Base di Dati - Relazione

Giacomo De Liberali - 857174
Daniele Rigon - 857319
Denny Ruffato - 859171
Luca Fortin - 858986

19 ottobre 2017

Indice

1 Introduzione

L'applicazione, reperibile qui, è un forum di recensione libri composta da un insieme di pagine dinamiche sviluppate in *PHP* e un database *postgreSQL*. L'utente, una volta registratosi, potrà usufruire del catalogo, integrarlo con nuovi dati e fornire recensioni e commenti.

L'applicazione permette le seguenti operazioni:

- Visualizzazione catalogo libri
- Visualizzazione dettaglio di un libro con le recensioni e commenti
- Aggiunta di una recensione per ogni libro
- Aggiunta di commenti ad una recensione
- Aggiunta di commenti di risposta ad un altro commento
- Aggiunta di un giudizio positivo/negativo a commenti e/o recensioni
- Ordinamento del catalogo libri per giudizio positivo crescente/decrescente
- Registrazione di un nuovo utente
- Possibilità di Login e Logout

2 Progetto e funzionalità

2.1 Homepage

La pagina iniziale, visualizzabile solamente dopo aver effettuato il login, permette di visualizzare il catalogo dei libri in modo ordinato. In questa pagina si possono ordinare i libri in base al giudizio e ricercarli per titolo o autore.

2.2 Login

Questa pagina, composta da un semplice form a due input, permette ad un utente già registrato di accedere al portale e di visualizzare, quindi, il catalogo.

2.3 Logout

La funzione di logout è permessa solo da utenti registrati e autenticati al sito. Quando l'utente vorrà scollegarsi dal sito cliccherà sulla voce "Logout" presente nella navbar e sarà disconnesso senza ulteriori richieste. Dopo essersi disconnesso, sarà possibile autenticarsi con un altro profilo.

2.4 Registrazione

Un utente può registrarsi al sito compilando il form proposto, scegliendo anche il proprio username e la propria password, oltre alle informazioni personali quali nome, cognome, email e data di nascita. L'email viene inserita, sebbene non riporti utilità per questo progetto, per eventuali sviluppi futuri dell'applicazione. Dopo aver inserito le informazioni richieste l'utente dovrà cliccare sul bottone per registrarsi e, se le informazioni sono corrette (ovvero l'username scelto non appartiene ad un altro utente) allora l'utente sarà inserito nel database. Dopo essersi registrato, sarà possibile effettuare il login inserendo le nuove credenziali.

2.5 Ricerca di un libro

Un utente può cercare libri o autori nel database tramite l'area di ricerca posta nella navbar in alto a destra, che rimanderà nella homepage con i soli risultati filtrati. A questo punto è possibile cliccare sul libro cercato (se presente nel database) per visualizzare la scheda del libro, con le informazioni su di esso ed eventuali recensioni e commenti.

2.6 Inserire un libro

Se l'utente è autenticato viene permesso di inserire un libro. Cliccando su "Add book", presente nella navbar, si viene reindirizzati ad una pagina contenente un form da completare con le informazioni del libro: titolo, autore, genere e immagine.

2.7 Inserire un autore

Se l'utente è autenticato viene permesso di inserire un autore, il quale sarà poi disponibile nel form di aggiunta libro. Cliccando su "Add author", presente nella navbar, si viene reindirizzati ad una pagina contenente un form da completare con le informazioni dell'autore: nome, cognome, data di nascita e nazionalità

2.8 Recensire un libro

Ogni utente può aggiungere una nuova recensione ad un libro, la quale può essere votata e commentata. Ogni utente può votare "Up" o "Down" la recensione, e togliere il proprio voto se necessario. Un contatore sarà aumentato e decrementato, tenendo così conto dei voti dati fino a quel momento.

2.9 Commentare una recensione e/o un commento

Sotto ogni recensione e ogni commento è possibile aggiungere un nuovo commento.

2.10 Votare commenti e recensioni

Ogni recensione e ogni commento possono essere votati con un voto "Up" o "Down".

3 Sitemap

La seguente immagine rappresenta la struttura del sito e la sua navigazione.

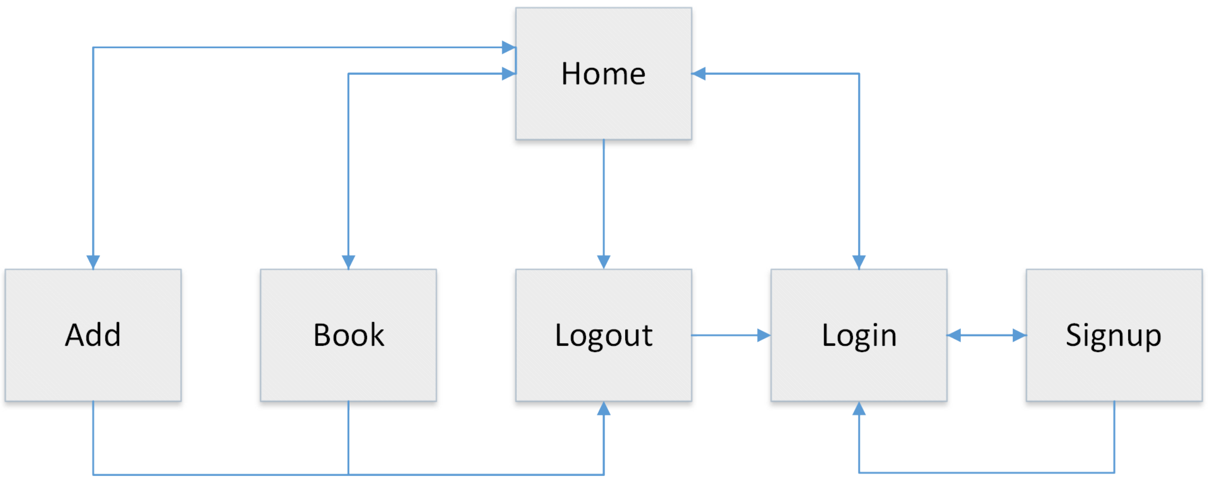

Figura 1: Struttura di navigazione del portale

4 Database

4.1 Modello concettuale

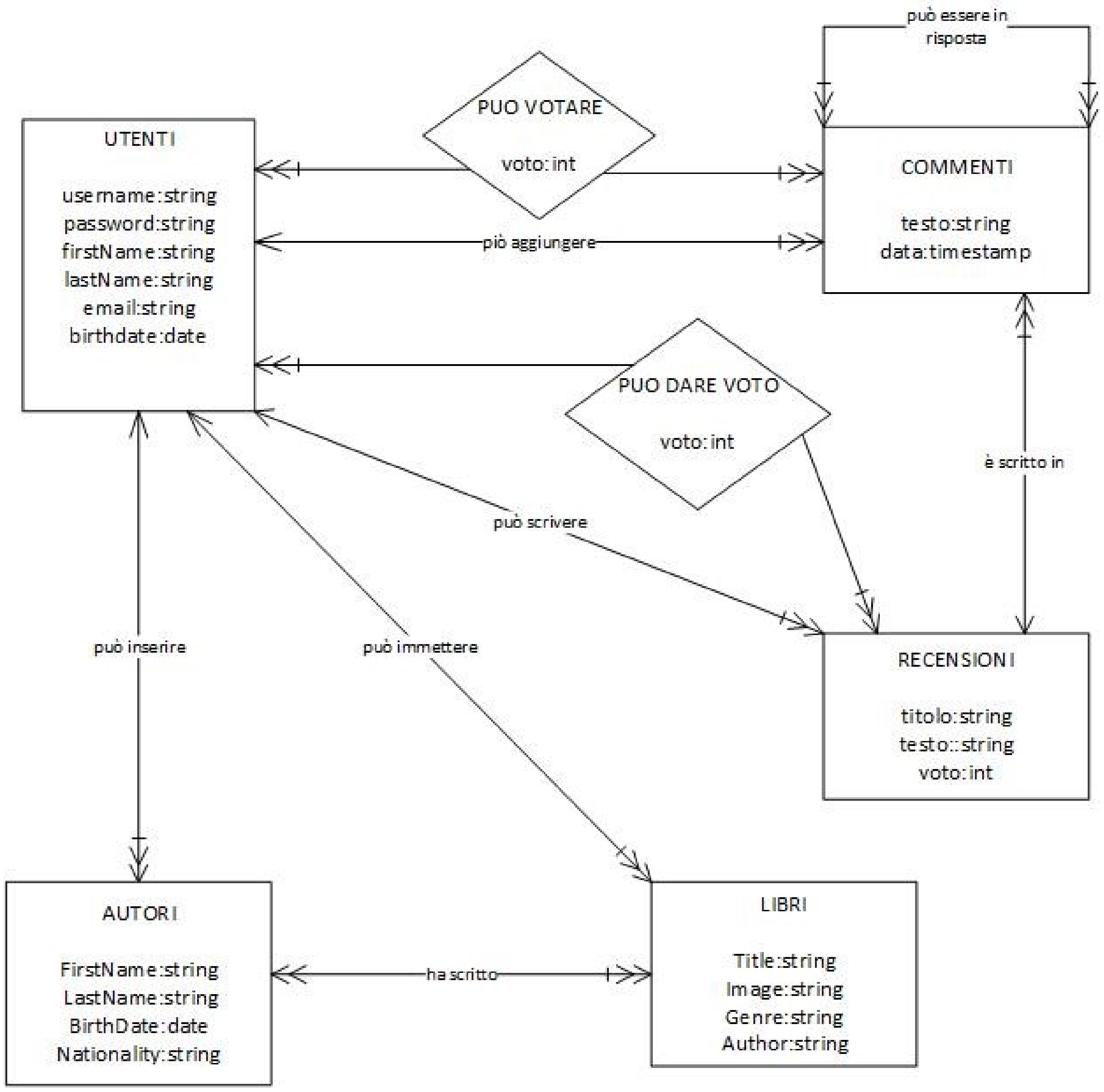

Figura 2: Schema concettuale del database dell'applicazione.

4.2 Modello relazionale

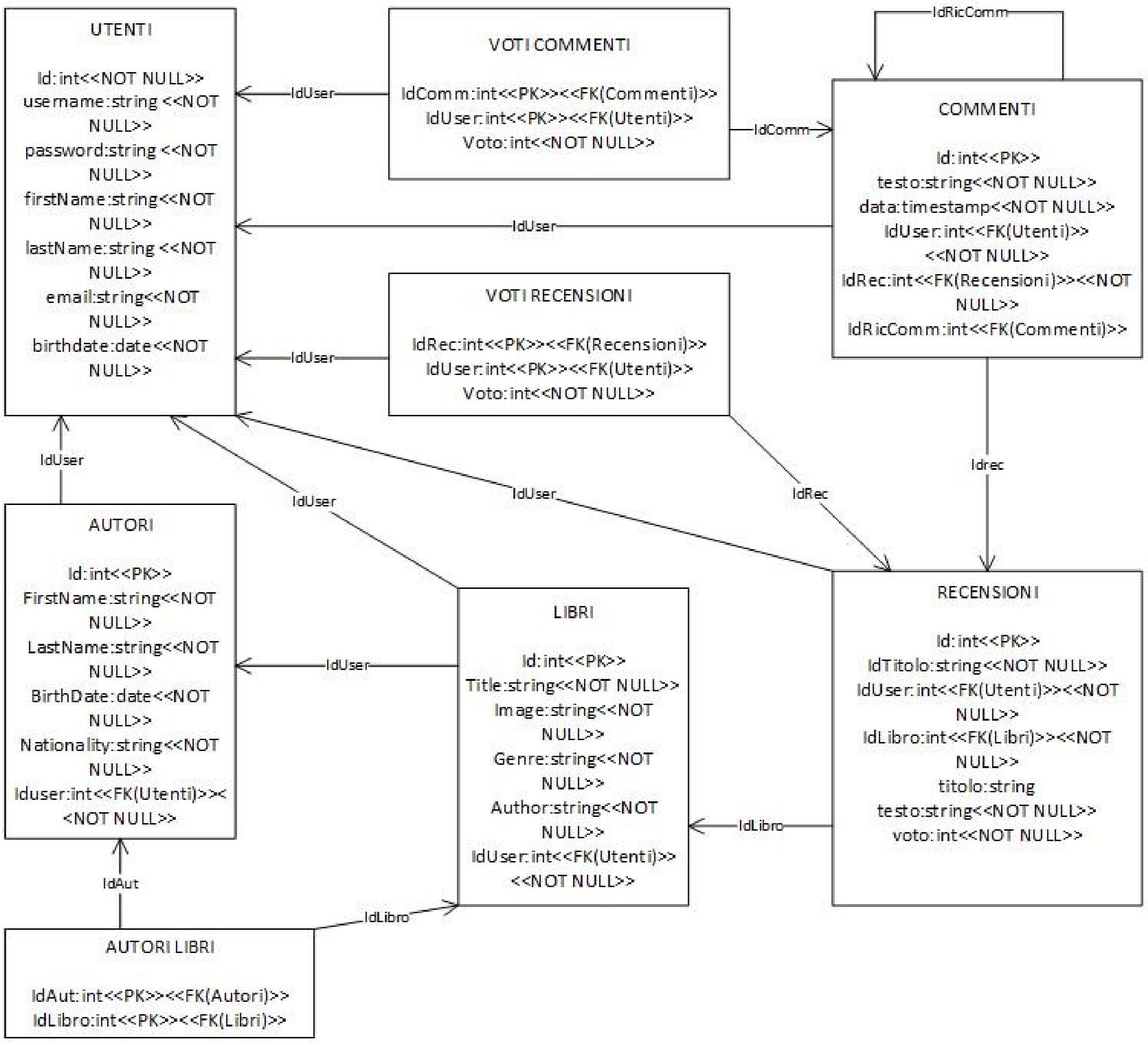
UTENTI
Id:int<<NOT NULL>>
username:string <<NOT NULL>>
password:string <<NOT NULL>>
firstName:string<<NOT NULL>>
lastName:string <<NOT NULL>>
email:string<<NOT NULL>>
birthdate:date<<NOT NULL>>
VOTI COMMENTI
IdComm:int<<PK>><<FK(Commenti)>>
IdUser:int<<PK>><<FK(Utenti)>>
Voto:int<<NOT NULL>>
IdRicComm
COMMENTI
Id:int<<PK>>
testo:string<<NOT NULL>>
data:timestamp<<NOT NULL>>
IdUser:int<<FK(Utenti)>> <<NOT NULL>>
IdRec:int<<FK(Recensioni)>><<NOT NULL>>
IdRicComm:int<<FK(Commenti)>>
IdUser
IdComm
IdUser
VOTI RECENSIONI
IdRec:int<<PK>><<FK(Recensioni)>>
IdUser:int<<PK>><<FK(Utenti)>>
Voto:int<<NOT NULL>>
IdUser
IdUser
IdUser
IdUser
IdRec
Idrec
AUTORI
Id:int<<PK>>
FirstName:string<<NOT NULL>>
LastName:string<<NOT NULL>>
BirthDate:date<<NOT NULL>>
Nationality:string<<NOT NULL>>
Iduser:int<<FK(Utenti)>><<NOT NULL>>
IdUser
LIBRI
Id:int<<PK>>
Title:string<<NOT NULL>>
Image:string<<NOT NULL>>
Genre:string<<NOT NULL>>
Author:string<<NOT NULL>>
IdUser:int<<FK(Utenti)>> <<NOT NULL>>
RECENSIONI
Id:int<<PK>>
IdTitolo:string<<NOT NULL>>
IdUser:int<<FK(Utenti)>><<NOT NULL>>
IdLibro:int<<FK(Libri)>><<NOT NULL>>
titolo:string
testo:string<<NOT NULL>>
voto:int<<NOT NULL>>
IdLibro
IdAut
IdLibro
AUTORI LIBRI
IdAut:int<<PK>><<FK(Autori)>>
IdLibro:int<<PK>><<FK(Libri)>>

Figura 3: Schema relazionale del database dell'applicazione.

5 Codice

Sarà presentato di seguito il codice dell'intera applicazione, suddiviso per responsabilità.

5.1 Models

Tutti i file in questa sezioni sono nella directory */php/models* e rappresentano i modelli dei dati, nonché delle tabelle del database.

5.1.1 DatabaseConnection.php

Rappresenta una connessione al database.

```php
<?php

    /**
     * The database connection information
     */
    class DatabaseConnection {

        /**
         * The database url
         * @var type string
         */
        public $Url;

        /**
         * The databse connection port
         * @var type number
         */
        public $Port;

        /**
         * The database name
         * @var type string
         */
        public $Name;

        /**
         * The database username
         * @var type string
         */
        public $User;

        /**
         * The database password
         * @var type string
         */
        public $Password;

    }

?>
```

5.1.2 Author.php

Rappresenta un autore di libri.

```php
<?php

```

```
    /**
     * A Book author
     */
    class Author {

        /** The author id
         * @var type number */
        public $id;

        /**
         * The author first name
         * @var type string
         */
        public $firstname;

        /**
         * The author last name
         * @var type string
         */
        public $lastname;

        /**
         * The author birthdate
         * @var type Date
         */
        public $birthdate;

        /**
         * The author nationality
         * @var type string
         */
        public $nationality;

    }

?>
```

5.1.3 Book.php

Rappresenta un libro scritto da un autore.

```
<?php

    /**
     * A Book
     */
    class Book {

        /**
         * The book identifier
         * @var type number
         */
        public $id;

        /**
         * The book title
         * @var type string
         */
        public $title;

        /**
```

```php
         * The book image url
         * @var type string
         */
        public $image;

        /**
         * The book genre
         * @var type string
         */
        public $genre;

        /**
         * The book rating (number of stars)
         * @var type float
         */
        public $rating;

        /**
         * The autho's book full name
         * @var type string
         */
        public $author;

        public function render($hasRate=false){
            $this->rating = BooksService::getBookReviewSummary($this->id);
            if (!$this->rating)
                $this->rating = new BookReviewSummary ();
            echo'<div class="row ">
                <div class="col-md-4">
                    <img src="' . $this->image . '" height="200"/>
                </div>
                <div class="col-md-8">
                    <div class="row">
                        <div class="col-xs-12">
                        <a href="./php/book.php?id=' . $this->id . '"><h4>' . $this
                            ->title . '<small>&nbsp;' . $this->author . '</small></
                            h4></a>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-xs-12">
                        <i>' . $this->genre . '</i>
                        </div>
                    </div>';
                    if($hasRate){ echo'
                            <div class="row">
                                <div class="col-xs-12">
                                <div class="rating-block">
                                    <h4>Average user rating</h4>
                                    <h2 class="bold padding-bottom-7">' . ($this->
                                        rating->avarage ? $this->rating->avarage : '
                                        0') . ' <small>/ 5</small></h2>';
                                        for ($i = 1; $i <= 5; $i++) {
                                        echo '
                                            <button type="button" class="btn btn-
                                                warning btn-xs ' . (($i*1.0) > $this
                                                ->rating->avarage ? 'btn-grey' : '')
                                                . '" aria-label="Left Align">
                                                <span class="fa fa-star" aria-hidden
                                                    ="true"></span>
                                            </button>
```

```
                                      ';
                                  }
                                  echo '
                              </div>
                              </div>
                          </div>';
                    } echo '
                   </div>
               </div>';
        }

    }

?>
```

5.1.4 Review.php

Rappresenta una recensione di un libro da parte di un utente.

```
<?php

    /**
     * A book review
     */
    class Review {

        /**
         * The review id
         * @var integer
         */
        public $id_review;

        /**
         * The review title
         * @var string
         */
        public $title;

        /**
         * The review text content
         * @var string
         */
        public $text;

        /**
         * The review number of stars (1-5)
         * @var integer
         */
        public $grade;

        /**
         * The total review score (+1/-1)
         * @var integer
         */
        public $score;

        /**
         * The review author's id
         * @var integer
         */
        public $id_author;

        /**
         * The review author full name
         * @var string
         */
        public $author;

        /**
         * The review book's id
         * @var integer
         */
        public $id_book;

        /**
         * Constructor
         * @param integer $bookId The option book id (for the form rendering)
         */
        public function __construct($bookId = null) {
            if ($bookId)
                $this->id_book = $bookId;
        }

        /**
         * Renders the template of the current element
         */
        public function render() {
            echo '
                <div class="review-block">
                <span>' . $this->author . '</span>
                <div class="row">
                    <div class="col-sm-12">
                            <div class="review-block-rate">';
            for ($i = 1; $i <= 5; $i++) {
                echo '
                    <button type="button" class="btn btn-warning btn-xs ' . ($i >
                        $this->grade ? 'btn-grey' : '') . '" aria-label="Left Align
                        ">
                        <span class="fa fa-star" aria-hidden="true"></span>
                    </button>
                ';
            }
            echo '<a class="btn btn btn-outline-success" href="' . Defaults::
                DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=' .
                Defaults::SCORE_UP . '&userId=' . AuthenticationService::getUserId()
                 . '&reviewId=' . $this->id_review . '&prevUrl=' . $_SERVER["
                REQUEST_URI"] . '"><i class="fa fa-thumbs-o-up"></i></a>
                  <a class="btn btn btn-outline-danger" href="' . Defaults::
                      DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=
                      ' . Defaults::SCORE_DOWN . '&userId=' . AuthenticationService
                      ::getUserId() . '&reviewId=' . $this->id_review . '&prevUrl='
                      . $_SERVER["REQUEST_URI"] . '"><i class="fa fa-thumbs-o-down
                      "></i></a>
                  <a class="btn btn btn-outline-primary" href="' . Defaults::
                      DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=
                      ' . Defaults::SCORE_REMOVE . '&userId=' .
                      AuthenticationService::getUserId() . '&reviewId=' . $this->
                      id_review . '&prevUrl=' . $_SERVER["REQUEST_URI"] . '"><i
                      class="fa fa-times"></i></a>
                  <a class="btn btn btn-outline-primary">' . $this->score . '</a>
                    </div>
                            <div class="review-block-title">' . $this->title . '</
```

```
div>
                        <div class="review-block-description">' . $this->text . '</div>

                        <a class="btn btn-outline-primary btn-sm comments-dialog-opener" data-review-id=' . $this->id_review . ' data-toggle="modal" data-target="#comments" style="margin-left: calc(100% - 85px);margin-top: 10px;">
                            Comments
                        </a>
                    </div>
                </div>
                </div>
        ';
    }

    public function renderForm() {
        echo '
            <form action="' . Defaults::DEFAULT_BASE_URL . '/php/rest/add-review.php" method="post" name="AddReviewForm">
                    <div class="review-block" style="min-width: 100%">
                        <span>' . AuthenticationService::getFullName() . '</span>
                        <div class="row">
                            <div class="col-sm-12">
                                <div class="review-block-rate">
                                    <button type="button" class="btn btn-warning btn-xs" id="star1" onclick="setStar(1)">
                                        <span class="fa fa-star" aria-hidden="true"></span>
                                    </button>
                                    <button type="button" class="btn btn-warning btn-xs" id="star2" onclick="setStar(2)">
                                        <span class="fa fa-star" aria-hidden="true"></span>
                                    </button>
                                    <button type="button" class="btn btn-warning btn-xs" id="star3" onclick="setStar(3)">
                                        <span class="fa fa-star" aria-hidden="true"></span>
                                    </button>
                                    <button type="button" class="btn btn-warning btn-xs" id="star4" onclick="setStar(4)">
                                        <span class="fa fa-star" aria-hidden="true"></span>
                                    </button>
                                    <button type="button" class="btn btn-warning btn-xs" id="star5" onclick="setStar(5)">
                                        <span class="fa fa-star" aria-hidden="true"></span>
                                    </button>
                                </div>
                                <div class="review-block-title">
                                    <input type="text" placeholder="Title" name="Title" class="form-control">
                                </div>
```

```
                                        <div class="review-block-description">
                                            <textarea placeholder="Description" name="
                                                Text" class="form-control"></textarea>
                                        </div>
                                    </div>
                                </div>
                                <input type="submit" value="Submit" class="btn btn-sm"
                                    style="margin-left: calc(100% - 60px); margin-top:
                                    10px;"/>
                            </div>
                            <input type="hidden" name="Grade" id="Grade" value="5"/>
                            <input type="hidden" name="IdBook" value="' . $this->id_book
                                . '"/>
                            <input type="hidden" name="PrevUrl" value="' . $_SERVER['
                                REQUEST_URI'] . '"/>
                            <input type="hidden" name="IdAuthor" value="' .
                                AuthenticationService::getUserId() . '"/>
                        </form>

                        <script>
                            function setStar(value) {
                                for (i = 1; i <= 5; i++) {
                                    if (i <= value) {
                                        $("#star" + i).removeClass("btn-grey");
                                    } else {
                                        $("#star" + i).addClass("btn-grey");
                                    }
                                }
                                $("#Grade").val(value);
                            }
                        </script>
                    ';
        }

    }

?>
```

5.1.5 Comment.php

Rappresenta un commento ad una recensione od ad un altro commento scritto da un utente.

```
<?php

    /**
     * A Comment
     */
    class Comment {

        /**
         * The comment identifier
         * @var integer
         */
        public $id_comment;

        /**
         * The text content
         * @var string
         */
        public $text;

        /**
         * The comment score
         * @var integer
         */
        public $score = 0;

        /**
         * The user identifier
         * @var integer
         */
        public $id_user;

        /**
         * The user full name
         * @var string
         */
        public $userfullname;

        /**
         * The review id
         * @var integer
         */
        public $id_review;

        /**
         * The parent comment identifier
         * @var integer
         */
        public $id_ref_comm;

        /**
         * The comment datetime
         * @var string
         */
        public $date_comment;


        /**
         * The children comment of the current comment
         * @var Array<Comment>
         */
        public $children;

    }

?>
```

5.1.6 BookReviewSummary.php

Rappresenta un riassunto delle recensioni degli utenti rispetto ad un libro.

```
<?php

    /**
     * A Book review summary
     */
    class BookReviewSummary {

        /**
         * The total numer of reviews
         * @var integer
         */
        public $total;

        /**
         * The avarage value
         * @var float
         */
        public $avarage;

        /**
         * The number of reviews with one star
         * @var integer
         */
        public $oneStar;

        /**
         * The number of reviews with two star
         * @var integer
         */
        public $twoStar;

        /**
         * The number of reviews three two star
         * @var integer
         */
        public $threeStar;

        /**
         * The number of reviews with four star
         * @var integer
         */
        public $fourStar;

        /**
         * The number of reviews with five star
         * @var integer
         */
        public $fiveStar;

        /**
         * Renders the HTML for the current item
         */
        public function render() {
            $currentTotal = $this->total > 0 ? $this->total : 1;

            echo '
                <div class="row">
                    <div class="col-sm-6">
                        <div class="rating-block">
                            <h4>Average user rating</h4>
                            <h2 class="bold padding-bottom-7">' . ($this->avarage ?
                                $this->avarage : '0') . ' <small>/ 5</small></h2>';
                            for ($i = 1; $i <= 5; $i++) {
                                echo '
                                    <button type="button" class="btn btn-warning btn
                                        -xs ' . (($i*1.0) > $this->avarage ? 'btn-
                                        grey' : '') . '" aria-label="Left Align">
                                        <span class="fa fa-star" aria-hidden="true
                                            "></span>
                                    </button>
                                ';
```

```
                                    }
                                    echo'
                                </div>
                            </div>
                            <!-- STARS -->
                            <div class="col-sm-6 rating-block">
                                <h4>Rating breakdown</h4>
                                <div class="pull-left">
                                    <div class="pull-left" style="width:35px; line-height
                                        :1;">
                                        <div style="height:9px; margin:5px 0;">5 <span class
                                            ="fa fa-star"></span></div>
                                    </div>
                                    <div class="pull-left" style="width:180px;">
                                        <div class="progress" style="height:9px; margin:8px
                                            0;">
                                            <div class="progress-bar progress-bar-success"
                                                role="progressbar" aria-valuenow="5" aria-
                                                valuemin="0" aria-valuemax="5" style="width:
                                                 ' . $this->fiveStar/$currentTotal*100 . '
                                                %"></div>
                                        </div>
                                    </div>
                                    <div class="pull-right" style="margin-left:10px;">' .
                                        $this->fiveStar . '</div>
                                </div>


                                <div class="pull-left">
                                    <div class="pull-left" style="width:35px; line-height
                                        :1;">
                                        <div style="height:9px; margin:5px 0;">4 <span class
                                            ="fa fa-star"></span></div>
                                    </div>
                                    <div class="pull-left" style="width:180px;">
                                        <div class="progress" style="height:9px; margin:8px
                                            0;">
                                            <div class="progress-bar progress-bar-success"
                                                role="progressbar" aria-valuenow="5" aria-
                                                valuemin="0" aria-valuemax="5" style="width:
                                                 ' . $this->fourStar/$currentTotal*100 . '
                                                %"></div>
                                        </div>
                                    </div>
                                    <div class="pull-right" style="margin-left:10px;">' .
                                        $this->fourStar . '</div>
                                </div>

                                <div class="pull-left">
                                    <div class="pull-left" style="width:35px; line-height
                                        :1;">
                                        <div style="height:9px; margin:5px 0;">3 <span class
                                            ="fa fa-star"></span></div>
                                    </div>
                                    <div class="pull-left" style="width:180px;">
                                        <div class="progress" style="height:9px; margin:8px
                                            0;">
                                            <div class="progress-bar progress-bar-success"
                                                role="progressbar" aria-valuenow="5" aria-
                                                valuemin="0" aria-valuemax="5" style="width:
                                                 ' . $this->threeStar/$currentTotal*100 . '
                                                %”></div>
                                        </div>
                                    </div>
                                    <div class=”pull-right” style=”margin-left:10px;”>’ . $this->threeStar . ’</div>
                                </div>

                                <div class=”pull-left”>
                                    <div class=”pull-left” style=”width:35px; line-height:1;”>
                                        <div style=”height:9px; margin:5px 0;”>2 <span class=”fa fa-star”></span></div>
                                    </div>
                                    <div class=”pull-left” style=”width:180px;”>
                                        <div class=”progress” style=”height:9px; margin:8px 0;”>
                                                <div class=”progress-bar progress-bar-success” role=”progressbar” aria-valuenow=”5” aria-valuemin=”0” aria-valuemax=”5” style=”width:’ . $this->twoStar/$currentTotal*100 . ’%”></div>
                                        </div>
                                    </div>
                                    <div class=”pull-right” style=”margin-left:10px;”>’ . $this->twoStar . ’</div>
                                </div>

                                <div class=”pull-left”>
                                    <div class=”pull-left” style=”width:35px; line-height:1;”>
                                        <div style=”height:9px; margin:5px 0;”>1 <span class=”fa fa-star”></span></div>
                                    </div>
                                    <div class=”pull-left” style=”width:180px;”>
                                        <div class=”progress” style=”height:9px; margin:8px 0;”>
                                                <div class=”progress-bar progress-bar-success” role=”progressbar” aria-valuenow=”5” aria-valuemin=”0” aria-valuemax=”5” style=”width:’ . $this->oneStar/$currentTotal*100 . ’%”></div>
                                        </div>
                                    </div>
                                    <div class=”pull-right” style=”margin-left:10px;”>’ . $this->oneStar . ’</div>
                                </div>

                            </div>
                        </div>
                    ’;
            }

        }

    ?>
```

5.2 Services

Tutti i file in questa sezioni sono nella directory */php/services* e rappresentano i servizi che comunicano direttamente con il database.

5.2.1 Database.php

Gestisce la connessione al database per l'applicazione, è un singleton.

```php
<?php
    require(__DIR__ . '/../models/DatabaseConnection.php');

    // psql --host dblab.dsi.unive.it --username a2016u104

    /**
     * Rappresent the application database connection
     */
    class Database {

        /**
         * The database connection configuration. Read from settings.json
         * @var Models\DatabaseConnection
         */
        private $dbConfig = null;

        /**
         * The application database instance
         * @var Database
         */
        private static $instance = null;

        /**
         * Mark private for singleton use
         */
        private function __construct() {
            $settings = file_get_contents(__DIR__ . "/settings.json");
            $configs = json_decode($settings, true);
            if ($this->isDebug())
                $this->dbConfig = $configs["Debug"]["Database"];
            else
                $this->dbConfig = $configs["Stage"]["Database"];
        }

        /**
         * Returns the databse instance
         * @return Database
         */
        public static function getInstance() {
            if (Database::$instance == null)
                Database::$instance = new Database();
            return Database::$instance;
        }

        /**
         * Returns a new connection
         * @return Connection
         */
        public function getConnection() {
            $connectionString = 'pgsql:host=' . $this->dbConfig["Url"] . ';port=' .
                $this->dbConfig["Port"] . ';dbname=' . $this->dbConfig["Name"];
            $connection = new PDO($connectionString, $this->dbConfig["User"], $this
                ->dbConfig["Password"]);
            $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $connection;
        }

        /**
```

```
         * Return true if the current REMOTE_ADDRESS is local
         * @return boolean
         */
        private function isDebug() {
            $whitelist = array('127.0.0.1', '::1', 'localhost');
            return in_array($_SERVER['REMOTE_ADDR'], $whitelist);
        }

    }

?>
```

5.2.2 Defaults.php

Contiene tutte le stringhe di default dell'applicazione.

```
<?php

    /**
     * A static class with all default values used all across the application
     */
    class Defaults {

        /** Default ASC mode */
        const ASC = 'asc';

        /** Defualt DESC mode */
        const DESC = 'desc';

        /** Server default base starting url */
        const DEFAULT_BASE_URL = '/bd2017';

        /**
         * The file where exception will be logged in. Relative path from web root.
         */
        const LOG_EXCEPTION_FILE = '/php/logs/exceptions.txt';

        /**
         * The score remove value
         */
        const SCORE_REMOVE = 0;

        /**
         * The score up value
         */
        const SCORE_UP = 1;

        /**
         * The score down value
         */
        const SCORE_DOWN = -1;

    }

?>
```

5.2.3 LogsService.php

Gestisce i log delle eccezioni dell'applicazione.

```php
<?php
    require(__DIR__ . '/Defaults.php');

    /**
     * Provides methods to log information or expection in a log file
     */
    class LogsService {

        /**
         * Append the given string in the specified file
         * @param string $filePath The file path
         * @param string $contentToAppend The string to append
         */
        private static function log($filePath, $contentToAppend) {
            // Write the contents to the file,
            // using the FILE_APPEND flag to append the content to the end of the
                file
            // and the LOCK_EX flag to prevent anyone else writing to the file at
                the same time
            file_put_contents($filePath, $contentToAppend, FILE_APPEND | LOCK_EX);
        }

        /**
         * Logs in the 'logs/exceptions.txt' the given excpetion object
         * @param Object $exceptionObj
         */
        public static function logException($exceptionObj) {
            $now = new DateTime();
            $str = "---------------\n";
            $str .= $now->format('Y-m-d H:i:s');
            $str .= "\n";
            $str .= $exceptionObj->__toString();
            $str .= "\n---------------\n";
            self::log(__DIR__ . "/../../" . Defaults::LOG_EXCEPTION_FILE, $str);
        }

    }

?>
```

5.2.4 AuthenticationService.php

Gestisce l'autenticazione dell'utente e mette a disposizione la CRUD per la tabella *users*.

```php
<?php

    ob_start();
    session_start();
    require_once(__DIR__ . '/Database.php');
    require_once(__DIR__ . '/LogsService.php');

    /**
     * Provides base methods for authenticating the users and all user CRUD
        operations.
     */
    class AuthenticationService {

        /**
         * Given a username and a MD5 password returns true if a user with the given
             credentials exisits or not.
         * @param username string The user username
         * @param password string The user password (in MD5)
         * @return boolean
         *
         */
        public static function login($username, $password) {
            try {
                $dbconn = Database::getInstance()->getConnection();
                $statement = $dbconn->prepare('
                    SELECT
                        id,
                        username,
                        concat_ws(\'\', firstName, \' \', lastname) as userfullname,
                        firstname
                    FROM
                        users
                    WHERE
                        lower(username) = lower(:username) AND password = :password
                ');
                $statement->bindParam(":username", $username, PDO::PARAM_STR);
                $statement->bindParam(":password", $password, PDO::PARAM_STR);
                $statement->execute();
                $result = $statement->fetch(PDO::FETCH_NUM);
                if ($result[0] != null) {
                    // set session username
                    $_SESSION["UserId"] = $result[0];
                    $_SESSION["Username"] = $result[1];
                    $_SESSION["UserFullName"] = $result[2];
                    $_SESSION["UserFirstName"] = $result[3];
                    return true;
                }
                // destry the session username and return false
                unset($_SESSION["Username"]);
                unset($_SESSION["UserId"]);
                unset($_SESSION["UserFullName"]);
                unset($_SESSION["UserFirstName"]);
                return false;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

        /**
         * Returns true if a session username exisits, false othwrwise.
         * @return boolean
         */
        public static function isLoggedIn() {
            return (isset($_SESSION["Username"]) && $_SESSION["Username"] != null) ?
                 true : false;
        }

        /**
         * Returns the current logged in user username or null
         * @return string
         */
        public static function getUsername() {
            return self::isLoggedIn() ? $_SESSION["Username"] : null;
        }

        /**
```

```php
         * Returns the current logged in user full name or null
         * @return string
         */
        public static function getFullName() {
            return self::isLoggedIn() ? $_SESSION["UserFullName"] : null;
        }

        /**
         * Returns the current logged in user first name or null
         * @return string
         */
        public static function getFirstName() {
            return self::isLoggedIn() ? $_SESSION["UserFirstName"] : null;
        }

        /**
         * Returns the current logged in user id or null
         * @return integer
         */
        public static function getUserId() {
            return self::isLoggedIn() ? $_SESSION["UserId"] : null;
        }

        /**
         * Register a new user
         * @param string username The user username
         * @param string password The user password (in MD5)
         * @param string firstName The user name
         * @param string lastName The user surname
         * @param string email The user email
         * @param string birthdate The user birthdate
         * @return boolean
         *
         */
        public static function register($username, $password, $firstName, $lastName, $email, $birthdate) {
            try {
                $dbconn = Database::getInstance()->getConnection();
                $statement = $dbconn->prepare('INSERT INTO users VALUES (
            DEFAULT,
            :username,
            :password,
            :firstName,
            :lastName,
            :email,
            :birthdate
        )');
                $statement->bindParam(":username", $username, PDO::PARAM_STR);
                $statement->bindParam(":password", $password, PDO::PARAM_STR);
                $statement->bindParam(":firstName", $firstName, PDO::PARAM_STR);
                $statement->bindParam(":lastName", $lastName, PDO::PARAM_STR);
                $statement->bindParam(":email", $email, PDO::PARAM_STR);
                $statement->bindParam(":birthdate", $birthdate, PDO::PARAM_STR);
                $statement->execute();
                return $statement->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

```

```
    }

?>
```

5.2.5 AuthorsService.php

Gestisce la CRUD per la tabella *authors*.

```
<?php

    require_once(__DIR__ . '/AuthenticationService.php');
    require_once(__DIR__ . '/../models/Book.php');
    require_once(__DIR__ . '/../models/Author.php');

    /**
     * Provides all Author CRUD operations.
     */
    class AuthorsService {

        /**
         * Adds a new author
         * @param string firstName The author first name
         * @param string lastName The author last name
         * @param string birhDate The author birth date
         * @param string nationality The author nationality (string)
         * @return The new added author id
         */
        public static function addAuthor($firstName, $lastName, $birhDate,
            $nationality) {
            try {
                if (!$birhDate || strlen($birhDate))
                    $birhDate = null;
                if (!$nationality || strlen($nationality))
                    $nationality = null;

                $dbconn = Database::getInstance()->getConnection();
                $authorStatment = $dbconn->prepare('
                INSERT INTO authors VALUES (
                    DEFAULT,
                    :firstName,
                    :lastName,
                    :birhDate,
                    :nationality
                )
            ');
                $authorStatment->bindParam(":firstName", $firstName, PDO::PARAM_STR)
                    ;
                $authorStatment->bindParam(":lastName", $lastName, PDO::PARAM_STR);
                $authorStatment->bindParam(":birhDate", $birhDate, PDO::PARAM_STR);
                $authorStatment->bindParam(":nationality", $nationality, PDO::
                    PARAM_STR);
                $authorStatment->execute();
                $isAdded = $authorStatment->rowCount() == 1;
                if ($isAdded)
                    return $dbconn->lastInsertId("authors_id_seq");
                else
                    return 0;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return 0;
            }
```

```php
        }

        /**
         * Returns all the authors
         * @return Array<Author>
         */
        public static function getAuthors() {
            try {
                $dbconn = Database::getInstance()->getConnection();
                $statement = $dbconn->prepare('
                SELECT
                    *
                FROM
                    authors
                '
                );
                $statement->execute();
                return $statement->fetchAll(PDO::FETCH_CLASS, "Author");
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         * Deletes an Author given its id
         * @param $authorId integer
         * @return boolean
         */
        public static function deleteAuthor($authorId) {
            if ($authorId == null || $authorId < 0) {
                return false;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        DELETE
                        FROM
                            authors
                        WHERE
                            id = :$authorId
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":$authorId", $authorId, PDO::PARAM_INT);
                $result = $statement->execute();
                return $result->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

    }

?>
```

5.2.6 BooksService.php

Gestisce la CRUD per la tabella *books* e *books_authors*.

```php
<?php

    require_once(__DIR__ . '/AuthenticationService.php');
    require_once(__DIR__ . '/../models/Book.php');
    require_once(__DIR__ . '/../models/Author.php');
    require_once(__DIR__ . '/../models/Review.php');
    require_once(__DIR__ . '/../models/BookReviewSummary.php');

    /**
     * Provides all Book CRUD operations.
     */
    class BooksService {

        /**
         * Adds a new book
         * @param string title The book title
         * @param string image The book image url
         * @param string genre The book genre
         * @param integer authorId The book authorId
         * @return The new added book id
         */
        public static function addBook($title, $image, $genre, $authorId) {
            try {
                $dbconn = Database::getInstance()->getConnection();
                $statement = $dbconn->prepare('
                INSERT INTO books VALUES (
                    DEFAULT,
                    :title,
                    :image,
                    :genre
                )
            ');
                $statement->bindParam(":title", $title, PDO::PARAM_STR);
                $statement->bindParam(":image", $image, PDO::PARAM_STR);
                $statement->bindParam(":genre", $genre, PDO::PARAM_STR);
                $statement->execute();
                $isBookAdded = $statement->rowCount() == 1;

                if ($isBookAdded) {
                    $statement = $dbconn->prepare('
                    INSERT INTO books_authors VALUES (
                        :authorId,
                        :bookId
                    )
                ');
                    $bookId = $dbconn->lastInsertId("books_id_seq");
                    $statement->bindParam(":authorId", $authorId, PDO::PARAM_INT);
                    $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                    $statement->execute();
                    $isLinkAdded = $statement->rowCount() == 1;
                }

                return $isBookAdded && $isLinkAdded;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

        /**
```

```
         * Adds a new review
         * @param string $title
         * @param string $text
         * @param integer $grade
         * @param integer $IdAuthor
         * @param integer $idBook
         * @return boolean
         */
        public static function addReview($title, $text, $grade, $IdAuthor, $idBook)
            {
            try {
                $dbconn = Database::getInstance()->getConnection();
                $statement = $dbconn->prepare('
                INSERT INTO reviews VALUES (
                        DEFAULT,
                        :title,
                        :text,
                        :grade,
                        0,
                        :idAuthor,
                        :idBook
                    )
                ');
                $statement->bindParam(":title", $title, PDO::PARAM_STR);
                $statement->bindParam(":text", $text, PDO::PARAM_STR);
                $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
                $statement->bindParam(":idAuthor", $IdAuthor, PDO::PARAM_INT);
                $statement->bindParam(":idBook", $idBook, PDO::PARAM_INT);
                $statement->execute();
                return $statement->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

        /**
         * Adds a review grade (+1/-1)
         * @param integer $reviewId The review id
         * @param integer $userId The user id
         * @param integer $grade The grade
         * @return boolean
         */
        public static function addReviewGrade($reviewId, $userId, $grade) {
            try {
                $dbconn = Database::getInstance()->getConnection();

                $statement = $dbconn->prepare('
                SELECT *
                FROM
                    grades_reviews
                WHERE
                        id_user = :userId
                        AND
                        id_review = :reviewId
                ');
                $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
                $statement->execute();
                $isExisting = $statement->rowCount() == 1;

```

```php
                if ($isExisting) {
                    $currentValue = $statement->fetchAll()[0]["grade"];
                    $statement = $dbconn->prepare('
                        UPDATE
                            grades_reviews
                        SET
                            grade = :grade
                        WHERE
                                id_user = :userId
                                AND
                                id_review = :reviewId
                    ');
                    $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                    $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
                    $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
                    $statement->execute();
                    return $statement->rowCount() == 1;
                }

                // ELSE NEW GRADE
                $statement = $dbconn->prepare('
                INSERT INTO grades_reviews VALUES (
                        :userId,
                        :reviewId,
                        :grade
                    )
                ');
                $currentScore = $grade == Defaults::SCORE_DOWN ? -1 : 1;
                $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
                $statement->bindParam(":grade", $currentScore, PDO::PARAM_INT);
                $statement->execute();
                return $statement->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

        /**
         * Given a word returns the book's title (one or more) that contains that
             word.
         * @param string keyword The keywork to search
         * @return Array<Book>
         */
        public static function research($keyword = null, $sort = Defaults::DESC) {
            try {
                if (!isset($keyword))
                    $keyword = null;
                if (!isset($sort))
                    $sort = Defaults::DESC;

                $dbconn = Database::getInstance()->getConnection();
                $key = ($keyword ? $keyword : '');
                $query = '
                SELECT
                    b.id,
                    b.title,
                    b.image,
                    b.genre,
                    AVG(r.grade) AS rate,
                    concat_ws(\'\', a.firstName, \' \', a.lastname) as authorfullname
                FROM
                    books AS b JOIN books_authors AS ba
                    ON ba.id_book= b.id JOIN authors AS a ON a.id= ba.id_author
                    LEFT JOIN reviews AS r ON r.id_book=b.id
                WHERE
                    lower(b.title) LIKE lower(:key)
                        OR
                    lower(concat_ws(\'\', a.firstName, \' \', a.lastname)) LIKE lower(:key)
                        OR
                    lower(b.genre) LIKE lower(:key)
                GROUP BY
                    b.id,
                    b.title,
                    b.image,
                    b.genre,
                    authorfullname
                ORDER BY
                    rate '. $sort . ' NULLS LAST';
                $statement = $dbconn->prepare($query);
                $statement->bindValue(":key", '%' . $keyword . '%');
                $row = $statement->execute();

                $books = array();
                while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
                    $book = new Book();
                    $book->id = $row['id'];
                    $book->title = $row['title'];
                    $book->image = $row['image'];
                    $book->genre = $row['genre'];
                    $book->author = $row['authorfullname'];
                    $book->rating = 0;
                    $books[] = $book;
                }
                return $books;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         * Deletes a Book given its id
         * @param integer $bookId integer
         * @return boolean
         */
        public static function deleteBook($bookId) {
            if ($bookId == null || $bookId < 0) {
                return false;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        DELETE
                        FROM
                            books
                        WHERE
                            id = :bookId
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                $result = $statement->execute();
                return $result->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         * Gets a Book given its id
         * @param $bookId integer
         * @return Book
         */
        public static function getBook($bookId) {
            if ($bookId == null || $bookId < 0) {
                return null;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        SELECT
                            *
                        FROM
                            books
                        WHERE
                            id = :bookId
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                $statement->execute();
                $books = $statement->fetchAll(PDO::FETCH_CLASS, "Book");

                if (count($books) > 0)
                    return $books[0];

                return null;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         *
         * Return all the reviews of the given book
         * @param $bookId integer The book id
         * @return Array<Review>
         */
        public static function getReviews($bookId) {
            if ($bookId == null || $bookId < 0) {
                return null;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        SELECT
                            reviews.*,
                            concat_ws(\'\', users.firstname, \' \', users.lastname) as author,
                            SUM(coalesce(grades_reviews.grade,0)) as score
                        FROM
                            reviews JOIN users ON reviews.id_author = users.id
                            LEFT JOIN grades_reviews ON reviews.id_review = grades_reviews.id_review
                        WHERE
                            id_book = :bookId
                        GROUP BY reviews.id_review,author
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                $statement->execute();
                $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "Review");

                return $reviews;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         * Returns true if the user has already done a review on the specified book
         * @param integer $authorId The author id
         * @param integer $bookId The book id
         * @return boolean
         */
        public static function hasReview($authorId, $bookId) {
            if ($bookId == null || $bookId < 0 || $authorId == null || $authorId < 0) {
                return null;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        SELECT
                            *
                        FROM
                            reviews JOIN users ON reviews.id_author = users.id
                        WHERE
                            id_book = :bookId
                                AND
                            id_author = :authorId
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                $statement->bindParam(":authorId", $authorId, PDO::PARAM_INT);
                $statement->execute();
                return $statement->rowCount() == 1;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         *
```

```php
         * Return the book's reviews summary
         * @param $bookId integer The book id
         * @return BookReviewSummary
         */
        public static function getBookReviewSummary($bookId) {
            if ($bookId == null || $bookId < 0) {
                return null;
            }
            try {
                $dbconn = Database::getInstance()->getConnection();
                $query = '
                        SELECT
                            COUNT(*) AS total,
                            ROUND(AVG(grade),1) AS avarage,
                            COUNT(CASE WHEN grade = 1 THEN grade END) AS "oneStar",
                            COUNT(CASE WHEN grade = 2 THEN grade END) AS "twoStar",
                            COUNT(CASE WHEN grade = 3 THEN grade END) AS "threeStar
                                ",
                            COUNT(CASE WHEN grade = 4 THEN grade END) AS "fourStar",
                            COUNT(CASE WHEN grade = 5 THEN grade END) AS "fiveStar"
                        FROM
                            reviews
                        WHERE
                            reviews.id_book = :bookId
                    ';

                $statement = $dbconn->prepare($query);
                $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
                $statement->execute();
                $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "BookReviewSummary
                    ");

                if (count($reviews) > 0)
                    return $reviews[0];

                return null;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

    }

?>
```

5.2.7 CommentsService.php

Gestisce la CRUD per la tabella *comments*.

```php
<?php

    require_once(__DIR__ . '/AuthenticationService.php');
    require_once(__DIR__ . '/../models/Comment.php');

    /**
     * Provides all Comments CRUD operations.
     */
    class CommentsService {

        /**
     * Returns all the comments of a review (in a hierarcic tree)
     * @return Array<Comment>
     */
    public static function getCommentsByReviewId($reviewId) {
        try {
            $dbconn = Database::getInstance()->getConnection();
            $statement = $dbconn->prepare('
                SELECT
                    c.*,
                    COALESCE(SUM(gc.grade),0) as score,
                    concat_ws(\'\', u.firstName, \' \', u.lastname) as
                        userfullname
                FROM
                    comments c JOIN users u ON c.id_user = u.id
                    LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
                WHERE id_review = :id_review AND c.id_ref_comm IS NULL
                GROUP BY c.id_comment, userfullname
                ORDER BY c.date_comment ASC
            ');
            $statement->bindParam(":id_review", $reviewId);
            $row = $statement->execute();
            $comments = array();
            while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
                $comment = new Comment();
                $comment->id_comment = $row['id_comment'];
                $comment->text = $row['text'];
                $comment->score = $row['score'];
                $comment->id_user = $row['id_user'];
                $comment->userfullname = $row['userfullname'];
                $comment->id_review = $row['id_review'];
                $comment->id_ref_comm = $row['id_ref_comm'];
                $comment->date_comment = $row['date_comment'];
                $comment->canEdit = $row['id_user'] == AuthenticationService::
                    getUserId();
                $comment->children = self::getCommentsByParentId($row['
                    id_comment']);
                $comments[] = $comment;
            }
            return $comments;
        } catch (PDOException $e) {
            LogsService::logException($e);
            return null;
        }
    }

    /**
     * Add a new comment
     * @param Comment $comment
     * @return boolean
     */
    public static function addComment($comment) {
        try {
            $dbconn = Database::getInstance()->getConnection();
            $statement = $dbconn->prepare('
            INSERT INTO COMMENTS VALUES (
                DEFAULT,
                :text,
                :id_user,
                :id_review,
                :id_ref_comm,
                :date_comment
            )'
            );
            $statement->bindParam(":text", $comment->text);
            $statement->bindParam(":id_user", $comment->id_user);
            $statement->bindParam(":id_review", $comment->id_review);
            $statement->bindParam(":id_ref_comm", $comment->id_ref_comm);
            $statement->bindParam(":date_comment", $comment->date_comment);
            $statement->execute();
            $isAdded = $statement->rowCount() == 1;
            if ($isAdded)
                return $dbconn->lastInsertId("comments_id_comment_seq");
            else
                return 0;
        } catch (PDOException $e) {
            LogsService::logException($e);
            return null;
        }
    }

    /**
     * Deletes a comment and all its children, given its id
     * @param integer $commentId
     * @return boolean
     */
    public static function deleteAuthor($commentId) {
        if ($commentId == null || $commentId < 0) {
            return false;
        }
        try {
            $dbconn = Database::getInstance()->getConnection();
            $query = '
                    DELETE
                    FROM
                        comments
                    WHERE
                        id_comment = :commentId
                ';

            $statement = $dbconn->prepare($query);
            $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
            $result = $statement->execute();
            return $result->rowCount() == 1;
        } catch (PDOException $e) {
            LogsService::logException($e);
            return null;
        }
    }

    /**
     * Given a parent id comment, return all the hierarchic tree
     * @param integer $parentId
     * @return Array<Comment>
     */
    public static function getCommentsByParentId($parentId) {
        try {
            $dbconn = Database::getInstance()->getConnection();
            $statement = $dbconn->prepare('
                SELECT
                    c.*,
                    COALESCE(SUM(gc.grade),0) as score,
                    concat_ws(\'\', u.firstName, \' \', u.lastname) as
```

```
                            userfullname
                    FROM
                        comments c JOIN users u ON c.id_user = u.id
                        LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
                    WHERE
                        c.id_ref_comm = :parentId
                    GROUP BY c.id_comment, userfullname
                ');
                $statement->bindParam(":parentId", $parentId);
                $row = $statement->execute();
                $comments = array();
                while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
                    $comment = new Comment();
                    $comment->id_comment = $row['id_comment'];
                    $comment->text = $row['text'];
                    $comment->score = $row['score'];
                    $comment->id_user = $row['id_user'];
                    $comment->userfullname = $row['userfullname'];
                    $comment->id_review = $row['id_review'];
                    $comment->id_ref_comm = $row['id_ref_comm'];
                    $comment->date_comment = $row['date_comment'];
                    $comment->canEdit = $row['id_user'] == AuthenticationService::
                        getUserId();
                    $comment->children = self::getCommentsByParentId($row['
                        id_comment']);
                    $comments[] = $comment;
                }
                return $comments;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return null;
            }
        }

        /**
         * Adds a comment grade (+1/-1)
         * @param integer $commentId The comment id
         * @param integer $userId The user id
         * @param integer $grade The grade
         * @return boolean
         */
        public static function addCommentGrade($commentId, $userId, $grade) {
            try {
                $dbconn = Database::getInstance()->getConnection();

                $statement = $dbconn->prepare('
                    SELECT
                        *
                    FROM
                        grades_comments
                    WHERE
                            id_user = :userId
                            AND
                            id_comment = :commentId
                ');
                $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
                $statement->execute();
                $isExisting = $statement->rowCount() == 1;

                if ($isExisting) {
                    $currentValue = $statement->fetchAll()[0]["grade"];
                    $statement = $dbconn->prepare('
                        UPDATE
                            grades_comments
                        SET
                            grade = :grade
                        WHERE
                                id_user = :userId
                                AND
                                id_comment = :commentId
                    ');
                    $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                    $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
                    $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
                    $statement->execute();
                    if ($statement->rowCount() == 1)
                        return self::getScore($commentId);
                    return null;
                }

                // ELSE NEW GRADE
                $statement = $dbconn->prepare('
                INSERT INTO grades_comments VALUES (
                        :userId,
                        :commentId,
                        :grade
                    )
                ');
                $currentScore = $grade == Defaults::SCORE_DOWN ? -1 : 1;
                $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
                $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
                $statement->bindParam(":grade", $currentScore, PDO::PARAM_INT);
                $statement->execute();
                if ($statement->rowCount() == 1)
                    return self::getScore($commentId);
                return null;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

        /**
         * Return the score of the given comment
         * @param integer $commentId
         * @return integer
         */
        public static function getScore($commentId) {
            try {
                $dbconn = Database::getInstance()->getConnection();

                $statement = $dbconn->prepare('
                SELECT
                    SUM(grade) as score
                FROM
                    grades_comments
                WHERE
                        id_comment = :commentId
                GROUP BY id_comment
                ');
                $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
```

```php
                $statement->execute();
                $result = $statement->fetchAll()[0];

                return $result["score"] ? $result["score"] : 0;
            } catch (PDOException $e) {
                LogsService::logException($e);
                return false;
            }
        }

    }

?>
```

5.3 Rest

Tutti i file in questa sezione sono nella directory */php/rest* e rappresentano il collegamento fra le view e il database (analoghi al controller nel pattern *MVC*).

5.3.1 register.php

Permette di registrare un nuovo utente.

```php
<?php
    require_once('../services/AuthenticationService.php');
    require_once('../services/Defaults.php');

    $isRegistred = false;

    // if a username and a password is given in POST, then try to login
    if(isset($_POST["Username"]) &&
        isset($_POST["Password"]) &&
        isset($_POST["RepeatPassword"]) &&
        isset($_POST["Email"]) &&
        isset($_POST["FirstName"]) &&
        isset($_POST["LastName"]) &&
        isset($_POST["BirthDate"])){
        $isRegistred = AuthenticationService::register($_POST["Username"], md5(
            $_POST["Password"]),$_POST["FirstName"],$_POST["LastName"],$_POST["Email
            "],$_POST["BirthDate"]);
    }

    if($isRegistred){
        // return on the previous page
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/register.php?signup=
            true');
    }else{
         // return on theregister page with a GET param
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/register.php?errors=
            true');
    }

?>
```

5.3.2 login.php

Permette ad un utente di effettuare il login.

```php
<?php
    require_once('../services/AuthenticationService.php');
    require_once('../services/Defaults.php');

    $isLogged = false;

    // if a username and a password is given in POST, then try to login
    if(isset($_POST["Username"]) && isset($_POST["Password"])){
        $isLogged = AuthenticationService::login($_POST["Username"], md5($_POST["
            Password"]));
    }

    if($isLogged){
        // return on the previous page
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/login.php');
    }else{
         // return on the login page with a GET param
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/login.php?
            wrongCredentials=true');
    }

?>
```

5.3.3 logout.php

Permette ad un utente di effettuare il logout.

```
<?php
    require_once('../services/AuthenticationService.php');

    // if logged in, destroy the session
    if(isset($_SESSION["Username"]))
        unset($_SESSION["Username"]);

    // return on the previous page
    header('Location: ' . $_SERVER['HTTP_REFERER']);
?>
```

5.3.4 add-author.php

Permette di aggiungere al database un nuovo autore di libri.

```
<?php
    require_once('../services/AuthenticationService.php');
    require_once('../services/AuthorsService.php');
    require_once('../services/Defaults.php');

    $authorId = false;

    // if a username and a password is given in POST, then try to login
    if(isset($_POST["FirstName"]) &&
        isset($_POST["LastName"]) &&
        isset($_POST["BirthDate"]) &&
        isset($_POST["Nationality"])){
        $authorId = AuthorsService::addAuthor($_POST["FirstName"], $_POST["LastName"
            ], $_POST["BirthDate"], $_POST["Nationality"]);
    }

    if($authorId != null && $authorId > 0){
        // return on the previous page
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?added=
            true');
    }else{
         // return on the register page with a GET param
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?errors=
            true');
    }

?>
```

5.3.5 add-book.php

Permette di aggiungere al database un nuovo libro.

```
<?php
    require_once('../services/AuthenticationService.php');
    require_once('../services/BooksService.php');
    require_once('../services/Defaults.php');

    $isReviewAdded = false;

    // if a username and a password is given in POST, then try to login
    if(isset($_POST["Title"]) &&
        isset($_POST["Image"]) &&
        isset($_POST["Genre"]) &&
        isset($_POST["Author"])){
        $isReviewAdded = BooksService::addBook($_POST["Title"],$_POST["Image"],
            $_POST["Genre"],$_POST["Author"]);
    }

    if($isReviewAdded){
        // return on the previous page
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?added=
            true');
    }else{
         // return on the register page with a GET param
        header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?errors=
            true');
    }

?>
```

5.3.6 add-review.php

Permette di aggiungere al database una nuova recensione per un libro.

```
<?php

    require_once('../services/AuthenticationService.php');
    require_once('../services/BooksService.php');
    require_once('../services/Defaults.php');

    $isReviewAdded = false;

    // if a username and a password is given in POST, then try to login
    if (isset($_POST["Title"]) &&
            isset($_POST["Text"]) &&
            isset($_POST["Grade"]) &&
            isset($_POST["IdBook"]) &&
            isset($_POST["PrevUrl"]) &&
            isset($_POST["IdAuthor"])) {
```

```
        $isReviewAdded = BooksService::addReview($_POST["Title"], $_POST["Text"],
            $_POST["Grade"], $_POST["IdAuthor"], $_POST["IdBook"]);
    }

    if ($isReviewAdded && isset($_POST["PrevUrl"])) {
        header('Location: ' . $_POST["PrevUrl"]);
    } else {
        // return on the register page with a GET param
        header('Location: ' . Defaults::DEFAULT_BASE_URL);
    }
?>
```

5.3.7 add-comment.php

Permette di aggiungere al database un nuovo commento.

```
<?php

    require_once('../services/CommentsService.php');

    $request_body = file_get_contents('php://input');
    $comment = json_decode($request_body);
    $result = null;
    if ($comment) {
        $result = CommentsService::addComment($comment);
    }
    echo json_encode($result);
?>
```

5.3.8 add-grade.php

Permette di aggiungere un voto ad una recensione esistente.

```
<?php

    require_once('../services/AuthenticationService.php');
    require_once('../services/BooksService.php');
    require_once('../services/Defaults.php');

    $isReviewAdded = false;

    // if a username and a password is given in POST, then try to login
    if (isset($_GET["type"]) &&
            isset($_GET["reviewId"]) &&
            isset($_GET["userId"]) &&
            isset($_GET["prevUrl"])&&
            isset($_GET["grade"])) {
        $isReviewAdded = BooksService::addReviewGrade($_GET["reviewId"], $_GET["
            userId"], $_GET["grade"]);
    }

    if ($isReviewAdded && isset($_GET["prevUrl"])) {
        header('Location: ' . $_GET["prevUrl"]);
    } else {
        // return on the register page with a GET param
        header('Location: ' . Defaults::DEFAULT_BASE_URL);
    }
?>
```

5.3.9 add-comment-grade.php

Permette di aggiungere un voto ad un commento esistente.

```php
<?php

    require_once('../services/CommentsService.php');
    require_once('../services/Defaults.php');

    // if a username and a password is given in POST, then try to login
    if (isset($_GET["type"]) &&
            isset($_GET["userId"]) &&
            isset($_GET["grade"])) {
        echo json_encode(CommentsService::addCommentGrade($_GET["commentId"], $_GET[
            "userId"], $_GET["grade"]));
    } else {
        echo "null";
    }
?>
```

5.3.10 get-comments.php

Permette di recuperare tutti i commenti di una recensione, sotto forma di albero gerarchico.

```php
<?php

require_once('../services/AuthenticationService.php');
require_once('../services/CommentsService.php');

// if a username and a password is given in POST, then try to login
if (isset($_GET["reviewId"])) {
    $result = CommentsService::getCommentsByReviewId($_GET["reviewId"]);
    echo json_encode($result);
} else {
    echo json_encode([]);
}
?>
```

5.4 Partials

Tutti i file in questa sezione sono nella directory */php/partials* e rappresentano delle view riutilizzate in n pagine.

5.4.1 navbar.php

Partial view per la navbar dell'applicazione.

```php
<?php
require_once( __DIR__ . '/../services/AuthenticationService.php');
require_once( __DIR__ . '/../services/Defaults.php');


/**
 *  Indicates if the current user is logged in or not
 */
$isLogged = AuthenticationService::isLoggedIn();
?>

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>"/>BD2017
        </a>
```

```
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
            <a class="nav-link" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>">Home <span class="sr-only">(current)</span></a>
        </li>
        <?php
        if (!$isLogged) {
            echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::DEFAULT_BASE_URL . '/php/login.php">Login</a></li>';
            echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::DEFAULT_BASE_URL . '/php/register.php">Sign up</a></li>';
        } else {
            echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::DEFAULT_BASE_URL . '/php/rest/logout.php">Logout</a></li>';
            echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::DEFAULT_BASE_URL . '/php/add-book.php">Add</a></li>';
        }
        ?>
    </ul>
    <?php
    if ($isLogged) {
        require(__DIR__ . '/search-box.php');
    }
    ?>

</div>
</nav>
```

5.4.2 search-box.php

Form di ricerca libri utilizzata nella parte destra della navbar.

```
 <!-- The navbar search box -->

<div class="input-group input-group-sm" id="searchBox">
    <input id="keyword" type="text" class="form-control form-control-sm" value="<?php echo isset($_GET["keyword"]) ? $_GET["keyword"] : ""; ?>" placeholder="<?php echo $isLogged ? AuthenticationService::getFirstName() . " search" : "Search" ?> a book..." aria-label="Search for...">
    <span class="input-group-btn hidden" id="cancelButtonGroup">
        <button class="btn btn-secondary btn-sm" type="button" id="cancelButton">&times;</button>
    </span>
    <span class="input-group-btn">
        <button class="btn btn-secondary btn-sm" type="button">
            <?php
            if (isset($_GET["sort"]) && $_GET["sort"] == Defaults::ASC) {
                echo '<i class="fa fa-sort-numeric-asc" id="bookSort"></i>';
            } else {
                echo '<i class="fa fa-sort-numeric-desc" id="bookSort"></i>';
            }
            ?>
        </button>
    </span>
    <span class="input-group-btn">
```

```
        <button class="btn btn-secondary btn-sm" type="button" id="searchButton">Go
            !</button>
    </span>
</div>
```

5.4.3 scripts.php

Contiene tutti gli script (*Javascript*) utilizzati nell'applicazione.

```
<?php
    require_once(__DIR__ . '/../services/Defaults.php');
?>

<script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-
    hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" crossorigin="anonymous"></script><
    script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.
    min.js" integrity="sha384-b/U6ypiBEHpOf/4+1nzFpr53nxSS+
    GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4" crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/js/bootstrap.min.
    js" integrity="sha384-
    h0AbiXch4ZDo7tp9hKZ4TsHbi047NrKGLO3SEJAg45jXxnGIfYzk4Si90RDIqNm1" crossorigin="
    anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.19.1/moment-with-
    locales.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment-timezone/0.5.13/moment-
    timezone.min.js"></script>
<script>
    moment.tz.add('Europe/Rome|CET CEST|-10
        -20|01010101010101010101010101010101010101010101010101010101010101010101010101010101010101
        arB0 Lz0 1cN0 1db0 1410 1on0 Wp0 1qL0 17d0 1cL0 M3B0 5M20 WM0 1fA0 1cM0 16M0
         1iM0 16m0 1de0 1lc0 14m0 1lc0 WO0 1qM0 GTW0 On0 1C10 LA0 1C00 LA0 1EM0 LA0
        1C00 LA0 1zc0 Oo0 1C00 Oo0 1C00 LA0 1zc0 Oo0 1C00 LA0 1C00 LA0 1zc0 Oo0 1C00
         Oo0 1zc0 Oo0 1fC0 1a00 1cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1fA0 1cM0 1cM0 1
        cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1cM0 1fA0 1cM0 1cM0 1cM0 1cM0 1cM0 1
        cM0 1cM0 1cM0 1cM0 1cM0 1fA0 1o00 11A0 1o00 11A0 1o00 11A0 1qM0 WM0 1qM0 WM0
         1qM0 11A0 1o00 11A0 1o00 11A0 1qM0 WM0 1qM0 WM0 1qM0 WM0 1qM0 11A0 1o00 11
        A0 1o00 11A0 1qM0 WM0 1qM0 WM0 1qM0 11A0 1o00 11A0 1o00 11A0 1o00 11A0 1qM0
        WM0 1qM0 WM0 1qM0 11A0 1o00 11A0 1o00 11A0 1qM0 WM0 1qM0 WM0 1qM0 11A0 1o00
        11A0 1o00 11A0 1o00 11A0 1qM0 WM0 1qM0 WM0 1qM0 11A0 1o00 11A0 1o00 11A0 1
        qM0 WM0 1qM0 WM0 1qM0 WM0 1qM0 11A0 1o00 11A0 1o00|39e5');
</script>
<script src="<?php echo Defaults::DEFAULT_BASE_URL ?>/assets/scripts/vue.js"></
    script>
<script src="<?php echo Defaults::DEFAULT_BASE_URL ?>/assets/scripts/vue-resource.
    min.js"></script>


<script>
    // START NAVBAR SEARCH BOX

    /*
    * Checks if the cancel button should be visible or not
    */
    $(document).ready(function() {
        checkCancelButton();
    });

    /*
    * When a click in the GO! button is recived, navigates to the index page with
        the get keyword param valorized with the input value
    */
    $(document).on("click", "#searchButton", function() {
        // when click on the Go button
        searchBook();
    });

    $(document).on("click", "#cancelButton", function() {
        // when click on the X button
        $("#keyword").val("");
        $(this).hide();
        $("#searchButton").click();
    });

    /**
    * Every time a button is pressed in the input text, checks if the cancel button should be visible or not.
    * If the button pressed is the Enter (invio), search the book with the current keyword.
    */
    $(document).on("keyup", "#keyword", function(e) {
        // when enter in the keyword input text
        checkCancelButton();

        if (e.which == 13)
            searchBook();
    });

    /**
    * Shows or hides the cancel button if the keyword input is valorized or empty
    */
    function checkCancelButton() {
        var keyword = $("#keyword").val();
        if (!keyword || keyword.length == 0)
            $("#cancelButtonGroup").addClass("hidden");
        else
            $("#cancelButtonGroup").removeClass("hidden");
    }

    /**
    * Navigates to the index page with the get keyword param valorized with the input value
    */
    function searchBook() {
        // navigate to homepage with a get parameter
        var keyword = $("#keyword").val();
        var currentSort = "<?php echo isset($_GET["sort"]) ? $_GET["sort"] : Defaults::ASC ?>";
        var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
        // if no value is provided, navigate without the param
        if (!keyword || keyword.length == 0)
            window.location.href = defaultBaseUrl + currentSort ? '?sort=' + currentSort : '';
        else
            window.location.href = defaultBaseUrl + "/?keyword=" + keyword  + (currentSort ? '&sort=' + currentSort : '');
    }

    /*
    * When click on sort change button in navbar
    */
    $(document).on("click","#bookSort",function(){
        var currentSort = "<?php echo isset($_GET["sort"]) ? $_GET["sort"] :
            Defaults::DESC ?>";
        var currentKeyword = "<?php echo isset($_GET["keyword"]) ? $_GET["keyword"]
            : null ?>";
        var defaultAsc = "<?php echo Defaults::ASC; ?>";
        var defaultDesc= "<?php echo Defaults::DESC; ?>";
        var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";

        if (!currentKeyword || currentKeyword.length == 0)
            window.location.href = defaultBaseUrl + currentSort ? '?sort=' + (
                currentSort == defaultDesc ? defaultAsc : defaultDesc) : '';
        else{
            window.location.href = defaultBaseUrl + "/?keyword=" + currentKeyword  +
                 (currentSort ? '&sort=' + (currentSort == defaultDesc ? defaultAsc
                : defaultDesc) : '');
        }
    });
    // END NAVBAR SEARCH BOX
</script>
```

5.4.4 styles.php

Contiene tutti gli stili (*CSS*) utilizzati nell'applicazione.

```
<?php
    require_once(__DIR__ . '/../services/Defaults.php');
?>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no
    ">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/
    css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/
    font-awesome.min.css">

 <!-- Custom CSS -->
<link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
    /custom.css">
<link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
    /reviews.css">
<link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
    /comments.css">
```

5.4.5 register-form.php

Form di registrazione.

```
<?php
    require_once( __DIR__ . '/../services/Defaults.php');
?>
<div class="row">
    <div class="col-md-6 ml-md-auto mr-md-auto">

        <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/register.
            php'; ?>" method="post" name="RegisterForm">
            <h3>Sign Up</h3>
            <br>
            <div class="form-group">
                <input type="text" class="form-control form-control-sm" name="
                    FirstName" placeholder="First name" autofocus="true" required="
                    true" autocomplete="off">
            </div>
            <div class="form-group">
                <input type="text" class="form-control form-control-sm" name="
                    LastName" placeholder="Last name" autofocus="true" required="
                    true" autocomplete="off"/>
            </div>
            <div class="form-group">
                <input type="date" class="form-control form-control-sm" name="
                    BirthDate" placeholder="Birth date" autofocus="true" required="
                    true" />
            </div>
            <div class="form-group">
                <input type="text" class="form-control form-control-sm" name="
                    Username" placeholder="Username" autofocus="true" required="true
                    " autocomplete="off"/>
            </div>
            <div class="form-group">
                <input type="email" class="form-control form-control-sm" name="Email
                    " placeholder="Email" required="true" autocomplete="off"/>
            </div>
            <div class="form-group">
                <input type="password" class="form-control form-control-sm" name="
                    Password" placeholder="Password" required="true" autocomplete="
                    off"/>
            </div>
            <button class="btn btn-sm btn-primary"  name="Submit" value="SignUp"
                type="Submit">Register</button>
        </form>
    </div>
</div>
```

5.4.6 login-form.php

Form di login.

```
<?php
    require_once( __DIR__ . '/../services/Defaults.php');
?>
<form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/login.php'; ?>"
    method="post" name="LoginForm">
    <h3>Login</h3>
    <br>
    <div class="form-group">
        <input type="text" class="form-control form-control-sm" name="Username"
            placeholder="Username" required="" autofocus="" />
    </div>
    <div class="form-group">
        <input type="password" class="form-control form-control-sm" name="Password"
            placeholder="Password" required=""/>
    </div>
    <button class="btn btn-sm btn-primary"  name="Submit" value="Login" type="Submit
        ">Sign in</button>
</form>
```

5.4.7 add-author-form.php

Form di creazione di un nuovo autore di libri.

```
<?php
    require_once( __DIR__ . '/../services/Defaults.php');

    if (!AuthenticationService::isLoggedIn()) {
        die();
    }
?>
<div class="col-md-6">
    <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/add-author.php
        '; ?>" method="post" name="AddAuthorForm">
        <h3>Add author</h3>
        <br>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="FirstName"
                 placeholder="First name" required="true" autocomplete="off">
        </div>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="LastName"
                placeholder="Last name" required="true" autocomplete="off"/>
        </div>
        <div class="form-group">
            <input type="date" class="form-control form-control-sm" name="BirthDate"
                 placeholder="BirthDate" />
        </div>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="
                Nationality" placeholder="Nationality" />
        </div>
        <button class="btn btn-sm btn-primary"  name="Submit" value="Add" type="
            Submit">Add</button>
    </form>
</div>
```

5.4.8 add-book-form.php

Form di creazione di un nuovo libro.

```
<?php
    require_once(__DIR__ . '/../services/AuthenticationService.php');
    require_once(__DIR__ . '/../services/AuthorsService.php');
    require_once(__DIR__ . '/../services/Defaults.php');


    if (!AuthenticationService::isLoggedIn()) {
        die();
    }

    $authors = AuthorsService::getAuthors();
?>
<div class="col-md-6">
    <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/add-book.php';
        ?>" method="post" name="AddBookForm">
        <h3>Add book</h3>
        <br>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="Title"
                placeholder="Title" autofocus="true" required="true" autocomplete="
                off">
        </div>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="Image"
                placeholder="Image url" required="true" autocomplete="off"/>
        </div>
        <div class="form-group">
            <input type="text" class="form-control form-control-sm" name="Genre"
                placeholder="Genre" required="true" />
        </div>
        <div class="form-group">
            <select name="Author" class="form-control">
                <?php
                    foreach ($authors as $author) {
                        echo '<option value="' . $author->id . '">' . $author->
                            firstname . ' ' . $author->lastname . '</option>';
                    }
                ?>
            </select>
        </div>
        <button class="btn btn-sm btn-primary"  name="Submit" value="Add" type="
            Submit">Add</button>
    </form>
</div>
```

5.4.9 comments.php

Contiene il dialogo modale utilizzato per visualizzare i commenti di una recensione. Il rendering dei commenti è stato realizzato con il framework *Javascript* Vue.js.

```
<?php
require_once( __DIR__ . '/../services/AuthenticationService.php');
require_once( __DIR__ . '/../services/Defaults.php');
?>
<!-- Modal -->
<div class="modal fade" id="comments" tabindex="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label=
                    "Close"><span aria-hidden="true">&times;</span></button>
                <h4 class="modal-title" id="myModalLabel">Comments</h4>
            </div>
            <div class="modal-body" id="commentsList">
                 <!-- START COMMENTS -->

                <template v-if="comments.length">
                    <comment
                        v-for="comment in comments"
                        v-bind:comment="comment"
                        v-bind:level="0"
                        v-bind:key="comment.id_comment">
                    </comment>
                </template>
                <div class="alert alert-warning" v-else>
                    {{emptyMessage}}
                </div>

                 <!-- END COMMENTS -->
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-outline-primary btn-sm"  data-
                    dismiss="modal">Close</button>
            </div>
        </div>
    </div>
```

```
</div>
<script>
    Vue.prototype.window = window;
    Vue.http.headers.common['content-type'] = 'application/json';

    Vue.component('comment', {
        name: 'comment',
        props: ["comment", "level"],
        template: `
                <div>
                    <div v-bind:class="'card comment-' + level">
                        <div class="card-header">
                                {{comment.userfullname}}
                                <span v-if="comment.id_comment > 0">commented {{
                                    window.moment(comment.date_comment).tz("Europe/
                                    Rome").fromNow()}}</span>
                                <span v-else>aggiungi un nuovo commento</span>
                                <template v-if="comment.id_comment > 0">
                                    <a class="btn btn-sm btn-outline-success" v-on:
                                        click="gradeUp"><i class="fa fa-thumbs-o-up"
                                        ></i></a>
                                    <a class="btn btn-sm btn-outline-danger"  v-on:
                                        click="gradeDown"><i class="fa fa-thumbs-o-
                                        down"></i></a>
                                    <a class="btn btn-sm btn-outline-primary" v-on:
                                        click="gradeRemove"><i class="fa-times"></i
                                        ></a>
                                    <a class="btn btn-sm btn-outline-primary">{{
                                        comment.score}}</a>
                                    <button class="btn btn-sm" style="float: right"
                                        v-on:click="reply">
                                        <i class="fa fa-reply"></i>
                                    </button>
                                </template>
                        </div>
                        <div class="card-block">
                            <template v-if="!(comment.id_comment > 0)">
                                <input type="text" class="form-control" v-model="
                                    comment.text" style="max-width: calc(100% - 38px
                                    ); display: inline"/>
                                <button class="btn btn-sm btn-outline-success" v-on:
                                    click="save">
                                    <i class="fa fa-save" v-if="!isLoading"></i>
                                    <i class="fa fa-circle-o-notch fa-spin" v-else
                                        ></i>
                                </button>
                            </template>
                            <p class="card-text p-1" v-else>{{comment.text}}</p>
                        </div>
                    </div>
                    <template v-if="comment.children && comment.children.length > 0"
                        >
                        <comment
                            v-for="child in comment.children"
                            v-bind:comment="child"
                            v-bind:level="level+1"
                            v-bind:key="child.id_comment">
                        </comment>
                    </template>
                </div>
            `,
        methods: {
            save: function () {
                if (this.comment.text && this.comment.text.length) {
                    this.isLoading = true;
                    this.$http.post(
                            "<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/add-
                                comment.php",
                            this.comment
                            ).then(data => data.json())
                            .then(data => {
                                if (data != null) {
                                    this.comment.id_comment = data;
                                    this.refresh();
                                    this.isLoading = false;
                                }
                            }).catch((ex) => {
                        console.error(ex);
                        this.isLoading = false;
                    });
                }
            },
            refresh() {
                this.$forceUpdate();
            },
            reply() {
                if (!this.comment.children)
                    this.comment.children = [];
                if (this.comment.children.length == 0 || (this.comment.children[this
                    .comment.children.length - 1].id_comment > 0)) {
                    // push?
                    this.comment.children.unshift({
                        userfullname: newEmptyComment.userfullname,
                        id_user: newEmptyComment.id_user,
                        id_review: newEmptyComment.id_review,
                        date_comment: newEmptyComment.date_comment,
                        id_ref_comm: this.comment.id_comment || null,
                        score: 0,
                        children: []
                    });
                    this.refresh();
                }
            },
            gradeUp() {
                var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
                    comment-grade.php?type=comment&grade=<?php echo Defaults::
                    SCORE_UP ?>&userId=<?php echo AuthenticationService::getUserId()
                     ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php echo
                     $_SERVER["REQUEST_URI"] ?>';
                this.$http.get(url)
                        .then(data => data.json())
                        .then(data => {
                            if (data != null)
                                this.comment.score = data;
                        });
            },
            gradeDown() {
                var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
                    comment-grade.php?type=comment&grade=<?php echo Defaults::
                    SCORE_DOWN ?>&userId=<?php echo AuthenticationService::getUserId
                    () ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php
                    echo $_SERVER["REQUEST_URI"] ?>';
                this.$http.get(url)
                        .then(data => data.json())
                        .then(data => {
                            if (data != null)
                                this.comment.score = data;
                        });
            },
            gradeRemove() {
                var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-comment-grade.php?type=comment&grade=<?php echo Defaults::SCORE_REMOVE ?>&userId=<?php echo AuthenticationService::getUserId() ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php echo $_SERVER["REQUEST_URI"] ?>';
                this.$http.get(url)
                        .then(data => data.json())
                        .then(data => {
                            if (data != null)
                                this.comment.score = data;
                        });
            }
        },
        data: function () {
            return {
                isLoading: false
            }
        }
    });
    var commentsSection = new Vue({
        el: '#commentsList',
        data: {
            comments: [],
            emptyMessage: "No comments found"
        }
    });
    var newEmptyComment = null;
    $(document).on("click", ".comments-dialog-opener", function () {
        var reviewId = $(this).attr("data-review-id");

        newEmptyComment = {
            userfullname: "<?php echo AuthenticationService::getFullName() ?>",
            id_user: <?php echo AuthenticationService::getUserId() ?>,
            id_review: Number(reviewId),
            date_comment: moment().tz("Europe/Rome").format(),
            score: 0
        };
        commentsSection.comments = [];
        $.getJSON("<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/get-comments.php?reviewId=" + reviewId, function (data) {
            commentsSection.comments = data || [];
            commentsSection.comments.push(newEmptyComment);
        });
    });
</script>
```