ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

PROGETTAZIONE E SVILUPPO DEL BACK-END PER IL MODULO DI CORREZIONE AUTOMATICA DEL CORSO DI PROGRAMMAZIONE

Relazione finale in

PROGRAMMAZIONE

Relatore:

Prof.ssa Antonella Carbonaro

Presentata da:

Elvis Pina

Sessione 3
Anno Accademico 2012-2013

Indice

Introduzione

Il web ha cambiato radicalmente le nostre vite. Grazie ad esso, oggi si possono fare cose che solo qualche decennio fa erano pura fantascienza, come ad esempio la telepresenza o gli interventi chirurgici da remoto, ma anche cose più "semplici" come seguire corsi di formazione (anche universitaria), effettuare la spesa, operare con il proprio conto corrente, tutto restando comodamente a casa propria, semplificando così la vita di tutti.

Allo stesso tempo il web è stato utilizzato per fini tutt'altro che nobili, ad esempio per commettere crimini informatici, recare danni alla concorrenza, compiere varie forme di truffe ecc.

Ogni persona dovrebbe comportarsi in modo corretto e nel pieno rispetto del prossimo, sia sul mondo reale che sul web, ma purtroppo non è sempre così. Per quanto riguarda il mondo del web, sta agli sviluppatori soddisfare le necessità dei propri utenti, assicurandosi però che la propria applicazione non verrà usata per recare qualche tipo di danno a terzi o alla propria infrastruttura.

Questa tesi nasce da un idea dei docenti del corso di Programmazione riguardo alla realizzazione di un modulo del sito web del corso che si occupa della correzione automatica di esercizi scritti in linguaggio C dagli studenti del corso, dove per correzione automatica si intende la verifica della correttezza degli esercizi.

La realizzazione della web application apporterebbe benefici sia ai docenti che agli studenti: ai docenti perché toglierebbe ad essi un compito piuttosto oneroso in termini di tempo, lasciando così più tempo da dedicare alla didattica o alla ricerca; agli studenti perché fornirebbe ad essi una risposta immediata sull'esito della correzione permettendo così una migliore distribuzione del tempo dedicato allo studio.

L'obbiettivo di questa tesi è quello di realizzare il back-end del modulo della correzione automatica, ovvero fornire una basse sulla quale verranno poi aggiunti i diversi componenti al fine di permettere la correzione in modo completamente automatizzata.

Il punto centrale del progetto è il database, che servirà per mantenere lo storico degli studenti e degli esercizi consegnati e per la generazione di report che saranno utili per avere una visione chiara della situazione in qualsiasi momento. È stato scelto di utilizzare il database MySQL in quanto soddisfa tutte le necessità di questo progetto, fornisce delle ottime prestazioni ed è completamente gratuito.

Sono stati implementati in PHP i meccanismi di registrazione e login degli utenti, indispensabili per una web application del genere, appoggiandosi all'database per la memorizzazione dei dati. Essendoci la concreta possibilità che in un dato momento vi siano più utenti che utilizzano il sito, vengono gestite le sessioni multiple.

Utilizzando sempre il linguaggio PHP, sono stati implementati gli script che gestiscono il caricamento e il salvataggio degli esercizi consegnati. I file di ogni utente vengono salvati una cartella che è solo di quell'utente. In questo modo i file risultano più organizzati anche a livello di filesystem. Inoltre, un'organizzazione del genere permette l'esecuzione di alcune operazioni direttamente da PHP, come ad esempio la generazione dell'elenco di tutti i file di un dato utente, alleggerendo così il carico di lavoro del database e diminuendo il tempo di risposta del server.

Una attenzione particolare è stata data alla sicurezza. Stono state adottate le strategie più consigliate dagli esperti di sicurezza per evitare gli attacchi che possono causare i danni più gravi ad una database web application quali SQL Injection, Session Hijacking, Cross Site Scripting e Cross Site Request Forgeries.

La tesi si articola come segue.

Nel capitolo 1 viene descritto il contesto di utilizzo della applicazione web tramite la spiegazione dell'iter che deve essere seguito dagli studenti per accedere all'esame di programmazione, gli obblighi e le scadenze da rispettare, e il modo in cui vengono assegnati dei bonus di punti.

Nel capitolo 2 vengono descritti gli strumenti utilizzati per la realizzazione della tesi ovvero i web server e in particolare il web server Apache, i database, i DBMS e in particolare MySQL, il linguaggio SQL e il linguaggio PHP.

Nel capitolo 3 viene introdotta la sicurezza delle web application. Vengono descritti gli attacchi che causano i danni maggiori ad una web application e le relative misure di difesa da adottare.

Nel capitolo 4 viene descritta la costruzione dell'database della applicazione.

Nel capitolo 5 viene presentata l'implementazione degli script e viene fornita una descrizione sulle scelte adottate.

Nel capitolo 6 vengono riassunte le nostre conclusioni e vengono proposti alcuni sviluppi futuri della web application.

1. Il Contesto di utilizzo

Durante lo svolgimento del corso ogni lunedì mattina vengono assegnati un insieme di esercizi che lo studente dovrà svolgere autonomamente e consegnare. Ogni esercizio viene opportunamente valutato e corretto, in caso di errori o mancanze lo studente dovrà apportare i relativi aggiornamenti e provvedere alla nuova consegna. Gli esercizi devono essere consegnati attraverso gli appositi pannelli messi a disposizione dall'sito del corso. Ogni studente attraverso la propria pagina di consegna può gestire i file sottomessi (esclusivamente file sorgente .c, file di testo .txt oppure .pdf), visionando i precedenti o sottomettendo nuovi file corrispondenti a versioni aggiornate dello stesso esercizio. La sottomissione di file corrotti o errati implica la valutazione negativa dell'esercizio.

In base al numero di esercizi assegnati settimanalmente vengono indicate un insieme di soglie che servono per l'attribuzione del punteggio settimanale che serve per individuare la valutazione finale del progetto.

Il numero totale di punti disponibili è 33; il punteggio minimo necessario per accedere all'esame scritto è 18 punti con raggiungimento in tutte le consegne almeno della soglia Dummy.

Le soglie di consegna sono:

- Dummy, consiste nel numero minimo di esercizi sviluppati correttamente per ritenere la consegna del blocco valida (assegna 0 punti)
- Pro, consiste nel numero minimo di esercizi sviluppati correttamente per ottenere 1 punto nella consegna
- Genius, consiste nel numero minimo di esercizi sviluppati correttamente per ottenere 3 punti nella consegna

A seconda del punteggio raggiunto, ed esclusivamente per gli appelli della sessione di gennaio-febbraio, verranno aggiunti al voto dello scritto (valutato almeno 18/30) 1,2 o 3 punti a seconda del punteggio raggiunto nelle consegne settimanali:

- +1 al voto dello scritto con punteggio maggiore a 22 punti
- +2 al voto dello scritto con punteggio maggiore a 26 punti
- +3 al voto dello scritto con punteggio maggiore a 30 punti

Esistono due modalità di consegna:

1. Settimanale. Questa modalità e attiva durante lo svolgimento del corso ed è valida solo per gli appelli di gennaio-febbraio. Consiste nella consegna degli esercizi entro le ore 24.00 della domenica successiva alla data di assegnazione.
2. Completa. Valgono le stesse regole indicate nella consegna settimanale con l'unica differenza che gli esercizi possono essere consegnati tutti entro 48 ore dalla data dell'appello scritto che si intende sostenere. Anche questa modalità di consegna permette di ottenere punti aggiuntivi negli appelli di gennaio-febbraio.

2. Strumenti utilizzati

2.1 Strumenti necessari per la realizzazione di una web database application.

Una web application non è un unico blocco a se stante, ma piuttosto un sistema complesso, in cui un numero anche elevato di componenti eterogenei interagiscono e collaborano, al fine di fornire un servizio o un prodotto agli utenti che ne fanno richiesta. In questo contesto viene definito l'acronimo XAMPP, una piattaforma software gratuita che ha permesso l'implementazione dell'applicazione web su una macchina che monta Windows come sistema operativo. XAMPP è l'acronimo di:

- X dovrebbe essere letto come "cross" e serve per indicare che si tratta di una cross-platform, ovvero una piattaforma universale. Infatti XAMPP può essere installato sui sistemi operativi più popolari che sono Windows, Linux e Unix.
- Apache: Il server web più diffuso attualmente.
- MySQL: Un Relational Database Management System (RDBMS) open source tra i più usati nel mondo.
- PHP: Il linguaggio di scripting più usato attualmete sul web.
- Perl: Un linguaggio di programmazione ad alto livello, anch'esso molto usato in ambito di programmazione lato server.

Oltre ai programmi menzionati, l'installazione di XAMPP comprende ulteriori strumenti molto utili per lo sviluppo web come ad esempio il web server Tomcat, il client FTP (File Transfer Protocol) FileZilla, il modulo OpenSSL e PhpMyAdmin.

Il principale vantaggio di XAMPP è che permette l'installazione di quasi tutti gli strumenti necessari per lo sviluppo web in poco tempo. Nel caso di Windows, si deve scaricare un unico file eseguibile (http://www.apachefriends.org/it/index.html). La procedura di installazione è molto semplice. Basta mandare in esecuzione il file .exe scaricato e confermare l'installazione. Il processo è completamente automatizzato e in pochi minuti si è pronti per potere utilizzare gli strumenti messi a disposizione. Nei casi di Unix o Linux, si scarica un file *tarball* che andrà scompattato nella cartella desiderata.

Oltre al vantaggio di utilizzare un unico file per l'installazione di tutti gli strumenti, un altro grande vantaggio di XAMPP è che viene fornito con una configurazione di base, quindi nella maggior parte dei casi, non necessita di alcuna configurazione. Ovviamente questo dipende dalla macchina utilizzata e dai programmi installati, ma sicuramente la configurazione è molto più semplice se confrontata alla installazione manuale di ogni singolo componente.

Un'altra considerazione da fare è che XAMPP è concepito per lo sviluppo web, quindi per un uso locale. Per rendere il processo più semplice possibile, alcune impostazioni di sicurezza sono disabilitate di default. Tuttavia, si potrebbe utilizzare XAMPP anche per servire le pagine sul web, in quanto e possibile abilitare le impostazioni di sicurezza e proteggere con password le parti più importanti del pacchetto.

Per la realizzazione di questa tesi non sono stati utilizzati tutti gli strumenti messi a disposizione dalla piattaforma XAMPP. Di seguito verrà fatta una descrizione solo degli strumenti e delle tecnologie utilizzate.

2.2 Web server

Prima di descrivere il server web Apache, spieghiamo cos'è un server web.

La funzione principale di un server web è quella di fornire pagine web per i client (di solito browser web). La comunicazione tra client e server avviene utilizzando HTTP (Hypertext Transfer Protocol). Le pagine consegnate dal server web sono solitamente documenti HTML, che possono includere immagini, fogli di stile e script (ad esempio JavaScript) in aggiunta al contenuto di testo.

Uno *user agent*, solitamente un browser web o *web crawler*, avvia la comunicazione facendo una richiesta per una risorsa specifica utilizzando HTTP e il server risponde con il contenuto di tale risorsa o un messaggio di errore se non è in grado di recuperare tale risorsa. La risorsa è in genere un file reale sulla memoria secondaria del server, ma non è sempre così e dipende da come il server web è stato implementato.

Mentre la funzione primaria è quella di servire i contenuti, una implementazione completa di HTTP comprende anche la possibilità di ricevere contenuti da parte dei client. Questa funzione viene utilizzata per la sottomissione di form web, compreso anche l'upload di file.

Molti server web generici supportano anche lo scripting lato server utilizzando PHP, Active Server Pages (ASP) o altri linguaggi di scripting. Ciò significa che il comportamento del server web può essere definito in script separati, mentre il vero software del server rimane invariato. Di solito, questa funzione viene utilizzata per creare pagine HTML dinamiche (on-the-fly).

I server web non sono sempre utilizzati per servire le pagine del World Wide Web. Possono anche essere trovati incorporati in dispositivi quali stampanti, router, webcam dove servono solo una rete locale. Un server web può essere utilizzato come parte di un sistema di monitoraggio e/o della gestione del sistema in questione. Questo di solito significa che nessun software aggiuntivo deve essere installato sul computer client, in quanto è necessario solo un browser web.

2.2.1 Apache

L'Apache HTTP Server, comunemente conosciuto come Apache, è una applicazione web server molto conosciuta per avere svolto un ruolo chiave nella crescita iniziale del World Wide Web. Originariamente basato sul server HTTPd della NCSA (National Center for Supercomputing Applications), lo sviluppo di Apache iniziò nei primi mesi del 1995, dopo che il lavoro sul codice della NCSA andò in stallo. Apache rapidamente superò HTTPd come server HTTP dominante, ed è rimasto il più popolare server HTTP in uso da aprile 1996. Nel 2009, divenne il primo software web server a servire più di 100 milioni di siti web.

Apache è sviluppato e mantenuto da una comunità aperta di sviluppatori sotto la supervisione della Apache Software Foundation. Più comunemente utilizzato su sistemi Unix-like, il software è disponibile per una vasta gamma di sistemi operativi, tra cui Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows, OS/2, TPF, OpenVMS e eComStation. Rilasciato sotto licenza Apache, Apache è un software open-source [1].

2.2.2 Funzionalità di Apache

Apache supporta una varietà di funzioni, molte implementate come moduli compilati che estendono le funzionalità di base. Queste possono variare dal supporto ai linguaggio di programmazione lato server a schemi di autenticazione. Alcune interfacce di linguaggio comuni supportano Perl, Python, Tcl, e PHP. Alcuni moduli di autenticazione popolari includono mod_access, mod_auth, mod_digest, e

mod_auth_digest, il successore di mod_digest. Un insieme di altre caratteristiche include il supporto a Secure Sockets Layer e Transport Layer Security (mod_ssl) e file di log personalizzati (mod_log_config).

I metodi di compressione più popolari in Apache comprendono il modulo esterno di espansione, mod_gzip, implementato per aiutare con la riduzione delle dimensioni delle pagine web servite su HTTP. I log di Apache possono essere analizzati attraverso un browser web utilizzando script gratuiti come AWStats/W3Perl.

L'hosting virtuale permette a una singola installazione di Apache di servire diversi siti web.

Apache permette la configurazione dei messaggi di errore, l'autenticazione basata su DBMS, e la negoziazione dei contenuti. È inoltre supportato da diverse interfacce grafiche (GUI).

Apache supporta l'autenticazione tramite password e l'autenticazione tramite certificato digitale. Poiché il codice sorgente è liberamente disponibile, chiunque può adattare il server per esigenze specifiche, e vi è una grande libreria pubblica di estensioni Apache.

2.2.3 Prestazioni di Apache

Anche se l'obiettivo principale del progetto Apache non è quello di essere il "più veloce" server web, Apache ha prestazioni simili ad altri server web ad alte prestazioni. Invece di implementare una architettura singola, Apache fornisce una varietà di moduli multiprocessing (MPM), che permettono ad Apache di essere eseguito in una modalità basata sui processi, ibrida (ovvero basata sui processi e i thread), o in una modalità basata sugli eventi, in modo da soddisfare al meglio le esigenze di ogni particolare infrastruttura. Ciò implica che la scelta del corretto MPM e la configurazione corretta è importante. Nei casi in cui bisogna fare dei compromessi sulle prestazioni, Apache è progettato in modo da ridurre la latenza e aumentare il throughput, assicurando così un'elaborazione omogenea e affidabile delle richieste in tempi ragionevoli.

2.3 Database

Un database è una collezione di dati correlati fra loro. Per essere considerato come database, la collezione deve godere in modo implicito delle seguenti proprietà:

- Un database rappresenta alcuni aspetti del mondo reale, a volte chiamato *minimondo*. I Cambiamenti del minimondo vengono riflessi sul database.
- Un database è un insieme logicamente coerente di dati che hanno un certo significato intrinseco. Un assortimento casuale di dati non può essere correttamente indicato come un database.
- Un database viene concepito, costruito e popolato con dei dati per uno scopo specifico.

In altre parole, un database ha qualche fonte da cui si ricavano i dati, un certo grado di interazione con eventi nel mondo reale, e un pubblico che è attivamente interessato ai suoi contenuti. Le transazioni effettuate dagli utenti o eventi del mondo reale fanno sì che i dati del database cambino. Un database deve sempre rappresentare una vera immagine del minimondo per essere considerato affidabile ed accurato.

Un database management system (DBMS) è una collezione di programmi che consente agli utenti di creare e mantenere un database. Il DBMS è un sistema software *general-purpose* che facilita i processi di definizione, costruzione, manipolazione e condivisione dei database tra i vari utenti e applicazioni:

- La definizione di un database comporta lo specificare i tipi di dati, le strutture e i vincoli dei dati da memorizzare nel database. La definizione del database o le informazioni descrittive sono anch'essi memorizzati dal DBMS nella forma di un catalogo database o dizionario conosciuto come meta-dati.
- La costruzione del database è il processo di memorizzazione dei dati su un supporto di memorizzazione che è controllato dal DBMS.
- La manipolazione di un database comprende funzioni come: fare delle richieste al database per recuperare dati specifici, aggiornamento del database per riflettere le variazioni del minimondo, e generazione di report dai dati.
- La condivisione di un database consente a più utenti e programmi di accedere al database contemporaneamente.

Un programma accede al database tramite l'invio di query (richieste) al DBMS. Una qyery comporta in genere il recupero di alcuni dati mentre una transazione può comportare la lettura di alcuni dati e la scrittura di altri dati sul database.

Altre funzioni importanti fornite dal DBMS includono la protezione del database e la sua manutenzione per un lungo periodo di tempo. La protezione comprende:

- La protezione del sistema da malfunzionamenti del hardware o del software.
- La protezione di sicurezza che previene l'accesso non autorizzato.

L'insieme composto dal database e dal DBMS verrà chiamato in seguito database system e viene illustrato in figura 2.1.

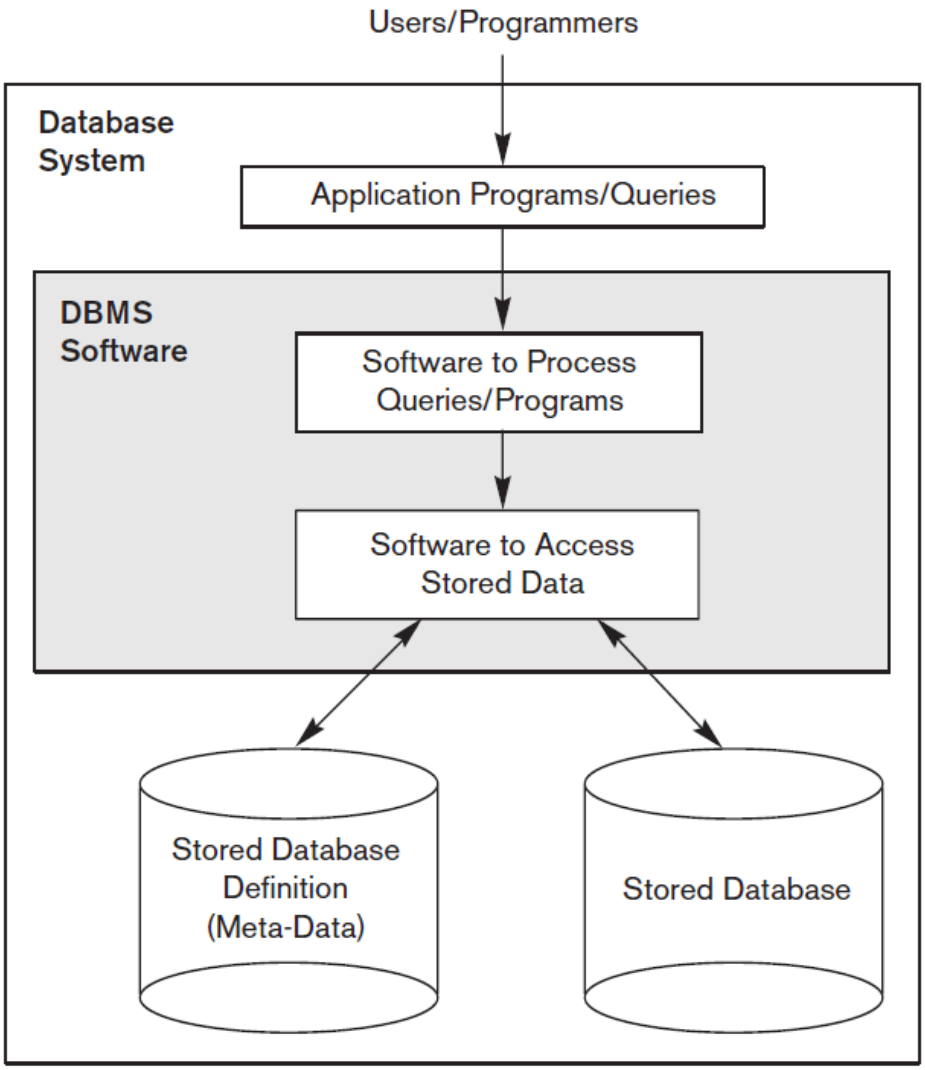


Figura 2.1 Schema di un database system.

2.3.1 Tipologie di database systems

Esistono diverse tipologie di database systems, ma faremo una breve descrizione delle tipologie più utilizzate attualmente, che sono: ad oggetti, relazionali ad oggetti e relazionali

2.3.1.1 I Database System ad ogetti

Object-Oriented Database Systems (OODBS), sistemi di database orientati agli oggetti, sono meccanismi di memorizzazione e recupero che supportano il paradigma della programmazione orientata agli oggetti (OOP), attraverso la manipolazione diretta dei dati come oggetti. Essi contengono veri sistemi di tipo object-oriented (OO) che consentono agli oggetti di persistere tra le applicazioni. La maggior parte, tuttavia, sono privi di un linguaggio di query standard (l'accesso ai dati avviene tipicamente tramite un'interfaccia di programmazione) e quindi non sono veri e propri DBMS. Gli OODBS sono un'alternativa interessante agli RDBMS, soprattutto nelle aree applicative in cui la potenza di modellazione o le prestazioni degli RDBMS nel memorizzare i dati nelle tabelle sotto forma di oggetti è insufficiente. Queste applicazioni mantengono grandi quantità di dati che non vengono mai cancellati, gestendo così lo storico dei singoli oggetti. La caratteristica più insolita degli OODBS è che forniscono un supporto per oggetti complessi specificando sia la struttura che le operazioni che possono essere eseguite su questi oggetti tramite delle interfacce OOP. Gli OODBS sono particolarmente adatti a modellare il mondo reale senza forzare relazioni non naturali tra le entità.

Le aree di utilizzo dei sistemi di database OO includono i sistemi di informazione geografica (GIS), database scientifici e statistici, sistemi multimediali, sistemi di archiviazione di immagini ecc [2].

2.3.1.2 Sistemi relazionali ad oggetti

Gli Object-Relational Database Management System (ORDBMS) sono una applicazione della teoria OO agli RDBMS. Un ORDBMS fornisce un meccanismo che permette ai progettisti di database di implementare un meccanismo di memorizzazione e recupero dei dati OO. Un ORDBMS fornisce le basi di un modello relazionale come l'integrità, le relazioni, e così via, mentre estende il modello di archiviazione e recupero dei dati in modo centrato sugli oggetti. L'implementazione è puramente concettuale in molti casi, poiché la mappatura dei concetti OO ai concetti relazionali è, nel caso migliore, un tentativo. Le modifiche o estensioni, alle tecnologie relazionali comprendono modifiche al SQL in modo da permettere la rappresentazione di tipi di oggetti, identità, incapsulamento delle operazioni e l'ereditarietà. Sebbene espressive, le estensioni di SQL non consentono una vera manipolazione degli oggetti o il livello di

controllo offerto dagli OODBS. Alcuni esempi di ORDBMS sono Illustra, PostgreSQL, e Informix [2].

2.3.1.3 Sistemi di database relazionali

Un RDBMS (Relational Database Management System) è un servizio di archiviazione e recupero dati basato sul modello relazionale dei dati, come proposto da E. F. Codd nel 1970. Questi sistemi sono i meccanismi standard di memorizzazione e recupero di dati strutturati.

Il modello relazionale è un concetto intuitivo di un archivio (database) che può essere facilmente interrogato utilizzando un meccanismo chiamato linguaggio di interrogazione (query language) per recuperare, aggiornare e inserire i dati. Il modello relazionale è stato implementato da molte società perché ha una solida teoria sistematica, una solida base matematica, e una struttura semplice. Il meccanismo di interrogazione più comunemente usato è lo Structured Query Language (SQL), che assomiglia al linguaggio naturale (ovviamente alla lingua inglese). Sebbene SQL non è incluso nel modello relazionale, SQL costituisce una parte fondamentale nella applicazione pratica del modello relazionale in un RDBMS.

I dati vengono rappresentati come pezzi di informazioni (attributi) correlati su una certa entità. L'insieme dei valori di un attributo forma una *tupla* (a volte conosciuto come *record*). Le tuple vengono memorizzate in tabelle che hanno lo stesso insieme di attributi. Le tabelle possono essere correlate ad altre tabelle tramite vincoli sui domini, chiavi, attributi e tuple [2].

I sistemi di database relazionali forniscono un indipendenza robusta dei dati e l'astrazione di essi. Utilizzando il concetto di relazioni, gli RDBMS forniscono un meccanismo di memorizzazione e recupero dei dati veramente generalizzato. Il rovescio della medaglia è, che questi sistemi sono molto complessi e richiedono una notevole esperienza per essere costruiti e modificati.

2.4 MySQL

MySQL è un RDBMS ad alte prestazioni, multithreaded e multiuser, basato sull'architettura client-server. Negli ultimi anni questo database system veloce, robusto e user-friendly è diventato la scelta di fatto sia per l'uso aziendale, che per quello

personale. Questo è dovuto alla suite di strumenti messi a disposizione per la gestione dei dati, ma anche alle politiche sulle licenze.

2.4.1 Storia di MySQL

MySQL è nacque nel 1979 quando Michael Widenius creò un database system chiamato UNIREG per conto della compagnia Svedese TcX. Tuttavia UNIREG non disponeva dell'interfaccia SQL e questo causò il suo declino a metà del 1990. TcX iniziò a cercare un alternativa e una di esse fu mSQL, un DBMS concorrente creato da David Hughes. Neanche mSQL funziono per la TcX, e per questo motivo Widenius decise di creare un nuovo server database personalizzato. Quel sistema, fu completato e rilasciato a un piccolo gruppo di persone a maggio del 1996 e diventò quello che oggi è conosciuto come MySQL.

Qualche mese dopo, MySQL 3.11 fu rilasciato al pubblico sotto forma di distribuzione binaria per Solaris, e poco tempo dopo anche per Linux

Qualche anno dopo Tcx diventò MySQL AB, una compagnia privata che era l'unica proprietaria del codice sorgente di MySQL, ed era responsabile per la manutezione, il marketing e lo sviluppo del server MySQL.

Nel 2008, MySQL AB fu acquisita da Sun Microsystems, e nel 2009, Sun Microsystems fu acquisita da Oracle, la quale al giorno d'oggi si occupa dello sviluppo del motore di MySQL. Il codice sorgente di MySQL rimane disponibile alla comunità sotto la licenza GNU General Public License (gli utenti possono acquistare supporto commerciale da Oracle).

2.4.2 Funzionalità uniche di MySQL

La popolarità di MySQL è dovuta a una combinazione particolare di funzionalità, che sono: velocità, affidabilità, estendibilità, e il codice open-source.

2.4.2.1 Velocità

In un RDBMS, la velocità, ovvero il tempo necessario per eseguire una query e restituire i risultati al chiamante è tutto. Sotto qualsiasi standard, MySQL è veloce, spesso ordini di grandezza più veloce della concorrenza. Benchmark disponibili sul sito web di MySQL dimostrano che MySQL supera quasi tutti gli altri database attualmente disponibili, tra cui concorrenti commerciali come Microsoft SQL Server 2000 e IBM DB2. Ad esempio, un studio effettuato da eWeek nel febbraio 2002 che ha confrontato

IBM DB2, Microsoft SQL Server, MySQL, Oracle9i e Sybase ha concluso che "MySQL ha le migliori prestazioni complessive e che la scalabilità di MySQL corrisponde a quella di Oracle… MySQL ha avuto il più alto throughput, superando anche i numeri generati da Oracle.".

2.4.2.2 Affidabilità

Il più delle volte, alte prestazioni del database hanno un costo, ovvero poca affidabilità. MySQL è progettato per offrire la massima affidabilità, ed è stato testato e certificato per l'utilizzo in applicazioni ad alto volume di dati. MySQL supporta le transazioni, che assicurano la consistenza dei dati e prevengono la perdita di dati. Inoltre supporta la replicazione e il clustering, due tecniche che riducono notevolmente il downtime in caso di guasto al server. Per ultimo, l'ampia base di utenti aiuta a scovare e risolvere velocemente i bug, e testare il software nei casi più disparati. Questo approccio proattivo ha dato come risultato un software che è virtualmente senza bug.

2.4.2.3 Scalabilità

MySQL può gestire database molto grandi e complessi senza un calo considerevole delle prestazioni. Non sono rari i casi di tabelle da diversi gigabyte e migliaia di record, e lo stesso sito di MySQL sostiene di usare database che contengono 50 milioni di record. Un test del 2005 svolto da MySQL Test Labs ha dimostrato che MySQL mostra una scalabilità quasi lineare in un ambiente multi-CPU, le prestazioni incrementano in proporzione al numero di CPU aggiunti al sistema. L'abilità di scalare in base alle richieste, ha reso MySQL popolare in siti ad alto volume come Google, Facebook ecc.

2.4.2.4 Codice open-source

Il codice sorgente di MySQL e disponibile gratuitamente sotto i termini della licenza GNU GPL. Questo è un beneficio chiave in quanto permette agli utenti di scaricare e modificare il software per soddisfare le proprie necessità.

2.5 Il linguaggio SQL

La sigla SQL, acronimo di *Structured Query Language* ovvero linguaggio di richiesta strutturato, è ormai diventata sinonimo di linguaggio standard per la definizione, l'interrogazione e l'aggiornamento di database relazionali. Nasce nel 1974 nei laboratori dell'IBM, come strumento per lavorare con database che seguano il modello relazionale. Nel 1975 viene sviluppato un prototipo chiamato *Sequel-XRM*; con esso si eseguirono sperimentazioni che portarono, nel 1977, a una nuova versione del linguaggio, che

inizialmente avrebbe dovuto chiamarsi *Sequel/2* ma che poi divenne, per motivi legali, SQL. Su di esso si sviluppò il prototipo *System R*, che venne utilizzato da IBM per usi interni e per alcuni suoi clienti. Dato il suo successo, anche altre società iniziarono subito a sviluppare prodotti basati su SQL. Nel 1981 IBM iniziò a vendere alcuni prodotti relazionali e nel 1983 rilasciò DB2, il suo DBMS relazionale diffuso ancora oggi. SQL divenne subito lo standard industriale per i software che utilizzano il modello relazionale. L'ANSI (American National Standards Institute) lo adottò come standard fin dal 1986, senza apportare modifiche sostanziali alla versione inizialmente sviluppata da IBM. Nel 1987 la ISO (International Prganisation for Standardization) fece lo stesso. Questa prima versione standard è denominata SQL/86. Negli anni successivi si realizzarono altre versioni, che furono SQL/89, SQL/92 e SQL/2003. Tale processo di standardizzazione mirava alla creazione di un linguaggio che funzionasse su tutti i DBMS relazionali, ma questo obiettivo non fu raggiunto. Infatti, i vari produttori implementarono il linguaggio con numerose variazioni e, in pratica, adottarono gli standard ad un livello non superiore al minimo, definito dall'ANSI come Entry Level.

Essendo un linguaggio dichiarativo, SQL non richiede la stesura di sequenze di operazioni (come ad es. i linguaggi imperativi), ma piuttosto di specificare le proprietà logiche delle informazioni ricercate. Esso si divide in tre sottoinsiemi:

- Data Definition Language (DDL), linguaggio di definizione dei dati, viene utilizzato per definire lo schema del database. Le definizioni dello schema vengono memorizzate in una base di dati speciale, detta Dizionario dei dati (*Data Dictionary*). Il DDL opera sia a livello esterno, sia a livello logico, sia a livello interno; in particolare, le istruzioni che definiscono il livello esterno prendono talvolta il nome di *view definition* (definizione delle viste, cioè degli schemi dell'utente), mentre le definizioni che definiscono il livello interno prendono il nome di DCL (*device control language*, o linguaggio per il controllo dei dispositivi).

- Data Manipulation Language (DML), viene utilizzato per manipolare la base di dati, cioè per svolgere le seguenti funzioni: Estrarre informazioni dalla base di dati, cioè formulare un'interrogazione anche complessa sulla base di dati ed ottenere la riposta all'interrogazione; modificare il contenuto della base di dati, cioè svolgere operazioni di inserimento di nuovi dati, cancellazioni di dati preesistenti, e modifica di specifici valori di dati preesistenti. I comandi utilizzati

sono: *SELECT* per la ricerca, *INSERT* per l'inserimento, *UPDATE* per l'aggiornamento e *DELETE* per la cancellazione.

- Data Control Language (DCL), linguaggio utilizzato per fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi DML e DDL, oltre agli stessi comandi DCL (che servono per poter a sua volta modificare i permessi su alcuni oggetti).

Una caratteristica importante del linguaggio è il suo grado di proceduralità: un linguaggio è procedurale se il programmatore deve specificare in modo esatto quale metodo di accesso ai dati dovrà essere utilizzato per ritrovare l'informazione; un linguaggio è non procedurale se l'utente deve solo indicare quale informazione dovrà essere ritrovata, e viceversa omettere di indicare il modo in cui l'informazione verrà ritrovata. In tal caso, il sistema determina in modo automatico il cammino di accesso ai dati.

2.6 Il linguaggio PHP

PHP, acronimo ricorsivo di (PHP Hypertext Prepocessor), attualmente è uno dei linguaggi di programmazione per il web più diffusi su internet. PHP può essere usato principalmente in tre modi:

- Scripting lato server: PHP è stato progettato principalmente per creare contenuti dinamici per il web. PHP e diventato popolare grazie anche al fatto che è in grado di generare documenti XML, file PDF e tanto altro.
- Scripting da linea di comando: PHP può eseguire gli script anche da linea di comando come ad esempio dalla shell di Unix. Si possono usare script dalla linea di comando per svolgere compiti amministrativi come ad esempio il backup.
- Applicazioni GUI lato client: Tramite l'utilizzo di PHP-GTK si possono scrivere applicazioni GUI cross-platform in PHP.

Per questa tesi, PHP è stato utilizzato come linguaggio di scripting lato server in modo da consegnare pagine dinamiche. Per pagine dinamiche si intendono pagine il cui contenuto viene, almeno in parte, generato nel momento in cui le stesse vengono richieste al web server. Un esempio classico di pagina dinamica è fornito dai motori di ricerca: i risultati che vengono restituiti a seguito di una interrogazione non sono pagine web statiche, bensì documenti generati "su misura", sulla base della richiesta. PHP è un vero e proprio linguaggio di programmazione a differenza di HTML, che

non è un linguaggio di programmazione, ma un linguaggio di descrizione e formattazione del contenuto di una pagina. Dalla documentazione ufficiale il PHP viene definito come un "linguaggio di scripting lato server, inserito (embeded) nel codice HTML". Di default gli script PHP hanno l'estensione *.php*. Quando un server vede questa estensione in un file richiesto, lo passa automaticamente al preprocessore PHP, il quale lo interpreta e genera il file HTML che viene poi inviato tramite HTTP. Dato che i server web sono molto personalizzabili, si potrebbero configurare in modo da forzare anche l'interpretazione dei file che hanno l'estensione *.htm* o *.html* dal preprocessore PHP. Di solito questo si fa per nascondere il fatto che una pagina sta usando PHP.

PHP è un linguaggio interpretato. Questo significa che gli script scritti in PHP non devono essere compilati come succede ad esempio con i file scritti in linguaggio C, Java ecc, ma ogni istruzione viene interpretata, ovvero vengono eseguite le operazioni corrispondenti.

PHP opera sul server. Questo implica che il richiedente di una pagina PHP non può vedere direttamente il codice, ma vede solo il risultato dell'esecuzione di esso.

2.6.1 Storia del PHP

Nel 1995 Rasmus Ledorf sviluppò uno script Perl/CGI che gli permetteva di sapere quante persone avevano letto il suo résumé. Lo script eseguiva due compiti: tenere traccia dei dati degli utenti e visualizzare il numero dei visitatori della pagina web. All'epoca il web era molto giovane e strumenti del genere non esistevano. Questo portò molti sviluppatori a volere sapere di più su questi script. Ledorf iniziò a rilasciare questi script sotto il nome Personal Home Page tools.

Il clamore generato da PHP tools spinse Ledorf a continuare lo sviluppo del linguaggio che portò all'implementazione di una funzionalità che convertiva i dati inseriti in una form HTML in variabili simboliche, incoraggiando la possibilità di esportare tali dati anche in altri sistemi. Per raggiungere questo traguardo, Ledorf cambiò il linguaggio di sviluppo, passando al linguaggio C. Ulteriori funzionalità aggiunte culminarono nel novembre del 1997 con il rilascio di PHP 2.0, altrimenti chiamato Personal Home Page/Form Interpreter (PHP/FI). Come risultato della crescente popolarità, la release 2.0 fu accompagnata da diversi miglioramenti da programmatori di tutto il mondo.

La nuova release di PHP fu molto popolare e un piccolo gruppo di sviluppatori si unì a Ledorf. Loro tenerono il concetto originale di incorporare il codice direttamente nel file

HTML e riscrissero il motore di parsing, dando vita a PHP 3.0. A giugno del 1998 circa 50'000 utenti stavano usando PHP per aggiungere funzionalità ai propri siti web.

Lo sviluppo continuò con ritmi frenetici nei seguenti due anni. Centinaia di funzioni vennero aggiunte e il numero di utenti andò sempre crescendo. All'inizio del 1999, la Netcraft (www.netcraft.com), una compagnia di ricerca e analisi di Internet, pubblicò una ricerca secondo la quale più di un milione di utenti usavano PHP, classificando così il PHP come il linguaggio di scripting web più popolare del mondo. La sua popolarità superò anche le migliori delle aspettative dei suoi sviluppatori e fu chiaro che gli utenti erano intenzionati a usare PHP per la realizzazione di applicazioni molto più grandi di quanto fosse concepito inizialmente. Due sviluppatori centrali, Zeev Zuraski e Andi Gutmans, presero l'iniziativa di ripensare completamente il modo in cui PHP operava, riscrivendo il parser PHP che fu chiamato *Zend scripting engine*. Il risultato di questo lavoro fu la release di PHP 4.0

Il rilascio di PHP 4.0 viene considerato da molti come il debutto di PHP a livello Enterprise in quanto aggiungeva molti miglioramenti del linguaggio orientate alle grandi aziende. Tali miglioramenti comprendevano:

- Miglioramento della gestione delle risorse, che aumentò considerevolmente la scalabilità del prodotto.
- Supporto Object-Oriented: La versione 4.0 incorporò un certo grado di funzionalità Object-Oriented.
- Supporto nativo per la gestione delle sessioni.
- Criptaggio: Furono incorporati molti algoritmi di criptaggio come *MD5*, *SHA1* e *TripleDES*.
- Supporto *ISAPI*: Offriva agli utenti la possibilità di usare PHP in modo congiunto con Microsoft IIS Web server.
- Supporto nativo *COM/DCOM*: Questa funzionalità aprì un ampio raggio di interoperabilità con le applicazioni Windows.
- Supporto nativo al Java: Supporto al binding di oggetti Java da una applicazione PHP.
- La libreria Perl Compatible Regular Expressions (PCRE): Rese possibile l'utilizzo di Regular Expressions scritte per Perl.

La versione 5 fu ancora una svolta nell'evoluzione del linguaggio PHP. Diversamente dalle precedenti major release, che implementavano un grande numero di funzionalità, in questa versione vengono migliorate molte funzionalità esistenti e vengono aggiunte funzionalità tipiche dei linguaggi di programmazione maturi:

- Notevole miglioramento delle capacità Object-Oriented. La versione 5 include funzionalità aggiuntive come costruttori e distruttori espliciti, clonazione di oggetti, astrazione e interfacce.
- Gestione delle eccezioni con try/catch.
- Supporto migliorato per XML e Servizi Web.
- Supporto nativo per SQLite.

La versione che doveva essere rilasciata come 6.0 ha avuto molti ritardi, ma una grande parte delle funzionalità pensate per essere rilasciate con quella versione, sono state rilasciate con la versione 5.3 e sono:

- Supporto ai namespace.
- Late static binding.

3. Sicurezza

3.1 Introduzione alla sicurezza

Può sembrare inconcepibile che una persona razionale lasci incustoditi dei beni preziosi in dei posti dove possono essere sottratti, eppure vediamo succedere una cosa del genere tutti i giorni nel mondo del web, dove vengono scritti ogni giorno script che non prendono neanche le minime misure di precauzione per salvaguardare i dati che gestiscono o gli ambienti sui quali vengono eseguiti.

Differentemente dalle funzionalità offerte da un linguaggio di programmazione come ad esempio espressioni condizionali o costrutti di looping, la sicurezza è una cosa astratta. La sicurezza non è una caratteristica di un linguaggio quanto lo è di uno sviluppatore. Nessun linguaggio può proteggere il codice scritto in maniera sbagliata, anche se ci sono caratteristiche del linguaggio che possono aiutare od ostacolare uno sviluppatore attento alla sicurezza.

In questo capitolo, verranno spiegate le vulnerabilità più pericolose di una applicazione PHP e le relative tecniche di difesa.

3.2 Validazione e sanificazione dell'input dell'utente.

I dati forniti dagli utenti sono inutili se non vengono usati, ma, paradossalmente, il semplice accesso ad essi costituisce un pericolo. Particolarmente pericolose sono le query effettuate dalle richieste degli utenti, tipicamente sottomesse tramite qualche form. Utenti legittimi potrebbero accidentalmente fare delle richieste che poi si rivelano pericolose, mentre utenti "illegittimi" potrebbero creare intenzionalmente delle richieste pericolose, sperando che queste passino oltre le protezioni del sistema.

Il tipo di attacco più comune, intenzionale o meno, avviene quando un utente fornisce dati del tipo o della dimensione sbagliata, oppure inserisce dati che contengono caratteri speciali o sequenze di codice binario. L'input di dati del formato sbagliato, potrebbe causare la chiusura in modo anomalo dell'applicazione, scrittura di dati non corretti su un database, o persino la cancellazione di dati dal database.

3.3 Input contenente metacaratteri

Persino il più comune input di caratteri alfanumerici potrebbe essere pericoloso se contiene qualcuno dei caratteri conosciuti come metacaratteri, caratteri che hanno un significato particolare quando vengono processati da parti di un sistema. Questi caratteri potrebbero essere inviati facilmente da un malintenzionato in quanto possono essere semplicemente digitati da tastiera.

Un insieme di metacaratteri comprende quelli che innescano vari comandi o funzioni implementate nella shell. Alcuni esempi sono:

| ! $ & / ( ) ? ^ - [ ] { } \ ` "

Questi caratteri inseriti in una stringa, passata come parametro alla shell da PHP, se non trattati in modo adeguato, potrebbero causare un azione che lo sviluppatore molto probabilmente non intendeva eseguire.

Un altro set di caratteri comprende quelli che hanno un significato particolare nelle query ai database. Alcuni esempi sono:

` " - ; \

A seconda di come la query è strutturata ed eseguita, questi caratteri potrebbero essere usati per iniettare statement aggiuntivi in essa, e potenzialmente eseguire query arbitrarie.

Esiste un altro set di caratteri che non sono facili da digitare, e la cui pericolosità non è così ovvia, ma che potrebbero rappresentare un pericolo per il sistema e il database. Questi sono i primi 32 caratteri nel set ASCII standard, a volte conosciuti come caratteri di controllo in quanto originariamente venivano usati per controllare alcuni aspetti di controllo della visualizzazione e della stampa del testo. Anche se sarebbe perfettamente legittimo se alcuni di questi caratteri comparissero in alcuni campi contenenti valori binari (ad esempio immagini), non sarebbe altrettanto normale se comparissero in una comune stringa ti testo. Esistono comunque alcuni di questi caratteri che sarebbero perfettamente legittimi anche nelle stringhe di testo. Questi caratteri sono:

- Il carattere \x00, altrimenti conosciuto come ASCII 0, NULL o FALSE.
- I caratteri \x10 e \x13, altrimenti conosciuti come ASCII 10 e 13, o \n e \r, i caratteri di new line e line feed.
- Il carattere \x1a, altrimenti conosciuto come ASCII 26, che funge da marcatore di fine del file.

Alcuni di questi caratteri o codici, se compaiono in modo inaspettato nell'input testuale dell'utente potrebbero, nel migliore dei casi, corrompere l'input dell'utente, e nel peggiore dei casi potrebbero permettere l'iniezione di script da parte di un attaccante.

Infine, esiste un grande numero di caratteri multibyte Unicode, caratteri maggiori di \xff, che rappresentano caratteri non latini e segni di interpunzione. Unicode definisce sequenze particolari lunghe due o quattro byte, che servono per rappresentare quasi tutti gli alfabeti umani e un ampio numero di simboli. Questi caratteri multibyte non hanno significato se spezzati in sequenze da un byte, ma potrebbero essere molto pericolosi se dati in input a programmi che si aspettano in input testo ASCII. Di per sé PHP gestisce i caratteri mulibyte in modo sicuro, ma altri programmi, database o filesystem potrebbero non fare lo stesso.

3.4 Input del tipo sbagliato

I valori in input che non sono di un tipo corretto o sono in un formato non valido, molto probabilmente avrebbero effetti non intenzionati e perciò indesiderati su una applicazione. Nel caso migliore potrebbero causare errori che potrebbero fornire informazioni sul sistema sottostante. Nel caso peggiore potrebbero fornire un vettore per un potenziale attacco. Alcuni esempi sono:

- Se si aspetta in input una data, che verrà poi usata per creare una timestamp Unix, ma viene fornito un altro tipo di valore, il risultato sarebbe la creazione di una timestamp con il valore 31 Dicembre 1969, ovvero il secondo -1 nei sistemi Unix.
- Le applicazioni che manipolano immagini, probabilmente si bloccherebbero se vi venisse fornito un input non di tipo immagine.
- Le operazioni dei filesystem fallirebbero con risultati non predicibili in caso venga fornito ad essi dati binari (o, a seconda del sistema operativo, alcuni segni di interpunzione) come parte del nome del file.

3.5 Troppo input

Un input troppo lungo potrebbe causare la terminazione dell'applicazione, creare effetti indesiderati, oppure creare le condizioni di *buffer overflow* nelle librerie sottostanti o nelle applicazioni in esecuzione. Alcuni possibili esempi sono:

- Alcuni campi delle tabelle nei database sono limitate a 255 caratteri o meno. Un eventuale input troppo lungo sottomesso dall'utente potrebbe essere troncato senza generare dei warning, perdendo in questo modo parte dell'informazione fornita dall'utente.
- I nomi dei file hanno dei limiti di lunghezza. Le utilità dei filesystem che ricevono un input troppo lungo potrebbero semplicemente continuare l'esecuzione troncando il nome (con risultati probabilmente disastrosi) oppure andare in crash.
- Il buffer overflow è sicuramente il pericolo maggiore con un input troppo lungo. Si ha un buffer overflow quando l'utente sottomette una quantità di dati maggiore della quantità di memoria allocata dall'applicazione per ricevere tale informazione. La parte finale dei dati finisce per essere scritta in memoria, con i seguenti possibili risultati:
    - Una variabile esistente potrebbe essere sovrascritta.
    - Potrebbe essere generato dall'applicazione un errore o l'applicazione potrebbe andare in crash.
    - Un istruzione potrebbe essere sovrascritta da un istruzione che esegue il codice sottomesso.

3.6 Strategie per validare l'input dell'utente

3.6.1 Dichiarazione delle variabili

In molti linguaggi di programmazione, è obbligatorio dichiarare le variabili prima di utilizzarle, ma PHP essendo un linguaggio molto flessibile, crea in automatico una variabile che si cerca di usare senza averla dichiarata prima. Questo comportamento di PHP lascia spazio per un possibile utilizzo scorretto della applicazione. Normalmente in molte applicazioni l'amministratore ha la possibilità di eseguire operazioni non permesse ad un utente normale. Un comportamento tipico è quello di controllare il livello di accesso dell'utente e, se l'utente è un amministratore, creare una variabile nella quale viene memorizzata questa informazione. Un esempio di questa pratica potrebbe essere il seguente:

```php
<?php
    if ($user->isAdmin()) {
        $admin = true;
    }
    if ($admin) {
        // Esegui operazioni da amministratore
    }
?>
```

Questo codice potrebbe sembrare corretto, ma bisogna notare il fatto che ad un malintenzionato basterebbe aggiungere alla URL la stringa `?admin=true` e php creerebbe al volo la variabile `$admin` e le assegnerebbe il valore *true*. In quel caso l'attaccante avrebbe i privilegi da amministratore e questo potrebbe avere effetti disastrosi. Per difendersi da questo tipo di attacco, nel caso del esempio, basterebbe creare la variabile `$admin` e assegnarle il valore *false* come nell'esempio seguente:

```php
<?php
    $admin = false;
    if ($user->isAdmin()) {
        $admin = true;
    }
    if ($admin) {
        // Esegui operazioni da amministratore
    }
?>
```

Questo è solo un esempio semplice, ma è considerata buona norma dichiarare sempre le variabili prima di utilizzarle.

3.6.2 Permettere solo l'input atteso

Anche nelle applicazioni poco complesse, viene considerata una buona pratica elencare in modo esplicito le variabili che si aspettano in input. Una volta elencate le variabili, tramite, ad esempio, un loop *foreach()* vengono recuperati solo i valori attesi. Un esempio semplice potrebbe essere:

```php
<?php
    $expected=array('nome','email','password');
    foreach ($expected as $key) {
        // Fai qualcosa con $_POST[$key]
    }
?>
```

In questo modo, vengono completamente ignorati eventuali dati aggiuntivi inseriti da un malintenzionato con l'intento di causare problemi.

3.6.3 Controllo del tipo, lunghezza e formato dell'input

Quando viene data la possibilità di inserire dei dati a un utente, generalmente tramite la sottomissione di una form, si sa in anticipo che tipo di dati ci si aspetta che l'utente fornisca. Sfruttando questa conoscenza, è relativamente facile controllare che i dati inseriti dall'utente siano del tipo, lunghezza e formato giusto.

3.6.4 Controllo del tipo

3.6.4.1 Stringhe

Le stringhe sono il tipo di dati più facile da controllare in PHP, in parte dovuto al fatto che quasi tutto può essere considerato una stringa. Nonostante ciò, alcuni dati non possono essere considerati strettamente come stringhe e in questi casi l'uso della funzione *is_string()* ci viene in aiuto. L'utilizzo di questa funzione sarebbe corretto nei casi in cui anche i numeri sono da considerare come stringhe. Nella maggior parte dei casi, ciò che ci serve realmente è sapere che la stringa non sia vuota e in questi casi si potrebbe utilizzare la funzione *empty()* che restituisce *false* nei casi in cui la stringa contenga dei dati, ovvero non è vuota.

Nei casi in cui anche una stringa vuota si vuole considerare comunque una stringa valida, sarebbe opportuno effettuare un controllo come nell'esempio seguente:

```php
<?php
    if ( isset( $valore ) && $valore !== NULL ) {
    // $valore è una stringa (possibilmente vuota) secondo PHP
    }
?>
```

3.6.4.2 Numeri

Se si aspetta in input un numero, l'inserimento di un dato non numerico potrebbe causare dei problemi. Nonostante PHP consideri di default tutti dati provenienti dalle form come stringhe, il suo type casting automatico ci permette di determinare se una data stringa può essere interpretata come un valore numerico. Per fare ciò si potrebbero usare le funzioni *is_int()* oppure *is_long()* in modo simile al esempio seguente:

```
<?php
    $numero = $_POST['numero'];
    if (!is_int($numero))
        exit ("$numero è un valore non permesso per numero!");
?>
```

Un altro modo per effettuare questa verifica è quello di usare la funzione *gettype()*

```
if ( gettype( $numero ) != 'integer' ) {
    exit ( "$numero è un valore non permesso per numero!" );
}
```

I precedenti test fallirebbero in caso il valore sia 0 o un numero non intero, quindi nei casi in cui si aspetta un valore del tipo floating point andrebbe usata la funzione is_numeric().

Una volta verificato che il valore sia numerico, bisogna fare un casting esplicito per recuperare il valore e convertirlo nel tipo apropriato.

3.6.4.3 True e false

Come le stringhe, i valori booleani non sono un problema generalmente, ma bisogna stare attenti perché a volte possono generare confusione. Un esempio potrebbe essere una stringa che contiene il valore "false". Dato che tale stringa non è vuota, sarà valutata come True da PHP.

3.6.5 Controllo della lunghezza

Tramite il controllo della lunghezza dell'input fornito da un utente, si possono prevenire gli attacchi di buffer overflow e *denial of service*. Per esempio, se si aspetta che l'utente fornisca un anno, e si riceve una stringa che invece di quattro caratteri ne contiene cento, sarebbe ragionevole pensare che in quei dati c'è qualcosa che non va. La lunghezza di una stringa si può controllare con la funzione *strlen()*, che conta il numero dei caratteri da un byte in una stringa. Il controllo necessario per l'esempio precedente potrebbe essere effettuato facilmente in questo modo:

```
if (strlen($anno) > 4)
    exit ("$anno non è un valore permesso per anno!");
```

3.6.6 Controllo del formato

Oltre al tipo e alla dimensione, è importante controllare anche il formato dei dati forniti dall'utente. Strettamente parlando, una stringa contenente un indirizzo e mail e una stringa contenente un anno non hanno lo stesso tipo di valore. Entrambe sono di tipo stringa, ma ognuna ha un formato particolare, ed è importante assicurarsi che tale formato venga rispettato per fare sì che l'applicazione funzioni senza errori. Dal punto di vista della sicurezza, il formato corretto è importante in modo particolare quando PHP passa i dati ad un'altra applicazione come ad esempio un database.

3.6.7 Nomi dei file e i loro percorsi

Le stringhe che contengono i nomi dei file sono ristrette dal sistema operativo in modi in cui le altre stringhe non lo sono.

- I nomi dei file non possono contenere dati binari. Un attaccante che riesce a inserire questo tipo di dati nel nome di un file potrebbe causare dei problemi.
- I nomi dei file in alcuni sistemi operativi possono contenere caratteri unicode, ma in altri sistemi no. Per questo motivo è meglio restringere i caratteri permessi per i nomi dei file ai soli caratteri ASCHII.
- Anche se i sistemi operativi Unix-based permettono la maggior parte dei segni di interpunzione nei nomi dei file, è meglio evitare i segni di interpunzione in quanto potrebbero avere un significato particolare per il sistema. Viene considerata buona norma permettere solo i caratteri trattino, o *hyphen* (-), *underscore* (_) e punto (.) come parte dei nomi dei file.
- I nomi dei file hanno dei limiti di lunghezza. Questo limite comprende anche il percorso del file, il che vuol dire che in sistemi complessi viene ridotta la dimensione disponibile per il vero nome del file.

Le variabili usate nelle operazioni del filesystem, come *fopen()* o *file_get_cntents()*, possono essere costruite in modo da rivelare informazioni sul sistema sottostante. Il colpevole principale in questo tipo di attacco è l'operatore '..' (punto punto) che si usa nei sistemi operativi per salire di un livello nel filesystem. Per fare un esempio, un utente potrebbe caricare un file che si chiama ../../../cartella_finta/nomeFIle. Quando si cercherà di spostare tale file nella cartella destinata al salvataggio dei file caricati, il sistema vedrà una parte del nome come parte del percorso di destinazione, e cercherà di spostarla in tale percorso che ovviamente non esiste. Questo genererebbe un avviso che

rivelerebbe il percorso dello script e altre informazioni sul sistema. Ovviamente questo è un esempio semplice, ma si potrebbero creare script molto più pericolosi che avrebbero effetti disastrosi. Una difesa possibile da questo tipo ti attacco potrebbe essere quella di non permettere nel nome del file due punti consecutivi e segni di interpunzione in generale, eccetto quelli descritti sopra.

3.7 Cross Site Scripting (XSS)

Il Cross Site Scripting (XSS), è una delle vulnerabilità più comuni che affliggono le web application. In questo tipo di attacco, l'attaccante cerca di memorizzare markup maligno o codice JavaScript in variabili che saranno utilizzate successivamente per visualizzare dei contenuti in una pagina web. Questo codice maligno cercherà di fare eseguire al browser dell'utente alcune operazioni sicuramente non gradite, come ad esempio rubare i cookie dell'utente o reindirizzare informazioni confidenziali su un sito di terze parti.

Ci sono principalmente due tipi di attacchi XSS:

- *Stored XSS* generalmente accade quando l'input dell'utente viene memorizzato nel server bersaglio, ad esempio in un database o una pagina di un forum, e la vittima recupera queste informazioni senza che esse vengano prima rese sicure per la visualizzazione in un browser.
- *Reflected XSS*. In questo tipo di attacco il codice maligno viene aggiunto ad un URL che successivamente viene inviato alla vittima. Quando la vittima visita il link, l'attacco è andato a buon fine. Questo tipo di attacco di solito si trova sotto forma di pagine di errore, risultati di ricerca o link resi insospettabili tramite l'utilizzo di siti come tinyurl.com.

Ogni web application che visualizza informazioni fornite dagli utenti e vulnerabile a questo tipo di attacco. Nonostante non si possa essere immuni al 100% da XSS, si possono prendere delle precauzioni che eliminano una buona parte dei pericoli, ad esempio, l'uso di POST per l'invio dei dati dalle form rende molto più sicura l'applicazione in quanto diventa più difficile sottomettere certi tipi di dati, ma ovviamente questo non basta. Un altro grado di protezione viene fornito dall'utilizzo della funzione *htmlspecialchars()*. Questa funzione converte tutti i caratteri che possono

essere interpretati dal parser HTML in entità HTML. Quello che succede in pratica è che un tentativo di inserire uno script del tipo:

```
<script>alert('XSS');</script>
```

verrebbe convertito in

```
&lt;script&gt;alert('XSS');&lt;/script&gt;
```

e questo comporterebbe la non esecuzione dello script. In generale, la miglior difesa è considerare ogni dato fornito dall'utente come insicuro e quindi usarlo con cautela.

3.8 SQL injection

SQl Injection è una delle vulnerabilità più comuni in una web database application. La cosa più sorprendente è che un SQL injection è il risultato di un doppio errore da parte dello sviluppatore. Il primo errore è il mancato filtraggio dei dati in entrata, mentre il secondo e il mancato trattamento adeguato dei dati prima di essere inviati al database. Nessuno di questi passaggi fondamentali deve essere ommesso ed entrambi necessitano di attenzioni particolari in modo da minimizzare gli errori.

Un attacco SQL injection consiste nell'inserimento, o iniezione, di una query SQL tramite l'input dei dati, dall'utente all'applicazione. Quando l'iniezione va a buon fine possono essere letti dati sensibili dal database, essere modificati i dati del database (inserimento, cancellazione, modifica), essere eseguite operazioni amministrative sul database (ad esempio spegnere il database), recuperare il contenuto di un dato file presente nel file system del DBMS e in alcuni casi fare eseguire dei comandi al sistema operativo.

Un esempio semplice di un SQL injection si potrebbe avere nelle form di login. Quello che succede tipicamente è che le credenziali di ogni utente risiedono in un database. L'utente fornisce il nome utente e la password. Con questi dati si costruisce una query al database (quindi l'utente viene reso parte attiva nella creazione della query). Una query molto usata in questi casi è:

```php
<?php
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT * FROM User WHERE username='$username'
    AND password='$password'";
    $result = mysql_query($query);
?>
```

A un utente malintenzionato basterebbe inserire come nome utente la stringa

```
' OR 1=1;--
```

L'inserimento di una stringa del genere comporterebbe la creazione di una query del tipo:

```
SELECT * FROM User WHERE Username='' OR 1=1;--AND password ='';
```

Dato che 1=1 è sempre vero e -- indica l'inizio di un commento, una query del genere restituirebbe i dati di tutti gli utenti presenti sul database. Ovviamente una cosa del genere è inaccettabile ed è da evitare assolutamente.

Una soluzione usata frequentemente è quella di sanificare i dati che saranno utilizzati per la realizzazione della query tramite la funzione *mysql_real_escape_string()*, che è una funziona studiata appositamente per questo scopo. Questa funzione rende inoffensivi i caratteri pericolosi tramite l'inserimento di un backslash davanti a ogni carattere tale, in modo da non permettere l'interpretazione dei suddetti caratteri. Nonostante questa funzione sia molto valida, a partire dalla versione 5.5 di PHP, tale funzione è stata deprecata e i creatori di PHP invitano a utilizzare altre soluzioni.

Una soluzione alternativa, utilizzata anche in questa tesi, è quella di adottare PDO. PDO (PHP Data Objects) è una delle estensioni più interessanti introdotte nell'implementazione della versione 5 di PHP. Essa nasce infatti dall'esigenza di recuperare una grave mancanza che per lungo tempo aveva caratterizzato questo linguaggio di sviluppo, cioè l'assenza di un'unica interfaccia per l'accesso ai database. PDO fornisce un *data-access abstraction layer*, cioè un livello di astrazione per l'accesso ai dati. Si tratta infatti di una classe, definita forse impropriamente anche come "libreria", che mette a disposizione un insieme di sotto-classi derivate che agiscono in modo trasparente rispetto all'utente. Il suo utilizzo permette di creare applicazioni dotate della massima portabilità possibile, e l'utilizzatore può scegliere il DBMS di riferimento sulla base delle proprie esigenze senza particolari costrizioni relative alla tipologia di accesso ai dati dell'applicazione.

Un altro vantaggio di PDO è che fornisce supporto ai prepared statementes. Uno prepared statement, conosciuto anche come statement parametrizzato, è usato per eseguire lo stesso statement ripetutamente con un alta efficienza. L'esecuzione di uno prepared statement avviene in due fasi: la preparazione e l'esecuzione. Durante la fase di preparazione un template della query viene inviato al server del database. Il server effettua un controllo della sintassi e alloca le risorse interne per un utilizzo successivo. Nella fase di esecuzione, il client associa i valori ai parametri e li invia al server. Il server crea un statement a partire dal template precedentemente memorizzato e i valori ad esso legato e lo esegue usando le risorse precedentemente allocate. Questo approccio e sicuro perché i valori legati ai parametri vengono inviati separatamente dalla query e perciò non possono interferire con essa. Il server usa questi valori direttamente al momento dell'esecuzione, successivamente alla valutazione del template. I parametri legati non hanno bisogno di essere sanificati in quanto non vengono mai sostituiti

direttamente nella stringa usata per la query. Al server potrebbe essere fornito un consiglio sul tipo di parametro in modo da creare una conversione appropriata.

3.9 Cross Site Request Forgery

Cross Site Request Forgery (CSRF) è un attacco nel quale la vittima viene indotta in maniera ingannevole a caricare una pagina contenente una richiesta dannosa. Tale richiesta eredita l'identità e i privilegi della vittima ed esegue un azione indesiderata per conto della vittima, come ad esempio cambiare l'indirizzo email della vittima, la sua password, o effettuare un acquisto. Gli attacchi CSRF solitamente mirano a funzioni che causano un cambiamento di stato sul server, ma possono anche essere usate per accedere a dati sensibili.

Per molti siti web, i browser includono automaticamente sulle richieste effettuate tutte le credenziali associate a quel sito, come ad esempio i cookie della sessione dell'utente, l'indirizzo IP ecc. Per questo motivo, se l'utente è autenticato sul sito, il sito non ha nessuna possibilità di distinguere una richiesta legittima da una indotta in maniera ingannevole.

Nella maggior parte dei casi questo attacco viene effettuato tramite l'inserimento di un tag *img* che invece di contenere l'indirizzo dell'immagine, contiene l'indirizzo dell'azione da eseguire, ad esempio:

```
<img src="http://store.example.org/buy.php?item=pencil&quantity=50" />
```

Quando il browser dell'utente vede il tag precedente, tenterà di visitare il link indicato per richiedere l'immagine. Se l'utente fosse loggato sul sito, molto probabilmente la visita a quel link comporterebbe l'acquisto di alcuni prodotti senza che l'utente ne sia consapevole.

Ancora una volta, questo è un esempio molto semplice, ma ovviamente si possono costruire richieste ben più pericolose.

Per difendersi da questi di attacco, bisognerebbe usare il metodo POST per l'invio dei dati, piuttosto che il generico REQUEST o il meno sicuro GET. Ovviamente questo è solo l'inizio. Un ulteriore misura di precauzione è quella di non creare pagine statiche per l'esecuzione di operazioni "importanti", come nell'esempio precedente la pagina buy.php che passandole un articolo e la quantità, senza fare ulteriori verifiche procede all'acquisto. Un'altra misura di protezione, che è anche quella più consigliata, è quella di appendere un token generato casualmente ad ogni richiesta. Ovviamente il token deve essere associato alla sessione dell'utente, altrimenti un attaccante potrebbe "catturare" un token valido e utilizzarlo per un attacco.

3.10 Session Hijacking

Per capire l'attacco Session Hijacking bisogna comprendere le connessioni persistenti.

3.10.1 Come funzionano le connessioni persistenti

Le comunicazioni HTTP sono state concepite per essere *stateless*, ovvero senza stato. Questo significa che una connessione tra due entità esiste solo per il breve lasso di tempo necessario per l'invio di una richiesta al server e la restituzione del risultato corrispondente al client. Una volta completato il trasferimento, le due entità non sono più consapevoli l'una dell'altra più di quanto lo fossero prima che la connessione avesse avuto luogo.

Oggi il web è evoluto e viene usato per eseguire transazioni, ad esempio effettuare acquisti. Un utente si iscrive, visualizza informazioni sui prodotti, sceglie i prodotti e così via, quindi la natura di queste operazioni diventa dipendente dallo stato delle operazioni precedenti.

Per risolvere questa mancanza di stato di HTTP furono sviluppate le sessioni. Una sessione è un pacchetto di informazioni relative a una transazione. Questo pacchetto tipicamente viene memorizzato sul server in un file temporaneo e li viene assegnato un ID, normalmente un numero casuale, più la data e l'ora in cui la sessione è stata creata. L'ID della sessione viene inviato al client alla prima risposta del server e il client lo allega ad ogni richiesta successiva inviata al server. Questo permette al server di accedere ai dati memorizzati e associati a quella sessione. Questo meccanismo fa sì che ogni transazione sia correlata logicamente alla precedente.

3.10.2 Le sessioni in PHP

PHP fornisce un supporto nativo per la gestione delle sessioni. La funzione *session_start()* inizializza il motore della sessione, il quale genera l'ID della sessione e lo salva nella costante PHPSESID. Il motore inizializza anche l'array superglobale $_SESSION nel quale si possono salvare tutte le informazioni desiderate. Queste informazioni vengono scritte su un file temporaneo sul server. Il file ha il nome dell'ID della sessione, perciò si può accedere ai dati finché si conosce l'ID della sessione.

Ci sono due modi per preservare l'ID della sessione attraverso le transazioni:

- Se i cookie sono abilitati sul client, allora viene scritto un cookie nel formato *name=value*, dove *name* è PHPSESID e *value* è il valore della costante, ovvero l'ID della sessione.
- Se i cookie non sono abilitati, allora PHP può essere configurato in modo da appendere una variabile a $_GET contenente la stessa stringa *nome=valore* ad ogni risposta. Questo metodo di funzionamento si chiama *Transparent Session ID*.

Quando uno script chiamato contiene l'istruzione session_start(), la prima verifica effettuata è sull'esistenza di una variabile $_COOKIE contenente il valore PHPSESID. Se l'ID della sessione non viene trovato in quel modo e se Transparent Session ID è abilitato, viene controllato se l'URI con il quale lo script viene chiamato contiene la variabile PHPSESID. Se l'ID della sessione viene recuperato, allora viene usato per accedere alle informazioni sulla sessione salvate sul server e tali informazioni vengono caricate sull'array $_SESSION. Se l'ID non viene trovato, allora viene generato un nuovo ID e viene creato un array $_SESSION vuoto.

3.10.3 Session Hijacking

Il Session Hijacking (Dirottamento di Sessione) è un attacco nel quale l'attaccante riesce ad impersonare un utente legittimo tramite l'utilizzo dell'identificatore di sessione. Il primo passo per ogni attaccante è quello di recuperare un identificatore di sessione valido, perciò la segretezza del ID è di primaria importanza. Una volta recuperato l'ID, l'attaccante può impersonare a tutti gli effetti un utente legittimo aggiungendo l'ID alle sue richieste. Il server non ha la possibilità di distinguere l'attaccante da un utente legittimo quindi li permetterà di eseguire tutte le operazioni che potrebbe eseguire l'utente legittimo. Ovviamente se viene recuperato l'ID di un utente che ha i privilegi di amministratore, l'attacco diventa particolarmente pericoloso.

Ci sono molti modi per recuperare un ID valido, ma i più comuni sono gli attacchi XSS e il cosiddetto MITM (Man In The Middle, ovvero l'uomo in mezzo). Quest'ultimo attacco si realizza tramite la cattura dei pacchetti che viaggiano tra il client e il server.

La prima difesa contro questo tipo di attacco è di rigenerare frequentemente l'ID della sessione. Questo si può realizzare tramite l'utilizzo della funzione *session_regenerate_id()* che garantisce un ID nuovo, invalidando quello precedente.

Come ulteriore misura di sicurezza bisogna rafforzare l'identificazione dell'utente. Per realizzare questo obbiettivo bisogna sfruttare tutte le informazioni che si hanno a disposizione. Gli unici dati disponibili sono quelli inclusi ad ogni intestazione dei pacchetti http, ad esempio:

```
GET / HTTP/1.1
Host: example.org
User-Agent: Firefox/1.0
Accept: text/html, image/png, image/jpeg, image/gif, */*
Cookie: PHPSESSID=1234
```

Bisogna riconoscere la consistenza nelle richieste e trattare come sospette le richieste inconsistenti. Ad esempio, nonostante la segnalazione dello User-Agent sia facoltativa, quasi tutti i client che scelgano di inviarlo non lo modificano durante l'utilizzo. Se l'utente identificato dall'ID 1234 sta usando il browser Mozilla Firefox dal momento del login, il passaggio ad un altro browser potrebbe generare un ragionevole sospetto. In questo caso sarebbe opportuno chiedere all'utente di rifare il login.

Sarebbe ragionevole pensare che se un attaccante riesce a recuperare l'ID della sessione, molto probabilmente sarebbe in grado di recuperare anche lo User-Agent dell'utente. Per questo motivo sarebbe opportuno valutare anche altre informazioni come ad esempio l'indirizzo IP dell'utente.

Sarebbe opportuno limitare anche il tempo di vita di una sessione valida, e questo riduce ulteriormente il rischio di un attacco in quanto si lascia meno tempo ad un attaccante per recuperare un ID valido.

Nella realizzazione di questa tesi, per garantire l'autenticità dell'ID vengono effettuati controlli sullo User-Agent, la lingua preferita richiesta dal browser, l'indirizzo IP dell'utente e il tempo di vita della sessione.

3.11 Salvataggio delle password

Nei casi in cui viene gestito correttamente l'input dell'utente e l'esecuzione delle query ad un database, un attaccante necessita di altri metodi per accedere agli account degli utenti. Generalmente si cercherebbe di ottenere le credenziali di accesso di qualche utente.

Un modo per ottenere le credenziali sarebbe quello di violare il database usato dalla applicazione. A seconda del database usato, dalla sua versione, di come è configurato ecc, esistono diversi modi per comprometterlo. Questo esula dagli obbiettivi di questa tesi, quindi dobbiamo assumere che un attaccante sia riuscito ad accedere al database. L'obbiettivo è cercare di minimizzare i danni.

Avendo accesso al database, un attaccante potrebbe utilizzare le credenziali degli utenti per fare qualsiasi cosa. Il fatto diventa più grave se si pensa che la maggior parte delle persone usa le stesse credenziali per accedere a tutti i siti.

Per minimizzare i danni, le password non andrebbero mai salvate in chiaro sul database, ma piuttosto dovrebbero essere salvate modificate in qualche maniera, in modo da non permettere ad un attaccante di risalire alla password originale. La soluzione più usata è quella di usare uno degli algoritmi di hashing per ottenere una stringa di caratteri, chiamata *hash*, a partire dalla password originale. Sul database viene salvato l'hash della password, invece della password. Per la verifica della password, si usa lo stesso algoritmo per creare l'hash della password fornita, si recupera l'hash salvato sul database e i due hash vengono confrontati. Se combaciano, allora siamo abbastanza sicuri che l'utente abbia fornito la password corretta.

Fino a poco tempo fa, l'algoritmo più usato era *MD5*, che permetteva di ottenere una stringa di 32 byte a partire da una stringa di qualsiasi lunghezza. Recentemente tale algoritmo è stato "violato" e tramite l'utilizzo di diversi metodi, ad esempio le cosiddette *tabelle rainbow*, si riesce a risalire alla stringa originale.

Per ottenere un buon hash, PHP ci viene in aiuto fornendo molto algoritmi di hashing come ad esempio *SHA256, SHA512, WHIRPOOL* ecc. Per quanto sia sicuro un algoritmo di hashing, non servirebbe a niente se l'utente usa una password debole. Uno studio recente ha scoperto che la password più usata attualmente è *123456*, seguita da *password*. È ovvio che il punto debole del sistema è l'essere umano. Per aumentare la sicurezza delle password, si potrebbero fare dei controlli in fase di registrazione e

forzare l'utente a inserire delle password che soddisfano alcuni criteri, come ad esempio usare numeri e segni di interpunzione in aggiunta a caratteri maiuscoli e minuscoli.

Per rendere più difficile per un attaccante trarre vantaggio dallo username e dall'hash della vittima, si fanno delle modifiche all'algoritmo di hashing in modo da rendere necessario il codice sorgente per scoprire la modifica. Tale modifica si chiama *salting* e consiste nell'aggiunta di una stringa (chiamata *salt*, ovvero sale) alla password dell'utente prima che venga generato l'hash. Grazie a questa modifica, diventa molto più difficile usare tecniche come i dizionari o le tabelle rainbow per risalire alla password originale. Per usa sicurezza maggiore, non si dovrebbe usare lo stesso sale per tutte le password, ma bisognerebbe generare uno nuovo per ogni password, e più e complesso il sale utilizzato, più sarà difficile risalire alla password dell'utente. Un buon metodo per generare il sale in PHP viene offerto dalla funzione *mcrypt_create_iv()* che passandole come parametri un intero e la costante *MCRYPT_DEV_URANDOM* creerà il sale della lunghezza specificata leggendo in modo casuale dei dati prodotti dal "rumore" dei driver del sistema operativo, quindi praticamente impossibili da indovinare.

Un'altra tecnica utilizzata per diminuire la possibilità di risalire alla password originale è quella di applicare più volte uno o più algoritmi di hashing. Questa tecnica si chiama password stretching.

Per la massima sicurezza possibile si dovrebbero usare in modo congiunto una buon algoritmo di hashing, un buon sale e applicare lo stretching della password. Un algoritmo che offre tutte queste funzionalità si chiama PBKDF2 (Password-Based Key Derivation Function 2) e fa parte degli standard PKCS (Public-Key Cryptography standards) concepiti e pubblicati da RSA Security Inc, a partire dagli anni 1990 [1]. L'algoritmo usa una funzione pseudocasuale, come ad esempio un hash crittografico, alla password data in input insieme al valore del sale e ripete il processo diverse volte per produrre una chiave derivata che potrà essere utilizzata come chiave crittografica nelle operazioni desiderate. L'algoritmo PBKDF2 viene considerato sicuro, anche se non perfetto, e viene utilizzato in diversi sistemi, come ad esempio nei protocolli WPA e WPA2 per proteggere le reti Wi-Fi, nei sistemi Mac OS X Mountain Lion per proteggere le password degli utenti, in Cisco IOS per generare gli hash delle password ecc.

PBKDF2 è stato implementato in PHP a partire dalla versione 5.5. Per permettere l'utilizzo di questo algoritmo anche dalle versioni precedenti di PHP, è stato scelto di usare una libreria open source che implementa questo algoritmo. La libreria è stata implementata in PHP puro da Taylor Hornby reperibile all'indirizzo https://crackstation.net/source/password-hashing/PasswordHash.php in modo gratuito ed è libera da utilizzare e modificare.

4. Database

4.1 Database e motore di salvataggio.

Per la realizzazione del back-end abbiamo la necessità di appoggiarci ad un database per la memorizzazione delle informazioni. Per la realizzazione di questa tesi è stato utilizzato il database MySQL. Mysql permette di scegliere il motore di salvataggio (Storage Engine) che si vuole utilizzare. I due motori più utilizzati sono MyISAM e InnoDB. Per questa tesi è stato scelto di utilizzare InnoDB come motore del salvataggio. La differenza principale tra i due è che InnoDB supporta le transazioni, a differenza di MyISAM che non le supporta. Perché le transazioni operino in modo corretto sui dati è necessario che i meccanismi che le implementano soddisfino le proprietà ACID (Atomicity, Consistency, Isolation e Durability) [1]:

- Atomicity (atomicità) - La transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere o totale o nulla, non sono ammesse esecuzioni parziali.
- Consistency (coerenza) - Quando inizia una transazione il database si trova in uno stato coerente e quando la transazione termina il database deve essere in un altro stato coerente, ovvero non deve violare eventuali vincoli di integrità, quindi non devono verificarsi contraddizioni (*inconsistency*) tra i dati archiviati nel DB.
- Isolation (isolamento) - Ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione.
- Durability (persistenza) - Una volta che una transazione abbia richiesto un *commit work*, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log dove sono annotate tutte le operazioni sul DB.

Ulteriori differenze tra i due motori di salvataggio sono:

- Per riparare una tabella dopo un crash del sistema, InnoDB riesegue le ultime istruzioni registrate nei log. MyISAM deve invece eseguire una scansione completa della tabella per poi ripararla, ed eventualmente ricostruire gli indici. Di conseguenza, il tempo impiegato da InnoDB per la riparazione non aumenta

con il crescere dei dati contenuti nella tabella, mentre il tempo impiegato da MyISAM è proporzionale alle dimensioni della tabella.

- InnoDB ha una sua propria gestione della cache. Le pagine di dati modificate non vengono inviate immediatamente al sistema e questo, in alcuni casi, può rendere la modifica dei dati molto più rapida con InnoDB.
- MyISAM generalmente immagazzina i record di una tabella nell'ordine in cui sono stati creati, mentre InnoDB li immagazzina nell'ordine seguito dalla chiave primaria. Quando viene utilizzata la chiave per la lettura di una riga, l'operazione avviene più rapidamente.
- InnoDB comprime i record molto meno rispetto a MyISAM. Questo significa che la memoria e lo spazio su disco richiesti da InnoDB sono maggiori, nonostante nella versione 5 di MySQL lo spazio su disco richiesto sia diminuito del 20%.
- Allo stato attuale, InnoDB non supporta le ricerche fulltext.

Pare evidente che i vantaggi offerti da InnoDB sono di gran lunga superiori agli svantaggi, ed è per questo che è stato scelto.

4.2 Studio e creazione delle tabelle.

Durante la fase di analisi sono state identificate tre entità: User, Esercizio e File che sono così composte:

4.2.1 L'entità User

La prima entità identificata e l'entità *User*, nella quale vengono salvati: l'indirizzo email dell'utente che verrà usato anche come username, nome e cognome dell'utente, il suo numero di matricola e la password. Ad ogni utente viene assegnato un ID unico che servirà per distinguere eventuali omonimi. È stato anche aggiunto un campo chiamato livello, che servirà per distinguere gli utenti semplici dagli amministratori. Per permettere una maggiore flessibilità dell'applicazione rispetto a possibili futuri cambiamenti, è stato scelto di non usare un booleano per il livello di accesso, ma piuttosto un numero intero. Questo permetterebbe di gestire in modo diverso gli utenti, ad esempio, non permettere ad un utente registrato l'esecuzione di alcune operazioni fino al momento dell'attivazione del suo account, oppure impostare un utente come non attivo ecc. La chiave primaria è composta dai campi ID ed email. La query usata per creare la tabella è la seguente:

```
create table User (
    id int(11) unsigned NOT NULL AUTO_INCREMENT,
    email varchar(40) unique NOT NULL,
    nome varchar(40),
    cognome varchar(40),
    matricola int(10),
    password varchar(130),
    livello int,
    primary key (id, email)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Query pe la creazione della tabella User

Lo schema della tabella risultante sarà:

id	email	nome	cognome	matricola	password	livello

4.2.2 L'entità Esercizio

La seconda entità identificata è *Esercizio*. Con il termine esercizio ci riferiamo ad un modello di esercizio e non all'esercizio sottomesso da uno studente. Quando il docente carica sul sito l'esercizio da svolgere li assegna un id per distinguerlo dagli altri. Per ogni esercizio viene caricato un template che è un file contenente dei dati che servono

per testare l'esercizio svolto dallo studente, e i risultati attesi in output. Tutti i template vengono salvati in una cartella. Per permettere una maggiore flessibilità, si è scelto di salvare nel database solo il nome del template (ovvero solo il nome del file). Questa scelta si basa sul fatto che esiste la possibilità che venga modificata la cartella contenente i template. In tal caso, non si avrebbe la necessità di aggiornare tutti i dati della tabella per rispecchiare il nuovo stato, ma basterebbe solo cambiare il percorso della cartella contenente i template (tale percorso è memorizzato nella classe che gestisce la connessione al database, che viene descritto nel capitolo successivo). Ad ogni esercizio viene assegnato un punteggio. Il risultato è una tabella molto semplice e la query per crearla è la seguente:

```
create table Esercizio (

    id int(11) unsigned NOT NULL,
    templateName varchar(50),
    punteggio int,
    primary key (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Query per la creazione della tabella Esercizio

Lo schema risultante è il seguente:

id	templateName	punteggio

4.2.3 L'entità File

L'ultima entità identificata è *File*. Ogni studente sottomette uno o più file che sono la soluzione dell'esercizio proposto dai docenti. Viene data allo studente la possibilità di caricare più file per un dato esercizio in quanto una soluzione proposta potrebbe non essere corretta, ma si vuole comunque mantenere uno storico dei file caricati da ogni studente. Per ogni file viene memorizzato un id, l'id dell'utente che l'ha caricato, l'id dell'esercizio a cui fa riferimento, il nome del file caricato, l'estensione del file, l'eventuale punteggio guadagnato con la sottomissione di tale file, un commento sul file caricato, data e ora in cui il file è stato caricato. È stato scelto di salvare l'estensione del file in un campo del database in quanto uno studente potrebbe sottomettere diversi tipi di esercizi, quindi tale campo serve per filtrare la visualizzazione dei file caricati in base alla loro estensione. Il campo commento serve per memorizzare l'esito della correzione

e verrà usato per dare un feedback immediato allo studente all'atto della sottomissione dell'esercizio, nonché per permettere ai docenti di scoprire i punti deboli degli studenti, quindi distribuire nel modo migliore possibile il numero di ore assegnate alle lezioni.

Per permettere il controllo sul tempo di consegna, in modo da scartare gli esercizi consegnati fuori tempo, è stato creato il campo data, che in realtà e ti tipo *datetime* e quindi permette di memorizzare anche l'ora di consegna. Tale scelta è stata adottata per permettere un ordinamento più corretto dei file caricati, oppure filtrare i dati in modo più preciso, ad esempio visualizzare l'ultimo file caricato.

La query per creare la tabella è la seguente:

```
create table File (

    id int(11) unsigned NOT NULL AUTO_INCREMENT,
    user_id int(11) unsigned,
    es_id int(11) unsigned,
    file_name varchar(50),
    estensione varchar(10),
    valutazione int,
    valutazione_commento varchar(1000),
    data datetime NOT NULL,
    primary key (id, user_id, es_id),
    foreign key (user_id) references User(id),
    foreign key (es_id) references Esercizio(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Query per la creazione della tabella File

Lo schema della tabella risultante è il seguente:

id	user_id	es_id	file_name	estensione	valuazione	commento	data

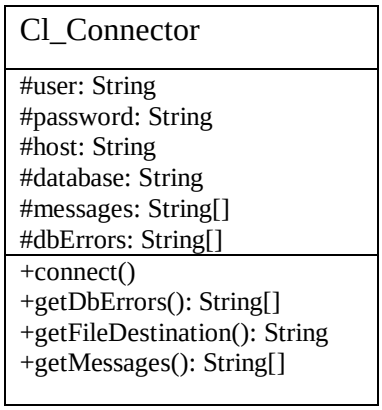
5. Implementazione degli script

Per la realizzazione di questa tesi è stato usato un mix di programmazione procedurale e programmazione ad oggetti. Gli script sono stati divisi in base alle funzionalità, in modo da permettere una migliore riusabilità e manutenibilità del codice.

5.1 Connessione al database

Trattandosi di una web database application, è ovvio che questa applicazione è fortemente legata al database. Per questo motivo, si è scelto di iniziare con l'implementazione delle funzionalità legate al database. Per semplificare il compito, eventuali dati necessari, che normalmente dovrebbero essere fornite dagli utenti, sono stati creati ad hoc e sono salvati su delle variabili.

Per la connessione al database e stata utilizzata l'estensione PDO. È stata implementata una classe chiamata *Cl_Connector* che si occupa appunto della connessione al database e fornisce dei *getter* di alcune variabili protette. La classe Cl_Connector è una classe base e le altre classi che dovranno fare operazioni sul database, estenderanno questa classe. È stata adottata questa scelta in modo da definire i parametri della connessione in un unico punto. Nell'eventualità di un cambiamento dei parametri della connessione, ad esempio il nome del database o la password, basterebbe aggiornare questa classe e tutto continuerebbe a funzionare senza altre modifiche. Il diagramma della classe è il seguente:



Implementazione della classe Cl_Connector:

```
class Cl_Connector
{
    protected $_user = '****';
    protected $_password = '****';
    protected $_host = '****';
    protected $_database = '****';
    protected $_messages = array();
    protected $_dbErrors = array();
    protected $_fileDestination = 'C:\\uploads\\';
    protected $_templateDestination = 'C:\\uploads\\templates\\';
    public function connect()
    {
        try {
            $db = new PDO("mysql:host=$this->_host;dbname=$this->_database",
$this->_user, $this->_password);
            $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $db;
        } catch (PDOException $e) {
            $this->_dbErrors[] = $e->getMessage();
            die();
        }
    }
    public function getDbErrors(){
        return $this->_dbErrors;
    }
    public function getFileDestination(){
        return $this->_fileDestination;
    }
    public function getMessages(){
        return $this->_messages;
    }
}
```

La classe Cl_Connector

Come si vede anche dal codice, e stato impostato che eventuali errori di connessione o errori generati dal database, generino delle eccezioni, in questo modo si evita di rivelare informazioni sul sistema sottostante. Ovviamente le eccezioni vano gestite in modo appropriato e serviranno al amministratore per le fasi di test o per il debug del codice, ma agli utenti verranno stampati dei messaggi che non trapelano informazioni sulla applicazione.

Gli array `$_messages` e `$_dbErrors` servono per memorizzare rispettivamente gli esiti delle operazioni sul database e gli errori generati dal database (ad esempio dall'esecuzione di una query sbagliata).

In questa classe vengono anche memorizzati i percorsi delle cartelle che manterranno i file caricati dagli utenti e i template che saranno usati dal modulo della correzione automatica.

Dato che tutti gli attributi sono protetti, sono stati implementati i relativi getter, eccezione fatta, ovviamente, per le credenziali di accesso.

5.2 Operazioni sul database

Per effettuare le modifiche sul database è stata implementata la classe Cl_dbOperations. Le funzionalità identificate sono: l'inserimento di un nuovo utente, l'inserimento di un nuovo file, controllo sull'esistenza di un dato indirizzo email o numero di matricola e il recupero dei dati di un utente. La classe Cl_dbOperations estende la classe Cl_Connector.

Il diagramma della classe Cl_dbOperations è il seguente:

Cl_dbOperations
+getUserData(email: String): String[] #checkExistenceMatrEmail(matricola: Integer, email: String): Boolean +insertUser(email: String, nome: String, cognome: String, matricola: Integer, password: String) +insertFile(userId: Integer, esercizioId: Integer, fileName: String, estensione: String, valutazione: String, commento: String): Boolean

5.2.1 Il metodo getUserData

L'implementazione del metodo getUserData è la seguente:

```php
public function getUserData($email) {
        if ($connection = $this->connect()) {
                $result = array();
                $query = "SELECT id, email, password, livello FROM User WHERE email =
:email;";
                $stmt = $connection->prepare($query);
                $stmt->bindParam(':email', $email, PDO::PARAM_STR);
                $stmt->execute();
                $count = $stmt->rowCount();
                if ($count > 0) {
                        $row = $stmt->fetch();
                        $result[] = $row['id'];
                        $result[] = $row['email'];
                        $result[] = $row['password'];
                        $result[] = $row['livello'];
                        return $result;
                }
                else {
                        return false;
                }
        }
        else {
                $this->_messages[] = "Impossibile connettersi al database";
                return false;
        }

        unset($stmt);
        unset($connection);
}
```

Il metodo getUserData()

Il metodo prende in input un indirizzo email che nel nostro caso viene usato come username e restituisce un array che contiene id, email, password e livello dell'utente, se tale utente esiste sul database, false se l'utente non è presente sul database. Verrà utilizzata al momento del login, per recuperare i dati dell'utente.

5.2.2 Il metodo insertUser

Il metodo *insertUser()*, come è evidente dal nome, serve per inserire i dati di un utente nella tabella User del database. Prima di inserire un utente vengono fatti dei controlli per garantire la consistenza dei dati ed evitare la generazione di errori dal database. Viene quindi controllato che l'email o la matricola dell'utente non esistano già nel database. Tale controllo viene effettuato dal metodo *checkExistenceMatrEmail()* che è così implementato:

```
protected function checkExistenceMatrEmail($matricola, $email) {
	if ($conn = $this->connect()) {
		$query = "SELECT COUNT(*) FROM User WHERE email = :email OR
matricola = :matricola";
		$stmt = $conn->prepare($query);
		$stmt->bindParam(':email', $email);
		$stmt->bindParam(':matricola', $matricola);
		$stmt->execute();
		$numRows = $stmt->fetchColumn();
		unset($conn);
		unset($stmt);
		if ($numRows > 0) {
			$this->_messages[] = "Email o matricola esistente.";
			return true;
		}
		else {
			return false;
		}
	}
	else {
		$this->_messages[] = "Impossibile connettersi al database";
		return true;
	}
}
```

Il metodo checkExistenceMatrEmail()

Il metodo restituisce *true* se l'email o la matricola sono già presenti sul database, *false* altrimenti. Nel caso non sia possibile connettersi al database, viene comunque restituito *true*, anche se non si è potuto verificare tale risultato, perché esiste la possibilità che questi dati siano presenti sul database quindi abbiamo preferito evitare una possibile collisione che porterebbe alla generazione di eccezioni e dunque potrebbe rilevare informazioni sul database.

Una volta verificato che l'utente non sia già presente nel database, viene inserito tramite il metodo *insertUser()* che è stato implementato nel seguente modo:

```
public function insertUser($email, $nome, $cognome, $matricola, $password) {
	$exists = $this->checkExistenceMatrEmail($matricola, $email);
	if (!$exists) {
		try {
			$conn = $this->connect();
			$query = "INSERT INTO User values (NULL, :email, :nome,
:cognome, :matricola, :password, 0)";
			$stmt = $conn->prepare($query);
			$stmt->bindParam(':email', $email, PDO::PARAM_STR);
			$stmt->bindParam(':nome', $nome, PDO::PARAM_STR);
			$stmt->bindParam(':cognome', $cognome, PDO::PARAM_STR);
			$stmt->bindParam(':matricola', $matricola, PDO::PARAM_INT);
			$stmt->bindParam(':password', $password, PDO::PARAM_STR);
			if ($stmt->execute()) {
				$id = $conn->lastInsertId();
				$this->_fileDestination .= $id . '/';
				if (!is_dir($this->_fileDestination)) {
					mkdir($this->_fileDestination);
				}
				$this->_messages[] = "Utente inserito correttamente";
			}
			else {
				$this->_dbMessages[] = "L'inserimento ha fallito con
errore: " . $stmt->errorCode();
			}
		}
		catch (Exception $e) {
			$this->_messages[] = "Errore di connessione al database.";
		}
	}
}
```

Il metodo insertUser()

Il metodo *insertUser()* prende in input email, nome, cognome, matricola e password dell'utente e scrive nell'array `$_messages` l'esito dell'inserimento. Ogni utente viene inserito con livello 0, ovvero studente.

Un altro compito eseguito da questo metodo è quello di creare la cartella dove andranno salvati i file caricati da quell'utente in modo da avere i file separati per ogni utente. Alla cartella creata viene dato come nome l'id dell'utente che viene recuperato tramite il metodo *lastInsertId()* implementato in PDO. Il metodo *lastInsertId()* è supportato anche da MySQL e quindi ci viene garantito un id consistente, anche in caso di connessioni multiple al database.

5.2.3 Il metodo insertFile

Il metodo *insertFile()* inserisce nella tabella File i dati riguardanti a un file caricato. Il metodo prende in input l'id dell'utente che l'ha caricato, l'id dell'esercizio a cui fa riferimento quel file, il nome del file, l'estensione, il voto assegnato alla soluzione proposta, un commento sulla valutazione, data e ora dell'inserimento. Il metodo restituisce *true* se l'inserimento è avvenuto con successo, *false* altrimenti. L'implementazione del metodo è la seguente:

```
public function insertFile($userId, $esercizioId, $fileName, $estensione,
$valutazione, $commento) {
	try {
		$conn = $this->connect();
		$query = "insert into File values(NULL, :user_id, :es_id,
:file_name, :estensione, :valutazione, :commento, NOW())";
		$stmt = $conn->prepare($query);
		$stmt->bindParam(':user_id', $userId, PDO::PARAM_INT);
		$stmt->bindParam(':es_id', $esercizioId, PDO::PARAM_INT);
		$stmt->bindParam(':file_name', $fileName, PDO::PARAM_STR);
		$stmt->bindParam(':estensione', $estensione, PDO::PARAM_STR);
		$stmt->bindParam(':valutazione', $valutazione, PDO::PARAM_INT);
		$stmt->bindParam(':commento', $commento, PDO::PARAM_STR);
		if ($stmt->execute()) {
			unset($conn);
			unset($stmt);
			return true;
		}
		else {
			$error = $stmt->errorInfo();
			if (isset($error[2])) {
				$this->_dbMessages[] = $error[2];
			}
			unset($conn);
			unset($stmt);
			return false;
		}
	}
	catch (Exception $e) {
		$this->_dbMessages[] = $e->getMessage();
		return false;
	}
}
```

Il metodo insertFile()

In caso di errori di inserimento o di connessione al database, viene scritto su `$_dbMessages` il codice di errore o l'eventuale eccezione generata.

5.3 Gli script di registrazione e login

Dopo l'implementazione delle funzionalità necessarie per eseguire le operazioni sul database, si è potuto procedere con gli script che gestiscono la registrazione e il login degli utenti.

5.3.1 Gestione della registrazione

Lo script che gestisce la registrazione viene chiamato dalla pagina *registrazione.php* nella quale è presente una form che contiene tutti i campi input necessari per l'inserimento dei dati da parte degli utenti. I dati vengono passati tramite il metodo POST in modo da rendere più difficile l'inserimento di dati fasulli o di metacaratteri, misura necessaria questa per aumentare la sicurezza della applicazione.

```
$expected = array('nome' => 40, 'cognome' => 40, 'matricola' => 10,
                  'password' => 50, 'cpassword' => 50, 'email' => 40);
$numElementi = count($expected);
$valoriOk = 0;
$missing = array();
foreach ($expected as $key => $dim) {
        if (empty($_POST[$key])) {
                $missing[] = $key;
                continue;
        }
        switch ($key) {
                case 'nome':
                case 'cognome':
                case 'email':
                        if (is_string($_POST[$key]) && strlen($_POST[$key]) <= $dim) {
                                ${$key} = strip_input($_POST[$key]);
                                $valoriOk++;
                        }
                        else
                                $missing[] = $key;
                        break;

                case 'matricola':
                        if (is_numeric($_POST[$key]) && strlen($_POST[$key]) <= $dim) {
                                ${$key} = $_POST[$key];
                                $valoriOk++;
                        }
                        else
                                $missing[] = $key;
                        break;
                case 'password':
                case 'cpassword':
                        if (is_string($_POST[$key]) && strlen($_POST[$key]) <= $dim) {
                                ${$key} = $_POST[$key];
                                $valoriOk++;
                        }
                        else
                                $missing[] = $key;
                        break;
        }
}
```

La parte dello script che effettua i controlli sulle variabili passate ad esso.

Nello script è memorizzato un array dove sono elencati i parametri attesi in input e la loro dimensione massima. Per motivi di sicurezza vengono presi da POST solo i valor attesi e per ognuno di essi vengono effettuati controlli sulla dimensione e il tipo.

Se i dati sono del tipo e della dimensione corretta, vengono sanificati tramite la funzione *strip_input()* che è implementata nel seguente modo:

```
function strip_input($data) {
        $data = trim($data);
        $data = str_replace(' ', '_', $data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;
}
```

La funzione strip_input()

La funzione *trim()* toglie eventuali spazi all'inizio ed alla fine della stringa, mentre tramite l'utilizzo della funzione *str_replace()* vengono sostituiti eventuali spazi nel corpo della stringa con il carattere underscore. Queste operazioni vengono effettuate per evitare l'inserimento di comandi nei dati.

La funzione *stripslashes()* toglie eventuali slash presenti nella stringa, mentre la funzione *htmlspecialchars()* converte i caratteri pericolosi in entità HTML. Queste funzioni vengono usate per evitare attacchi XSS o l'inserimento di query SQL nei dati.

I dati che passano i controlli, dopo essere sanificati vengono salvati in delle variabili, mentre i dati mancanti o che non passano i controlli vengono segnalati come mancanti, quindi viene invitato l'utente a tornare alla pagina di registrazione e di fornire dati corretti.

Quando un utente ha fornito tutti i dati corretti si procede con la registrazione. Si controlla che la password e la conferma della password combacino. Se le password combaciano si prosegue, in caso contrario lo script si ferma e l'utente viene avvisato del fatto che le password fornite non combaciano.

Se le password combaciano, viene istanziato un oggetto della classe Cl_dbOperations, e tramite il metodo insertUser() vengono inseriti i dati dell'utente nel database e viene informato l'utente sull'esito della registrazione.

5.3.2 Le password

Per i motivi descritti nel capitolo sulla sicurezza, le password non andrebbero mai salvate in chiaro sul database, ma piuttosto bisognerebbe criptarle usando un algoritmo di hashing, e salvare l'hash della password nel database.

Per il salvataggio delle password viene utilizzata una libreria open source che implementa l'algoritmo PBKDF2, il quale a sua volta usa sha256 iterato 1000 volte per generare l'hash della password (si possono comunque modificare l'algoritmo usato e il numero delle iterazioni per renderle ottimali per le proprie necessità). Alla password fornita dall'utente viene aggiunto un sale prodotto dalla lettura di dati casuali prodotti dal rumore dei driver del sistema operativo. Sul database viene salvato l'hash della password e il sale utilizzato, tutto convertito in base 64.

5.3.3 Gestione del login

Lo script che gestisce il login viene chiamato da una form nella pagina del login e li vengono passati tramite POST l'email e la password fornita dall'utente. Viene controllato che i dati non siano vuoti, e in tal caso viene sanificata la variabile email per poi essere usata per recuperare i dati dell'utente, se presenti sul database. Nello specifico vengono recuperati l'id, l'hash della password e il livello di accesso dell'utente.

La libreria utilizzata per creare l'hash delle password fornisce anche una funzione chiamata *validate_password()* che serve per verificare la correttezza di una password. La funzione prende in input la password da verificare e una stringa contenente l'hash della password corretta più il sale utilizzato e restituisce *true* se la password fornita è corretta, *false* altrimenti.

Data la particolarità e la complessità della tecnica utilizzata per criptare la password, il compito della verifica della password non può essere fatto direttamente nella query al database come succede di solito, ovvero eseguire una query del tipo "`SELECT * FROM User WHERE username='utente' AND password='1234';`".

Il controllo corretto può essere fatto solo dalla funzione validate_password(), quindi per verificare la validità della password inserita si dovrebbe recuperare dal database l'hash della password corrispondente ad un dato utente e passare i dati alla funzione validate_password() per avere una risposta attendibile.

Se l'utente ha fornito email e password corrette, li viene dato l'accesso al sito. Per motivi di sicurezza, ovvero per limitare il pericolo di Session Hijacking, viene rigenerato l'ID della sessione e vengono salvate nell'array $_SESSION l'ID dell'utente, il suo livello di accesso, l'indirizzo email e viene impostata a true la variabile auth che serve per indicare che l'utente è autenticato.

```
$password = $_POST['password'];
$email = strip_input($_POST['email']);
if (strlen($email) > 40 || strlen($password) > 50) {
	exit('email o password troppo lunga.');
}
if (!($email && $password)) {
	exit('I campi email e password non devono essere vuoti.');
}
else {
	try {
		$db = new Cl_dbOperations();
		$result = $db->getUserData($email);
		if ($result[1] == $email) {
			if (verifica_password($password, $result['2'])) {
				session_regenerate_id();
				$_SESSION['auth'] = true;
				$_SESSION['userId'] = strip_input($result[0]);
				$_SESSION['email'] = strip_input($result[1]);
				$_SESSION['livello'] = strip_input($result[3]);
				$profile['time']	= time();
				$profile['userAgt'] =
md5($_SERVER['HTTP_USER_AGENT']);
				$profile['lang']	=
md5($_SERVER['HTTP_ACCEPT_LANGUAGE']);
				$profile['ipAddr']	= md5($_SERVER['REMOTE_ADDR']);
				$profileDir = $db->getFileDestination();
				$profileDir .= $result[0] . '\\';
				$profileFile = $profileDir . 'profile.txt';
				file_put_contents($profileFile, serialize($profile));

				echo "<p>Login effettuato.</p>";
				echo "<p>A breve verrai indirizzato alla pagina
iniziale</p>";
				header('Refresh: 2; URL=../index.php');
			}
			else {
				echo "<p>Email o Password Sbagliata</p>";
			}
		}
		else {
			echo "<p>Email o Password Sbagliata</p>";
		}
	}
	catch (Exception $e) {
		echo "<p>Impossibile connettersi al database.</p>";
		die();
	}
}
```

Lo script che gestisce il login dell'utente.

Per motivi di sicurezza, anche i dati recuperati dal database vengono sanificati perché esiste il pericolo, seppur piccolo, che qualcosa sia sfuggita ai controlli precedenti oppure che l'utente sia riuscito in qualche modo a inserire dati compromessi nel database.

Per aumentare ulteriormente la sicurezza, in modo da garantire con più sicurezza l'autenticità dell'utente, viene creato un profilo utente. Il profilo utente è un file salvato nella cartella degli upload dell'utente sul quale vengono salvate data e ora di accesso, lo User Agent dell'utente, il linguaggio di default del browser dell'utente e il suo indirizzo IP. Questo profilo verrà poi usato nelle sezioni più delicate del sito, come ad esempio nella pagina di caricamento di un file. Per garantire che l'utente sia legittimo e correttamente loggato sul sito, si andrà a verificare la variabile $_SESSION['auth'], che deve essere impostata a true, in maniera congiunta con i dati del profilo. Inoltre, le sessioni più vecchie di dieci minuti saranno considerate scadute e l'utente verrà rimandato alla pagina di login.

5.3.4 gestione del logout

Per effettuare correttamente il logout dell'utente, viene prima azzerato l'array $_SESSION assegnandoli in array vuoto, poi viene reso invalido il cookie della sessione assegnandoli un valore vuoto e impostando il tempo di scadenza a un tempo precedente, infine viene distrutto il file locale che tiene le informazioni sulla sessione attuale tramite session_destroy(). L'utente viene rimandato nella pagina di login.

```
session_start();
if (isset($_SESSION['auth'])) {
        $_SESSION = array();
        if (isset($_COOKIE[session_name()])) {
                setcookie(session_name(), '', time() - 8640, '/');
        }
        session_destroy();
        header('Location: ../pages/login.php');
        exit;
}
```

Gestione del logout.

5.4 Gestione degli upload

Per gestire gli upload dei file è stata implementata la classe Cl_Upload. La classe è stata concepita per essere il più indipendente possibile nonché gestire al meglio gli aspetti concernenti la sicurezza. Prima di descrivere in dettaglio la classe implementata, descriviamo brevemente come funziona l'upload di un file in PHP.

5.4.1 L'upload dei file in PHP

In PHP un file può essere inviato solamente utilizzando il metodo POST. Quando il file viene ricevuto dal server, viene salvato temporaneamente nella cartella Temp e viene generato la variabile globale $_FILES. Tale variabile è un array associativo, che per semplicità possiamo vedere come un array bidimensionale, dove il primo indice indica sempre il nome del campo input che ha caricato il file, mentre il secondo indice indica un attributo del file che può essere uno tra: name, type, size, tmp_name o error. Name serve per indicare il nome originale del file, type per indicare il MIME type del file, size per la dimensione del file, tmp_name per il nome temporaneo assegnato al file, mentre error per indicare l'esito del caricamento del file. Il file caricato esiste fino al termine dello script quindi per poterlo utilizzare in un secondo momento si deve salvare in modo definitivo. Per fare questo si dovrebbe usare la funzione move_uploaded_file() che prende in input il nome del file temporaneo e il percorso dove andrà salvato il file (compreso il nome da assegnare al file). Ad esempio, se abbiamo una form del tipo:

```
<form enctype="multipart/form-data" action="script.php" method="post" >
    <input type="file" name="esercizio"/>
    <input type="submit" value="Upload" name="upload" />
</form>
```

e carichiamo il file *prova.c*, in `$_FILES['esercizio']['name']` troviamo il nome originale del file, ovvero *prova.c*, mentre per salvare il file in maniera definitiva chiamandolo ad esempio *nomeFile.c* e scegliendo come destinazione la cartella C:\uploads\ si potrebbe usare un comando simile:

```
move_uploaded_file($_FILES['esercizio']['tmp_name'], 'C\\uploads\\nomeFile.c');
```

5.4.2 La classe Cl_Upload

Il diagramma della classe Cl_Upload è il seguente:

Cl_Upload
#uploaded: String[] #destination: String #max: Integer #messages: String[] #permitted: String[] #permittedExtensions: String[] #renamed: Boolean #filePath: String #filename: String #fineNameLength: String #extension: String #uploadStatus: Boolean
+<<constructor>>construct(path: String) +getMaxSize(): String +getUploadStatus(): String +getFileName(): String +getExtension(): String +getMessages(): String[] +move(overwrite: Boolean) #checkError(filename: String, error: Integer): Boolean #checkSize(fileName: String, size: Integer): Boolean #checkType(fileName: String, type: String[]): Boolean #checkExtension(filename: String): Boolean #checkName(name: String, overwrite: Boolean): String #checkFileNameLength(name: String): Boolean +strip_input(data: String): String

Descriviamo brevemente gli attributi della classe:

L'attributo uploaded è un array utilizzato per rendere la classe indipendente dal nome del campo input che invia il file, destination serve per memorizzare il percorso della cartella dove saranno salvati i file, max serve per memorizzare la dimensione massima ammessa per un singolo file. mesages è un array sul quale verranno salvati gli esiti delle varie operazioni svolte dalla classe, permitted è un array sul quale vengono salvati i tipi dei file permessi per l'upload, permittedExtensions serve per memorizzare le estensioni permesse, renamed serve per memorizzare l'evento "il file è stato rinominato", filePath serve per memorizzare il percorso dove andrà salvato il file, filename serve per memorizzare il nome del file, fileNameLength è la dimensione massima permessa per il nome del file, extension serve per memorizzare l'estensione del file caricato, uploadStatus è un booleano che serve per memorizzare l'esito del caricamento del file.

5.4.3 Il costruttore

Il costruttore della classe prende in input il percorso della cartella dove verrà salvato il file e verifica che la cartella esista e che si abbiano i permessi per scrivere in tale cartella. Se la cartella non esiste o non si hanno i permessi di scrittura, viene generata un eccezione, e l'utente viene avvisato con un messaggio, altrimenti si salva nell'attributo destination il percorso della cartella dell'utente, inoltre vengono memorizzati nell'array uploaded i contenuti della variabile globale $_FILES.

```
public function __construct($path) {
	if (!is_dir($path) || !is_writable($path)) {
		throw new Exception("Non si hanno i permessi neccessari per scrivere
nella cartella di download!");
	}
	$this->_destination = $path;
	$this->_uploaded = $_FILES;
}
```

Il costruttore della classe CI_Uploads()

Prima di salvare il file caricato in maniera definitiva, si fanno dei controlli per garantire che il file sia sicuro.

5.4.4 Il metodo checkError

Il primo controllo effettuato è sullo stato del caricamento del file. Quando un file viene caricato, PHP salva l'esito del caricamento nella variabile `$_FILES['esercizio']['error']`. Il codice 0 indica che il file è stato caricato correttamente.

```
protected function _checkError($filename, $error) {
	switch ($error) {
		case 0:
			return true;
		case 1:
		case 2:
			$this->_messages[] = "Il file '" . strip_input($filename) .
"' eccede la dimensione massima: " . $this->getMaxSize();
			return false;
		case 3:
			$this->_messages[] = "Errore nel caricare il file.";
			return false;
		case 4:
			$this->_messages[] = "Nessun file selezionato.";
			return false;
		default:
			$this->_messages[] = "Errore del sistema nel caricare il
file. Contattare l'amministratore.";
			return false;
	}
}
```

Il metodo checkError()

5.4.5 Il metodo checkSize

Il secondo controllo da effettuare è sulla dimensione del file caricato. Solitamente la dimensione massima ammessa si mette in un campo hidden nella form di upload, ma non ci possiamo fidare ai controlli lato client in quanto sono facilmente superabili. Per questo motivo dobbiamo effettuare un controllo sul server. Il metodo prende in input il nome del file e la dimensione del file caricato che si trova in `$_FILES['esercizio']['size']` e restituisce true se i file è della dimensione ammessa, false altrimenti.

```
protected function _checkSize($filename, $size) {
	if ($size == 0)
		return false;
	elseif ($size > $this->_max) {
		$this->_messages[] = "Il file eccede la dimensione massima: " .
$this->getMaxSize();
		return false;
	}
	else
		return true;
}
```

Il metodo checkSize()

5.4.6 Il metodo checkType

Il terzo controllo effettuato è sul MIME type del file caricato. Il metodo prende in input il nome del file e il MIME type che il browser ha comunicato al server e si trova nella variabile `$_FILES['esercizio'][type]`. Il metodo usa i contenuti dell'attributo *permitted* sul quale sono elencati i tipi permessi. Per una maggiore flessibilità, *permitted* è un array, quindi se si vuole aggiungere uno o più tipi di file permessi, basterà aggiungerli nell'attributo *permitted*.

```
protected function _checkType($filename, $type) {
	if (empty($type)) {
		return false;
	}
	elseif (!in_array($type, $this->_permitted)) {
		$this->_messages[] = "Il file è di un tipo non permesso.";
		return false;
	}
	else
		return true;
}
```

Il metodo checkType()

5.4.7 Il metodo checkExtension

Il quarto controllo effettuato è sull'estensione del file caricato. Dato che anche il MIME type può essere modificato da un malintenzionato, il controllo sull'estensione è una ulteriore misura di sicurezza. Inoltre, la maggior parte dei file caricati saranno file sorgente che dovranno essere compilati e testati. Il compilatore GCC, che è il compilatore che verrà usato per compilare i file scritti in linguaggio C, non compila i file che non hanno l'estensione, quindi il controllo sull'estensione dei file risulta fondamentale. Il metodo prende in input il nome del file e lo confronta con i valori presenti nel attributo permittedExtensions. Anche in questo caso, le estensioni permesse sono memorizzate in un array, in modo da facilitare la modifica delle estensioni permesse. Il metodo restituisce true se l'estensione è permessa, false se assente o non permessa.

```
protected function _checkExtension($fileName) {
	$dot = strrpos($fileName, '.');
	if ($dot) {
		$extension = substr($fileName, $dot);
		if (in_array($extension, $this->_permittedExtensions)) {
			$this->_extension = $extension;
			return true;
		}
		else {
			$this->_messages[] = "Il file ha un estensione non permessa";
			return false;
		}
	}
	else {
		$this->_messages[] = "Il file non è permesso poichè senza
estensione.";
		return false;
	}
}
```

Il metodo checkExtension()

5.4.8 Il metodo checkName

Uno dei controlli più delicati quando si deve gestire il caricamento di un file è il controllo sul nome di esso. Potrebbe sembrare a prima vista un operazione banale, ma non lo è per niente. Basterebbe un solo carattere particolare nel nome del file per causare degli effetti disastrosi. La scelta più semplice sarebbe stata quella di assegnare un nome casuale ma sicuro al file, ma questi comprometterebbe l'usabilità della applicazione, ad esempio, se si volesse offrire all'utente la possibilità di visualizzare i file caricati, ovviamente egli non riconoscerebbe nessun file dal nome. La scelta

adottata per questa tesi è stata quella di mantenere il nome che l'utente aveva assegnato al file, ma rendendolo sicuro da utilizzare. Inoltre, è stato scelto di mantenere uno storico dei file caricati, quindi eventuali file che hanno lo stesso nome non vengono sovrascritti, ma al più recente viene aggiunto un numero alla fine e viene incrementato fino a quando in nome non risulta unico.

```
protected function _checkName($name, $overwrite) {
	$nospaces = str_replace(' ', '_', $name);
	$nospaces = str_replace('%', '_', rawurlencode($nospaces));
	$nospaces = preg_replace('/\.{2,}/', '.', $nospaces);
	$nospaces = basename($nospaces);
	if ($nospaces != $name)
		$this->_renamed = true;
	if (!$overwrite) {
		$existing = scandir($this->_destination);
		if (in_array($nospaces, $existing)) {
			$dot = strrpos($nospaces, '.');
			if ($dot) {
				$base = substr($nospaces, 0, $dot);
				$extension = substr($nospaces, $dot);
			}
			else {
				$base = $nospaces;
				$extension = '';
			}
			$i = 1;
			do {
				$nospaces = $base . '_' . $i++ . $extension;
			} while (in_array($nospaces, $existing));
			$this->_renamed = true;
		}
	}
	return $nospaces;
}
```

Il metodo checkName()

Per evitare problemi con il filesystem, vengono eliminati gli spazzi dal nome, eventuali caratteri non ASCII vengono eliminati e, se vi sono presenti sequenze di due o più punti vengono sostituite con un singolo punto per evitare il traversamento del filesystem.

Per permettere una maggiore flessibilità della classe, si può scegliere di sovrascrivere un file caricato passando al metodo il valore true oltre al nome del file da controllare.

5.4.9 Il metodo checkNameLength

Questo metodo molto semplice serve per controllare la lunghezza del nome del file in modo da evitare possibili buffer-overflow se un attaccante fornisce un nome troppo lungo per un file. La lunghezza massima è memorizzata su un attributo della classe, quindi è facilmente modificabile.

5.4.10 Il metodo move

Il metodo move() è responsabile per il salvataggio vero e proprio del file. Il metodo chiama i metodi descritti precedentemente per verificare che tutto sia in regola. Se il file passa tutti i test, allora viene salvato nella cartella dell'utente e viene impostato l'attributo uploadStatus a true.

```
public function move($overwrite = false) {
	$field = current($this->_uploaded);
	$ok = $this->_checkError($field['name'], $field['error']);
	if ($ok) {
		$sizeOK = $this->_checkSize($field['name'], $field['size']);
		$typeOK = $this->_checkType($field['name'], $field['type']);
		$extensionOK = $this->_checkExtension($field['name']);
		$nameLengthOK = $this->_checkFileNameLength($field['name']);
		if ($sizeOK && $typeOK && $extensionOK && $nameLengthOK) {
			$name = $this->_checkName($field['name'], $overwrite);
			$this->_filePath =  $this->_destination . $name;
			$success = move_uploaded_file($field['tmp_name'], $this-
>_filePath);
			if ($success) {
				$message = 'Il file è stato caricato con successo';
				if ($this->_renamed) {
					$message .= " ed è stato rinominato $name";
					$this->_fileName = $name;
				}
				if (!$this->_renamed) {
					$this->_fileName = $field['name'];
				}
				$this->_messages[] = $message;
				$this->_uploadStatus = true;
			}
			else {
				$this->_messages[] = 'Impossibile caricare il file.';
			}
		}
	}
}
```

Il metodo move()

5.4.11 Gli altri metodi della classe

Gli altri metodi della classe sono dei getter che servono per leggere i valori delle variabili protette della classe e servono per comunicare i dati alle altre funzioni o classi che ne hanno bisogno.

5.5 La consegna dei file

Per la consegna dei file, è stato implementato lo script consegna.php. Il compito principale dello script è di assicurarsi che sia un utente legittimo a consegnare il file. Per garantire l'autenticità dell'utente, oltre a verificare che sia impostata a true la corretta variabile di sessione, viene anche eseguito un controllo sul profilo creato durante la fase di login dell'utente. Viene controllato che lo User-Agent del browser, la lingua preferita e l'indirizzo IP siano quelli che l'utente aveva quando si è loggato. Viene inoltre controllato che non siano passati più di 10 minuti dal momento del login, altrimenti la sessione eventualmente attiva viene distrutta e all'utente viene chiesto di rifare il login. Tutti questi controlli vengono effettuati per minimizzare il pericolo di un attacco Session Hijacking.

Per garantire allo script che svolgerà la correzione automatica del esercizio che il file sia caricato dalla form ufficiale della applicazione e non da una form creata ad hoc, magari per eseguire un attacco CSRF, viene generato un token casuale e salvato come variabile della sessione. Il token viene scritto in un campo hidden della form che invia il file, e lo script che riceve il file caricato deve controllare che il token inviato dalla form combaci con il token della sessione per garantire l'autenticità della form che ha inviato il file, e quindi che l'utente sia legittimo.

La funzione che controlla il profilo dell'utente

```php
function checkProfile($profile) {
	$result = true;
	$ip = $_SERVER['REMOTE_ADDR'];
	if ($profile['ipAddr'] != md5($ip)) {
		$result = false;
		return $result;
	}
	if ($profile['userAgt'] != md5($_SERVER['HTTP_USER_AGENT'])) {
		$result = false;
		return $result;
	}
	$currTime = time();
	if ($currTime - $profile['time'] > 600) {
		$result = false;
		return $result;
	}
	if ($profile['lang'] != md5($_SERVER['HTTP_ACCEPT_LANGUAGE'])) {
		$result = false;
		return $result;
	}
	return $result;
}
```

La funzione checkProfile()

La parte iniziale dello script che effettua i controlli di sicurezza e genera il token

```php
<?php
session_start();
session_regenerate_id();
$token = md5(uniqid(rand(), TRUE));
$_SESSION['token'] = $token;
if (!isset($_SESSION['auth'])) {
        header('Location: ../pages/login.php');
        exit;
}
else {
    $db = new Cl_Connector();
    $profileDir = $db->getFileDestination();
    $profileDir .= $_SESSION['userId'] . '\\';
    $profileFile = $profileDir . 'profile.txt';
    if (file_exists($profileFile)) {
        $profile = unserialize(file_get_contents($profileFile));
        if (!checkProfile($profile)) {
            header('Location: ../includes/sto.php');
        }
    }
    else {
        header('Location: ../includes/logout.php');
    }
}
?>
```

Una form usata per la sottomissione di un esercizio.

La form per l'upload del file viene generata dinamicamente e ad essa viene aggiunto un campo hidden che contiene il token associato alla sessione.

```php
<form enctype="multipart/form-data" action="correggi.php" method="post" >
    <input type="hidden" name="token" value="<?php echo $token; ?>" />
    <p>
        Scegliere il file da caricare:
        <input type="file" name="esercizio" id="es1" />
        <input type="hidden" name="esercizio" value="1" />
    </p>
    <p>
        <input type="submit" value="Upload" name="upload" />
    </p>
</form>
```

5.6 La correzione automatica

L'implementazione della correzione automatica esula dagli obbiettivi di questa tesi, ma comunque sono state gettate le basi per l'utilizzo di tale funzionalità. Lo script correggi.php viene chiamato da una form che l'utente usa per sottomettere un esercizio. Vengono effettuati nuovamente tutti i controlli sull'autenticità dell'utente per evitare eventuali attacchi, ovvero viene verificato che l'utente sia loggato e che sia effettivamente l'utente che ha effettuato il login, e non ad esempio un attaccante che è

riuscito a recuperare un identificatore di sessione valido. I controlli sono gli stessi che vengono eseguiti nella pagina di consegna del file, quindi ommettiamo di allegare il codice utilizzato, in quanto si può vedere nel paragrafo precedente.

In questo script viene controllato che la sottomissione del file sia eseguita dalla form legittima della applicazione e non da una form creata per eseguire un attacco.

```
if (!(isset($_SESSION['token']) && $_POST['token'] == $_SESSION['token'])) {
    header('Location: ../includes/sto.php');
}
```

Controllo sulla legittimità della form

Se il token non viene inviato dalla form o se è diverso dal token associato alla sessione attuale, abbiamo il ragionevole sospetto che la form non sia legittima, quindi l'utente viene rimandato nella pagina di sessione scaduta, dove viene distrutta la sessione corrente, se esistente, e viene avvisato l'utente che la sessione è scaduta e quindi si invita ad effettuare il login. Se tutti i controlli vengono passati, il file caricato viene salvato nella cartella dell'utente.

Allo script che implementa la correzione automatica si darebbe in input il file caricato dall'utente e si aspetta che ci restituisca il voto assegnato dopo la correzione e un commento sull'esito della correzione.

A questo punto si avrebbero tutte le informazioni necessarie per inserire i dati nel database, quindi utilizzando il metodo insertUser() della classe Cl_dbOperations, viene aggiunta la tupla nella tabella File.

```
$db = new Cl_dbOperations;
$userId = $_SESSION['userId'];
try {
    $upload = new Cl_Upload($profileDir);
    $upload->move();
    $idEsercizio = $upload->strip_input($_POST['esercizio']);
    if ($upload->getUploadStatus()) {
        /*
            Chiamata allo script che esegue la correzione automatica e recupero
            dell'esito della correzione.
        */
        $db->insertFile($_SESSION['userId'], $idEsercizio, $upload->getFileName(),
$upload->getExtension(), 1, 'Commento');
    }
    $result = $upload->getMessages();
}
catch (Exception $e) {
    echo 'Errore nella connessione al database';
}
```

Esempio di un inserimento dei dati dopo un'ipotetica correzione.

6. Conclusioni

Nell'ambito di questa tesi, è stato realizzato il back-end necessario per gestire la correzione automatica degli esercizi svolti dagli studenti del corso di programmazione, in particolare:

È stato realizzato il database MySQL necessario per accomodare i dati degli studenti, lo storico degli esercizi consegnati e l'esito della correzione per ogni esercizio consegnato. Per garantire la consistenza dei dati è stato usato InnoDB come storage engine in quanto supporta le transazioni.

Sono state implementate le classi necessarie per la connessione, la lettura e la scrittura dei dati sul database, cercando di rendere il codice il più generale possibile, in modo da rendere minime le modifiche da apportare ad esso in caso di cambiamenti.

Sono state implementate le funzionalità di registrazione e login degli studenti, gestendo correttamente anche le sessioni multiple.

È stato realizzato il meccanismo di caricamento dei file da parte degli studenti, concepito per permettere solo i file del tipo e della dimensione giusta, facendo in modo che sia comunque facile per l'amministratore modificare questi parametri, garantendo in questo modo una grande flessibilità e facilità di manutenzione.

Data la natura del progetto, una particolare attenzione è stata data alla sicurezza. È stato adottato il principio "Defense in Depth" che suggerisce di costruire una difesa a più livelli, in modo che al fallimento di una misura di sicurezza, ce ne sia un'altra che offre protezione.

La sicurezza assoluta in informatica è impossibile da realizzare, ma si è cercato di minimizzare la vulnerabilità agli attacchi più pericolosi che sono SQL Injection, Cross Site Scripting, Cross Site Request Forgeries e Session Hijacking.

6.1 Sviluppi futuri

La realizzazione di questa tesi fa parte di un progetto molto ampio. Sono state costruite delle fondamenta solide sulle quali potere costruire le altre funzionalità e possiamo distinguere due direzioni principali da seguire, che sono funzionalità e sicurezza.

Per quanto riguarda le funzionalità, rimane da unire il lavoro svolto per questa tesi con il lavoro svolto da altri studenti che hanno implementato la funzionalità della correzione automatica. Inoltre andrebbero adottate adeguate misure per verificare l'autenticità delle soluzioni proposte dagli studenti, in modo da non permettere la sottomissione della stessa soluzione da parte di più studenti.

Un'altra funzionalità che rimane da aggiungere è la generazione di report settimanali utili ai docenti per tenere sotto controllo lo stato delle consegne e la generazione automatica di liste pre-esame dove vengono elencati gli studenti abilitati a sostenere l'esame e relativi punti bonus, se presenti.

Per quanto riguarda la sicurezza, rimane da adottare l'utilizzo di un certificato SSL, che comporterebbe un incremento notevole della sicurezza, a fronte di una spesa piuttosto contenuta.

Bibliografia

Alshanetsky, I. (2005). *Guide to PHP Security.* Marco Tabini & Associates, Inc.

Ballad, T., & Ballad, W. (2009). *Securing PHP Web Applications.* Pearson Education.

Bell, C. (2012). *Expert MySQL: Second Edition.* Apress.

Converse, T., Park, J., & Morgan, C. (2004). *PHP5 and MySQL Bible.* Wiley.

Elmasri, R., & Navathe, S. B. (2011). *Database Systems.* Addison-Wesley.

Gilmore, W. J. (2008). *Beginning PHP and MySQL.* Apress.

Hopkins, C. (2013). *Jump Start PHP.* Sitepoint.

Hornby, T. (s.d.). *Defuse Security*. https://defuse.ca/

Lavin, P. (2006). *Object-Oriented PHP.* No Starch Press.

Lerdorf, R., Tatroe, K., & Peter, M. (2013). *Programming PHP*. O'Reilly.

MacIntyre, P. B. (2010). *PHP: The Good Parts.* O'Reilly.

Mitchell, L., Shafik, D., & Turland, M. (2011). *PHP Master.* SitePoint.

Nixon, R. (2012). *Learning PHP, MySQL, JavaScript, and CSS, Second Edition.* O'Reilly.

OWASP: www.owasp.org/

PHP Security Consortium: phpsec.org/

Pratt, P. J., & Last, M. Z. (2009). *A Guide To SQL.* Cengage Learning.

Schlossnagle, G. (2004). *Advanced PHP Programming.* Sams Publishing.

Shiflett, C. (2005). *Essential PHP Security.* Oreilly.

Snyder, C., Myer, T., & Southwell, M. (2010). *Pro PHP Security.* Apress.

Steinmetz, W., & Ward, B. (2008). *Wicked Cool PHP.* No starchPress.

Tahaghoghi, S. M., & Williams, H. E. (2007). *Learning MySQL.* O'Reilly.

Vaswani, V. (2010). *MySQL Database Usage and Administration.* Mc-Graw-Hill.

Wikipedia. en.wikipedia.org

Yank, K. (2012). *PHP and MYSQL, Novice to Ninja.* SitePoint.

Zandstra, M. (2010). *PHP Objects, Patterns, and Practice.* Apress.

Zervaas, Q. (2008). *Practical Web 2.0 Applications with PHP.* Apress.

Ringraziamenti

Vorrei dedicare questa tesi a Marti, per avermi sopportato e supportato in questi anni. Grazie Amore, senza di te non ce l'avrei mai fatta.

Un grazie ai miei genitori, per il sostegno che mi hanno sempre dato.

Un grazie alla Professoressa Carbonaro e al dottor Ravaioli per il loro aiuto e i preziosi consigli.

Last but not least, un grazie ai miei colleghi e amici, il dottor Riccardo Guglielmo, il dottor Andrea Battaglia e il dottor Emanuele Reggi per le ore di studio insieme e che grazie a loro non è stato mai noioso.