



7 FEBBRAIO 2020

RELAZIONE D'ESAME TECNOLOGIE WEB
MFN0634

VERONICA DE SANTIS
UNIVERSITÀ DEGLI STUDI DI TORINO, A.A. 2019-2020
Matricola: 870166

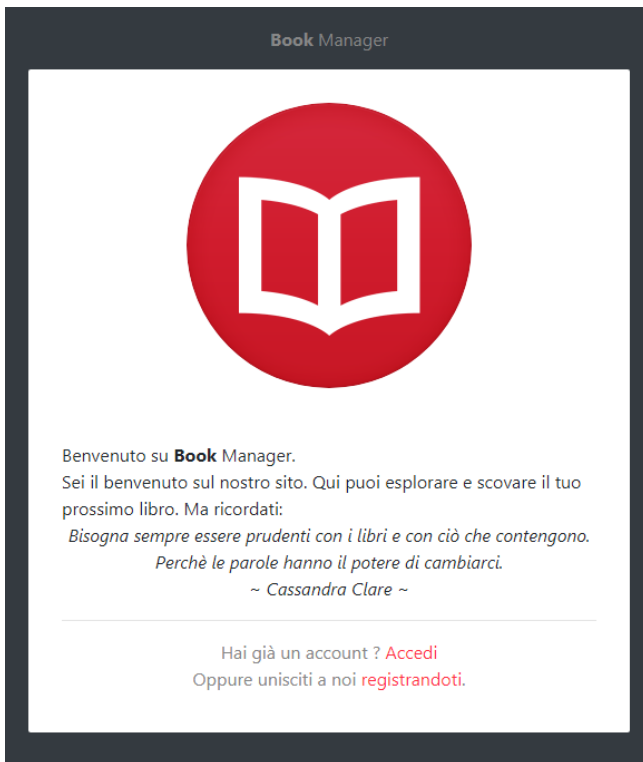
Sommario

Tema del sito	2
Sezioni principali	2
Front-end.....	3
Separazione presentazione/contenuto/comportamento.....	3
Soluzioni cross-platform.....	3
Organizzazione file e cartelle di progetto	4
Back-end	4
Architettura generale classi/funzioni	4
Schema del db	5
Descrizione delle funzioni remote.....	5
Funzionalità.....	6
Login/logout.....	6
Registrazione	7
Gestione del contenuto generato dall'utente	8
Caratteristiche.....	9
Usabilità	9
Interazione / animazione	11
Sessioni.....	14
Interrogazione del database	15
Validazione dati input.....	15
Sicurezza	15



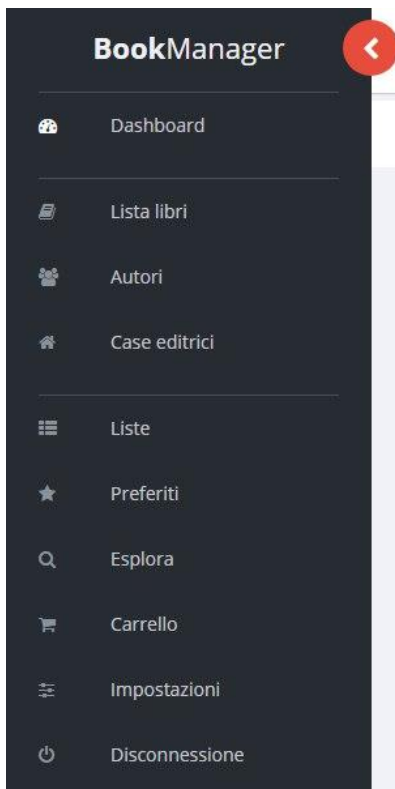
Tema del sito

Book manager è un sito utile all'esplorazione del mondo dei libri. Permette all'utente iscritto di navigare le liste di altri utenti in base a dei tag, di creare delle liste proprie, di impostare dei preferiti o di inserire in autonomia i suoi libri preferiti. Con una semplice iscrizione è possibile gestire la libreria in maniera digitale, visualizzando una wishlist, i libri letti e non e trovandone di nuovi. Può inoltre inserire i libri in un carrello virtuale, con gestione automatica dell'importo e delle quantità.



Sezioni principali

Le sezioni principali del sito possono essere navigate partendo dal menù posto a sinistra. L'utente ha da qui accesso alla gestione base delle anagrafiche, con possibilità di effettuare operazioni C.R.U.D, alla gestione del proprio account e visualizzazione della propria dashboard. Fino alle sezioni per la gestione delle liste, del carrello e di esplorazione e ricerca libri.



Front-end

Separazione presentazione/contenuto/comportamento

In ogni pagina sono stati importati i fogli di stile e delle icone e i javascript utilizzati. Inoltre il menù laterale (uguale per tutte le pagine) viene caricato utilizzando jQuery all'avvio della pagina.

```
... <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Book Manager</title>
  <meta name="description" content="Gestione libri">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="shortcut icon" href="../book.ico">
  <link rel="stylesheet" href="../vendors/bootstrap/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="../vendors/font-awesome/css/font-awesome.min.css">
  <link rel="stylesheet" href="../vendors/css/style.css">
  <link href='../vendors/css/font.css' rel='stylesheet' type='text/css'>

  <script src="../vendors/jquery/dist/jquery.min.js"></script>
  <script src="../vendors/bootstrap/dist/js/bootstrap.min.js"></script>
  <script src="../vendors/main.js"></script>

</head>

<body>
<!-- Left Panel -->

  <aside id="left-panel" class="left-panel">
    <nav class="navbar navbar-expand-sm navbar-default" id="includedContent">
      </nav>
    </aside>
  <script>
```

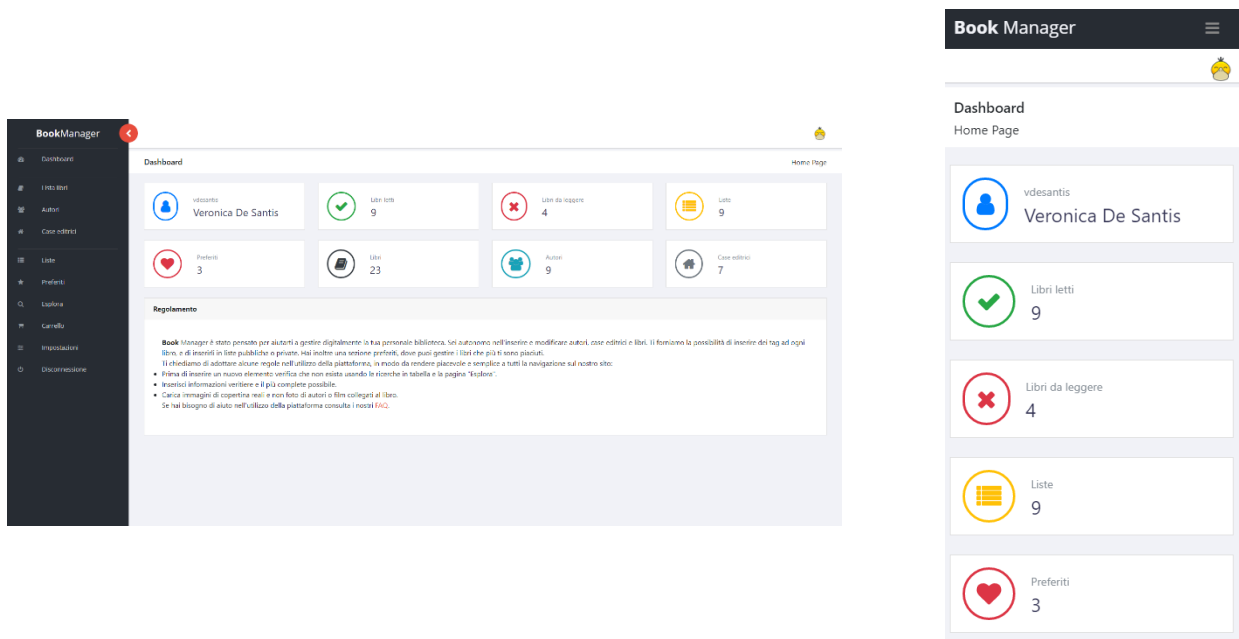
Soluzioni cross-platform

Per rendere la soluzione cross-platform si è utilizzato bootstrap. Il menù laterale, e le card utilizzate nella dashboard per esempio sono completamente dinamiche in base alla dimensione dello schermo.

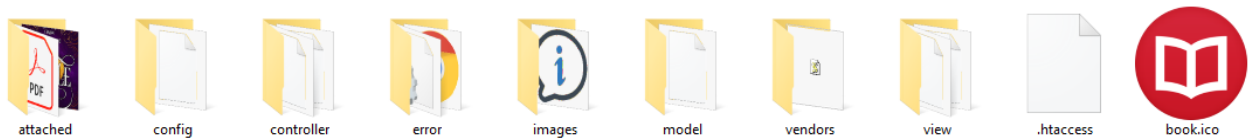
Book Manager

Tecnologie Web (MFN0634)

Studentessa De Santis Veronica – 870166



Organizzazione file e cartelle di progetto



Il sito segue la tecnologia MVC. Nella cartella view infatti sono inserite tutte le viste e le pagine del sistema. Nella cartella controller sono inserite le classi utili alla gestione degli oggetti esistenti nella soluzione, mentre nella cartella model troviamo la classe Query che si occupa di dialogare e di gestire i dati del DB. Inoltre nella cartella config sono gestiti, tramite il file db_config.php i parametri di connessione al database. Nelle altre cartelle troviamo:

Cartella	Contenuto
<i>attached</i>	Contiene le copertine dei libri caricati dagli utenti
<i>error</i>	Contiene le pagine di errore (404, 500, 403)
<i>images</i>	Contiene le immagini utili al sistema (avatar)
<i>vendors</i>	Contiene le librerie esterne utilizzate (Bootstrap, fogli di stile, icone e file js)

Back-end

Architettura generale classi/funzioni

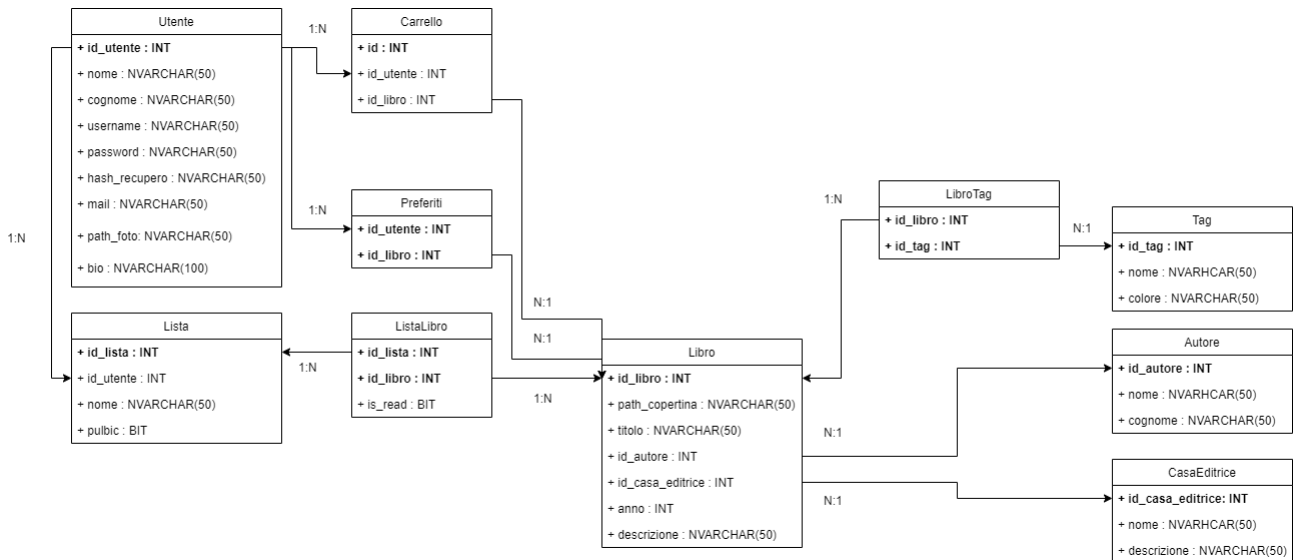
Il back-end è stato gestito utilizzando php ad oggetti. Si ha quindi un oggetto, che afferisce al model, Query, che contiene tutte le funzioni richiamabili dai controller per estrapolare, manipolare e modificare i dati. I controller invece sono suddivisi per macro-sezioni. Si ha quindi un oggetto per la gestione di autori, case editrici, libri e utenti. Tutti gli oggetti che fanno riferimento ai libri sono stati creati nello stesso controller, per mantenere



logicamente unite funzioni che separate perderebbero di senso. I controller vengono utilizzati come semplici “Wrapper”, si occupano di raccogliere i dati che ricevono dalle view, inviarli al modello e gestire un eventuale ritorno di dati.

Schema del db

Il database è relazionato e normalizzato. Nell'immagine si trova il dettaglio lo schema, con relativa visualizzazione dei legami e delle chiavi primarie.



Descrizione delle funzioni remote

Per la visualizzazione delle tabelle si è scelto di usare la libreria [Datatables.net](https://datatables.net), che permette la gestione di filtri, ricerca e ordinamento, alimentate dalle seguenti funzioni remote:

dtListe.php

Metodo get: `http://localhost/controller/dtListe.php?id_utente=1`

JSON:

```
{
  "nome": "Magic";
  "public": 1;
  "azione": "details";
}
```

dtLibri.php

Metodo get: `http://localhost/controller/dtLibri.php`

JSON:

```
{
  "img": "../attached/Bozze.png";
  "codeISBN": "978-8804708810";
  "titolo": "Bozze";
}
```

```
"autore": "Antonio Dikele";  
"casaeditrice": "Mondadori";  
"azione": "details";  
}
```

dtPreferiti.php

Metodo get: `http://localhost/controller/dtPreferiti.php?id_utente=1`

JSON:

```
{  
  "libro": "Notti in bianco, baci a colazione";  
}
```

Gestione errori

```
function exec_query($sql)  
{  
    $connection = open_connection();  
    $query = $connection->prepare($sql);  
    if ($query == false) {  
        header("location: ../error/page_server_error.html");  
    }  
    $result = $query->execute();  
    if($result == false)  
    {  
        header("location: ../error/page_server_error.html");  
    }  
    close_connection($connection);  
    return true;  
}
```

Funzionalità

Login/logout

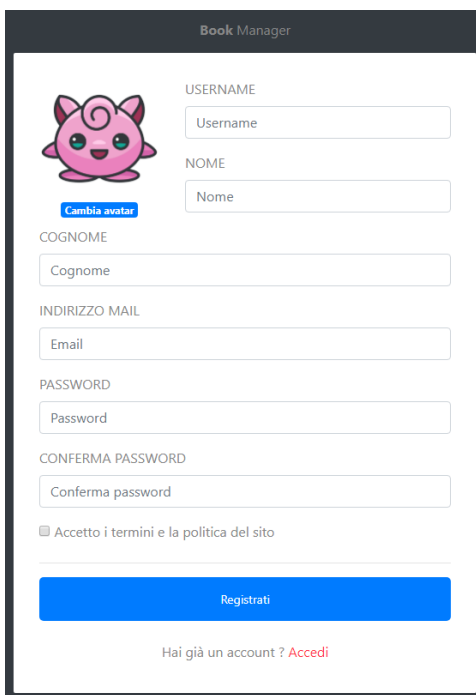
L'utente arrivato in home page deve **obbligatoriamente** inserire le proprie credenziali per continuare la navigazione. Nello specifico sono state usate le SESSION di php per la gestione dell'utente loggato.

Nella pagina db_connection.php (utile per centralizzare il controllo delle sessioni e l'accesso al database, con metodi specifici per l'esecuzione di query in db) viene verificato se l'utente è loggato (**is_logged**), in caso non sia così l'utente viene reindirizzato alla pagina gestione non autorizzati.



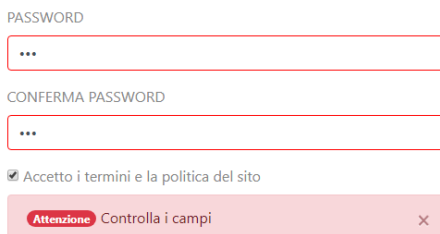
```
<?php
//verifica se l'utente è loggato
function is_logged() {
    if (!isset($_SESSION["username"])) {
        header("Location: ../error/page_unauthorized.html");
    }
}
```

In caso di smarrimento della password l'utente è autonomamente in grado di effettuare il reset semplicemente confermando i suoi dati e l'avatar scelto in fase di registrazione.



Registrazione

Nel caso l'utente non disponesse di un account è autonomo nel crearlo tramite l'apposita pagina. Viene richiesto l'inserimento di alcuni dati basilari come username, nome, cognome, e-mail e la scelta di una password. In questa fase l'utente sceglie il proprio avatar. In caso di successo nella registrazione l'utente viene fatto accedere direttamente alla pagina principale del sito, la dashboard. Nel caso ci fossero alcuni problemi nella compilazione del form di registrazione, l'utente viene avvisato di ricontrollare i campi, con un dettaglio dei campi in cui è stato trovato un errore.



```
<?php
/* User viene utilizzata come una classe di utility. Senza importazioni esterne
 * perchè vengono fatte nelle classi principali */
class User
{
    //Funzione per la creazione di un utente
    public function CreateNewUser($nome, $cognome, $username, $hashedPassword, $mail, $path_foto)
    {
        $qry = new Query;
        return $qry->CreateNewUser($nome, $cognome, $username, $hashedPassword, $mail, $path_foto);
    }
    //Funzione per l'aggiornamento di un utente
    public function UpdateUser($nome, $cognome, $username, $path_foto, $bio, $id_utente)
    {
        $qry = new Query;
        $qry->UpdateUser($nome, $cognome, $username, $path_foto, $bio, $id_utente);
    }
    //Funzione per l'aggiornamento della password di un utente
    public function UpdatePassword($hashedPassword, $id_usente)
    {
        $qry = new Query;
        $qry->UpdatePassword($hashedPassword, $id_usente);
    }
    #region DETAIL USER ...
    #endregion
}
?>
```



```
public function CreateNewUser($nome, $cognome, $username, $hashedPassword, $mail, $path_foto)
{
    $connection = open_connection();
    $query = $connection->prepare('INSERT INTO utente(nome, cognome, username, password, mail, path_foto) VALUES (?, ?, ?, ?, ?, ?)');
    $query->bindParam(1, $nome, PDO::PARAM_STR);
    $query->bindParam(2, $cognome, PDO::PARAM_STR);
    $query->bindParam(3, $username, PDO::PARAM_STR);
    $query->bindParam(4, $hashedPassword, PDO::PARAM_STR);
    $query->bindParam(5, $mail, PDO::PARAM_STR);
    $query->bindParam(6, $path_foto, PDO::PARAM_STR);
    $query->execute();
    return $this->GetLastIdInserted($connection);
}
```

La funzione di inserimento utente in questo caso prepara la query utilizzando la funzione di php apposita per evitare l'sql injection. La funzione inoltre restituisce l'identità della riga appena creata.

Gestione del contenuto generato dall'utente

Il sito **Book Manager** è fortemente caratterizzato da contenuto generato dall'utente. Le sezioni denominate "Liste", "Preferiti" e "Carrello" sono differenti per ogni utente, in quanto personali. Ogni utente infatti è libero di creare liste (con la possibilità di settarle pubbliche o private), di indicare i suoi libri preferiti e di inserirli nel carrello.

Nel caso delle liste l'utente visualizza solo quelle da lui create, tramite l'API [dtListe.php](#). (Illustrata precedentemente). In queste sezioni è possibile visualizzare solamente i contenuti da scelti e, passando con il mouse sopra i libri, avere un dettaglio di alcune informazioni che risultano indispensabili.

Lista

Informazioni

Nome

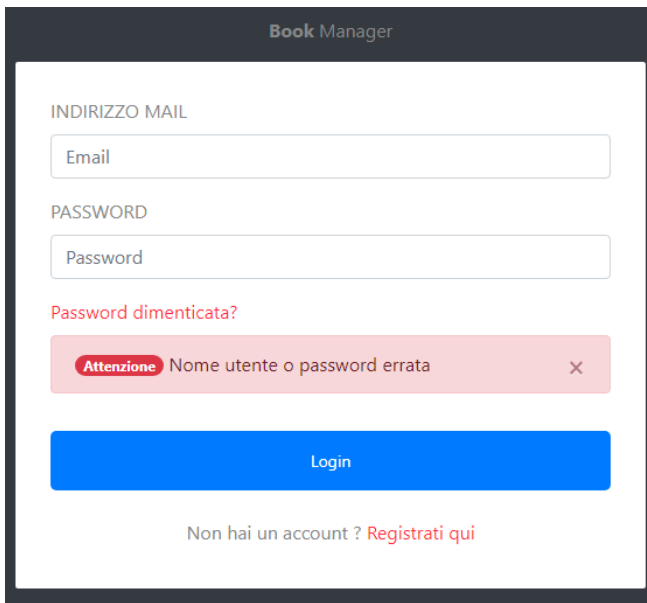
☒ Pubblica

☐ Privata

Salva

Elimina

Annulla



Caratteristiche

Usabilità

Ad ogni operazione effettuata dall'utente viene visualizzato il feedback delle modifiche, sia in caso di successo che in caso di errore.

Nel caso illustrato viene verificata la correttezza delle credenziali inserite dall'utente in fase di login. Se il login va a buon fine l'utente viene fatto accedere alla piattaforma, altrimenti viene visualizzato un errore.

```
function goLogin(){

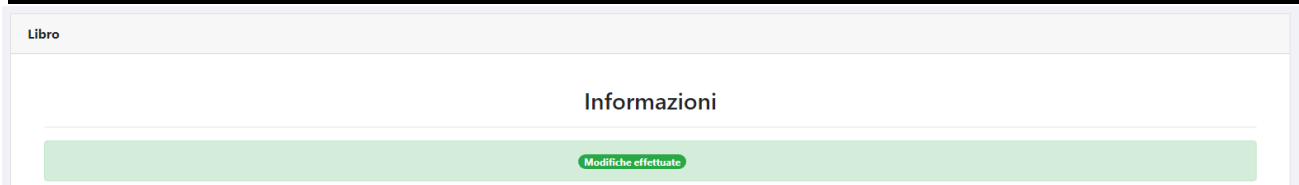
    $mail = $_POST["Email"];
    $password = $_POST["Password"];

    $hashedPassword = hash('sha256', $password);
    $user = new User;
    $row = $user->GetUserByLogin($mail, $hashedPassword);
    if($row!=null)
    {
        start_session($row);
        header("Location: dashboard.php");
    }
    else
    {
        $need_to_display = 1;
        return $need_to_display;
    }
}
if(isset($_POST["login"]))
{
    $need_to_display = goLogin();
}
```

E, successivamente viene visualizzato. Vengono utilizzate le sessioni per visualizzare messaggi in modo dinamico per il reset della password.

```
<?php
    if ($need_to_display == 1)
    {
?>
<script>error_login();</script>
```

```
<?php
}
if (isset($_SESSION["flash_msg"]))
{
    unset($_SESSION["flash_msg"]);
}
?>
<div class="row col-md-12">
    <div class="sufee-alert alert with-close alert-success
    alert-dismissible fade show" id="error_alert">
        La password adesso corrisponde al tuo nome utente
        <button type="button" class="close" data-dismiss="alert"
        aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>
</div>
<?php
}
?>
```



```
<?php if($update == true)
{
    //Modifiche effettuate
    echo '<div class="sufee-alert alert with-close alert-success
    alert-dismissible fade show">
        <center><span class="badge badge-pill
        badge-success">Modifiche effettuate</span></center>
    </div>';
}
?>
```

Per rendere il sito lineare sono stati riutilizzati i seguenti colori:

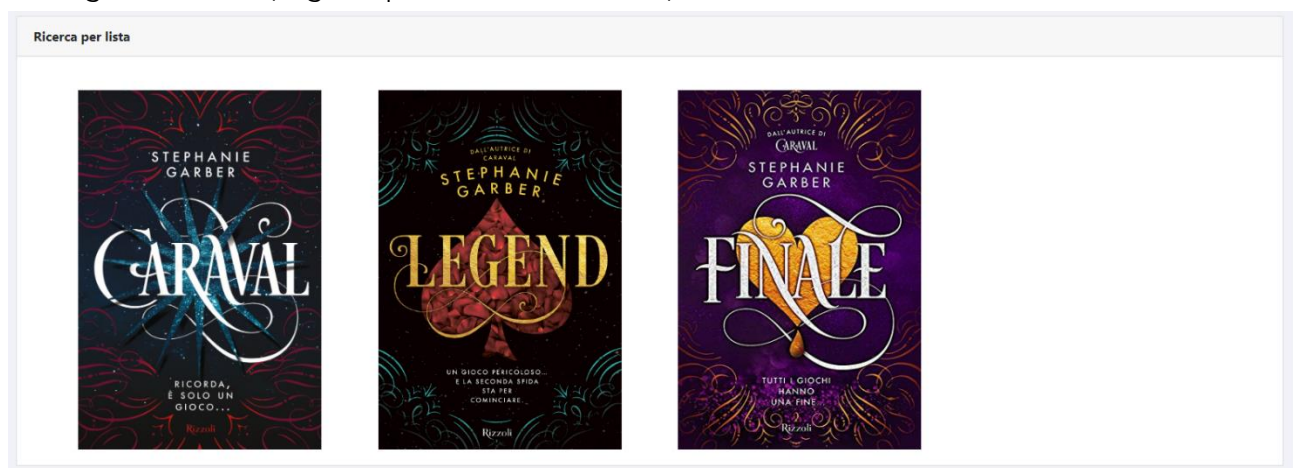
- #e74c3c
- #272c33

Interazione / animazione

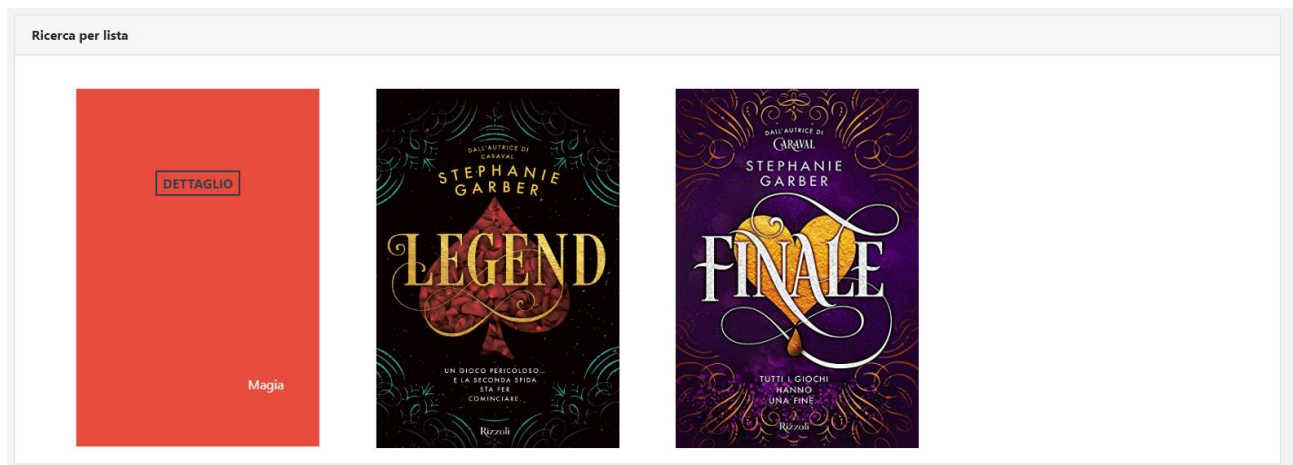


Il sito ha diverse interazioni con l'utente. Nella pagina "Esplora" per esempio i libri possono essere sfogliati. Passando con il mouse sopra il libro le pagine si aprono e viene visualizzato il pulsante per accedere al dettaglio del libro.

In fase di visualizzazione dettaglio liste e ricerca vengono visualizzate solamente le copertine dei libri, passando con il mouse sopra alla copertina è possibile visualizzare altri dettaglio del libro (tag e/o parametri di ricerca).



:hover



Html:

```
<div class="flip-box">
  <div class="flip-box-inner">
    <div class="flip-box-front">
      " style="width:300px;height:440px">
    </div>
    <div class="flip-box-back">
      <span style="position:absolute; top: 80%;">
        Dettagli
      </span>
    </div>
  </div>
</div>
```

Css:

```
.flip-box:hover .flip-box-inner {
  transform: rotateY(180deg);
}
```

Inoltre durante la visualizzazione del libro è possibile, trascinandolo nell'angolo in alto a destra, inserirlo nel carrello.



Html:

```
<div ondrop="drop(event)" ondragover="allowDrop(event)">
  <a href="#" class="dropdown-toggle">
    <i class="fa fa-shopping-cart"></i>
  </a>
</div>

```

Js:

```
function allowDrop(ev) {
  ev.preventDefault();
}
```

```
function drag(ev, id_libro) {
    ev.dataTransfer.setData("text", id_libro);
}

function drop(ev) {
    ev.preventDefault();
    var id_libro = ev.dataTransfer.getData("text");
    //Prendo i dati del libro e lo inserisco nel carrello
    jQuery.ajax({
        url: '../view/esploraLibro.php',
        data: {
            GetBook: 'test',
            id_libro: id_libro,
        },
        type: 'post',
        success: function(output) {
            var splitted = output.split('$$$');
            //Se il libro è già in lista aumento solo la quantità
            if (jQuery("#qta_" + id_libro).text() != "")
            {
                //Aumento la quantità
                var qta = jQuery("#qta_" + id_libro).text();
                //qta = qta.replace("Qtà. ", "");
                qta = parseInt(qta, 10)
                qta = qta + 1;
                jQuery("#qta_" + id_libro).text(qta);
            }
            else
            {
                var div = document.createElement("div");
                div.classList.add("nav-link");
                ... Inserimento libro nel carrello ...
                jQuery("#cart").append(div);
            }
            //Aumento il prezzo in entrambi i casi
            var total = jQuery("#totalPrice").text();
            total = parseFloat(parseFloat(total) + parseFloat(splitted[2]));
            jQuery("#totalPrice").text(total);
            //Apro il dropdown
            jQuery('#cartDropdown').dropdown().dropdown('toggle');
            ev.stopPropagation();
        }
    });
}
```

Sessioni

I metodi `start_session` e `unset_session` vengono usati in fase di login e logout per, rispettivamente, inserire i parametri necessari al funzionamento del sito in session e distruggere la sessione esistente.

La gestione della sessione viene effettuato nel controller `db_connection.php`. I due metodi vengono richiamati rispettivamente durante il login, per iniziare la sessione, e durante il logout, per svuotare e terminare la sessione.

Login:

```
<?php
//verifica se l'utente è loggato
function is_logged() {
    if (!isset($_SESSION["username"])) {
        header("Location: ../error/page_unauthorized.html");
    }
}

//Fa partire la sessione
function start_session($user_info)
{
    session_start();
    $_SESSION['username'] = $user_info['username'];
    $_SESSION['id_utente'] = $user_info['id_utente'];
    $_SESSION['session_id'] = session_id();
}

//Chiude tutte le sessioni
function unset_session()
{
    session_unset();
    session_destroy();
    session_start();
    header("Location: login.php");
}
?>
```

```
function goLogin(){

    $mail = $_POST["Email"];
    $password = $_POST["Password"];

    $hashedPassword = hash('sha256', $password);
    $user = new User;
    $row = $user->GetUserByLogin($mail, $hashedPassword);
    if($row!=null)
    {
        start_session($row);
        header("Location: dashboard.php");
    }
    else
    {
        $need_to_display = 1;
        return $need_to_display;
    }
}
if(isset($_POST["login"]))
{
    $need_to_display = goLogin();
}
```

Logout:

```
<?php
include '../controller/Utility.php';
unset_session();
?>
```



Interrogazione del database



Search results for books. The search bar is empty, and the results are categorized by title, author, publisher, tag, and list. All categories show 'Nessun risultato trovato' (No results found).

Nella funzione che permette la ricerca viene usata la condizione SQL like che permette, utilizzando come parametro la stringa '%\$ricerca%' di ricercare in db tutti i libri che hanno titolo, autore, casa editrice, tag o lista contenente la stringa di ricerca.

```
public function SearchByTitle($search)
{
    $tit = "SELECT libro.*
    FROM libro
    WHERE enabled = 1 AND titolo LIKE '%'. $search .'%'";
    return $tit = exec_query($tit);
}
```

Validazione dati input

La validazione dei dati viene eseguita tramite l'utilizzo dei tag HTML. In quanto il sistema non richiede dati complicati i campi vengono gestiti impostando il corretto type all'input e settando l'attributo required, in modo da rendere obbligatoria la compilazione prima dell'invio del form.

Sicurezza

Per evitare attacchi XSS e HTML/SQL Injection viene sfruttata la funzionalità PDO::prepare, che in combo con la funzione bindParam consente la protezione del db durante l'esecuzione delle query.

```
public function InsertNewAuthor($nome, $cognome, $luogo_nascita, $data_nascita)
{
    $connection = open_connection();
    $query = $connection->prepare('INSERT INTO autore(nome, cognome,
    luogo_nascita, data_nascita, enabled) VALUES (?, ?, ?, ?, 1)');
    $query->bindParam(1, $nome, PDO::PARAM_STR);
    $query->bindParam(2, $cognome, PDO::PARAM_STR);
    $query->bindParam(3, $luogo_nascita, PDO::PARAM_STR);
    $query->bindParam(4, $data_nascita, PDO::PARAM_STR);
    $query->execute();
    close_connection($connection);
}

public function UpdateBook($isbn, $titolo, $year, $description,
$id_autore, $id_casa_editrice, $prezzo, $id_libro)
{
    $connection = open_connection();
```



```
$query = $connection->prepare("UPDATE libro SET titolo = ?, anno = ?,  
descrizione = ?, id_autore = ?, id_casa_editrice = ?, codeISBN = ?,  
prezzo = ? WHERE libro.id_libro = ?");  
$query->bindParam(1, $titolo, PDO::PARAM_STR);  
$query->bindParam(2, $year, PDO::PARAM_STR);  
$query->bindParam(3, $description, PDO::PARAM_STR);  
$query->bindParam(4, $id_autore, PDO::PARAM_INT);  
$query->bindParam(5, $id_casa_editrice, PDO::PARAM_INT);  
$query->bindParam(6, $isbn, PDO::PARAM_STR);  
$query->bindParam(7, $prezzo, PDO::PARAM_STR);  
$query->bindParam(8, $id_libro, PDO::PARAM_STR);  
$query->execute();  
close_connection($connection);  
}
```