Gestione biblioteca

Sommario

[1.1 Informazioni sul progetto 3](#_Toc91514419)

[1.2 Abstract 3](#_Toc91514420)

[1.3 Scopo 3](#_Toc91514421)

[2 Analisi 5](#_Toc91514422)

[2.1 Analisi del dominio 5](#_Toc91514423)

[2.2 Analisi e specifica dei requisiti 5](#_Toc91514424)

[Use case 9](#_Toc91514425)

[2.3 Pianificazione 10](#_Toc91514426)

[2.4 Analisi dei mezzi 11](#_Toc91514427)

[2.4.1 Software 11](#_Toc91514428)

[2.4.2 Hardware 11](#_Toc91514429)

[3 Progettazione 12](#_Toc91514430)

[3.1 Design dell’architettura del sistema 12](#_Toc91514431)

[3.2 Design dei dati e database 13](#_Toc91514432)

[3.3 Design delle interfacce 15](#_Toc91514433)

[4 Implementazione 17](#_Toc91514434)

[4.1 Database 17](#_Toc91514435)

[4.1.1 Inserimento utente bibliotecario 19](#_Toc91514436)

[4.2 Classi controller 19](#_Toc91514437)

[4.2.1 Index 20](#_Toc91514438)

[4.2.2 Home 20](#_Toc91514439)

[4.2.3 Librarianhome 21](#_Toc91514440)

[4.2.4 Myerror 23](#_Toc91514441)

[4.2.5 Login 23](#_Toc91514442)

[4.2.6 Registration 24](#_Toc91514443)

[4.3 Classi Model 24](#_Toc91514444)

[4.3.1 Home\_book\_model 25](#_Toc91514445)

[4.3.2 Home\_rating\_model 26](#_Toc91514446)

[4.3.3 Home\_rent\_model 27](#_Toc91514447)

[4.3.4 Librarian\_management\_model 28](#_Toc91514448)

[4.3.5 Login\_model 30](#_Toc91514449)

[4.3.6 Registration\_model 31](#_Toc91514450)

[4.4 Libs 32](#_Toc91514451)

[4.4.1 View 32](#_Toc91514452)

[4.4.2 Controller 33](#_Toc91514453)

[4.4.3 Model 33](#_Toc91514454)

[4.5 Connect 33](#_Toc91514455)

[4.6 Control 34](#_Toc91514456)

[5 Test 35](#_Toc91514457)

[5.1 Protocollo di test 35](#_Toc91514458)

[5.2 Risultati test 38](#_Toc91514459)

[5.3 Mancanze/limitazioni conosciute 42](#_Toc91514460)

[6 Consuntivo 43](#_Toc91514461)

[7 Conclusioni 44](#_Toc91514462)

[7.1 Sviluppi futuri 44](#_Toc91514463)

[7.2 Considerazioni personali 44](#_Toc91514464)

[8 Bibliografia 44](#_Toc91514465)

[8.1 Sitografia 44](#_Toc91514466)

[9 Allegati 44](#_Toc91514467)

Introduzione

## Informazioni sul progetto

Candidato: Damian Campesi

Azienda: SAM Trevano

Periodo: 16.09.2021-23.12.2021

Presentazione: gennaio 2022

## Abstract

**Situazione iniziale**

*Lo scopo del progetto è quello di creare un applicativo web che permetta la gestione dei noleggi di una libreria. Ora una persona se vuole noleggiare un libro, deve recarsi fisicamente alla libreria e controllare che il libro che desidera sia disponibile. Questo progetto permetterebbe di sapere in modo più comodo se il libro che si cerca è disponibile oppure è già stato noleggiato.*

*In più ha altre funzionalità:*

* *Dopo il noleggio di un libro, è possibile fare una valutazione su di esso, che è visibile a tutti gli utenti connessi al sito.*
* *Una volta scaduto il termine di consegna, l’utente che ha noleggiato il libro riceverà una e-mail che gli ricorda di riconsegnare il libro.*
* *Potranno essere visualizzati i primi 10 libri con valutazione migliore.*
* *Questo applicativo può essere utilizzato su ogni browser.*

**Attuazione**

Per questo progetto utilizzo HTML, CSS e PHP. Il database che utilizzo è un database MySQL.

Per prima cosa, implemento le pagine web e le rendo user friendly e responsive, in modo che possano essere adatte a ogni dispositivo e possano essere pure semplici ed efficaci nel loro utilizzo. In un secondo momento collego il database all’interfaccia web in modo che i dati come libri, valutazioni e noleggi contenuti nel database possano essere visualizzati. Per l’invio della e-mail creo un programma in Python che mi permette di controllare le date di fine noleggio presenti nel database, per poi inviare le informazioni agli utenti desiderati (a meno di una settimana dalla fine del noleggio).

**Risultati**

## Scopo

Lo scopo di questo progetto è quello di rendere più semplice ed efficace noleggiare un libro e per i bibliotecari la gestione dei noleggi. Con questo applicativo si potrà noleggiare un libro anche se non si è fisicamente nella biblioteca ed oltre a questo, si potrà verificare la disponibilità di un libro in ogni momento.

In più in questo applicativo viene implementata la possibilità di inserire delle valutazioni, che permettono agli utenti di scegliere i libri con valutazioni alte rispetto a libri che non sono molto apprezzati da altri utenti.

# Analisi

## Analisi del dominio

Per questo progetto mi è stato chiesto di realizzare un applicativo web per la gestione di una biblioteca. Ora per poter noleggiare un libro da una biblioteca bisogna recarsi fisicamente sul posto e bisogna verificare che il libro che si sta cercando sia disponibile. Con l’applicativo web che mi viene chiesto di sviluppare, si potrà verificare in qualsiasi momento e in qualsiasi posto la disponibilità dei libri di una biblioteca. Tutti gli utenti registrati potranno vedere il catalogo dei libri, i libri più apprezzati e potranno noleggiare i libri che più gli interessano. Ci sarà poi il bibliotecario che potrà gestire il sito web, modificando il catalogo dei libri, quindi aggiungendo e togliendo libri, eliminando i noleggi scaduti o aggiungendo le valutazioni che i clienti danno ai libri noleggiati.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Installazione WebServer e DBMS |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano permessi di amministratore |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Creazione struttura DB |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Il DB deve contenere le tabelle per registrare libri, noleggi, valutazioni e utenti |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Creazione pagina Registrazione e LogIn |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Nuovi utenti possono registrarsi |
| **002** | Utenti già registrati possono effettuare l’accesso |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Creazione pagina WEB principale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Pagina WEB per noleggiare libri |
| **Sotto requisiti** | |
| **001** | La pagina web deve essere responsive |
| **002** | La pagina web deve essere utilizzabile con facilità |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Collegamento Pagine WEB al DB |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dalla pagina web si possono visualizzare i libri presenti nel DB |
| **Sotto requisiti** | |
| **001** | Solo utenti presenti nel DB possono accedere |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Gestione Libri |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti possono utilizzare dei libri |
| **Sotto requisiti** | |
| **001** | Si devono avere filtri di ricerca per nome e autore |

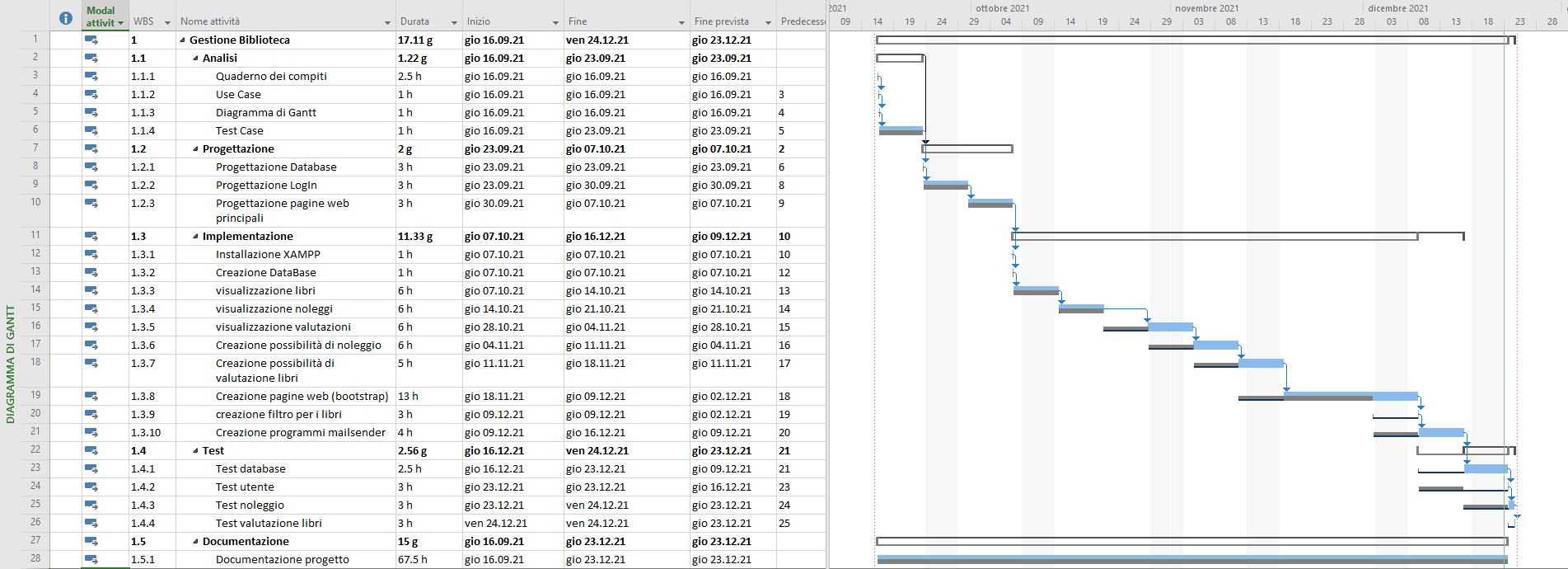
|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Gestione noleggi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti possono noleggiare i libri |
| **Sotto requisiti** | |
| **001** | Il tempo di noleggio dei libri può essere determinato |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Gestione avvisi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Avviso tramite e-mail quando un noleggio sta pe terminare. |
| **Sotto requisiti** | |
| **001** | Sulla e-mail ci deve essere la data di invio |
| **002** | Sulla e-mail ci deve essere la data di riconsegna del libro noleggiato |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Gestione valutazioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti possono visualizzare la valutazione dei libri sul sito WEB |
| **Sotto requisiti** | |
| **001** | La valutazione deve essere da zero a cinque stelle |

## Use case

## Pianificazione

Nel diagramma di Gantt si può notare in nero la pianificazione iniziale (versione preventiva).

## Analisi dei mezzi

### Software

Questo progetto è stato sviluppato su un PC con sistema operativo Windows 10. Per lo sviluppo ho utilizzato XAMPP versione 3.3.0. XAMPP contiene PHP versione 8.0.2 e MySQL versione 15.1. Come ambiente di sviluppo ho utilizzato VS Code versione 1.58.2 e MySQL Workbech 8.0.26. La versione di Python che ho utilizzato è la 3.10.1, invece per HTML è HTML5 e per CSS, CSS3.

Per la progettazione ho utilizzato Microsoft Visio, Project e Micrsoft Word.

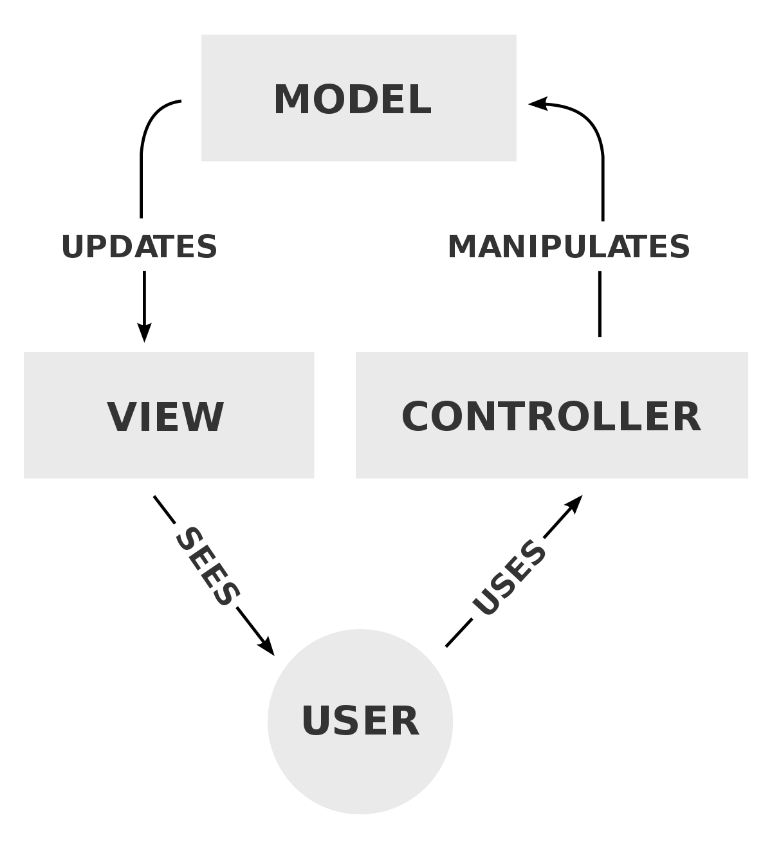
### Hardware

L’hardware utilizzato durante lo sviluppo del progetto è un normale PC da ufficio. Le specifiche sono queste:

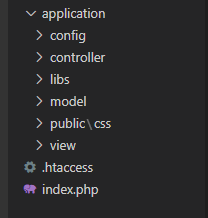
* CPU Intel Xeon E3-1240 v5
* 16GB RAM
* GPU Nvidia quadro K620

# Progettazione

## Design dell’architettura del sistema

Per questo progetto ho deciso di utilizzare la struttura MVC, ovvero Model, View e Controller. Qui sotto viene mostrato graficamente il funzionamento della struttura:

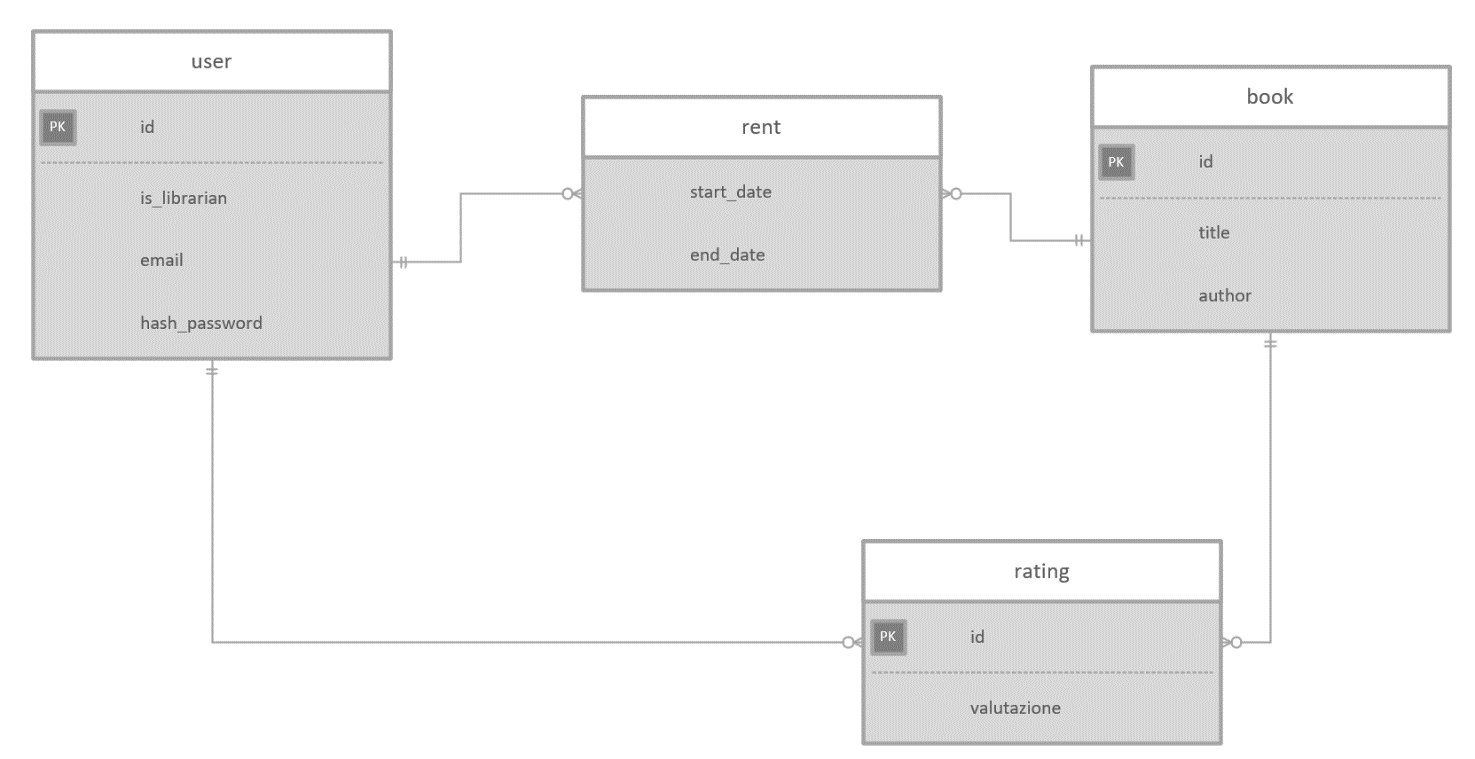
I controllers sono le classi con cui gli utenti comunicano direttamente. Questi controller si occupano di gestire tutte le azioni dell’utente. Questo lo fanno grazie all’aiuto delle classi models, in queste classi vengono creati tutte le funzioni con le operazioni necessarie per ogni funzionalità. Le views invece contengono tutta la parte grafica (principalmente codice html) che permette la visualizzazione grafica dell’applicativo. Il template MVC che ho utilizzato è quello messomi a disposizione dal docente Massimo Sartori:



* All’interno della cartella “application” è inserito tutto l’applicativo. All’esterno invece ci sono i file “. htdocs” che serve per reindirizzare tutte le richieste, mentre il file “index.php” serve per far partire l’intero applicativo.
* La cartella “config” contiene il file con il codice per la connessione con il database, e in aggiunta anche varie costanti che mi sono utili in tutti gli altri files.
* Nella cartella “controller” invece ci sono tutti quei files che contengono classi che svolgono appunto il compito da “controller”, che servono a gestire tutte le funzioni.
* Invece nella cartella “libs” ho inserito le classi “View”,” Controller” e “Model”. Contengono lo scheletro delle classi della loro categoria, quindi la classe “View” ha il metodo “render()”, mentre la classe “Controller” contiene gli attributi che fanno riferimento a un model e a una view.
* Nella cartella “model” ho inserito le specializzazioni della classe “Model” presente nella cartella “libs”. In questa cartella ogni model esegue una funzione, dal login alla registrazione, noleggio, modifiche ecc…
* Nella cartella “public” si trovano i file per la formattazione CSS.
* Nella cartella “view” si trovano i file per la visualizzazione delle pagine web (HTML).

Per il programma che si occupa di inviare le email in caso di imminente scadenza di un noleggio ho creato un programma in Python, chiamato “mailsender.py”.

## Design dei dati e database

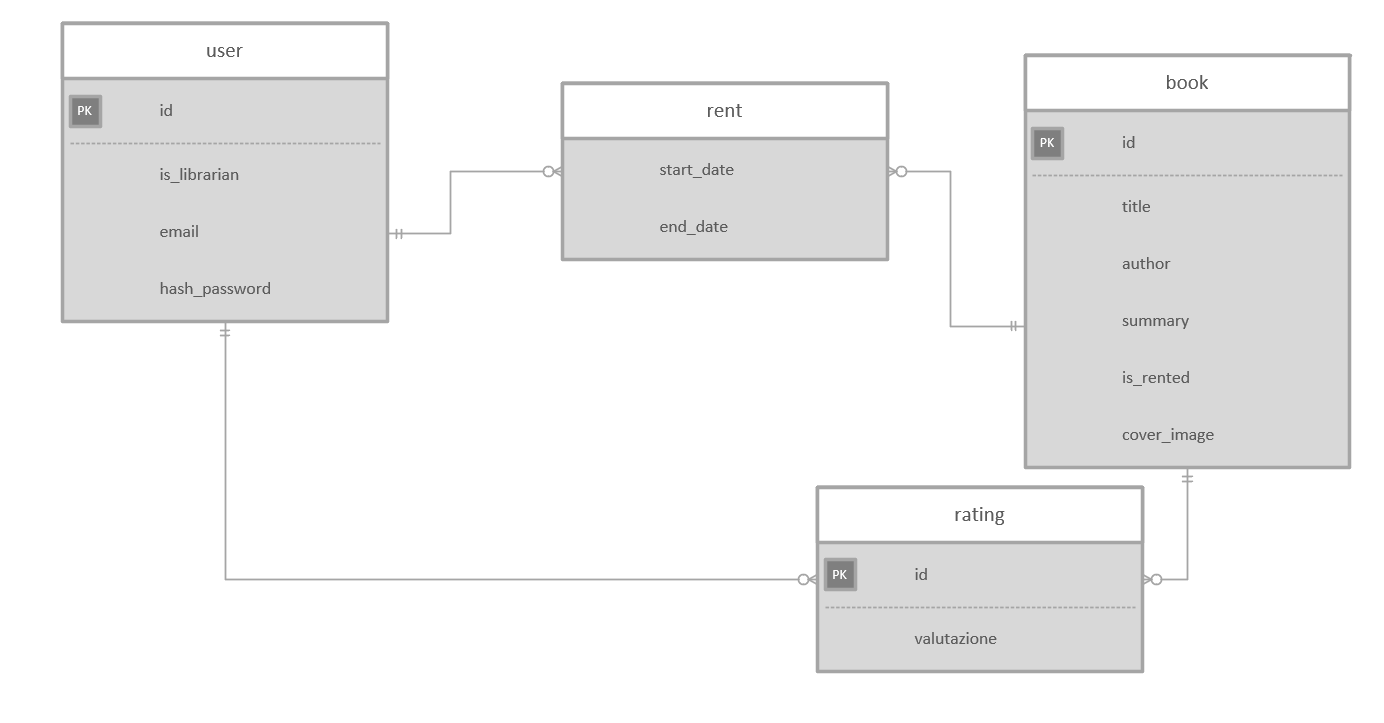


Questa è la prima versione di database che ho utilizzato per il progetto. Il database che ho creato contiene quattro tabelle: la tabella user contiene le informazioni sugli utenti che sono registrati nel database. Un utente è identificato dal suo id che è univoco, e per accedere al profilo si devono utilizzare e-mail e password. La password è salvata in “hash\_password” ed è ovviamente cifrata.

La tabella “rent” permette invece di memorizzare i noleggi che gli utenti fanno inserendo la data di inizio e la data di fine. Questa tabella contiene delle chiavi esterne, una che è “user\_id” e l’altra che è “book\_id”. Ogni noleggio viene identificato in base alle due chiavi esterne, in modo che un utente possa noleggiare solo una volta lo stesso libro nello stesso momento.

In “book” invece vengono memorizzati tutti i libri del catalogo di una ipotetica biblioteca. Anche i libri come gli utenti sono identificati con un id univoco. Ogni libro ha un proprio campo per inserire: titolo e autore.

Nella tabella “rating” vengono salvati tutte le valutazioni sui libri, ed ognuna di esse è sempre identificata da un id.

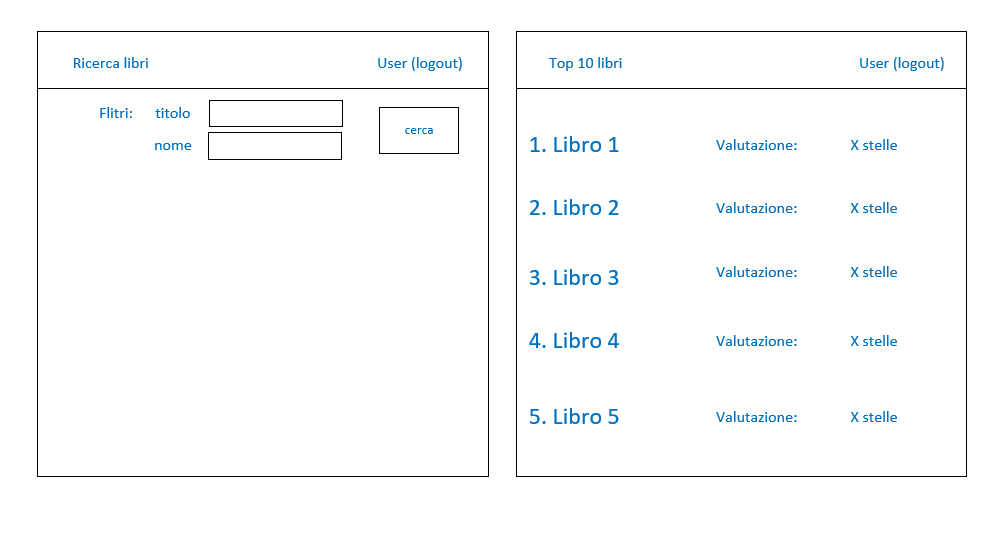


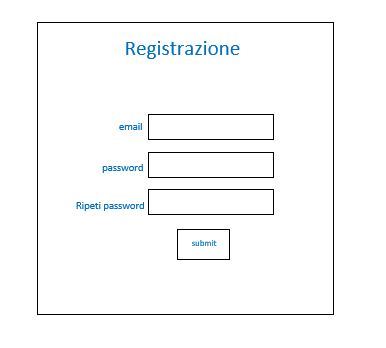
Questa immagine rappresenta invece la versione finale del database. Ho aggiunto alcuni attributi in alcune tabelle.

Nella tabella user ho inserito un attributo boolean di nome “is\_librarian”. Questo mi permette di poter verificare se un utente è un utente normale oppure ha il ruolo da bibliotecario, che gli permette di effettuare modifiche dall’applicativo web.

In book ho invece inserito degli attributi che mi servono per dare una migliore descrizione dei libri: in “summary” si inserisce la trama di un libro e infine in “cover\_image” viene salvato il percorso dell’immagine di copertina. L’attributo “is\_rented” permettere di sapere se il libro in questione è noleggiato in uno specifico momento.

## Design delle interfacce



Le interfacce mostrate sopra sono una prima progettazione delle pagine che poi ho sviluppato. Ci sono le pagine di login e di registrazione, che permettono appunto la creazione di un utente oppure l’accesso al sito. Le pagine interne al sito le ho pensate con un menu sempre visibile sulla parte alta della pagina, in modo da poter navigare più facilmente.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

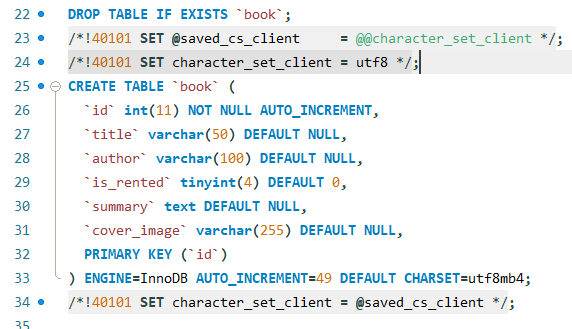
In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

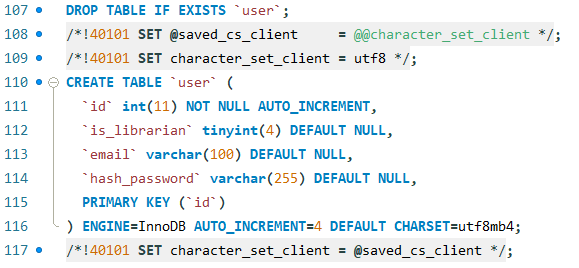
Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

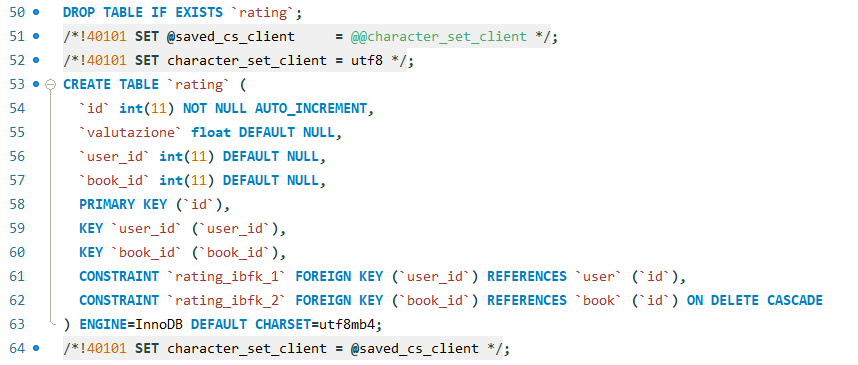
## Database

La creazione del database è stata effettuata con “CREATE DATABASE gestione\_biblioteca;”, seguito da “use gestione\_biblioteca;”. Dopodiché ho creato le tabelle.

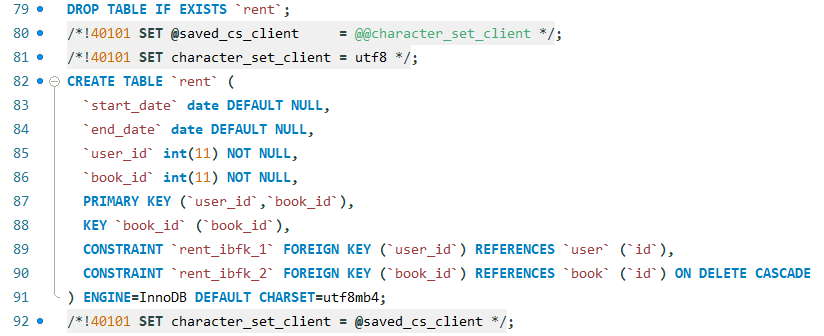
In questa immagine viene visualizzata la creazione della tabella book in SQL.



Questa invece è la creazione della tabella user.



La seguente immagine contiene la creazione della tabella rating.



Questa immagine contiene la creazione della rent. Si possono notare le creazioni delle chiavi esterne (foreign key), che servono per avere un riferimento ad un utente e ad un libro.

### Inserimento utente bibliotecario

Con questo comando ho inserito un utente che ha il ruolo da bibliotecario, in modo da poter testare tutte le funzioni disponibili.

## Classi controller

Nella cartella mostrata sopra sono presenti tutti i files che contengono le classi controller che ho utilizzato.

### Immagine che contiene testo Descrizione generata automaticamenteIndex

Questo controller viene utilizzato quando nell’URL non è specificato il controller da utilizzare. Il metodo index visualizza il login o la pagina home in base a se si è autenticati o no.

### Home

La classe Home contiene tutte le funzioni che servono per la visualizzazione dei libri, dei noleggi e delle valutazioni.

Immagine che contiene testo

Descrizione generata automaticamenteIn più si occupa anche del creare e registrare i noleggi grazie alla classe model “Home\_Rent\_Model”:

Se il noleggio avviene correttamente, viene mostrata una pagina di successo, al contrario se il noleggio non è andato a buon fine si renderizza una pagina di errore.

### Librarianhome

La classe librarianhome contiene tutte le funzionalità che servono all’utente bibliotecario per poter:

* Modificare, aggiungere e eliminare libri
* Eliminare, aggiungere valutazioni
* Eliminare noleggi

Tutte queste funzionalità sono implementate grazie al model “librarian\_management\_model”. C’è comunque la funzione viewBook che utilizza una funzione nel model “Home\_Book\_Model”.

Per esempio, qui sotto mostro le funzioni che utilizzo per gestire i libri nel catalogo:

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

* Funzione viewBooks:
  + Ritorna la lista di tutti i libri.
* Funzione viewBook:
  + Ritorna un libro specifico.
* Funzione editBook:
  + Utilizzando la funzione editBook del model permette la modifica dei libri.
* Funzione removeBook:
  + Permette la rimozione di un libro dal catalogo.
* Funzione addBook:
  + permette l’aggiunta di un nuovo libro nel catalogo.
* Funzione viewForm:
  + Permette la visualizzazione del form di aggiunta di un nuovo libro o per la modifica di un libro.

### Immagine che contiene testo Descrizione generata automaticamenteMyerror

Utilizzo questa classe per visualizzare un messaggio di errore in caso il controller specificato nell’URL non esista. La funzione “index” permette appunto la renderizzazione della pagina di errore.

### Immagine che contiene testo Descrizione generata automaticamenteLogin

Questo controller si occupa della gestione del login. La funzione “login”, infatti, utilizza il model “Login\_Model” per effettuare il login. In base all’esito si renderizza la home page oppure di nuovo il login.

### Immagine che contiene testo Descrizione generata automaticamenteRegistration

Il controller registration si occupa della registrazione degli utenti. Se la registrazione avviene con successo viene renderizzata la pagina per poter fare il login. Se la registrazione non avviene, si renderizza di nuovo la pagina per la registrazione.

## Classi Model

Immagine che contiene testo

Descrizione generata automaticamente

### Home\_book\_model

Questo model si occupa di tutto ciò che riguarda i libri, più nello specifico di:

* Visualizzazione dei libri del catalogo, con o senza filtri.
* Visualizzazione di un singolo libro che si vuole noleggiare.

Per queste funzionalità ho creato dei metodi, viewBooks e viewBook:

Immagine che contiene testo

Descrizione generata automaticamente

Questa funzione fa delle query al database, in base ai filtri. Se non si sono filtri da applicare, ritorna tutti i libri nel catalogo che non sono noleggiati.

Immagine che contiene testo, schermo, screenshot

Descrizione generata automaticamente

Questa classe invece ritorna tutte le informazioni sullo specifico libro che si vuole noleggiare, che poi saranno visualizzate nella pagina grazie ad una view.

### Immagine che contiene testo Descrizione generata automaticamenteHome\_rating\_model

Questo model ritorna tutte le valutazioni dei libri. La query che utilizzo è abbastanza complessa:

SELECT book.title, book.cover\_image, ROUND(AVG(rating.valutazione),1) as media from rating

INNER JOIN book ON book.id = rating.book\_id

GROUP BY book.id

ORDER BY ROUND(AVG(rating.valutazione),1) DESC;

Per prima cosa, arrotondo il valore “valutazione”, perché in MySQL il tipo FLOAT è impreciso; quindi, così evito alcuni problemi di visualizzazione. Per avere il nome dei libri faccio una INNER JOIN con la tabella “book” e infine raggruppo tutte le valutazioni per l’id di ogni libro e ordino i risultati dal migliore al peggiore.

### Immagine che contiene testo Descrizione generata automaticamenteHome\_rent\_model

Questo model si occupa dei noleggi. Sia della visualizzazione dei noleggi di ogni utente ma anche della creazione dei noleggi.

Quando si noleggia un libro parte la funzione “doRent”. Questa funzione, se il valore di giorni per cui si vuole noleggiare è valido, prende la data di quel momento, che è la data di inizio noleggio e poi calcola la data di fine noleggio. Dopo aver fatto tutto questo inserisce tutto nel database. Infine, effettua un controllo per vedere se il noleggio è stato inserito nel DB correttamente e ne ritorna l’esito.

### Librarian\_management\_model

Immagine che contiene testo

Descrizione generata automaticamenteQuesto model contiene tutte le funzioni che servono per poter modificare, eliminare, aggiungere libri al catalogo, eliminare noleggi oppure aggiungere o eliminare valutazioni. Le funzioni in questo model sono simili tra loro, ma per ogni funzione la query da fare è differente.

Questo metodo serve per poter modificare le informazioni sui libri, come autore, titolo, trama o copertina.

L’inserimento delle nuove informazioni avviene tramite form. Tramite POST il metodo riceve le modifiche che poi applica anche nel DB con un UPDATE. Per il file della copertina utilizzo “$\_FILES” che mi permette di ricevere tramite POST anche delle immagini. Con “move\_uploaded\_file” salvo nella cartella “images” le immagini in modo che possano essere raggiungibili. Come ultima cosa controllo che nel database sia stata inserita la modifica e ritorno l’esito.

Immagine che contiene testo, interni, schermo

Descrizione generata automaticamenteLa stessa cosa la faccio con la funzione “addBook”, però in quella funzione faccio un INSERT nel database e non un UPDATE.

Per l’eliminazione dei libri invece faccio un DELETE nel database, e poi controllo che il libro si stato eliminato. Stessa cosa per la rimozione dei noleggi e delle valutazioni.

Immagine che contiene testo

Descrizione generata automaticamentePer l’inserimento delle valutazioni ho bisogno di poter scegliere l’utente che ha dato la valutazione. La funzione qui sotto ritorna un array con all’interno l’e-mail e l’id di ogni utente, in modo da poter quindi assegnare una valutazione ad una persona:

### Immagine che contiene testo Descrizione generata automaticamenteLogin\_model

Questo model permette di effettuare il login agli utenti. Per prima cosa si fanno dei controlli sui dati inseriti nel form di login, poi viene effettuata una query che seleziona l’utente in base alla e-mail, per verificare che la password criptata corrisponda. Se il login non ha successo, la sessione viene distrutta e viene mostarato un messaggio d’errore.

### Immagine che contiene testo Descrizione generata automaticamenteRegistration\_model

Questo model permette di effettuare la registrazione degli utenti. Per prima cosa si fanno dei controlli sui dati inseriti nel form, poi viene effettuata una query che verifica che non ci sia un utente già registrato con la e-mail inserita nel form. Infine, se tutto va a buon fine si cripta la password inserita nel form e si inserisce il nuovo utente nel database. Se la registrazione non ha successo, viene mostrato un messaggio d’errore in base al problema.

Ovviamente c’è sempre il controllo che verifica che l’utente sia stato effettivamente inserito nel DB. Questa funzione ritorna quindi l’esito della registrazione.

## Libs

### Immagine che contiene testo Descrizione generata automaticamenteView

Questa classe mi permette di renderizzare le varie pagine HTML in base agli utenti:

* se l’utente non è loggato allora renderizzo la pagina con header e footer vuoti
* se è loggato renderizzo la pagina con header e footer che utilizzano gli utenti
* se l’utente è loggato ed è un bibliotecario allora renderizzo il footer e l’header con il menu le funzionalità di amministrazione aggiunte.

### Controller

Immagine che contiene testo

Descrizione generata automaticamenteQui sotto ho messo un’immagine della classe Controller che utilizzo come sottoclasse di tutti i miei controller.

### Model

Qui sotto invece ho inserito la sottoclasse Model che uso per tutti i model. Ho inserito sono l’attributo “msg” che mi server per i messaggi di errore

Immagine che contiene testo

Descrizione generata automaticamente

## Connect

In questo file creo la connessione con il database:

Immagine che contiene testo

Descrizione generata automaticamente

Ho impostato in delle variabili tutte le credenziali d’accesso, il database e l’host che mi servono. In caso di errore la connessione viene interrotta e viene sollevato un errore.

## Immagine che contiene testo, schermo, screenshot Descrizione generata automaticamenteControl

In questo file ho inserito delle funzioni che mi sono utili in tutte le altre classi, per poter validare le e-mail e validare le stringhe.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Controllo funzionamento XAMPP |
| **Descrizione:** | Verificare che il programma XAMPP funzioni correttamente | | |
| **Prerequisiti:** | XAMPP deve essere installato. | | |
| **Procedura:** | 1. Avviare XAMPP control panel. 2. Premere start dei servizi apache e mysql. 3. Aspettare che i nomi dei due servizi diventino evidenziati con il colore verde. | | |
| **Risultati attesi:** | Apache e MySql funzionano e sono operativi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Controllo funzionamento DB |
| **Descrizione:** | Verificare che il DB funzioni correttamente | | |
| **Prerequisiti:** | Il DB è già stato creato. | | |
| **Procedura:** | 1. Inserire in ogni tabella del DB dei dati di prova, con il comando INSERT. 2. Dopo aver inserito i dati, verificare con una SELECT che i dati siano presenti nel DB e siano registrati correttamente. 3. Rimuovere i dati inseriti con un comando DELETE e poi riverificare la rimozione con un comando SELECT. | | |
| **Risultati attesi:** | Il database è in grado di salvare I dati che gli sono stati passati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Controllo funzionamento LogIn |
| **Descrizione:** | Verificare che gli utenti possano inserire I propri dati | | |
| **Prerequisiti:** | L’interfaccia di LogIn è già stata creata. | | |
| **Procedura:** | 1. Andare nella pagina di LogIn del sito. 2. Provare ad inserire una mail e password per la registrazione. 3. Verificare che i dati inseriti dall’utente siano salvati nella tabella user del DB in modo corretto. | | |
| **Risultati attesi:** | Si riesce ad effettuare l’accesso o la registrazione al sito. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Controllo funzionamento Pagina web principale |
| **Descrizione:** | Verificare che la pagina principale si adatta alle esigenze | | |
| **Prerequisiti:** | La pagina web è già stata creata. | | |
| **Procedura:** | 1. Aprire la pagina web principale. 2. Verificare che la pagina web si adatti a diverse grandezze della finestra. | | |
| **Risultati attesi:** | La pagina web si adatta a diverse grandezze di schermo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-005 | **Nome:** | Controllo collegamento DB a pagina WEB |
| **Descrizione:** | Verificare che si possano avere i dati presenti sul DB e poterli modificare in base alle esigenze | | |
| **Prerequisiti:** | Pagina web e DB sono funzionanti. | | |
| **Procedura:** | 1. Inserire manualmente dei libri all’interno del DB. 2. Utilizzare un profilo creato per verificare che i libri presenti nel DB si visualizzino sulla pagina web. 3. Ripetere questi passaggi con le valutazioni e i noleggi. | | |
| **Risultati attesi:** | I dati presenti nel DB si modificano in base ai dati inseriti via WEB dal bibliotecario | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Controllo funzionamento libri |
| **Descrizione:** | Verificare che i libri possano essere filtrati. | | |
| **Prerequisiti:** | Pagina web e DB sono funzionanti. | | |
| **Procedura:** | 1. Andare nella pagina per la ricerca dei libri ed inserire dei filtri per autore o titolo. 2. Verificare che i libri corrispondenti soddisfino i criteri dei filtri. | | |
| **Risultati attesi:** | Tutti I libri all’ interno del DB vengono visualizzati correttamente | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-007 | **Nome:** | Controllo funzionamento noleggi |
| **Descrizione:** | Verificare che I noleggi di libri avvengano correttamente | | |
| **Prerequisiti:** | Pagina web e DB sono funzionanti. | | |
| **Procedura:** | 1. Andare nella pagina web e noleggiare un libro presente. 2. Verificare che il noleggio sia registrato sul DB e che il libro risulti noleggiato. | | |
| **Risultati attesi:** | Il libro è stato noleggiato e non può essere noleggiato da altri. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-008 | **Nome:** | Controllo funzionamento avvisi |
| **Descrizione:** | Verificare che gli avvisi di scadenza noleggio avvengano correttamente | | |
| **Prerequisiti:** | Pagina web e DB sono funzionanti. | | |
| **Procedura:** | 1. Noleggiare un libro per 1 settimana. 2. Verificare che sia stata inviata una mail all’indirizzo e-mail corrispondente a chi ha noleggiato il libro. | | |
| **Risultati attesi:** | A una settimana dal termine di consegna l’utente riceve una e-mail di avviso. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-009 | **Nome:** | Controllo funzionamento valutazioni |
| **Descrizione:** | Verificare che il Sistema di valutazione per libri funzioni correttamente. | | |
| **Prerequisiti:** | Pagina web e DB sono funzionanti. | | |
| **Procedura:** | 1. Utilizzando l’utente del bibliotecario, si inseriscono alcune valutazioni a dei libri. 2. Una volta inseriti i valori verificare che i dati siano salvati sul DB in modo corretto e che siano visibili anche sul sito web da altri utenti. | | |
| **Risultati attesi:** | La valutazione viene inserita nel DB e viene visualizzata da altri utenti. | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Esito | Risultati | Data |
| TC-001 | OK |  | 27.12.2021 |
| TC-002 | OK |  | 27.12.2021 |
| TC-003 | OK |  | 27.12.2021 |
| TC-004 | OK |  | 27.12.2021 |
| TC-005 | OK |  | 27.12.2021 |
| TC-006 | OK |  | 27.12.2021 |
| TC-007 | OK |  | 27.12.2021 |
| TC-008 | OK |  | 27.12.2021 |
| TC-009 | OK |  | 27.12.2021 |

## Mancanze/limitazioni conosciute

Durante questo progetto non sono riuscito ad implementare l’invio della e-mail alla registrazione. Questo per motivi di vari ritardi sul piano di lavoro su alcuni imprevisti.

# Consuntivo

Confrontando le differenze posso dire che ci sono stati parecchi ritardi e alcuni imprevisti che non mi aspettavo. Questi hanno impattato molto sulla mia fase di implementazione, tanto da non essere riuscito ad inserire delle funzionalità richieste.

# Conclusioni

## Sviluppi futuri

Una aggiunta interessante per questo progetto sarebbe quella di aumentare l’efficacia dei filtri, in modo da poter filtrare i libri per la loro valutazione, oppure si aggiungere un attributo ai libri in modo da poter filtrare questi in base a delle categorie.

## Considerazioni personali

In questo progetto ho migliorato sicuramente la mia conoscenza sul linguaggio PHP, ho imparato ad utilizzare Bootstrap, e ho imparato qualcosa anche sul linguaggio Python. Questa esperienza mi sarà sicuramente utile per progetti futuri, in cui cercherò di risolvere tutte quelle imperfezioni che ho avuto.

# Bibliografia

## Sitografia

* <https://stackoverflow.com/>, 27.12.2021
* <https://www.php.net/>, 27.12.2021
* <https://www.w3schools.com/>, 27.12.2021
* <https://moodle.edu.ti.ch/cpt/>, 27.12.201

# Allegati

* Diari di lavoro
* Diagramma di Gantt
* Design del database
* Quaderno dei compiti
* Codici in GitHub