

```
1  <?php
2
3  /*
4   Fichier: function.php
5   Auteur: Kevin Zaffino
6   Date: 15/06/2016
7   Version:1.10
8   Description: Contient les différentes fonctions de traitement
9   Copyright (TPI 2016 - Kevin Zaffino © 2016)
10  */
11
12  require_once './phpScript/mysql.inc.php';
13
14  /**
15   * Créé un pointeur sur la base
16   * @staticvar type $dbc
17   * @return PDO
18   */
19  function &myDatabase() {
20      static $dbc = null;
21
22      if ($dbc == null) {
23          try {
24              $dbc = new PDO('mysql:host=' . DB_HOST . ';dbname=' . DB_NAME, DB_USER,
25                  DB_PASSWORD, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8", PDO
26                      ::ATTR_PERSISTENT => true));
27          } catch (Exception $e) {
28              echo 'Erreur : ' . $e->getMessage() . '<br />';
29              echo 'N : ' . $e->getCode();
30              die('Could not connect to MySQL');
31          }
32      }
33      return $dbc;
34  }
35
36  //Gestion des Utilisateurs
37
38  /**
39   * Ajoute une adresse dans la base
40   * @staticvar type $ps_adress
41   * @staticvar type $ps_id
42   * @param string $country
43   * @param string $city
44   * @param string $street
45   * @return int => le dernier id ajouté dans la base
46   */
47  function addAddress($country, $city, $street) {
48      //Prepared statements
49      static $ps_adress = null;
50      static $ps_id = null;
51
52      //query
53      $sql_adress = 'insert into adress (adress.city,adress.street,adress.country_iso)
54          values (:city,:street,:iso)';
```

```

55     if ($ps_adress == null) {
56         $ps_adress = myDatabase()->prepare($sql_adress);
57     }
58     if ($ps_id == null) {
59         $ps_id = myDatabase()->prepare('SELECT LAST_INSERT_ID()');
60     }
61
62     try {
63         $ps_adress->bindParam(':city', $city, PDO::PARAM_STR);
64         $ps_adress->bindParam(':street', $street, PDO::PARAM_STR);
65         $ps_adress->bindParam(':iso', $country, PDO::PARAM_STR);
66         $ps_adress->execute();
67
68         $ps_id->execute();
69         $last_id = $ps_id->fetchAll(PDO::FETCH_NUM);
70
71         return $last_id[0][0];
72     } catch (PDOException $e) {
73         $isok = null;
74     }
75     return $isok;
76 }
77
78 /**
79  * Insere un nouvel utilisateur dans la base de données
80  * @param string $lastName      Le nom de famille
81  * @param string $firstName     Le prénom
82  * @param string $gender        Le genre
83  * @param string $mail          Le email
84  * @param string $pwd           Le mot de passe pas encore crypté
85  * @param string $phone         Le numero de telephone
86  * @param string $country       Le code ISO du pays
87  * @param string $city          Le nom de la ville
88  * @param string $street        Le nom de la rue
89  * @param int $image            L'id de l'image
90  * @return boolean True si correctement ajouté, autrement False.
91  */
92 function insertUser($lastName, $firstName, $gender, $mail, $pwd, $phone, $country,
93 $city, $street, $image = DEFAULT_IMAGE_ID) {
94     //Prepared statements
95     static $ps_user = null;
96
97     //Query
98     $sql_user = 'insert into users
99 (users.firstname,users.lastname,users.gender,users.mail,users.phone,users.banned,u
100 sers.password,users.id_Images,users.id_Adress, users.privilege) values
101 (:firstname,:lastname,:gender,:mail,:phone,0,:password,:idimage,:idaddress,
102 :privilege)';
103
104     if ($ps_user == null) {
105         $ps_user = myDatabase()->prepare($sql_user);
106     }
107
108     //ajout et récupération de l'identifiant de l'adresse
109     $id_adress = intval(addAddress($country, $city, $street));

```

```

107     //cryptage du mot de passe
108     $pwd_shal = sha1($pwd);
109
110     $priv = PRIV_USER;
111
112     try {
113
114         $sps_user->bindParam(':firstname', $firstName, PDO::PARAM_STR);
115         $sps_user->bindParam(':lastname', $lastName, PDO::PARAM_STR);
116         $sps_user->bindParam(':gender', $gender, PDO::PARAM_INT);
117         $sps_user->bindParam(':mail', $mail, PDO::PARAM_STR);
118         $sps_user->bindParam(':phone', $phone, PDO::PARAM_STR);
119         $sps_user->bindParam(':password', $pwd_shal, PDO::PARAM_STR);
120         $sps_user->bindParam(':idimage', $image, PDO::PARAM_INT);
121         $sps_user->bindParam(':idadress', $id_adress, PDO::PARAM_INT);
122         $sps_user->bindParam(':privilege', $priv, PDO::PARAM_INT);
123
124         $isok = $sps_user->execute();
125     } catch (PDOException $e) {
126         $isok = false;
127     }
128     return $isok;
129 }
130
131 /**
132  * Verification des identifiants de l'utilisateur
133  * @staticvar type $ps
134  * @param string $mail      L'adresse email
135  * @param string $pwd        Le mot de passe pas encore crypté
136  * @return boolean
137  */
138 function login($mail, $pwd) {
139
140     //Prepared statement
141     static $ps = null;
142
143     //Query
144     $sql = 'select users.id, users.password from users where users.mail = :mail';
145
146     if ($ps == null) {
147         $ps = myDatabase()->prepare($sql);
148     }
149
150     //cryptage du mot de passe
151     $pwd_shal = sha1($pwd);
152
153     try {
154         $ps->bindParam(':mail', $mail, PDO::PARAM_STR);
155         $isok = $ps->execute();
156         $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
157         if (isset($isok[0])) {
158             if ($isok[0]['password'] == $pwd_shal) {
159                 $isok = intval($isok[0]['id']);
160             } else {
161                 $isok = FALSE;
162             }
163         } else {

```

```

164         $isok = FALSE;
165     }
166 } catch (PDOException $e) {
167     $isok = false;
168 }
169 return $isok;
170 }
171
172 /**
173  * Deconnecte l'utilisateur
174  */
175 function logOut() {
176     destroySession();
177     header('Location: index.php');
178 }
179
180 /**
181  * Retourne un tableau avec les infos de l'utilisateur
182  * @staticvar type $ps
183  * @param int $id    Id de l'utilisateur
184  * @return Array     Tableau associatif contenant les données de l'utilisateur
185  */
186 function getUserInfo($id) {
187     //Prepared statement
188     static $ps = null;
189
190     //Query
191     $sql = 'select * from user_info where user_info.id = :id';
192
193     if ($ps == null) {
194         $ps = myDatabase()->prepare($sql);
195     }
196
197     try {
198         $ps->bindParam(':id', $id, PDO::PARAM_INT);
199
200         $isok = $ps->execute();
201         $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
202         $isok = $isok[0];
203     } catch (PDOException $e) {
204         $isok = false;
205     }
206     return $isok;
207 }
208
209 /**
210  * Inverse l'etat de bannissement de l'utilisateur
211  * @staticvar type $ps
212  * @param int $uid    Identifiant de l'utilisateur
213  * @return boolean
214  */
215 function banUnbanUser($uid) {
216     $banstate = getUserInfo($uid)['banned'];
217     $finalestate = ($banstate == "0") ? 1 : 0;
218
219     //Prepared statement
220     static $ps = null;

```

```

221
222     //Query
223     $sql = 'UPDATE `users` SET `banned` = :ban WHERE `id` = :id';
224     if ($ps == null) {
225         $ps = myDatabase()->prepare($sql);
226     }
227     try {
228         $ps->bindParam(':ban', $finalestate, PDO::PARAM_INT);
229         $ps->bindParam(':id', $uid, PDO::PARAM_INT);
230         $isok = $ps->execute();
231     } catch (PDOException $e) {
232         $isok = false;
233     }
234     return $isok;
235 }
236
237 /**
238  * Modifie les informations de l'utilisateur
239  * @staticvar type $ps
240  * @param string $lastName      Nom de l'utilisateur
241  * @param string $firstName     Prename de l'utilisateur
242  * @param Bool $gender          Genre de l'utilisateur (0 pour Femme et 1 pour Homme)
243  * @param string $mail          Email de l'utilisateur
244  * @param string $phone         Numero de telephone de l'utilisateur
245  * @param int $id               Identifiant d el'utilisateur
246  * @return boolean
247  */
248 function updateUserInfo($lastName, $firstName, $gender, $mail, $phone, $id) {
249
250     //Prepared statement
251     static $ps = null;
252
253     //Query
254     $sql = "update users set users.firstname = :firstname ,users.lastname =
:lastname ,users.gender = :gender ,users.mail = :mail ,users.phone = :phone
where users.id = :id";
255
256     if ($ps == null) {
257         $ps = myDatabase()->prepare($sql);
258     }
259
260     try {
261         $ps->bindParam(':firstname', $firstName, PDO::PARAM_STR);
262         $ps->bindParam(':lastname', $lastName, PDO::PARAM_STR);
263         $ps->bindParam(':gender', $gender, PDO::PARAM_INT);
264         $ps->bindParam(':mail', $mail, PDO::PARAM_STR);
265         $ps->bindParam(':phone', $phone, PDO::PARAM_STR);
266
267         $ps->bindParam(':id', $id, PDO::PARAM_INT);
268
269         $isok = $ps->execute();
270     } catch (PDOException $e) {
271         $isok = false;
272     }
273     return $isok;
274 }
275

```

```
276  /**
277  * Modifie le chemin de l'avatar de l'utilisateur
278  * @staticvar type $ps
279  * @param int $id
280  * @param type $path
281  * @return boolean
282  */
283  function updateUserImage($id, $path) {
284      static $ps = null;
285
286      $sql = "update images set images.path = :path where images.id = :id";
287
288      if ($ps == null) {
289          $ps = myDatabase()->prepare($sql);
290      }
291
292      try {
293          $ps_user->bindParam(':path', $city, PDO::PARAM_STR);
294          $ps_user->bindParam(':id', $id, PDO::PARAM_INT);
295
296          $isok = $ps_user->execute();
297      } catch (PDOException $e) {
298          $isok = false;
299      }
300      return $isok;
301  }
302
303  /**
304  * Modifie l'adresse d'un utilisateur
305  * @staticvar type $ps
306  * @param int $id                Identifiant de l'utilisateur
307  * @param string $countryiso      Code ISO du pays
308  * @param string $city            Nom de la ville
309  * @param string $street          Nom de la rue
310  * @return boolean
311  */
312  function updateUserAdress($id, $countryiso, $city, $street) {
313
314      //Prepared stateent
315      static $ps = null;
316
317      //Query
318      $sql = "update adress set adress.city = :city ,adress.street = :street
319      ,adress.country_iso = :iso where adress.id = :id";
320
321      if ($ps == null) {
322          $ps = myDatabase()->prepare($sql);
323      }
324
325      try {
326
327          $ps->bindParam(':city', $city, PDO::PARAM_STR);
328          $ps->bindParam(':street', $street, PDO::PARAM_STR);
329          $ps->bindParam(':iso', $countryiso, PDO::PARAM_STR);
330          $ps->bindParam(':id', $id, PDO::PARAM_INT);
331
332          $isok = $ps->execute();
```

```
332     } catch (PDOException $e) {
333         $isok = false;
334     }
335     return $isok;
336 }
337
338 /**
339  * Retourne un tableau avec tous les utilisateurs
340  * @staticvar type $ps
341  * @return array      Tableau associatif contenant tous les utilisateurs
342  */
343 function getAllUser() {
344     static $ps = null;
345
346     $sql = "SELECT id, firstname as Prenom, lastname as Nom, mail, banned as Bannis
347     FROM user_info WHERE privilege=1;";
348
349     if ($ps == null) {
350         $ps = myDatabase()->prepare($sql);
351     }
352
353     try {
354         $isok = $ps->execute();
355         $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
356     } catch (PDOException $e) {
357         $isok = false;
358     }
359
360     return $isok;
361 }
362
363 /**
364  * Modifie l'id de l'image d'un utilisateur avec un id donné
365  * @staticvar type $ps
366  * @param int $user_id      Identifiant d el'utilisateur
367  * @param int $image_id      Identifiant d el'image a attribuer
368  * @return boolean
369  */
370 function changeUserImage($user_id, $image_id) {
371     static $ps = null;
372
373     $sql = "update users set users.id_Images = :iid where users.id = :uid";
374
375     if ($ps == null) {
376         $ps = myDatabase()->prepare($sql);
377     }
378
379     try {
380         $ps->bindParam(':iid', $image_id, PDO::PARAM_INT);
381         $ps->bindParam(':uid', $user_id, PDO::PARAM_INT);
382
383         $isok = $ps->execute();
384     } catch (PDOException $ex) {
385         $isok = false;
386     }
387
388     return $isok;
389 }
```

```
388 }
389
390 /**
391  * Récupere tous les pays
392  * @staticvar type $ps
393  * @return boolean
394  */
395 function getAllCountry() {
396
397     //Prepared statement
398     static $ps = null;
399
400     //Query
401     $sql = 'select * from country order by name';
402
403     if ($ps == null) {
404         $ps = myDatabase()->prepare($sql);
405     }
406
407     try {
408         $isok = $ps->execute();
409         $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
410     } catch (PDOException $e) {
411         $isok = false;
412     }
413     return $isok;
414 }
415
416 //Gestion des annonces
417
418 /**
419  * Insert une nouvelle annonce dans la base de donnée
420  * @staticvar type $ps
421  * @param string $name          libelle de l'annonce
422  * @param string $description    Description de l'annonce
423  * @param string $price          Prix de l'annonce
424  * @param string $date          Date de creation de l'annonce
425  * @param string $uid           Identifiant du créateur de l'annonce
426  * @param bool $mailvisible     Email visible ou non
427  * @param bool $phonevisible    Numero de telephone visible ou non
428  * @param bool $adressvisible   Adresse visible ou non
429  * @return boolean
430  */
431 function insertArticle($name, $description, $price, $date, $uid, $mailvisible,
432 $phonevisible, $adressvisible) {
433
434     //Prepared statements
435     static $ps = null;
436     static $ps_id = null;
437
438     //Query
439     $sql = 'insert into articles
440 (name,description,price,state,creationdate,banned,id_Users,mailvisible,phonevisibl
441 e,adressvisible) values
442 (:name,:description,:price,1,:date,0,:uid,:mvis,:pvis,:avis)';
```



```

441     if ($ps == null) {
442         $ps = myDatabase()->prepare($sql);
443     }
444
445     if ($ps_id == null) {
446         $ps_id = myDatabase()->prepare('SELECT LAST_INSERT_ID()');
447     }
448
449
450     try {
451         $ps->bindParam(':name', $name, PDO::PARAM_STR);
452         $ps->bindParam(':description', $description, PDO::PARAM_STR);
453         $ps->bindParam(':price', $price, PDO::PARAM_STR);
454         $ps->bindParam(':date', $date, PDO::PARAM_STR);
455         $ps->bindParam(':uid', $uid, PDO::PARAM_INT);
456         $ps->bindParam(':mvis', $mailvisible, PDO::PARAM_INT);
457         $ps->bindParam(':pvis', $phonevisible, PDO::PARAM_INT);
458         $ps->bindParam(':avis', $adressvisible, PDO::PARAM_INT);
459
460
461         $isok = $ps->execute();
462
463         $ps_id->execute();
464         $last_id = $ps_id->fetchAll(PDO::FETCH_NUM);
465
466         return intval($last_id[0][0]);
467     } catch (PDOException $e) {
468         $isok = false;
469     }
470     return $isok;
471 }
472
473 /**
474  * Edition des informations d'une annonce
475  * @staticvar type $ps
476  * @param int $idarticle      Identifiant de l'annonce a editer
477  * @param string $name        Libelle del'annonce
478  * @param string $description  Description del'annonce
479  * @param string $price       Prix de l'annonce
480  * @param bool $mailvisible   Email visible ou non
481  * @param bool $phonevisible  Numero de telephone visible ou non
482  * @param bool $adressvisible Adresse visible ou non
483  * @return boolean
484  */
485 function editArticleInfo($idarticle, $name, $description, $price, $mailvisible,
486 $phonevisible, $adressvisible) {
487     //Prepared statement
488     static $ps = null;
489
490     //Query
491     $sql = 'update articles set name=:name ,description=:description ,price=:price
492 ,mailvisible=:mvis, phonevisible=:pvis ,adressvisible =:avis where id=:id';
493
494     if ($ps == null) {
495         $ps = myDatabase()->prepare($sql);
496     }

```

```

496
497     try {
498
499         $ps->bindParam(':name', $name, PDO::PARAM_STR);
500         $ps->bindParam(':description', $description, PDO::PARAM_STR);
501         $ps->bindParam(':price', $price, PDO::PARAM_STR);
502         $ps->bindParam(':mvis', $mailvisible, PDO::PARAM_INT);
503         $ps->bindParam(':pvis', $phonevisible, PDO::PARAM_INT);
504         $ps->bindParam(':avis', $adressvisible, PDO::PARAM_INT);
505         $ps->bindParam(':id', $idarticle, PDO::PARAM_INT);
506
507
508         $isok = $ps->execute();
509
510         return $isok;
511     } catch (PDOException $e) {
512         $isok = false;
513     }
514     return $isok;
515 }
516
517 /**
518  * Insert une image de l'article dans la base de données
519  * @staticvar type $ps
520  * @param int $aid      Identifiant de l'article
521  * @param int $iid      Identifiant de l'image
522  * @return boolean
523  */
524 function insertArticleImage($aid, $iid) {
525     //Prepared statement
526     static $ps = null;
527
528     //Query
529     $sql = "update images set images.id_articles = :articleid where images.id =
530         :imageid";
531
532     if ($ps == null) {
533         $ps = myDatabase()->prepare($sql);
534     }
535
536     try {
537         $ps->bindParam(':articleid', $aid, PDO::PARAM_INT);
538         $ps->bindParam(':imageid', $iid, PDO::PARAM_INT);
539         $isok = $ps->execute();
540     } catch (PDOException $e) {
541         $isok = false;
542     }
543
544     return $isok;
545 }
546
547 /**
548  * Retourne les images des articles en fonction de l'id de l'article
549  * @staticvar type $ps
550  * @param int $idarticle      Identifiant de l'article
551  * @return boolean
552  */

```

```

552 function articleImages($idarticle) {
553     //Prepared statement
554     static $ps = null;
555
556     //Query
557     $sql = 'SELECT id, path FROM images where images.id_articles = :id';
558
559     if ($ps == null) {
560         $ps = myDatabase()->prepare($sql);
561     }
562
563     try {
564         $ps->bindParam(':id', $idarticle, PDO::PARAM_INT);
565         $isok = $ps->execute();
566         $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
567     } catch (PDOException $e) {
568         $isok = false;
569     }
570     return $isok;
571 }
572
573 /**
574  * Ouvre/Ferme un article
575  * @staticvar type $ps
576  * @param int $aid      Identifiant del'article
577  * @return boolean
578  */
579 function openCloseArticle($aid) {
580     //Recuperation des infos de l'article
581     $banstate = articleInfo($aid)['state'];
582     $finalestate = ($banstate == "0") ? 1 : 0;
583
584     //prepared statement
585     static $ps = null;
586
587     //Query
588     $sql = 'UPDATE `articles` SET `state` = :state WHERE `id` = :id';
589     if ($ps == null) {
590         $ps = myDatabase()->prepare($sql);
591     }
592     try {
593         $ps->bindParam(':state', $finalestate, PDO::PARAM_INT);
594         $ps->bindParam(':id', $aid, PDO::PARAM_INT);
595         $isok = $ps->execute();
596     } catch (PDOException $e) {
597         $isok = false;
598     }
599     return $isok;
600 }
601
602 /**
603  * Ban/Unban un article
604  * @staticvar type $ps
605  * @param int $aid      Identifiant de l'article
606  * @return boolean
607  */
608 function banunbanArticle($aid) {

```

```
609 //Recuperation de l'etat actuel
610 $banstate = articleInfo($aid)['banned'];
611
612 //inversement de l'etat
613 $finalestate = ($banstate == "0") ? 1 : 0;
614
615 //Prepared statement
616 static $ps = null;
617
618 //Query
619 $sql = "UPDATE `articles` SET `banned` = :ban WHERE `id` = :id";
620
621 if ($ps == null) {
622     $ps = myDatabase()->prepare($sql);
623 }
624 try {
625     $ps->bindParam(':ban', $finalestate, PDO::PARAM_INT);
626     $ps->bindParam(':id', $aid, PDO::PARAM_INT);
627     $isok = $ps->execute();
628 } catch (PDOException $e) {
629     $isok = false;
630 }
631 return $isok;
632 }
633
634 /**
635  * Recupere toutes les annonces de la base
636  * @staticvar type $ps
637  * @return boolean
638  */
639 function getAllArticles() {
640
641     //Prepared statement
642     static $ps = null;
643
644     //Query
645     $sql = 'select id, name as Libelle,description,price as Prix,state as
        Etat,creationdate as "Date de creation",banned as Bannis from articles';
646
647     if ($ps == null) {
648         $ps = myDatabase()->prepare($sql);
649     }
650
651     try {
652         $isok = $ps->execute();
653         $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
654     } catch (PDOException $e) {
655         $isok = false;
656     }
657
658     return $isok;
659 }
660
661 /**
662  * Recupere tous les articles qui sont actif et non bannis
663  * @staticvar type $ps
664  * @return boolean
```

```
665     */
666     function listArticles() {
667
668         //Prepared statement
669         static $ps = null;
670
671         //Query
672         $sql = 'SELECT * FROM articles where state = 1 and banned = 0 order by
        creationdate DESC';
673
674         if ($ps == null) {
675             $ps = myDatabase()->prepare($sql);
676         }
677
678         try {
679             $isok = $ps->execute();
680             $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
681         } catch (PDOException $e) {
682             $isok = false;
683         }
684
685         return $isok;
686     }
687
688     /**
689     * Recupere toutes les annonces d'un utilisateur en particulier
690     * @staticvar type $ps
691     * @param int $uid          Identifiant dle'utilisateur
692     * @return boolean
693     */
694     function getUserArticles($uid) {
695
696         //Prepared statement
697         static $ps = null;
698
699         //Query
700         $sql = 'SELECT * FROM articles where banned = 0 and articles.id_Users = :id
        order by creationdate';
701
702         if ($ps == null) {
703             $ps = myDatabase()->prepare($sql);
704         }
705
706         try {
707             $ps->bindParam(':id', $uid, PDO::PARAM_INT);
708             $isok = $ps->execute();
709             $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
710         } catch (PDOException $e) {
711             $isok = false;
712         }
713
714         return $isok;
715     }
716
717     /**
718     * Retourne les informations d'un article
719     * @staticvar type $ps
```

```

720  * @param int $id          $identifiant de l'article
721  * @return boolean
722  */
723  function articleInfo($id) {
724
725      //Prepared statement
726      static $ps = null;
727
728      //Query
729      $sql = 'select * from articles where id = :id';
730
731      if ($ps == null) {
732          $ps = myDatabase()->prepare($sql);
733      }
734
735      try {
736          $ps->bindParam(':id', $id, PDO::PARAM_INT);
737
738          $isok = $ps->execute();
739          $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
740          $isok = $isok[0];
741      } catch (PDOException $e) {
742          $isok = false;
743      }
744
745      return $isok;
746  }
747
748  /**
749   * Supprime toutes les images d'un article dans la base de donnée
750   * @staticvar type $ps
751   * @param int $id_article      Identifiant del'article
752   */
753  function deleteArticleImages($id_article) {
754      $articleimage = articleImages($id_article);
755
756      foreach ($articleimage as $value) {
757          $id = intval($value['id']);
758          deleteFile(deleteImageEntry($id));
759      }
760  }
761
762  //Gestion des commentaires
763
764  /**
765   * Ajoute un commentaire dans la base
766   * @staticvar type $ps
767   * @param int $uid            Identifiant de l'utilisateur
768   * @param int $aid            Identifiant de l'article
769   * @param string $date        Date du commentaire
770   * @param string $com         Texte du commentaire
771   * @return boolean
772   */
773  function addComment($uid, $aid, $date, $com) {
774
775      //Prepared statement
776      static $ps = null;

```

```

777
778     //Query
779     $sql = "insert into comments
            (comments.id_Users,comments.id_articles,comments.date_com,comments.comm,comments.s
            tate,comments.banned) values (:uid,:aid,:date,:com,1,0)";

780     if ($ps == null) {
781         $ps = myDatabase()->prepare($sql);
782     }
783
784     try {
785         $ps->bindParam(':uid', $uid, PDO::PARAM_STR);
786         $ps->bindParam(':aid', $aid, PDO::PARAM_STR);
787         $ps->bindParam(':date', $date, PDO::PARAM_STR);
788         $ps->bindParam(':com', $com, PDO::PARAM_STR);
789
790         $isok = $ps->execute();
791     } catch (PDOException $e) {
792         $isok = false;
793     }
794     return $isok;
795 }
796
797 /**
798  * Recupere les commentaires d'une annonce
799  * @staticvar type $ps
800  * @param int $aid          Identifiant de l'annonce
801  * @return boolean
802  */
803 function getArticleComments($aid) {
804
805     //Prepared statement
806     static $ps = null;
807
808     //Query
809     $sql = "select * from comments where id_articles = :id and comments.banned = 0";
810     if ($ps == null) {
811         $ps = myDatabase()->prepare($sql);
812     }
813
814     try {
815         $ps->bindParam(':id', $aid, PDO::PARAM_INT);
816         $isok = $ps->execute();
817         $isok = $isok = $ps->fetchAll(PDO::FETCH_ASSOC);
818     } catch (PDOException $e) {
819         $isok = false;
820     }
821
822     return $isok;
823 }
824
825 /**
826  * met a jour l'etat du commentaire (en attente de moderation ou non)
827  * @staticvar type $ps
828  * @param int $idcomment    Identifiant du commentaire
829  * @param bool $state        Valeur a assigner
830  * @return boolean
831  */

```

```
832 function changeComState($idcomment, $state) {
833
834     //Prepared statement
835     static $ps = null;
836
837     //Query
838     $sql = 'update comments set comments.state = :state where comments.id = :id';
839
840     if ($ps == null) {
841         $ps = myDatabase()->prepare($sql);
842     }
843
844     try {
845         $ps->bindParam(':id', $idcomment, PDO::PARAM_INT);
846         $ps->bindParam(':state', $state, PDO::PARAM_INT);
847         $isok = $ps->execute();
848     } catch (PDOException $e) {
849         $isok = false;
850     }
851
852     return $isok;
853 }
854
855 /**
856  * Bannis un commentaire
857  * @staticvar type $ps
858  * @param int $idcomment      Identifiant du commentaire
859  * @return boolean
860  */
861 function banComment($idcomment) {
862
863     //Prepared statement
864     static $ps = null;
865
866     //Query
867     $sql = 'update comments set comments.banned = 1 where comments.id = :id';
868
869     if ($ps == null) {
870         $ps = myDatabase()->prepare($sql);
871     }
872
873     try {
874         $ps->bindParam(':id', $idcomment, PDO::PARAM_INT);
875         $isok = $ps->execute();
876     } catch (PDOException $e) {
877         $isok = false;
878     }
879
880     return $isok;
881 }
882
883 //Gestion des images
884
885 /**
886  * Insert une image dans la base de données et retourne son id
887  * @staticvar type $ps_image
888  * @staticvar type $ps_id
```



```
889  * @param string $path          Chemin de l'image
890  * @param int $idarticle        (a specifier si on souhaite assigner l'image a un
    article)
891  * @return int                  Identifiant de l'image
892  */
893  function insertImage($path, $idarticle = FALSE) {
894
895      //Prepared statements
896      static $ps_image = null;
897      static $ps_id = null;
898
899
900      //Query
901      if ($idarticle) {
902          $sql_image = 'insert into images (path,images.id_articles) values
    (:path,:id)';
903      } else {
904          $sql_image = 'insert into images (path) values (:path)';
905      }
906
907      $sql_id = 'select id from images where path = :path';
908
909      if ($ps_image == NULL) {
910          $ps_image = myDatabase()->prepare($sql_image);
911      }
912
913      if ($ps_id == NULL) {
914          $ps_id = myDatabase()->prepare($sql_id);
915      }
916
917      try {
918          $ps_image->bindParam(':path', $path, PDO::PARAM_STR);
919          if ($idarticle) {
920              $ps_image->bindParam(':id', $idarticle, PDO::PARAM_INT);
921          }
922
923          $ps_image->execute();
924
925          $ps_id->bindParam(':path', $path, PDO::PARAM_STR);
926
927          $isok = $ps_id->execute();
928          $isok = $ps_id->fetchAll(PDO::FETCH_NUM);
929      } catch (PDOException $ex) {
930          $isok = false;
931      }
932
933      return $isok[0][0];
934  }
935
936  /**
937   * Upload une image
938   * @return type
939   */
940  function imageUpload() {
941
942      //Extensions autorisées
943      $extArray = array('jpg', 'gif', 'png', 'jpeg');
```

```
944
945     $ext = '';
946     $error = '';
947     $imageName = '';
948
949 //Si le dossier cible n'existe pas, alors on essaye de le créer
950     if (!is_dir(TARGET)) {
951         if (!mkdir(TARGET, 0755)) {
952             exit('Erreur : le répertoire cible ne peut-être créé ! Vérifiez que vous
                diposiez des droits suffisants pour le faire ou créez le manuellement !');
953         }
954     }
955
956 //Verification si le champ n'est pas vide
957     if (!empty($_FILES[INPUT]['name'])) {
958
959 //recupère l'extension du fichier
960         $ext = pathinfo($_FILES[INPUT]['name'], PATHINFO_EXTENSION);
961
962 //verification de l'extension
963         if (in_array(strtolower($ext), $extArray)) {
964             $infosImg = getimagesize($_FILES[INPUT]['tmp_name']);
965
966
967 //Verification du type de l'image
968             if ($infosImg[2] >= 1 && $infosImg[2] <= 14) {
969
970
971 //Verification des dimensions et de la taille
972                 if (($infosImg[0] <= WIDTH_MAX) && ($infosImg[1] <= HEIGHT_MAX) && (
                    filesize($_FILES[INPUT]['tmp_name']) <= MAX_SIZE)) {
973
974                     if (isset($_FILES[INPUT]['error']) && UPLOAD_ERR_OK === $_FILES[
                        INPUT]['error']) {
975
976                         $imageName = uniqid() . '.' . $ext;
977
978
979                     if (move_uploaded_file($_FILES[INPUT]['tmp_name'], TARGET .
                        $imageName)) {
980                         $path = TARGET . $imageName;
981
982                         //insertion dans la base de donnée et retour de l'id
983                         return insertImage($path);
984                     } else {
985                         return 'Problème lors de l\'upload !';
986                     }
987                 } else {
988                     return 'Une erreur interne a empêché l\'upload de l\'image';
989                 }
990             } else {
991                 return 'Erreur dans les dimensions de l\'image !';
992             }
993         } else {
994             return 'Le fichier à uploader n\'est pas une image !';
995         }
996     } else {
```

```

997         return 'L\'extension du fichier est incorrecte !';
998     }
999 } else {
1000     return 'Image non renseignée!';
1001 }
1002 }
1003
1004 /**
1005  * Parcours le $_FILES pour envoyer toutes les images dans un article
1006  * @param int $id_article      Identifiant de l'article
1007  * @return boolean
1008  */
1009 function multiUpload($id_article) {
1010
1011     //Nombre d'image
1012     $nb = count($_FILES[INPUT]['name']);
1013     for ($i = 0; $i < $nb; $i++) {
1014         $name = uniqid();
1015         $extension_upload = strtolower(substr(strrchr($_FILES[INPUT]['name'][$i], '.'), 1));
1016         $destination = TARGET . $name . "." . $extension_upload;
1017         //deplacement des fichiers
1018         move_uploaded_file($_FILES[INPUT]['tmp_name'][$i], $destination);
1019         //insertion de l'image dans la base
1020         insertImage($destination, $id_article);
1021     }
1022     return TRUE;
1023 }
1024
1025 /**
1026  * Modifie le chemin d'une image dans la base d edonnée
1027  * @staticvar type $ps
1028  * @param type $id_image      Identifiant de l'image a modifier
1029  * @param type $newpath       Nouveau chemin
1030  * @return boolean
1031  */
1032 function editImagePath($id_image, $newpath) {
1033     static $ps = null;
1034
1035     $sql = "update images set images.path = :path where images.id = :id";
1036
1037     if ($ps == null) {
1038         $ps = myDatabase()->prepare($sql);
1039     }
1040
1041     try {
1042         $ps->bindParam(':path', $newpath, PDO::PARAM_STR);
1043         $ps->bindParam(':id', $id_image, PDO::PARAM_INT);
1044
1045         $isok = $ps->execute();
1046     } catch (PDOException $ex) {
1047         $isok = false;
1048     }
1049
1050     return $isok;
1051 }
1052

```

```
1053  /**
1054  * Supprime une image de la base de données
1055  * @staticvar type $ps_path
1056  * @staticvar type $ps_delete
1057  * @param int $image_id      Identifiant de l'image
1058  * @return boolean
1059  */
1060  function deleteImageEntry($image_id) {
1061
1062      static $ps_path = null;
1063      static $ps_delete = null;
1064
1065      $sql_path = "SELECT path FROM images where id=:id;";
1066      $sql_delete = "DELETE from images WHERE id=:id";
1067
1068
1069      if ($ps_path == null) {
1070          $ps_path = myDatabase()->prepare($sql_path);
1071      }
1072
1073      if ($ps_delete == null) {
1074          $ps_delete = myDatabase()->prepare($sql_delete);
1075      }
1076
1077      try {
1078          $ps_path->bindParam(':id', $image_id, PDO::PARAM_INT);
1079          $path = $ps_path->execute();
1080          $path = $ps_path->fetchAll(PDO::FETCH_ASSOC);
1081
1082
1083          $ps_delete->bindParam(':id', $image_id, PDO::PARAM_INT);
1084          $delete = $ps_delete->execute();
1085      } catch (PDOException $ex) {
1086          $path = false;
1087      }
1088
1089      return $path[0]['path'];
1090  }
1091
1092  /**
1093  * Supprime un fichier
1094  * @param string $path      Emplacement du fichier a supprimer
1095  */
1096  function deleteFile($path) {
1097      if (file_exists($path)) {
1098          unlink($path);
1099      }
1100  }
1101
```