infatti dovrà avere come attributi: nome, sconto e copie.

Creata la matrice di procede a prelevare da libro_venditore i dati richiesti; si effettua un ulteriore controllo per verificare che delle tuple sia state effettivamente selezionate e in caso positivo si prelevano i dati dal ResultSet, altrimenti viene ritornata una matrice vuota.

leggiNomeLibreria:

```
public String leggiNomeLibreria(String email);
```

La funzione legge il nome della libreria richiesta.

Dettagli: la funzione riceve come parametro l'email della libreria di cui si vuol recuperare il nome. Procede ad una semplice query in cui viene inserita come clausola WHERE l'email; si controlla che il ResultSet contenga la tupla e in caso positivo si estrae il nome richiesto, altrimenti viene restituito errore.

La funzione ritorna una stringa che conterrà il nome della libreria nel caso in cui non si siano presentati errori, altrimenti una stringa vuota.

leggiNumeroRecensioni:

```
public int leggiNumeroRecensioni(String isbn);
```

La funzione il numero di recensioni presenti per il libro richiesto.

Dettagli: la funzione riceve come parametro l'ISBN del libro richiesto e procede all'interrogazione della tabella *commenti* contando il numero di tuple presenti che abbiamo come valore dell'attributo ISBN l'ISBN passato per parametro. Anche in questa situazione viene controllato che il ResultSet abbia effettivamente un contenuto e in caso positivo si estrae il dato richiesto, altrimenti viene restituito errore.

La funzione ritorna un intero che sarà 0 in caso di errore, e avrà, invece, valore positivo negli altri casi.

leggiRecensione:

```
public String[] leggiRecensione(String nickname, String isbn);
```

La funzione preleva i dati necessari a visualizzare la recensione rilasciata da un particolare utente su uno specifico libro.

Dettagli: la funzione riceve come parametro il nickname dell'utente richiesto e l'ISBN del libro.

Viene prima recuperato il titolo del libro interrogando la tabella *libri_table* utilizzando nella clausola WHERE l'ISBN del libro. Se una tupla è stata selezionata si estrae il dato, altrimenti viene restituito errore.

Si passa poi al recupero del corpo del commento: si interroga la tabella commenti specificando nickname e ISBN; se una tupla è stata selezionata si estrae il dato, altrimenti viene restituito errore.

Viene infine ritornato un array contente il titolo del libro e il corpo della recensione.

La funzione, quindi, ritorna un array di stringhe che conterrà titolo e corpo in caso errore, vuoto altrimenti.

leggiRecensioneLibro:

```
public String[][] leggiRecensioniLibro(String isbn);
```

La funzione si occupa di restituire tutte le recensioni rilasciate su un particolare libro.

Dettagli: la funzione riceve un unico parametro che è l'ISBN del libro di cui si vuole prelevare tutte le recensioni; viene da prima richiamata la funzione *leggiNumeroRecensioni* per recuperare il numero di recensioni presenti per quel libro; il numero servirà per creare la matrice che accoglierà i dati richiesti. Il numero, in particolare, verrà utilizzato per la lunghezza delle righe, mentre il numero di colonne rimarrà fisso a 4.

Creata la matrice si procede ad effettuare la query; la tabella interrogata è commenti e da essa vengono estratti il corpo, la data e il nickname e l'id; si controlla che il ResultSet contenga effettivamente delle tuple e in caso negativo viene restituito un errore, altrimenti si procede all'estrazione dei dati richiesti. Infine si popola la matrice con i dati prelevati e si restituisce questa come risultato.

La funzione ritorna una matrice non vuota se l'esito è stato positivo, altrimenti avrà valore nullo.

leggiRecensioniUtente:

```
public String[][] leggiRecensioniUtente(String nickname);
```

La funzione restituisce tutte le recensioni sottomesse da un particolare utente.

Dettagli: la funzione ha come unico parametro il nickname dell'utente del quale si vogliono recuperare le recensioni.

Vengono prima contate il numero di recensioni presenti all'interno della tabella commenti per quell'utente; tale numero verrà utilizzato per creare la matrice che andrà a contenere i risultati richiesti. Se, però, il numero sarà uguale a 0 verrà restituito errore, altrimenti creata la matrice.

A questo punto si procede all'estrazione di tutte le tuple di quel particolare utente. I valori estratti verranno inseriti nella matrice e restituito come risultato della funzione.

La funzione ritorna - appunto - una matrice di stringhe che avrà contenuto non nulla in caso di esito positivo, altrimenti avrà valori nulli.

leggiScontoCopie:

```
public String[] leggiScontoCopie(String ISBN);
```

La funzione si occupa di leggere il valore di sconto e copie di un particolare libro.

Dettagli: la funzione riceve come parametro l'ISBN del libro richiesto e procede ad estrarre dalla tabella *libro_venditore* titolo, sconto e copie. L'attributo titolo non è presente in tale tabella, ma viene recuperato mediante un JOIN dalla tabella *libri_table*.

Un controllo viene realizzato sul ResulSet per verificare che contenga delle tuple: in caso negativo si ritorna errore, altrimenti si procede all'estrazione dei dati.

La funzione ritorna un array di stringhe che conterrà valori non nulli in caso di esito positivo, altrimenti avrà valori nulli.

leggiVotoUtente:

```
public int leggiVotoUtente(String isbn, String nickname);
```

La funzione legge il voto rilasciato da un particolare utente su uno specifico libro.

Dettagli: la funzione riceve come parametri il nickname e l'ISBN richiesti.

Interroga la tabella voti_libri prelevando il voto sottomesso dall'utente richiesto su quel particolare libro; viene controllato il contenuto del ResultSet e se questo è vuoto viene restituito errore, altrimenti si procede.

Viene estratto il valore recuperato con la query e restituito come valore di ritorno della funzione.

La funzione, quindi, ritorna un intero che avrà valori positivi maggiori di 0 in caso di successo, altrimenti valore uguale a 0.

listaNovita:

```
public String[][] listaNovita();
```

La funzione recupera gli ultimi libri inseriti all'interno del catalogo.

Dettagli: la funzione non riceve alcun parametro e procede al conteggio del libri presenti in *libri_table* attraverso la funzione *contaLibriMod*; il numero recuperato viene controllato e utilizzando in questo modo: se il numero è minore o uguale a 0 viene restituito errore, se è compreso da 0 (escluso) e 4 (incluso) viene creata una matrice di righe pari a numero e colonne pari a 3, se invece numero è maggiore di 4 (escluso) viene creata una matrice di righe pari a 4 e colonne pari a 3.

La matrice così creata conterrà il risultato che la funzione ritornerà.

A questo punto si passa all'interrogazione di *libri_table* e all'estrazione dei dati necessari;

vengono selezionati dalla tabella il titolo, l'autore e la data di pubblicazione di ogni libro; prima di prelevare i valori viene però controllato il ResultSet, e nel caso in cui fosse vuoto viene ritornato errore. Passato questo controllo si passa a popolare la matrice precedentemente creata che verrà ritornata dalla funzione in caso di successo.

La funzione, quindi, ritorna una matrice con valori non nulli in caso di esito positivo, altrimenti una matrice nulla.

loginLibrerie:

```
public int loginLibrerie(String email, String passwd);
```

La funzione si occupa di recuperare i dati necessari all'approvazione della login della libreria.

Dettagli: la funzione riceve come parametri l'email e la password; entrambi vengono utilizzati per interrogare la tabella *vendors* al fine di controllare i dati ricevuti ed approvare o respingere il tentativo di login.

Se il ResultSet ottenuto tramite la query contiene tuple, allora la login ha avuto successo, altrimenti si rifiuta la login perché o email e password non coincidono, oppure l'email inserita non è presente all'interno della tabella.

La funzione ritorna valori interi che sarà 0 in caso di esito positivo, 1 in caso di errore, -1 in caso di rifiuto della login e -2 in caso di mancata connessione al database.

loginUsers:

```
public int loginUsers(String nickuser, String passwd);
```

La funzione si occupa di recuperare i dati necessari all'approvazione della login dell'utente.

Dettagli: la funzione riceve come parametri il nickname e la password; entrambi vengono utilizzati per interrogare la tabella *users* al fine di controllare i dati ricevuti ed approvare o respingere il tentativo di login.

Se il ResultSet ottenuto tramite la query contiene tuple, allora la login ha avuto successo, altrimenti si rifiuta la login perché o il nickname e la password non coincidono, oppure il nickname inserito non è presente all'interno della tabella.

La funzione ritorna valori interi che sarà 0 in caso di esito positivo, 1 in caso di errore, -1 in caso di rifiuto della login e -2 in caso di mancata connessione al database.

modificaRecensione:

```
public int modificaRecensione(String nickname, String ISBN, String
                corpo);
```

La funzione realizza la modifica della recensione inserita da uno specifico utente su quel particolare libro.

Dettagli: la funzione riceve come parametri il nickname, l'ISBN e il corpo della recensione e procede ad aggiornare la tupla corrispondente presente all'interno della tabella commenti, dove nickname ed ISBN siano quelli dei parametri.

La funzione ritorna un intero che sarà 0 in caso di successo, -2 in caso di errore generico e -1 in caso di mancata connessione al database.

mostraVoti:

```
public String[][] mostraVoti();
```

La funzione si recupera i libri che hanno il maggior punteggio nel voto.

Dettagli: la funzione non riceve nessun parametro e procede al conteggio del libri presenti in

libri_table attraverso la funzione *contaLibriMod*; il numero recuperato viene controllato e utilizzando in questo modo: se il numero è minore o uguale a 0 viene restituito errore, se è compreso da 0 (escluso) e 4 (incluso) viene creata una matrice di righe pari a numero e colonne pari a 3, se invece numero è maggiore di 4 (escluso) viene creata una matrice di righe pari a 4 e colonne pari a 3.

La matrice così creata conterrà il risultato che la funzione ritornerà.

A questo punto si passa all'interrogazione di l*ibri_table* e all'estrazione dei dati necessari; vengono selezionati dalla tabella il titolo, l'autore e il voto di ogni libro; prima di prelevare i valori viene però controllato il ResultSet, e nel caso in cui fosse vuoto viene ritornato errore. Passato questo controllo si passa a popolare la matrice precedentemente creata che verrà ritornata dalla funzione in caso di successo.

La funzione, quindi, ritorna una matrice con valori non nulli in caso di esito positivo, altrimenti una matrice nulla.

ricerca:

```
public String[][] ricerca(String campo);
```

La funzione restituisce i libri che hanno come valori degli attributi una particolare stringa.

Dettagli: la funzione ha un unico parametro che rappresenta la stringa ricercata.

Si interroga la tabella *libri_table* inserendo la stringa ricercata in ogni attributo presente nella tabella e tutti i campi vengono selezionati. Particolare attenzione deve essere riservato all'utilizzo dell'operatore LIKE e dei simboli '%'; infatti verranno selezionate anche tuple che hanno come valore dell'attributo la sotto-stringa ricercata.

Viene poi richiamata la funzione *contaLibri* così da creare una matrice che abbia le giuste dimensioni per accogliere i libri selezionati; se il numero trovato è uguale a 0 viene restituito errore, mentre se è diverso da 0 (ma comunque maggiore di 0), si procede alla creazione della matrice che accoglierà i risultati ottenuti dalla query.

A questo punto viene popolata la matrice con i dati prelevati dalla tabella.

La funzione ritorna una matrice con valori non nulli in caso di successo, altrimenti una matrice con valori nulli.

ricercaAnno, ricercaAutore, ricercaCasaEditrice, ricercaISBN, ricercaLingua, ricercaPrezzo, ricercaTitolo, ricercaVoto:

```
public String[][] ricercaAnno(String campo);
public String[][] ricercaAutore(String campo);
public String[][] ricercaCasaEditrice(String campo);
public String[][] ricercaISBN(String campo);
public String[][] ricercaLingua(String campo);
public String[][] ricercaPrezzo(String campo);
public String[][] ricercaTitolo(String campo);
public String[][] ricercaVoto(String campo);
```

Tutte le funzioni si occupano di effettuare la ricerca di una stringa su un particolare attributo del catalogo.

Dettagli: tutte le funzioni ricevono un unico parametro che rappresenta la chiave di ricerca; la funzione prosegue esattamente come per la funzione ricerca, differenziandosi su un solo punto: la query di ricerca viene fatta su un unico campo, specifico per ogni funzione;

La funzione ritorna una matrice con valori non nulli in caso di successo, altrimenti una

matrice con valori nulli.

ricercaAvanzata:

```
public String[][] ricercaAvanzata(String[] campi);
```

La funzione realizza una ricerca all'interno del catalogo selezionando una particolare stringa su di uno specifico attributo della tabella.

Dettagli: la funzione riceve come unico parametro un array di stringhe; il compito iniziale svolto dalla funzione è quello di capire quali campi sono stati scelti per la ricerca: per fare questo controlla quali campi siano vuoti e quali no; se sono vuoti li ignora, altrimenti li esamina. Un contatore tiene traccia della posizione in cui si dovranno posizionare nella query i campi richiesti dalla ricerca e non vuoti; questo (il contatore) avrà anche il compito di segnalare se l'attributo preso in esame è il primo o no, e quindi dare la possibilità di differenziare il lavoro svolto dalla funzione. Un ulteriore controllo sull'indice del ciclo farà capire su quale campi si sta lavorando.

Dopo aver capito quali campi sono stati scelti per la ricerca e costruita la query per la ricerca, si passa a settare la query con l'aiuto del contatore: infatti il contatore ha segnato su ogni stringa titolo, anno, editore... la posizione che essi occupano all'interno della query; tale comportamento si è dovuto adottare dato che a tempo di compilazione non è possibile sapere quali e quanti campi verranno utilizzati per la ricerca avanzata.

Settata la query, si richiama la funzione *contaLibri* per recuperare il numero di libri presenti, così da costruire una matrice sufficiente ad accogliere tutti i libri selezionata dalla query di ricerca.

Terminati questi lavori, si passa a popolare la matrice con i dati prelevati dall'interrogazione.

La funzione ritorna una matrice che avrà valori non nulli se l'esito è stato positivo, nulli altrimenti.

ricercaLibriLibreria:

```
public String[][] ricercaLibriLibreria(String nomeLibreria);
```

La funzione si occupa di selezionare tutti i libri presenti all'interno della libreria personale di una particolare libreria.

Dettagli: la funzione riceve come unico parametro il nome della libreria.

Inizialmente vengono contati il numero di libri presenti nella tabella libro_venditori appartenenti a quella libreria: se tale numero è uguale a 0 viene restituito errore, altrimenti viene creata una matrice in grado di accogliere tutti le tuple selezionate.

Vengono poi prelevati dalla tabella libro_venditore ISBN, sconto e copie dei libri il cui attributo nome risulta essere quello della libreria ricercata. I dati prelevati vengono così inseriti nella matrice.

A questo punta la matrice è parzialmente piena, infatti devono ancora essere inseriti dati necessari che però non sono presenti in libro_venditore, bensì in *libri_table*.

Viene quindi fatta un ulteriore interrogazione, ma questa volta sulla tabella *libri_table* e prelevati gli attributi (o meglio i valori degli attributi) relativi a titolo, autore, casa editrice, lingua, anno e prezzo.

Il problema di selezionare soltanto i libri che sono presenti in libro_venditore ed sono di tale libreria è risolto mediante l'utilizzo di una query annidata (si poteva anche utilizzare un JOIN), dove nella query interna viene selezionato l'ISBN dei libri della libreria.

La matrice viene quindi ulteriormente popolata con i nuovi dati estratti.

La funzione ritorna una matrice che avrà valori non nulli in caso di esito positivo, nulli altrimenti.

rimuoviRecensione:

```
public int rimuoviRecensione(String nickname, String ISBN);
```

La funzione effettua l'eliminazione di un commento di uno specifico utente su di un particolare libro.

Dettagli: la funzione riceve come parametri il nickname e l'ISBN del libro.

Costruisce una query in cui gli attributi della clausola WHERE sono settati con i parametri.

La funzione ritorna un intero che sarà uguale ad 1 in caso di esito positivo, 3 in caso di errore generico e 2 in caso di mancata connessione con il database.

ultimeRecensioni:

```
public String[][] ultimeRecensioni();
```

La funzione si recupera le ultime recensione inserite.

Dettagli: la funzione non riceve nessun parametro e procede al conteggio dei commenti presenti nella tabella commenti attraverso la funzione *contaRecensioni*; il numero recuperato viene controllato e utilizzando in questo modo: se il numero è minore o uguale a 0 viene restituito errore, se è compreso da 0 (escluso) e 4 (incluso) viene creata una matrice di righe pari a numero e colonne pari a 3, se invece numero è maggiore di 4 (escluso) viene creata una matrice di righe pari a 4 e colonne pari a 3.

La matrice così creata conterrà il risultato che la funzione ritornerà.

A questo punto si passa all'interrogazione di *libri_table* e all'estrazione dei dati necessari; vengono selezionati dalla tabella il titolo, l'autore e il nickname di ogni libro recensito; prima di prelevare i valori viene però controllato il ResultSet, e nel caso in cui fosse vuoto viene ritornato errore. Passato questo controllo si passa a popolare la matrice precedentemente creata che verrà ritornata dalla funzione in caso di successo.

Da notare che il nickname non è presente in *libri_table*, ma viene prelevato dalla tabella commenti attraverso un JOIN tra essa e *libri_table*, sfruttando l'ISBN con clausola ON.

La funzione, quindi, ritorna una matrice con valori non nulli in caso di esito positivo, altrimenti una matrice nulla.

uploadAvatar:

```
public int uploadAvatar(String percorso,String image,String username);
```

La funzione realizza l'upload/modifica dell'avatar dell'utente.

Dettagli: la funzione riceve come parametri il percorso su cui dovrà inserire l'immagine, l'immagine codificata e l'username dell'utente.

Da prima prima settato il percorso in cui dovrà inserirsi l'immagine (variabile address).

Si passa poi alla decodifica dell'immagine da stringa a array di byte; viene poi creato il file e in esso l'array che rappresenta l'immagine.

Come ultima istanza si modifica attraverso una query il campo location della tabella *users* in cui è contenuto il path relativo dell'immagine.

La funzione ritorna un intero che sarà uguale a 0 in caso di esito positivo, -3 in caso di errore generico nell'aggiornamento della tabella *users*, -2 in caso di errore generico nella creazione del file immagine e -1 in caso di mancata connessione al database.

votaLibro:

```
public int votaLibro(String ISBN, int voto, String nickname);
```

La funzione concretizza la possibilità di un utente di votare un particolare libro.

Dettagli: la funzione riceve come parametri il nickname, il voto e l'ISBN del libro votato.

Per prima cosa si controlla se l'utente ha già votato il libro; per fare questo si effettua una query sulla tabella voti_libri cercando corrispondenze per quel nick su quel particolare libro; viene così controllato il ResultSet e nel caso in cui contenga tuple viene restituito errore, altrimenti si procede con la funzione.

Se l'utente non aveva votato, si effettua la query nella tabella voti_libri ed inserisce una nuova tupla che rappresenta il voto di quell'utente su quel libro.

Infine si aggiorna la tabella *libri_table* riportando l'ultimo voto inserito: quindi si cerca quel libro e in caso esista si procede ad aggiornare il campo voto della tabella con il nuovo voto appena inserito dall'utente.

La funzione ritorna un intero che sarà uguale ad 1 in caso di esito positivo, -3 in caso di errore generico nell'aggiornamento della tabella *libri_table*, -2 in caso di errore generico nell'aggiornamento della tabella voti_libri, 2 in caso di voto già presente da parte quell'utente e -1 in caso di mancata connessione al database.

Sito WEB

Che linguaggi si è utilizzato? Le pagine quali sono? A cosa servono?

Il sito web è stato sviluppato utilizzando PHP, HTML, CSS e JavaScript.
Sia PHP sia JavaScript ci hanno permesso di generare pagine dinamiche, senza le quali non si sarebbe potuto realizzare nulla.
In particolare,
. PHP: ha svolto il ruolo maggiore; lo abbiamo utilizzando per recuperare dati dalle form, per controllare la login, per presentare differenti layout all'utente a seconda di chi fosse, per richiamare le funzioni Java;
. HTML: è servito per creare i layout e la struttura del sito web;
. CSS: il suo ruole è sempre quello, fornire un design accattivante al noiosissimo HTML;
. JavaScript: è stato in quantità modifica; controlli sulla correttezza dei dati inseriti (così da non appesantire ulteriormente il server), recupero di qualche informazioni altrimenti impossibile da fare con altri strumenti (neanche PHP).

Le pagine web e il loro contenuto

addLibro.php:

E' la pagina che consente ad una libreria di inserire un libro.
Il libro viene aggiunto sia al catalogo - se il libro non era già presente - e alla libreria personale di ogni libreria.

aggiungiRecensione.php:

La pagina permette all'utente - e soltanto ad esso - di inserire recensioni su di un particolare libro.
E' permesso aggiungere una sola recensione per utente.

autoreLibro.php:

La pagina realizza il "profilo" di ogni autore presente all'interno del catalogo.
Sulla pagina sono presenti tutti i libri di quell'autore.

cerca.php:

La pagina permette la ricerca dei libri all'interno del catalogo.

index.php:

E' la pagina principale del sito web. In essa è possibile vedere gli ultimi inserimenti, le ultime recensioni e i libri più votati.
Da essa è possibile iscriversi e loggarsi.

iscrizioneLibreria.php:

E' la pagina che permette di iscriversi al sito web come libreria. E' accessibile soltanto ai visitatori, e non ai loggati (sia come utenti che come librerie).

iscrizioneUser.php:

E' la pagina che permette di iscriversi al sito come utente. E' accessibile soltanto ai visitatori, e non ai loggati (sia come utente che come librerie).

libriLibreria.php:

E' la pagina personale di ogni libreria. Da essa è possibile vedere tutti i libri inseriti nella propria libreria.

E' possibile modificare ogni libro e anche cancellarlo.

E' accessibile soltanto alla librerie loggate.

libro.php:

E' la pagina che contiene tutte le informazioni relative ad un qualsiasi libro presente nel catalogo.

Oltre alle informazioni relative al libro, sono presenti tutte le recensioni rilasciate dagli utenti su quel libro.

Inoltre, sono presentate tutte quelle librerie che vendono tale libro, con sconti e copie.

E' possibile - inoltre - per la libreria correttamente loggata - modificare le informazioni relative al libro, ma solo se quel libro è stato inserito dalla medesima libreria.

La possibilità di rilasciare recensioni è consentita soltanto agli utenti.

E' su questa pagina che gli utenti - e soltanto loro - possono votare il libro.

login.php:

E' la pagina in cui utenti e librerie possono effettuare la login. E' ovviamente accessibili soltanto ai visitatori; infatti utenti e librerie già loggate vengono reindirizzate alla pagina di *index.php*.

La login da parte dell'utente è effettuata con nickname e password, mentre per le librerie con email e password.

modificaAvatar.php:

E' la pagina che permette all'utente di cambiare il proprio avatar. Da qui infatti l'utente può selezionare il suo avatar (presente localmente al macchina che sta utilizzando) e e upparlo per utilizzarlo come immagine personale.

modificaLibro.php:

E' la pagina che consente alla libreria di modificare i dati relativi a sconto e copie di un proprio libro. A questa pagina si accede da *libriLibreria.php*.

modificaLibroCatalogo.php:

E' la pagina che consente alla libreria che ha inserito il libro nel catalogo di modificare i dati. Tutti i dati sono modificabili tranne l'ISBN. La modifica di questo infatti è demandata all'admin, che ha il controllo totale sul database e può valutare correttamente se è il caso di cambiare ISBN o no. Infatti modifiche da parte della libreria potrebbero eliminare riferimenti tra le varie tabelle che compongono il database e quindi compromettere il catalogo.

modificaRecensione.php:

E' la pagina che permette ad ogni utente di modificare la recensione da lui sottomessa di un libro.

E' accessibile dalla pagina personale di ogni utente.

L'utente può modificare il corpo della recensione.

profiloLibreria.php:

E' la pagina che contiene tutti i dati relativi alla registrazione della libreria.

E' ovviamente accessibile soltanto alla libreria.

Oltre a mostrare i dati e anche possibile andarli a modificare; tutti tranne il nome della libreria.

profiloPubblicoLibreria.php:

La pagina mostra tutti i libri - con le relative informazioni - che quella libreria ha nel proprio catalogo personale.

profiloUser.php:

E' la pagina che contiene tutti i dati relativi alla registrazione dell'utente.

E' ovviamente accessibile soltanto all'utente.

Oltre a mostrare i dati e anche possibile andarli a modificare; tutti tranne il nickname.

recensioniUtente.php:

La pagina mostra tutte le recensioni rilasciate dall'utente su i libri presenti nel catalogo.

Oltre ad elencare le recensioni, è anche possibile modificarle e cancellarle singolarmente.

ricercaAvanzata.php:

La pagina consente una ricerca più mirata sui libri presenti all'interno del catalogo.

Sono presenti due tipi di strumenti:

- il primo consente all'utente di specificare qualsiasi campo del libro (titolo e autore e isbn etc); può essere riempito un solo campo, come anche tutti i campi.
- il secondo consente all'utente di specificare un solo campo alla volta (o titolo o autore o isbn etc);

I risultati di entrambi gli strumenti sono riportati in basso, unificando la ricerche.

rimuoviLibro.php:

La pagina consente ad una libreria di rimuovere il libro da lei inserito. La rimozione avviene soltanto da proprio catalogo personale, e non dal catalogo generale. Infatti non è possibile eliminare

il libro dal catalogo. Se si fosse commesso un errore, è necessario contattare gli admin.

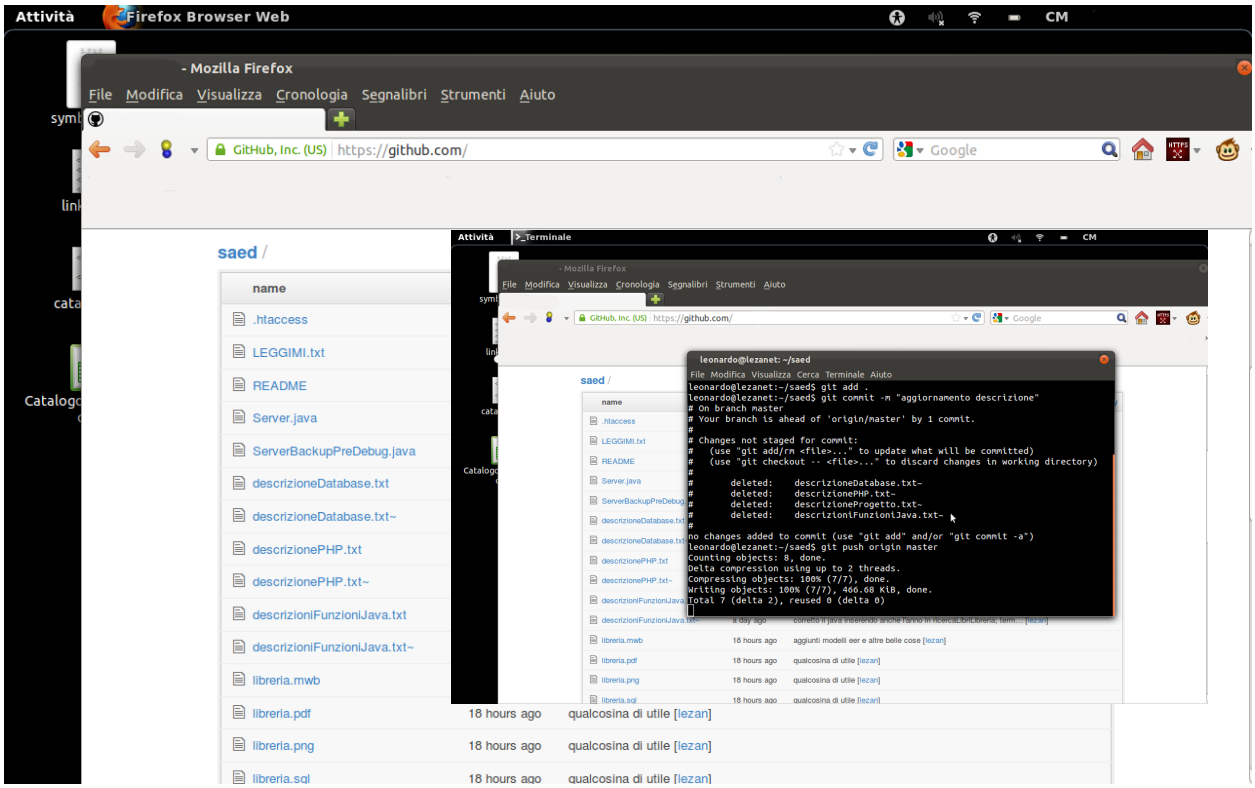
rimuoviRecensione.php:

La pagina consente ad un utente di rimuovere la recensione da lui inserita per un libro.

Collaborazione e Github

Come lavorare in coppia? Come condividere il codice?

Il progetto è stato sviluppato da due persone e quindi è sorto il problema di condividere il codice, senza perdere aggiornamenti e modifiche, perdendo meno tempo possibile.
Inizialmente sono state vagliate diverse ipotesi: svn, ftp etc; ricordando lo sviluppo di pagine web, mentre svn lo sviluppo di veri e propri software.
Alla fine ci siamo affidati a github; non conoscevamo lo strumento, ma lo sviluppo del progetto ci è sembrato un buon modo per capire il funzionamento di questo nuovo servizio.
Github è uno dei tanti servizi che sono nati in seguito alla pubblicazione del software Git.
Git è un sofware di controllo di versione distribuito sviluppato da Torvalds, ed è attualmente il sistema utilizzando dal kernel linux per il controllo versione.
Github è un servizio web di hosting che appunto utilizza Git. C'è chi lo chiama social-code, cioè un social network per programmatori.
Esiste un plugin da integrare su Eclipse, così da velocizzare ancora di più lo scambio di codice, ma abbiamo deciso di evitarlo ed affidarci direttamente a Git da riga di comando. Perché? Perché ci piace complicarci la vita con il terminale; perché dovevamo condividere altro codice che non veniva sviluppato su Eclipse.

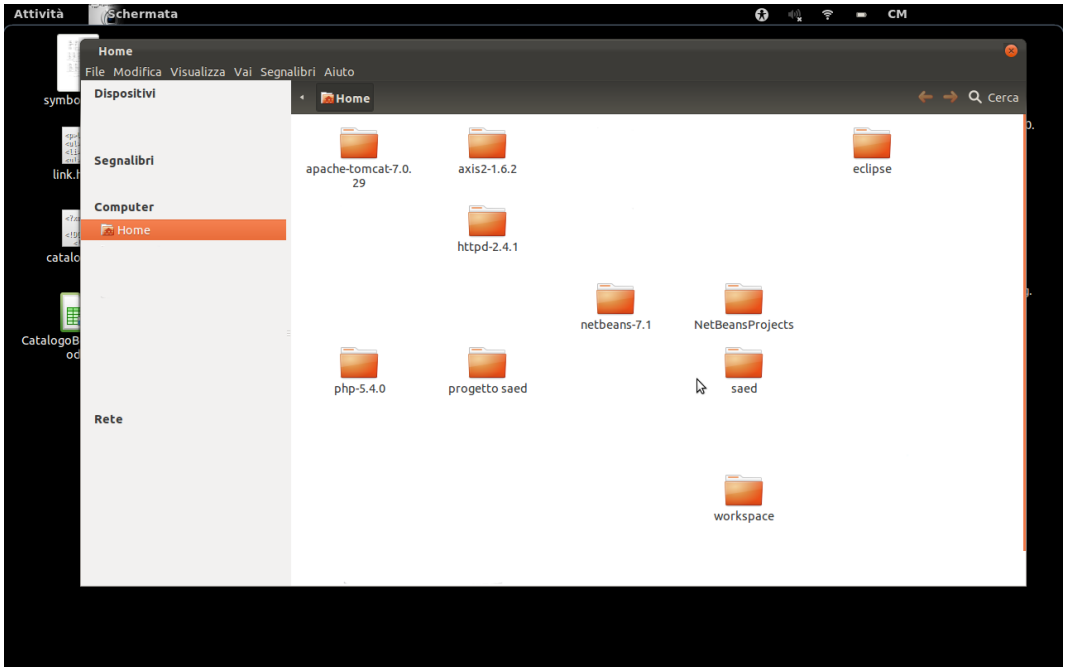


La pagina del progetto Github e il software Git all'opera.

Su che sistema operativo è stato sviluppato il codice?

Il codice (sia Java, che PHP che HTML) è stato sviluppato su differenti sistemi operativi. Poi,

avendo lavorato in due sul progetto, ancora di più.
Il codice è stato testato su:
- Ubuntu 11.10;
- Ubuntu 12.04;
- Fedora 16;
- Windows XP, SP3;
- Windows 7;



Le cartelle di installazione degli strumenti utilizzati per lo sviluppo del progetto.

Abbiamo però deciso di presentare il progetto su sistemi GNU/Linux.

Chi ha lavorato sul progetto?

Sul progetto abbiamo lavorato in due.
Ci siamo divisi funzioni Java e pagine PHP.
Ad ogni nuovo commit si procedeva al test del nuovo codice, cercando di commettere quante più cavolate un utente medio può commettere durante la sua navigazione.

QUA CI METTIAMO LE FOTO NOSTRE. BASUHAUSHAS.

Debugging

Come si è debuggato il codice?

Il debug non ha portato via più di tanto tempo, ma sicuramente il codice conterrà ancora diversi bug.
L'utilizzo dell'IDE - in particolare di Eclipse - ha permesso, nei casi di più estremi, di ricorrere all'avvio del server Tomcat in modalità debug, così da scovare l'errore che si celava all'interno dell'albero. Uno strumento veramente utile che fortunatamente è stato utilizzato poche volte.
Sicuramente i misteri più grandi sono stati sollevati da CSS e PHP che in maniera piuttosto randomica - almeno ai nostri occhi - producevano malfunzionamenti davvero esilaranti.