

Base di Dati - Relazione

Giacomo De Liberali - 857174

Daniele Rigon - 857319

Denny Ruffato - 859171

Luca Fortin - 858986

25 ottobre 2017

Indice

1	Introduzione	2
2	Progetto e funzionalità	2
2.1	Homepage	2
2.2	Login	2
2.3	Logout	2
2.4	Registrazione	2
2.5	Ricerca di un libro	3
2.6	Inserire un libro	3
2.7	Inserire un autore	3
2.8	Recensire un libro	3
2.9	Commentare una recensione e/o un commento	3
2.10	Votare commenti e recensioni	3
3	Sitemap	4
4	Database	5
4.1	Modello concettuale	5
4.2	Modello relazionale	6
5	Codice	7
5.1	Models	7
5.1.1	DatabaseConnection.php	7
5.1.2	Author.php	7
5.1.3	Book.php	8
5.1.4	Review.php	10
5.1.5	Comment.php	13
5.1.6	BookReviewSummary.php	14
5.2	Services	18
5.2.1	settings.json	18
5.2.2	Database.php	18
5.2.3	Defaults.php	19
5.2.4	LogsService.php	20
5.2.5	AuthenticationService.php	21
5.2.6	AuthorsService.php	23
5.2.7	BooksService.php	25
5.2.8	CommentsService.php	32
5.3	Rest	36
5.3.1	register.php	36

5.3.2	login.php	37
5.3.3	logout.php	37
5.3.4	add-author.php	38
5.3.5	add-book.php	38
5.3.6	add-review.php	39
5.3.7	add-comment.php	39
5.3.8	add-grade.php	40
5.3.9	add-comment-grade.php	40
5.3.10	get-comments.php	40
5.4	Partials	41
5.4.1	navbar.php	41
5.4.2	search-box.php	42
5.4.3	scripts.php	42
5.4.4	styles.php	44
5.4.5	register-form.php	45
5.4.6	login-form.php	46
5.4.7	add-author-form.php	46
5.4.8	add-book-form.php	47
5.4.9	comments.php	47

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.

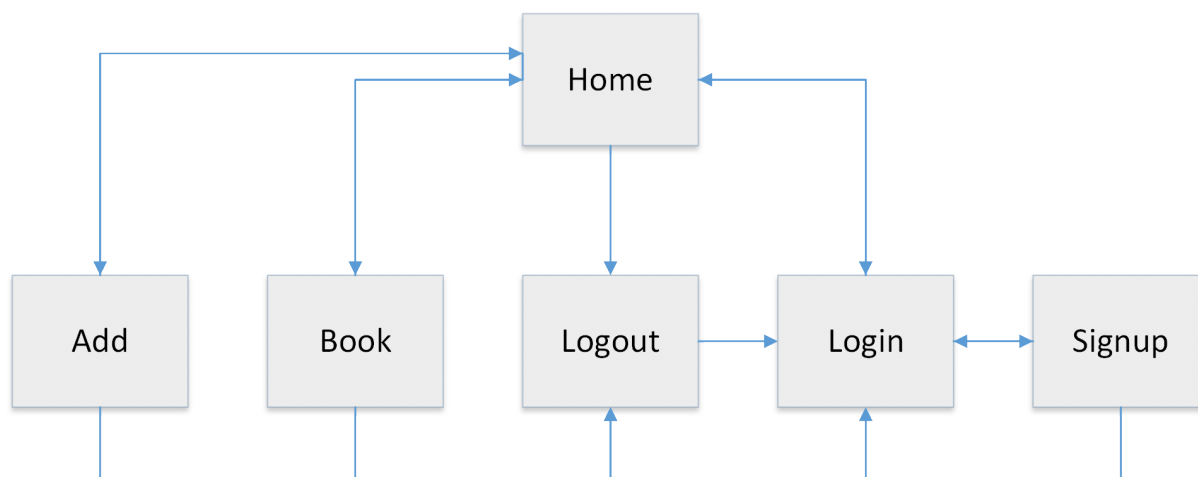


Figura 1: Struttura di navigazione del portale

4 Database

4.1 Modello concettuale

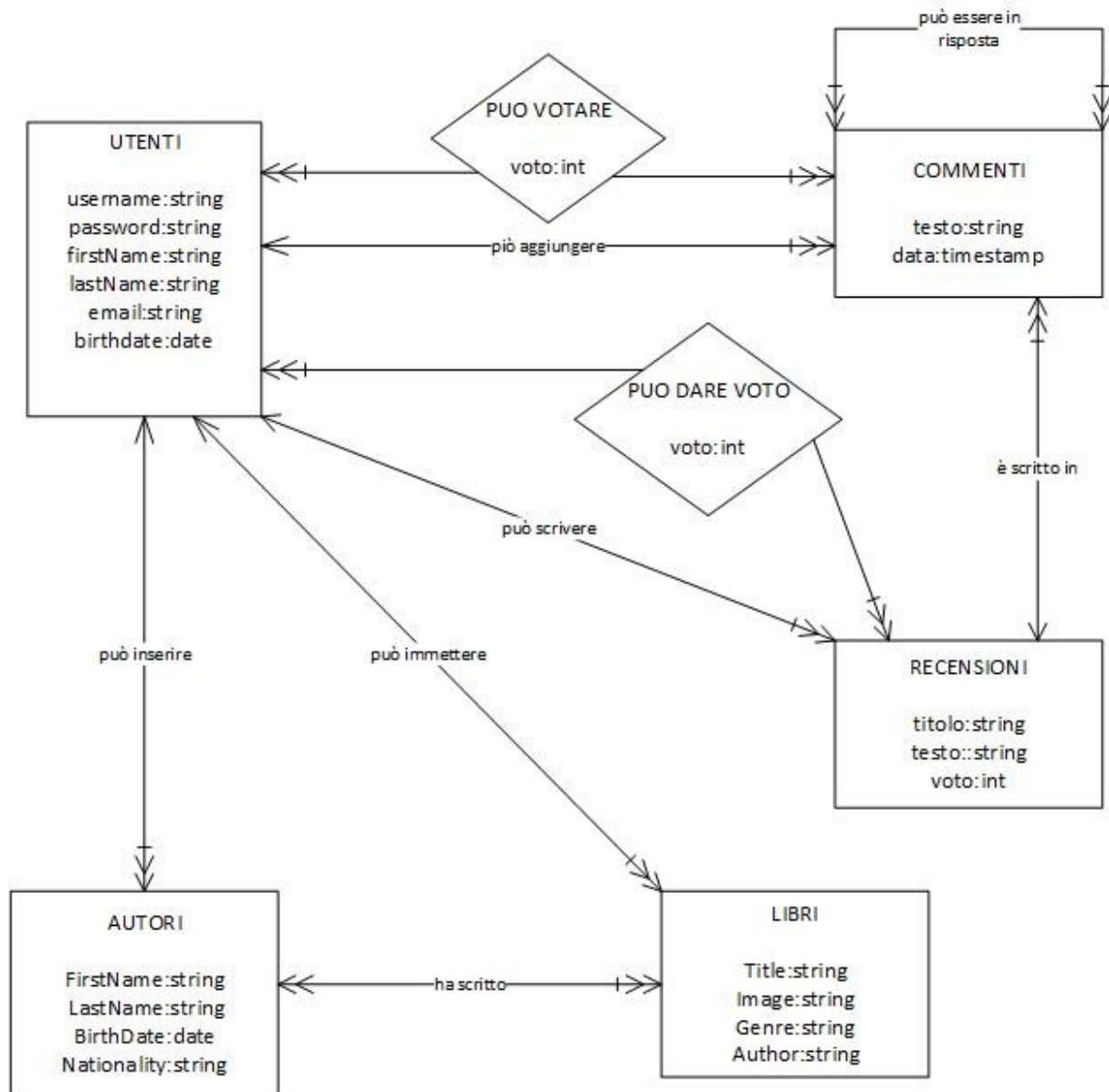


Figura 2: Schema concettuale del database dell'applicazione.

4.2 Modello relazionale

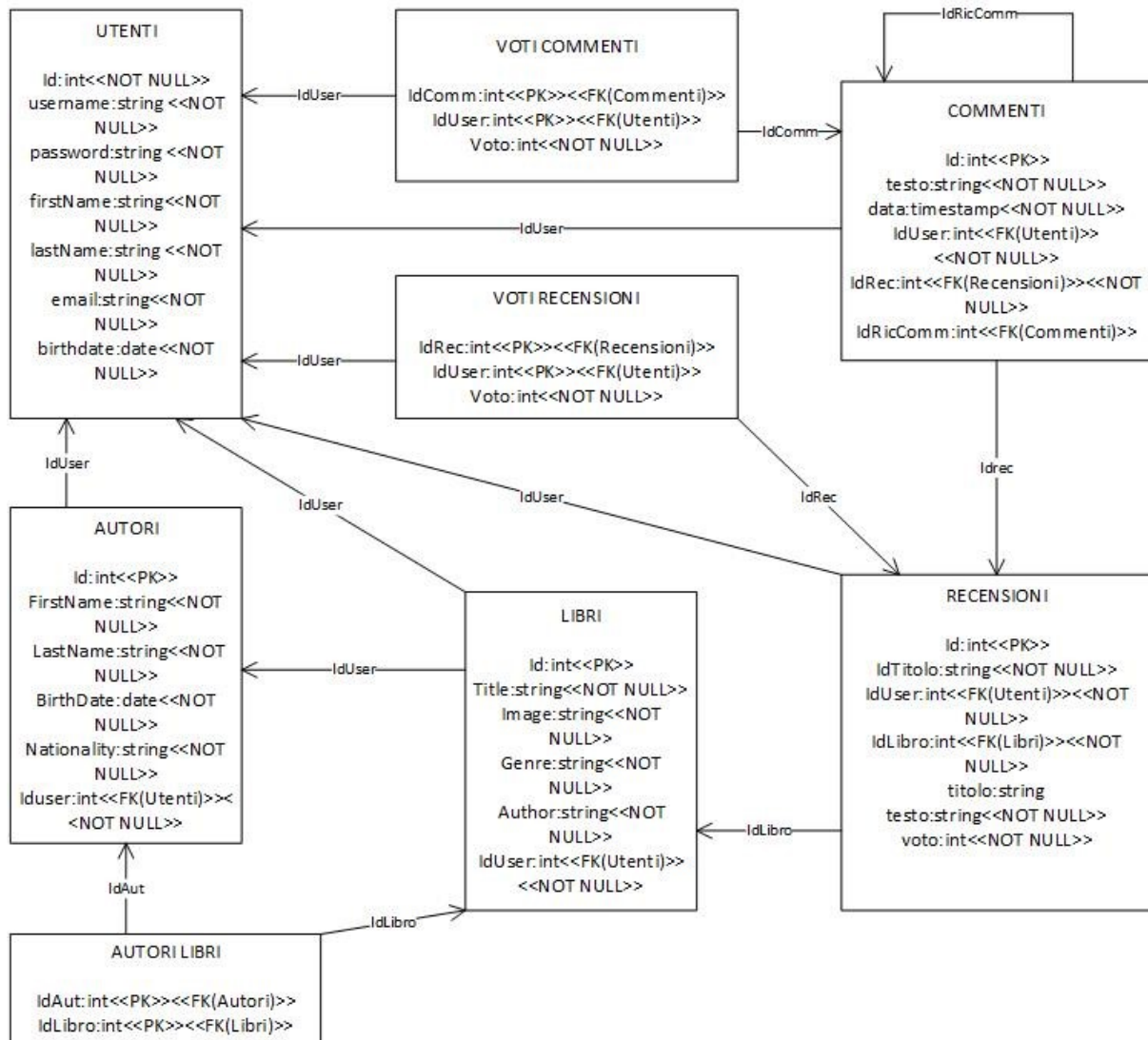


Figura 3: Schema relazionale del database dell'applicazione.

5 Codice

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

5.1 Models

Tutti i file in questa sezione 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.

```
1 <?php
2
3 /**
4  * The database connection information
5  */
6 class DatabaseConnection {
7
8     /**
9      * The database url
10     * @var type string
11     */
12     public $Url;
13
14     /**
15      * The database connection port
16      * @var type number
17      */
18     public $Port;
19
20     /**
21      * The database name
22      * @var type string
23      */
24     public $Name;
25
26     /**
27      * The database username
28      * @var type string
29      */
30     public $User;
31
32     /**
33      * The database password
34      * @var type string
35      */
36     public $Password;
37
38 }
39
40 ?>
```

5.1.2 Author.php

Rappresenta un autore di libri.

```
1 <?php
2
```



```

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             $this->rating = new BookReviewSummary ();
48         echo '<div class="row ">
49             <div class="col-md-4">
50                 
51             </div>
52             <div class="col-md-8">
53                 <div class="row">
54                     <div class="col-xs-12">
55                         ' . ($hasRate ? '<a href="/php/book.php?id=' . $this->id .
56                             '"><h4>' . $this->title . '<small>&nbsp;' . $this->
57                             author . '</small></h4></a>' : '<h4>' . $this->title . '
58                             <small>&nbsp;' . $this->author . '</small></h4>') .
59                         '</div>
60                     </div>
61                     <div class="row">
62                         <div class="col-xs-12">
63                             <i>' . $this->genre . '</i>
64                         </div>
65                     </div>';
66         if($hasRate){ echo '
67             <div class="row">
68                 <div class="col-xs-12">
69                     <div class="rating-block">
70                         <h4>Average user rating</h4>
71                         <h2 class="bold padding-bottom-7">' . ($this->
72                             rating->avarage ? $this->rating->avarage : '
73                             0') . ' <small>/ 5</small></h2>';
74                         for ($i = 1; $i <= 5; $i++) {
75                             echo '
76                                 <button type="button" class="btn btn-
77                                     warning btn-xs ' . (($i*1.0) > $this
78                                     ->rating->avarage ? 'btn-grey' : '')
79                                     . '" aria-label="Left Align">
80                                 <span class="fa fa-star" aria-hidden
81                                     ="true"></span>

```

```

73                                     </button>
74                                     ';
75                                     }
76                                     echo '
77                                     </div>
78                                     </div>
79                                     </div>';
80                                     } echo '
81                                     </div>';
82                                     </div>';
83     }
84 }
85 }
86
87 ?>

```

5.1.4 Review.php

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

```

1 <?php
2
3 /**
4  * A book review
5  */
6 class Review {
7
8     /**
9      * The review id
10     * @var integer
11     */
12     public $id_review;
13
14     /**
15      * The review title
16      * @var string
17      */
18     public $title;
19
20     /**
21      * The review text content
22      * @var string
23      */
24     public $text;
25
26     /**
27      * The review number of stars (1-5)
28      * @var integer
29      */
30     public $grade;
31
32     /**
33      * The total review score (+1/-1)
34      * @var integer
35      */
36     public $score;
37
38     /**
39      * The review author's id
40      * @var integer
41      */

```

```

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 >
78                             $this->grade ? 'btn-grey' : '') . '" aria-label="Left Align
79                             ">
80                             <span class="fa fa-star" aria-hidden="true"></span>
81                         </button>
82                     ' ;
83                 }
84                 echo '<a class="btn btn btn-outline-success" href="' . Defaults::
85                     DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=' .
86                     Defaults::SCORE_UP . '&userId=' . AuthenticationService::getUserId()
87                     . '&reviewId=' . $this->id_review . '&prevUrl=' . $_SERVER["
88                     REQUEST_URI"] . '"><i class="fa fa-thumbs-o-up"></i></a>
89                     <a class="btn btn btn-outline-danger" href="' . Defaults::
90                     DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=
91                     ' . Defaults::SCORE_DOWN . '&userId=' . AuthenticationService
92                     ::getUserId() . '&reviewId=' . $this->id_review . '&prevUrl='
93                     . $_SERVER["REQUEST_URI"] . '"><i class="fa fa-thumbs-o-down
94                     "></i></a>
95                     <a class="btn btn btn-outline-primary" href="' . Defaults::
96                     DEFAULT_BASE_URL . '/php/rest/add-grade.php?type=review&grade=
97                     ' . Defaults::SCORE_REMOVE . '&userId=' .
98                     AuthenticationService::getUserId() . '&reviewId=' . $this->
99                     id_review . '&prevUrl=' . $_SERVER["REQUEST_URI"] . '"><i
100                     class="fa fa-times"></i></a>
101                     <a class="btn btn btn-outline-primary">' . $this->score . '</a>
102                 </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 </div>
96     ' ;
97 }
98
99 public function renderForm() {
100     echo '
101         <form action="' . Defaults::DEFAULT_BASE_URL . '/php/rest/add-review
            .php" method="post" name="AddReviewForm">
102             <div class="review-block" style="min-width: 100%">
103                 <span>' . AuthenticationService::getFullName() . '</span>
104                 </div>
105                 <div class="row">
106                     <div class="col-sm-12">
107                         <div class="review-block-rate">
108                             <button type="button" class="btn btn-warning
109                                 btn-xs" id="star1" onclick="setStar(1)
110                                 ">
111                             <span class="fa fa-star" aria-hidden="
112                                 true"></span>
113                             </button>
114                             <button type="button" class="btn btn-warning
115                                 btn-xs" id="star2" onclick="setStar(2)
116                                 ">
117                             <span class="fa fa-star" aria-hidden="
118                                 true"></span>
119                             </button>
120                             <button type="button" class="btn btn-warning
121                                 btn-xs" id="star3" onclick="setStar(3)
122                                 ">
123                             <span class="fa fa-star" aria-hidden="
124                                 true"></span>
125                             </button>
126                             <button type="button" class="btn btn-warning
127                                 btn-xs" id="star4" onclick="setStar(4)
128                                 ">
129                             <span class="fa fa-star" aria-hidden="
130                                 true"></span>
131                             </button>
132                             <button type="button" class="btn btn-warning
133                                 btn-xs" id="star5" onclick="setStar(5)
134                                 ">
135                             <span class="fa fa-star" aria-hidden="
136                                 true"></span>
137                             </button>
138                         </div>
139                         <div class="review-block-title">
140                             <input type="text" placeholder="Title" name
141                                 ="Title" class="form-control">

```

```

125         </div>
126         <div class="review-block-description">
127             <textarea placeholder="Description" name="
128                 Text" class="form-control"></textarea>
129         </div>
130     </div>
131     <input type="submit" value="Submit" class="btn btn-sm"
132         style="margin-left: calc(100% - 60px); margin-top:
133         10px;" />
134 </div>
135 <input type="hidden" name="Grade" id="Grade" value="5"/>
136 <input type="hidden" name="IdBook" value="" . $this->id_book
137     . "" />
138 <input type="hidden" name="PrevUrl" value="" . $_SERVER[ '
139     REQUEST_URI' ] . "" />
140 <input type="hidden" name="IdAuthor" value="" .
141     AuthenticationService::getUserId() . "" />
142 </form>
143
144 <script>
145     function setStar(value) {
146         for (i = 1; i <= 5; i++) {
147             if (i <= value) {
148                 $("#star" + i).removeClass("btn-grey");
149             } else {
150                 $("#star" + i).addClass("btn-grey");
151             }
152         }
153         $("#Grade").val(value);
154     }
155 </script>
156
157     ;
158 }
159
160 }
161
162 ?>

```

5.1.5 Comment.php

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

```

1 <?php
2
3 /**
4  * A Comment
5  */
6 class Comment {
7
8     /**
9      * The comment identifier
10     * @var integer
11     */
12     public $id_comment;
13
14     /**
15      * The text content
16      * @var string
17      */
18     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 ?
62                             $this->avarage : '0') . ' <small>/ 5</small></h2>';
63                         for ($i = 1; $i <= 5; $i++) {
64                             echo '
65                                 <button type="button" class="btn btn-warning btn
66                                     -xs ' . (($i*1.0) > $this->avarage ? 'btn-
67                                     grey' : '') . '" aria-label="Left Align">
68                                     <span class="fa fa-star" aria-hidden="true
69                                     "></span>
70                                 </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
77             :1;">
78             <div style="height:9px; margin:5px 0;">5 <span class
79                 ="fa fa-star"></span></div>
80         </div>
81         <div class="pull-left" style="width:180px;">
82             <div class="progress" style="height:9px; margin:8px
83                 0;">
84                 <div class="progress-bar progress-bar-success"
85                     role="progressbar" aria-valuenow="5" aria-
86                     valuelmin="0" aria-valuemax="5" style="width:
87                         ' . $this->fiveStar/$currentTotal*100 . '
88                         %"></div>
89             </div>
90         </div>
91         <div class="pull-right" style="margin-left:10px;">' .
92             $this->fiveStar . '</div>
93     </div>
94
95     <div class="pull-left">
96         <div class="pull-left" style="width:35px; line-height
97             :1;">
98             <div style="height:9px; margin:5px 0;">4 <span class
99                 ="fa fa-star"></span></div>
100         </div>
101         <div class="pull-left" style="width:180px;">
102             <div class="progress" style="height:9px; margin:8px
103                 0;">
104                 <div class="progress-bar progress-bar-success"
105                     role="progressbar" aria-valuenow="5" aria-
106                     valuelmin="0" aria-valuemax="5" style="width:

```

```

107         ' . $this->threeStar/$currentTotal*100 . '
108         %"></div>
109     </div>
110     <div class="pull-right" style="margin-left:10px;">' .
111         $this->threeStar . '</div>
112 </div>
113 <div class="pull-left">
114     <div class="pull-left" style="width:35px; line-height
115         :1;">
116         <div style="height:9px; margin:5px 0;">2 <span class
117             ="fa fa-star"></span></div>
118     </div>
119     <div class="pull-left" style="width:180px;">
120         <div class="progress" style="height:9px; margin:8px
121             0;">
122             <div class="progress-bar progress-bar-success"
123                 role="progressbar" aria-valuenow="5" aria-
124                 valuelmin="0" aria-valuemax="5" style="width:
125                 ' . $this->twoStar/$currentTotal*100 . '
126                 %"></div>
127         </div>
128     </div>
129     <div class="pull-right" style="margin-left:10px;">' .
130         $this->twoStar . '</div>
131 </div>
132 <div class="pull-left">
133     <div class="pull-left" style="width:35px; line-height
134         :1;">
135         <div style="height:9px; margin:5px 0;">1 <span class
136             ="fa fa-star"></span></div>
137     </div>
138     <div class="pull-left" style="width:180px;">
139         <div class="progress" style="height:9px; margin:8px
140             0;">
141             <div class="progress-bar progress-bar-success"
142                 role="progressbar" aria-valuenow="5" aria-
143                 valuelmin="0" aria-valuemax="5" style="width:
144                 ' . $this->oneStar/$currentTotal*100 . '
145                 %"></div>
146         </div>
147     </div>
148     <div class="pull-right" style="margin-left:10px;">' .
149         $this->oneStar . '</div>
150 </div>
151 </div>
152 </div>
153 </div>
154 }
155 }
156 }
157 ?>

```

5.2 Services

Tutti i file in questa sezione 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.

```
1 {
2     "Debug": {
3         "Database": {
4             "Url": "localhost",
5             "Port": 5432,
6             "Name": "orsini",
7             "User": "postgres",
8             "Password": "root"
9         }
10    },
11    "Stage": {
12        "Database": {
13            "Url": "dblab.dsi.unive.it",
14            "Port": 5432,
15            "Name": "a2016u104",
16            "User": "a2016u104",
17            "Password": "XPhVUqk6"
18        }
19    }
20 }
```

5.2.2 Database.php

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

```
1 <?php
2 require(__DIR__ . '/../models/DatabaseConnection.php');
3
4 // psql --host dblab.dsi.unive.it --username a2016u104
5
6 /**
7  * Rappresent the application database connection
8  */
9 class Database {
10
11     /**
12      * The database connection configuration. Read from settings.json
13      * @var Models\DatabaseConnection
14      */
15     private $dbConfig = null;
16
17     /**
18      * The application database instance
19      * @var Database
20      */
21     private static $instance = null;
22
23     /**
24      * Mark private for singleton use
25      */
26     private function __construct() {
27         $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 = $configs["Stage"]["Database"];
33     }
34
35     /**
36      * Returns the database instance
37      * @return Database
38      */
39     public static function getInstance() {
40         if (Database::$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=' .
51             $this->dbConfig["Port"] . ';dbname=' . $this->dbConfig["Name"];
52         $connection = new PDO($connectionString, $this->dbConfig["User"], $this
53             ->dbConfig["Password"]);
54         $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
55         return $connection;
56     }
57
58     /**
59      * Return true if the current REMOTE_ADDRESS is local
60      * @return boolean
61      */
62     private function isDebug() {
63         $whitelist = array('127.0.0.1', '::1', 'localhost');
64         return in_array($_SERVER['REMOTE_ADDR'], $whitelist);
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         /** Default 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 exception 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, $contentToAppend) {
15             // Write the contents to the file,
16             // using the FILE_APPEND flag to append the content to the end of the
17             // file
18             // and the LOCK_EX flag to prevent anyone else writing to the file at
19             // the same time
20             file_put_contents($filePath, $contentToAppend, FILE_APPEND | LOCK_EX);
21         }
22
23         /**
24          * Logs in the 'logs/exceptions.txt' the given exception object
25          * @param Object $exceptionObj
26          */
27         public static function logException($exceptionObj) {
28             $now = new DateTime();
29             $str = "-----\n";
30             $str .= $now->format('Y-m-d H:i:s');
31             $str .= "\n";

```

```

30         $str .= $exceptionObj->__toString();
31         $str .= "\n-----\n";
32         self::log(__DIR__ . '/../..' . Defaults::LOG_EXCEPTION_FILE, $str);
33     }
34 }
35 }
36
37 ?>

```

5.2.5 AuthenticationService.php

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

```

1 <?php
2
3 ob_start();
4 session_start();
5 require_once(__DIR__ . '/Database.php');
6 require_once(__DIR__ . '/LogsService.php');
7
8 /**
9  * Provides base methods for authenticating the users and all user CRUD
10  * operations.
11  */
12 class AuthenticationService {
13
14     /**
15      * Given a username and a MD5 password returns true if a user with the given
16      * credentials exists or not.
17      * @param username string The user username
18      * @param password string The user password (in MD5)
19      * @return boolean
20      */
21     public static function login($username, $password) {
22         try {
23             $dbconn = Database::getInstance()->getConnection();
24             $statement = $dbconn->prepare('
25                 SELECT
26                     id,
27                     username,
28                     concat_ws(\' \', firstName, \' \', lastname) as userfullname,
29                     firstname
30                 FROM
31                     users
32                 WHERE
33                     lower(username) = lower(:username) AND password = :password
34             ');
35             $statement->bindParam(":username", $username, PDO::PARAM_STR);
36             $statement->bindParam(":password", $password, PDO::PARAM_STR);
37             $statement->execute();
38             $result = $statement->fetch(PDO::FETCH_NUM);
39             if ($result[0] != null) {
40                 // set session username
41                 $_SESSION["UserId"] = $result[0];
42                 $_SESSION["Username"] = $result[1];
43                 $_SESSION["UserFullName"] = $result[2];
44                 $_SESSION["UserFirstName"] = $result[3];
45                 return true;
46             }
47             // destroy 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 exists, false otherwise.
60  * @return boolean
61  */
62 public static function isLoggedIn() {
63     return (isset($_SESSION["Username"]) && $_SESSION["Username"] != null) ?
64         true : false;
65 }
66
67 /**
68  * Returns the current logged in user username or null
69  * @return string
70  */
71 public static function getUsername() {
72     return self::isLoggedIn() ? $_SESSION["Username"] : null;
73 }
74
75 /**
76  * Returns the current logged in user full name or null
77  * @return string
78  */
79 public static function getFullName() {
80     return self::isLoggedIn() ? $_SESSION["UserFullName"] : null;
81 }
82
83 /**
84  * Returns the current logged in user first name or null
85  * @return string
86  */
87 public static function getFirstName() {
88     return self::isLoggedIn() ? $_SESSION["UserFirstName"] : null;
89 }
90
91 /**
92  * Returns the current logged in user id or null
93  * @return integer
94  */
95 public static function getUserId() {
96     return self::isLoggedIn() ? $_SESSION["UserId"] : null;
97 }
98
99 /**
100  * Register a new user
101  * @param string username The user username
102  * @param string password The user password (in MD5)
103  * @param string firstName The user name
104  * @param string lastName The user surname
105  * @param string email The user email
106  * @param string birthdate The user birthdate
107  * @return boolean

```

```

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

```



```

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     ;
39     $authorStatment->bindParam(":lastName", $lastName, PDO::PARAM_STR);
40     $authorStatment->bindParam(":birhDate", $birhDate, PDO::PARAM_STR);
41     $authorStatment->bindParam(":nationality", $nationality, PDO::
42     PARAM_STR);
43     $authorStatment->execute();
44     $isAdded = $authorStatment->rowCount() == 1;
45     if ($isAdded)
46         return $dbconn->lastInsertId("authors_id_seq");
47     else
48         return 0;
49     } catch (PDOException $e) {
50         LogsService::logException($e);
51         return 0;
52     }
53 }
54
55 /**
56  * Returns all the authors
57  * @return Array<Author>
58  */
59 public static function getAuthors() {
60     try {
61         $dbconn = Database::getInstance()->getConnection();
62         $statement = $dbconn->prepare('
63         SELECT
64             *
65         FROM
66             authors
67         ,
68         );
69         $statement->execute();
70         return $statement->fetchAll(PDO::FETCH_CLASS, "Author");
71     } catch (PDOException $e) {
72         LogsService::logException($e);
73         return null;
74     }
75 }
76
77 /**
78  * Deletes an Author given its id
79  * @param $authorId integer
80  * @return boolean
81  */
82 public static function deleteAuthor($authorId) {
83     if ($authorId == null || $authorId < 0) {
84         return false;
85     }

```

```

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 {
71     try {
72         $dbconn = Database::getInstance()->getConnection();
73         $statement = $dbconn->prepare('
74             INSERT INTO reviews VALUES (
75                 DEFAULT,
76                 :title,
77                 :text,
78                 :grade,
79                 0,
80                 :idAuthor,
81                 :idBook
82             )
83         ');
84         $statement->bindParam(":title", $title, PDO::PARAM_STR);
85         $statement->bindParam(":text", $text, PDO::PARAM_STR);
86         $statement->bindParam(":grade", $grade, PDO::PARAM_INT);
87         $statement->bindParam(":idAuthor", $IdAuthor, PDO::PARAM_INT);
88         $statement->bindParam(":idBook", $idBook, PDO::PARAM_INT);
89         $statement->execute();
90         return $statement->rowCount() == 1;
91     } catch (PDOException $e) {
92         LogsService::logException($e);
93         return false;
94     }
95 }

```

```

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

```

```

212         $book->rating = 0;
213         $books[] = $book;
214     }
215     return $books;
216 } catch (PDOException $e) {
217     LogService::logException($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 $e) {
246         LogService::logException($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 $e) {
282         LogService::logException($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)
303                     as author,
304                 SUM(coalesce(grades_reviews.grade,0)) as score
305             FROM
306                 reviews JOIN users ON reviews.id_author = users.id
307                 LEFT JOIN grades_reviews ON reviews.id_review =
308                     grades_reviews.id_review
309             WHERE
310                 id_book = :bookId
311             GROUP BY reviews.id_review ,author
312         ';
313
314         $statement = $dbconn->prepare($query);
315         $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
316         $statement->execute();
317         $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "Review");
318
319         return $reviews;
320     } catch (PDOException $e) {
321         LogService::logException($e);
322         return null;
323     }
324 }
325
326 /**
327  * Returns true if the user has already done a review on the specified book
328  * @param integer $authorId The author id
329  * @param integer $bookId The book id
330  * @return boolean
331  */
332 public static function hasReview($authorId, $bookId) {

```

```

331         if ($bookId == null || $bookId < 0 || $authorId == null || $authorId <
332             0) {
333             return null;
334         }
335         try {
336             $dbconn = Database::getInstance()->getConnection();
337             $query = '
338                 SELECT
339                     *
340                 FROM
341                     reviews JOIN users ON reviews.id_author = users.id
342                 WHERE
343                     id_book = :bookId
344                     AND
345                     id_author = :authorId
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         if (count($reviews) > 0)
390             return $reviews[0];
391
392         return null;
393     } catch (PDOException $e) {
394         LogsService::logException($e);
395         return null;
396     }
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
23                         userfullname
24                 FROM
25                     comments c JOIN users u ON c.id_user = u.id
26                     LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
27                 WHERE id_review = :id_review AND c.id_ref_comm IS NULL
28                 GROUP BY c.id_comment, userfullname
29                 ORDER BY c.date_comment ASC
30             ');
31             $statement->bindParam(":id_review", $reviewId);
32             $row = $statement->execute();
33             $comments = array();
34             while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
35                 $comment = new Comment();
36                 $comment->id_comment = $row['id_comment'];
37                 $comment->text = $row['text'];
38                 $comment->score = $row['score'];
39                 $comment->id_user = $row['id_user'];
40                 $comment->userfullname = $row['userfullname'];
41                 $comment->id_review = $row['id_review'];
42                 $comment->id_ref_comm = $row['id_ref_comm'];

```

```

42         $comment->date_comment = $row['date_comment'];
43         $comment->canEdit = $row['id_user'] == AuthenticationService::
44             getUserId();
45         $comment->children = self::getCommentsByParentId($row['
46             id_comment']);
47         $comments[] = $comment;
48     }
49     return $comments;
50 } catch (PDOException $e) {
51     LogsService::logException($e);
52     return null;
53 }
54
55 /**
56  * Add a new comment
57  * @param Comment $comment
58  * @return boolean
59  */
60 public static function addComment($comment) {
61     try {
62         $dbconn = Database::getInstance()->getConnection();
63         $statement = $dbconn->prepare('
64             INSERT INTO COMMENTS VALUES (
65                 DEFAULT,
66                 :text,
67                 :id_user,
68                 :id_review,
69                 :id_ref_comm,
70                 :date_comment
71             )'
72         );
73         $statement->bindParam(":text", $comment->text);
74         $statement->bindParam(":id_user", $comment->id_user);
75         $statement->bindParam(":id_review", $comment->id_review);
76         $statement->bindParam(":id_ref_comm", $comment->id_ref_comm);
77         $statement->bindParam(":date_comment", $comment->date_comment);
78         $statement->execute();
79         $isAdded = $statement->rowCount() == 1;
80         if ($isAdded)
81             return $dbconn->lastInsertId("comments_id_comment_seq");
82         else
83             return 0;
84     } catch (PDOException $e) {
85         LogsService::logException($e);
86         return null;
87     }
88 }
89
90 /**
91  * Deletes a comment and all its children, given its id
92  * @param integer $commentId
93  * @return boolean
94  */
95 public static function deleteAuthor($commentId) {
96     if ($commentId == null || $commentId < 0) {
97         return false;
98     }
99     try {
100         $dbconn = Database::getInstance()->getConnection();
101         $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
131                     userfullname
132             FROM
133                 comments c JOIN users u ON c.id_user = u.id
134                 LEFT JOIN grades_comments gc ON gc.id_comment = c.id_comment
135             WHERE
136                 c.id_ref_comm = :parentId
137             GROUP BY c.id_comment, userfullname
138         ');
139         $statement->bindParam(":parentId", $parentId);
140         $row = $statement->execute();
141         $comments = array();
142         while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
143             $comment = new Comment();
144             $comment->id_comment = $row['id_comment'];
145             $comment->text = $row['text'];
146             $comment->score = $row['score'];
147             $comment->id_user = $row['id_user'];
148             $comment->userfullname = $row['userfullname'];
149             $comment->id_review = $row['id_review'];
150             $comment->id_ref_comm = $row['id_ref_comm'];
151             $comment->date_comment = $row['date_comment'];
152             $comment->canEdit = $row['id_user'] == AuthenticationService::
153                 getUserId();
154             $comment->children = self::getCommentsByParentId($row['
155                 id_comment']);
156             $comments[] = $comment;
157         }
158         return $comments;
159     } catch (PDOException $e) {
160         LogsService::logException($e);
161         return null;
162     }
163 }

```

```

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);

```

```

220         $statement->bindParam(":grade", $currentScore, PDO::PARAM_INT);
221         $statement->execute();
222         if ($statement->rowCount() == 1)
223             return self::getScore($commentId);
224         return null;
225     } catch (PDOException $e) {
226         LogsService::logException($e);
227         return false;
228     }
229 }
230
231 /**
232  * Return the score of the given comment
233  * @param integer $commentId
234  * @return integer
235  */
236 public static function getScore($commentId) {
237     try {
238         $dbconn = Database::getInstance()->getConnection();
239
240         $statement = $dbconn->prepare('
241 SELECT
242     SUM(grade) as score
243 FROM
244     grades_comments
245 WHERE
246     id_comment = :commentId
247 GROUP BY id_comment
248 ');
249         $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
250         $statement->execute();
251         $result = $statement->fetchAll()[0];
252
253         return $result["score"] ? $result["score"] : 0;
254     } catch (PDOException $e) {
255         LogsService::logException($e);
256         return false;
257     }
258 }
259
260 }
261
262 ?>

```

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.

```

1 <?php
2     require_once(' ../services/AuthenticationService.php');
3     require_once(' ../services/Defaults.php');
4
5     $isRegistered = false;
6
7     // 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        $isRegistered = AuthenticationService::register($_POST["Username"], md5(
15            $_POST["Password"],$_POST["FirstName"],$_POST["LastName"],$_POST["Email"]
16            ),$_POST["BirthDate"]);
17    }
18    if ($isRegistered){
19        // return on the previous page
20        header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/register.php?signup=
21            true');
22    }else{
23        // return on theregister page with a GET param
24        header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/register.php?errors=
25            true');
26    }
27    ?>

```

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($_POST["
10             Password"]));
11     }
12
13     if ($isLogged){
14         // return on the previous page
15         header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/login.php');
16     }else{
17         // return on the login page with a GET param
18         header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/login.php?
19             wrongCredentials=true');
20     }
21    ?>

```

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"]
14     ], $_POST["BirthDate"], $_POST["Nationality"]);
15 }
16
17 if ($authorId != null && $authorId > 0){
18     // return on the previous page
19     header('Location: ' . Defaults::DEFAULT_BASE_URL . '/php/add-book.php?added=
20     true ');
21 }else{
22     // return on the register page with a GET param
23     header('Location: ' . Defaults::DEFAULT_BASE_URL . '/php/add-book.php?errors=
24     true ');
25 }
26 ?>

```

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"],
14     $_POST["Genre"], $_POST["Author"]);
15 }
16
17 if ($isReviewAdded){
18     // return on the previous page
19 }
20 ?>

```

```

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"],
17            $_POST["Grade"], $_POST["IdAuthor"], $_POST["IdBook"]);
18    }
19
20    if ($isReviewAdded && isset($_POST["PrevUrl"])) {
21        header('Location: ' . $_POST["PrevUrl"]);
22    } else {
23        // return on the register page with a GET param
24        header('Location: ' . Defaults::DEFAULT_BASE_URL);
25    }
26 ?>

```

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.

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

5.3.9 add-comment-grade.php

Permette di aggiungere un voto ad un commento esistente.

```
1 <?php
2
3     require_once( '../services/CommentsService.php' );
4     require_once( '../services/Defaults.php' );
5
6     // if a username and a password is given in POST, then try to login
7     if ( isset($_GET["type"]) &&
8           isset($_GET["userId"]) &&
9           isset($_GET["grade"]) ) {
10        echo json_encode(CommentsService::addCommentGrade($_GET["commentId"], $_GET[
11        "userId"], $_GET["grade"]));
12    } else {
13        echo "null";
14    }
15 ?>
```

5.3.10 get-comments.php

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

```
1 <?php
2
3 require_once( '../services/AuthenticationService.php' );
4 require_once( '../services/CommentsService.php' );
5
6 // if a username and a password is given in POST, then try to login
7 if ( isset($_GET["reviewId"]) ) {
```

```

8     $result = CommentsService::getCommentsByReviewId($_GET["reviewId"]);
9     echo json_encode($result);
10 } else {
11     echo json_encode([]);
12 }
13 ?>

```

5.4 Partial

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.

```

1 <?php
2 require_once( __DIR__ . '/../services/AuthenticationService.php' );
3 require_once( __DIR__ . '/../services/Defaults.php' );
4
5
6 /**
7  * Indicates if the current user is logged in or not
8  */
9 $isLogged = AuthenticationService::isLoggedIn();
10 ?>
11
12 <nav class="navbar navbar-expand-lg navbar-light bg-light">
13     <a class="navbar-brand" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>" />BD2017
14     </a>
15     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#
16         navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="
17         false" aria-label="Toggle navigation">
18         <span class="navbar-toggler-icon"></span>
19     </button>
20     <div class="collapse navbar-collapse" id="navbarSupportedContent">
21         <ul class="navbar-nav mr-auto">
22             <li class="nav-item active">
23                 <a class="nav-link" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>">
24                     Home <span class="sr-only">(current)</span></a>
25             </li>
26             <?php
27             if (!$isLogged) {
28                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
29                     DEFAULT_BASE_URL . '/php/login.php">Login</a></li>';
30                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
31                     DEFAULT_BASE_URL . '/php/register.php">Sign up</a></li>';
32             } else {
33                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
34                     DEFAULT_BASE_URL . '/php/rest/logout.php">Logout</a></li>';
35                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
36                     DEFAULT_BASE_URL . '/php/add-book.php">Add</a></li>';
37             }
38             ?>
39         </ul>
40         <?php
41         if ($isLogged) {
42             require(__DIR__ . '/search-box.php');
43         }
44         ?>

```

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 $isLoggedIn ? AuthenticationService::getFirstName() . " search" : "
    Search" ?> a book..." aria-label="Search for...">
  <span class="input-group-btn hidden" id="cancelButtonGroup">
    <button class="btn btn-secondary btn-sm" type="button" id="cancelButton">&
      times;</button>
  </span>
  <span class="input-group-btn">
    <button class="btn btn-secondary btn-sm" type="button">
      <?php
        if (isset($_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-hwg4gsxgFZHoEEamdoYGBfL3FyQuiTwlAQgxVSNgt4=" 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+GLCKfwBdFNtXtclqenISfwAzpKaMNFNMj4" 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|01010101010101010101010101010101010101010101010101010101010101
```

```

arB0 Lz0 1cN0 1db0 1410 1on0 Wp0 1qL0 17d0 1cL0 M3B0 5M20 WMD 1fA0 1cM0 16M0
1iM0 16m0 1de0 1lc0 14m0 1lc0 W00 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 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 WMD 1qM0 WMD
1qM0 11A0 1o00 11A0 1o00 11A0 1qM0 WMD 1qM0 WMD 1qM0 WMD 1qM0 11A0 1o00 11
A0 1o00 11A0 1qM0 WMD 1qM0 WMD 1qM0 11A0 1o00 11A0 1o00 11A0 1o00 11A0 1qM0
WMD 1qM0 WMD 1qM0 11A0 1o00 11A0 1o00 11A0 1qM0 WMD 1qM0 WMD 1qM0 11A0 1o00
11A0 1o00 11A0 1o00 11A0 1qM0 WMD 1qM0 WMD 1qM0 11A0 1o00 11A0 1o00 11A0 1
qM0 WMD 1qM0 WMD 1qM0 WMD 1qM0 11A0 1o00 11A0 1o00|39e5');
11 </script>
12 <script src="<?php echo Defaults::DEFAULT_BASE_URL ?>/assets/scripts/vue.js"></
script>
13 <script src="<?php echo Defaults::DEFAULT_BASE_URL ?>/assets/scripts/vue-resource.
min.js"></script>
14
15
16 <script>
17 // START NAVBAR SEARCH BOX
18
19 /*
20 * Checks if the cancel button should be visible or not
21 */
22 $(document).ready(function() {
23     checkCancelButton();
24 });
25
26 /*
27 * When a click in the GO! button is received, navigates to the index page with
28 the get keyword param valorized with the input value
29 */
30 $(document).on("click", "#searchButton", function() {
31     // when click on the Go button
32     searchBook();
33 });
34
35 $(document).on("click", "#cancelButton", function() {
36     // when click on the X button
37     $("#keyword").val("");
38     $(this).hide();
39     $("#searchButton").click();
40 });
41
42 /**
43 * Every time a button is pressed in the input text, checks if the cancel button
44 should be visible or not.
45 * If the button pressed is the Enter (invio), search the book with the current
46 keyword.
47 */
48 $(document).on("keyup", "#keyword", function(e) {
49     // when enter in the keyword input text
50     checkCancelButton();
51
52     if (e.which == 13)
53         searchBook();
54 });
55
56 /**
57 * Shows or hides the cancel button if the keyword input is valorized or empty
58 */

```

```

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
66  * input value
67  */
68 function searchBook() {
69     // navigate to homepage with a get parameter
70     var keyword = $("#keyword").val();
71     var currentSort = "<?php echo isset($_GET['sort']) ? $_GET['sort'] :
72         Defaults::ASC ?>";
73     var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
74     // if no value is provided, navigate without the param
75     if (!keyword || keyword.length == 0)
76         window.location.href = defaultBaseUrl + currentSort ? '?sort=' +
77             currentSort : '';
78     else
79         window.location.href = defaultBaseUrl + "?keyword=" + keyword + (
80             currentSort ? '&sort=' + currentSort : '');
81 }
82
83 /**
84  * When click on sort change button in navbar
85  */
86 $(document).on("click", "#bookSort", function() {
87     var currentSort = "<?php echo isset($_GET['sort']) ? $_GET['sort'] :
88         Defaults::DESC ?>";
89     var currentKeyword = "<?php echo isset($_GET['keyword']) ? $_GET['keyword']
90         : null ?>";
91     var defaultAsc = "<?php echo Defaults::ASC; ?>";
92     var defaultDesc = "<?php echo Defaults::DESC; ?>";
93     var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
94
95     if (!currentKeyword || currentKeyword.length == 0)
96         window.location.href = defaultBaseUrl + currentSort ? '?sort=' + (
97             currentSort == defaultDesc ? defaultAsc : defaultDesc) : '';
98     else {
99         window.location.href = defaultBaseUrl + "?keyword=" + currentKeyword +
100             (currentSort ? '&sort=' + (currentSort == defaultDesc ? defaultAsc
101                 : defaultDesc) : '');
102     }
103 });
104 // END NAVBAR SEARCH BOX
105 </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" />

```

```

6 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no
  ">
7 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/
  css/bootstrap.min.css">
8 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/
  font-awesome.min.css">
9
10 <!-- Custom CSS -->
11 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
  /custom.css">
12 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
  /reviews.css">
13 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
  /comments.css">

```

5.4.5 register-form.php

Form di registrazione.

```

1 <?php
2     require_once( __DIR__ . '/../services/Defaults.php');
3 ?>
4 <div class="row">
5     <div class="col-md-6 ml-md-auto mr-md-auto">
6
7         <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/register.
  php'; ?>" method="post" name="RegisterForm">
8             <h3>Sign Up</h3>
9             <br>
10            <div class="form-group">
11                <input type="text" class="form-control form-control-sm" name="
  FirstName" placeholder="First name" autofocus="true" required="
  true" autocomplete="off">
12            </div>
13            <div class="form-group">
14                <input type="text" class="form-control form-control-sm" name="
  LastName" placeholder="Last name" autofocus="true" required="
  true" autocomplete="off"/>
15            </div>
16            <div class="form-group">
17                <input type="date" class="form-control form-control-sm" name="
  BirthDate" placeholder="Birth date" autofocus="true" required="
  true" />
18            </div>
19            <div class="form-group">
20                <input type="text" class="form-control form-control-sm" name="
  Username" placeholder="Username" autofocus="true" required="true
  " autocomplete="off"/>
21            </div>
22            <div class="form-group">
23                <input type="email" class="form-control form-control-sm" name="Email
  " placeholder="Email" required="true" autocomplete="off"/>
24            </div>
25            <div class="form-group">
26                <input type="password" class="form-control form-control-sm" name="
  Password" placeholder="Password" required="true" autocomplete="
  off"/>
27            </div>
28            <button class="btn btn-sm btn-primary" name="Submit" value="SignUp"
  type="Submit">Register</button>
29        </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'; ?>"
5     method="post" name="LoginForm">
6     <h3>Login</h3>
7     <br>
8     <div class="form-group">
9         <input type="text" class="form-control form-control-sm" name="Username"
10             placeholder="Username" required="" autofocus="" />
11     </div>
12     <div class="form-group">
13         <input type="password" class="form-control form-control-sm" name="Password"
14             placeholder="Password" required="" />
15     </div>
16     <button class="btn btn-sm btn-primary" name="Submit" value="Login" type="Submit"
17         ">Sign in</button>
18 </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'; ?>"
10         method="post" name="AddAuthorForm">
11         <h3>Add author</h3>
12         <br>
13         <div class="form-group">
14             <input type="text" class="form-control form-control-sm" name="FirstName"
15                 placeholder="First name" required="true" autocomplete="off">
16         </div>
17         <div class="form-group">
18             <input type="text" class="form-control form-control-sm" name="LastName"
19                 placeholder="Last name" required="true" autocomplete="off">
20         </div>
21         <div class="form-group">
22             <input type="date" class="form-control form-control-sm" name="BirthDate"
23                 placeholder="BirthDate" />
24         </div>
25         <div class="form-group">
26             <input type="text" class="form-control form-control-sm" name="Nationality"
27                 placeholder="Nationality" />
28         </div>
29         <button class="btn btn-sm btn-primary" name="Submit" value="Add" type="Submit"
30             ">Add</button>
```

```

25     </form>
26 </div>

```

5.4.8 add-book-form.php

Form di creazione di un nuovo libro.

```

1 <?php
2     require_once( __DIR__ . '/../services/AuthenticationService.php' );
3     require_once( __DIR__ . '/../services/AuthorsService.php' );
4     require_once( __DIR__ . '/../services/Defaults.php' );
5
6
7     if ( ! AuthenticationService::isLoggedIn() ) {
8         die();
9     }
10
11     $authors = AuthorsService::getAuthors();
12 ?>
13 <div class="col-md-6">
14     <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/add-book.php';
15         ?>" method="post" name="AddBookForm">
16         <h3>Add book</h3>
17         <br>
18         <div class="form-group">
19             <input type="text" class="form-control form-control-sm" name="Title"
20                 placeholder="Title" autofocus="true" required="true" autocomplete="
21                 off">
22         </div>
23         <div class="form-group">
24             <input type="text" class="form-control form-control-sm" name="Image"
25                 placeholder="Image url" required="true" autocomplete="off"/>
26         </div>
27         <div class="form-group">
28             <input type="text" class="form-control form-control-sm" name="Genre"
29                 placeholder="Genre" required="true" />
30         </div>
31         <div class="form-group">
32             <select name="Author" class="form-control">
33                 <?php
34                     foreach ( $authors as $author ) {
35                         echo '<option value="' . $author->id . '">' . $author->
36                             firstname . ' ' . $author->lastname . '</option>';
37                     }
38                 ?>
39             </select>
40         </div>
41         <button class="btn btn-sm btn-primary" name="Submit" value="Add" type="
42             Submit">Add</button>
43     </form>
44 </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](#).

```

1 <?php
2 require_once( __DIR__ . '/../services/AuthenticationService.php' );
3 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"
52                 ></i></a>
53               <a class="btn btn-sm btn-outline-danger" v-on:
                 click="gradeDown"><i class="fa fa-thumbs-o-
                  down"></i></a>
54               <a class="btn btn-sm btn-outline-primary" v-on:
                 click="gradeRemove"><i class="fa -times"></i>
55             </a>

```

```

54         <a class="btn btn-sm btn-outline-primary">{{
55             comment.score}}</a>
56         <button class="btn btn-sm" style="float: right"
57             v-on:click="reply">
58             <i class="fa fa-reply"></i>
59         </button>
60     </template>
61 </div>
62 <div class="card-block">
63     <template v-if="!(comment.id_comment > 0)">
64         <input type="text" class="form-control" v-model="
65             comment.text" style="max-width: calc(100% - 38px
66             ); display: inline"/>
67         <button class="btn btn-sm btn-outline-success" v-on:
68             click="save">
69             <i class="fa fa-save" v-if="!isLoading"></i>
70             <i class="fa fa-circle-o-notch fa-spin" v-else
71                 ></i>
72         </button>
73     </template>
74     <p class="card-text p-1" v-else>{{comment.text}}</p>
75 </div>
76 <template v-if="comment.children && comment.children.length > 0"
77     >
78     <comment
79         v-for="child in comment.children"
80         v-bind:comment="child"
81         v-bind:level="level+1"
82         v-bind:key="child.id_comment">
83     </comment>
84 </template>
85 </div>
86 ,
87 methods: {
88     save: function () {
89         if (this.comment.text && this.comment.text.length) {
90             this.isLoading = true;
91             this.$http.post(
92                 "<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/add-
93                 comment.php",
94                 this.comment
95             ).then(data => data.json())
96             .then(data => {
97                 if (data != null) {
98                     this.comment.id_comment = data;
99                     this.refresh();
100                     this.isLoading = false;
101                 }
102             }).catch((ex) => {
103                 console.error(ex);
104                 this.isLoading = false;
105             });
106         }
107     },
108     refresh() {
109         this.$forceUpdate();
110     },
111     reply() {
112         if (!this.comment.children)
113             this.comment.children = [];
114     }
115 }

```

```

107         if (this.comment.children.length == 0 || (this.comment.children[this
108             .comment.children.length - 1].id_comment > 0)) {
109             // push?
110             this.comment.children.unshift({
111                 userfullname: newEmptyComment.userfullname,
112                 id_user: newEmptyComment.id_user,
113                 id_review: newEmptyComment.id_review,
114                 date_comment: newEmptyComment.date_comment,
115                 id_ref_comm: this.comment.id_comment || null,
116                 score: 0,
117                 children: []
118             });
119             this.refresh();
120         },
121         gradeUp() {
122             var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
123                 comment-grade.php?type=comment&grade=<?php echo Defaults::
124                 SCORE_UP ?>&userId=<?php echo AuthenticationService::getUserId()
125                 ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php echo
126                 $_SERVER["REQUEST_URI"] ?>';
127             this.$http.get(url)
128                 .then(data => data.json())
129                 .then(data => {
130                     if (data != null)
131                         this.comment.score = data;
132                 });
133         },
134         gradeDown() {
135             var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
136                 comment-grade.php?type=comment&grade=<?php echo Defaults::
137                 SCORE_DOWN ?>&userId=<?php echo AuthenticationService::getUserId
138                 () ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php
139                 echo $_SERVER["REQUEST_URI"] ?>';
140             this.$http.get(url)
141                 .then(data => data.json())
142                 .then(data => {
143                     if (data != null)
144                         this.comment.score = data;
145                 });
146         },
147         gradeRemove() {
148             var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
149                 comment-grade.php?type=comment&grade=<?php echo Defaults::
150                 SCORE_REMOVE ?>&userId=<?php echo AuthenticationService::
151                 getUserId() ?>&commentId=' + this.comment.id_comment + '&prevUrl
152                 =<?php echo $_SERVER["REQUEST_URI"] ?>';
153             this.$http.get(url)
154                 .then(data => data.json())
155                 .then(data => {
156                     if (data != null)
157                         this.comment.score = data;
158                 });
159         }
160     },
161     data: function () {
162         return {
163             isLoading: false
164         }
165     }
166 }
167 });

```

```

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.
175             php?reviewId=" + reviewId, function (data) {
176             commentsSection.comments = data || [];
177             commentsSection.comments.push(newEmptyComment);
178         });
179     </script>

```