Base di Dati - Relazione

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

25 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.

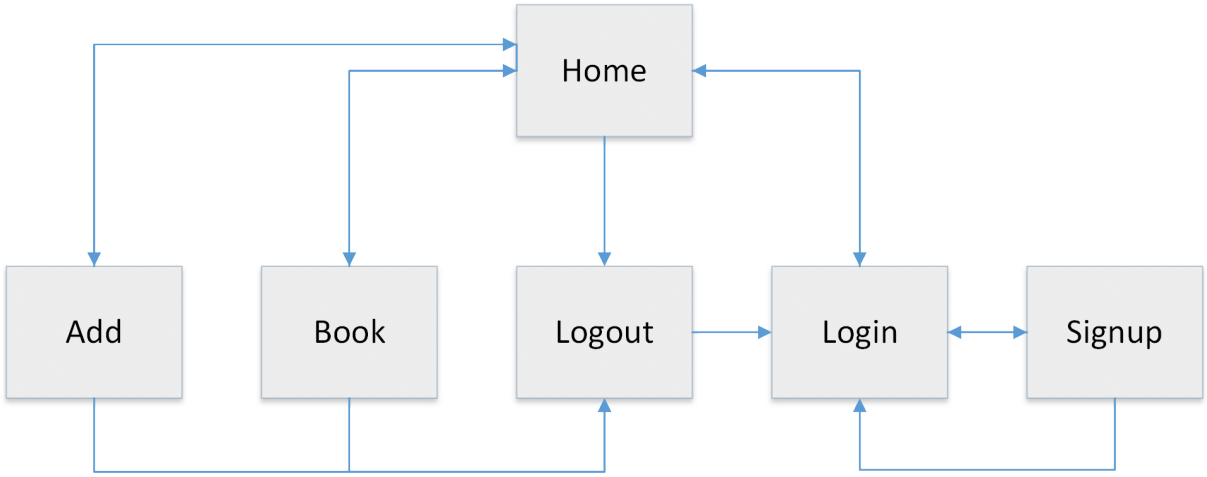

Figura 1: Struttura di navigazione del portale

4 Database

4.1 Modello concettuale

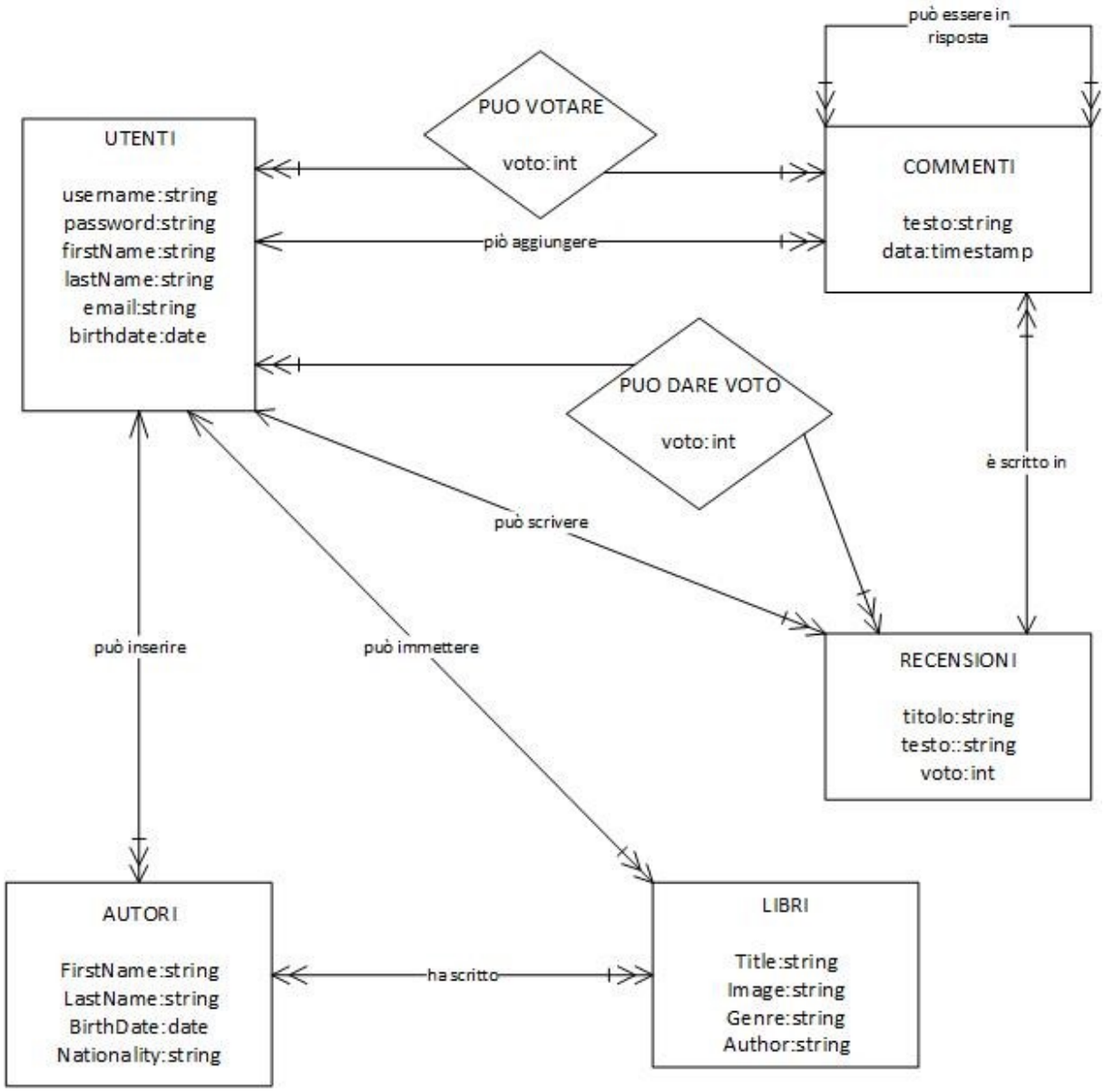

Figura 2: Schema concettuale del database dell'applicazione.

4.2 Modello relazionale

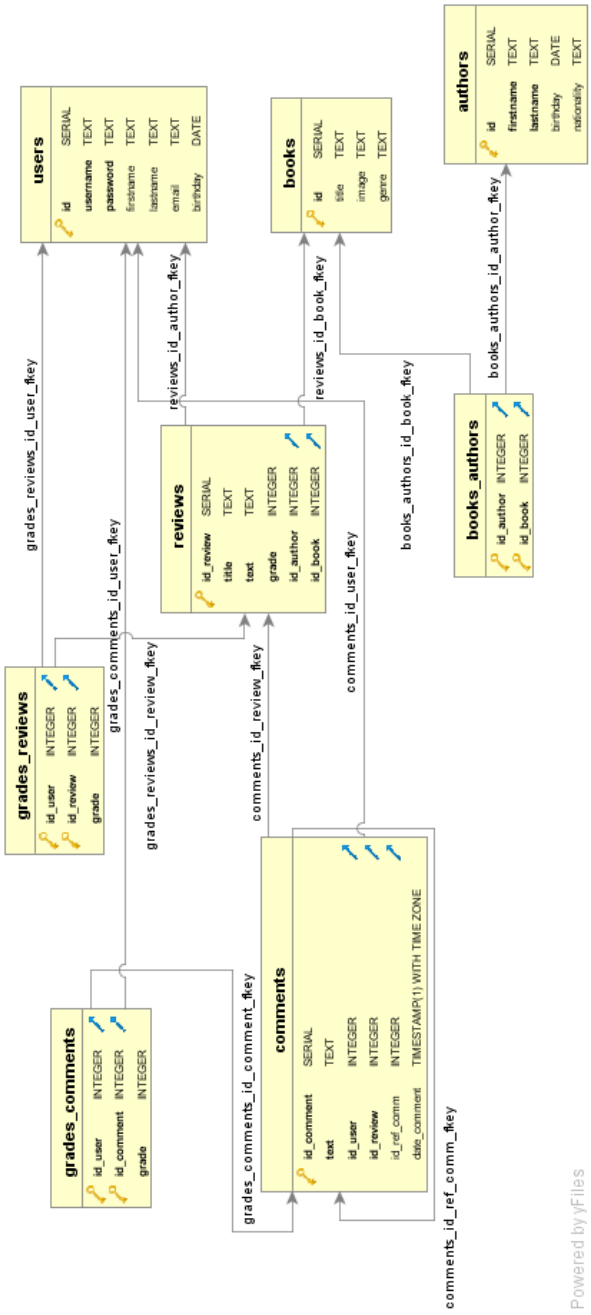

Figura 3: Schema relazionale del database dell'applicazione.

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

```

```
3       /**
4        * A Book author
5        */
6       class Author {
7
8           /** The author id
9            * @var type number */
10          public $id;
11
12          /**
13           * The author first name
14           * @var type string
15           */
16          public $firstname;
17
18          /**
19           * The author last name
20           * @var type string
21           */
22          public $lastname;
23
24          /**
25           * The author birthdate
26           * @var type Date
27           */
28          public $birthdate;
29
30          /**
31           * The author nationality
32           * @var type string
33           */
34          public $nationality;
35
36      }
37
38  ?>
```

5.1.3 Book.php

Rappresenta un libro scritto da un autore.

```
1   <?php
2
3       /**
4        * A Book
5        */
6       class Book {
7
8           /**
9            * The book identifier
10           * @var type number
11           */
12          public $id;
13
14          /**
15           * The book title
16           * @var type string
17           */
18          public $title;
19
20          /**
```

```
21          * The book image url
22          * @var type string
23          */
24         public $image;
25
26         /**
27          * The book genre
28          * @var type string
29          */
30         public $genre;
31
32         /**
33          * The book rating (number of stars)
34          * @var type float
35          */
36         public $rating;
37
38         /**
39          * The autho's book full name
40          * @var type string
41          */
42         public $author;
43
44         public function render($hasRate=false){
45             $this->rating = BooksService::getBookReviewSummary($this->id);
46             if (!$this->rating)
47                 $this->rating = new BookReviewSummary();
48             echo'<div class="row ">
49                 <div class="col-md-4">
50                     <img src="' . $this->image . '" height="200"/>
51                 </div>
52                 <div class="col-md-8">
53                     <div class="row">
54                         <div class="col-xs-12">
55                         ' . ($hasRate ? '<a href="./php/book.php?id=' . $this->id .
                                '"><h4>' . $this->title . '<small> ' . $this->
                                author . '</small></h4></a>' : '<h4>' . $this->title . '
                                <small> ' . $this->author . '</small></h4>') .
56                         '</div>
57                     </div>
58                     <div class="row">
59                         <div class="col-xs-12">
60                         <i>' . $this->genre . '</i>
61                         </div>
62                     </div>';
63                     if($hasRate){ echo'
64                             <div class="row">
65                                 <div class="col-xs-12">
66                                 <div class="rating-block">
67                                     <h4>Average user rating</h4>
68                                     <h2 class="bold padding-bottom-7">' . ($this->
                                        rating->avarage ? $this->rating->avarage : '
                                        0') . ' <small>/ 5</small></h2>';
69                                     for ($i = 1; $i <= 5; $i++) {
70                                     echo '
71                                         <button type="button" class="btn btn-
                                            warning btn-xs ' . (($i*1.0) > $this
                                            ->rating->avarage ? 'btn-grey' : '')
                                             . '" aria-label="Left Align">
72                                             <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
         */
```

```
42          public $id_author;
43
44          /**
45           * The review author full name
46           * @var string
47           */
48          public $author;
49
50          /**
51           * The review book's id
52           * @var integer
53           */
54          public $id_book;
55
56          /**
57           * Constructor
58           * @param integer $bookId The option book id (for the form rendering)
59           */
60          public function __construct($bookId = null) {
61              if ($bookId)
62                  $this->id_book = $bookId;
63          }
64
65          /**
66           * Renders the template of the current element
67           */
68          public function render() {
69              echo '
70                  <div class="review-block">
71                  <span>' . $this->author . '</span>
72                  <div class="row">
73                      <div class="col-sm-12">
74                              <div class="review-block-rate">';
75              for ($i = 1; $i <= 5; $i++) {
76                  echo '
77                      <button type="button" class="btn btn-warning btn-xs ' . ($i >
                            $this->grade ? 'btn-grey' : '') . '" aria-label="Left Align
                            ">
78                          <span class="fa fa-star" aria-hidden="true"></span>
79                      </button>
80                  ';
81              }
82              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>
83                    <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>
84                    <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>
85                    <a class="btn btn btn-outline-primary">' . $this->score . '</a>
86                    </div>
```

```
87                              <div class="review-block-title">' . $this->title . '</
                                    div>
88                              <div class="review-block-description">' . $this->text .
                                    '</div>
89
90                              <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
91                              </a>
92                          </div>
93                      </div>
94                      </div>
95                  ';
96          }
97
98          public function renderForm() {
99              echo '
100                 <form action="' . Defaults::DEFAULT_BASE_URL . '/php/rest/add-review
                        .php" method="post" name="AddReviewForm">
101                     <div class="review-block" style="min-width: 100%">
102                         <span>' . AuthenticationService::getFullName() . '</span
                                >
103                         <div class="row">
104                             <div class="col-sm-12">
105                                 <div class="review-block-rate">
106                                     <button type="button" class="btn btn-warning
                                            btn-xs" id="star1" onclick="setStar(1)
                                        ">
107                                         <span class="fa fa-star" aria-hidden="
                                                true"></span>
108                                     </button>
109                                     <button type="button" class="btn btn-warning
                                            btn-xs" id="star2" onclick="setStar(2)
                                        ">
110                                         <span class="fa fa-star" aria-hidden="
                                                true"></span>
111                                     </button>
112                                     <button type="button" class="btn btn-warning
                                            btn-xs" id="star3" onclick="setStar(3)
                                        ">
113                                         <span class="fa fa-star" aria-hidden="
                                                true"></span>
114                                     </button>
115                                     <button type="button" class="btn btn-warning
                                            btn-xs" id="star4" onclick="setStar(4)
                                        ">
116                                         <span class="fa fa-star" aria-hidden="
                                                true"></span>
117                                     </button>
118                                     <button type="button" class="btn btn-warning
                                            btn-xs" id="star5" onclick="setStar(5)
                                        ">
119                                         <span class="fa fa-star" aria-hidden="
                                                true"></span>
120                                     </button>
121                                 </div>
122                                 <div class="review-block-title">
123                                     <input type="text" placeholder="Title" name
                                            ="Title" class="form-control">
124
```

```
                                    </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;
```

```
19
20          /**
21           * The comment score
22           * @var integer
23           */
24          public $score = 0;
25
26          /**
27           * The user identifier
28           * @var integer
29           */
30          public $id_user;
31
32          /**
33           * The user full name
34           * @var string
35           */
36          public $userfullname;
37
38          /**
39           * The review id
40           * @var integer
41           */
42          public $id_review;
43
44          /**
45           * The parent comment identifier
46           * @var integer
47           */
48          public $id_ref_comm;
49
50          /**
51           * The comment datetime
52           * @var string
53           */
54          public $date_comment;
55
56
57          /**
58           * The children comment of the current comment
59           * @var Array<Comment>
60           */
61          public $children;
62
63      }
64
65  ?>
```

5.1.6 BookReviewSummary.php

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

```
1   <?php
2
3       /**
4        * A Book review summary
5        */
6       class BookReviewSummary {
7
8           /**
9            * The total numer of reviews
```

```
10          * @var integer
11          */
12         public $total;
13 
14         /**
15          * The avarage value
16          * @var float
17          */
18         public $avarage;
19 
20         /**
21          * The number of reviews with one star
22          * @var integer
23          */
24         public $oneStar;
25 
26         /**
27          * The number of reviews with two star
28          * @var integer
29          */
30         public $twoStar;
31 
32         /**
33          * The number of reviews three two star
34          * @var integer
35          */
36         public $threeStar;
37 
38         /**
39          * The number of reviews with four star
40          * @var integer
41          */
42         public $fourStar;
43 
44         /**
45          * The number of reviews with five star
46          * @var integer
47          */
48         public $fiveStar;
49 
50         /**
51          * Renders the HTML for the current item
52          */
53         public function render() {
54             $currentTotal = $this->total > 0 ? $this->total : 1;
55 
56             echo '
57                 <div class="row">
58                     <div class="col-sm-6">
59                         <div class="rating-block">
60                             <h4>Average user rating</h4>
61                             <h2 class="bold padding-bottom-7">' . ($this->avarage ?
                                   $this->avarage : '0') . ' <small>/ 5</small></h2>';
62                             for ($i = 1; $i <= 5; $i++) {
63                                 echo '
64                                     <button type="button" class="btn btn-warning btn
                                           -xs ' . (($i*1.0) > $this->avarage ? 'btn-
                                           grey' : '') . '" aria-label="Left Align">
65                                         <span class="fa fa-star" aria-hidden="true
                                               "></span>
66                                     </button>
```

```
67                                  ';
68                              }
69                              echo'
70                          </div>
71                      </div>
72                      <!-- STARS -->
73                      <div class="col-sm-6 rating-block">
74                          <h4>Rating breakdown</h4>
75                          <div class="pull-left">
76                              <div class="pull-left" style="width:35px; line-height
                                    :1;">
77                                  <div style="height:9px; margin:5px 0;">5 <span class
                                        ="fa fa-star"></span></div>
78                              </div>
79                              <div class="pull-left" style="width:180px;">
80                                  <div class="progress" style="height:9px; margin:8px
                                        0;">
81                                      <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>
82                                  </div>
83                              </div>
84                              <div class="pull-right" style="margin-left:10px;">' .
                                    $this->fiveStar . '</div>
85                          </div>
86
87
88                          <div class="pull-left">
89                              <div class="pull-left" style="width:35px; line-height
                                    :1;">
90                                  <div style="height:9px; margin:5px 0;">4 <span class
                                        ="fa fa-star"></span></div>
91                              </div>
92                              <div class="pull-left" style="width:180px;">
93                                  <div class="progress" style="height:9px; margin:8px
                                        0;">
94                                      <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>
95                                  </div>
96                              </div>
97                              <div class="pull-right" style="margin-left:10px;">' .
                                    $this->fourStar . '</div>
98                          </div>
99
100                         <div class="pull-left">
101                             <div class="pull-left" style="width:35px; line-height
                                    :1;">
102                                 <div style="height:9px; margin:5px 0;">3 <span class
                                        ="fa fa-star"></span></div>
103                             </div>
104                             <div class="pull-left" style="width:180px;">
105                                 <div class="progress" style="height:9px; margin:8px
                                        0;">
106                                     <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>
107                                 </div>
108                             </div>
109                             <div class="pull-right" style="margin-left:10px;">' .
                                    $this->threeStar . '</div>
110                         </div>
111
112                         <div class="pull-left">
113                             <div class="pull-left" style="width:35px; line-height
                                    :1;">
114                                 <div style="height:9px; margin:5px 0;">2 <span class
                                        ="fa fa-star"></span></div>
115                             </div>
116                             <div class="pull-left" style="width:180px;">
117                                 <div class="progress" style="height:9px; margin:8px
                                        0;">
118                                     <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>
119                                 </div>
120                             </div>
121                             <div class="pull-right" style="margin-left:10px;">' .
                                    $this->twoStar . '</div>
122                         </div>
123
124                         <div class="pull-left">
125                             <div class="pull-left" style="width:35px; line-height
                                    :1;">
126                                 <div style="height:9px; margin:5px 0;">1 <span class
                                        ="fa fa-star"></span></div>
127                             </div>
128                             <div class="pull-left" style="width:180px;">
129                                 <div class="progress" style="height:9px; margin:8px
                                        0;">
130                                     <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>
131                                 </div>
132                             </div>
133                             <div class="pull-right" style="margin-left:10px;">' .
                                    $this->oneStar . '</div>
134                         </div>
135
136                     </div>
137                 </div>
138             ';
139         }
140
141     }
142
143 ?>
```

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 settings.json

Contiene le informazioni per la connessione al database.

```
{
    "Debug": {
        "Database": {
            "Url": "localhost",
            "Port": 5432,
            "Name": "orsini",
            "User": "postgres",
            "Password": "root"
        }
    },
    "Stage": {
        "Database": {
            "Url": "dblab.dsi.unive.it",
            "Port": 5432,
            "Name": "a2016u104",
            "User": "a2016u104",
            "Password": "XPhVUqk6"
        }
    }
}
```

5.2.2 Database.php

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

```
<?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");
```

```
28              $configs = json_decode($settings, true);
29              if ($this->isDebug())
30                  $this->dbConfig = $configs["Debug"]["Database"];
31              else
32                  $this->dbConfig = $configs["Stage"]["Database"];
33          }
34
35          /**
36           * Returns the databse instance
37           * @return Database
38           */
39          public static function getInstance() {
40              if (Database::$instance == null)
41                  Database::$instance = new Database();
42              return Database::$instance;
43          }
44
45          /**
46           * Returns a new connection
47           * @return Connection
48           */
49          public function getConnection() {
50              $connectionString = 'pgsql:host=' . $this->dbConfig["Url"] . ';port=' .
                    $this->dbConfig["Port"] . ';dbname=' . $this->dbConfig["Name"];
51              $connection = new PDO($connectionString, $this->dbConfig["User"], $this
                    ->dbConfig["Password"]);
52              $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
53              return $connection;
54          }
55
56          /**
57           * Return true if the current REMOTE_ADDRESS is local
58           * @return boolean
59           */
60          private function isDebug() {
61              $whitelist = array('127.0.0.1', '::1', 'localhost');
62              return in_array($_SERVER['REMOTE_ADDR'], $whitelist);
63          }
64
65      }
66
67  ?>
```

5.2.3 Defaults.php

Contiene tutte le stringhe di default dell'applicazione.

```
1   <?php
2
3       /**
4        * A static class with all default values used all across the application
5        */
6       class Defaults {
7
8           /** Default ASC mode */
9           const ASC = 'asc';
10
11          /** Defualt DESC mode */
12          const DESC = 'desc';
13
14          /** Server default base starting url */
```

```
15          const DEFAULT_BASE_URL = '/bd2017';
16
17          /**
18           * The file where exception will be logged in. Relative path from web root.
19           */
20          const LOG_EXCEPTION_FILE = '/php/logs/exceptions.txt';
21
22          /**
23           * The score remove value
24           */
25          const SCORE_REMOVE = 0;
26
27          /**
28           * The score up value
29           */
30          const SCORE_UP = 1;
31
32          /**
33           * The score down value
34           */
35          const SCORE_DOWN = -1;
36
37      }
38
39  ?>
```

5.2.4 LogsService.php

Gestisce i log delle eccezioni dell'applicazione.

```
1   <?php
2       require(__DIR__ . '/Defaults.php');
3
4       /**
5        * Provides methods to log information or expection in a log file
6        */
7       class LogsService {
8
9           /**
10           * Append the given string in the specified file
11           * @param string $filePath The file path
12           * @param string $contentToAppend The string to append
13           */
14          private static function log($filePath, $contentToAppend) {
15              // Write the contents to the file,
16              // using the FILE_APPEND flag to append the content to the end of the
                    file
17              // and the LOCK_EX flag to prevent anyone else writing to the file at
                    the same time
18              file_put_contents($filePath, $contentToAppend, FILE_APPEND | LOCK_EX);
19          }
20
21          /**
22           * Logs in the 'logs/exceptions.txt' the given excpetion object
23           * @param Object $exceptionObj
24           */
25          public static function logException($exceptionObj) {
26              $now = new DateTime();
27              $str = "---------------\n";
28              $str .= $now->format('Y-m-d H:i:s');
29              $str .= "\n";
```

```
            $str .= $exceptionObj->__toString();
            $str .= "\n--------------\n";
            self::log(__DIR__ . "/../../" . Defaults::LOG_EXCEPTION_FILE, $str);
        }

    }

?>
```

5.2.5 AuthenticationService.php

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

```
<?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
```

```
47                  unset($_SESSION["Username"]);
48                  unset($_SESSION["UserId"]);
49                  unset($_SESSION["UserFullName"]);
50                  unset($_SESSION["UserFirstName"]);
51                  return false;
52              } catch (PDOException $e) {
53                  LogsService::logException($e);
54                  return false;
55              }
56          }
57
58          /**
59           * Returns true if a session username exisits, false othwrwise.
60           * @return boolean
61           */
62          public static function isLoggedIn() {
63              return (isset($_SESSION["Username"]) && $_SESSION["Username"] != null) ?
                     true : false;
64          }
65
66          /**
67           * Returns the current logged in user username or null
68           * @return string
69           */
70          public static function getUsername() {
71              return self::isLoggedIn() ? $_SESSION["Username"] : null;
72          }
73
74          /**
75           * Returns the current logged in user full name or null
76           * @return string
77           */
78          public static function getFullName() {
79              return self::isLoggedIn() ? $_SESSION["UserFullName"] : null;
80          }
81
82          /**
83           * Returns the current logged in user first name or null
84           * @return string
85           */
86          public static function getFirstName() {
87              return self::isLoggedIn() ? $_SESSION["UserFirstName"] : null;
88          }
89
90          /**
91           * Returns the current logged in user id or null
92           * @return integer
93           */
94          public static function getUserId() {
95              return self::isLoggedIn() ? $_SESSION["UserId"] : null;
96          }
97
98          /**
99           * Register a new user
100          * @param string username The user username
101          * @param string password The user password (in MD5)
102          * @param string firstName The user name
103          * @param string lastName The user surname
104          * @param string email The user email
105          * @param string birthdate The user birthdate
106          * @return boolean
```

```
107             *
108             */
109         public static function register($username, $password, $firstName, $lastName,
                $email, $birthdate) {
110             try {
111                 $dbconn = Database::getInstance()->getConnection();
112                 $statement = $dbconn->prepare('INSERT INTO users VALUES (
113             DEFAULT,
114             :username,
115             :password,
116             :firstName,
117             :lastName,
118             :email,
119             :birthdate
120         )');
121                 $statement->bindParam(":username", $username, PDO::PARAM_STR);
122                 $statement->bindParam(":password", $password, PDO::PARAM_STR);
123                 $statement->bindParam(":firstName", $firstName, PDO::PARAM_STR);
124                 $statement->bindParam(":lastName", $lastName, PDO::PARAM_STR);
125                 $statement->bindParam(":email", $email, PDO::PARAM_STR);
126                 $statement->bindParam(":birthdate", $birthdate, PDO::PARAM_STR);
127                 $statement->execute();
128                 return $statement->rowCount() == 1;
129             } catch (PDOException $e) {
130                 LogsService::logException($e);
131                 return false;
132             }
133         }
134 
135     }
136 
137 ?>
```

5.2.6 AuthorsService.php

Gestisce la CRUD per la tabella *authors*.

```
1  <?php
2  
3      require_once(__DIR__ . '/AuthenticationService.php');
4      require_once(__DIR__ . '/../models/Book.php');
5      require_once(__DIR__ . '/../models/Author.php');
6  
7      /**
8       * Provides all Author CRUD operations.
9       */
10     class AuthorsService {
11 
12         /**
13          * Adds a new author
14          * @param string firstName The author first name
15          * @param string lastName The author last name
16          * @param string birhDate The author birth date
17          * @param string nationality The author nationality (string)
18          * @return The new added author id
19          */
20         public static function addAuthor($firstName, $lastName, $birhDate,
                $nationality) {
21             try {
22                 if (!$birhDate || strlen($birhDate))
23                     $birhDate = null;
```

```
24                  if (!$nationality || strlen($nationality))
25                      $nationality = null;
26
27                  $dbconn = Database::getInstance()->getConnection();
28                  $authorStatment = $dbconn->prepare('
29                  INSERT INTO authors VALUES (
30                      DEFAULT,
31                      :firstName,
32                      :lastName,
33                      :birhDate,
34                      :nationality
35                  )
36              ');
37                  $authorStatment->bindParam(":firstName", $firstName, PDO::PARAM_STR)
                        ;
38                  $authorStatment->bindParam(":lastName", $lastName, PDO::PARAM_STR);
39                  $authorStatment->bindParam(":birhDate", $birhDate, PDO::PARAM_STR);
40                  $authorStatment->bindParam(":nationality", $nationality, PDO::
                        PARAM_STR);
41                  $authorStatment->execute();
42                  $isAdded = $authorStatment->rowCount() == 1;
43                  if ($isAdded)
44                      return $dbconn->lastInsertId("authors_id_seq");
45                  else
46                      return 0;
47              } catch (PDOException $e) {
48                  LogsService::logException($e);
49                  return 0;
50              }
51          }
52
53          /**
54           * Returns all the authors
55           * @return Array<Author>
56           */
57          public static function getAuthors() {
58              try {
59                  $dbconn = Database::getInstance()->getConnection();
60                  $statement = $dbconn->prepare('
61                  SELECT
62                      *
63                  FROM
64                      authors
65                  '
66                  );
67                  $statement->execute();
68                  return $statement->fetchAll(PDO::FETCH_CLASS, "Author");
69              } catch (PDOException $e) {
70                  LogsService::logException($e);
71                  return null;
72              }
73          }
74
75          /**
76           * Deletes an Author given its id
77           * @param $authorId integer
78           * @return boolean
79           */
80          public static function deleteAuthor($authorId) {
81              if ($authorId == null || $authorId < 0) {
82                  return false;
```

```
83              }
84              try {
85                  $dbconn = Database::getInstance()->getConnection();
86                  $query = '
87                          DELETE
88                          FROM
89                              authors
90                          WHERE
91                              id = :$authorId
92                      ';
93
94                  $statement = $dbconn->prepare($query);
95                  $statement->bindParam(":$authorId", $authorId, PDO::PARAM_INT);
96                  $result = $statement->execute();
97                  return $result->rowCount() == 1;
98              } catch (PDOException $e) {
99                  LogsService::logException($e);
100                 return null;
101             }
102         }
103
104     }
105
106 ?>
```

5.2.7 BooksService.php

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

```
1  <?php
2
3      require_once(__DIR__ . '/AuthenticationService.php');
4      require_once(__DIR__ . '/../models/Book.php');
5      require_once(__DIR__ . '/../models/Author.php');
6      require_once(__DIR__ . '/../models/Review.php');
7      require_once(__DIR__ . '/../models/BookReviewSummary.php');
8
9      /**
10      * Provides all Book CRUD operations.
11      */
12     class BooksService {
13
14         /**
15          * Adds a new book
16          * @param string title The book title
17          * @param string image The book image url
18          * @param string genre The book genre
19          * @param integer authorId The book authorId
20          * @return The new added book id
21          */
22         public static function addBook($title, $image, $genre, $authorId) {
23             try {
24                 $dbconn = Database::getInstance()->getConnection();
25                 $statement = $dbconn->prepare('
26                 INSERT INTO books VALUES (
27                     DEFAULT,
28                     :title,
29                     :image,
30                     :genre
31                 )
32             ');
```

```
33                  $statement->bindParam(":title", $title, PDO::PARAM_STR);
34                  $statement->bindParam(":image", $image, PDO::PARAM_STR);
35                  $statement->bindParam(":genre", $genre, PDO::PARAM_STR);
36                  $statement->execute();
37                  $isBookAdded = $statement->rowCount() == 1;
38
39                  if ($isBookAdded) {
40                      $statement = $dbconn->prepare('
41                      INSERT INTO books_authors VALUES (
42                          :authorId,
43                          :bookId
44                      )
45                  ');
46                      $bookId = $dbconn->lastInsertId("books_id_seq");
47                      $statement->bindParam(":authorId", $authorId, PDO::PARAM_INT);
48                      $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
49                      $statement->execute();
50                      $isLinkAdded = $statement->rowCount() == 1;
51                  }
52
53                  return $isBookAdded && $isLinkAdded;
54              } catch (PDOException $e) {
55                  LogsService::logException($e);
56                  return false;
57              }
58          }
59
60          /**
61           * Adds a new review
62           * @param string $title
63           * @param string $text
64           * @param integer $grade
65           * @param integer $IdAuthor
66           * @param integer $idBook
67           * @return boolean
68           */
69          public static function addReview($title, $text, $grade, $IdAuthor, $idBook)
                {
70              try {
71                  $dbconn = Database::getInstance()->getConnection();
72                  $statement = $dbconn->prepare('
73                  INSERT INTO reviews VALUES (
74                          DEFAULT,
75                          :title,
76                          :text,
77                          :grade,
78                          0,
79                          :idAuthor,
80                          :idBook
81                      )
82                  ');
83                  $statement->bindParam(":title", $title, PDO::PARAM_STR);
84                  $statement->bindParam(":text", $text, PDO::PARAM_STR);
85                  $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
86                  $statement->bindParam(":idAuthor", $IdAuthor, PDO::PARAM_INT);
87                  $statement->bindParam(":idBook", $idBook, PDO::PARAM_INT);
88                  $statement->execute();
89                  return $statement->rowCount() == 1;
90              } catch (PDOException $e) {
91                  LogsService::logException($e);
92                  return false;
```

```
93              }
94          }
95
96          /**
97           * Adds a review grade (+1/-1)
98           * @param integer $reviewId The review id
99           * @param integer $userId The user id
100          * @param integer $grade The grade
101          * @return boolean
102          */
103         public static function addReviewGrade($reviewId, $userId, $grade) {
104             try {
105                 $dbconn = Database::getInstance()->getConnection();
106
107                 $statement = $dbconn->prepare('
108                 SELECT *
109                 FROM
110                     grades_reviews
111                 WHERE
112                         id_user = :userId
113                         AND
114                         id_review = :reviewId
115                 ');
116                 $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
117                 $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
118                 $statement->execute();
119                 $isExisting = $statement->rowCount() == 1;
120
121                 if ($isExisting) {
122                     $currentValue = $statement->fetchAll()[0]["grade"];
123                     $statement = $dbconn->prepare('
124                         UPDATE
125                             grades_reviews
126                         SET
127                             grade = :grade
128                         WHERE
129                                 id_user = :userId
130                                 AND
131                                 id_review = :reviewId
132                     ');
133                     $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
134                     $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
135                     $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
136                     $statement->execute();
137                     return $statement->rowCount() == 1;
138                 }
139
140                 // ELSE NEW GRADE
141                 $statement = $dbconn->prepare('
142                 INSERT INTO grades_reviews VALUES (
143                         :userId,
144                         :reviewId,
145                         :grade
146                     )
147                 ');
148                 $currentScore = $grade == Defaults::SCORE_DOWN ? -1 : 1;
149                 $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
150                 $statement->bindParam(":reviewId", $reviewId, PDO::PARAM_INT);
151                 $statement->bindParam(":grade", $currentScore, PDO::PARAM_INT);
152                 $statement->execute();
153                 return $statement->rowCount() == 1;
```

```
154             } catch (PDOException $e) {
155                 LogsService::logException($e);
156                 return false;
157             }
158         }
159 
160         /**
161          * Given a word returns the book's title (one or more) that contains that
                word.
162          * @param string keyword The keywork to search
163          * @return Array<Book>
164          */
165         public static function research($keyword = null, $sort = Defaults::DESC) {
166             try {
167                 if (!isset($keyword))
168                     $keyword = null;
169                 if (!isset($sort))
170                     $sort = Defaults::DESC;
171 
172                 $dbconn = Database::getInstance()->getConnection();
173                 $key = ($keyword ? $keyword : '');
174                 $query = '
175                 SELECT
176                     b.id,
177                     b.title,
178                     b.image,
179                     b.genre,
180                     AVG(r.grade) AS rate,
181                     concat_ws(\'\', a.firstName, \' \', a.lastname) as
                        authorfullname
182                 FROM
183                     books AS b JOIN books_authors AS ba
184                     ON ba.id_book= b.id JOIN authors AS a ON a.id= ba.id_author
185                     LEFT JOIN reviews AS r ON r.id_book=b.id
186                 WHERE
187                     lower(b.title) LIKE lower(:key)
188                         OR
189                     lower(concat_ws(\'\', a.firstName, \' \', a.lastname)) LIKE
                        lower(:key)
190                         OR
191                     lower(b.genre) LIKE lower(:key)
192                 GROUP BY
193                     b.id,
194                     b.title,
195                     b.image,
196                     b.genre,
197                     authorfullname
198                 ORDER BY
199                     rate '. $sort . ' NULLS LAST';
200                 $statement = $dbconn->prepare($query);
201                 $statement->bindValue(":key", '%' . $keyword . '%');
202                 $row = $statement->execute();
203 
204                 $books = array();
205                 while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
206                     $book = new Book();
207                     $book->id = $row['id'];
208                     $book->title = $row['title'];
209                     $book->image = $row['image'];
210                     $book->genre = $row['genre'];
211                     $book->author = $row['authorfullname'];
```

```
212                     $book->rating = 0;
213                     $books[] = $book;
214                 }
215                 return $books;
216             } catch (PDOException $e) {
217                 LogsService::logException($e);
218                 return null;
219             }
220         }
221 
222         /**
223          * Deletes a Book given its id
224          * @param integer $bookId integer
225          * @return boolean
226          */
227         public static function deleteBook($bookId) {
228             if ($bookId == null || $bookId < 0) {
229                 return false;
230             }
231             try {
232                 $dbconn = Database::getInstance()->getConnection();
233                 $query = '
234                         DELETE
235                         FROM
236                             books
237                         WHERE
238                             id = :bookId
239                     ';
240 
241                 $statement = $dbconn->prepare($query);
242                 $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
243                 $result = $statement->execute();
244                 return $result->rowCount() == 1;
245             } catch (PDOException $e) {
246                 LogsService::logException($e);
247                 return null;
248             }
249         }
250 
251         /**
252          * Gets a Book given its id
253          * @param $bookId integer
254          * @return Book
255          */
256         public static function getBook($bookId) {
257             if ($bookId == null || $bookId < 0) {
258                 return null;
259             }
260             try {
261                 $dbconn = Database::getInstance()->getConnection();
262                 $query = '
263                         SELECT
264                             b.*,
265                             concat_ws(\'\', a.firstname, \' \', a.lastname) as
                                author
266                         FROM books b JOIN books_authors ba ON ba.id_book = b.id
267                         JOIN authors a ON ba.id_author = a.id
268                         WHERE
269                             b.id = :bookId
270                     ';
271 
```

```
272                 $statement = $dbconn->prepare($query);
273                 $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
274                 $statement->execute();
275                 $books = $statement->fetchAll(PDO::FETCH_CLASS, "Book");
276 
277                 if (count($books) > 0)
278                     return $books[0];
279 
280                 return null;
281             } catch (PDOException $e) {
282                 LogsService::logException($e);
283                 return null;
284             }
285         }
286 
287         /**
288          *
289          * Return all the reviews of the given book
290          * @param $bookId integer The book id
291          * @return Array<Review>
292          */
293         public static function getReviews($bookId) {
294             if ($bookId == null || $bookId < 0) {
295                 return null;
296             }
297             try {
298                 $dbconn = Database::getInstance()->getConnection();
299                 $query = '
300                         SELECT
301                             reviews.*,
302                             concat_ws(\'\', users.firstname, \' \', users.lastname)
                                    as author,
303                             SUM(coalesce(grades_reviews.grade,0)) as score
304                         FROM
305                             reviews JOIN users ON reviews.id_author = users.id
306                             LEFT JOIN grades_reviews ON reviews.id_review =
                                    grades_reviews.id_review
307                         WHERE
308                             id_book = :bookId
309                         GROUP BY reviews.id_review,author
310                     ';
311 
312                 $statement = $dbconn->prepare($query);
313                 $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
314                 $statement->execute();
315                 $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "Review");
316 
317                 return $reviews;
318             } catch (PDOException $e) {
319                 LogsService::logException($e);
320                 return null;
321             }
322         }
323 
324         /**
325          * Returns true if the user has already done a review on the specified book
326          * @param integer $authorId The author id
327          * @param integer $bookId The book id
328          * @return boolean
329          */
330         public static function hasReview($authorId, $bookId) {
```

```
331             if ($bookId == null || $bookId < 0 || $authorId == null || $authorId <
                    0) {
332                 return null;
333             }
334             try {
335                 $dbconn = Database::getInstance()->getConnection();
336                 $query = '
337                         SELECT
338                             *
339                         FROM
340                             reviews JOIN users ON reviews.id_author = users.id
341                         WHERE
342                             id_book = :bookId
343                                 AND
344                             id_author = :authorId
345                     ';
346 
347                 $statement = $dbconn->prepare($query);
348                 $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
349                 $statement->bindParam(":authorId", $authorId, PDO::PARAM_INT);
350                 $statement->execute();
351                 return $statement->rowCount() == 1;
352             } catch (PDOException $e) {
353                 LogsService::logException($e);
354                 return null;
355             }
356         }
357 
358         /**
359          *
360          * Return the book's reviews summary
361          * @param $bookId integer The book id
362          * @return BookReviewSummary
363          */
364         public static function getBookReviewSummary($bookId) {
365             if ($bookId == null || $bookId < 0) {
366                 return null;
367             }
368             try {
369                 $dbconn = Database::getInstance()->getConnection();
370                 $query = '
371                         SELECT
372                             COUNT(*) AS total,
373                             ROUND(AVG(grade),1) AS avarage,
374                             COUNT(CASE WHEN grade = 1 THEN grade END) AS "oneStar",
375                             COUNT(CASE WHEN grade = 2 THEN grade END) AS "twoStar",
376                             COUNT(CASE WHEN grade = 3 THEN grade END) AS "threeStar
                                    ",
377                             COUNT(CASE WHEN grade = 4 THEN grade END) AS "fourStar",
378                             COUNT(CASE WHEN grade = 5 THEN grade END) AS "fiveStar"
379                         FROM
380                             reviews
381                         WHERE
382                             reviews.id_book = :bookId
383                     ';
384 
385                 $statement = $dbconn->prepare($query);
386                 $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
387                 $statement->execute();
388                 $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "BookReviewSummary
                        ");
```

```
389
390                 if (count($reviews) > 0)
391                     return $reviews[0];
392
393                 return null;
394             } catch (PDOException $e) {
395                 LogsService::logException($e);
396                 return null;
397             }
398         }
399
400     }
401
402 ?>
```

5.2.8 CommentsService.php

Gestisce la CRUD per la tabella *comments*.

```
1  <?php
2
3      require_once(__DIR__ . '/AuthenticationService.php');
4      require_once(__DIR__ . '/../models/Comment.php');
5
6      /**
7       * Provides all Comments CRUD operations.
8       */
9      class CommentsService {
10
11         /**
12          * Returns all the comments of a review (in a hierarcic tree)
13          * @return Array<Comment>
14          */
15         public static function getCommentsByReviewId($reviewId) {
16             try {
17                 $dbconn = Database::getInstance()->getConnection();
18                 $statement = $dbconn->prepare('
19                     SELECT
20                         c.*,
21                         COALESCE(SUM(gc.grade),0) as score,
22                         concat_ws(\'\', u.firstName, \' \', u.lastname) as
                               userfullname
23                     FROM
24                         comments c JOIN users u ON c.id_user = u.id
25                         LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
26                     WHERE id_review = :id_review AND c.id_ref_comm IS NULL
27                     GROUP BY c.id_comment, userfullname
28                     ORDER BY c.date_comment ASC
29                 ');
30                 $statement->bindParam(":id_review", $reviewId);
31                 $row = $statement->execute();
32                 $comments = array();
33                 while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
34                     $comment = new Comment();
35                     $comment->id_comment = $row['id_comment'];
36                     $comment->text = $row['text'];
37                     $comment->score = $row['score'];
38                     $comment->id_user = $row['id_user'];
39                     $comment->userfullname = $row['userfullname'];
40                     $comment->id_review = $row['id_review'];
41                     $comment->id_ref_comm = $row['id_ref_comm'];
```

```
42                      $comment->date_comment = $row['date_comment'];
43                      $comment->canEdit = $row['id_user'] == AuthenticationService::
                            getUserId();
44                      $comment->children = self::getCommentsByParentId($row['
                            id_comment']);
45                      $comments[] = $comment;
46                  }
47                  return $comments;
48              } catch (PDOException $e) {
49                  LogsService::logException($e);
50                  return null;
51              }
52          }
53
54          /**
55           * Add a new comment
56           * @param Comment $comment
57           * @return boolean
58           */
59          public static function addComment($comment) {
60              try {
61                  $dbconn = Database::getInstance()->getConnection();
62                  $statement = $dbconn->prepare('
63                  INSERT INTO COMMENTS VALUES (
64                      DEFAULT,
65                      :text,
66                      :id_user,
67                      :id_review,
68                      :id_ref_comm,
69                      :date_comment
70                  )'
71                  );
72                  $statement->bindParam(":text", $comment->text);
73                  $statement->bindParam(":id_user", $comment->id_user);
74                  $statement->bindParam(":id_review", $comment->id_review);
75                  $statement->bindParam(":id_ref_comm", $comment->id_ref_comm);
76                  $statement->bindParam(":date_comment", $comment->date_comment);
77                  $statement->execute();
78                  $isAdded = $statement->rowCount() == 1;
79                  if ($isAdded)
80                      return $dbconn->lastInsertId("comments_id_comment_seq");
81                  else
82                      return 0;
83              } catch (PDOException $e) {
84                  LogsService::logException($e);
85                  return null;
86              }
87          }
88
89          /**
90           * Deletes a comment and all its children, given its id
91           * @param integer $commentId
92           * @return boolean
93           */
94          public static function deleteAuthor($commentId) {
95              if ($commentId == null || $commentId < 0) {
96                  return false;
97              }
98              try {
99                  $dbconn = Database::getInstance()->getConnection();
100                 $query = '
```

```
101                         DELETE
102                         FROM
103                             comments
104                         WHERE
105                             id_comment = :commentId
106                     ';
107
108                 $statement = $dbconn->prepare($query);
109                 $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
110                 $result = $statement->execute();
111                 return $result->rowCount() == 1;
112             } catch (PDOException $e) {
113                 LogsService::logException($e);
114                 return null;
115             }
116         }
117
118         /**
119          * Given a parent id comment, return all the hierarchic tree
120          * @param integer $parentId
121          * @return Array<Comment>
122          */
123         public static function getCommentsByParentId($parentId) {
124             try {
125                 $dbconn = Database::getInstance()->getConnection();
126                 $statement = $dbconn->prepare('
127                     SELECT
128                         c.*,
129                         COALESCE(SUM(gc.grade),0) as score,
130                         concat_ws(\'\', u.firstName, \' \', u.lastname) as
                            userfullname
131                     FROM
132                         comments c JOIN users u ON c.id_user = u.id
133                         LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
134                     WHERE
135                         c.id_ref_comm = :parentId
136                     GROUP BY c.id_comment,userfullname
137                 ');
138                 $statement->bindParam(":parentId", $parentId);
139                 $row = $statement->execute();
140                 $comments = array();
141                 while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
142                     $comment = new Comment();
143                     $comment->id_comment = $row['id_comment'];
144                     $comment->text = $row['text'];
145                     $comment->score = $row['score'];
146                     $comment->id_user = $row['id_user'];
147                     $comment->userfullname = $row['userfullname'];
148                     $comment->id_review = $row['id_review'];
149                     $comment->id_ref_comm = $row['id_ref_comm'];
150                     $comment->date_comment = $row['date_comment'];
151                     $comment->canEdit = $row['id_user'] == AuthenticationService::
                            getUserId();
152                     $comment->children = self::getCommentsByParentId($row['
                            id_comment']);
153                     $comments[] = $comment;
154                 }
155                 return $comments;
156             } catch (PDOException $e) {
157                 LogsService::logException($e);
158                 return null;
```

```
159             }
160         }
161 
162         /**
163          * Adds a comment grade (+1/-1)
164          * @param integer $commentId The comment id
165          * @param integer $userId The user id
166          * @param integer $grade The grade
167          * @return boolean
168          */
169         public static function addCommentGrade($commentId, $userId, $grade) {
170             try {
171                 $dbconn = Database::getInstance()->getConnection();
172 
173                 $statement = $dbconn->prepare('
174                     SELECT
175                         *
176                     FROM
177                         grades_comments
178                     WHERE
179                             id_user = :userId
180                             AND
181                             id_comment = :commentId
182                 ');
183                 $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
184                 $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
185                 $statement->execute();
186                 $isExisting = $statement->rowCount() == 1;
187 
188                 if ($isExisting) {
189                     $currentValue = $statement->fetchAll()[0]["grade"];
190                     $statement = $dbconn->prepare('
191                         UPDATE
192                             grades_comments
193                         SET
194                             grade = :grade
195                         WHERE
196                                 id_user = :userId
197                                 AND
198                                 id_comment = :commentId
199                     ');
200                     $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
201                     $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
202                     $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
203                     $statement->execute();
204                     if ($statement->rowCount() == 1)
205                         return self::getScore($commentId);
206                     return null;
207                 }
208 
209                 // ELSE NEW GRADE
210                 $statement = $dbconn->prepare('
211                 INSERT INTO grades_comments VALUES (
212                         :userId,
213                         :commentId,
214                         :grade
215                     )
216                 ');
217                 $currentScore = $grade == Defaults::SCORE_DOWN ? -1 : 1;
218                 $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
219                 $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);
                $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
    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
```

```
8       if(isset($_POST["Username"]) &&
9           isset($_POST["Password"]) &&
10          isset($_POST["Email"]) &&
11          isset($_POST["FirstName"]) &&
12          isset($_POST["LastName"]) &&
13          isset($_POST["BirthDate"])){
14          $isRegistred = AuthenticationService::register($_POST["Username"], md5(
                $_POST["Password"]),$_POST["FirstName"],$_POST["LastName"],$_POST["Email
                "],$_POST["BirthDate"]);
15      }
16
17      if($isRegistred){
18          // return on the previous page
19          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/register.php?signup=
                true');
20      }else{
21           // return on theregister page with a GET param
22          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/register.php?errors=
                true');
23      }
24
25  ?>
```

5.3.2 login.php

Permette ad un utente di effettuare il login.

```
1   <?php
2       require_once('../services/AuthenticationService.php');
3       require_once('../services/Defaults.php');
4
5       $isLogged = false;
6
7       // if a username and a password is given in POST, then try to login
8       if(isset($_POST["Username"]) && isset($_POST["Password"])){
9           $isLogged = AuthenticationService::login($_POST["Username"], md5($_POST["
                Password"]));
10      }
11
12      if($isLogged){
13          // return on the previous page
14          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/login.php');
15      }else{
16           // return on the login page with a GET param
17          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/login.php?
                wrongCredentials=true');
18      }
19
20  ?>
```

5.3.3 logout.php

Permette ad un utente di effettuare il logout.

```
1   <?php
2       require_once('../services/AuthenticationService.php');
3
4       // if logged in, destroy the session
5       if(isset($_SESSION["Username"]))
6           unset($_SESSION["Username"]);
```

```
7
8       // return on the previous page
9       header('Location: ' . $_SERVER['HTTP_REFERER']);
10  ?>
```

5.3.4 add-author.php

Permette di aggiungere al database un nuovo autore di libri.

```
1   <?php
2       require_once('../services/AuthenticationService.php');
3       require_once('../services/AuthorsService.php');
4       require_once('../services/Defaults.php');
5
6       $authorId = false;
7
8       // if a username and a password is given in POST, then try to login
9       if(isset($_POST["FirstName"]) &&
10          isset($_POST["LastName"]) &&
11          isset($_POST["BirthDate"]) &&
12          isset($_POST["Nationality"])){
13          $authorId = AuthorsService::addAuthor($_POST["FirstName"], $_POST["LastName"
                ], $_POST["BirthDate"], $_POST["Nationality"]);
14      }
15
16      if($authorId != null && $authorId > 0){
17          // return on the previous page
18          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?added=
                true');
19      }else{
20           // return on the register page with a GET param
21          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?errors=
                true');
22      }
23
24  ?>
```

5.3.5 add-book.php

Permette di aggiungere al database un nuovo libro.

```
1   <?php
2       require_once('../services/AuthenticationService.php');
3       require_once('../services/BooksService.php');
4       require_once('../services/Defaults.php');
5
6       $isReviewAdded = false;
7
8       // if a username and a password is given in POST, then try to login
9       if(isset($_POST["Title"]) &&
10          isset($_POST["Image"]) &&
11          isset($_POST["Genre"]) &&
12          isset($_POST["Author"])){
13          $isReviewAdded = BooksService::addBook($_POST["Title"],$_POST["Image"],
                $_POST["Genre"],$_POST["Author"]);
14      }
15
16      if($isReviewAdded){
17          // return on the previous page
```

```
18          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?added=
                true');
19      }else{
20           // return on the register page with a GET param
21          header('Location: '. Defaults::DEFAULT_BASE_URL .'/php/add-book.php?errors=
                true');
22      }
23
24  ?>
```

5.3.6 add-review.php

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

```
1   <?php
2
3       require_once('../services/AuthenticationService.php');
4       require_once('../services/BooksService.php');
5       require_once('../services/Defaults.php');
6
7       $isReviewAdded = false;
8
9       // if a username and a password is given in POST, then try to login
10      if (isset($_POST["Title"]) &&
11              isset($_POST["Text"]) &&
12              isset($_POST["Grade"]) &&
13              isset($_POST["IdBook"]) &&
14              isset($_POST["PrevUrl"]) &&
15              isset($_POST["IdAuthor"])) {
16          $isReviewAdded = BooksService::addReview($_POST["Title"], $_POST["Text"],
                $_POST["Grade"], $_POST["IdAuthor"], $_POST["IdBook"]);
17      }
18
19      if ($isReviewAdded && isset($_POST["PrevUrl"])) {
20          header('Location: ' . $_POST["PrevUrl"]);
21      } else {
22          // return on the register page with a GET param
23          header('Location: ' . Defaults::DEFAULT_BASE_URL);
24      }
25  ?>
```

5.3.7 add-comment.php

Permette di aggiungere al database un nuovo commento.

```
1   <?php
2
3       require_once('../services/CommentsService.php');
4
5       $request_body = file_get_contents('php://input');
6       $comment = json_decode($request_body);
7       $result = null;
8       if ($comment) {
9           $result = CommentsService::addComment($comment);
10      }
11      echo json_encode($result);
12  ?>
```

5.3.8 add-grade.php

Permette di aggiungere un voto ad una recensione esistente.

```php
<?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
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|0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101
```

```
        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
    */
```

```
56      function checkCancelButton() {
57          var keyword = $("#keyword").val();
58          if (!keyword || keyword.length == 0)
59              $("#cancelButtonGroup").addClass("hidden");
60          else
61              $("#cancelButtonGroup").removeClass("hidden");
62      }
63
64      /**
65      * Navigates to the index page with the get keyword param valorized with the
            input value
66      */
67      function searchBook() {
68          // navigate to homepage with a get parameter
69          var keyword = $("#keyword").val();
70          var currentSort = "<?php echo isset($_GET["sort"]) ? $_GET["sort"] :
                Defaults::ASC ?>";
71          var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
72          // if no value is provided, navigate without the param
73          if (!keyword || keyword.length == 0)
74              window.location.href = defaultBaseUrl + currentSort ? '?sort=' +
                    currentSort : '';
75          else
76              window.location.href = defaultBaseUrl + "/?keyword=" + keyword  + (
                    currentSort ? '&sort=' + currentSort : '');
77      }
78
79      /*
80      * When click on sort change button in navbar
81      */
82      $(document).on("click","#bookSort",function(){
83          var currentSort = "<?php echo isset($_GET["sort"]) ? $_GET["sort"] :
                Defaults::DESC ?>";
84          var currentKeyword = "<?php echo isset($_GET["keyword"]) ? $_GET["keyword"]
                : null ?>";
85          var defaultAsc = "<?php echo Defaults::ASC; ?>";
86          var defaultDesc= "<?php echo Defaults::DESC; ?>";
87          var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
88
89          if (!currentKeyword || currentKeyword.length == 0)
90              window.location.href = defaultBaseUrl + currentSort ? '?sort=' + (
                    currentSort == defaultDesc ? defaultAsc : defaultDesc) : '';
91          else{
92              window.location.href = defaultBaseUrl + "/?keyword=" + currentKeyword  +
                     (currentSort ? '&sort=' + (currentSort == defaultDesc ? defaultAsc
                    : defaultDesc) : '');
93          }
94      });
95      // END NAVBAR SEARCH BOX
96  </script>
```

5.4.4 styles.php

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

```
1  <?php
2      require_once(__DIR__ . '/../services/Defaults.php');
3  ?>
4
5  <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>
```

```
30      </div>
31  </div>
```

5.4.6 login-form.php

Form di login.

```
1  <?php
2      require_once( __DIR__ . '/../services/Defaults.php');
3  ?>
4  <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/login.php'; ?>"
       method="post" name="LoginForm">
5      <h3>Login</h3>
6      <br>
7      <div class="form-group">
8          <input type="text" class="form-control form-control-sm" name="Username"
               placeholder="Username" required="" autofocus="" />
9      </div>
10     <div class="form-group">
11         <input type="password" class="form-control form-control-sm" name="Password"
               placeholder="Password" required=""/>
12     </div>
13     <button class="btn btn-sm btn-primary"  name="Submit" value="Login" type="Submit
           ">Sign in</button>
14 </form>
```

5.4.7 add-author-form.php

Form di creazione di un nuovo autore di libri.

```
1  <?php
2      require_once( __DIR__ . '/../services/Defaults.php');
3
4      if (!AuthenticationService::isLoggedIn()) {
5          die();
6      }
7  ?>
8  <div class="col-md-6">
9      <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/add-author.php
           '; ?>" method="post" name="AddAuthorForm">
10         <h3>Add author</h3>
11         <br>
12         <div class="form-group">
13             <input type="text" class="form-control form-control-sm" name="FirstName"
                   placeholder="First name" required="true" autocomplete="off">
14         </div>
15         <div class="form-group">
16             <input type="text" class="form-control form-control-sm" name="LastName"
                   placeholder="Last name" required="true" autocomplete="off"/>
17         </div>
18         <div class="form-group">
19             <input type="date" class="form-control form-control-sm" name="BirthDate"
                   placeholder="BirthDate" />
20         </div>
21         <div class="form-group">
22             <input type="text" class="form-control form-control-sm" name="
                   Nationality" placeholder="Nationality" />
23         </div>
24         <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');
```

```
4  ?>
5  <!-- Modal -->
6  <div class="modal fade" id="comments" tabindex="-1" role="dialog" aria-labelledby="
       myModalLabel">
7      <div class="modal-dialog modal-lg" role="document">
8          <div class="modal-content">
9              <div class="modal-header">
10                 <button type="button" class="close" data-dismiss="modal" aria-label=
                       "Close"><span aria-hidden="true">&times;</span></button>
11                 <h4 class="modal-title" id="myModalLabel">Comments</h4>
12             </div>
13             <div class="modal-body" id="commentsList">
14                 <!-- START COMMENTS -->
15
16                 <template v-if="comments.length">
17                     <comment
18                         v-for="comment in comments"
19                         v-bind:comment="comment"
20                         v-bind:level="0"
21                         v-bind:key="comment.id_comment">
22                     </comment>
23                 </template>
24                 <div class="alert alert-warning" v-else>
25                     {{emptyMessage}}
26                 </div>
27
28                 <!-- END COMMENTS -->
29             </div>
30             <div class="modal-footer">
31                 <button type="button" class="btn btn-outline-primary btn-sm"  data-
                       dismiss="modal">Close</button>
32             </div>
33         </div>
34     </div>
35 </div>
36 <script>
37     Vue.prototype.window = window;
38     Vue.http.headers.common['content-type'] = 'application/json';
39
40     Vue.component('comment', {
41         name: 'comment',
42         props: ["comment", "level"],
43         template: `
44                 <div>
45                     <div v-bind:class="'card comment-' + level">
46                         <div class="card-header">
47                                 {{comment.userfullname}}
48                                 <span v-if="comment.id_comment > 0">commented {{
                                       window.moment(comment.date_comment).tz("Europe/
                                       Rome").fromNow()}}</span>
49                                 <span v-else>aggiungi un nuovo commento</span>
50                                 <template v-if="comment.id_comment > 0">
51                                     <a class="btn btn-sm btn-outline-success" v-on:
                                           click="gradeUp"><i class="fa fa-thumbs-o-up"
                                           ></i></a>
52                                     <a class="btn btn-sm btn-outline-danger"  v-on:
                                           click="gradeDown"><i class="fa fa-thumbs-o-
                                           down"></i></a>
53                                     <a class="btn btn-sm btn-outline-primary" v-on:
                                           click="gradeRemove"><i class="fa-times"></i
                                           ></a>
```

```
54                                      <a class="btn btn-sm btn-outline-primary">{{
                                            comment.score}}</a>
55                                      <button class="btn btn-sm" style="float: right"
                                            v-on:click="reply">
56                                          <i class="fa fa-reply"></i>
57                                      </button>
58                                  </template>
59                              </div>
60                              <div class="card-block">
61                                  <template v-if="!(comment.id_comment > 0)">
62                                      <input type="text" class="form-control" v-model="
                                            comment.text" style="max-width: calc(100% - 38px
                                            ); display: inline"/>
63                                      <button class="btn btn-sm btn-outline-success" v-on:
                                            click="save">
64                                          <i class="fa fa-save" v-if="!isLoading"></i>
65                                          <i class="fa fa-circle-o-notch fa-spin" v-else
                                                ></i>
66                                      </button>
67                                  </template>
68                                  <p class="card-text p-1" v-else>{{comment.text}}</p>
69                              </div>
70                          </div>
71                          <template v-if="comment.children && comment.children.length > 0"
                                >
72                              <comment
73                                  v-for="child in comment.children"
74                                  v-bind:comment="child"
75                                  v-bind:level="level+1"
76                                  v-bind:key="child.id_comment">
77                              </comment>
78                          </template>
79                      </div>
80              `,
81          methods: {
82              save: function () {
83                  if (this.comment.text && this.comment.text.length) {
84                      this.isLoading = true;
85                      this.$http.post(
86                              "<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/add-
                                    comment.php",
87                              this.comment
88                              ).then(data => data.json())
89                              .then(data => {
90                                  if (data != null) {
91                                      this.comment.id_comment = data;
92                                      this.refresh();
93                                      this.isLoading = false;
94                                  }
95                              }).catch((ex) => {
96                          console.error(ex);
97                          this.isLoading = false;
98                      });
99                  }
100             },
101             refresh() {
102                 this.$forceUpdate();
103             },
104             reply() {
105                 if (!this.comment.children)
106                     this.comment.children = [];
```

```
107                 if (this.comment.children.length == 0 || (this.comment.children[this
                        .comment.children.length - 1].id_comment > 0)) {
108                     // push?
109                     this.comment.children.unshift({
110                         userfullname: newEmptyComment.userfullname,
111                         id_user: newEmptyComment.id_user,
112                         id_review: newEmptyComment.id_review,
113                         date_comment: newEmptyComment.date_comment,
114                         id_ref_comm: this.comment.id_comment || null,
115                         score: 0,
116                         children: []
117                     });
118                     this.refresh();
119                 }
120             },
121             gradeUp() {
122                 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
                         $_SERVER["REQUEST_URI"] ?>';
123                 this.$http.get(url)
124                         .then(data => data.json())
125                         .then(data => {
126                             if (data != null)
127                                 this.comment.score = data;
128                         });
129             },
130             gradeDown() {
131                 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 $_SERVER["REQUEST_URI"] ?>';
132                 this.$http.get(url)
133                         .then(data => data.json())
134                         .then(data => {
135                             if (data != null)
136                                 this.comment.score = data;
137                         });
138             },
139             gradeRemove() {
140                 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 $_SERVER["REQUEST_URI"] ?>';
141                 this.$http.get(url)
142                         .then(data => data.json())
143                         .then(data => {
144                             if (data != null)
145                                 this.comment.score = data;
146                         });
147                 }
148         },
149         data: function () {
150             return {
151                 isLoading: false
152             }
153         }
154     });
```

```
155     var commentsSection = new Vue({
156         el: '#commentsList',
157         data: {
158             comments: [],
159             emptyMessage: "No comments found"
160         }
161     });
162     var newEmptyComment = null;
163     $(document).on("click", ".comments-dialog-opener", function () {
164         var reviewId = $(this).attr("data-review-id");
165 
166         newEmptyComment = {
167             userfullname: "<?php echo AuthenticationService::getFullName() ?>",
168             id_user: <?php echo AuthenticationService::getUserId() ?>,
169             id_review: Number(reviewId),
170             date_comment: moment().tz("Europe/Rome").format(),
171             score: 0
172         };
173         commentsSection.comments = [];
174         $.getJSON("<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/get-comments.
                php?reviewId=" + reviewId, function (data) {
175             commentsSection.comments = data || [];
176             commentsSection.comments.push(newEmptyComment);
177         });
178     });
179 </script>
```