

```
1 <?php
2     require "../lib/constantes.inc.php";
3     /**
4      * Function which return a pdo object
5      * @return PDO
6      */
7     function dbConnect()
8     {
9         static $dbc = null;
10        if ($dbc == null) {
11            try {
12                $dbc = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, DBUSER,
13DBPWD, array(
14                    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
15                    PDO::ATTR_PERSISTENT => true
16                ));
17            } catch (PDOException $e) {
18                echo 'Erreur : ' . $e->getMessage() . '<br />';
19                echo 'N° : ' . $e->getCode();
20                die('Could not connect to MySQL');
21            }
22        }
23        return $dbc;
24    }
25
26    /**
27     * Function which create a user in the database
28     * @param $email
29     * @param $passwordHash
30     * @param $streetName
31     * @param $streetNumber
32     * @param $postalCode
33     * @param $city
34     * @return bool
35     */
36    function createUser($email, $passwordHash, $streetName, $streetNumber,
37$postalCode, $city)
38    {
39        $answer = false;
40        if (!checkUserExists($email))
41        {
42            static $ps = null;
43            $sql = "INSERT INTO USER (`email`, `passwordHash`, `isAdmin`,
44`streetName`, `streetNumber`, `city`, `postalCode`) VALUES (:EMAIL, :PASSWORD_HASH,
450, :STREET_NAME, :STREET_NUMBER, :CITY, :POSTAL_CODE);";
46            if ($ps == null)
47            {
48                $ps = dbConnect()->prepare($sql);
49            }
50            try {
51                $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
52                $ps->bindParam(':PASSWORD_HASH', $passwordHash, PDO::PARAM_STR);
53                $ps->bindParam(':STREET_NAME', $streetName, PDO::PARAM_STR);
54                $ps->bindParam(':STREET_NUMBER', $streetNumber, PDO::PARAM_STR);
55                $ps->bindParam(':POSTAL_CODE', $postalCode, PDO::PARAM_STR);
56                $ps->bindParam(':CITY', $city, PDO::PARAM_STR);
57
58                $answer = $ps->execute();
59            }
60        }
61    }
62}
```

```
56         }
57         catch (PDOException $e) {
58             echo $e->getMessage();
59         }
60     }
61     return $answer;
62 }
63
64 /**
65  * Function which check if a user exists in the database
66  * @param $email
67  * @return bool
68  */
69 function checkUserExists($email)
70 {
71     static $ps = null;
72     $sql = "SELECT * FROM USER WHERE email = :EMAIL;";
73     if ($ps == null)
74     {
75         $ps = dbConnect()->prepare($sql);
76     }
77     $answer = false;
78     try {
79         $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
80         if ($ps->execute())
81         {
82             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
83         }
84     }
85     catch (PDOException $e) {
86         echo $e->getMessage();
87     }
88     return $answer;
89 }
90
91 /**
92  * Function which if a user
93  * @param $email
94  * @return bool|array
95  */
96 function getUserInfo($email)
97 {
98     static $ps = null;
99     $sql = 'SELECT * FROM USER WHERE email = :EMAIL;';
100
101     if ($ps == null)
102     {
103         $ps = dbConnect()->prepare($sql);
104     }
105     $answer = false;
106     try {
107         $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
108         if ($ps->execute())
109         {
110             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
111         }
112     }
113     catch (PDOException $e) {
114         echo $e->getMessage();
115     }
116 }
```

```
116         return $answer[0];
117     }
118
119     /**
120      * Function which return the products in the database
121      * @return bool|array
122      */
123     function getAllProducts()
124     {
125         static $ps = null;
126         $sql = 'SELECT * FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct =
127 PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
128 PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 ORDER BY productName ASC;';
129
130         if ($ps == null)
131         {
132             $ps = dbConnect()->prepare($sql);
133         }
134         $answer = false;
135         try {
136             if ($ps->execute())
137             {
138                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
139             }
140         } catch (PDOException $e) {
141             echo $e->getMessage();
142         }
143         return $answer;
144     }
145
146     /**
147      * Function which return the product whit the id given in parameter
148      * @param $idProduct
149      * @return bool|array
150      */
151     function getProduct($idProduct)
152     {
153         static $ps = null;
154         $sql = 'SELECT * FROM PRODUCT WHERE idProduct = :ID_PRODUCT;';
155
156         if ($ps == null)
157         {
158             $ps = dbConnect()->prepare($sql);
159         }
160         $answer = false;
161         try {
162             $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
163             if ($ps->execute())
164             {
165                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
166             }
167         } catch (PDOException $e) {
168             echo $e->getMessage();
169         }
170         return $answer[0];
171     }
172
173     /**
```

```
174     * Function which return the pictures of a product
175     * @param $idProduct
176     * @return bool|array
177     * @throws PDOException
178     *
179     */
180     function getPictures($idProduct)
181     {
182         static $ps = null;
183         $sql = 'SELECT * FROM PICTURE JOIN PICTURE_PRODUCT ON
184 PICTURE_PRODUCT.idPicture = PICTURE.idPicture WHERE idProduct = :ID_PRODUCT ORDER
185 BY isDefaultPicture DESC';
186
187         if ($ps == null)
188         {
189             $ps = dbConnect()->prepare($sql);
190         }
191         $answer = false;
192         try {
193             $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
194             if ($ps->execute())
195             {
196                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
197             }
198         } catch (PDOException $e) {
199             echo $e->getMessage();
200         }
201         return $answer;
202     }
203
204     /**
205     * Function which return the products which have a name or a description which
206     is like the term given in parameter
207     * @param $termToSearch
208     * @return bool|array
209     */
210     function searchProducts($termToSearch)
211     {
212         static $ps = null;
213         $sql = 'SELECT * FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct =
214 PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
215 PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 AND (productName LIKE
216 :TERM_TO_SEARCH OR description LIKE :TERM_TO_SEARCH) ORDER BY PRODUCT.productName
217 ASC';
218
219         $termToSearch = "%" . $termToSearch . "%";
220         if ($ps == null)
221         {
222             $ps = dbConnect()->prepare($sql);
223         }
224         $answer = false;
225         try {
226             $ps->bindParam(':TERM_TO_SEARCH', $termToSearch, PDO::PARAM_STR);
227             if ($ps->execute())
228             {
229                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
230             }
231         } catch (PDOException $e) {
```

```
227         echo $e->getMessage();
228     }
229     return $answer;
230 }
231
232 /**
233  * Function which add a product in the $_SESSION['shoppingBasket']
234  * @param $idProduct
235  * @param $quantity
236  * @return bool
237  *
238  */
239 function addProductToShoppingBasketSession($idProduct, $quantity)
240 {
241     session_start();
242     if (isset($_SESSION['shoppingBasket'][$idProduct]))
243     {
244         $_SESSION['shoppingBasket'][$idProduct] += $quantity;
245     }
246     else {
247         $_SESSION['shoppingBasket'][$idProduct] = $quantity;
248     }
249 }
250
251 /**
252  * Function which add a product to the shopping basket
253  * @param $email
254  * @param $idProduct
255  * @param $quantity
256  * @return bool
257  *
258  */
259 function addProductToShoppingBasket($email, $idProduct, $quantityToChange)
260 {
261     static $ps = null;
262     $answer = false;
263     if (orderExist($email))
264     {
265         updateRemainingNumber($quantityToChange, $idProduct, true);
266         $totalPrice = getTotalPrice(getIdOrder($email));
267         $totalPrice += $quantityToChange * getProductPrice($idProduct);
268         updateTotalPrice($totalPrice, $_SESSION["email"]);
269         if (productOrderedExists($email, $idProduct))
270         {
271             $sql = "UPDATE PRODUCT_ORDERED SET quantity = (SELECT quantity FROM
PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER) + :QUANTITY
WHERE idProduct = :ID_PRODUCT AND idOrder = :ID_ORDER;";
272             if ($ps == null)
273             {
274                 $ps = dbConnect()->prepare($sql);
275             }
276             try {
277                 $idOrder = getIdOrder($email);
278                 $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
279                 $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
280                 $ps->bindParam(':QUANTITY', $quantityToChange, PDO::PARAM_INT);
281                 $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
282                 $answer = $ps->execute();
283             }
284             catch (PDOException $e) {
```

```

285         echo $e->getMessage();
286     }
287 }
288 else {
289     $sql = "INSERT INTO PRODUCT_ORDERED (`idProduct`, `idOrder`,
`quantity`) VALUES (:ID_PRODUCT, (SELECT idOrder FROM ORDERED WHERE email = :EMAIL
AND isPaid = 0), :QUANTITY);";
290     if ($ps == null)
291     {
292         $ps = dbConnect()->prepare($sql);
293     }
294     try {
295         $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
296         $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
297         $ps->bindParam(':QUANTITY', $quantityToChange, PDO::PARAM_INT);
298         $answer = $ps->execute();
299     }
300     catch (PDOException $e) {
301         echo $e->getMessage();
302     }
303 }
304 }
305 else {
306     newOrder($email, 0);
307     $answer = addProductToShoppingBasket($email, $idProduct,
$quantityToChange);
308 }
309 return $answer;
310 }
311
312 /**
313  * Function which return the price in CHF of a product
314  * @param $idProduct
315  * @return bool|array
316  *
317  */
318 function getProductPrice($idProduct)
319 {
320     static $ps = null;
321     $sql = 'SELECT priceInCHF FROM PRODUCT WHERE idProduct = :ID_PRODUCT;';
322     if ($ps == null)
323     {
324         $ps = dbConnect()->prepare($sql);
325     }
326     $answer = false;
327     try {
328         $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
329         if ($ps->execute())
330         {
331             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
332         }
333     }
334     catch (PDOException $e) {
335         echo $e->getMessage();
336     }
337     return $answer[0]["priceInCHF"];
338 }
339
340 /**
341  * Function which return the total price of the order

```

```
342     * @param $idOrder
343     * @return bool|array
344     *
345     */
346     function getTotalPrice($idOrder)
347     {
348         static $ps = null;
349         $sql = 'SELECT totalPrice FROM ORDERED WHERE idOrder = :ID_ORDER AND isPaid
= 0;';
350         if ($ps == null)
351         {
352             $ps = dbConnect()->prepare($sql);
353         }
354         $answer = false;
355         try {
356             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
357             if ($ps->execute())
358             {
359                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
360             }
361         }
362         catch (PDOException $e) {
363             echo $e->getMessage();
364         }
365         return $answer[0]["totalPrice"];
366     }
367
368     /**
369     * Function which add a product for a user which is not connected
370     * @param $productId
371     * @param $quantityToAdd
372     *
373     */
374     function addProductToShoppingBasketInSession($idProduct, $quantityToAdd)
375     {
376         session_start();
377         updateRemainingNumber($quantityToAdd, $idProduct, true);
378         $_SESSION['shoppingBasket'][$idProduct] += $quantityToAdd;
379     }
380
381     /**
382     * Function which return the id of the "active" order of a user
383     * @param $email
384     * @return bool|int
385     *
386     */
387     function getIdOrder($email)
388     {
389         static $ps = null;
390         $sql = 'SELECT idOrder FROM ORDERED WHERE email = :EMAIL AND isPaid = 0;';
391         if ($ps == null)
392         {
393             $ps = dbConnect()->prepare($sql);
394         }
395         $answer = false;
396         try {
397             $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
398             if ($ps->execute())
399             {
400                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
```

```
401     }
402 }
403 catch (PDOException $e) {
404     echo $e->getMessage();
405 }
406 return $answer[0]["idOrder"];
407 }
408
409 /**
410  * Function which return the products which are in the shopping basket
411  * @param $idProduct
412  * @param $idOrder
413  * @return bool|array
414  *
415  */
416 function getUserQuantityForProduct($idProduct, $idOrder)
417 {
418     static $ps = null;
419     $sql = 'SELECT quantity FROM PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT
AND idOrder = :ID_ORDER;';
420     if ($ps == null)
421     {
422         $ps = dbConnect()->prepare($sql);
423     }
424     $answer = false;
425     try {
426         $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
427         $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
428         if ($ps->execute())
429         {
430             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
431         }
432     }
433     catch (PDOException $e) {
434         echo $e->getMessage();
435     }
436     return $answer[0]['remainingNumber'];
437 }
438
439 /**
440  * Function which calculate the remaining number of a product and change it
441  * @param $quantity
442  * @param $idProduct
443  * @param $hasReduce
444  *
445  */
446 function updateRemainingNumber($quantity, $idProduct, $hasReduce)
447 {
448     $remainingNumber = getQuantity($idProduct);
449     if ($hasReduce)
450     {
451         $remainingNumber -= $quantity;
452     }
453     else {
454         $remainingNumber += $quantity;
455     }
456     updateQuantity($idProduct, $remainingNumber);
457 }
458
459 /**
```



```
460      * Function which update the quantity of a product
461      * @param $idProduct
462      * @param $quantity
463      * @return bool
464      *
465      */
466      function updateQuantity($idProduct, $quantity)
467      {
468          static $ps = null;
469          $sql = "UPDATE PRODUCT SET remainingNumber = :QUANTITY WHERE idProduct =
:ID_PRODUCT;";
470          if ($ps == null)
471          {
472              $ps = dbConnect()->prepare($sql);
473          }
474          $answer = false;
475          try {
476              $ps->bindParam(':QUANTITY', $quantity, PDO::PARAM_INT);
477              $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
478              $answer = $ps->execute();
479          }
480          catch (PDOException $e) {
481              echo $e->getMessage();
482          }
483          return $answer;
484      }
485
486      /**
487      * Function which return the quantity of a product
488      * @param $idProduct
489      * @return int
490      */
491      function getQuantity($idProduct)
492      {
493          static $ps = null;
494          $sql = 'SELECT remainingNumber FROM PRODUCT WHERE idProduct =
:ID_PRODUCT;';
495          if ($ps == null)
496          {
497              $ps = dbConnect()->prepare($sql);
498          }
499          $answer = false;
500          try {
501              $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
502              if ($ps->execute())
503              {
504                  $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
505              }
506          }
507          catch (PDOException $e) {
508              echo $e->getMessage();
509          }
510          return $answer[0]['remainingNumber'];
511      }
512
513      /**
514      * Function which check if the user has already ordered a product
515      * @param string $email
516      * @param int $idProduct
517      * @return bool|array
```

```
518      *
519      */
520      function productOrderedExists($email, $idProduct)
521      {
522          static $ps = null;
523          $sql = 'SELECT * FROM PRODUCT_ORDERED JOIN ORDERED ON
PRODUCT_ORDERED.idOrder = ORDERED.idOrder WHERE isPaid = 0 AND EMAIL = :EMAIL AND
idProduct = :ID_PRODUCT;';
524          if ($ps == null)
525          {
526              $ps = dbConnect()->prepare($sql);
527          }
528          $answer = false;
529          try {
530              $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
531              $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_STR);
532              if ($ps->execute())
533              {
534                  $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
535              }
536          }
537          catch (PDOException $e) {
538              echo $e->getMessage();
539          }
540          return $answer;
541      }
542
543      /**
544       * Function which check if an order is already "active" for a user
545       * @param string $email
546       * @return bool|array
547       *
548       */
549      function orderExist($email)
550      {
551          static $ps = null;
552          $sql = 'SELECT * FROM ORDERED WHERE isPaid = 0 AND email = :EMAIL;';
553          if ($ps == null)
554          {
555              $ps = dbConnect()->prepare($sql);
556          }
557          $answer = false;
558          try {
559              $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
560              if ($ps->execute())
561              {
562                  $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
563              }
564          }
565          catch (PDOException $e) {
566              echo $e->getMessage();
567          }
568          return $answer;
569      }
570
571      /**
572       * Function which create an "active" order for the user in the database
573       * @param $email
574       * @param $totalPrice
575       * @return bool
```

```

576     *
577     */
578     function newOrder($email, $totalPrice)
579     {
580         static $ps = null;
581         $sql = "INSERT INTO ORDERED (`isPaid`, `isSent`, `totalPrice`, `email`)
VALUES (0, 0, :TOTAL_PRICE, :EMAIL);";
582         if ($ps == null)
583         {
584             $ps = dbConnect()->prepare($sql);
585         }
586         $answer = false;
587         try {
588             $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
589             $ps->bindParam(':TOTAL_PRICE', $totalPrice, PDO::PARAM_STR);
590             $answer = $ps->execute();
591         }
592         catch (PDOException $e) {
593             echo $e->getMessage();
594         }
595         return $answer;
596     }
597
598     /**
599     * Function which return the detailed description of an order
600     * @param $email
601     * @return bool|array
602     *
603     */
604     function getShoppingBasket($email)
605     {
606         static $ps = null;
607         $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
fileName, quantity FROM PRODUCT_ORDERED JOIN ORDERED ON PRODUCT_ORDERED.idOrder =
ORDERED.idOrder JOIN PRODUCT ON PRODUCT.idProduct = PRODUCT_ORDERED.idProduct JOIN
PICTURE_PRODUCT ON PICTURE_PRODUCT.idProduct = PRODUCT.idProduct JOIN PICTURE ON
PICTURE_PRODUCT.idPicture = PICTURE.idPicture WHERE isPaid = 0 AND email = :EMAIL
AND isDefaultPicture = 1;';
608         if ($ps == null)
609         {
610             $ps = dbConnect()->prepare($sql);
611         }
612         $answer = false;
613         try {
614             $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
615             if ($ps->execute())
616             {
617                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
618             }
619         }
620         catch (PDOException $e) {
621             echo $e->getMessage();
622         }
623         return $answer;
624     }
625
626     /**
627     * Function which update the total price of an order
628     * @param $newPrice
629     * @param $email

```

```
630     * @return bool|array
631     *
632     */
633     function updateTotalPrice($newPrice, $email)
634     {
635         $idOrder = orderExist($email)[0]['idOrder'];
636         static $ps = null;
637         $sql = "UPDATE ORDERED SET totalPrice = :NEW_PRICE WHERE idOrder =
:ID_ORDER;";
638         if ($ps == null)
639         {
640             $ps = dbConnect()->prepare($sql);
641         }
642         $answer = false;
643         try {
644             $ps->bindParam(':NEW_PRICE', $newPrice, PDO::PARAM_INT);
645             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
646             $answer = $ps->execute();
647         }
648         catch (PDOException $e) {
649             echo $e->getMessage();
650         }
651         return $answer;
652     }
653
654     /**
655     * Function which update the profil of an user
656     * @param $oldEmail
657     * @param $newEmail
658     * @param $passwordHash
659     * @param $streetName
660     * @param $streetNumber
661     * @param $postalCode
662     * @param $city
663     *
664     */
665     function updateProfil($oldEmail, $newEmail, $passwordHash, $streetName,
666     $streetNumber, $postalCode, $city)
667     {
668         $answer = false;
669         if (!checkUserExists($newEmail) || $oldEmail == $newEmail)
670         {
671             static $ps = null;
672             $sql = "UPDATE USER SET email = :NEW_EMAIL, passwordHash =
:PASSWORD_HASH, streetName = :STREET_NAME, streetNumber = :STREET_NUMBER, city =
:CITY, postalCode = :POSTAL_CODE WHERE email = :OLD_EMAIL;";
673             if ($ps == null)
674             {
675                 $ps = dbConnect()->prepare($sql);
676             }
677             try {
678                 $ps->bindParam(':NEW_EMAIL', $newEmail, PDO::PARAM_STR);
679                 $ps->bindParam(':PASSWORD_HASH', $passwordHash, PDO::PARAM_STR);
680                 $ps->bindParam(':STREET_NAME', $streetName, PDO::PARAM_STR);
681                 $ps->bindParam(':STREET_NUMBER', $streetNumber, PDO::PARAM_STR);
682                 $ps->bindParam(':POSTAL_CODE', $postalCode, PDO::PARAM_STR);
683                 $ps->bindParam(':CITY', $city, PDO::PARAM_STR);
684                 $ps->bindParam(':OLD_EMAIL', $oldEmail, PDO::PARAM_STR);
685
686                 $answer = $ps->execute();
687             }
688             catch (PDOException $e) {
689                 echo $e->getMessage();
690             }
691         }
692         return $answer;
693     }
694 }
```

```
686     }
687     catch (PDOException $e) {
688         echo $e->getMessage();
689     }
690 }
691 return $answer;
692 }
693
694 /**
695  * Function which return all the orders of a user
696  * @param $email
697  * @return bool|array
698  *
699  */
700 function getOrders($email)
701 {
702     static $ps = null;
703     $sql = 'SELECT * FROM ORDERED WHERE email = :EMAIL;';
704     if ($ps == null)
705     {
706         $ps = dbConnect()->prepare($sql);
707     }
708     $answer = false;
709     try {
710         $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
711         if ($ps->execute())
712         {
713             $orders = $ps->fetchAll(PDO::FETCH_ASSOC);
714         }
715     }
716     catch (PDOException $e) {
717         echo $e->getMessage();
718     }
719     foreach ($orders as $order)
720     {
721         $answer[$order['idOrder']]['orderInfo'] = getOrder($order['idOrder'])
722 [0];
723         $answer[$order['idOrder']]['detailOrder'] =
724 getProductsOfAnOrder($order['idOrder']);
725     }
726     return $answer;
727 }
728
729 /**
730  * Function which return all the users in the database
731  * @return bool|array
732  *
733  */
734 function getClients()
735 {
736     static $ps = null;
737     $sql = 'SELECT * FROM USER;';
738     if ($ps == null)
739     {
740         $ps = dbConnect()->prepare($sql);
741     }
742     $answer = false;
743     try {
744         $ps->bindParam(':EMAIL', $email, PDO::PARAM_STR);
745         if ($ps->execute())
```

```
744         {
745             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
746         }
747     }
748     catch (PDOException $e) {
749         echo $e->getMessage();
750     }
751     return $answer;
752 }
753
754 /**
755  * Function which return the order of a user
756  * @param $idOrder
757  * @return bool|array
758  *
759  */
760 function getOrder($idOrder)
761 {
762     static $ps = null;
763     $sql = 'SELECT * FROM ORDERED WHERE idOrder = :ID_ORDER;';
764     if ($ps == null)
765     {
766         $ps = dbConnect()->prepare($sql);
767     }
768     $answer = false;
769     try {
770         $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
771         if ($ps->execute())
772         {
773             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
774         }
775     }
776     catch (PDOException $e) {
777         echo $e->getMessage();
778     }
779     return $answer;
780 }
781
782 /**
783  * Function which return the products of an order
784  * @param $idOrder
785  * @return bool|array
786  *
787  */
788 function getProductsOfAnOrder($idOrder)
789 {
790     $answer = false;
791     static $ps = null;
792     $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
remainingNumber, fileName, quantity FROM PRODUCT JOIN PICTURE_PRODUCT ON
PRODUCT.idProduct = PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
PICTURE_PRODUCT.idPicture JOIN PRODUCT_ORDERED ON PRODUCT_ORDERED.idProduct =
PRODUCT.idProduct WHERE isDefaultPicture = 1 AND idOrder = :ID_ORDER;';
793
794     if ($ps == null)
795     {
796         $ps = dbConnect()->prepare($sql);
797     }
798
799     try {
```

```
800     $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
801     if ($ps->execute())
802     {
803         $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
804     }
805 }
806 catch (PDOException $e) {
807     echo $e->getMessage();
808 }
809 return $answer;
810 }
811
812 /**
813  * Function which rename the pictures and give them a unique name
814  * @param $pictures
815  * @return array
816  *
817  */
818 function renamePictures($pictures)
819 {
820     for ($i=0; $i < count($pictures["name"]); $i++)
821     {
822         $pictures["name"][$i] = "picture_" . date('Y-m-d-H-i-s') . "_" .
uniqid() . "." . substr($pictures["type"][$i], 6);
823     }
824     return $pictures;
825 }
826
827 /**
828  * Function which add a picture to the database
829  * @param $fileName
830  * @return bool
831  *
832  */
833 function addPicture($fileName)
834 {
835     static $ps = null;
836     $sql = "INSERT INTO `PICTURE` (`fileName`) VALUES (:FILENAME);";
837
838     if ($ps == null)
839     {
840         $ps = dbConnect()->prepare($sql);
841     }
842     $answer = false;
843     try {
844         $ps->bindParam(':FILENAME', $fileName, PDO::PARAM_STR);
845         $answer = $ps->execute();
846     } catch (PDOException $e) {
847         echo $e->getMessage();
848     }
849     return $answer;
850 }
851
852 /**
853  * Function which link a picture to a product
854  * @param $idProduct
855  * @param $idPicture
856  * @param $isDefaultPicture
857  * @return bool
858  *
```

```

859     */
860     function linkPictureToProduct($idProduct, $idPicture, $isDefaultPicture)
861     {
862         static $ps = null;
863         $sql = "INSERT INTO `PICTURE_PRODUCT` (`idProduct`, `idPicture`,
`isDefaultPicture`) VALUES (:ID_PRODUCT, :ID_PICTURE, :IS_DEFAULT_PICTURE);";
864
865         if ($ps == null)
866         {
867             $ps = dbConnect()->prepare($sql);
868         }
869         $answer = false;
870         try {
871             $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
872             $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
873             $ps->bindParam(':IS_DEFAULT_PICTURE', $isDefaultPicture,
PDO::PARAM_BOOL);
874
875             $answer = $ps->execute();
876         } catch (PDOException $e) {
877             echo $e->getMessage();
878         }
879         return $answer;
880     }
881
882     /**
883     * Function which add a product to the database
884     * @param $productName
885     * @param $description
886     * @param $priceInCHF
887     * @param $remainingNumber
888     * @return bool
889     */
890     */
891     function addProduct($productName, $description, $priceInCHF, $remainingNumber)
892     {
893         static $ps = null;
894         $sql = "INSERT INTO `PRODUCT` (`productName`, `description`, `priceInCHF`,
`remainingNumber`) VALUES (:PRODUCT_NAME, :DESCRIPTION, :PRICE_IN_CHF,
:REMAINING_NUMBER);";
895         /*if ($ps == null)
896         {
897             $pdo = dbConnect();
898             $pdo->beginTransaction();
899             $ps = $pdo->prepare($sql);
900         }*/
901         if ($ps == null)
902         {
903             $ps = dbConnect()->prepare($sql);
904         }
905         $answer = false;
906         try {
907             $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);
908             $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
909             $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
910             $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);
911
912             $answer = $ps->execute();
913
914         } catch (PDOException $e) {

```



```
915         //$pdo->rollBack();
916         echo $e->getMessage();
917     }
918     return $answer;
919 }
920
921 /**
922  * Function which add a product and its pictures to the database and in the
server
923  * @param $productName
924  * @param $description
925  * @param $priceInCHF
926  * @param $remainingNumber
927  * @param $hasOthersPictures
928  * @param $defaultPicture
929  * @param $othersPictures
930  *
931  */
932 function addProductWithPictures($productName, $description, $priceInCHF,
    $remainingNumber, $hasOthersPictures, $defaultPicture, $othersPictures,
    $addProduct, $idProduct = 0)
933 {
934     try {
935         if ($addProduct)
936         {
937             addProduct($productName, $description, $priceInCHF,
    $remainingNumber);
938             $idProduct = getIdProduct($productName, $description, $priceInCHF,
    $remainingNumber);
939         }
940         if (isset($defaultPicture["name"][0]))
941         {
942             addPicture($defaultPicture["name"][0]);
943             $idPicture = getIdPicture($defaultPicture["name"][0]);
944             linkPictureToProduct($idProduct, $idPicture, true);
945         }
946         if ($hasOthersPictures)
947         {
948             for ($i=0; $i < count($othersPictures["name"]); $i++)
949             {
950                 addPicture($othersPictures["name"][$i]);
951                 $idPicture = getIdPicture($othersPictures["name"][$i]);
952                 linkPictureToProduct($idProduct, $idPicture, false);
953             }
954         }
955         savePictures($defaultPicture);
956         savePictures($othersPictures);
957     }
958     catch (Exception $e) {
959         return false;
960     }
961     return true;
962 }
963
964 /**
965  * Function which return the id of a picture
966  * @param $fileName
967  * @return int
968  *
969  */
```

```
970     function getIdPicture($fileName)
971     {
972         static $ps = null;
973         $sql = 'SELECT idPicture FROM PICTURE WHERE fileName = :FILENAME;';
974
975         if ($ps == null)
976         {
977             $ps = dbConnect()->prepare($sql);
978         }
979         $answer = false;
980         try {
981             $ps->bindParam(':FILENAME', $fileName, PDO::PARAM_STR);
982             if ($ps->execute())
983             {
984                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
985             }
986         }
987         catch (PDOException $e) {
988             echo $e->getMessage();
989         }
990         return $answer[0]["idPicture"];
991     }
992
993     /**
994     * Function which return the id of a product
995     * @param $productName
996     * @param $description
997     * @param $priceInCHF
998     * @param $remainingNumber
999     * @return int
1000     */
1001     */
1002     function getIdProduct($productName, $description, $priceInCHF,
1003     $remainingNumber)
1004     {
1005         static $ps = null;
1006         $sql = 'SELECT idProduct FROM PRODUCT WHERE productName = :PRODUCT_NAME AND
description = :DESCRIPTION AND priceInCHF = :PRICE_IN_CHF AND remainingNumber =
:REMAINING_NUMBER;';
1007         if ($ps == null)
1008         {
1009             $ps = dbConnect()->prepare($sql);
1010         }
1011         $answer = false;
1012         try {
1013             $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);
1014             $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
1015             $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
1016             $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);
1017             if ($ps->execute())
1018             {
1019                 $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
1020             }
1021         }
1022         catch (PDOException $e) {
1023             echo $e->getMessage();
1024         }
1025         return $answer[0]["idProduct"];
1026     }
1027 }
```

```
1027  /**
1028  * Function which save the pictures on the server
1029  * @param $pictures
1030  * @return bool|array
1031  *
1032  */
1033  function savePictures($pictures)
1034  {
1035      $picturesNb = count($pictures["name"]);
1036      $picturesMoved = 0;
1037      for ($i=0; $i < count($pictures["name"]); $i++)
1038      {
1039          $pictureMoved = false;
1040          $pictureMoved = move_uploaded_file($pictures['tmp_name'][$i],
PICTURES_FOLDER . $pictures["name"][$i]);
1041          if ($pictureMoved)
1042          {
1043              $picturesMoved++;
1044          }
1045      }
1046      if ($picturesMoved != $picturesNb)
1047      {
1048          return false;
1049      }
1050      else {
1051          return $pictures;
1052      }
1053  }
1054
1055  /**
1056  * Function which delete a product and its associated pictures from the
database and the server
1057  * @param $idProduct
1058  * @param $deleteProduct
1059  *
1060  */
1061  function deleteProductAndPictures($idProduct, $deleteProduct)
1062  {
1063      $pictures = getPictures($idProduct);
1064      for ($i = 0; $i < count($pictures); $i++)
1065      {
1066          deleteLinkPictureToProduct($pictures[$i]["idPicture"]);
1067          unlink(PICTURES_FOLDER . $pictures[$i]["fileName"]);
1068          deletePicture($pictures[$i]["idPicture"]);
1069      }
1070      if ($deleteProduct)
1071      {
1072          deleteProduct($idProduct);
1073      }
1074  }
1075
1076  /**
1077  * Function which delete a product from the database
1078  * @param $idProduct
1079  * @return bool|array
1080  *
1081  */
1082  function deleteProduct($idProduct)
1083  {
1084      static $ps = null;
```

```
1085     $sql = "DELETE FROM PRODUCT WHERE idProduct = :ID_PRODUCT;";
1086     if ($ps == null)
1087     {
1088         $ps = dbConnect()->prepare($sql);
1089     }
1090     $answer = false;
1091     try {
1092         $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
1093         $answer = $ps->execute();
1094     }
1095     catch (PDOException $e) {
1096         echo $e->getMessage();
1097     }
1098     return $answer;
1099 }
1100
1101 /**
1102  * Function which delete the link in the database between the product and the
picture
1103  * @param $idPicture
1104  * @return bool|array
1105  */
1106 function deleteLinkPictureToProduct($idPicture)
1107 {
1108     static $ps = null;
1109     $sql = "DELETE FROM PICTURE_PRODUCT WHERE idPicture = :ID_PICTURE;";
1110     if ($ps == null)
1111     {
1112         $ps = dbConnect()->prepare($sql);
1113     }
1114     $answer = false;
1115     try {
1116         $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
1117         $answer = $ps->execute();
1118     }
1119     catch (PDOException $e) {
1120         echo $e->getMessage();
1121     }
1122     return $answer;
1123 }
1124
1125 /**
1126  * Function which delete a picture from the database
1127  * @param $idPicture
1128  * @return bool|array
1129  *
1130  */
1131 function deletePicture($idPicture)
1132 {
1133     static $ps = null;
1134     $sql = "DELETE FROM PICTURE WHERE idPicture = :ID_PICTURE;";
1135     if ($ps == null)
1136     {
1137         $ps = dbConnect()->prepare($sql);
1138     }
1139     $answer = false;
1140     try {
1141         $ps->bindParam(':ID_PICTURE', $idPicture, PDO::PARAM_INT);
1142         $answer = $ps->execute();
1143     }
```

```

1144         catch (PDOException $e) {
1145             echo $e->getMessage();
1146         }
1147         return $answer;
1148     }
1149
1150     /**
1151      * Function which update the product and its pictures using the fonction to add
a product
1152      * @param $idProduct
1153      * @param $productName
1154      * @param $description
1155      * @param $priceInCHF
1156      * @param $remainingNumber
1157      * @param $hasOthersPictures
1158      * @param $defaultPicture
1159      * @param $othersPictures
1160      *
1161      */
1162     function updateProductPictures($idProduct, $productName, $description,
    $priceInCHF, $remainingNumber, $hasOthersPictures, $defaultPicture,
    $othersPictures)
1163     {
1164         try {
1165             updateProductInfos($idProduct, $productName, $description, $priceInCHF,
    $remainingNumber);
1166             addProductWithPictures($productName, $description, $priceInCHF,
    $remainingNumber, $hasOthersPictures, $defaultPicture, $othersPictures, false,
    $idProduct);
1167         }
1168         catch (Exception $e) {
1169             return false;
1170         }
1171         return true;
1172     }
1173
1174     /**
1175      * Function which update the informations of a product
1176      * @param $idProduct
1177      * @param $productName
1178      * @param $description
1179      * @param $priceInCHF
1180      * @param $remainingNumber
1181      * @return bool|array
1182      *
1183      */
1184     function updateProductInfos($idProduct, $productName, $description,
    $priceInCHF, $remainingNumber)
1185     {
1186         $answer = false;
1187         static $ps = null;
1188         $sql = 'UPDATE PRODUCT SET productName = :PRODUCT_NAME, description =
    :DESCRIPTION, priceInCHF = :PRICE_IN_CHF, remainingNumber = :REMAINING_NUMBER WHERE
    idProduct = :ID_PRODUCT;';
1189         if ($ps == null)
1190         {
1191             $ps = dbConnect()->prepare($sql);
1192         }
1193         try {
1194             $ps->bindParam(':PRODUCT_NAME', $productName, PDO::PARAM_STR);

```

```
1195     $ps->bindParam(':DESCRIPTION', $description, PDO::PARAM_STR);
1196     $ps->bindParam(':PRICE_IN_CHF', $priceInCHF, PDO::PARAM_STR);
1197     $ps->bindParam(':REMAINING_NUMBER', $remainingNumber, PDO::PARAM_INT);
1198     $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
1199     $answer = $ps->execute();
1200 }
1201 catch (PDOException $e) {
1202     echo $e->getMessage();
1203 }
1204 return $answer;
1205 }
1206
1207 /**
1208  * Function which delete a picture
1209  * @param $idPicture
1210  *
1211  */
1212 function deleteSelectedPicture($idPicture)
1213 {
1214     deleteLinkPictureToProduct($idPicture);
1215     deletePicture($idPicture);
1216 }
1217
1218
1219 /**
1220  * Function which return product's information
1221  * @param $idProduct
1222  * @return bool|array
1223  *
1224  */
1225 function getProductDetails($idProduct)
1226 {
1227     $answer = false;
1228     static $ps = null;
1229     $sql = 'SELECT PRODUCT.idProduct, productName, description, priceInCHF,
remainingNumber, fileName FROM PRODUCT JOIN PICTURE_PRODUCT ON PRODUCT.idProduct =
PICTURE_PRODUCT.idProduct JOIN PICTURE ON PICTURE.idPicture =
PICTURE_PRODUCT.idPicture WHERE isDefaultPicture = 1 AND PRODUCT.idProduct =
:ID_PRODUCT;';
1230
1231     if ($ps == null)
1232     {
1233         $ps = dbConnect()->prepare($sql);
1234     }
1235
1236     try {
1237         $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
1238         if ($ps->execute())
1239         {
1240             $answer = $ps->fetchAll(PDO::FETCH_ASSOC);
1241         }
1242     }
1243     catch (PDOException $e) {
1244         echo $e->getMessage();
1245     }
1246     return $answer[0];
1247 }
1248
1249 /**
1250  * Function which change the state of the payment
```

```
1251     * @param $idOrder
1252     * @param $isPaid
1253     * @return bool
1254     *
1255     */
1256     function switchPaid($idOrder, $isPaid)
1257     {
1258         $answer = false;
1259         static $ps = null;
1260         $sql = 'UPDATE ORDERED SET isPaid = :IS_PAID WHERE idOrder = :ID_ORDER;';
1261         if ($ps == null)
1262         {
1263             $ps = dbConnect()->prepare($sql);
1264         }
1265         try {
1266             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
1267             $ps->bindParam(':IS_PAID', $isPaid, PDO::PARAM_BOOL);
1268             $answer = $ps->execute();
1269         }
1270         catch (PDOException $e) {
1271             echo $e->getMessage();
1272         }
1273         return $answer;
1274     }
1275
1276     /**
1277     * Function which change the state of the sending
1278     * @param $idOrder
1279     * @param $isPaid
1280     * @return bool
1281     *
1282     */
1283     function switchSent($idOrder, $isSent)
1284     {
1285         $answer = false;
1286         static $ps = null;
1287         $sql = 'UPDATE ORDERED SET isSent = :IS_SENT WHERE idOrder = :ID_ORDER;';
1288         if ($ps == null)
1289         {
1290             $ps = dbConnect()->prepare($sql);
1291         }
1292         try {
1293             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
1294             $ps->bindParam(':IS_SENT', $isSent, PDO::PARAM_BOOL);
1295             $answer = $ps->execute();
1296         }
1297         catch (PDOException $e) {
1298             echo $e->getMessage();
1299         }
1300         return $answer;
1301     }
1302
1303     /**
1304     * Function which delete a product from the shoppingBasket
1305     * @param $idProduct
1306     * @param $quantity
1307     * @param $idOrder
1308     * @return bool
1309     *
1310     */
```

```
1311     function deleteProductFromShoppingBasket($idProduct, $quantity, $idOrder)
1312     {
1313         static $ps = null;
1314         $sql = "DELETE FROM PRODUCT_ORDERED WHERE idProduct = :ID_PRODUCT AND
idOrder = :ID_ORDER;";
1315         if ($ps == null)
1316         {
1317             $ps = dbConnect()->prepare($sql);
1318         }
1319         $answer = false;
1320         try {
1321             $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
1322             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
1323             $answer = $ps->execute();
1324         }
1325         catch (PDOException $e) {
1326             echo $e->getMessage();
1327         }
1328         return $answer;
1329     }
1330
1331
1332     /**
1333     * Function which update a product from the shoppingBasket
1334     * @param $idProduct
1335     * @param $quantity
1336     * @param $idOrder
1337     * @return bool
1338     */
1339
1340     function updateProductInShoppingBasket($idProduct, $quantity, $idOrder)
1341     {
1342         static $ps = null;
1343         $sql = "UPDATE PRODUCT_ORDERED SET quantity = :QUANTITY WHERE idProduct =
:ID_PRODUCT AND idOrder = :ID_ORDER;";
1344         if ($ps == null)
1345         {
1346             $ps = dbConnect()->prepare($sql);
1347         }
1348         $answer = false;
1349         try {
1350             $ps->bindParam(':ID_PRODUCT', $idProduct, PDO::PARAM_INT);
1351             $ps->bindParam(':QUANTITY', $quantity, PDO::PARAM_INT);
1352             $ps->bindParam(':ID_ORDER', $idOrder, PDO::PARAM_INT);
1353             $answer = $ps->execute();
1354         }
1355         catch (PDOException $e) {
1356             echo $e->getMessage();
1357         }
1358         return $answer;
1359     }
1360 ?>
```