

Base di Dati - Relazione

Giacomo De Liberali - 857174

Daniele Rigon - 857319

Denny Ruffato - 859171

Luca Fortin - 858986

19 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	17
5.2.1	Database.php	18
5.2.2	Defaults.php	19
5.2.3	LogsService.php	19
5.2.4	AuthenticationService.php	20
5.2.5	AuthorsService.php	23
5.2.6	BooksService.php	24
5.2.7	CommentsService.php	31
5.3	Rest	36
5.3.1	register.php	36
5.3.2	login.php	36

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

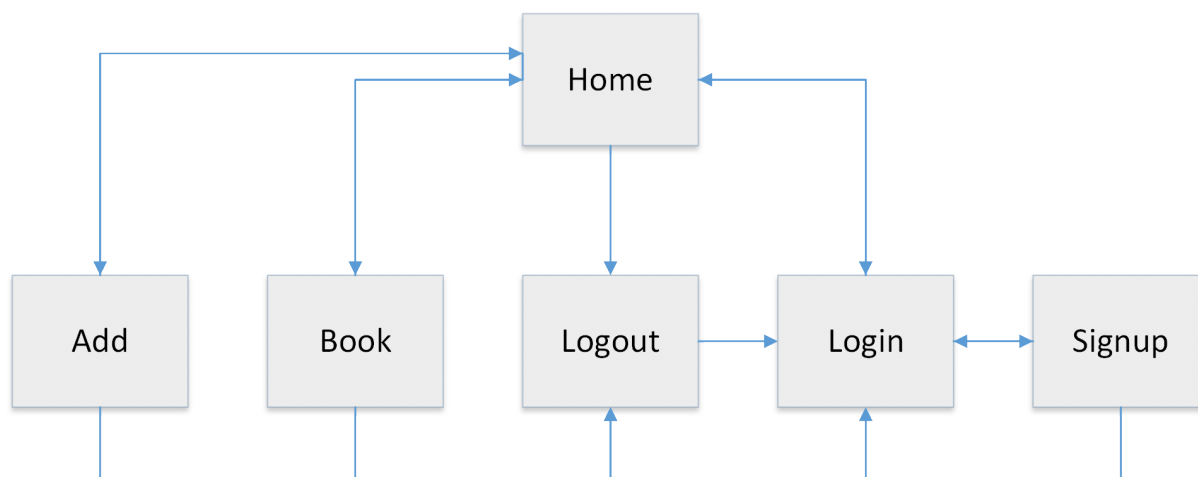


Figura 1: Struttura di navigazione del portale

4 Database

4.1 Modello concettuale

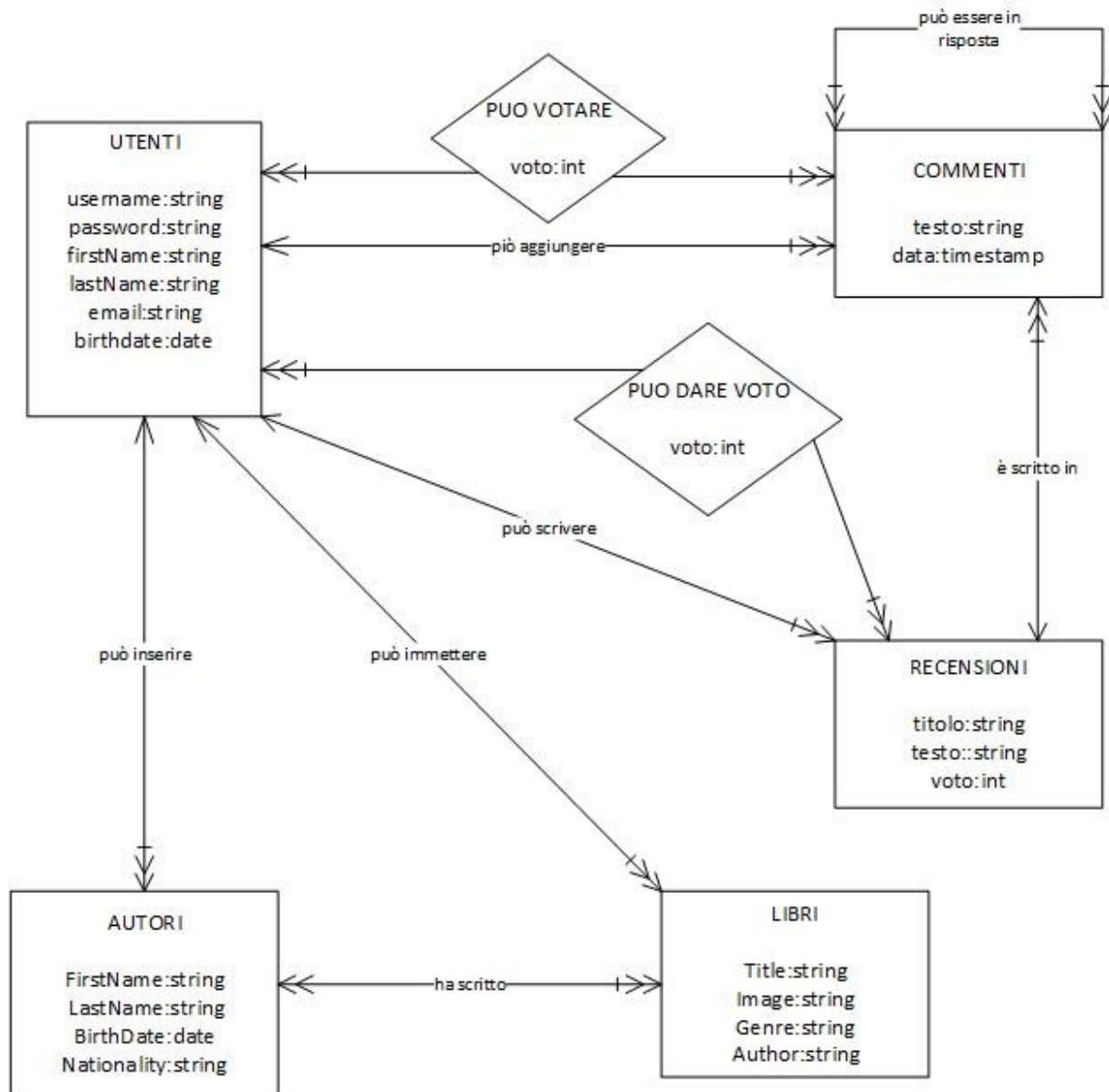


Figura 2: Schema concettuale del database dell'applicazione.

4.2 Modello relazionale

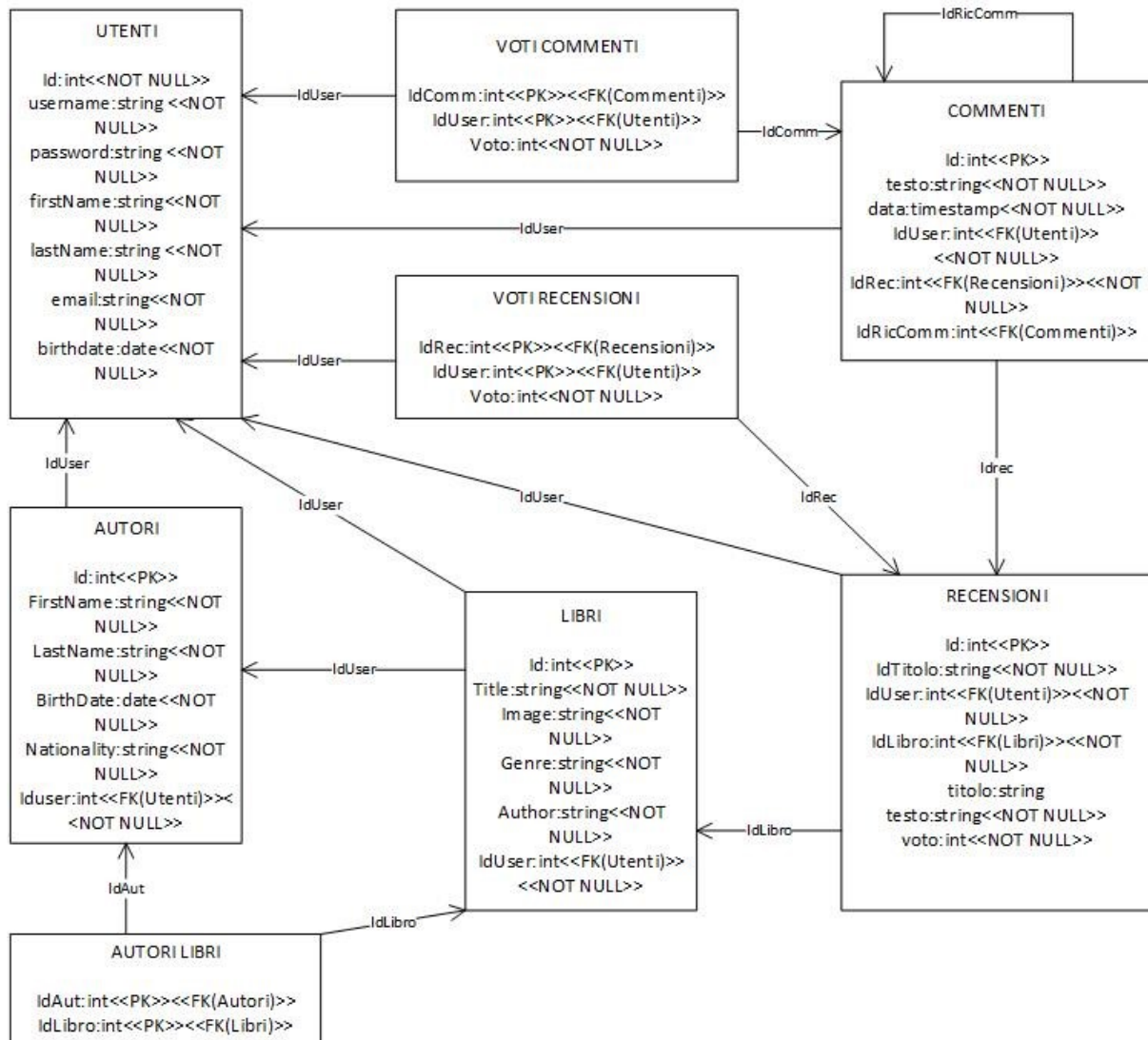


Figura 3: Schema relazionale del database dell'applicazione.

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                 <img src="" . $this->image . "" height="200"/>
51             </div>
52             <div class="col-md-8">
53                 <div class="row">
54                     <div class="col-xs-12">
55                         <a href="./php/book.php?id=' . $this->id . '"><h4>' . $this
56                             ->title . '<small>&nbsp;' . $this->author . '</small></
57                             h4></a>
58                     </div>
59                 </div>
60                 <div class="row">
61                     <div class="col-xs-12">
62                         <i>' . $this->genre . '</i>
63                     </div>
64                 </div>';
65                 if($hasRate){ echo '
66                     <div class="row">
67                         <div class="col-xs-12">
68                             <div class="rating-block">
69                                 <h4>Average user rating</h4>
70                                 <h2 class="bold padding-bottom-7">' . ($this->
71                                     rating->avarage ? $this->rating->avarage : '
72                                     0') . ' <small>/ 5</small></h2>';
73                                 for ($i = 1; $i <= 5; $i++) {
74                                     echo '
75                                     <button type="button" class="btn btn-
76                                         warning btn-xs ' . (($i*1.0) > $this
77                                             ->rating->avarage ? 'btn-grey' : '')
78                                             . '" aria-label="Left Align">
79                                     <span class="fa fa-star" aria-hidden
80                                         ="true"></span>
81                                     </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>
103         <div class="review-block-title">' . $this->title . '</

```

```



```

```

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>
71                             '
72             '
73         '
74     '
75     '
76     '
77     '
78     '
79     '
80     '
81     '
82     '
83     '
84     '
85     '
86     '
87     '
88     '
89     '
90     '
91     '
92     '
93     '
94     '
95     '
96     '
97     '
98     '
99     '
100    '
101    '
102    '
103    '
104    '
105    '
106    '
107    '
108    '
109    '
110    '
111    '
112    '
113    '
114    '
115    '
116    '
117    '
118    '
119    '
120    '
121    '
122    '
123    '
124    '
125    '
126    '
127    '
128    '
129    '
130    '
131    '
132    '
133    '
134    '
135    '
136    '
137    '
138    '
139    '
140    '
141    '
142    '
143    '
144    '
145    '
146    '
147    '
148    '
149    '
150    '
151    '
152    '
153    '
154    '
155    '
156    '
157    '
158    '
159    '
160    '
161    '
162    '
163    '
164    '
165    '
166    '
167    '
168    '
169    '
170    '
171    '
172    '
173    '
174    '
175    '
176    '
177    '
178    '
179    '
180    '
181    '
182    '
183    '
184    '
185    '
186    '
187    '
188    '
189    '
190    '
191    '
192    '
193    '
194    '
195    '
196    '
197    '
198    '
199    '
200    '
201    '
202    '
203    '
204    '
205    '
206    '
207    '
208    '
209    '
210    '
211    '
212    '
213    '
214    '
215    '
216    '
217    '
218    '
219    '
220    '
221    '
222    '
223    '
224    '
225    '
226    '
227    '
228    '
229    '
230    '
231    '
232    '
233    '
234    '
235    '
236    '
237    '
238    '
239    '
240    '
241    '
242    '
243    '
244    '
245    '
246    '
247    '
248    '
249    '
250    '
251    '
252    '
253    '
254    '
255    '
256    '
257    '
258    '
259    '
260    '
261    '
262    '
263    '
264    '
265    '
266    '
267    '
268    '
269    '
270    '
271    '
272    '
273    '
274    '
275    '
276    '
277    '
278    '
279    '
280    '
281    '
282    '
283    '
284    '
285    '
286    '
287    '
288    '
289    '
290    '
291    '
292    '
293    '
294    '
295    '
296    '
297    '
298    '
299    '
300    '
301    '
302    '
303    '
304    '
305    '
306    '
307    '
308    '
309    '
310    '
311    '
312    '
313    '
314    '
315    '
316    '
317    '
318    '
319    '
320    '
321    '
322    '
323    '
324    '
325    '
326    '
327    '
328    '
329    '
330    '
331    '
332    '
333    '
334    '
335    '
336    '
337    '
338    '
339    '
340    '
341    '
342    '
343    '
344    '
345    '
346    '
347    '
348    '
349    '
350    '
351    '
352    '
353    '
354    '
355    '
356    '
357    '
358    '
359    '
360    '
361    '
362    '
363    '
364    '
365    '
366    '
367    '
368    '
369    '
370    '
371    '
372    '
373    '
374    '
375    '
376    '
377    '
378    '
379    '
380    '
381    '
382    '
383    '
384    '
385    '
386    '
387    '
388    '
389    '
390    '
391    '
392    '
393    '
394    '
395    '
396    '
397    '
398    '
399    '
400    '
401    '
402    '
403    '
404    '
405    '
406    '
407    '
408    '
409    '
410    '
411    '
412    '
413    '
414    '
415    '
416    '
417    '
418    '
419    '
420    '
421    '
422    '
423    '
424    '
425    '
426    '
427    '
428    '
429    '
430    '
431    '
432    '
433    '
434    '
435    '
436    '
437    '
438    '
439    '
440    '
441    '
442    '
443    '
444    '
445    '
446    '
447    '
448    '
449    '
450    '
451    '
452    '
453    '
454    '
455    '
456    '
457    '
458    '
459    '
460    '
461    '
462    '
463    '
464    '
465    '
466    '
467    '
468    '
469    '
470    '
471    '
472    '
473    '
474    '
475    '
476    '
477    '
478    '
479    '
480    '
481    '
482    '
483    '
484    '
485    '
486    '
487    '
488    '
489    '
490    '
491    '
492    '
493    '
494    '
495    '
496    '
497    '
498    '
499    '
500    '
501    '
502    '
503    '
504    '
505    '
506    '
507    '
508    '
509    '
510    '
511    '
512    '
513    '
514    '
515    '
516    '
517    '
518    '
519    '
520    '
521    '
522    '
523    '
524    '
525    '
526    '
527    '
528    '
529    '
530    '
531    '
532    '
533    '
534    '
535    '
536    '
537    '
538    '
539    '
540    '
541    '
542    '
543    '
544    '
545    '
546    '
547    '
548    '
549    '
550    '
551    '
552    '
553    '
554    '
555    '
556    '
557    '
558    '
559    '
560    '
561    '
562    '
563    '
564    '
565    '
566    '
567    '
568    '
569    '
570    '
571    '
572    '
573    '
574    '
575    '
576    '
577    '
578    '
579    '
580    '
581    '
582    '
583    '
584    '
585    '
586    '
587    '
588    '
589    '
590    '
591    '
592    '
593    '
594    '
595    '
596    '
597    '
598    '
599    '
600    '
601    '
602    '
603    '
604    '
605    '
606    '
607    '
608    '
609    '
610    '
611    '
612    '
613    '
614    '
615    '
616    '
617    '
618    '
619    '
620    '
621    '
622    '
623    '
624    '
625    '
626    '
627    '
628    '
629    '
630    '
631    '
632    '
633    '
634    '
635    '
636    '
637    '
638    '
639    '
640    '
641    '
642    '
643    '
644    '
645    '
646    '
647    '
648    '
649    '
650    '
651    '
652    '
653    '
654    '
655    '
656    '
657    '
658    '
659    '
660    '
661    '
662    '
663    '
664    '
665    '
666    '
667    '
668    '
669    '
670    '
671    '
672    '
673    '
674    '
675    '
676    '
677    '
678    '
679    '
680    '
681    '
682    '
683    '
684    '
685    '
686    '
687    '
688    '
689    '
690    '
691    '
692    '
693    '
694    '
695    '
696    '
697    '
698    '
699    '
700    '
701    '
702    '
703    '
704    '
705    '
706    '
707    '
708    '
709    '
710    '
711    '
712    '
713    '
714    '
715    '
716    '
717    '
718    '
719    '
720    '
721    '
722    '
723    '
724    '
725    '
726    '
727    '
728    '
729    '
730    '
731    '
732    '
733    '
734    '
735    '
736    '
737    '
738    '
739    '
740    '
741    '
742    '
743    '
744    '
745    '
746    '
747    '
748    '
749    '
750    '
751    '
752    '
753    '
754    '
755    '
756    '
757    '
758    '
759    '
760    '
761    '
762    '
763    '
764    '
765    '
766    '
767    '
768    '
769    '
770    '
771    '
772    '
773    '
774    '
775    '
776    '
777    '
778    '
779    '
780    '
781    '
782    '
783    '
784    '
785    '
786    '
787    '
788    '
789    '
790    '
791    '
792    '
793    '
794    '
795    '
796    '
797    '
798    '
799    '
800    '
801    '
802    '
803    '
804    '
805    '
806    '
807    '
808    '
809    '
810    '
811    '
812    '
813    '
814    '
815    '
816    '
817    '
818    '
819    '
820    '
821    '
822    '
823    '
824    '
825    '
826    '
827    '
828    '
829    '
830    '
831    '
832    '
833    '
834    '
835    '
836    '
837    '
838    '
839    '
840    '
841    '
842    '
843    '
844    '
845    '
846    '
847    '
848    '
849    '
850    '
851    '
852    '
853    '
854    '
855    '
856    '
857    '
858    '
859    '
860    '
861    '
862    '
863    '
864    '
865    '
866    '
867    '
868    '
869    '
870    '
871    '
872    '
873    '
874    '
875    '
876    '
877    '
878    '
879    '
880    '
881    '
882    '
883    '
884    '
885    '
886    '
887    '
888    '
889    '
890    '
891    '
892    '
893    '
894    '
895    '
896    '
897    '
898    '
899    '
900    '
901    '
902    '
903    '
904    '
905    '
906    '
907    '
908    '
909    '
910    '
911    '
912    '
913    '
914    '
915    '
916    '
917    '
918    '
919    '
920    '
921    '
922    '
923    '
924    '
925    '
926    '
927    '
928    '
929    '
930    '
931    '
932    '
933    '
934    '
935    '
936    '
937    '
938    '
939    '
940    '
941    '
942    '
943    '
944    '
945    '
946    '
947    '
948    '
949    '
950    '
951    '
952    '
953    '
954    '
955    '
956    '
957    '
958    '
959    '
960    '
961    '
962    '
963    '
964    '
965    '
966    '
967    '
968    '
969    '
970    '
971    '
972    '
973    '
974    '
975    '
976    '
977    '
978    '
979    '
980    '
981    '
982    '
983    '
984    '
985    '
986    '
987    '
988    '
989    '
990    '
991    '
992    '
993    '
994    '
995    '
996    '
997    '
998    '
999    '
1000   '

```



```

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                         ' . $this->fourStar/$currentTotal*100 . '
108                         %"></div>
109             </div>
110         </div>
111         <div class="pull-right" style="margin-left:10px;">' .
112             $this->fourStar . ' </div>
113     </div>
114
115     <div class="pull-left">
116         <div class="pull-left" style="width:35px; line-height
117             :1;">
118             <div style="height:9px; margin:5px 0;">3 <span class
119                 ="fa fa-star"></span></div>
120         </div>
121         <div class="pull-left" style="width:180px;">
122             <div class="progress" style="height:9px; margin:8px
123                 0;">
124                 <div class="progress-bar progress-bar-success"
125                     role="progressbar" aria-valuenow="5" aria-
126                     valuelmin="0" aria-valuemax="5" style="width:
127                         ' . $this->threeStar/$currentTotal*100 . '

```

```

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

```

5.2 Services

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

5.2.1 Database.php

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

```
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 = $configs["Stage"]["Database"];
33         }
34
35         /**
36          * Returns the databse instance
37          * @return Database
38          */
39         public static function getInstance() {
40             if (Database::$instance == null)
41                 Database::$instance = new Database();
42             return Database::$instance;
43         }
44
45         /**
46          * Returns a new connection
47          * @return Connection
48          */
49         public function getConnection() {
50             $connectionString = 'pgsql:host=' . $this->dbConfig["Url"] . ';port=' .
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     /**
```

```

57     * Return true if the current REMOTE_ADDRESS is local
58     * @return boolean
59     */
60     private function isDebug() {
61         $whitelist = array('127.0.0.1', '::1', 'localhost');
62         return in_array($_SERVER['REMOTE_ADDR'], $whitelist);
63     }
64 }
65 }
66
67 ?>

```

5.2.2 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 ?>

```

5.2.3 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";
32             $str .= $exceptionObj->__toString();
33             $str .= "\n-----\n";
34             self::log(__DIR__ . "/../.." . Defaults::LOG_EXCEPTION_FILE, $str);
35         }
36     }
37 }

```

5.2.4 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.

```

```

15      * @param username string The user username
16      * @param password string The user password (in MD5)
17      * @return boolean
18      *
19      */
20      public static function login($username, $password) {
21          try {
22              $dbconn = Database::getInstance()->getConnection();
23              $statement = $dbconn->prepare('
24                  SELECT
25                      id,
26                      username,
27                      concat_ws('\', firstName, '\', lastname) as userfullname,
28                      firstname
29                  FROM
30                      users
31                  WHERE
32                      lower(username) = lower(:username) AND password = :password
33              ');
34              $statement->bindParam(":username", $username, PDO::PARAM_STR);
35              $statement->bindParam(":password", $password, PDO::PARAM_STR);
36              $statement->execute();
37              $result = $statement->fetch(PDO::FETCH_NUM);
38              if ($result[0] != null) {
39                  // set session username
40                  $_SESSION["UserId"] = $result[0];
41                  $_SESSION["Username"] = $result[1];
42                  $_SESSION["UserFullName"] = $result[2];
43                  $_SESSION["UserFirstName"] = $result[3];
44                  return true;
45              }
46              // destroy the session username and return false
47              unset($_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      /**

```

```

75     * Returns the current logged in user full name or null
76     * @return string
77     */
78     public static function getFullName() {
79         return self::isLoggedIn() ? $_SESSION["UserFullName"] : null;
80     }
81
82     /**
83     * Returns the current logged in user first name or null
84     * @return string
85     */
86     public static function getFirstName() {
87         return self::isLoggedIn() ? $_SESSION["UserFirstName"] : null;
88     }
89
90     /**
91     * Returns the current logged in user id or null
92     * @return integer
93     */
94     public static function getUserId() {
95         return self::isLoggedIn() ? $_SESSION["UserId"] : null;
96     }
97
98     /**
99     * Register a new user
100     * @param string username The user username
101     * @param string password The user password (in MD5)
102     * @param string firstName The user name
103     * @param string lastName The user surname
104     * @param string email The user email
105     * @param string birthdate The user birthdate
106     * @return boolean
107     */
108     public static function register($username, $password, $firstName, $lastName,
109         $email, $birthdate) {
110         try {
111             $dbconn = Database::getInstance()->getConnection();
112             $statement = $dbconn->prepare('INSERT INTO users VALUES (
113                 DEFAULT,
114                 :username,
115                 :password,
116                 :firstName,
117                 :lastName,
118                 :email,
119                 :birthdate
120             )');
121             $statement->bindParam(":username", $username, PDO::PARAM_STR);
122             $statement->bindParam(":password", $password, PDO::PARAM_STR);
123             $statement->bindParam(":firstName", $firstName, PDO::PARAM_STR);
124             $statement->bindParam(":lastName", $lastName, PDO::PARAM_STR);
125             $statement->bindParam(":email", $email, PDO::PARAM_STR);
126             $statement->bindParam(":birthdate", $birthdate, PDO::PARAM_STR);
127             $statement->execute();
128             return $statement->rowCount() == 1;
129         } catch (PDOException $e) {
130             LogsService::logException($e);
131             return false;
132         }
133     }
134

```

```

135     }
136
137 ?>

```

5.2.5 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 birthDate 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, $birthDate,
21                                         $nationality) {
22            try {
23                if (!$birthDate || strlen($birthDate))
24                    $birthDate = null;
25                if (!$nationality || strlen($nationality))
26                    $nationality = null;
27
28                $dbconn = Database::getInstance()->getConnection();
29                $authorStatment = $dbconn->prepare('
30                INSERT INTO authors VALUES (
31                    DEFAULT,
32                    :firstName,
33                    :lastName,
34                    :birthDate,
35                    :nationality
36                )
37            ');
38            $authorStatment->bindParam(":firstName", $firstName, PDO::PARAM_STR);
39            $authorStatment->bindParam(":lastName", $lastName, PDO::PARAM_STR);
40            $authorStatment->bindParam(":birthDate", $birthDate, PDO::PARAM_STR);
41            $authorStatment->bindParam(":nationality", $nationality, PDO::PARAM_STR);
42            $authorStatment->execute();
43            $isAdded = $authorStatment->rowCount() == 1;
44            if ($isAdded)
45                return $dbconn->lastInsertId("authors_id_seq");
46            else
47                return 0;
48        } catch (PDOException $e) {
49            LogsService::logException($e);
50            return 0;
51        }
52    }

```



```

51     }
52
53     /**
54     * Returns all the authors
55     * @return Array<Author>
56     */
57     public static function getAuthors() {
58         try {
59             $dbconn = Database::getInstance()->getConnection();
60             $statement = $dbconn->prepare('
61             SELECT
62                 *
63             FROM
64                 authors
65             ,
66             );
67             $statement->execute();
68             return $statement->fetchAll(PDO::FETCH_CLASS, "Author");
69         } catch (PDOException $e) {
70             LogService::logException($e);
71             return null;
72         }
73     }
74
75     /**
76     * Deletes an Author given its id
77     * @param $authorId integer
78     * @return boolean
79     */
80     public static function deleteAuthor($authorId) {
81         if ($authorId == null || $authorId < 0) {
82             return false;
83         }
84         try {
85             $dbconn = Database::getInstance()->getConnection();
86             $query = '
87             DELETE
88             FROM
89                 authors
90             WHERE
91                 id = :$authorId
92             ';
93
94             $statement = $dbconn->prepare($query);
95             $statement->bindParam(":$authorId", $authorId, PDO::PARAM_INT);
96             $result = $statement->execute();
97             return $result->rowCount() == 1;
98         } catch (PDOException $e) {
99             LogService::logException($e);
100             return null;
101         }
102     }
103 }
104 }
105
106 ?>

```

5.2.6 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 $e) {
92              LogService::logException($e);
93              return false;
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 ,

```

```

181         concat_ws('\ ', a.firstName, '\ ', a.lastname) as
            authorfullname
182     FROM
183         books AS b JOIN books_authors AS ba
184         ON ba.id_book= b.id JOIN authors AS a ON a.id= ba.id_author
185         LEFT JOIN reviews AS r ON r.id_book=b.id
186     WHERE
187         lower(b.title) LIKE lower(:key)
188         OR
189         lower(concat_ws('\ ', a.firstName, '\ ', a.lastname)) LIKE
            lower(:key)
190         OR
191         lower(b.genre) LIKE lower(:key)
192     GROUP BY
193         b.id,
194         b.title,
195         b.image,
196         b.genre,
197         authorfullname
198     ORDER BY
199         rate '. $sort . ' NULLS LAST';
200     $statement = $dbconn->prepare($query);
201     $statement->bindValue(":key", '% ' . $keyword . '%');
202     $row = $statement->execute();
203
204     $books = array();
205     while (($row = $statement->fetch(PDO::FETCH_ASSOC)) !== false) {
206         $book = new Book();
207         $book->id = $row['id'];
208         $book->title = $row['title'];
209         $book->image = $row['image'];
210         $book->genre = $row['genre'];
211         $book->author = $row['authorfullname'];
212         $book->rating = 0;
213         $books[] = $book;
214     }
215     return $books;
216 } catch (PDOException $e) {
217     LogService::logException($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         $statement = $dbconn->prepare($query);
241         $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
242         $result = $statement->execute();
243         return $result->rowCount() == 1;
244     } catch (PDOException $e) {
245         LogsService::logException($e);
246         return null;
247     }
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                 *
265             FROM
266                 books
267             WHERE
268                 id = :bookId
269         ';
270
271         $statement = $dbconn->prepare($query);
272         $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
273         $statement->execute();
274         $books = $statement->fetchAll(PDO::FETCH_CLASS, "Book");
275
276         if (count($books) > 0)
277             return $books[0];
278
279         return null;
280     } catch (PDOException $e) {
281         LogsService::logException($e);
282         return null;
283     }
284 }
285
286 /**
287  *
288  * Return all the reviews of the given book
289  * @param $bookId integer The book id
290  * @return Array<Review>
291  */
292 public static function getReviews($bookId) {
293     if ($bookId == null || $bookId < 0) {
294         return null;
295     }
296     try {
297         $dbconn = Database::getInstance()->getConnection();
298         $query = '
299             SELECT
300                 reviews.*,

```

```

301         concat_ws('\ ', users.firstname, '\ ' \ ', users.lastname)
302         as author,
303         SUM(coalesce(grades_reviews.grade,0)) as score
304     FROM
305         reviews JOIN users ON reviews.id_author = users.id
306         LEFT JOIN grades_reviews ON reviews.id_review =
307             grades_reviews.id_review
308     WHERE
309         id_book = :bookId
310     GROUP BY reviews.id_review, author
311 ';
312
313     $statement = $dbconn->prepare($query);
314     $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
315     $statement->execute();
316     $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "Review");
317
318     return $reviews;
319 } catch (PDOException $e) {
320     LogService::logException($e);
321     return null;
322 }
323
324 /**
325  * Returns true if the user has already done a review on the specified book
326  * @param integer $authorId The author id
327  * @param integer $bookId The book id
328  * @return boolean
329  */
330 public static function hasReview($authorId, $bookId) {
331     if ($bookId == null || $bookId < 0 || $authorId == null || $authorId <
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
348         $statement = $dbconn->prepare($query);
349         $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
350         $statement->bindParam(":authorId", $authorId, PDO::PARAM_INT);
351         $statement->execute();
352         return $statement->rowCount() == 1;
353     } catch (PDOException $e) {
354         LogService::logException($e);
355         return null;
356     }
357 }
358
359 /**
360  *

```

```

359     * Return the book's reviews summary
360     * @param $bookId integer The book id
361     * @return BookReviewSummary
362     */
363     public static function getBookReviewSummary($bookId) {
364         if ($bookId == null || $bookId < 0) {
365             return null;
366         }
367         try {
368             $dbconn = Database::getInstance()->getConnection();
369             $query = '
370                 SELECT
371                     COUNT(*) AS total,
372                     ROUND(AVG(grade),1) AS avarage,
373                     COUNT(CASE WHEN grade = 1 THEN grade END) AS "oneStar",
374                     COUNT(CASE WHEN grade = 2 THEN grade END) AS "twoStar",
375                     COUNT(CASE WHEN grade = 3 THEN grade END) AS "threeStar",
376                     COUNT(CASE WHEN grade = 4 THEN grade END) AS "fourStar",
377                     COUNT(CASE WHEN grade = 5 THEN grade END) AS "fiveStar"
378                 FROM
379                     reviews
380                 WHERE
381                     reviews.id_book = :bookId
382             ';
383
384             $statement = $dbconn->prepare($query);
385             $statement->bindParam(":bookId", $bookId, PDO::PARAM_INT);
386             $statement->execute();
387             $reviews = $statement->fetchAll(PDO::FETCH_CLASS, "BookReviewSummary");
388
389             if (count($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 ?>

```

5.2.7 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'];
43                 $comment->date_comment = $row['date_comment'];
44                 $comment->canEdit = $row['id_user'] == AuthenticationService::
45                     getUserId();
46                 $comment->children = self::getCommentsByParentId($row['
47                     id_comment']);
48                 $comments[] = $comment;
49             }
50             return $comments;
51         } catch (PDOException $e) {
52             LogsService::logException($e);
53             return null;
54         }
55     }
56
57     /**
58     * Add a new comment
59     * @param Comment $comment
60     * @return boolean
61     */
62     public static function addComment($comment) {
63         try {
64             $dbconn = Database::getInstance()->getConnection();
65             $statement = $dbconn->prepare('
66                 INSERT INTO COMMENTS VALUES (
67                     DEFAULT,
68                     :text,
69                     :id_user,
70                     :id_review,
71                     :id_ref_comm,
72                     :date_comment

```

```

70         ),
71     );
72     $statement->bindParam(":text", $comment->text);
73     $statement->bindParam(":id_user", $comment->id_user);
74     $statement->bindParam(":id_review", $comment->id_review);
75     $statement->bindParam(":id_ref_comm", $comment->id_ref_comm);
76     $statement->bindParam(":date_comment", $comment->date_comment);
77     $statement->execute();
78     $isAdded = $statement->rowCount() == 1;
79     if ($isAdded)
80         return $dbconn->lastInsertId("comments_id_comment_seq");
81     else
82         return 0;
83 } catch (PDOException $e) {
84     LogsService::logException($e);
85     return null;
86 }
87 }
88
89 /**
90  * Deletes a comment and all its children, given its id
91  * @param integer $commentId
92  * @return boolean
93  */
94 public static function deleteAuthor($commentId) {
95     if ($commentId == null || $commentId < 0) {
96         return false;
97     }
98     try {
99         $dbconn = Database::getInstance()->getConnection();
100         $query = '
101             DELETE
102             FROM
103                 comments
104             WHERE
105                 id_comment = :commentId
106             ';
107
108         $statement = $dbconn->prepare($query);
109         $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
110         $result = $statement->execute();
111         return $result->rowCount() == 1;
112     } catch (PDOException $e) {
113         LogsService::logException($e);
114         return null;
115     }
116 }
117
118 /**
119  * Given a parent id comment, return all the hierarchic tree
120  * @param integer $parentId
121  * @return Array<Comment>
122  */
123 public static function getCommentsByParentId($parentId) {
124     try {
125         $dbconn = Database::getInstance()->getConnection();
126         $statement = $dbconn->prepare('
127             SELECT
128                 c.*,
129                 COALESCE(SUM(gc.grade),0) as score,
130                 concat_ws('\', u.firstName, '\', u.lastname) as

```

```

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 }
164
165 /**
166  * Adds a comment grade (+1/-1)
167  * @param integer $commentId The comment id
168  * @param integer $userId The user id
169  * @param integer $grade The grade
170  * @return boolean
171  */
172 public static function addCommentGrade($commentId, $userId, $grade) {
173     try {
174         $dbconn = Database::getInstance()->getConnection();
175
176         $statement = $dbconn->prepare('
177             SELECT
178             *
179             FROM
180             grades_comments
181             WHERE
182                 id_user = :userId
183                 AND
184                 id_comment = :commentId
185         ');
186         $statement->bindParam(":userId", $userId, PDO::PARAM_INT);
187         $statement->bindParam(":commentId", $commentId, PDO::PARAM_INT);
188         $statement->execute();
189         $isExisting = $statement->rowCount() == 1;
190
191         if ($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 $e) {
255         LogService::logException($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["RepeatPassword"]) &&
11        isset($_POST["Email"]) &&
12        isset($_POST["FirstName"]) &&
13        isset($_POST["LastName"]) &&
14        isset($_POST["BirthDate"])) {
15        $isRegistered = AuthenticationService::register($_POST["Username"], md5(
16            $_POST["Password"], $_POST["FirstName"], $_POST["LastName"], $_POST["Email"]
17            ), $_POST["BirthDate"]);
18    }
19
20    if ($isRegistered) {
21        // return on the previous page
22        header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/register.php?signup=
23            true ');
24    } else {
25        // return on theregister page with a GET param
26        header('Location: '. Defaults::DEFAULT_BASE_URL . '/php/register.php?errors=
27            true ');
28    }
29
30 ?>

```

5.3.2 login.php

Permette ad un utente di effettuare il login.

```

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

```

```

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

```

5.3.3 logout.php

Permette ad un utente di effettuare il logout.

```

1 <?php
2  require_once(' ../ services/AuthenticationService.php');
3
4  // if logged in, destroy the session
5  if (isset($_SESSION["Username"]))
6      unset($_SESSION["Username"]);
7
8  // return on the previous page
9  header('Location: '. $_SERVER['HTTP_REFERER']);
10 ?>

```

5.3.4 add-author.php

Permette di aggiungere al database un nuovo autore di libri.

```

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

```

```

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

```

5.3.5 add-book.php

Permette di aggiungere al database un nuovo libro.

```

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

```

5.3.6 add-review.php

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

```

1 <?php
2
3     require_once(' ../ services/AuthenticationService.php ');
4     require_once(' ../ services/BooksService.php ');
5     require_once(' ../ services/Defaults.php ');
6
7     $isReviewAdded = false;
8
9     // if a username and a password is given in POST, then try to login
10    if (isset($_POST["Title"]) &&
11        isset($_POST["Text"]) &&
12        isset($_POST["Grade"]) &&
13        isset($_POST["IdBook"]) &&
14        isset($_POST["PrevUrl"]) &&
15        isset($_POST["IdAuthor"])) {

```

```

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

```

5.3.7 add-comment.php

Permette di aggiungere al database un nuovo commento.

```

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

```

5.3.8 add-grade.php

Permette di aggiungere un voto ad una recensione esistente.

```

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 Partials

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

5.4.1 navbar.php

Partial view per la navbar dell'applicazione.

```
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>
```

```

14 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#
    navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="
    false" aria-label="Toggle navigation">
15     <span class="navbar-toggler-icon"></span>
16 </button>
17 <div class="collapse navbar-collapse" id="navbarSupportedContent">
18     <ul class="navbar-nav mr-auto">
19         <li class="nav-item active">
20             <a class="nav-link" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>">
                Home <span class="sr-only">(current)</span></a>
21         </li>
22         <?php
23             if (! $isLogged) {
24                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
                    DEFAULT_BASE_URL . '/php/login.php">Login</a></li>';
25                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
                    DEFAULT_BASE_URL . '/php/register.php">Sign up</a></li>';
26             } else {
27                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
                    DEFAULT_BASE_URL . '/php/rest/logout.php">Logout</a></li>';
28                 echo '<li class="nav-item"><a class="nav-link" href="' . Defaults::
                    DEFAULT_BASE_URL . '/php/add-book.php">Add</a></li>';
29             }
30         ?>
31     </ul>
32     <?php
33         if ( $isLogged) {
34             require(__DIR__ . '/search-box.php');
35         }
36     ?>
37 </div>
38 </nav>

```

5.4.2 search-box.php

Form di ricerca libri utilizzata nella parte destra della navbar.

```

1 <!-- The navbar search box -->
2
3 <div class="input-group input-group-sm" id="searchBox">
4     <input id="keyword" type="text" class="form-control form-control-sm" value="<?
        php echo isset($_GET["keyword"]) ? $_GET["keyword"] : ""; ?>" placeholder="
        <?php echo $isLogged ? AuthenticationService::getFirstName() . " search" : "
        Search" ?> a book..." aria-label="Search for ...">
5     <span class="input-group-btn hidden" id="cancelButtonGroup">
6         <button class="btn btn-secondary btn-sm" type="button" id="cancelButton">&
            times;</button>
7     </span>
8     <span class="input-group-btn">
9         <button class="btn btn-secondary btn-sm" type="button">
10             <?php
11                 if (isset($_GET["sort"]) && $_GET["sort"] == Defaults::ASC) {
12                     echo '<i class="fa fa-sort-numeric-asc" id="bookSort"></i>';
13                 } else {
14                     echo '<i class="fa fa-sort-numeric-desc" id="bookSort"></i>';
15                 }
16             ?>
17         </button>
18     </span>
19 </div>

```

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

5.4.3 scripts.php

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

[illegible]

```

29 $(document).on("click", "#searchButton", function() {
30     // when click on the Go button
31     searchBook();
32 });
33
34 $(document).on("click", "#cancelButton", function() {
35     // when click on the X button
36     $("#keyword").val("");
37     $(this).hide();
38     $("#searchButton").click();
39 });
40
41 /**
42  * Every time a button is pressed in the input text, checks if the cancel button
43  * should be visible or not.
44  * If the button pressed is the Enter (invio), search the book with the current
45  * keyword.
46  */
47 $(document).on("keyup", "#keyword", function(e) {
48     // when enter in the keyword input text
49     checkCancelButton();
50
51     if (e.which == 13)
52         searchBook();
53 });
54
55 /**
56  * Shows or hides the cancel button if the keyword input is valorized or empty
57  */
58 function checkCancelButton() {
59     var keyword = $("#keyword").val();
60     if (!keyword || keyword.length == 0)
61         $("#cancelButtonGroup").addClass("hidden");
62     else
63         $("#cancelButtonGroup").removeClass("hidden");
64 }
65
66 /**
67  * Navigates to the index page with the get keyword param valorized with the
68  * input value
69  */
70 function searchBook() {
71     // navigate to homepage with a get parameter
72     var keyword = $("#keyword").val();
73     var currentSort = "<?php echo isset($_GET['sort']) ? $_GET['sort'] :
74         Defaults::ASC ?>";
75     var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
76     // if no value is provided, navigate without the param
77     if (!keyword || keyword.length == 0)
78         window.location.href = defaultBaseUrl + currentSort ? '?sort=' +
79             currentSort : '';
80     else
81         window.location.href = defaultBaseUrl + "?keyword=" + keyword + (
82             currentSort ? '&sort=' + currentSort : '');
83 }
84
85 /**
86  * When click on sort change button in navbar
87  */
88 $(document).on("click", "#bookSort", function() {
89     var currentSort = "<?php echo isset($_GET['sort']) ? $_GET['sort'] :

```

```

84         Defaults::DESC ?>";
85     var currentKeyword = "<?php echo isset($_GET["keyword"]) ? $_GET["keyword"]
86         : null ?>";
87     var defaultAsc = "<?php echo Defaults::ASC; ?>";
88     var defaultDesc = "<?php echo Defaults::DESC; ?>";
89     var defaultBaseUrl = "<?php echo Defaults::DEFAULT_BASE_URL; ?>";
90
91     if (!currentKeyword || currentKeyword.length == 0)
92         window.location.href = defaultBaseUrl + currentSort ? '?sort=' + (
93             currentSort == defaultDesc ? defaultAsc : defaultDesc) : '';
94     else{
95         window.location.href = defaultBaseUrl + "?keyword=" + currentKeyword +
96             (currentSort ? '&sort=' + (currentSort == defaultDesc ? defaultAsc
97                 : defaultDesc) : '');
98     }
99 }
100 });
101 // END NAVBAR SEARCH BOX
102 </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/
8     css/bootstrap.min.css">
9 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/
10     font-awesome.min.css">
11
12 <!-- Custom CSS -->
13 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
14     /custom.css">
15 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
16     /reviews.css">
17 <link rel="stylesheet" href="<?php echo Defaults::DEFAULT_BASE_URL; ?>/assets/styles
18     /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.
8             php'; ?>" method="post" name="RegisterForm">
9             <h3>Sign Up</h3>
10             <br>
11             <div class="form-group">
12                 <input type="text" class="form-control form-control-sm" name="
13                     FirstName" placeholder="First name" autofocus="true" required="
14                         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 '; ?>"
    method="post" name="LoginForm">
5     <h3>Login</h3>
6     <br>
7     <div class="form-group">
8         <input type="text" class="form-control form-control-sm" name="Username"
            placeholder="Username" required="" autofocus="" />
9     </div>
10    <div class="form-group">
11        <input type="password" class="form-control form-control-sm" name="Password"
            placeholder="Password" required="" />
12    </div>
13    <button class="btn btn-sm btn-primary" name="Submit" value="Login" type="Submit
        ">Sign in</button>
14 </form>

```

5.4.7 add-author-form.php

Form di creazione di un nuovo autore di libri.

```

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

```

```

3
4     if (!AuthenticationService::isLoggedIn()) {
5         die();
6     }
7 ?>
8 <div class="col-md-6">
9     <form action="<?php echo Defaults::DEFAULT_BASE_URL . '/php/rest/add-author.php'; ?>" method="post" name="AddAuthorForm">
10         <h3>Add author</h3>
11         <br>
12         <div class="form-group">
13             <input type="text" class="form-control form-control-sm" name="FirstName"
14                 placeholder="First name" required="true" autocomplete="off">
15         </div>
16         <div class="form-group">
17             <input type="text" class="form-control form-control-sm" name="LastName"
18                 placeholder="Last name" required="true" autocomplete="off"/>
19         </div>
20         <div class="form-group">
21             <input type="date" class="form-control form-control-sm" name="BirthDate"
22                 placeholder="BirthDate" />
23         </div>
24         <div class="form-group">
25             <input type="text" class="form-control form-control-sm" name="
26                 Nationality" placeholder="Nationality" />
27         </div>
28         <button class="btn btn-sm btn-primary" name="Submit" value="Add" type="
29             Submit">Add</button>
30     </form>
31 </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>

```

```

23     <div class="form-group">
24         <input type="text" class="form-control form-control-sm" name="Genre"
                placeholder="Genre" required="true" />
25     </div>
26     <div class="form-group">
27         <select name="Author" class="form-control">
28             <?php
29                 foreach ($authors as $author) {
30                     echo '<option value="' . $author->id . '">' . $author->
                        firstname . ' ' . $author->lastname . '</option>';
31                 }
32             ?>
33         </select>
34     </div>
35     <button class="btn btn-sm btn-primary" name="Submit" value="Add" type="
        Submit">Add</button>
36 </form>
37 </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 {{
49               window.moment(comment.date_comment).tz("Europe/
50               Rome").fromNow()}}</span>
51             <span v-else>aggiungi un nuovo commento</span>
52             <template v-if="comment.id_comment > 0">
53               <a class="btn btn-sm btn-outline-success" v-on:
54                 click="gradeUp"><i class="fa fa-thumbs-o-up"
55                 ></i></a>
56               <a class="btn btn-sm btn-outline-danger" v-on:
57                 click="gradeDown"><i class="fa fa-thumbs-o-
58                 down"></i></a>
59               <a class="btn btn-sm btn-outline-primary" v-on:
60                 click="gradeRemove"><i class="fa fa-times"></i>
61                 </a>
62               <a class="btn btn-sm btn-outline-primary">{{
63                 comment.score}}</a>
64               <button class="btn btn-sm" style="float: right"
65                 v-on:click="reply">
66                 <i class="fa fa-reply"></i>
67               </button>
68             </template>
69           </div>
70           <div class="card-block">
71             <template v-if="!(comment.id_comment > 0)">
72               <input type="text" class="form-control" v-model="
73                 comment.text" style="max-width: calc(100% - 38px
74                 ); display: inline"/>
75               <button class="btn btn-sm btn-outline-success" v-on:
76                 click="save">
77                 <i class="fa fa-save" v-if="!isLoading"></i>
78                 <i class="fa fa-circle-o-notch fa-spin" v-else
79                 ></i>
80               </button>
81             </template>
82             <p class="card-text p-1" v-else>{{comment.text}}</p>
83           </div>
84         </div>
85       </div>
86       <template v-if="comment.children && comment.children.length > 0">
87         >
88         <comment
89           v-for="child in comment.children"
90           v-bind:comment="child"
91           v-bind:level="level+1"
92           v-bind:key="child.id_comment">
93       </comment>
94     </template>
95   </div>
96 `
97   },

```

```

81     methods: {
82         save: function () {
83             if (this.comment.text && this.comment.text.length) {
84                 this.isLoading = true;
85                 this.$http.post(
86                     "<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/add-
                        comment.php",
87                     this.comment
88                 ).then(data => data.json())
89                     .then(data => {
90                         if (data != null) {
91                             this.comment.id_comment = data;
92                             this.refresh();
93                             this.isLoading = false;
94                         }
95                     }).catch((ex) => {
96                         console.error(ex);
97                         this.isLoading = false;
98                     });
99             }
100         },
101         refresh() {
102             this.$forceUpdate();
103         },
104         reply() {
105             if (!this.comment.children)
106                 this.comment.children = [];
107             if (this.comment.children.length == 0 || (this.comment.children[this
                .comment.children.length - 1].id_comment > 0)) {
108                 // push?
109                 this.comment.children.unshift({
110                     userfullname: newEmptyComment.userfullname,
111                     id_user: newEmptyComment.id_user,
112                     id_review: newEmptyComment.id_review,
113                     date_comment: newEmptyComment.date_comment,
114                     id_ref_comm: this.comment.id_comment || null,
115                     score: 0,
116                     children: []
117                 });
118                 this.refresh();
119             }
120         },
121         gradeUp() {
122             var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
                comment-grade.php?type=comment&grade=<?php echo Defaults::
                SCORE_UP ?>&userId=<?php echo AuthenticationService::getUserId()
                ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php echo
                $_SERVER["REQUEST_URI"] ?>';
123             this.$http.get(url)
124                 .then(data => data.json())
125                 .then(data => {
126                     if (data != null)
127                         this.comment.score = data;
128                 });
129         },
130         gradeDown() {
131             var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
                comment-grade.php?type=comment&grade=<?php echo Defaults::
                SCORE_DOWN ?>&userId=<?php echo AuthenticationService::getUserId()
                ?>&commentId=' + this.comment.id_comment + '&prevUrl=<?php
                echo $_SERVER["REQUEST_URI"] ?>';

```

```

132         this.$http.get(url)
133             .then(data => data.json())
134             .then(data => {
135                 if (data != null)
136                     this.comment.score = data;
137             });
138     },
139     gradeRemove() {
140         var url = '<?php echo Defaults::DEFAULT_BASE_URL ?>/php/rest/add-
            comment-grade.php?type=comment&grade=<?php echo Defaults::
            SCORE_REMOVE ?>&userId=<?php echo AuthenticationService::
            getUserId() ?>&commentId=' + this.comment.id_comment + '&prevUrl
            =<?php echo $_SERVER["REQUEST_URI"] ?>';
141         this.$http.get(url)
142             .then(data => data.json())
143             .then(data => {
144                 if (data != null)
145                     this.comment.score = data;
146             });
147     }
148 },
149 data: function () {
150     return {
151         isLoading: false
152     }
153 }
154 });
155 var commentsSection = new Vue({
156     el: '#commentsList',
157     data: {
158         comments: [],
159         emptyMessage: "No comments found"
160     }
161 });
162 var newEmptyComment = null;
163 $(document).on("click", ".comments-dialog-opener", function () {
164     var reviewId = $(this).attr("data-review-id");
165
166     newEmptyComment = {
167         userfullname: "<?php echo AuthenticationService::getFullName() ?>",
168         id_user: <?php echo AuthenticationService::getUserId() ?>,
169         id_review: Number(reviewId),
170         date_comment: moment().tz("Europe/Rome").format(),
171         score: 0
172     };
173     commentsSection.comments = [];
174     $.getJSON("<?php echo Defaults::DEFAULT_BASE_URL; ?>/php/rest/get-comments.
        php?reviewId=" + reviewId, function (data) {
175         commentsSection.comments = data || [];
176         commentsSection.comments.push(newEmptyComment);
177     });
178 });
179 </script>

```