Area Informatica

Documentazione del progetto per l'esame di “Programmazione per il web”.

Sviluppato dal Gruppo ??, composto da:

- Antonio Daniele Gialluisi [???]
  (Progettazione lato server, ideazione)

- Alii [???]
  (Progettazione lato client, ideazione)

INDICE

Ideazione e motivi dello sviluppo

Il progetto qui documentato si propone di fornire una futura piattaforma di e-learning per un'Università.

La filosofia dietro questo progetto, è permettere agli istituti, non solo di sfruttare le nuove tecnologie in maniera intelligente, ma anche di migliorare il loro arsenale di strumenti, migliorando così, anche le tecniche d'apprendimento, incitando alla condivisione libera del sapere (soprattutto tramite video lezioni e libri con licenza libera), permettendo così anche a persone che non possono accedere facilmente agli insegnamenti (problemi economici e/o disabilità), un modo per poter migliorare le loro conoscenze.

Ecco una panoramica delle funzioni supportate dalla piattaforma:

- Gestione degli utenti, tra utenti standard e amministratori.
- Un blog per le comunicazioni e le news, sia utenti standard che amministratori possono scriverci.
- Contenuti visibili anche ai visitatori.
- Gli utenti standard e gli amministratori possono scrivere e modificare contenuti.
- I contenuti sono articoli a cui possono essere allegati vari tipi di documenti.
- Gli amministratori possono creare e modificare materie.

Progettazione del lato server

Il lato server di questa piattaforma è stato sviluppato utilizzando due tecnologie in particolare, ovvero PHP (gestione delle pagine), e il Database Management System MySQL (gestione dei dati).
A questi due si aggiunge, naturalmente, uno scheletro del sito scritto nel solo HTML5.

Creazione del Database

Il nome del database è StudyPlatform ed è composto dalle seguenti tabelle:

- User

  Tabella usata per memorizzare gli utenti iscritti. Possiede i classici attributi personali quali nome, cognome, nickname e password. In origine si era pensato di immagazzinare in questa tabella anche degli attributi riguardanti l'attivazione dell'utente (quel meccanismo in cui viene mandata un'e-mail all'indirizzo registrato cosìcche l'utente confermi di essere stato proprio lui ad effettuare l'operazione di iscrizione). Per tempo e semplicità, tali meccanismi non sono stati implementati, ma sarebbe una buona idea per un'implementazione futura.

- Article

  Tabella usata per memorizzare tutti gli articoli scritti fin'ora.
  Un articolo ha una data di pubblicazione, un tipo (se lezione o news), un titolo, un contenuto, e l'autore.
  Inizialmente, questa tabella ha causato un po' di problemi a livello di progettazione. Durante la progettazione logica infatti, c'è stata un'indecisione su come strutturare le informazioni relative agli articoli.
  Sappiamo infatti che le news e le lezioni sono articoli, ma le news sono diverse dalle lezioni.
  La questione poteva essere risolta solo in questi due modi:

  - Mantenere l'entità padre “Articolo” che avrebbe distinto tra i due tipi grazie ad un attributo.
    Il vantaggio di questo approccio è l'uso di UNA SOLA tabella per gestire gli articoli, utile per risparmiare spazio, ma un po' oneroso durante la ricerca, a causa della presenza di TUTTI gli articoli immagazzinati, a prescindere dal tipo.
    Questa è stata la scelta implementativa finale.

  - Mantenere due entità separate: Lezioni e News.
    Il vantaggio di questo approccio è, oltre alla netta distinzione tra i tipi d'articolo, la ricerca più veloce poiché si ricercano le informazioni in due tabelle diverse, quindi ignorando completamente articoli che non sono di un determinato tipo. Lo svantaggio sta nel maggior spazio occupato dovuto a una duplicazione delle tabelle, poiché ambedue, hanno gli stessi attributi.

  Voglio far notare, che il contenuto di un articolo nell'attuale implementazione, è memorizzato nel database stesso.
  In realtà, in futuro ci si potrebbe adeguare a come fanno gran parte dei siti web odierni, ovvero conservare nel database solo un riferimento ad un file presente nelle cartelle del server, questo file appunto è il contenuto dell'articolo.

Gli ovvi vantaggi di questo approccio stanno in primo luogo nella minor quantità di informazioni registrate nel database (un percorso ad un file occuperà sempre molto meno di un intero contenuto), e di conseguenza anche durante le ricerche il DBMS risulterà molto reattivo.

- Comment
  Tabella usata per memorizzare i commenti.
  Attualmente è prevista solo la memorizzazione del contenuto del commento e dell'autore, non è prevista alcuna memorizzazione della data, anche qui, implementazione futura fattibile.

- Tag
  Tabella usata per memorizzare tutti i tag creati fin'ora.

- ArticleTags
  Tabella creata per l'associazione dei tag ai rispettivi articoli. Con questa tabella è possibile stabilire la relazione molti a molti tra articoli e etichette.

- Subject
  Tabella usata per memorizzare tutte le materie create fin'ora.

- Lessons
  Tabella usata per memorizzare tutte le lezioni scritte fin'ora. Tutto quel che è stato detto nella tabella Article ha portato poi alla creazione di questa tabella.
  La scelta progettuale per la tabella Article, ha portato alla conclusione che una lezione non è l'equivalente dell'articolo, ma che una lezione CONTIENE un articolo.
  Questa tabella ha infatti come attributi un riferimento ad un articolo, un numero di lezione e una materia di appartenenza.

- News
  In realtà è una vista (view), utilizzata per tenere traccia di articoli che NON sono contenuti in alcuna lezione.
  Questi sono destinati al blog della piattaforma.

Un'altra cosa molto importante nella gestione dei DBMS è evitare che un utente malintenzionato riesca ad avere più privilegi del necessario.
Per questo motivo, è stato creato un utente apposta per questa piattaforma che ha permessi di accesso, lettura e scrittura SOLO sul database StudyPlatform, tale utente è “student@localhost”.

Implementazione delle pagine

Come già menzionato, tutte le pagine del sito sfruttano le potenzialità del PHP per mostrare il contenuto.
Le pagine sono state divise in base all'estensione dei loro file, per una miglior compresione del loro scopo:

- File con estensione “.php”:
  Vengono usati per effettuare operazioni, definire classi, funzioni, ergo, sono file che contengono SOLO codice PHP (eccezione a questa regola sono l'indice del sito e l'header).
  I file che iniziano con action_* sono file adibiti ad operazioni effettuate dagli utenti (quindi una sorta di “middleware”, questi codici prelevano, validano i dati ed effettuano le operazioni).

- File con estensione “.html.php”:
  Questi al contrario dei primi, contengono HTML5 e PHP, e sono le vere e proprie pagine web, cioè, quelle che appaiono ai client.

Nel server la piattaforma è divisa nelle seguenti cartelle:

- /articles/
  Qui sono presenti tutti i file inerenti gli articoli, editor inclusi.

- /avatars/
  Dove sono situati gli avatar degli utenti, raggruppati per id utente. All'inizio c'è un solo file di placeholder.

- /base_classes/
  Qui sono situate le classi base, senza le quali questa piattaforma non funzionerebbe.

- /other/
  Qui risiedono altri file utili al sito, che però, non hanno una categoria particolare.

- /subjects/
  Qui risiedono tutti i file utili per la creazione e modifica delle materie.

- /uploads/
  Qui sono contenuti tutti i file caricati sulla piattaforma.

- /users/
  Qui sono situati tutti i file utili alla gestione degli utenti.

Comunicazione tra server e database

Sia nelle pagine web che nelle pagine operative, è necessario poter effettuare interfacciarsi al database (prelevare informazioni e mostrarle, prendere in input dei dati dai form e immagazinarli...).
In questa piattaforma, tutte le comunicazioni tra server e database sono state gestite da un'apposita classe PHP creata per tale scopo: DatabaseHandler.

Per gestire le connessioni, DatabaseHandler utilizza una classe nativa presente in PHP fin dalla versione 5.1: PDO (PHP Data Object). Implementazioni precedenti usavano dapprima le funzioni mysql_* (in seguito mysqli_*), poi, rimosse completamente in favore di PDO, poiché, tramite appositi driver, è possibile interfacciarsi a più di un DBMS, usando praticamente sempre la stessa API (cosa non vera invece per le funzioni mysql_* e mysqli_*, che funzionano SOLO con MySQL).

In caso di errore, DatabaseHandler può sollevare una tra le due seguenti eccezioni create appositamente:

- DBConnect_Exception:
  In caso di fallimento durante la connessione al database

- DBQuery_Exception:
  In caso di errore nelle query o nelle transazioni

A livello implementativo, le query per il database vengono eseguite in maniera grezza, ovvero, direttamente nelle pagine web e nei file operativi. Una miglioria futura al progetto, potrebbe essere quella di creare tante specializzazioni di DatabaseHandler. Ognuna di esse gestisce le informazioni di una parte di sito, fornendo delle API da usare poi, nelle pagine e nei file operativi. Nella pratica, tutte le query attualmente presenti nelle pagine, verrebbero sostituite da metodi delle specializzazioni di DatabaseHandler, migliorando così, l'information hiding sulle query utilizzate.
Oltre all'ovvio vantaggio di diventare un'implementazione più object oriented, il codice diventa più leggibile, ed eventuali modifiche alle query, risultano più facili e meno pericolose da effettuare, poiché raggruppate in file dedicati e non all'interno del codice operativo.

Memorizzazione e verifica dei dati

La prima forma di gestione dati che è stata incontrata durante lo sviluppo, è stato il raccoglimento dei dati dai form.

Come ben si sa però, è necessario non solo prelevarli, ma anche verificare la loro consistenza (insomma, validarli, non avrebbe senso per esempio, accettare come nome una stringa vuota).
Nonostante tecnologie come Javascript, o lo stesso HTML5, permettano dei controlli a livello di inserimento, esse sono facilmente by-passabili (basti pensare che Javascript è disattivabile), motivo per cui, è imperativo effettuare controlli server-side.
Al controllo di integrità però, si aggiunge ben presto la necessità di memorizzare tali dati.

Si pensi ad esempio, all'azione di login alla piattaforma: i dati vengono verificati, l'utente effettua il login, ma adesso, si rende necessario un meccanismo per MANTENERE tali dati, perché l'utente possa continuare ad usare il sito.
Questa piattaforma, utilizza ben tre classi per la gestione dei dati, una classe base e due specializzazioni.
La classe base è CustomSessionHandler. Essa sfrutta quelle che in PHP vengono chiamate “Sessioni”, ovvero, dei cookie memorizzati su server (anzichè su client, come i cookie tradizionali), aumentando così, la sicurezza del sito.
CustomSessionHandler ha un'API molto limitata, implementata tramite metodi getter e setter (utili per information hiding) che effettuano le operazioni basilari utili alla piattaforma:

- Apertura dei file di sessione
- Memorizzazione, recupero e cancellazione sul file di sessione:
    - Dei dati dell'utente
    - Delle stringhe d'errore
- Rigenerazione dell'id del file di sessione

L'ultimo punto, è molto importante a livello di sicurezza, ogni file di sessione ha un id, che il server utilizza per comunicare con i singoli client. Se l'id rimane immutato, un potenziale attaccante può intercettare un client che effettua un'operazione di login, ottenere l'id di sessione, e spacciarsi per quel client.
Questo attacco viene chiamato “Session fixation attack”, ovvero, un attacco basato sul fatto che l'id di sessione è statico (Fisso come dice il nome).

Per ridurre questo tipo di attacchi, non solo ad ogni istanziazione di CustomSessionHandler (e derivati) l'id viene immediatamente rigenerato, ma in più, viene memorizzato un contatore, che raggiunta una certa cifra, ri-genera ancora una volta l'id.
Anche se un attaccante riuscisse a impadronirsi di un id di sessione, esso potrebbe già non essere più valido, poiché rigenerato, vanificando così la possibilità di attacco.
Come già descritto, CustomSessionHandler si occupa della memorizzazione degli errori, e dei dati utente, utili per il login.
Tuttavia, non v'è traccia di metodi per la memorizzazione dei dati di form, quindi questa classe, ancora non è completa.

A causa dei form, è stata implementata una specializzazione, che oltre ad ereditare le capacità di CustomSessionHandler, ne introduce altre, in grado, non solo di memorizzare i dati presenti nei form, ma anche di validare tali dati: FormSessionHandler.
FormSessionHandler introduce un'API più completa contenente funzioni di memorizzazione, recupero e cancellazione dei dati di form (come per CustomSessionHandler, implementati tramite getter e setter). In pratica, tutti i dati richiesti nei form della piattaforma, hanno un loro posto nei file di sessione.
I vantaggi di ciò, oltre alla già discussa possibilità di constatare l'integrità dei dati inseriti, sono anche quelli che, in caso di errore nella digitazione, è possibile correggere il solo parametro errato, senza per questo dover riscrivere tutti gli altri (si pensi di avere impostato durante l'iscrizione, una data di nascita non valida, ovviamente, la data è errata, ma non gli altri parametri, essi rimarranno disponibili e non cancellati).
In caso di errore durante la validazione, FormSessionHandler solleva l'eccezione InvalidInsertedDataException, utile, una volta “catturata”, per comprendere quali siano stati gli errori riscontrati, e così permettere di correggerli.
L'unico grande difetto a livello implementativo, è il fatto che questa sola classe gestisce TUTTE le variabili presenti nei form, questo porta ad un'eccessiva centralizzazione, che potrebbe non essere ideale durante la manutenzione.

Una possibile miglioria, potrebbe essere quella di decentralizzare la classe, delegando la validazione e la memorizzazione di alcuni dati di form, a classi derivate da questa, in maniera tale, da gestire tutti i dati in maniera più semplice e adatta alla manutenzione.
Un ultimo problema riguardante i dati, si è presentato quando è iniziato lo sviluppo dell'editor degli articoli. Si sa che ad un articolo possono essere associati dei tag (o etichette), per una miglior categorizzazione dei contenuti. Come ricordare quali tag voglio associare all'articolo?
L'idea più ovvia, è stata quella di usare lo stesso meccanismo adottato per i form, quindi sfruttare i file di sessione.

Da questo, è nata la necessità di creare una classe derivata da FormSessionHandler, che gestisse i soli tag, sempre però come dati di form: PostSessionHandler.
Usata solo nei file riguardanti l'editing degli articoli, questa classe implementa delle funzioni di validazione per i nuovi tag (che poi vengono inseriti nel database), e di aggiunta/rimozione di un tag dall'articolo corrente (che rimane solo nel file di sessione).
A livello implementativo, PostSessionHandler non è molto diverso dalle classi genitrici, anche questa classe infatti, mette a disposizione dei metodi getter e setter, in grado di gestire in maniera più o meno semplice, i tag durante l'editing di un articolo.

Caricamento dei file sulla piattaforma

Gran parte dei siti, permettono il caricamento di file su server, per aumentare i contenuti del sito stesso, e questa piattaforma non fa eccezione.
La classe utilizzata per tale scopo è UploadHandler.
Sebbene questa classe, sia stata creata con l'intento di essere un uploader generico, in questa piattaforma, viene utilizzata solo in due occasioni:

- Durante la modifica del profilo dell'utente, per caricare immagini da usare come avatar.
- Durante la creazione/modifica di un articolo, per caricare ulteriore contenuto, da usare in seguito o nell'articolo stesso.

Una volta effettuata l'azione di upload, tramite un form in particolare (=che possieda gli attributi “enctype='multipart/form-data'”, “input type='file' e in particolare per questa classe “name='content_file'”), che permetta l'upload, questa classe si occupa di gestire il file appena caricato, con i seguenti passi:

- Decidere se immagazzinare il file sulle cartelle del server:
  Per limitare il problema dei file dannosi, si utilizzano funzioni native di PHP, che permettono l'analisi del MIME (Multipurpose Internet Mail Extensions).
  È possibile infatti, limitare i tipi di file ammessi impostando una whitelist, con i MIME dei file che ci si aspetta di caricare.

- Spostare il file in una cartella del server decisa a priori:
  Solitamente durante l'instanziazione della classe, anche se è possibile modificare il percorso anche in seguito.

- Opzionalmente, rinominare il file:
  È possibile rinominare il file, impostando il nuovo nome in anticipo.

- In caso di errori, solleva l'eccezione UploadFailureException:
  Naturalmente con un messaggio, che permette di scovare l'errore.

Purtroppo, il sistema di gestione dei file caricati di questa piattaforma, non è molto sviluppato, infatti, allo stato attuale, permette solo di caricare dei file in certe cartelle del server, ma dopo l'upload non c'è alcun meccanismo di gestione dei file (ridenominazione, cancellazione, spostamento, permessi d'uso), che invece potrebbe essere utile a livello amministrativo, magari per catalogare i documenti.

Gestione delle lezioni

Il cuore di questa piattaforma, trattandosi di un e-learning, sono le lezioni.

Come già spiegato nella sezione “Creazione del database”, le lezioni, sono articoli associati ad una materia.
Vediamo ora in dettaglio, la gestione di queste due componenti:

Le materie

Una materia ha un nome, una descrizione, e zero o più lezioni.
Tutti i file operativi per la gestione delle materie, come già accennato nella sezione “Progettazione del lato server”, si trovano nella cartella “/subjects/”.
All'interno di questa cartella, sono presenti diversi file e una sottocartella.

Il primo file è “action_subject_content.php” che, come suggerisce il nome, si occupa di registrare su database, le modifiche effettuate tramite operazioni di creazione/modifca/rimozione delle materie. Naturalmente, essendo una pagina che modifica in maniera permanente dei dati, è necessario che possa operare in maniera sicura, motivo per cui sono presenti dei controlli. Il primo controllo, riguarda gli utenti, infatti esso assicura, che solo gli utenti che hanno eseguito l'accesso alla piattaforma, e che sono amministratori, possono effettuare, un'operazione di salvataggio su database. In caso contrario, gli utenti che hanno provato ad usare questa pagina, vengono rimandati alla pagina indice.
Il secondo controllo, riguarda i dati spediti alla pagina con metodo GET e POST, in pratica, si verifica che si siano mandati tutti i dati necessari per effettuare delle operazioni, in caso contrario, di nuovo, l'utente viene rimandato alla pagina indice.

Il secondo file è “form_subject.html.php”. A differenza del primo, questo è una pagina vera e propria, infatti, è la pagina che serve per creare/modificare materie. Essa contiene un form, che poi rimanda al file descritto prima. Faccio notare che, anche questa pagina, rimanda all'indice, in caso di utenti non amministratori o non “loggati”.

Il terzo file è “subject_remove_dialog.html.php”. Anche questo, pagina vera e propria, e con accesso consentito ai soli amministratori, serve per mostrare un dialog, che richiede se confermare la cancellazione di una materia.
In realtà, questa pagina esiste per il semplice fatto che lo scheletro della piattaforma non prevedeva linguaggi client-side, quindi, fingendo che non esistessero, l'unica maniera, era quella di creare una pagina supplementare. La miglior scelta, attualmente, e implementabile in futuro, sarebbe quella, di usare proprio quei linguaggi client-side (tipo Javascript) per eliminare questa pagina. Infatti, il tutto verrebbe implementato nel form, cioè nello stesso “form_subject.html.php”, che è sicuramente più comodo, dato che si avrà un dialog di conferma nella stessa pagina, anzichè doverne caricare un'altra.

Il quarto e ultimo file è “subjects_list_page.html.php”. Lo scopo di questa pagina, è mostrare le materie disponibili attualmente nella piattaforma, per permettere agli utenti, di accedere alle singole lezioni.
A differenza delle precedenti, questa pagina è visitabile da tutti, ma a seconda del tipo d'utente, ci saranno più o meno differenze.
Gli amministratori, a differenza degli utenti comuni, e dei visitatori, avranno per ogni materia, i pulsanti “Modifica” ed “Elimina”.
Questi due pulsanti rimanderanno alle pagine citate in precedenza.

Le lezioni

Si è fatto prima accenno, ad una sottocartella presente in “/subjects/”, chiamata “lessons”. In essa, si trovano i file che gesticono le lezioni (Attenzione, le **lezioni**, non gli *articoli*!).
Questa cartella contiene due file, il primo, denominato “lessons_page.html.php” è una pagina che mostra la lista delle lezioni di una data materia (impostata tramite parametro GET).
Le lezioni, vengono mostrate per numero crescente, e con un collegamento ipertestuale che, una volta cliccato, ne permette la lettura dell'articolo associato.

Poiché i meccanismi di lettura delle lezioni, sono pressoché identici a quelli di lettura delle news, essi verranno discussi nella sezione “Gestione del blog e creazione di contenuti” più avanti.

Anche in questo file (come “subjects_list_page.html.php”), vengono mostrati ulteriori pulsanti, in base al tipo di utente che visualizza la pagina.

- Nessun pulsante per i visitatori.
- Il pulsante “Modifica” per gli autori della lezione (Ricordo, che le lezioni possono essere create, anche da utenti standard, non necessariamente amministratori).
- Oltre al pulsante “Modifica”, vengono mostrati agli amministratori, i pulsanti “Elimina”, “Sposta giù” e “Sposta su”.

I pulsanti “Sposta giù” e “Sposta su”, sono pulsanti creati con uno scopo ben preciso: quello di poter cambiare eventualmente il numero di una certa lezione.

Ogni lezione, ha un attributo che la identifica all'interno della materia, ovvero il numero di lezione (nel database “nlesson”), che nella pratica, altro non è che, un metodo d'ordinamento, la lezione con attributo “nlesson = 1” sarà la prima, con “nlesson = 2” sarà la seconda, con “nlesson = n” sarà l'n-esima, e così via.
Le lezioni appena create, hanno sempre un numero di lezione pari a max(numero di lezioni presenti nella materia) + 1 (se in una materia sono presenti 5 lezioni, la nuova lezione sarà la 6°).
È comprensibile però, che non necessariamente, una nuova lezione corrisponde all'ultima lezione della materia (supponiamo che per mancanza di materiale, non è possibile creare una certa lezione al momento, un utente non deve per forza aspettare, deve poter tranquillamente lavorare alle altre lezioni, e poi, quando riceve il materiale, lavorare su questa, e ordinarla nel posto giusto).

Le operazioni qui sopra descritte, a livello realizzativo, sono state molto difficili da gestire.
La necessità era, che “nlesson” fosse un valore compreso tra 1 e max(numero di lezioni presenti nella materia), che potesse essere incrementato in caso di inserimento di un nuovo record, e soprattutto, che il valore della colonna “nlesson” di una lezione potesse essere scambiato con il valore della colonna “nlesson” di un'altra lezione.

Indagando su questa necessità, sembrerebbe che un'operazione di questo tipo, in maniera semplice e intuitiva, sia impossibile.

In una pagina Internet, dove si chiedeva aiuto, per una situazione molto simile (la richiesta era: scambio di valori tra colonne di record diversi), è stata riportata una soluzione algoritmica, adottata poi in questo progetto. La funzionalità, è stata implementata tramite una transazione, che effettua le operazioni qui sotto elencate.
Faccio notare che, “lezione corrente” è il valore “nlesson” della lezione che abbiam deciso di spostare, e che, “lezione seguente” è il valore “nlesson” della lezione precedente/successiva a quella da spostare (precedente se spostiamo in alto, successiva se spostiamo in basso).

- Impostare “lezione seguente” a (-1).
- Incrementare/descrementare “lezione corrente” di 1.
- Impostare ”lezione seguente” a (“lezione corrente” - 1).

Gestione del blog e creazione dei contenuti

A differenza delle lezioni, per gestire news bastano solo gli articoli.
A livello di database, come già accennato, viene usata la vista “News”, per filtrare gli articoli, e selezionare solo quelli che hanno l'attributo “type” con valore “news”, ovvero, articoli che sono effettivamente news.
In questa piattaforma, per news, si intendono tutti quegli articoli, che non sono inclusi in alcuna lezione, al contrario, sono destinati ad essere pubblicati, sul blog della piattaforma.
Il blog serve a pubblicare notizie (ovvero articoli, nel blogging vengono chiamati post).

Il file che fa da home per il blog, si trova nella cartella “/other/” e si chiama “blogindex.html.php”.
In questa pagina, vengono mostrati gli ultimi 5 post (in base alla data di pubblicazione, dal più recente al più vecchio), due hyperlink per vedere vecchi/nuovi post, rispetto alla pagina attuale (mostrati in grassetto, nel caso in cui, non ci siano più post da vedere), la lista di tutti i tag, un form per la ricerca, e un archivio, adatto per cercare i post, in base alla data di pubblicazione.

Il form di ricerca si spiega da solo: permette la ricerca di una stringa arbitraria, producendo come risultato, i post nei quali è stata trovata.
La ricerca, viene effettuata nel titolo, nel post vero e proprio (contenuto dell'articolo) e nei tag.
Si potrà notare, che i tag, vengono mostrati come collegamenti ipertestuali. Cliccando su uno di essi, è possibile mostrare tutti i post, che hanno in associazione quel tag. L'implementazione attuale, consente di filtrare post, in base a un solo tag, purtroppo, in futuro si potrebbe migliorare questo filtro, permettendo di selezionare più di un tag per la ricerca.
L'archivio invece, è un resoconto di tutti i post, pubblicati fin'ora, catalogati per anno e mese. Vengono mostrati i titoli dei post, come collegamenti ipertestuali, e anche qui, cliccandoci, è possibile procedere alla lettura di tali post.
Per ogni post nella home, viene mostrato: Il titolo, i primi 300 caratteri del contenuto, e un collegamento ipertestuale che ne permette la lettura.

In caso l'utente, abbia effettuato l'accesso, per ogni post creato da lui, ci sarà un pulsante “Modifica”, che appunto, gli permetterà di modificare i contenuti. In caso l'utente, sia un amministratore, vedrà il pulsante “Modifica” ed “Elimina” su tutti i post.

Gli articoli, in questa piattaforma, vengono gestiti dai file presenti nella cartella “/articles/” (tra cui, il già citato “PostSessionHandler.php”, che permette, durante l'editing, di mantere un riferimento ai tag che si intende associare all'articolo).
Uno dei file più importanti è “post_page.html.php”, il file in grado di mostrare i contenuti del sito.
Questo, è il file a cui si viene rimandati, quando si usano i collegamenti ipertestuali per leggere le news o le lezioni.
La prima cosa che la pagina mostra, sono due link denominati “Articolo precedente”, e “Articolo successivo”, utili, per leggere altri articoli.
Faccio notare, che se si sta visualizzando l'articolo di una lezione, il precedente/successivo altro non è, che l'articolo della lezione precedente/successiva a quella che si sta leggendo.
Se invece si legge una news, l'articolo sarà precedente/successivo, in base alla data di pubblicazione della news che si sta leggendo.
Oltre ai link, vengono mostrati: Il titolo dell'articolo, il contenuto, l'autore (come collegamento ipertestuale, cliccandoci, è possibile visualizzare le informazioni sull'utente), i tag associati, e, se c'è ne sono, i commenti.
In caso un utente “loggato”, legga un articolo, sarà anche in grado di pubblicare commenti, grazie ad un apposito form, o, se sono presenti altri suoi commenti, di poterli cancellare (per ogni commento scritto da quell'utente, apparirà un pulsante “Elimina”). Se tale utente, è amministratore, tutti i commenti avranno il pulsante “Elimina”.
Per una qualsiasi operazione legata ai commenti, viene invocato il file operativo “action_comment.php”, che, come suggerisce il nome, si occupa di gestire tutte le operazioni relative ai commenti.

Un altro file presente è “post_remove_dialog.html.php”, che, al pari di “subject_remove_dialog.html.php”, altro non è, che una schermata per chiedere conferma dell'eliminazione di un articolo.

Si viene rimandati a questo file, sia quando si intende cancellare una news (in questo caso, a livello di database, verrà eliminato il record relativo all'articolo, e tutti i record dei tag associati), sia quando si intende cancellare una lezione (in tal caso, oltre all'articolo e ai tag, verrà eliminato anche il record della lezione). Anche qui, come il suo predecessore, potrebbe essere migliorabile, tramite un linguaggio client-side, che chieda conferma nella stessa pagina, anziché rimandare ad un'altra.
Gli ultimi file di cui discutere, sono l'editor dei post, ovvero, la pagina che permette di immettere dati per la creazione degli articoli (lezioni e news), e il file operativo, che si occupa di effettuare le operazioni necessarie sugli articoli.
I file sono “form_post_editor.html.php”, e “action_post_content.php”.
L'editor, rimanda al file “action_post_content.php”, al momento di effettuare le operazioni, ed utilizza un form diviso nelle seguenti sezioni:

- Un selettore di materie (appare solo se si crea una nuova lezione), serve ad impostare la materia di destinazione dell'articolo.

- Un campo di testo, il contenuto sarà il titolo da assegnare all'articolo.

- Un box di testo per il contenuto dell'articolo:
  In caso Javascript sia abilitato, questo box, sfrutta un editor open source, rilasciato sotto licenza GPL, chiamato CKEditor (presente nella cartella ckeditor all'interno di “/articles/”).

- Una sezione dedicata ai tag:
  La prima parte, mostra tutti i tag associati all'articolo con l'opzione “Scarta”, per appunto, scartare tale associazione. Più sotto, ci sono i tag disponibili (ovvero, immagazzinati nel database) con le opzioni “Associa” ed “Elimina”, che servono, rispettivamente, per associare il tag all'articolo, e per eliminare il tag dal database (se tale tag, era associato all'articolo, viene rimossa anche l'associazione). L'ultima parte invece, consiste in un campo di testo e del pulsante “Salva”, che permette di creare nuovi tag, e memorizzarli nel database.

- Una sezione dedicata al caricamento dei file:
  Dapprima, vengono mostrati, come collegamenti ipertestuali, i nomi dei file presenti sul server nella cartella “/uploads/”.
  Questa soluzione in realtà, è molto basilare, e permette solo, di ottenere un collegamento diretto ai file, per poterli includere, come riferimento, nel contenuto dell'articolo.
  In futuro, si potrebbe creare un meccanismo come quello già descritto nella sezione “Caricamento di file sulla piattaforma”, magari sfruttando, oltre a PHP, anche un linguaggio client-side come Javascript.
  Infine, ci sono due pulsanti, il primo, per mostrare il file da caricare, e il secondo per effettuare l'upload.

- Un pulsante che pubblica l'articolo:

Non c'è molto da dire, sul file “action_post_content.php”. A livello di operazioni e controlli, è molto simile agli altri file action_*, la differenza sta solo in ciò che gestisce, che, in questo caso, si tratta di articoli.
Per la gestione dei tag, viene adoperata la classe PostSessionHandler già nominata prima, e per il caricamento di file, la classe UploadHandler.

Gestione degli utenti

Così come non è possibile, concepire una piattaforma senza contenuto, non è possibile, concepire il contenuto, senza utenti che lo creino.
I file per la gestione degli utenti, si trovano, come già detto nella sezione “Implementazione delle pagine”, nella cartella “/users/” del server, tuttavia, un'ulteriore funzionalità si trova invece in un file presente nella cartella “/other/”, ma di questo, se ne discuterà in seguito.
In tutto, sono presenti quattro pagine, e un file operativo. Le prime due, sono form, usati rispettivamente, durante il login (“form_login.html.php”) e durante l'iscrizione (“form_subscribe.html.php”).
Ambedue, rimandano al file operativo, al momento di effettuare le operazioni. Tali pagine, hanno un controllo, che non permette l'accesso ad utenti già “loggati”, redirezionando questi ultimi, all'indice del sito.

Le password in questa piattaforma, sono registrate tramite hash.
In realtà, il solo hashing della password non è molto sicuro, poiché l'hash, è sempre lo stesso.
L'approccio qui adottato invece, è quello di creare, una stringa casuale da accorpare alla password (la cosidetta “salt”), proprio per avere hash sempre diversi.
Durante il login, viene creato un hash della password immessa, usando come “salt” l'hash disponibile nel database. Se l'hash risultante e l'hash usato come “salt” sono identici, allora la password è corretta.
Appena effettuato l'accesso, viene rigenerato un nuovo hash e immagazzinato nel database.

Il terzo file, è una pagina adibita alla visualizzazione dei profili (“profile_page.html.php”). Immetendo un id utente, tramite metodo GET, è possibile ottenere tutte le informazioni di quell'utente.
In caso lo stesso utente, veda tramite questa pagina, le sue informazioni, appariranno ulteriori pulsanti.
In caso l'utente, NON sia amministratore, verrà mostrato, un primo pulsante con su scritto “Promuovi”, e una volta premuto, l'utente verrà appunto “promosso”, ad amministratore.
Naturalmente, quest'implementazione è volutamente banale, poiché è un progetto per un esame universitario, non andrà mai online.
Nella realtà, un'implementazione più realistica, potrebbe essere il rimando di quel pulsante, ad un form di richiesta agli amministratori, per appunto “richiedere” tale promozione. Ad ogni richiesta, gli amministratori decidono, magari in base all'attività svolta dall'utente, fino ad allora, che cosa fare, se promuoverlo oppure no.
Il secondo pulsante invece, rimanda l'utente all'ultimo form presente, ovvero, una pagina che permette di modificare password e descrizione (“form_profile.html.php”).
Inutile dire, che anche questo form, rimanda al file operativo per effettuare le operazioni.
Il file operativo, si chiama “action_users.php” e come potrete capire dal nome, effettua tutte le azioni riguardanti la memorizzazione/recupero delle informazioni sugli utenti.
Rispetto ad altri file trattati in precedenza, non c'è molto da dire su questo.
È il secondo file, ad utilizzare la classe UploadHandler citata nella sezione “Caricamento dei file sulla piattaforma”.

In questo contesto, viene utilizzata per caricare esclusivamente immagini, usate poi, come avatar per gli utenti.
Una volta effettuato l'upload, l'immagine viene rinominata, con l'id dell'utente e depositata nella cartella “/avatars/” della piattaforma.
La password dell'utente, viene modificata solo nel caso in cui almeno uno dei tre form (“password corrente”, “password nuova” o “ripeti password nuova”), contengono caratteri. Se ciò succede, dapprima, vengono validate le password per assicurarsi che rispettino le regole impostate (per le password, sono necessari almeno 8 caratteri), poi, si verifica che la “vecchia password” sia corretta e che i campi “password nuova” e “ripeti password nuova” coincidano. Infine, si procede alla memorizzazione della nuova password nel database.

All'inizio di questa sezione, si era parlato anche di un file presente nella cartella “/other/”. Si tratta del file “account_opts.php”, richiamato nel file “header.php” (anche questo, presente in “/other/”, utilizzato in varie pagine del sito, per mostrare la “testata”). Questo file, permette la visualizzazione, in alto nel sito, della barra delle opzioni per gli utenti: Login e Iscrizione per i visitatori, Nuova news, lezione, materia (solo amministratori), e così via.
Ognuna di queste opzioni, rimanda alle pagine contenenti i form, già analizzate in precedenza.

scritto da Antonio Daniele Gialluisi

Progettazione del lato client

Il lato client è stato sviluppato usando due importanti linguaggi, ovvero HTML (HyperText Markup Language) e CSS (Cascade Style Sheets).

Poiché questa sezione – nel documento originale – non è stata redatta da Antonio Daniele Gialluisi, i suoi contenuti non sono stati inclusi.
Lo scopo è comunque pubblicare una copia del documento, evitando l’inclusione di lavoro svolto da altri membri del gruppo, nel lontano 2015, senza il loro permesso.