Università degli Studi di Milano

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Triennale in Informatica

Realizzazione di un'applicazione Web di supporto alla gestione di documentazione e prodotti software relativi a progetti di un laboratorio di ricerca

Relatore: **Prof.ssa Silvana CASTANO**
Correlatore: **Dott. Alfio FERRARA**

Tesi di Laurea di:
Matteo Alberto SORRENTI
Matricola 607372

Anno Accademico 2002–2003

ai miei genitori

Indice

Ringraziamenti

Si ringraziano la professoressa Silvana Castano, il dottor Alfio Ferrara ed in particolar modo il dottor Gianpaolo Racca per la disponibilità e la pazienza durante tutte le varie fasi di sviluppo.

Elenco delle figure

Introduzione

Obiettivo del lavoro di tesi è la realizzazione di un'applicazione Web che gestisca i vari processi di sviluppo di un qualsiasi progetto software. A questa applicazione (denominata "Project Management") si può accedere attraverso l'intranet del laboratorio di Sistemi Informativi e solo gli iscritti ad esso possono utilizzarla.
La realizzazione di questa applicazione è motivata dall'esigenza di avere un sito comune di riferimento che gestisca i vari progetti e dove poter controllare e scaricare ogni aggiornamento.
Per poter facilmente distinguere le diverse fasi di sviluppo di un progetto ad ognuna di esse corrisponde un rilascio con rispettiva documentazione.
Un rilascio di un progetto contiene un insieme di moduli, i quali possono essere importati o aggiornati da una versione precedente, oppure creati ex-novo.
Un modulo può essere suddiviso in tre diversi file di archivio: uno che contiene i file sorgenti, uno che contiene i file compilati e uno che contiene la documentazione del modulo.
Bisogna distinguere tre diversi tipi di utenti che possono accedere al sito: l'amministratore del sito, l'utente che è amministratore di un progetto e l'utente che ha i privilegi solo su uno o più moduli di un progetto.
L'amministratore del sito ha la possibilità di effettuare qualsiasi operazione su qualsiasi progetto ed è l'unico ad avere il potere di validare o cancellare un rilascio.
L'utente che è amministratore di un progetto ha la possibilità di aggiornare, rinominare o cancellare completamente il progetto del quale possiede i privilegi.
Infine, l'utente che ha i privilegi di determinati moduli può aggiornare solo tali moduli.
Il download dei progetti e dei rispettivi moduli è permesso anche a chi non ha nessun tipo di privilegio ma è comunque iscritto al laboratorio di Sistemi Informativi.

Per ogni progetto si può scaricare o un solo modulo oppure un file d'archivio in formato tar.gz contenente tutti i moduli. Esistono tre file per ogni progetto: in uno sono contenuti i moduli con i file sorgenti, in un altro i moduli con i file compilati e in un altro i moduli con i file di documentazione. In più è presente un ulteriore file di documentazione che si riferisce a tutto il rilascio.
Per memorizzare i progetti (con la relativa documentazione), i moduli, gli aggiornamenti, i privilegi degli utenti e altre caratteristiche del sito viene utilizzato un database.
La tesi è stata organizzata in quattro capitoli. Nel primo vengono descritti gli strumenti utilizzati per realizzare il progetto, nel secondo la progettazione del database, nel terzo una documentazione dei file presenti nell'applicazione e nel quarto un manuale utente con relativi screenshot.

Capitolo 1

Requisiti architetturali del progetto

1.1 Dalle pagine statiche alle pagine dinamiche

Durante i primi anni il Web era basato su una comunicazione di tipo testuale. Un fisico del CERN, Tim Berners-Lee, propose il World Wide Web come un servizio ipertestuale attraverso il quale gli scenziati sarebbero stati in grado di scambiarsi in maniera più rapida i dati delle loro ricerche. I primi browser, offrivano solo testo sulle proprie pagine Web; si poteva accedere solo a pagine statiche, ovvero pagine memorizzate sul server Web, che venivano restituite al client senza alcun tipo di elaborazione. Il contenuto non poteva cambiare né in base al tipo di richiesta, né in base al tipo di utente. Nel 1993 Marc Anderseen ed i suoi colleghi dell'Università dell'Illinois crearono Mosaic, il primo browser grafico che ha permesso la visualizzazione insieme al teso anche di immagini all'interno delle pagine. In primo luogo, il Web cominciò a coinvolgere una platea più vasta di utenti, tra i quali ricercatori e professori universitari, che già usavano servizi come la posta elettronica, ai quali ben presto si aggiunse una larga parte delle persone tecnologicamente alfabetizzate del mondo industrializzato. Con l'introduzione di immagini e suoni le pagine cominciarono ad essere più complesse e ricche di contenuti, però l'interazione era ancora inesistente. Per risolvere questa limitazione è stato creato il Common Gateway Interface (CGI), un meccanismo che permette l'esecuzione di programmi sul server in risposta alle richieste di un utente. Le applicazioni CGI hanno permesso l'evoluzione del Web. Parallelamente, importante ruolo è stato giocato dalla programmazione lato-client che rende possibile all'utente di diventare parte attiva nell'interazione. La programmazione lato-client risparmia al

server una parte del lavoro perché i programmi vengono prima scaricati e poi eseguiti direttamente sull'elaboratore del client. Uno dei principali problemi di questa tecnica di programmazione è dovuto al fatto che il codice deve essere scritto per funzionare su browser di tipo diverso e spesso si riscontrano vari problemi di incompatibilità.
La programmazione lato-server, al contrario, permette di scrivere programmi universali dato che il client alla fine riceve solo una pagina in formato HTML da visualizzare.

1.2 Applicazioni lato-server

Le applicazioni lato-server risiedono sul server e vengono eseguite nel momento in cui il browser (che ricopre il ruolo di client) richiede al server Web una determinata pagina. Questa pagina, identificata da un URI, è in realtà un programma che necessita di una particolare elaborazione prima di poter essere inviata al client. Questo tipo di architettura è detta client/server, infatti sia il client sia il server sono responsabili del corretto funzionamento di tutta l'operazione. Il compito del server è di immagazzinare, interpretare e distribuire i vari dati, mentre quello del client è di accedere al server per ottenere questi dati.
In un'architettura client/server prima di tutto il browser invia al server una richiesta HTTP (Hyper Text Transfert Protocol), nella quale è indicato il metodo (HEAD, GET, POST), l'intestazione e il corpo (solo se usato il metodo POST). Una richiesta HTTP può essere fatta o compilando una form, che creerà la URL (Uniform Resource Locator) da richiedere, o indicando direttamente la URL, nella quale è specificato sia il protocollo di connessione sia la pagina desiderata.
All'arrivo della richiesta HTTP il server legge la risorsa e, se questa contiene script lato-server, essi vengono eseguiti o come DSO (Dynamic Shared Object) o come script CGI (Common Gateway Interface), il codice restituito rimpiazza il codice sorgente tenendo immutato eventuale codice HTML presente. A questo punto il server invia al client la risposta HTTP, formata da un'intestazione e da un corpo contente la pagina ottenuta, dopo di che viene chiusa la connessione e al browser non rimane che visualizzare la pagina appena ricevuta ed eseguire eventuali script lato-client (JavaScript, VBScript ...).

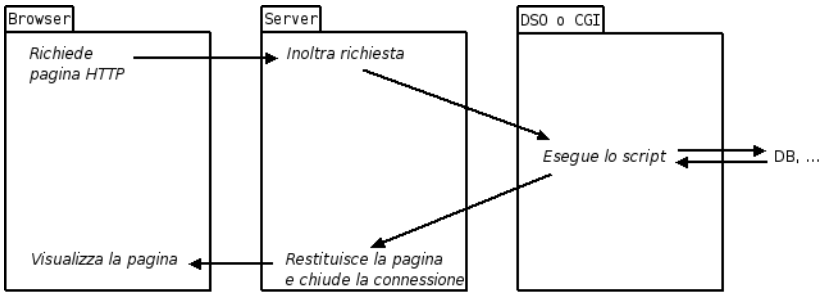

Figura 1.1: *Interazione client-server con script lato-server*

1.3 PHP

Il PHP, acronimo ricorsivo di PHP: Hypertext Preprocessor e ideato originariamente da Rasmus Lerdorf nel 1994, è un linguaggio di programmazione progettato principalmente per la costruzione di siti Web dinamici.
Appartiene ad una classe di linguaggi detti middleware. Questi linguaggi lavorano a stretto contatto con il server Web per interpretare le richieste fatte dal client, elaborarle, interagendo anche con altri programmi sul server, e producono l'output che il server Web fornirà al browser.
La sintassi del PHP è simile a quella di C, Java e Perl, questo permette ai programmatori che già conoscono questi linguaggi un più facile approccio.

1.3.1 Funzionalità

Il PHP è un linguaggio multipiattaforma, può essere utilizzato su tutti i principali sistemi operativi (Linux, Microsoft Windows, MacOS X, Unix e derivati ...) e un programma eseguito su un determinato sistema operativo generalmente può essere eseguito anche su uno diverso.
Il PHP supporta la maggior parte dei server Web esistenti (Apache, IIS, Caudium ...) e ha un modulo per molti di essi, per gli altri può funzionare come programma CGI, purché i server Web supportino questo standard.
In definitiva usando il PHP si è sostanzialmente liberi di scegliere sia il sistema operativo sia il server Web che si preferisce e che corrisponde maggiormente alle caratteristiche

necessarie per la creazione del sito.
Oltre a questo si ha anche piena libertà nella scelta del tipo di programmazione da usare per scrivere il codice, dati che, si può scegliere una programmazione procedurale, una orientata agli oggetti o addirittura entrambe, anche se al contrario di Java, C++ e altri linguaggi l'approccio object-oriented in PHP ha un supporto limitato anche se in evoluzione. Una classe PHP non può contenere costanti ma solo variabili. Per le classi derivate il costruttore delle superclassi non viene chiamato automaticamente. Non c'è modo di indicare i metodi come "privati" e non ci sono decostruttori. Non è previsto polimorfismo né overloading di funzioni: se in una sottoclasse è presente un metodo con lo stesso nome di uno della superclasse ma con un numero differente di parametri, l'oggetto che istanzia la sottoclasse non può chiamare il metodo della superclasse. Al contrario del C++ non è consentita l'ereditarietà multipla, comunque questa mancanza è una caratteristica presente anche in altri linguaggi orientati agli oggetti come Java.
Il PHP è un linguaggio HTML-embedded, il codice scritto contiene al suo interno sia istruzioni PHP sia tag HTML e questo è un grande vantaggio perché non bisogna utilizzare funzioni o classi specifiche per produrre HTML. Il codice PHP è delimitato da speciali start ed end tag che separano il codice che verrà interpretato dal contenuto HTML che passerà immutato in output.
Inoltre, non si è limitati a produrre solo HTML dal momento che il PHP permette anche di creare immagini, file PDF e addirittura filmati Flash utilizzando le apposite librerie. E' possibile generare o fare il parsing di testi XHTML e XML, salvarli nel file system o mandarli in output.
Dalla versione 4 il PHP è stato completamente riscritto per poter essere ancor più veloce grazie al "motore" Zend.
Il PHP è un linguaggio open source e questo gli permette di essere sempre costantemente aggiornato dal momento che c'è una gran quantità di sviluppatori che cercano di migliorarlo costantemente. Essendo un software libero è disponibile gratuitamente e sul Web si possono trovare molte librerie da poter utilizzare liberamente. Inoltre, la comunità degli sviluppatori e utilizzatori è numerosa, per cui non è difficile trovare in Internet qualcuno che sia disposto a fornire suggerimenti nel caso si riscontrassero

difficoltà nell'installazione o errori nel codice.
La funzione principale del PHP è quella di creare pagine Web dinamiche. Una qualunque pagina Web viene composta a partire dalle informazioni provenienti dal browser e su ogni altro dato disponibile sul server, cosa che il PHP permette di fare senza problemi. Nel momento in cui si costruisce un'applicazione si deve spesso provvedere alla persistenza dei dati e il modo più semplice per farlo nelle applicazioni Web è ricorrere all'uso di un database (solitamente residente sulla stessa macchina che ospita il server Web).
Utilizzando un database, infatti, si possono memorizzare e recuperare facilmente delle informazioni, mentre utilizzando un file di testo le stesse operazioni possono risultare molto più lunghe e difficoltose se non addirittura impossibili. Uno dei punti di forza del PHP è proprio la capacità di connettersi e interagire facilmente con i più diffusi database (PostgreSQL, Oracle, MySQL, IBM DB2, Sybase, Informix ...).

1.4 PostgreSQL

PostgesSQL è uno dei database open source di maggior popolarità e probabilmente il più avanzato; le sue funzionalità sono paragonabili a quelle di molti altri software commerciali. PostgreSQL è disponibile per piattaforme Linux, Unix, BSD e derivati. Può funzionare anche su una piattaforma di tipo Windows, ma per fare ciò deve girare in ambiente Cygwin in quanto non può interagire direttamente con Windows. Per ovviare a questo problema c'è in previsione di rilasciare a breve una versione nativa per Windows.

1.4.1 Storia

PostgreSQL nacque con il progetto Ingres a Berkeley presso l'Università della California e fu poi sviluppato per scopi commerciali dalla Relational Technologies/Ingres Corporation. Nel 1986 il professor Michael Stonebraker di Berkeley coordinò un gruppo per continuare a sviluppare il codice di Ingres. Quello che si ottenne fu un database relazionale ad oggetti al quale fu dato il nome di Postgres. Nel 1994 il nome venne cambiato in Postgres95 e nel 1996, grazie alle crescenti funzionalità del software ap-

portate dalla comunità open source gli venne il nome di PostgreSQL. Questo nome fu scelto per riflettere la relazione fra il POSTGRES originale e le versioni più recenti con possibilità di interpretazione attraverso un linguaggio SQL standard.

1.4.2 Versioni

Esistono due versioni di PostgreSQL, una commerciale e una open source. La versione commerciale corrisponde alla versione RedHat di PostgreSQL e viene chiamata RedHat Database, è basta sulla versione open source, comprende un installatore grafico, un'assistenza all'installazione e alla configurazione. La versione open source, invece, comprende solo il sistema di gestione dei database. La versione commerciale conviene solo nel caso in cui non si abbia dimestichezza con la configurazione, altrimenti la versione rilasciata dalla comunità open source è preferibile poiché il costo per acquisirla è nullo.

1.4.3 Funzionalità

PostgreSQL è un DBMS relazionale ad oggetti, ciò significa che utilizza un modello dei dati relazionale ad oggetti che lo rende in grado di gestire regole e procedure molto complesse. Tra le sue caratteristiche c'è la gestione di query dichiarative SQL, il controllo della concorrenza, le transazioni, l'ottimizzazione delle query, l'ereditarietà e gli array. L'implementazione degli array non è aderente allo standard dell'SQL99, quindi in futuro potrebbe cambiare.

PostgreSQL permette l'uso di operatori, funzioni, metodi d'accesso e tipi di dati definiti dall'utente e supporta l'integrità referenziale per garantire la validità dei dati inseriti.

PostgreSQL ha delle API molto flessibili che hanno permesso lo sviluppo di numerose interfacce, tra cui quelle per Object Pascal, PHP, Python, Perl, C/C++, ODBC, Java/JDBC, TCL.

Supporta inoltre linguaggi procedurali interni, quello nativo è PL/pgSQL che può essere paragonato al linguaggio procedurale PL/SQL di Oracle. In aggiunta è possibile usare Perl, C, Python, TCL e altri come linguaggi procedurali incorporati.

Viene usata una tecnologia MVCC (Multi-Version Concurrency Control) per evitare

lock non necessari. In altri database l'utente in lettura deve rimanere in attesa per accedere alle informazioni se contemporaneamente è presente un utente che sta aggiornando i record. Invece, grazie all'impiego di MVCC, viene tenuta traccia di tutte le transazioni effettuate dagli utenti del database, così si è in grado di gestire i record senza far attendere la loro disponibilità.
PostgreSQL impiega un'architettura client/server che permette di riservare un processo per ogni utente, un comportamento in modo simile a quello di un server Web.
E' presente una funzionalità nota come WAL (Write Ahead Ligging) che incrementa l'affidabilità del database, dato che permette di memorizzare le modifiche prima che vengano inserite. In questo modo, nel caso si verificasse un errore non gestibile è possibile recuperare le informazioni che non erano state ancora inserite.
Nella versione attuale PostgreSQL supporta le specifiche fondamentali dell'SQL99 e le funzionalità avanzate di SQL92. Inoltre permette la creazione di query innestate.

1.4.4 SQL

L'SQL (Structured Query Language) è un linguaggio che permette di interagire con il database e deriva dal linguaggio di interrogazione di System R.

Storia

Il modello relazionale fu definito da un ricercatore dell'IBM, il Dottor E. F. Codd, nel saggio del 1970: "A Relational Model Of Data For Shared Data Banks". Dopo questa pubblicazione cominciarono le ricerche per studiarne la fattibilità e le possibili applicazioni. Nel 1974, partendo dal lavoro di Donald Chamberlin, IBM iniziò il progetto System/R e sviluppò SEQUEL (Structured English Query Language). System/R fu implementato tra il 1974 e il 1975, successivamente riscritto tra il 1976 e il 1977. Al lavoro ultimato fu assegnato il nome di SEQUEL/2 poi in seguito ribattezzato SQL. Nel 1978 cominciarono le prime verifiche sul campo per vederne l'utilità e le risposte della clientela furono entusiasmanti. Per questo motivo IBM decise di sviluppare prodotti commerciali che implementavano SQL basandosi sul prototipo System/R; SQL/DS venne introdotto nel 1981 e DB2 nel 1983. Anche altre software house sfruttarono il successo del modello relazionale e commercializzarono i loro prodotti basati sull'SQL,

tra essi c'erano Oracle (che riuscì a superare IBM diventato leader nel settore), Sybase e Ingres.

Funzionalità

L'SQL si basa sull'algebra relazionale che è stata introdotta da E. F. Codd nel 1972. E' un linguaggio che permette la definizione di uno schema di una base di dati e la modifica o l'interrogazione delle istanze in essa contenute, inoltre:

- possiede funzionalità maggiori rispetto ad altri linguaggi basati sull'algebra relazionale;
- agli utenti è permesso di cancellare o modificare dati memorizzati all'interno della base di dati;
- è consentito l'uso sia di operatori aritmetici sia di operatori di confronto;
- gli utenti possono visualizzare le relazioni generate dalle query;
- è possibile rinominare una relazione ottenuta come risultato di una query;
- si possono usare funzioni aggregate grazie alle quali è possibile eseguire operazioni anche complesse sui dati. Ad esempio raggruppare delle righe correlate per poi contarle, calcolarne la media, la somma, trovare il valore massimo oppure il minimo.

Standardizzazione

La prima standardizzazione dell'SQL è avvenuta nel 1986 per opera dell'ANSI (American National Strandards Institute). Successivamente, nel 1987, anche l'ISO (International Standards Organization) se n'è occupato e infine nel 1989, dall'ANSI, è stato rilasciato uno standard revisionato e corretto col nome di SQL89 (detto anche SQL1). Questo standard presentava diverse lacune e imprecisioni a causa dei vari interessi contrastanti dei produttori e quindi si cercò di rafforzarlo; l'ANSI lo fece nel 1992 pubblicando lo standard SQL92 (chiamato anche SQL2).
L'SQL92 corresse molte imprecisioni dell'SQL89 oltre ad aumentarne le funzionalità.

Data l'enorme quantità di novità introdotte da SQL92 si decise di definire tre livelli di conformità allo standard: Entry, Intermediate e Full.
Gli organismi ANSI/ISO nel 1999 hanno rilasciato un nuovo standard l'SQL99 (detto anche SQL3), rivolto in particolare ai database relazionali ad oggetti, alla gestione dell'integrità e alle interfacce a livello di chiamata. L'SQL99 ha sostituito i livelli di conformità dell'SQL92 con due nuovi gradi di conformità: Core SQL99 e Enhanced SQL99.
Attualmente PostgreSQL è conforme alla maggior parte del livello Entry SQL92 e possiede molte caratteristiche dei livelli Intermediate e Full. Molte funzionalità dell'SQL99 come array, funzioni ed ereditarietà sono già presenti in PostgreSQL, sebbene in alcuni casi ci siano differenze di implementazione.

1.5 Apache

Il compito di un server Web in esecuzione è quello di attendere le richieste fatte da un client attraverso il protocollo TCP/IP e di soddisfarle inviando le risorse desiderate. In apparenza può sembrare un lavoro semplice, ma in realtà è piuttosto complicato e la stabilità di un server Web risulta essere un requisito fondamentale.
Apache è il server Web più diffuso e anch'esso (come PHP e PostgreSQL) è un prodotto open source e chiunque può scrivere del codice per estenderne le funzionalità.
Per rendere più snella la struttura e il funzionamento di Apache gli sviluppatori hanno deciso di suddividere in due parti il servizio; da una parte c'è il cuore vero e proprio dove sono contenute le funzionalità di base e dall'altra ci sono gli add-on, attraverso i quali sono implementati i moduli aggiuntivi.
Apache funziona sia sotto Windows che in ambiente Unix anche se è preferibile utilizzarlo su quest'ultima piattaforma perché in un sistema Windows può dare problemi nel caso ricevesse molte richieste. Questo problema deriva dal fatto che Apache è stato sviluppato su piattaforma Linux ed il porting è stato fatto solo in un secondo tempo. Inoltre il suo funzionamento su Windows non è stato testato così a fondo come sulla piattaforma nativa.
Apache è un server Web molto veloce e stabile, però può risultare piuttosto complesso

a chi gli si avvicini per la prima volta perché ci sono dei limitati strumenti grafici per poterlo configurare. Nel caso si vogliano cambiare le impostazioni, quindi, le modifiche vanno fatte da riga di comando o editando dei file di testo.
Il principale file di configurazione è "http.conf". Anche se si tratta di un lungo file di testo la consultazione risulta essere molto intuitiva e dopo un'attenta lettura risulta piuttosto semplice modificare le direttive in esso contenute. In questo file si possono distinguere due tipi di direttive: semplici e composte. Quelle semplici sono contenute su una sola riga, mentre quelle composte si possono trovare su più righe e sono facilmente riconoscibili perché contraddistinte da una semantica simile a quella dei tag html. E' possibile modificarle entrambe in base alle proprie necessità.

Capitolo 2

Progettazione della base di dati

2.1 Progettazione concettuale

Per progettare la base di dati si è partiti dall'entità project e dall'entità usr.
Mentre un progetto è identificato univocamente dal proprio titolo (*title*), un utente possiede un id che lo identifica, con un attributo admin che indica se è amministratore del sito oppure no.
Di un progetto, oltre al titolo, è opportuno conoscere la piattaforma (*platform*) sulla quale può essere eseguito. Dal momento che più utenti possono accedere allo stesso progetto si è deciso di includere un attributo lock per permetterne la modifica senza problemi. Se un utente sta bloccando un progetto, l'attributo id contiene l'identificativo di chi lo sta bloccando, altrimenti contiene quello dell'ultimo utente che l'ha bloccato. Per evitare problemi di starvation è presente un attributo time che memorizza quando un progetto è stato bloccato; se la modifica del progetto non è più inibita questo attributo contiene il momento in cui è stato rilasciato il lock. Infine è presente un attributo description che contiene una decrizione del progetto.
I progetti possono avere uno o più propretari, gli utenti possono essere proprietari di zero o più progetti e per specificare chi possiede o meno questi privilegi è presente la relazione privpro.
Un progetto possiede (*possess*) zero o più rilasci (*release*). Un rilascio viene identificato univocamente dal numero di versione (*version*) e dal progetto che lo possiede. La versione di un rilascio è un attributo composto di tre numeri (*n1*, *n2* e *n3*). In un rilascio bisogna conoscerne lo stato (*state*), che indica se si tratta di una versione alfa,

beta o stabile e anch'esso viene scelto arbitrariamente dall'utente. Infine un rilascio per poter essere scaricato deve essere validato dall'amministratore del sito, quindi è presente un attributo valid che indica se il rilascio è stato approvato.
Un rilascio può essere descritto (*describe*) da un file contenente la documentazione (*document*) e ovviamente questo file deve poter descrivere un solo rilascio. La tabella contenente i file di documentazione è identificata con il rilascio che descrive, in più è presente un attributo che indica l'estensione (*extension*) di tale file (solitamente un file di archivio: ".tar", ".tar.gz", ".zip" ...).
Un rilascio deve contenere (*contain*) uno o più moduli (*module*). Un modulo è un file di archivio che contiene i file che compongono il modulo. Per per una più chiara distinzione si è scelto di distinguere un modulo in almeno tre file: uno che contiene i file sorgenti (*sources*), uno che contiene i file di documentazione del modulo (*documented*) e uno che contiene i file compilati (*compiled*). Un modulo viene identificato dal rilascio che lo contiene, dal nome (*name*), dal tipo (*type*) e dalla versione (*version*). Del modulo, essendo un file, bisogna conoscere anche l'estensione (*extension*), in più è opportuno avere sia l'autore (*author*) che una descrizione (*description*).
Oltre ai progetti, anche i moduli possono avere uno o più proprietari e gli utenti possono essere proprietari di zero o più moduli. Per specificare chi possiede o meno questi privilegi è presente la relazione privmod.
Infine bisogna memorizzare ogni cambiamento che un utente fissa (*set*) nel database. Per fare ciò si utilizza una tabella (*changes*) che ha come chiave primaria l'identificatore dell'utente che ha effettuato il cambiamento e il momento (*time*) nel quale è avvenuto il cambiamento. Inoltre sono presenti l'attributo action, che serve ad indicare l'azione avvenuta (creazione progetto, cambio privilegi ...), e l'attributo specify, il quale indica l'oggetto che ha subito il cambiamento (titolo del progetto, identificatore dell'utente ...).

2.2 Progettazione logica

Nella tabella *usr* sono inseriti gli utenti che possono accedere al sito. Questa tabella non contiene tutti gli utenti, ma quando occorre questi vengono recuperati dal databa-

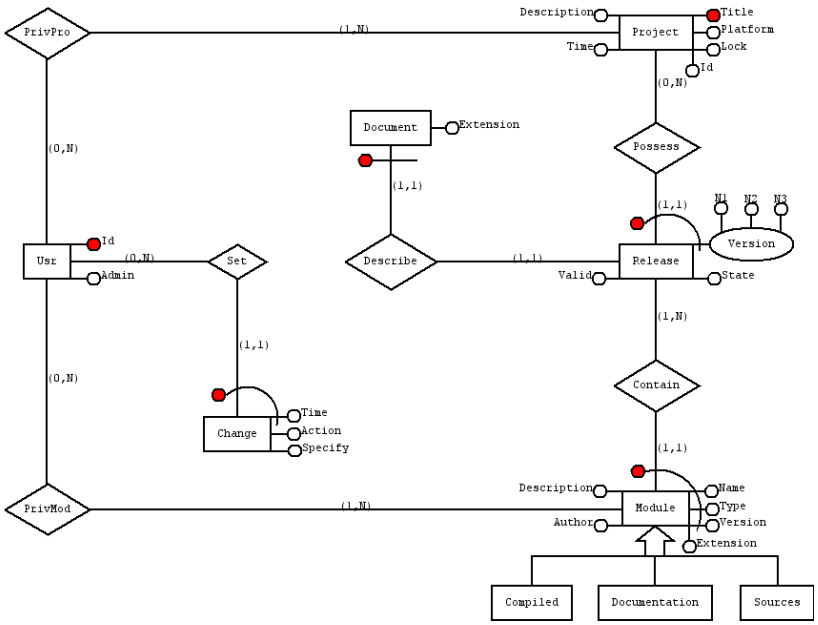

Figura 2.1: *Schema E-R*

se del Laboratorio di Sistemi Informativi (ISLab), dopodiché l'applicazione controlla se sono già presenti in *usr* e se non lo sono vengono inseriti. Questa tabella serve solo per controllare i privilegi che hanno gli utenti nei confronti dei progetti perché per la validità dei loro accessi l'applicazione utilizza un oggetto che consulta la tabella degli utenti del database dell'ISLab. L'attributo *admin* è solo un flag per controllare il privilegio di amministrazione del sito, quindi si tratta di un semplice bit che conterrà `1` se l'utente è amministratore del sito, altrimenti `0`.

Nei progetti, la piattaforma e la descrizione sono dei valori non obbligatori che possono assumere dei valori nulli. L'attributo *lock* serve per controllare il blocco di un progetto che viene incrementato per bloccare e decrementato per liberare, se il valore contenuto è `1` significa che il progetto è bloccato, se maggiore di `1` significa che più utenti stanno cercando di bloccare lo stesso progetto, invece se contiene `0` significa che il progetto è libero e può essere bloccato.

Nella tabella rilasci è presente un attributo composto di tre numeri: *n1*, *n2* e *n3*, questa scelta è stata fatta per poter più facilmente distinguere le varie versioni di uno stesso progetto. Il primo numero (*n1*) viene incrementato quando si ha un cambiamento

sostanziale rispetto al rilascio precedente, la scelta di modificarne il valore viene presa arbitrariamente dall'utente che effettua l'ultimo aggiornamento. Il secondo numero (*n2*) indica la variazione del numero dei nuovi moduli inseriti ed il terzo (*n3*) viene incrementato in relazione al numero dei moduli che hanno subito una variazione rispetto ai rilasci precedenti. Non essendo possibile rappresentarlo in maniera diretta, l'attributo multivalore version è stato partizionato nelle sue tre parti.

E' possibile che i documenti del rilascio, come i moduli, essendo memorizzati come file, non abbiano estensioni, quindi si è deciso di permettere il valore nullo all'attributo *extension* sia nella tabella document che nella tabella module, in quest'ultima tabella è presente anche una descrizione facoltativa, quindi anche l'attributo *description* può assumere un valore nullo. L'attributo *valid* serve per controllare se un rilascio è stato approvato oppure no: è sufficiente un solo bit che conterrà il valore 0 se il rilascio non è ancora valido, in caso contrario il valore sarà 1.

I moduli sono stati generalizzati in tre diverse entità a seconda che contengano file sorgenti, compilati o di documenti. Queste tre diverse entità sono state accorpate in module e per distinguerne il tipo è presente l'attributo type che conterrà la stringa "src" se si tratta di file sorgenti, "comp" di file compilati e "docs" di documenti.

Infine *privpro* e *privmod* essendo delle relazioni molti a molti diventano anch'esse delle tabelle nello schema relazionale.

Qui di seguito lo schema relazionale prodotto:

usr	(**id**, admin)
project	(**title**, platform, lock, id, time, description)
release	(**title, n1, n2, n3**, state, valid)
module	(**title, n1, n2, n3, name, type, version**, extension, author, description)
document	(**title, n1, n2, n3**, extension)
change	(**id, time**, action, specify)
privpro	(**title, id**)
privmod	(**title, n1, n2, n3, name, type, version, id**)

Legenda:
primary key
foreign key
default null

2.3 Implementazione

Ecco il codice sql utilizzato per la creazione delle tabelle:

```
CREATE TABLE usr (
id integer,
admin bit DEFAULT '0',
PRIMARY KEY (id)
);

CREATE TABLE project (
title varchar,
platform varchar DEFAULT NULL,
lock smallint NOT NULL,
id integer NOT NULL,
time integer NOT NULL,
description varchar DEFAULT NULL,
PRIMARY KEY (title)
);

CREATE TABLE release (
title varchar,
n1 integer DEFAULT '0', n2 integer DEFAULT '0', n3 integer DEFAULT '0',
state char DEFAULT 'a',
valid bit DEFAULT '0',
PRIMARY KEY (title, n1, n2 , n3),
FOREIGN KEY (title) REFERENCES project (title) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE module (
title varchar,
n1 integer DEFAULT '0', n2 integer DEFAULT '0', n3 integer DEFAULT '0',
name varchar,
type varchar,
version integer DEFAULT '0',
extension varchar DEFAULT NULL,
author varchar NOT NULL,
description varchar DEFAULT NULL,
PRIMARY KEY (title, n1, n2 , n3, name, version, type),
FOREIGN KEY (title, n1, n2 , n3) REFERENCES release (title, n1, n2, n3) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE document (
title varchar,
n1 integer DEFAULT '0', n2 integer DEFAULT '0', n3 integer DEFAULT '0',
extension varchar DEFAULT NULL,
PRIMARY KEY (title, n1, n2 , n3),
FOREIGN KEY (title, n1, n2 , n3) REFERENCES release (title, n1, n2, n3) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE change (
id integer,
time integer,
action varchar NOT NULL,
specify varchar NOT NULL,
PRIMARY KEY(id, time),
FOREIGN KEY (id) REFERENCES usr (id) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE privpro (
title varchar,
id integer,
PRIMARY KEY (title, id),
FOREIGN KEY (title) REFERENCES project (title) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (id) REFERENCES usr (id) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE privmod (
title varchar,
n1 integer DEFAULT '0', n2 integer DEFAULT '0', n3 integer DEFAULT '0',
name varchar,
type varchar,
version integer DEFAULT '0',
id integer,
PRIMARY KEY (title, n1, n2 , n3, name, version, type, id),
FOREIGN KEY (title, n1, n2 , n3, name, version, type) REFERENCES modules (title, n1, n2 , n3, name, version, type)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (id) REFERENCES usr (id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

Capitolo 3

Sviluppo dell'applicazione "Project Management"

3.1 Use case UML

In questo paragrafo verranno descritti i vari casi d'uso che si ottengono dalle interazioni dell'utente con l'applicazione.

3.1.1 Modifica i privilegi d'amministrazione

Per modificare i privilegi di amministrazione bisogna controllare i privilegi sia all'utente che effettua l'operazione, sia all'utente che la subisce. Dopo aver fatto ciò si può procedere nella modifica.

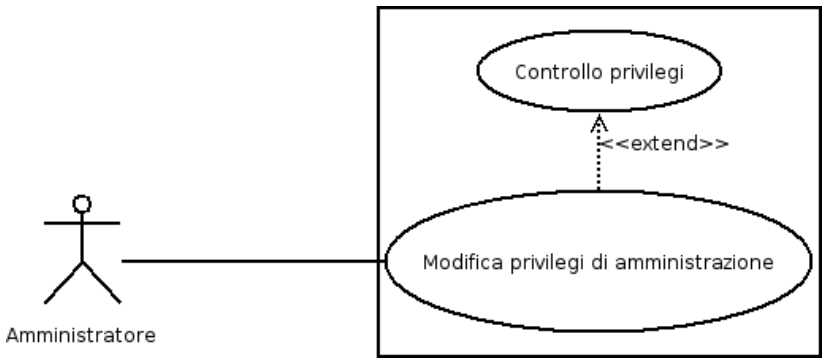

Figura 3.1: *Modifica i privilegi di amministrazione*

3.1.2 Creazione e aggiornamento di progetto

Sia nella creazione di un progetto nuovo che nell'aggiornamento di uno vecchio si utilizza la stessa azione di inserimento modulo. Nella creazione di progetto l'utente ottiene automaticamente i privilegi su di esso, invece nell'aggiornamento, prima di procedere, bisogna controllare se l'utente ha i privilegi per effettuare tale operazione.

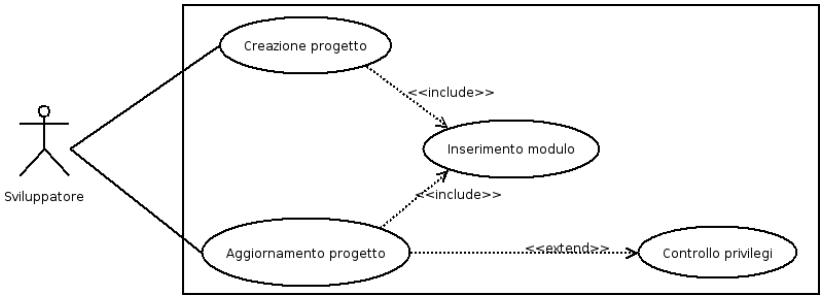

Figura 3.2: *Creazione e aggiornamento di progetto*

3.1.3 Elimina progetto

Per procedere nell'eliminazione di un progetto bisogna controllare se l'utente ha i privilegi per effettuare tale operazione.

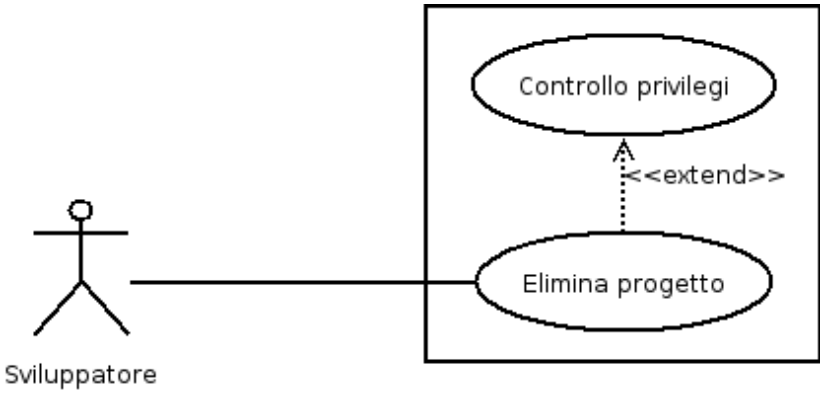

Figura 3.3: *Elimina progetto*

3.1.4 Rinomina progetto

Per procedere nella rinominazione del titolo, della piattaforma o della descrizione di un progetto bisogna controllare se l'utente ha i privilegi per effettuare tali operazioni.

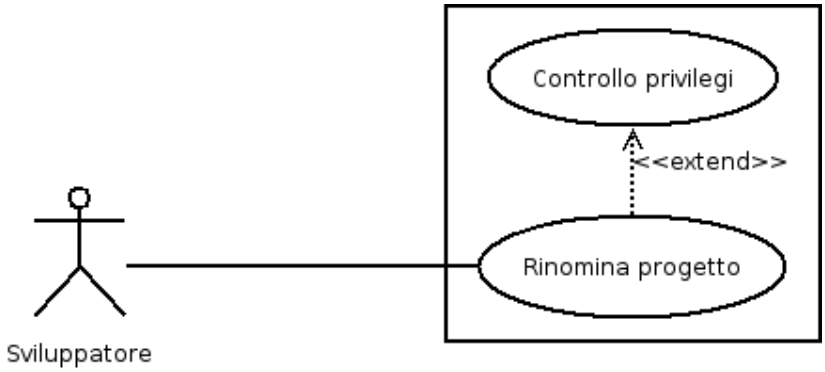

Figura 3.4: *Rinomina progetto*

3.1.5 Modifica privilegi di progetto

Per modificare i privilegi di progetto bisogna controllare i privilegi sia all'utente che effettua l'operazione, sia all'utente che la subisce. Dopo aver fatto ciò si può procedere nella modifica.

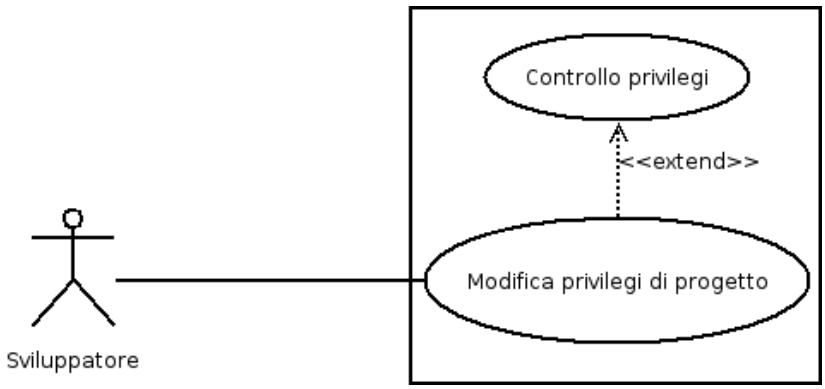

Figura 3.5: *Modifica privilegi di progetto*

3.1.6 Download, validazione ed eliminazione di rilascio

Chiunque può accedere alla pagina di download di un rilascio. Però quando si accede a tale pagina l'amministratore del sito ha anche la possibilità di validare o eliminare il rilascio. Quindi per effettuare queste due operazioni bisogna controllare se l'utente dispone dei privilegi di amministrazione.

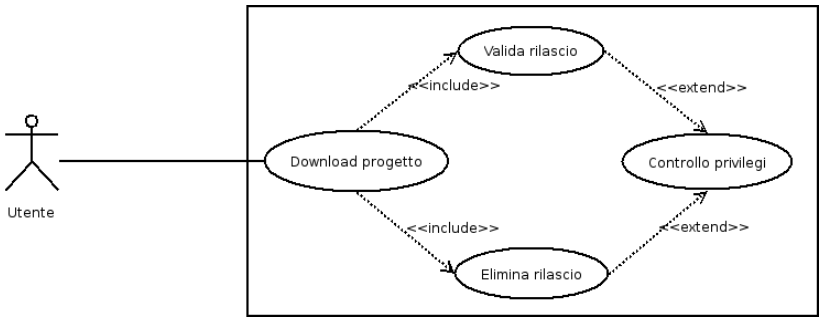

Figura 3.6: *Download, validazione ed eliminazione di rilascio*

3.2 File generici e di classe

begin.php: inizia le istruzioni richieste in tutte le pagine

begin_lock.php: inizia le istruzioni richieste in tutte le pagine quando deve essere presente un progetto bloccato

begin_unlock.php: inizia le istruzioni richieste in tutte le pagine quando deve essere presente un progetto non bloccato

home.php: home page dove scegliere l'azione da fare

classChange.php: contiene la classe classChange

classDB.php: contiene la classe classDB

classFile.php: contiene la classe classFile

classModule.php: contiene la classe classModule

classPriv.php: contiene la classe classPriv

classProject.php: contiene la classe classProject

classRelease.php: contiene la classe classRelease

classString.php: contiene la classe classString

classTag.php: contiene la classe classTag

classUser.php: contiene la classe classUser

3.3 Classi con relativi metodi e variabili

classChange: interfaccia le query con la tabella Changes

- classDB $db: interfaccia le query con il database
- int $id: numero id dell'utente che fa un'azione

- classChange(int $id)
  - int $id: id utente
- void insert(string $action, string $specify): inserisce il cambiamento nella tabella Changes
  - string $action: tipo d'azione
  - string $specify: specifica il tipo di azione

classDB: interfaccia le query con il database

- resource $connect: risorsa di connessione che è necessaria per le altre funzioni di PostgreSQL
- string $dbname: nome del database
- string $host: nome dell'host
- string $password: password
- int $port: numero della porta
- string $user: nome utente

- classDB()
- bool close(): chiude una connessione non persistente

- resource connect(): apre la connessione
- array fetchArray(resource $query): restituisce un array che corrisponde alla riga presa
  - resource $query: risultato della query
- int numRows(resource $query): restituisce il numero di righe in una risorsa risultato PostgreSQL
  - resource $query: risultato della query
- resource query(string $sql): apre la connessione, esegue la query, chiude la connessione
  - string $sql: query sql

classFile: operazioni con i file caricati

- bool $empty: vero se il file è vuoto
- string $extension: estensione del file
- array $file: file caricato
- string $name: nome del file
- string $tmp_name: path del file caricato sul server

- classFile(array $file)
  - array $file: file caricato
- void alertUploaded(): message se il file non è caricato
- string extension(): estensione del file
- bool getEmpty(): vero se il file è vuoto
- bool getExt(): estensione del file

- bool getName(): nome del file
- bool getTmp_name(): file caricato
- bool uploaded(): vero se il file è caricato, altrimenti falso

classModule: operazioni con i moduli

- classDB $db: interfaccia le query con il database
- string $name: nome del modulo
- int $present: numero della versione presente
- string $title: titolo del progetto

- classModule(string $title, string $name)
  - string $title: titolo del progetto
  - string $name: nome del modulo
- void alertExists(string $bool): messaggio se il modulo esiste o no
  - string $bool: può essere vuoto o contenere “not”
- void alertVoid(string $go): messaggio se non ci sono moduli inseriti
  - string $go: pagina di destinazione
- int future(): numero di versione del modulo in futuro
- string getName(): nome del modulo
- int getPresent(): numero di versione del modulo (-1 se non esiste)
- string getTitle(): titolo del progetto
- resource info(): query con l’info del modulo
- resource infoComplete(): query con l’info completa del modulo

- void insert(int $n1, int $n2, int $n3, string $type, int $version, string $extension, string $author, string $description): inserisce il modulo nel database
  - int $n1: primo numero del rilascio
  - int $n2: secondo numero del rilascio
  - int $n3: terzo numero del rilascio
  - string $type: tipo di modulo (“comp”, “docs”, “srcs”)
  - int version: numero di versione del modulo
  - string $extension: estensione del file del modulo
  - string $author: autore del modulo
  - string $description: descrizione del modulo
- string maxVersion(): stringa sql con la massima versione del modulo
- int number(): numero di moduli nel progetto
- int present(): numero di versione del modulo adesso

classPriv: operazioni con i privilegi

- classDB $db: interfaccia le query con il database
- int $id: numero id dell’utente che cambia i privilegi
- classPriv(int $id)
  - int $id: id utente
- bool admin(): vero se l’utente è un amministratore
- void adminUpdate(int $value): aggiorna i privilegi di admin nel database
  - int $value: può essere 0 (l’utente non sarà amministratore) oppure 1 (l’utente sarà amministratore)
- void alertAdmin(): messaggio se l’utente non ha i privilegi di amministratore

- void alertModuleAdd(string $name): messaggio se l'utente non ha i privilegi per aggiungere il modulo
  - string $name: nome del modulo
- void alertModulePriv(string $name): messaggio se l'utente non ha i privilegi per cambiare i privilegi al modulo
  - string $name: nome del modulo
- void alertProject(string $title): messaggio se l'utente non ha i privilegi per accedere al progetto
  - string $title: titolo del progetto
- void alertVoid(): messaggio se non ci sono altri utenti
- void alertYou(): messaggio se l'utente cambia i propri privilegi
  - bool apm(string $title): vero se l'utente ha privilegi sul progetto
- string getId(): numero id
- void modDelete(string $title, string $name): cancella i privmod nel database
  - string $title: titolo del progetto
  - string $name: nome del modulo
- void modInsert(string $title, int $n1, int $n2, int $n3, string $name, string $type, int $version): inserisce i privmod nel database
  - string $title: titolo del progetto
  - int $n1: primo numero del rilascio
  - int $n2: secondo numero del rilascio
  - int $n3: terzo numero del rilascio
  - string $name: nome del modulo

- string $type: tipo di modulo (“comp”, “docs”, “srcs”)
- int $version: numero di versione del modulo

- bool module(string $title, string $name): vero se l’utente è amministratore o ha i privilegi del progetto o del modulo
  - string $title: titolo del progetto
  - string $name: nome del modulo
- bool moduleOnly(string $title, string $name): vero se l’utente ha i privilegi sul modulo
  - string $title: titolo del progetto
  - string $name: nome del modulo
- bool modules(string $title): vero se l’utente ha i privilegi sui moduli del progetto
  - string $title: titolo del progetto
- void proDelete(string $title): cancella i privpro nel database
  - string $title: titolo del progetto
- void proInsert(string $title): insert privpro nel database
  - string $title: titolo del progetto
- bool project(string $title): vero se l’utente ha i privilegi del progetto
  - string $title: titolo del progetto
- void setId(int $id): cambia il numero id
  - int $id: id dell’utente

classProject: operazioni con i progetti

- classDB $db: interfaccia le query col database

- string $home: home page
- int $id: numero id dell'utente che accede al progetto
- int $timeout: timeout per bloccare il progetto
- string $title: titolo del progetto

• classProject(string $title, int $id):

  - string $title: titolo del progetto
  - int $id: id utente

• void alertDownload(): messaggio se l'utente non può scaricare il progetto

• void alertExists(): messaggio se esiste già un progetto con lo stesso titolo

• void alertLock(int $user): messaggio se il progetto è bloccato da un'altro utente

  - int $user: id utente

• void alertVoid(string $version): messaggio se il progetto non ha moduli

  - string $version: versione del progetto ("present" oppure un numero)

• void deletes(): cancella il progetto nel database

• bool exists(): vero se il progetto esiste

• string getTitle(): titolo del progetto

• array info(): informazioni del progetto

• void insert(string $platform, string $description): inserice un progetto nel database

  - string $platform: piattaforma del progetto
  - string $description: descrizione del progetto

• void lock(): incrementa lock se lock è uguale a 0

- void lockTimeout(): blocca il progetto se è scaduto il timeout
- resource modules(): query con i moduli del progetto
- int number(): numero di progetti
- resource projects(): query con i progetti
- void register(): blocca il progetto e lo registra nella sessione
- resource releases(): query con i rilasci del progetto
- void renames(): rinomina il progetto nel database
    - string $title: nuovo titolo del progetto
    - string $platform: nuova piattaforma del progetto
    - string $description: nuova descrizione del progetto
- void setTitle(string $title): cambia il progetto
    - string $title: titolo del progetto
- void unlock(): decrementa lock
- int unvalids(): numero dei rilasci non validati
- bool valid(): vero se il progetto contiene almeno un rilascio valido

classRelease: operazioni con i rilasci

- classDB $db: interfaccia le query con il database
- array $last: ultimo rilascio
- resource $lastModules: query con i moduli dell'ultimo rilascio
- int $n1: primo numero del rilascio
- int $n2: secondo numero del rilascio
- int $n3: terzo numero del rilascio

- int $number: numero di rilasci
- string $title: titolo del progetto

- classRelease(string $title, int $n1, int $n2, int $n3)
  - string $title: titolo del progetto
  - int $n1: primo numero del rilascio
  - int $n2: secondo numero del rilascio
  - int $n3: terzo numero del rilascio
- array AuthorDescription(string $name): riga con descrizione e autore dell'ultimo modulo
  - string $name: nome del modulo
- void deletes(): cancella i l rilascio nel database
- resource document(): query con il documento del rilascio
- array getLast(): riga con l'ultimo rilascio
- resource getLastModules(): query con i moduli dell'ultimo rilascio
- int getNumber(): numero di rilasci
- string getTitle(): titolo del progetto
- array getVersion(): versione del rilascio ("n1", "n2", "n3")
- void insert(): inserisce il rilascio nel database
  - string $state: stato del rilascio
- void insertDoc(string $extension): inserisce i documenti del rilascio nel database
  - string $extension: estensione del file di documento
- array last(): riga con l'ultimo rilascio

- resource lastModules(): query con i moduli dell'ultimo rilascio
- resource modules(): query con i moduli del rilascio
- int number(): numero di rilasci
- int numberLastModules(): numero degli ultimi moduli del rilascio
- bool type(string $type): vero se il rilascio contiene almeno un tipo ("docs", "comp" or "srcs")
- bool valid(): vero se il rilascio è valido
- void validate(): valida il rilascio

classString: operazioni con le stringhe

- bool $empty: vero se la stringa è vuota
- string $value: valore della stringa
- classString()
    - string $string: stringa inserita
- void alert(): messagio se il valore della stringa non è valido
- string get(): valore della stringa
- bool getEmpty(): vero se la stringa è vuota
- bool valid(): vero se la stringa è valida

classTag: stampa i tags html

- void alert(string $msg, string $go, string $to):stampa un alert javascript e torna alla pagina di destinzione
    - string $msg: messaggio di alert
    - string $go: pagina di destinazione

  - string $to: informazioni per la GET HTTP

- void foot(): stampa il tag di coda di una pagina html con i link per tornare indietro
- void head(): stampa una il tag di testa di una pagina html
- void headLink(): stampa i tag di testa di una pagina html con i link di navigazione
- string nominative(): nome, cognome e login dell'utente
  - int $id: id utente
  - object $log: utente
- void url(string $go): va alla url
  - string $go: pagina di destinazione
- void w3c(): stampa il tag del validatore del w3c in una pagina html

classUser: operazioni con gli utenti

- classDB $db: interfaccia le query con il database
- int $id: numero id dell'utente
- classUser(int $id)
  - int $id: id utente
- string getId(): numero id dell'utente
- void insert(): inserisce un utente se non è presente

3.4 File per creare e aggiornare un progetto

create.php: crea un nuovo progetto

⇒ string $_GET[“description”]: descrizione del progetto

⇒ string $_GET[“platform”]: piattaforma del progetto

⇒ string $_GET[“title”]: titolo del progetto

- void make_dir(string $title): crea una directory per il progetto

  - string $title: titolo del progetto

- void NP_create(classPriv $priv, string $title, string $platform, string $description, string $go, int $id): va alla pagina seguente dopo aver inserito il progetto nel database e nel filesystem

  - classPriv $priv: privilegi dell'utente
  - string $title: titolo del progetto
  - string $platform: piattaforma del progetto
  - string $go: pagina di destinazione
  - int $id: id utente

modules_add.php: aggiunge un modulo negli array dei moduli inseriti (aggiornati o creati)

⇒ string $_POST[“name”]: nome del modulo da aggiungere

⇒ array $_POST[“new”]: moduli nuovi

⇒ array $_POST[“old”]: moduli vecchi

⇒ array $_POST[“upd”]: moduli aggiornati

- array array_add(array $array, classModule $module, classFile$comp, classFile $docs, classFile $srcs, string $author, string $description): aggiunge il modulo inserito

  - array $array: contiene il modulo inserito (aggiornato o creato)
  - classModule $module: modulo inserito
  - classFile $comp: compilati
  - classFile $docs: documentati
  - classFile $srcs: sorgenti
  - string $author: autore del modulo
  - string $description: descrizione del modulo

- array array_dec(array $array, string $name): toglie il modulo inserito (aggiornato o creato)
  - array $array: modulo inserito
  - string $name: nome del modulo

- void move_file_ins(classFile $file, string $title, string $name, string $type, string $version): muove i file nella cartella "tmp"
  - classFile $file: file da muovere
  - string $title: titolo del progetto
  - string $name: nome del modulo
  - string $type: tipo del modulo ("docs", "comp", "srcs")
  - string $version: versione del modulo

- void rm_files_rewrited(array $upd, array $new, string $title, string $name): rimuove i file dei moduli riscritti dal filesystem
  - array $upd: moduli aggiornati
  - array $new: moduli creati
  - string $title: titolo del progetto
  - string $name: nome del modulo

modules_choose.php: scegliere i moduli da importare nella nuova versione

⇒ int $_GET[“numod”]: numero di moduli da importare

⇒ string $_GET[“0”]: modulo da importare (ad ogni modulo corrisponde un numero che va da 0 a $_GET[“numod”]-1)

- void NP_modules_choose(string $title, string $go, int $id, string $to): va alla pagina seguente

  - string $title: titolo di progetto
  - string $go: pagina di destinazione
  - int $id: id utente
  - string $to: informazioni per la GET HTTP

- string TO_modules_choose(int $numod): moduli vecchi che saranno importati nella nuova versione

  - int $numod: numero di moduli

module_features.php: inserisce i file e le informazioni del modulo

⇒ string $_POST[“insert”]: indica se il modulo è nuovo o aggiornato

⇒ string $_POST[“name_new”]: nome del modulo (se nuovo)

⇒ string $_POST[“name_upd”]: nome del modulo (se aggiornato)

⇒ array $_POST[“new”]: moduli nuovi

⇒ array $_POST[“old”]: moduli vecchi

⇒ array $_POST[“upd”]: moduli aggiornati

modules_insert.php: inserisce il nome del modulo del modulo da inserire

⇒ array $_GET[“new”]: moduli nuovi

⇒ array $_GET[“old”]: moduli vecchi

⇒ array $_GET[“upd”]: moduli aggiornati

- void print_hidden(array $old, array $upd, array $new): stampa i moduli nella form

  - array $old: moduli vecchi
  - array $upd: moduli aggiornati
  - array $new: moduli nuovi

- array print_ins(string $action, string $array): stampa i moduli inseriti

  - string $action: se i moduli saranno nuovi o aggiornati
  - string $array: moduli dell'azione (nuovi o aggiornati)

- array print_old(array $array): visualizza i moduli importati

  - array $array: moduli importati

- void print_radio(object $db, int $query): stampa i radio button per scegliere se i modui saranno nuovi o aggiornati

  - classDB $db: interfaccia le query con il database
  - resource $query: query con i vecchi moduli

modules_save.php: scegliere e salvare la versione del rilascio, lo stato del rilascio e il documento del rilascio

⇒ array $_POST["new"]: moduli nuovi

⇒ array $_POST["old"]: moduli vecchi

⇒ array $_POST["upd"]: moduli aggiornati

⇒ string $_POST["state"]: stato del rilascio

⇒ array $_POST["version"]: versione del rilascio

- void copy_file(classRelease $release, string $state, string $name, string $type, int $versionM, string $extension): copia i moduli nel filesystem

  - classRelease: rilascio
  - string $state: stato del rilascio
  - string $name: nome del modulo
  - string $type: tipo del rilascio
  - int $versionM: versione del modulo
  - string $extension: estensione del file del modulo

- void gzip(classRelease $release, string $state).comprime i file del rilascio
  - classRelease: rilascio da comprimere
  - string $state: stato del rilascio

- void insert_doc(): inserisce la documentazione del rilascio nel database
  - classFile $file: documentazione
  - classRelease $release: rilascio

- void insert_ins(classRelease $release, string $state, array $array, int $id): inserisce i moduli nel database e nel filesystem
  - classRelease $release: rilascio
  - string $state: stato del rilascio
  - array $array: moduli inseriti (aggiornamenti o nuovi)
  - int $id: id utente

- void insert_old(classRelease $release, string $state, array $array): inserisce i vecchi moduli nel database
  - classRelease: $release: rilascio
  - string $state: stato del rilascio
  - array $array: vecchi moduli

- void move_file_doc(classFile $file, classRelease $release, string $state): sposta il file nel nella diretory del progetto

  - classFile $file: documentazione
  - classRelease $release: rilascio
  - string $state: stato del rilascio

- void NP_save(classProject $project, array $old, array $upd, array $new, array $version, string $state, int $id): va alla pagina home page dopo aver inserito il rilascio e i moduli nel database e nel filesystem

  - classProject $project: progetto
  - array $old: moduli vecchi
  - array $upd: moduli aggiornati
  - array $new: moduli nuovi
  - array $version: versione del rilascio
  - string $state: stato del rilascio
  - int $id: id utente

- array version_pres(string $title, array $old, array $upd, array $new): presente versione del rilascio (“n1”, “n2”, “n3”)

  - string $title: titolo del progetto
  - array $old: moduli vecchi
  - array $upd: moduli aggiornati
  - array $new: nuovi moduli

3.5 File per cancellare un progetto

delete.php: cancella il progetto

⇒ string $_POST[“title”]: titolo del progetto

- void NP_delete(string $title, int $id): va alla home page dopo aver cancellato il progetto

  - string $title: titolo del progetto
  - int $id: id utente

3.6 File per rinominare un progetto

rename.php: rinomina il titolo, la piattaforma e la descrizione del progetto

⇒ string $_GET["description"]: vecchia descrizione del progetto

⇒ string $_GET["descriptionN"]: nuova descrizione del progetto

⇒ string $_GET["platform"]: vecchia piattaforma del progetto

⇒ string $_GET["platformN"]: nuova piattaforma del progetto

⇒ string $_GET["title"]: vecchio titolo del progetto

⇒ string $_GET["titleN"]: nuovo titolo del progetto

- void NP_rename(string $titleN, string $titleO, string $platformN, string $platformO, string $descriptionN, string $descriptionO, string $id): va alla home page dopo aver rinominato il titolo e la piattaforma del progetto

  - string $titleN: nuovo titolo del progetto
  - string $titleO: vecchio titolo del progetto
  - string $platformN: nuova piattaforma del progetto
  - string $platformO: vecchia piattaforma del progetto
  - string $descriptionN: nuova descrizione del progetto
  - string $descriptionO: vecchia descrizione del progetto
  - string $id: id utente

- void rename_files(string $old, string $new): rinomina i file dei rilasci e di documentazione

- string $old: vecchio titolo del progetto

- string $new: nuovo titolo del progetto

3.7 File per gestire i privilegi di progetti e moduli

privilege.php: scegliere quale progetto deve cambiare i privilegi

⇒ string $_GET[“title”]: titolo del progetto

privilege_module.php: cambia i privilegi dei moduli di un utente

⇒ string $_GET[“title”]: titolo del progetto

⇒ string $_GET[“name”]: nome del modulo

⇒ int $_GET[“user”]: id utente

⇒ string $_GET[“value”]: contiene “YES” se l’utente ha i privilegi del modulo, altrimenti “NO”

- void mod_insert(string $title, string $name, classPriv $priv): inserisce i privilegi dei moduli nel database

  - string $title: titolo del progetto

  - string $name: nome del modulo

  - classPriv $priv: privilegi

privilege_project.php: cambia i privilegi di progetto di un utente

⇒ string $_GET[“title”]: titolo del progetto

⇒ string $_GET[“value”]: contiene “YES” se l’utente ha i privilegi del progetto, altrimenti “NO”

privilege_user.php: scegliere quale utente deve cambiare i privilegi

⇒ string $_GET[“title”]: titolo del progetto

⇒ int $_GET[“user”]: id utente

- string print_modules_privilege(int $user, string $title, string $name, string $go): stampa i privilegi nella colonna dei moduli
  - int $user: id utente
  - string $title: titolo del progetto
  - string $name: nome del modulo
  - string $go: pagina di destinazione
- string print_project(int $user, string $title, string $go): stampa i privilegi nella colonna dei progetti
  - int $user: id utente
  - string $title: titolo del progetto
  - string $go: pagina di destinazione

3.8 File per gestire i privilegi di amministrazione

priv_admin.php: scegliere quale utente deve cambiare i privilegi di amministatore

- void print_privilege(int $user, string $go): stampa i privilegi nella colonna di amministrazione
  - int $user: id utente
  - string $go: pagina di destinazione

priv_admin_change.php: cambia i privilegi di amministratore

⇒ int $_GET[“user”]: id utente

⇒ string $_GET[“value”]: contiene “YES” se l’utente è amministratore, altrimenti “NO”

3.9 File per effettuare i download

download.php: elenco dove scegliere il rilascio del progetto da scaricare

⇒ string $_GET[“title”]: titolo del progetto

- void print_description(class Project $project): stampa la descrizione del progetto

  - classProject $project: progetto da scaricare

downoload_file.php: header HTTP per il download

⇒ string $_GET[“title”]: titolo del progetto

⇒ string $_GET[“filename”]: file da scaricare

downoload_release.php: tabella dove scegliere il file da scaricare

⇒ array $_GET[“release”]: titolo e rilascio da scaricare

- void ad(string $author, string $description): stampa l’autore e la descrizione del modulo

  - string $author: autore del modulo
  - string $description: descrizione del modulo

- string filename_document(array $row, string $state): file di documentazione

  - array $row: documentazione
  - string $state: stato del rilacio

- string filename_module(array $row): file del modulo

  - array $row: modulo

- string filename_release(classRelease $release, string $type, string $state): file del rilascio

  - classRelease $release: rilascio

- string $type: tipo del rilascio
- string $state: stato del rilascio

- void link_admin(classRelease $release, string $state, classPriv $priv, string $god, string $gov): collegamento per cancellare o validare un rilascio se l'utente è un amministratore
  - classRelease $release: rilascio
  - string $state: stato del rilascio
  - classPriv $priv: privilegi
  - string $god: collegamento per cancellare il rilascio
  - string $gov: collegamento per validare il rilascio
- void link_document(resource $query, string $state, string $go, classDB $db): collegamento scaricare la documentazione
  - resource $query: query della docuentazione
  - string $state: stato del rilascio
  - string $go: collegamento
  - classDB $db: oggetto della classe classDB
- void link_modules(int $query, string $go, classDB $db): collegamento al modulo da scaricare
  - resource $query: query del modulo
  - string $go: collegamento
  - classDB $db: oggetto della classe classDB
- void link_release(classRelease $release, string $type, string $state, string $go): collegamento per scaricare il tipo di rilascio
  - classRelease $release: rilascio
  - string $type: tipo del rilascio da scaricare

- string $state: stato del rilascio
- string $go: collegamento

- int print_modules(int $i, string $title, array $row, string $go): stampa i collegamenti ai moduli nella tabella. Restituisce la posizione del modulo
  - int $i: tipo stampato
  - string $title: titolo del progetto
  - array $row: modulo
  - string $go: collegamento
- void void_cell(): stampa una cella vuota

3.10 File per cancellare o approvare un rilascio

release_delete.php: cancella un rilascio

⇒ string $_GET[“title”]: titolo del progetto (che si vuole cancellare)

⇒ string $_POST[“title”]: titolo del progetto (che verrà cancellato)

release_validate.php: valida un rilascio

⇒ string $_POST[“title”]: titolo del progetto (che si vuole validare)

⇒ int $_GET[“n1”]: primo numero del rilascio (che si vuole validare)

⇒ int $_ GET [“n2”]: secondo numero del rilascio (che si vuole validare)

⇒ int $_ GET [“n3”]: terzo numero del rilascio (che si vuole validare)

⇒ sting $_ GET [“state”]: stato del rilascio (che si vuole validare)

⇒ string $_POST[“title”]: titolo del progetto (che verrà validato)

⇒ int $_POST[“n1”]: primo numero del rilascio (che verrà validato)

⇒ int $_POST[“n2”]: secondo numero del rilascio (che verrà validato)

⇒ int $_POST["n3"]: terzo numero del rilascio (che verrà validato)

⇒ string $_POST["state"]: stato del rilascio (che verrà validato)

- void next_page(string $title, int $n1, int $n2, int $n3, string $state, string $id, string $go): valida il rilascio e va alla home page

  - string $title: titolo del progetto
  - int $n1: primo numero del rilascio
  - int $n2: secondo numero del rilascio
  - int $n3: terzo numero di rilascio
  - string $state: stato del rilascio
  - string $id: id utente
  - string $go: pagina di destinazione

3.11 File per l'autenticazione

user_login.php: login utente

⇒ string $_POST["name"]: nome utente

⇒ string $_POST["password"]: password utente

user_logout: logout utente

Capitolo 4

Implementazione ed esempi d'uso

4.1 Login e home page

Prima di poter accedere all'applicazione bisogna autenticarsi. Per effettuare il login è necessario inserire il nome utente, che verrà fornito al momento della registrazione al laboratorio di Sistemi Informativi, e la rispettiva password.
Dopo essersi autenticati si accede alla home page del sito. Qui sono elencati i progetti

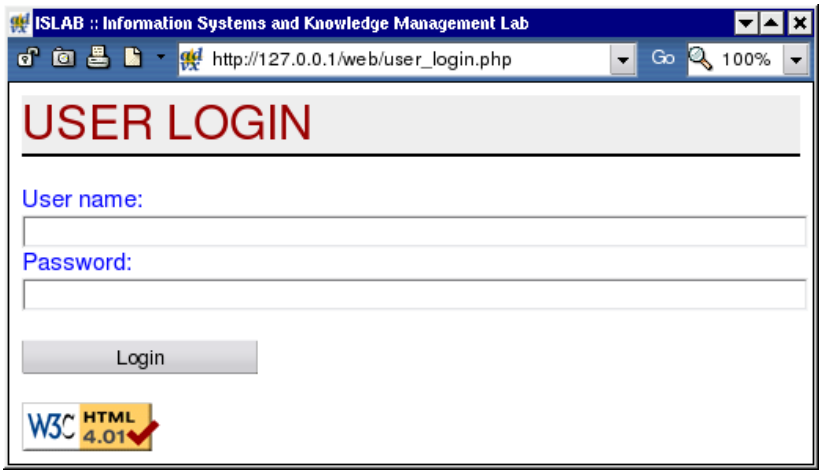

Figura 4.1: *Pagina di Login*

esistenti e si può scegliere che azione compiere su di essi, inoltre è presente un pulsante che permette la creazione di un nuovo progetto. Nel caso l'utente sia un amministratore del sito viene visualizzato un pulsante aggiuntivo per permettere di gestire i privilegi di amministrazione del sito.

Quando nel sito è presente un cospicuo numero di progetti ci si troverà con una

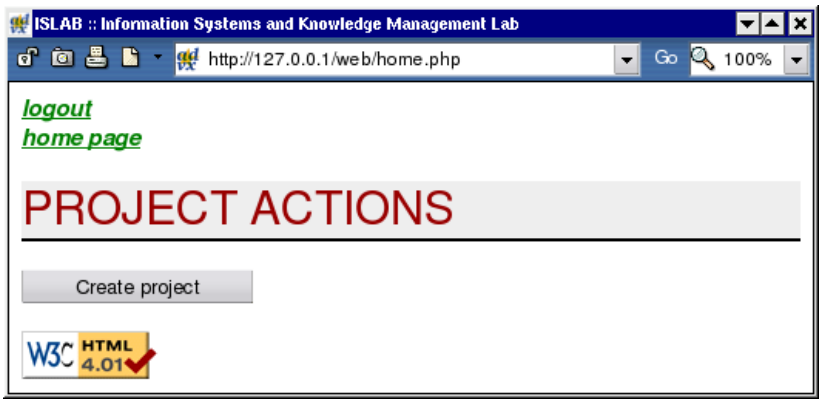

Figura 4.2: *Home page senza progetti vista da un utente semplice*

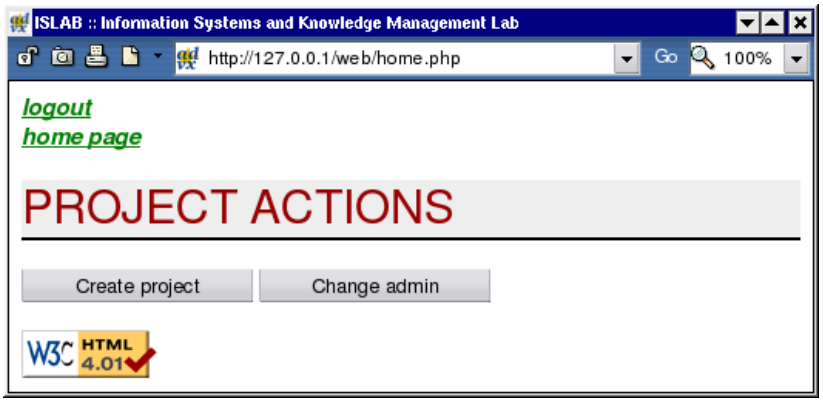

Figura 4.3: *Home page senza progetti vista da un amministratore*

home page completa dove ogni utente potrà scegliere l'azione da compiere in base ai propri privilegi.

4.2 Privilegi di amministratore

Un amministratore del sito può decidere se fornire o revocare ad altri utenti i privilegi di amministrazione. Cliccando sul pulsante **Change admin**, presente nella home page, si accede ad una pagina dove è presente una tabella con tutti gli utenti del sito.

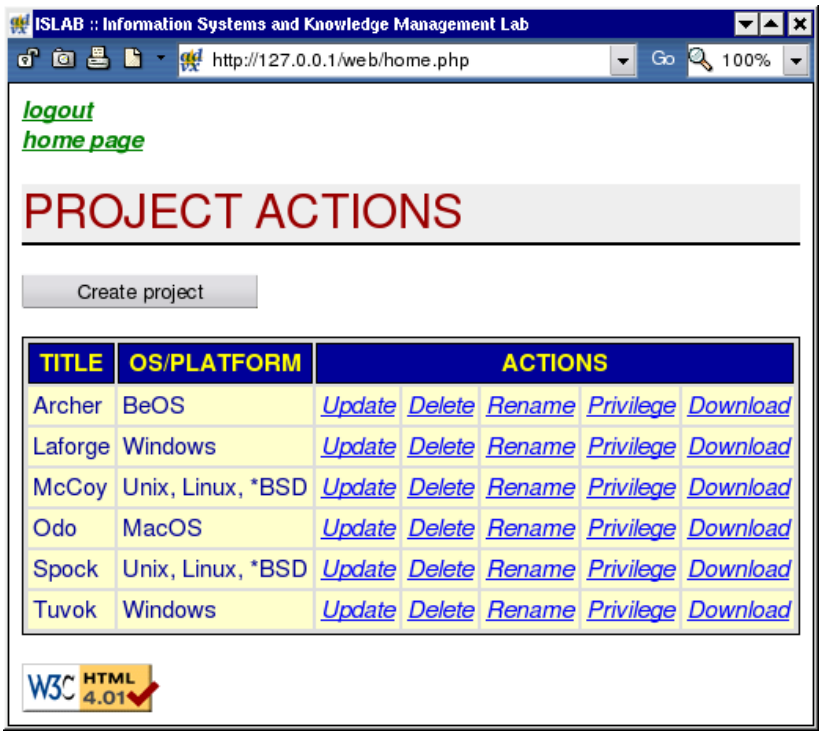

Figura 4.4: *Home page con sei progetti inseriti*

Nella prima colonna della tabella (**users**) è presente il nome, cognome e login degli utenti. Nella seconda colonna (**admin**) viene indicato se l'utente è amministratore del sito. Infine (**e-mail**) nella terza appare l'e-mail dell'utente. Per cambiare i privilegi è sufficiente cliccare sul link nella seconda colonna.
Se nella colonna **admin** è presente **NO** significa che l'utente non è amministratore. Per assegnare tali privilegi è sufficiente cliccare sul link presente nella seconda colonna, l'utente diviene amministratore e nella colonna **admin** verrà visualizzato **YES**. Per revocare i privilegi basta fare l'operazione inversa.

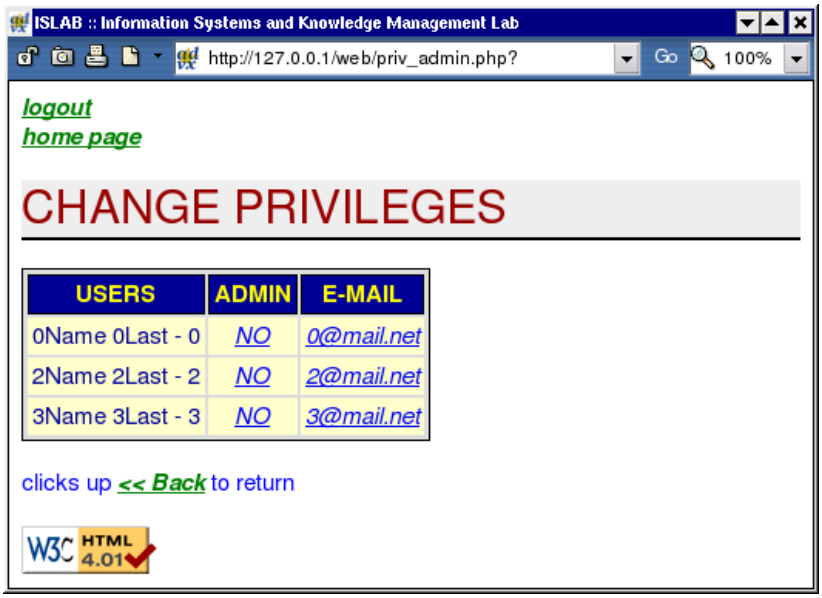

Figura 4.5: *Lista di utenti a cui poter cambiare i privilegi*

4.3 Struttura del filesystem

Le cartelle che contengono i progetti si trovano in una directory di nome archives. Questa directory viene creata automaticamente alla creazione del primo progetto e solo l'utente apache può scrivere all'interno di essa.
Una directory di progetto contiene al suo interno i file dei moduli che compongono un rilascio ed in più è presente una directory groups che raggruppa i file di ogni rilascio con le rispettive documentazioni.

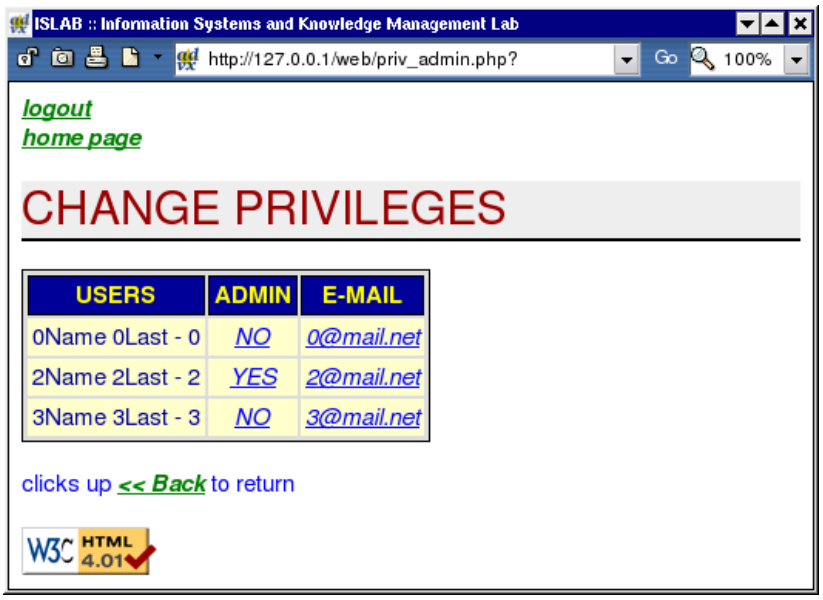

Figura 4.6: *Lista di utenti dopo aver fornito i privilegi all'utente* 2

Durante la creazione di un rilascio può essere creata una directory tmp che serve a contenere momentaneamente i file dei moduli prima che venga salvato il rilascio. Dopo il salvataggio i file vengono copiati nella directory del progetto e la directory tmp viene cancellata.

4.4 Creazione progetto

Cliccando sul pulsante **Create project**, presente nella home page, si accede ad una pagina dove è possibile creare un progetto.

Per fare una dimostrazione concreta ipotizziamo la creazione di un progetto di nome Prova che può essere eseguito su piattaforma *Unix*, *Linux* e **BSD*. Di questo progetto sarà creato un rilascio contenente tre moduli: Vedi, Scrivi e Controlla. Alla fine della creazione verrà inserito un file di documentazione di tutto il rilascio.

Per cominciare bisogna inserire il titolo del progetto, la piattaforma dove può essere eseguito e una breve descrizione (i campi con l'asterisco sono obbligatori).

Dopo aver cliccato sul pulsante **Submit** viene creata la directory archives, se questa non esiste, e all'interno di essa ci sarà la directory Prova (ha lo stesso nome del pro-

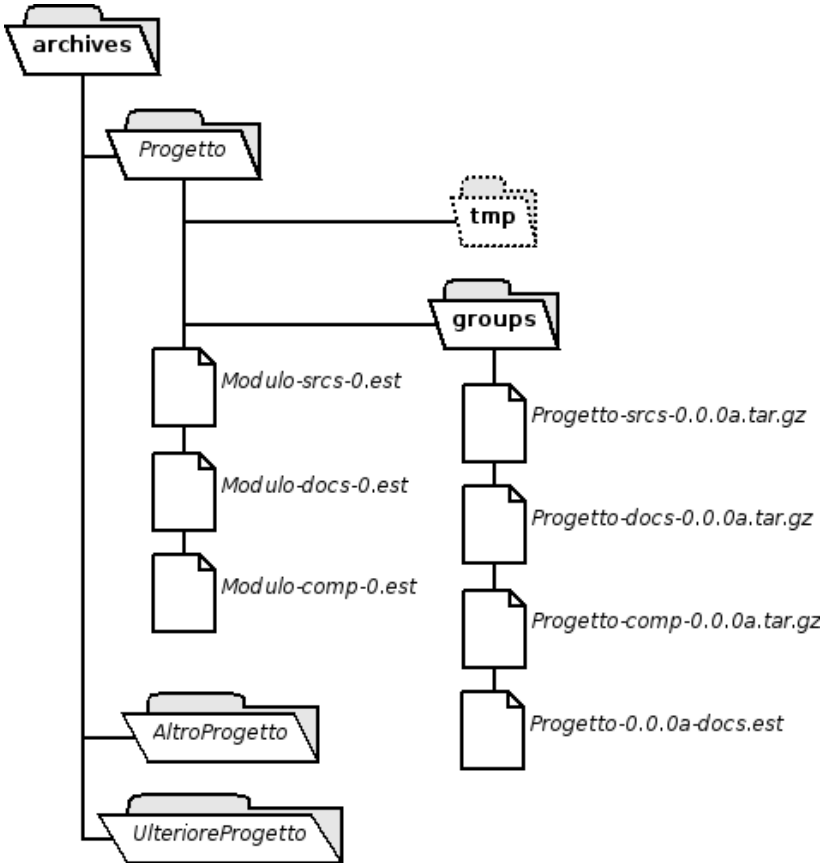

Figura 4.7: *Struttura dei progetti nel filesystem*

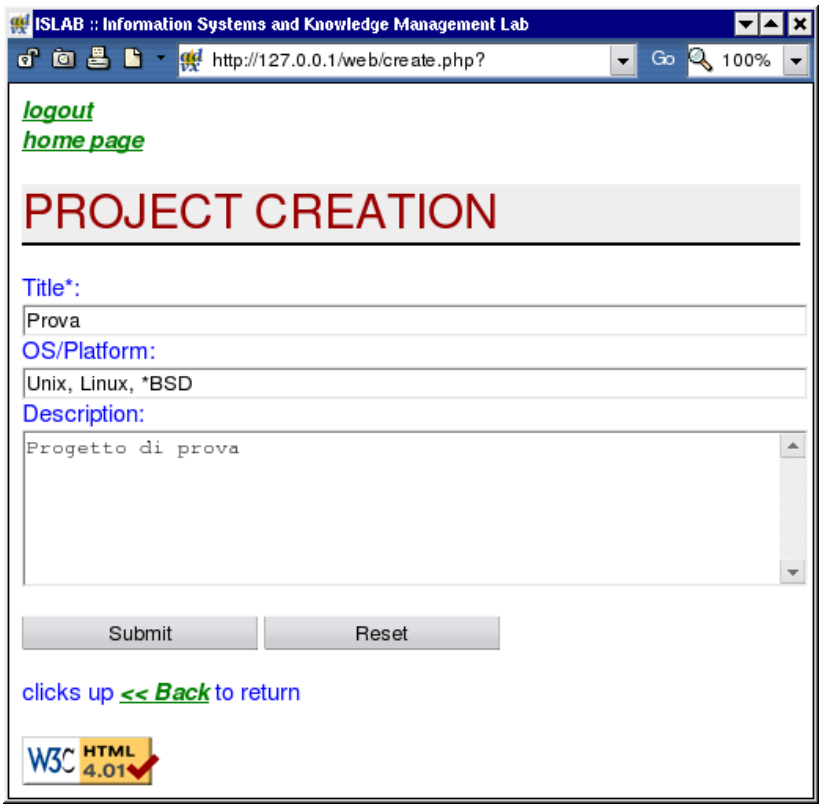

Figura 4.8: *Pagina dove creare il progetto*

getto), la quale a sua volta contiene la directory groups. A questo punto il progetto è stato creato e l'utente ne possiede tutti i privilegi.

Se non si vuole creare nessun rilascio del progetto basta cliccare sul link alla home page o al logout, altrimenti bisogna incominciare ad inserire i moduli. Per prima cosa dobbiamo inserire il nome del modulo (in questo caso Scrivi) e poi cliccare sul pulsante Submit.

Dopo aver inserito il nome bisogna inserire i file che compongono il modulo.

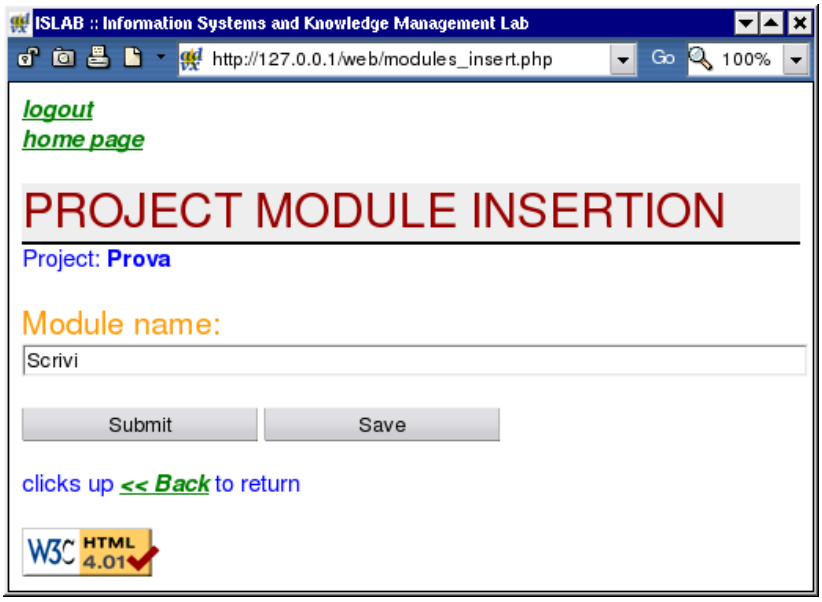

Figura 4.9: *Pagina dove inserire il nome del modulo*

I moduli sono stati divisi in tre tipi: sorgenti, documenti e compilati. Ogni tipo deve essere memorizzato in un file di archivio per permetterne una più facile gestione. Nel nostro esempio i file sorgenti del modulo Scrivi sono contenuti nel file sources.tar.gz, quelli di documentazione sono all'interno del file documented.zip ed i file compilati si trovano nel file d'archivio compilated.rar.

Oltre ai file bisogna inserire anche l'autore (obbligatorio) ed una breve descrizione (facoltativa) del modulo.

Cliccando sul pulsante **Submit** all'interno della directory Prova viene creata, se non esiste, una sottodirectory tmp dove vengono memorizzati i file di modulo che forme-

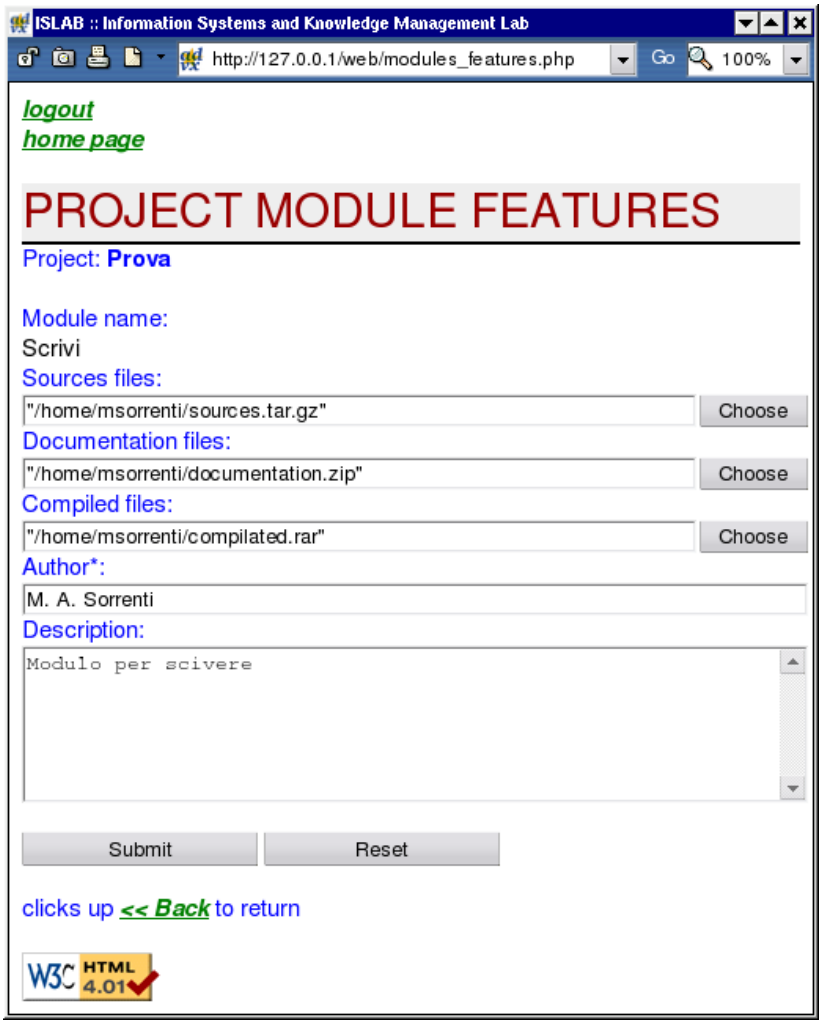

Figura 4.10: *Pagina dove inserire file, autore e descrizione del modulo*

ranno il futuro rilascio. In questo caso i file del modulo Scrivi vengono prima rinominati con nel seguente modo: nomemodulo-tipo-versionemodulo.estensione. Quindi i file sources.tar.gz, documented.zip e compilated.rar vengono rinominati rispettivamente in Scrivi-srcs-0.tar.gz, Scrivi-docs-0.zip e Scrivi-comp-0.zip. Dopodiché vengono copiati nella directory tmp.

A questo punto si ritorna nella pagina in cui occorre inserire il nome di un nuovo modulo. In questa pagina viene passato con il metodo POST un array associativo contente i nomi dei moduli inseriti, la versione e i file di estensione. La sintassi è la seguente:

```
(nome => (0 => versione, comp => .est, docs => .est, srcs => .est))
```

quindi nel nostro esempio l'array diventa:

```
(Scrivi => (0 => 0, comp =>.zip, docs =>.rar, srcs => .tar.gz))
```

Per inserire i moduli Vedi e Controlla verranno eseguiti gli stessi passi.

Se riteniamo che i moduli inseriti siano sufficienti il passo successivo è quello di salvare

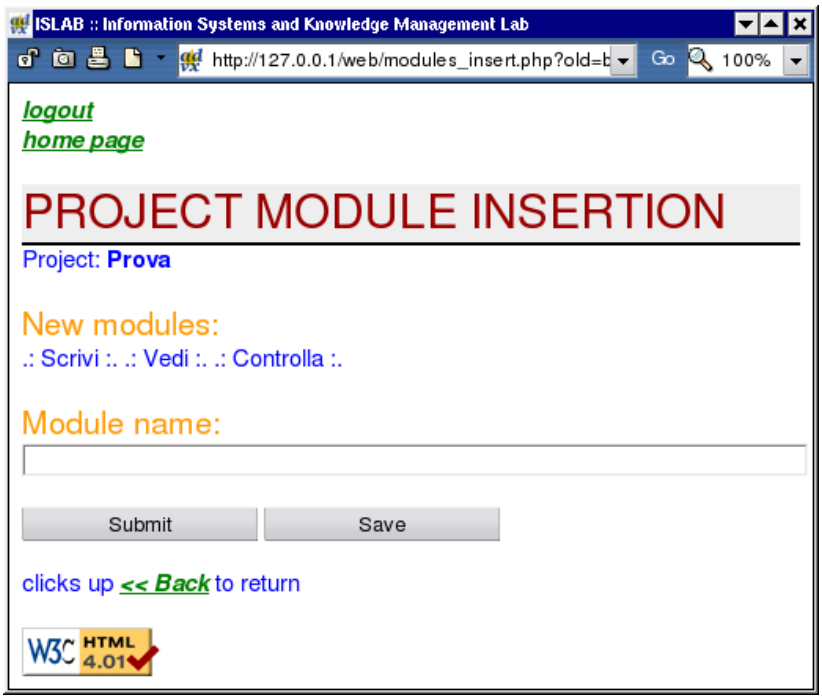

Figura 4.11: *Pagina di inserimento moduli dopo aver inserito* Scrivi, Vedi *e* Controlla

il tutto come un unico rilascio.

Occorre cliccare sul pulsante **Save** e si accede ad una pagina dove bisogna indicare la versione del rilascio, lo stato e un file di documentazione di tutto il rilascio.
La versione del rilascio deve essere scelta all'interno di una select, che fornisce due possibili opzioni. I numeri che compongono una versione da scegliere vengono calcolati dal computer in base ai moduli che sono stati inseriti (nel caso si tratti del primo rilascio del progetto i tre numeri saranno uguali a zero). Per i numeri dell'altra versione vengono presi quelli ottenuti dal calcolo precedente, il primo numero viene incrementato di uno mentre gli altri due vengono messi uguali a zero. La scelta tra quale dei due scegliere spetta all'utente che sta salvando il rilascio. Il primo numero di versione dovrebbe essere scelto in caso di aggiornamenti non troppo rilevanti, invece il secondo bisognerebbe sceglierlo quando tra il rilascio che si sta salvando e quello precedenti ci siano notevoli differenze.
Nell'esempio svolto fino a questo momento (trattandosi del primo rilascio del progetto) le due versioni che si possono scegliere sono: 0.0.0 e 1.0.0.
Anche lo stato del rilascio deve essere scelto all'interno di una select, in cui si hanno tre opportunità di scelta: alfa (viene aggiunta una lettera a alla versione), beta (viene aggiunta una lettera a alla versione) e stable (non viene aggiunta nessuna lettera alla versione).
Per il nostro esempio scegliamo di salvare il rilascio con versione 0.0.0 e stato beta.
Cliccando sul pulsante **Submit** viene creato il nuovo rilascio del progetto (in questo caso l'unico).
I file che contengono i moduli vengono spostati dalla directory **tmp** (che di conseguenza verrà cancellata) alla directory Prova. Il file di documentazione del rilascio viene rinominato con la seguente sintassi: nomeprogetto-versionestato-docs.estensione. Quindi nel caso preso in esame diventerà Prova-0.0.0b-docs.pdf e verrà memorizzato nella directory groups.
Inoltre vengono creati tre file di formato tar.gz che servono per raggruppare i tre diversi tipi dei moduli (sorgenti, compilati e documenti). Questi file sono creati con il seguente nome: nomeprogetto-tipo-versionestato.tar.gz (che vengono memorizzati nella directory groups) si chiameranno Prova-comp-0.0.0b.tar.gz, Prova-docs-0.0.0b.tar.gz e Prova-srcs-0.0.0b.tar.gz.

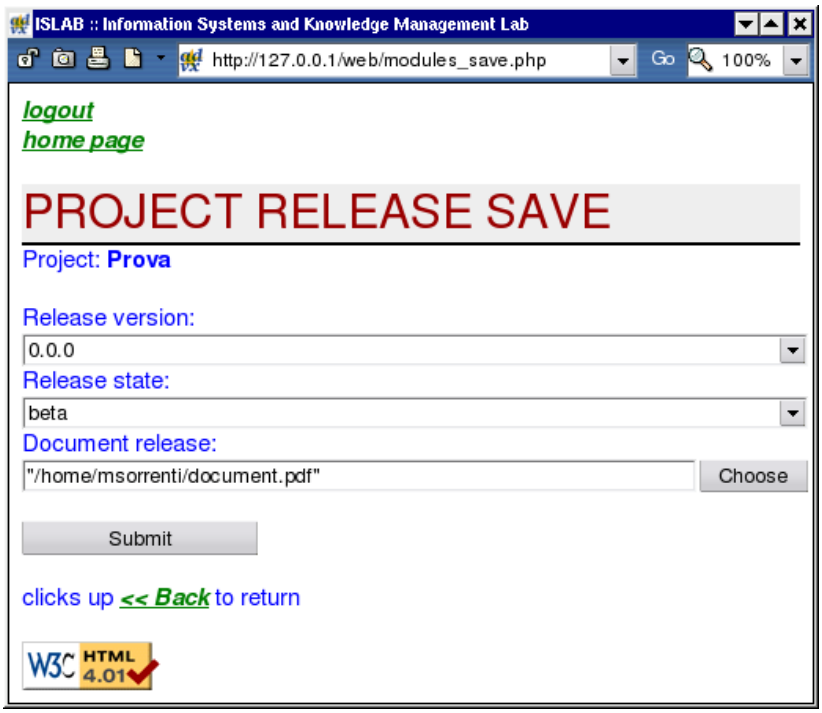

Figura 4.12: *Pagina dove salvare il rilascio*

A questo punto si torna alla home page e si possono avere due tipi di schermate a seconda che si sia amministratore del sito oppure no.
Agli amministratori del sito vengono evidenziati i progetti che hanno dei rilasci che devono essere ancora approvati, nel nostro caso abbiamo appena creato il rilascio 0.0.0b del progetto Prova, ma non potrà essere a disposizione finché non viene approvato.
La presenza di un rilascio da approvare viene segnalata da un colore diverso nel nome del progetto nella home page. Un utente normale non si accorge della presenza di un progetto non ancora approvato, infatti, esso rimarrà nascosto ai suoi occhi e se cerca di accedere ai rilasci presenti troverà solo quelli già approvati.

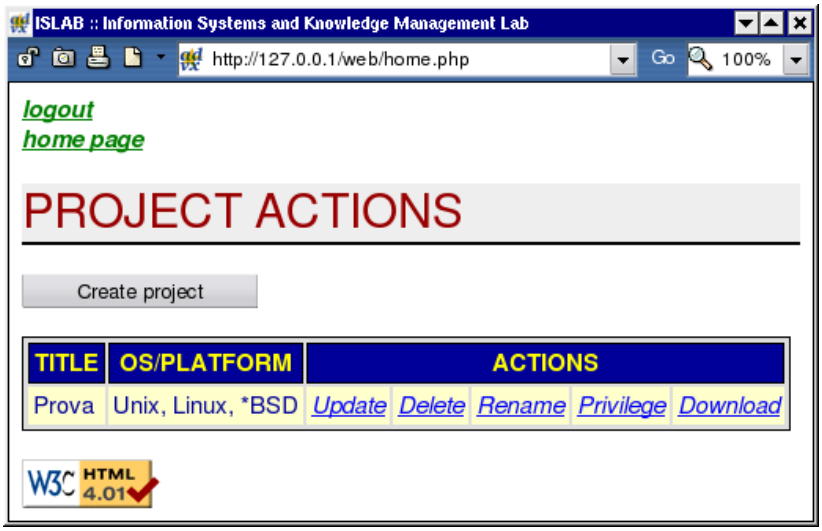

Figura 4.13: *Home page di un utente semplice dopo la creazione del progetto* Prova

4.5 Approvazione e download

Per approvare il rilascio di un progetto, l'amministratore del sito deve entrare nella sezione **Download**. Se un utente semplice prova ad entrare in tale sezione di un progetto che non contiene rilasci, o che contiene rilasci non ancora validi, viene restituito un errore (cosa che accadrebbe nel nostro caso siccome l'unico rilascio presente non è stato ancora approvato).
Entrando nella sezione **Download** è presente una select dove scegliere i rilasci da

Figura 4.14: *Home page di un amministratore dopo la creazione del progetto* Prova *(si evidenzia la presenza di un rilascio da approvare)*

scaricare, quelli non ancora approvati non sono visibili ad un utente normale, mentre all'amministratore del sito sono indicati con un asterisco.

Dopo aver selezionato il rilascio ci si trova di fronte ad una tabella con i link ai

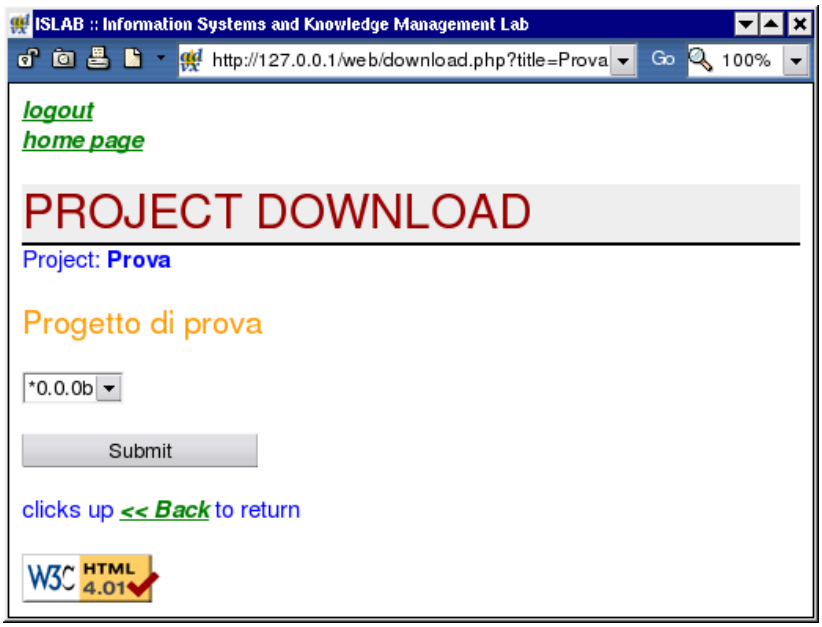

Figura 4.15: *Select con i rilasci da scaricare (l'asterisco indica all'amministratore che il rilascio è da approvare)*

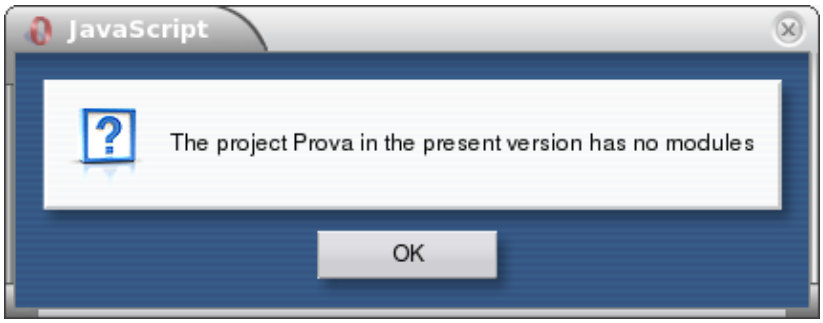

Figura 4.16: *Errore nell'effettuare un download non consentito*

moduli ed alla documentazione da scaricare. In più, se l'utente è un amministratore, è presente un link che permette la cancellazione del rilascio e l'approvazione se non è stato ancora approvato.

Cliccando su **DELETE IT** viene chiesto se si è sicuri di voler cancellare il rilascio,

Figura 4.17: *Pagina di download di un rilascio non ancora approvato*

in caso la risposta sia positiva il rilascio viene cancellato e si torna alla home page, in caso di risposta negativa, si torna alla pagina di download.

Cliccando su **VALIDATE IT** viene chiesto se si è sicuri di voler approvare il rilascio, in caso la risposta sia positiva il rilascio viene reso disponibile a tutti, altrimenti, il rilascio verrà cancellato.

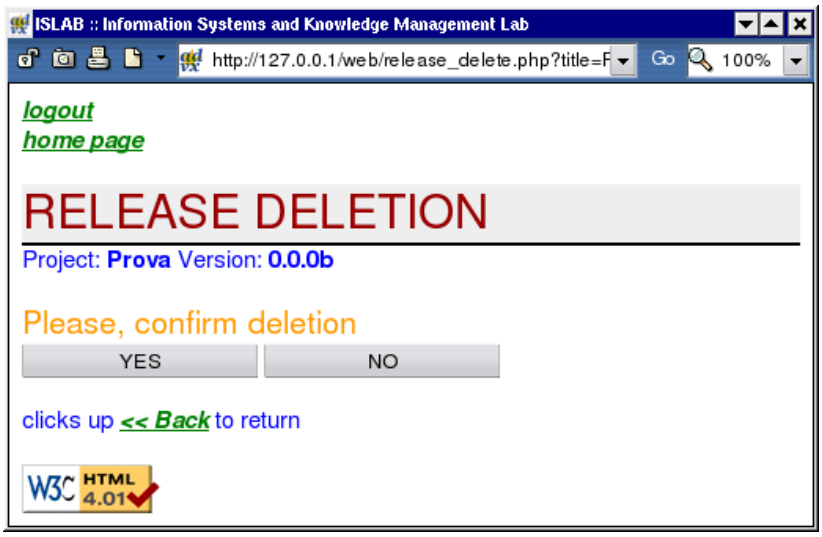

Figura 4.18: *Pagina dove viene chiesto se cancellare o no un rilascio*

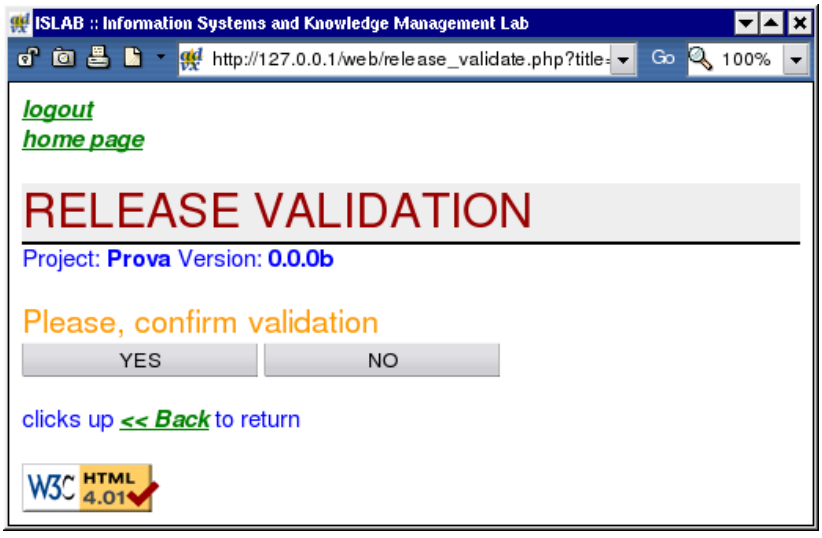

Figura 4.19: *Pagina dove viene chiesto se approvare o no un rilascio*

4.6 Aggiornamento di un progetto

Per aggiornare un progetto bisogna cliccare sul link **Update**, viene aperta una pagina dove sono elencati e selezionati i moduli che compongono l'ultimo rilascio. Se si vuole che uno o più moduli non facciano parte del futuro rilascio bisogna deselezionarli.
Dopo aver scelto i moduli da importare bisogna scrivere i nomi di quelli da inserire

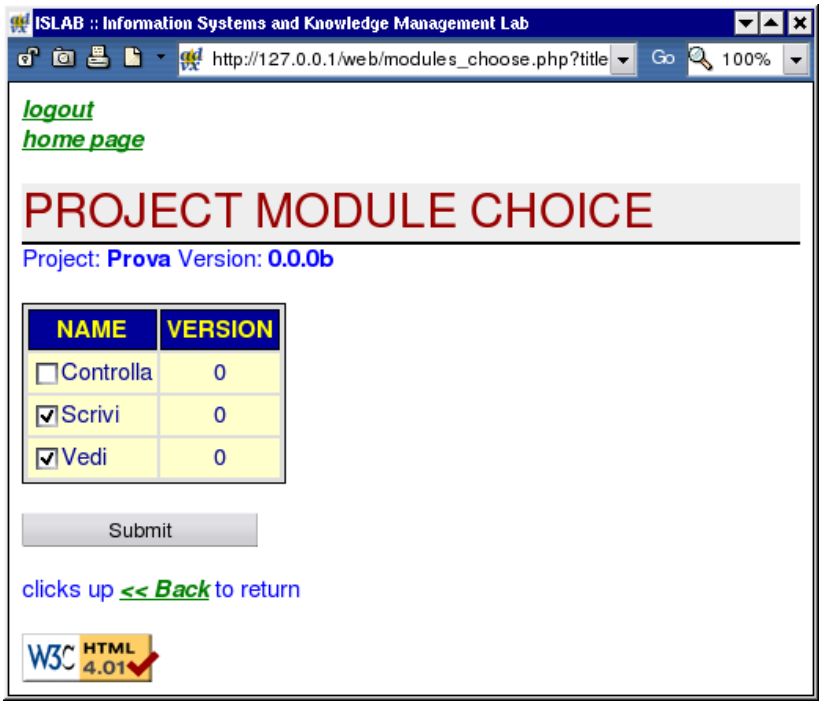

Figura 4.20: *Pagina dove scegliere i moduli da importare nel nuovo rilascio (il modulo* Controlla *non verrà importato)*

nella nuova versione. Nella stessa pagina compare un pulsante dove occorre scegliere se inserire un nuovo modulo o aggiornarne uno vecchio. Nel caso bisogna inserirne uno nuovo occorre digitare il nome, invece per aggiornarne uno vecchio compare una select con i nomi di tutti i moduli presenti nel progetto, dalla quale bisogna selezionare quello che si desidera aggiornare.
Quando si inserisce un nuovo modulo se ne diventa automaticamente proprietari, invece per aggiornarne uno vecchio bisogna esserne già proprietari, altrimenti viene restituito un messaggio di errore.

Dopo aver cliccato sul pulsante **Submit** o **Save** il procedimento diventa perfetta-

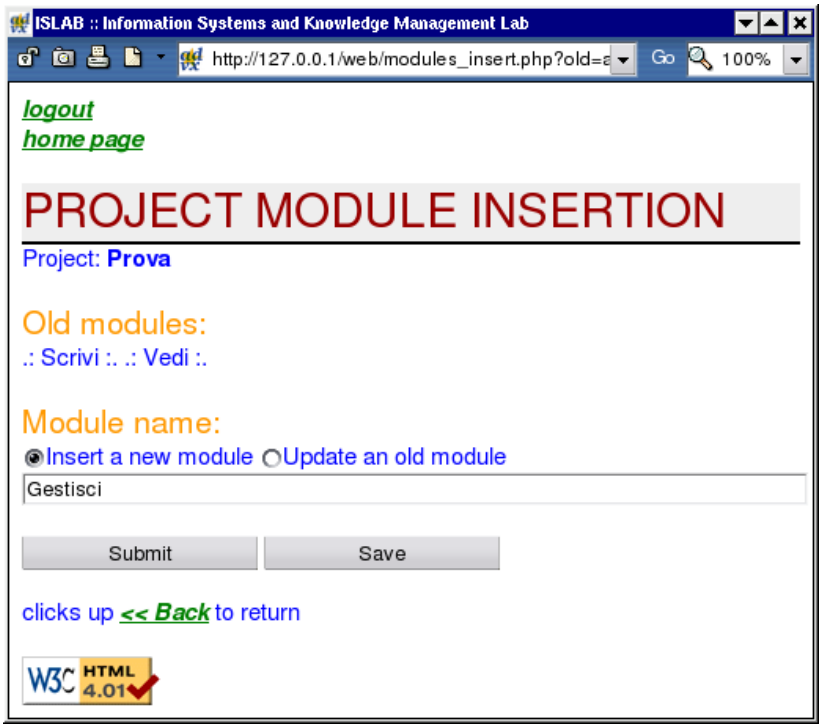

Figura 4.21: *Pagina dove inserire i nomi dei moduli del progetto da aggiornare (qui si è scelto di inserire un nuovo modulo)*

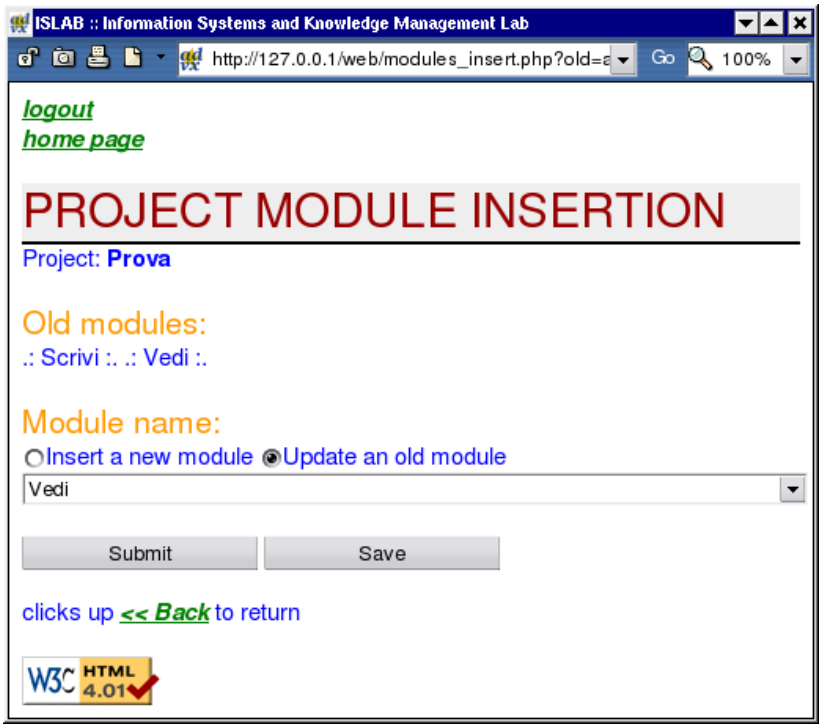

Figura 4.22: *Pagina dove inserire i nomi dei moduli del progetto da aggiornare (qui si è scelto di aggiornare un vecchio modulo)*

mente identico a quello della creazione di un progetto. L'unica differenza è nell'inserire il nome di un modulo, perché bisogna sempre scegliere se scriverne uno nuovo o selezionarne uno vecchio dall'elenco di quelli già inseriti.

4.7 Cancellazione progetto

Per cancellare un progetto bisogna cliccare sul link **Delete**. Per eseguire questa operazione bisogna essere o amministratori del progetto o amministratori del sito, altrimenti viene restituito un messaggio di errore.
Aprendo la pagina viene chiesto se si è sicuri di voler cancellare il progetto, in caso la risposta sia affermativa viene cancellato completamente sia dal database che dal filesystem.

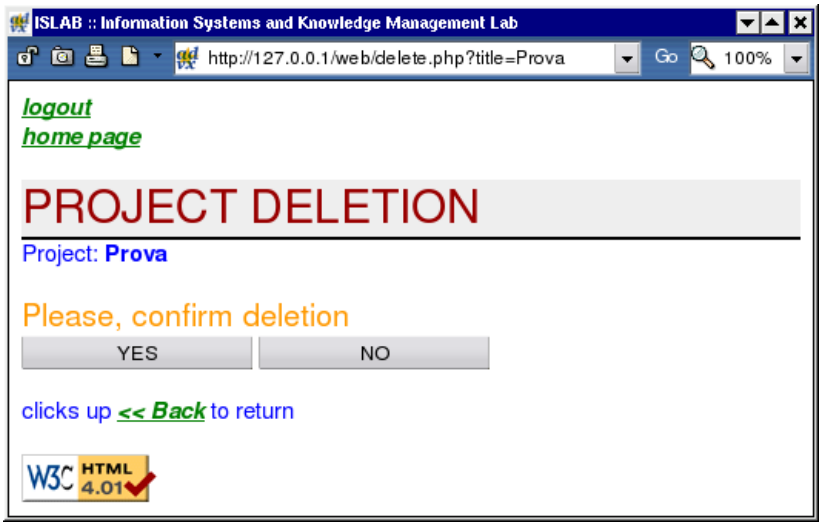

Figura 4.23: *Pagina dove viene chiesto se cancellare o no un progetto*

4.8 Rinomina progetto

Per rinominare un progetto bisogna cliccare sul link **Rename**. Per eseguire questa operazione bisogna essere o amministratori del progetto o amministratori del sito, altrimenti viene restituito un messaggio di errore.

Aprendo la pagina viene visualizzato il nome, la piattaforma e la descrizione del progetto. Basta cambiare quello che si desidera e cliccare sul pulsante **Submit** per rinominare quello che è stato cambiato. Nel caso in cui venga rinominato il nome del progetto anche i file presenti nella directory groups del progetto verranno rinominati di conseguenza.

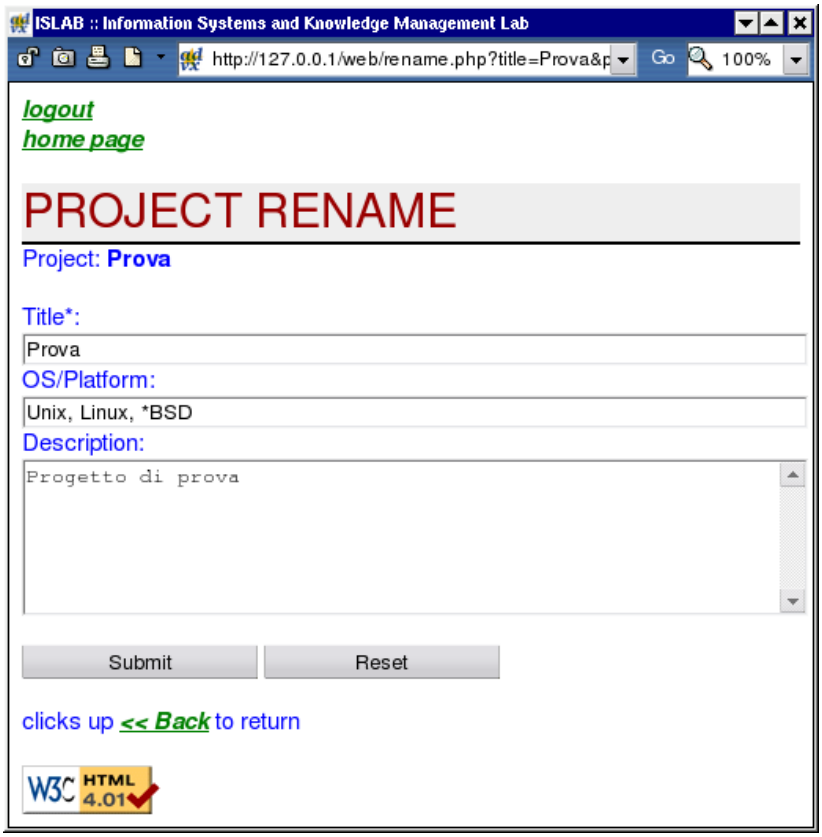

Figura 4.24: *Pagina dove rinominare il progetto*

4.9 Privilegi di progetto

Un amministratore del sito o l'amministratore di un progetto può decidere se fornire o revocare ad altri utenti i privilegi del progetto.Cliccando sul link **Privilege**, presente nella home page, si accede ad una pagina dove è presente una tabella con tutti gli utenti del sito.Nella prima colonna della tabella (**users**) è presente il nome, cognome

e login dell'utente, nella seconda (**e-mail**) viene indicata l'e-mail dell'utente.
Per cambiare i privilegi ad un utente è sufficiente cliccare sul nome prescelto.
Dopo aver deciso a che utente cambiare i privilegi, si accede ad una pagina dove sono

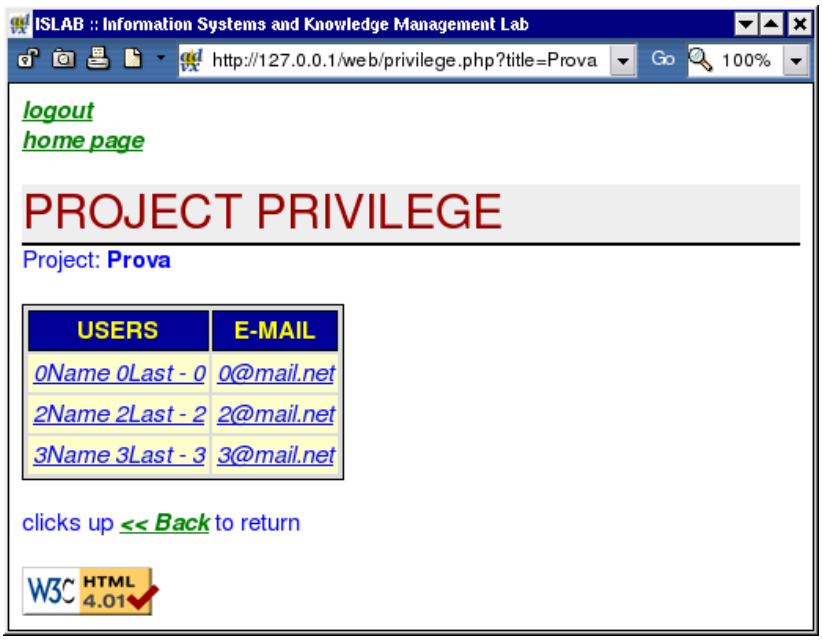

Figura 4.25: *Lista di utenti a cui poter cambiare i privilegi*

presenti due tabelle. Una indica i privilegi di progetto dell'utente, l'altra i privilegi di ogni singolo modulo di quel progetto. Se è presente **NO** significa che l'utente non possiede privilegi, ma cliccando su di esso l'utente li acquista ed il link diviene **YES**. Per revocare i privilegi basta fare l'operazione inversa.

Figura 4.26: *Privilegi dell'utente* 2 *sul progetto* Prova *ed i relativi moduli*

Conclusioni e sviluppi futuri

L'applicazione "Project Management" è stata studiata per girare in ambiente Linux ed utilizzare PostgreSQL come database. Nel caso si volesse cambiare piattaforma bisogna modificare alcune funzioni che interagiscono con il filesystem, perché sono stati utilizzati dei comandi Linux in modo da rendere più semplice ed immediata l'operazione. Se si volesse cambiare il database è necessario che quello nuovo gestisca l'integrità referenziale e le query innestate, altrimenti si rende indispensabile modificare i metodi delle classi che contengono le stringe delle istruzioni SQL.
Attualmente l'applicazione serve per gestire i vari progetti di tesi del Laboratorio di Sistemi Informativi ed è disponibile solo all'interno di esso.
In futuro, data la struttura molto versatile, il sito potrebbe essere utilizzato per poter gestire qualsiasi progetto software e reso accessibile anche dalla rete pubblica ad un numero sempre maggiore di utenti.
In caso di un gran numero di progetti l'approvazione di essi da parte dei soli amministratori del sito diverrebbe gravosa, in questo caso sarebbe consigliabile che tale responsabilità sia delegata anche agli amministratori dei progetti. Il cambiamento non è particolarmente gravoso, per farlo è sufficiente modificare alcune istruzioni di controllo dei privilegi di accesso.
Con il futuro per migliorare la comunicazione dei vari utenti può essere opportuno la creazione di un forum con diverse sezioni, alcune comuni, altre dedicate specificatamente allo sviluppo dei vari progetti presenti.

Bibliografia

[1] P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone. *Basi di dati.* McGraw–Hill, second edition, Sept. 1999.

[2] J. D. Bolter and R. Grusin. *Remediation.* Guerini e Associati, 2002.

[3] W. Choi, A. Kent, C. Lea, G. Prasad, and C. Ullman. *PHP 4 Guida per lo sviluppatore.* Hoepli, 2002.

[4] J. Greenspsan and B. Bulger. *Sviluppare applicazioni per database con MySQL/PHP.* Apogeo, July 2001.

[5] D. Tansley. *Pagine Web dinamiche con PHP e MySQL.* Addison–Wesley, first edition, Nov. 2002.

[6] F. Tornieri. *Linux: configuralo al meglio!* Master, June 2003.

[7] P. Wilton. *JavaScript Guida per lo sviluppatore.* Hoepli, 2002.

[8] J. C. Worsley and J. D. Drake. *Manuale pratico di PostgreSQL.* Hops, 2002.