```php
<?php
  /*****************************************************************************************
   getUserIdFromDb: Variante 1: Sucht mit der Mailadresse und dem Passwort den Benutzer in der
                    Datenbank (Authentifizierung mit den Login-Daten).
                    Variante 2: Sucht mit der Mailadresse den Benutzer in der Datenbank (um
                    doppelte Einträge zu vermeiden).
   $email:          Mailadresse, die der Benutzer eingegeben hat
   $password:       Passwort, das der Benutzer eingegeben hat. Bei Variante 2 leerer String "".
   md5():           Verschlüsselt das Passwort mit md5()
   Rückgabe:        - User-ID uid, falls erfolgreich
                    - 0, falls Benutzer nicht gefunden
   *****************************************************************************************/
  function getUserIdFromDb($email, $password) {
    $db = getValue('cfg_db');
    if (strlen($password) > 0) $result = $db->query("SELECT uid FROM user WHERE email='".$email."' AND password='".md5(
    $password)."'");
    else $result = $db->query("SELECT uid FROM user WHERE email='".$email."'");
    if ($user = $result->fetchArray()) return $user[0];
    else return 0;
  }

  /*****************************************************************************************
   getUserName: Liefert den Namen der übergebenen User-ID zurück
   Hinweis:     Ist nützlich, um den Benutzer z.B. mit "Willkommen 'Marc Muster'" zu begrüssen
   $uid:        User-ID des gewünschten Benutzers
   Rückgabe:    - Name, falls vorhanden (NULL-Wert möglich)
                - Mailadresse, falls Name = NULL
                - Leerer String, falls Benutzer-ID nicht vorhanden
   *****************************************************************************************/
  function getUserName($uid) {
    $db = getValue('cfg_db');
    $result = $db->query("SELECT name, email FROM user WHERE uid=".$uid);
    if ($user = $result->fetchArray()) {
      if (strlen($user[0]) > 0) return $user[0];
      else return $user[1];
    } else return "";
  }

  /*****************************************************************************************
```

```
39      getUserNames: Liefert die Namen aller registrierter Benutzer zurück
40      Hinweis:      Jeder Benutzer hat einen Blog, der auf seinen Namen lautet. Mit der Liste können
41                    demzufolge alle Blogs angezeigt werden. Die Funktion könnte auch getBlogs() heissen.
42      Rückgabe:     2-dimensionales Array,
43                    - 1. Dimension = Benutzer
44                    - 2. Dimension = Attribute des Benutzers
45                       * Name, falls vorhanden (NULL-Wert möglich)
46                       * Mailadresse, falls Name = NULL
47      Sortierung:   1. nach Name und 2. nach Mailadresse
48      ********************************************************************************************/
49     function getUserNames() {
50       $alle = [];
51       $db = getValue('cfg_db');
52       $users = $db->query("SELECT uid, name, email FROM user ORDER BY name, email");
53       while ($user = $users->fetchArray()) {
54         if (strlen($user[1]) > 0) $name = $user[1];
55         else $name = $user[2];
56         $alle[] = array($user[0], $name);
57       }
58       return $alle;
59     }
60
61     /********************************************************************************************
62      getUsers: Liefert alle registrierten Benutzer zurück
63      Hinweis:  Diese Funktion kann dazu benutzt werden, alle Benutzer in eine Textdatei zu exportieren
64      Rückgabe: 2-dimensionales Array,
65                - 1. Dimension = Benutzer
66                - 2. Dimension = Attribute des Benutzers
67                   * User-ID
68                   * Name, falls vorhanden (NULL-Wert möglich)
69                   * Mailadresse
70                   * md5-verschlüsseltes Passwort
71      Sortierung:   1. nach Name und 2. nach Mailadresse
72      ********************************************************************************************/
73     function getUsers() {
74       $alle = [];
75       $db = getValue('cfg_db');
76       $users = $db->query("SELECT uid, name, email, password FROM user ORDER BY name, email");
77       while ($user = $users->fetchArray()) {
```

```php
        $alle[] = $user;
      }
      return $alle;
    }

    /*****************************************************************************************
     addUser:   Schreibt einen neuen Benutzer in die Datenbank
     Hinweis:   Diese Funktion kann dazu benutzt werden, Benutzer aus einer Textdatei zu importieren
     $name:     Name des Benutzers, kann leer sein (NULL oder leerer String)
     $email:    Mailadresse des Benutzers, NOT NULL
     $passwort: Verschlüsseltes Passwort des Benutzers, NOT NULL
     Rückgabe: - True bei Erfolg
               - False bei Fehler
     *****************************************************************************************/
    function addUser($name, $email, $password) {
      $db = getValue('cfg_db');
      $name = SQLite3::escapeString($name);
      $email = SQLite3::escapeString($email);
      $sql = "INSERT INTO user (name, email, password) values ('$name', '$email', '$password')";
      return $db->exec($sql);
    }

    /*****************************************************************************************
     getEntries: Liefert alle Beiträge eines Benutzers/Blogs zurück
     Hinweis:    Möglichkeit 1: Es werden in einem ersten Schritt nur die Titel der Beiträge angezeigt. In diesem
                 Fall sind nur Entry-ID, Datum und Titel relevant.
                 Möglichkeit 2. Es werden gleich alle Blog-Beiträge untereinander angezeigt.
     $uid:       User-ID des gewünschten Benutzers
     Rückgabe:   2-dimensionales Array,
                 - 1. Dimension = Blog-Beitrag
                 - 2. Dimension = Attribute des Beitrags
                      * Entry-ID
                      * Datum als Unix-Timestamp (muss mit der Funktion date() in ein lesbares
                        Datum umgewandelt werden)
                      * Titel
                      * Inhalt (der eigentliche Beitrag)
                      * Pfad und Dateiname der Bilder 1-3
     Sortierung: Nach Entry-ID absteigend (d.h. der aktuellste zuerst)
     *****************************************************************************************/
```

```php
117     function getEntries($uid) {
118       $alle = [];
119       $db = getValue('cfg_db');
120       $entries = $db->query("SELECT eid, datetime, title, content, picture1, picture2, picture3 FROM entry WHERE uid=$uid ORDER
          BY eid DESC");
121       while ($entry = $entries->fetchArray()) {
122         $alle[] = $entry;
123       }
124       return $alle;
125     }
126
127     /*****************************************************************************************
128      getEntriesTheme: Siehe Beschreibung "getEntries"
129      Unterschied:     Es werden alle Beiträge eines Blogs zu einem bestimmten Thema zurückgegeben
130      $tid:            Thema-ID (damit wird die Abfrage auf das gewünschte Thema eingeschränkt)
131      *****************************************************************************************/
132     function getEntriesTheme($uid, $tid) {
133       $alle = [];
134       $db = getValue('cfg_db');
135       $entries = $db->query("SELECT eid, datetime, title, content, picture1, picture2, picture3 FROM entry WHERE uid=$uid AND
          tid=$tid ORDER BY eid DESC");
136       while ($entry = $entries->fetchArray()) {
137         $alle[] = $entry;
138       }
139       return $alle;
140     }
141
142     /*****************************************************************************************
143      getEntry: Liefert einen bestimmten Beitrag zurück
144      Hinweis:  Falls in einem ersten Schritt nur die Titel der Beiträge angezeigt werden, kann mit
145                dieser Funktion ein einzelner Beitrag zur Anzeige zurückgeliefert werden.
146      $eid:     Entry-ID eines Blog-Beitrags
147      Rückgabe:   1-dimensionales Array (Attribute des Beitrags)
148                       * Entry-ID
149                       * Datum als Unix-Timestamp (muss mit der Funktion date() in ein lesbares
150                         Datum umgewandelt werden)
151                       * Titel
152                       * Inhalt (der eigentliche Beitrag)
153                       * Pfad und Dateiname der Bilder 1-3
```

```
154      ********************************************************************************************/
155     function getEntry($eid) {
156       $db = getValue('cfg_db');
157       $result = $db->query("SELECT eid, datetime, title, content, picture1, picture2, picture3 FROM entry WHERE eid=$eid");
158       if ($entry = $result->fetchArray()) {
159         return $entry;
160       } else return "";
161     }
162
163     /********************************************************************************************
164      addEntry: Schreibt einen neuen Beitrag in die Datenbank, mit den min. erforderlichen Attributen
165      $uid:     User-ID - Jeder Beitrag muss einem Benutzer/Blog zugeordnet werden
166      $title:   Der Titel des Beitrags
167      $content: Der Inhalt des Beitrags
168      time():   Erstellt den aktuellen UNIX-Timestamp
169      Rückgabe: - True bei Erfolg
170                - False bei Fehler
171      ********************************************************************************************/
172     function addEntry($uid, $title, $content) {
173       $db = getValue('cfg_db');
174       $title = SQLite3::escapeString($title);
175       $content = SQLite3::escapeString($content);
176       $sql = "INSERT INTO entry (uid, datetime, title, content) values ($uid, ".time().", '$title', '$content')";
177       return $db->exec($sql);
178     }
179
180     /********************************************************************************************
181      addEntryExtended: Schreibt einen neuen Beitrag in die Datenbank, mit allen Attributen
182      $uid:             User-ID - Jeder Beitrag muss einem Benutzer/Blog zugeordnet werden
183      $tid:             Thema-ID (Falls kein Thema eingefügt wird, dann muss für $tid der String
184                        "NULL" übergeben werden)
185      $title:           Der Titel des Beitrags
186      $content:         Der Inhalt des Beitrags
187      $picture1:        Pfad + Dateiname des Bildes 1-3 (Falls kein Bild 1 eingefügt wird, dann muss
188                        für $picture1 ein leerer String "" übergeben werden - analog Bilder 2 und 3)
189      Rückgabe:         - True bei Erfolg
190                        - False bei Fehler
191      ********************************************************************************************/
192     function addEntryPlus($uid, $tid, $title, $content, $picture1, $picture2, $picture3) {
```

```php
    $db = getValue('cfg_db');
    $title = SQLite3::escapeString($title);
    $content = SQLite3::escapeString($content);
    $picture1 = SQLite3::escapeString($picture1);
    $picture2 = SQLite3::escapeString($picture2);
    $picture3 = SQLite3::escapeString($picture3);
    $sql = "INSERT INTO entry (uid, tid, datetime, title, content, picture1, picture2, picture3) values ($uid, $tid, ".time().
    ", '$title', '$content', '$picture1', '$picture2', '$picture3')";
    return $db->exec($sql);
  }

  /*****************************************************************************************
   updateEntry: Schreibt Änderungen eines bestehenden Blog-Beitrags in die DB - minimale Variante
   $eid:        Entry-ID des zu ändernden Beitrags
   $title:      Der Titel des Beitrags
   $content:    Der Inhalt des Beitrags
   Rückgabe:    - True bei Erfolg
                - False bei Fehler
   *****************************************************************************************/
  function updateEntry($eid, $title, $content) {
    $db = getValue('cfg_db');
    // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
    // UPDATE-Statement liefert auch TRUE zurück, wenn die Entry-ID nicht vorhanden ist
    $result = $db->query("SELECT * FROM entry WHERE eid=$eid");
    if ($entry = $result->fetchArray()) {
      $title = SQLite3::escapeString($title);
      $content = SQLite3::escapeString($content);
      $sql = "UPDATE entry set title='$title', content='$content' WHERE eid=$eid";
      return $db->exec($sql);
    } return false;
  }

  /*****************************************************************************************
   deleteEntry: Löscht einen bestimmten Blog-Beitrag aus der Datenbank
   $eid:        Entry-ID des zu löschenden Beitrags
   Rückgabe:    - True bei Erfolg
                - False bei Fehler
   *****************************************************************************************/
  function deleteEntry($eid) {
```

```php
    $db = getValue('cfg_db');
    // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
    // DELETE-Statement liefert auch TRUE zurück, wenn die Entry-ID nicht vorhanden ist
    $result = $db->query("SELECT * FROM entry WHERE eid=$eid");
    if ($entry = $result->fetchArray()) {
      $sql = "DELETE FROM entry WHERE eid=$eid";
      return $db->exec($sql);
    } false;
  }

  /*****************************************************************************************
   getComments: Liefert alle Kommentare eines Blog-Beitrags zurück
   $eid:        Entry-ID des gewünschten Beitrags
   Rückgabe:   2-dimensionales Array,
               - 1. Dimension = Kommentar
               - 2. Dimension = Attribute des Kommentars
                    * Comment-ID
                    * Entry-ID
                    * User-ID (> 0 falls Kommentare den Benutzern zugeordnet werden)
                    * Datum als Unix-Timestamp (muss mit der Funktion date() in ein lesbares
                      Datum umgewandelt werden)
                    * Der Inhalt des Kommentars
                    * Name des Kommentarerstellers (falls Kommentare nicht den registrierten
                                                    Benutzern zugeordnet werden)
                    * Zufallszahl, die fürs Löschen von Kommentaren verwendet werden kann
                      (falls Kommentare nicht den registrierten Benutzern zugeordnet werden)
   Sortierung: Nach Entry-ID absteigend (d.h. der aktuellste zuerst)
   *****************************************************************************************/
  function getComments($eid) {
    $alle = [];
    $db = getValue('cfg_db');
    $comments = $db->query("SELECT cid, eid, uid, date, content, name, randomnr FROM comment WHERE eid=$eid ORDER BY cid DESC"
    );
    while ($comment = $comments->fetchArray()) {
      $alle[] = $entry;
    }
    return $alle;
  }

```

```
269     /*****************************************************************************************
270      addComment: Schreibt einen neuen Kommentar in die DB, Variante 1
271      Variante 1: Der Benutzer muss angemeldet sein, um einen Kommentar zu schreiben. In diesem
272                  Fall wird die User-ID als Fremdschlüssel in die Tabelle geschrieben.
273      $eid:       Entry-ID - ID des Beitrgs, zu dem der Kommentar geschrieben wird
274      $uid:       User-ID - ID des Benutzers, der den Kommentar schreibt
275      $content:   Der Inhalt des Beitrags
276      time():     Erstellt den aktuellen UNIX-Timestamp
277      Rückgabe:   - True bei Erfolg
278                  - False bei Fehler
279      *****************************************************************************************/
280     function addComment($eid, $uid, $content) {
281       $db = getValue('cfg_db');
282       $content = SQLite3::escapeString($content);
283       $sql = "INSERT INTO comment (eid, uid, datetime, content) values ($eid, $uid, ".time().", '$content')";
284       return $db->exec($sql);
285     }
286
287     /*****************************************************************************************
288      addCommentNoUser: Schreibt einen neuen Kommentar in die DB, Variante 2
289      Variante 2:       Der Benutzer, der einen Kommentar schreibt, ist nicht angemeldet. In diesem
290                        Fall muss er einen Namen angeben und es wird eine Zufallszahl generiert und
291                        zurückgegeben, damit der Kommentar später gelöscht werden kann.
292      $eid:             Entry-ID - ID des Beitrgs, zu dem der Kommentar geschrieben wird
293      $content:         Der Inhalt des Beitrags
294      Rückgabe:         - Bei Erfolg wird die Zufallszahl zurückgegeben
295                        - Bei Fehler wird 0 zurückgegeben
296      *****************************************************************************************/
297     function addCommentNoUser($eid, $name, $content) {
298       $db = getValue('cfg_db');
299       $name = SQLite3::escapeString($name);
300       $content = SQLite3::escapeString($content);
301       $randomnr = mt_rand();
302       $sql = "INSERT INTO comment (eid, datetime, name, content, randomnr) values ($eid, ".time().", '$name', '$content',
          $randomnr)";
303       if ($db->exec($sql)) return $randomnr;
304       else return 0;
305     }
306
```

```
307     /*****************************************************************************************
308      deleteComment: Löscht einen bestimmten Kommentar zu einem Blog-Beitrag aus der Datenbank
309      Variante 1:    Der Benutzer muss angemeldet sein, um einen von ihm erstellten Kommentar zu
310                     löschen.
311      $cid:          Comment-ID des zu löschenden Kommentars
312      Rückgabe:      - True bei Erfolg
313                     - False bei Fehler
314      *****************************************************************************************/
315     function deleteComment($cid) {
316       $db = getValue('cfg_db');
317       // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
318       // DELETE-Statement liefert auch TRUE zurück, wenn die Comment-ID nicht vorhanden ist
319       $result = $db->query("SELECT * FROM comment WHERE cid=$cid");
320       if ($comment = $result->fetchArray()) {
321         $sql = "DELETE FROM comment WHERE cid=$cid";
322         return $db->exec($sql);
323       } false;
324     }
325
326     /*****************************************************************************************
327      deleteCommentNoUser: Löscht einen bestimmten Kommentar zu einem Blog-Beitrag aus der Datenbank
328      Variante 2:          Beim Erstellen eines Kommentars hat der Benutzer eine Zufallszahl erhalten.
329                           Mithilfe dieser Zahl kann er seinen Kommentar wieder löschen.
330      $cid:                Comment-ID des zu löschenden Kommentars
331      $randomnr:           Die Zufallszahl
332      Rückgabe:            - True bei Erfolg
333                           - False bei Fehler
334      *****************************************************************************************/
335     function deleteCommentPlus($cid, $randomnr) {
336       $db = getValue('cfg_db');
337       // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
338       // DELETE-Statement liefert auch TRUE zurück, wenn die Comment-ID nicht vorhanden ist
339       $result = $db->query("SELECT * FROM comment WHERE cid=$cid AND randomnr=$randomnr");
340       if ($comment = $result->fetchArray()) {
341         $sql = "DELETE FROM comment WHERE cid=$cid AND randomnr=$randomnr";
342         return $db->exec($sql);
343       } false;
344     }
345
```

```
346     /*****************************************************************************************
347      getTopics:   Liefert alle Themen eines Benutzers zurück
348      $uid:        User-ID des gewünschten Benutzers
349      Rückgabe:    2-dimensionales Array,
350                   - 1. Dimension = Thema
351                   - 2. Dimension = Attribute des Themas
352                       * Topic-ID
353                       * User-ID
354                       * Name bzw. Bezeichnung des Themas, darf nicht leer sein
355                       * Beschreibung des Themas, kann leer sein (NULL bzw. leerer String)
356      Sortierung:  Nach Name des Themas
357      *****************************************************************************************/
358     function getTopics($uid) {
359       $alle = [];
360       $db = getValue('cfg_db');
361       $topics = $db->query("SELECT tid, uid, name, description FROM topic WHERE uid=$uid ORDER BY name");
362       while ($topic = $topics->fetchArray()) {
363         $alle[] = $topic;
364       }
365       return $alle;
366     }
367
368     /*****************************************************************************************
369      getTopic: Liefert ein bestimmtes Thema zurück (z.B. zum Editieren)
370      $tid:     Topic-ID des gewünschten Themas
371      Rückgabe:   1-dimensionales Array (Attribute des Themas)
372                       * Topic-ID
373                       * User-ID
374                       * Name bzw. Bezeichnung des Themas
375                       * Beschreibung des Themas
376      *****************************************************************************************/
377     function getTopic($tid) {
378       $db = getValue('cfg_db');
379       $result = $db->query("SELECT tid, uid, name, description FROM topic WHERE tid=$tid");
380       if ($topic = $result->fetchArray()) {
381         return $topic;
382       } else return "";
383     }
384
```

```
385     /*****************************************************************************************
386      addTopic:     Schreibt ein neues Thema in die Datenbank
387      $uid:         User-ID - Jedes Thema muss einem Benutzer/Blog zugeordnet werden
388      $name:        Der Name bzw. die Bezeichnung des Themas
389      $description: Die Beschreibung des Themas
390      Rückgabe:     - True bei Erfolg
391                    - False bei Fehler
392      *****************************************************************************************/
393     function addTopic($uid, $name, $description) {
394       $db = getValue('cfg_db');
395       $name = SQLite3::escapeString($name);
396       $description = SQLite3::escapeString($description);
397       $sql = "INSERT INTO topic (uid, name, description) values ($uid, '$name', '$description')";
398       return $db->exec($sql);
399     }
400
401     /*****************************************************************************************
402      updateTopic:  Schreibt Änderungen eines bestehenden Themas in die DB
403      $tid:         Topic-ID des zu ändernden Themas
404      $name:        Der Name bzw. die Bezeichnung des Themas
405      $description: Die Beschreibung des Themas
406      Rückgabe:     - True bei Erfolg
407                    - False bei Fehler
408      *****************************************************************************************/
409     function updateTopic($tid, $name, $description) {
410       $db = getValue('cfg_db');
411       // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
412       // UPDATE-Statement liefert auch TRUE zurück, wenn die Topic-ID nicht vorhanden ist
413       $result = $db->query("SELECT * FROM topic WHERE tid=$tid");
414       if ($topic = $result->fetchArray()) {
415         $title = SQLite3::escapeString($title);
416         $content = SQLite3::escapeString($content);
417         $sql = "UPDATE topic set name='$name', description='$description' WHERE tid=$tid";
418         return $db->exec($sql);
419       } return false;
420     }
421
422     /*****************************************************************************************
423      deleteEntry: Löscht einen bestimmten Blog-Beitrag aus der Datenbank
```

```
424      $eid:        Entry-ID des zu löschenden Beitrags
425      Rückgabe:    - True bei Erfolg
426                   - False bei Fehler
427      ****************************************************************************************/
428     function deleteTopic($tid) {
429       $db = getValue('cfg_db');
430       // Zuerst wird mit einem SELECT sichergestellt, dass der Datensatz existiert, denn das
431       // DELETE-Statement liefert auch TRUE zurück, wenn die Topic-ID nicht vorhanden ist
432       $result = $db->query("SELECT * FROM topic WHERE tid=$tid");
433       if ($topic = $result->fetchArray()) {
434         $sql = "DELETE FROM topic WHERE tid=$tid";
435         return $db->exec($sql);
436       } false;
437     }
438   ?>
439
```