5)Webshop in PHP – Login, Logout, Zur Kassa gehen

Der Warenkorb kann ohne Anmeldung befüllt werden. Klickt man jedoch auf den Button „Zur Kassa“ muss man sich anmelden und wird auf die Seite „login.php“ geleitet.

Erstelle eine neue Datei im Ordner „templates“ mit dem Namen „login.php“

Da die Seite auch die gleiche Grundlage hat wie „main.php“ kopiere den Inhalt und füge ihn in „login.php“ ein.

Der Bereich im <container> ist jedoch nicht zu gebrauchen, entferne diesen.

```
17     <section class="container" id="produkte" >
18     <div class="row">
19         <?php while($row = $result->fetch()):?> <!--hier der
           Doppelpunkt-->
20             <div class="col">
21                 <?php include 'card.php'?>
22             </div>
23         <?php endwhile;?>
24     </div>
```

Ändere auch die „id“ darüber in „login“:

```
17     <section class="container" id="loginForm" >
18
19     </section>
```

In den <container“ wird jetzt ein Login-Formular eingefügt.

Dieses soll einerseits die schönen Umrahmungen einer <card> aufweisen und andererseits ein Formular-Schema aus Bootstrap erhalten.

5.1)Formular erstellen

```
17     <section class="container" id="loginForm" >
18     <form action="index.php/login" method="post">
19         <div class="card">
20         <div class="card-header"></div>
21         <div class="card-body"></div>
22         <div class="card-footer"></div>
23         </div>
24     </form>
25     </section>
```

```
<form action="index.php/login" method="post">
    <div class="card">
    <div class="card-header"></div>
    <div class="card-body"></div>
    <div class="card-footer"></div>
    </div>
  </form>
```

In den <card-header> kommt die Überschrift, in den <body> das Formular und in den <footer> die Button.

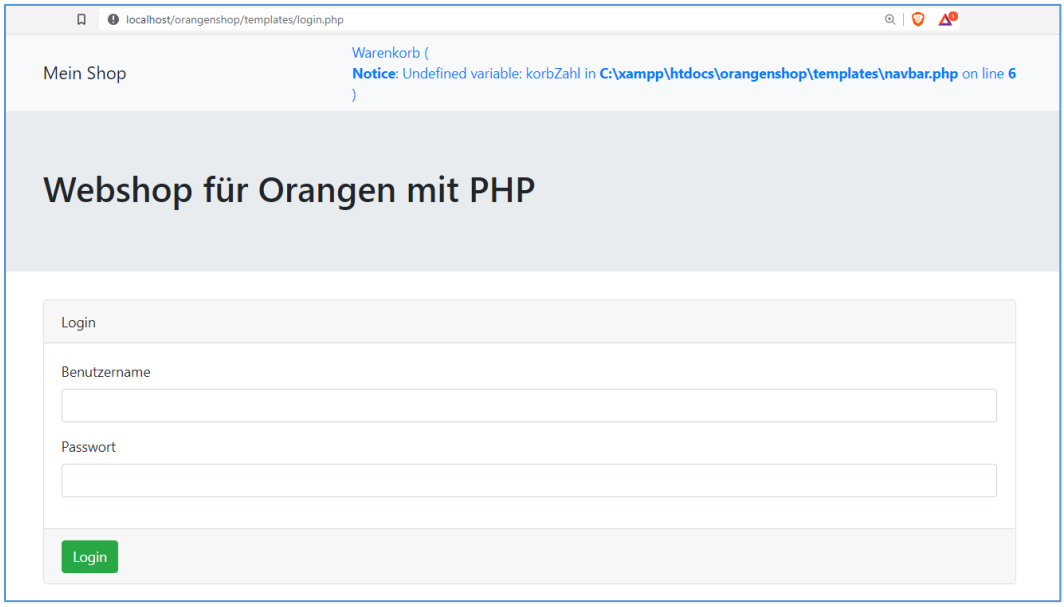
```
19          <div class="card">
20          <div class="card-header">Login</div>
21          <div class="card-body"></div>
22          <div class="card-footer">
23              <button class="btn btn-success" type="submit">Login</button>
24          </div>
25          </div>
```

Danach erfolgt die Erstellung des Formulars im <body>:

```
22          <div class="card-body">
23              <div class="form-group">
24                  <label for="username">Benutzername</label>
25                  <input type="text" name="username" class="form-control">
26              </div>
27              <div class="form-group">
28                  <label for="password">Passwort</label>
29                  <input type="password" name="password" class="form-
                    control">
30              </div>
31          </div>
32          <div class="card-footer">
```

```
<div class="form-group">
    <label for="username">Benutzername</label>
    <input type="text" name="username" class="form-control">
</div>
<div class="form-group">
    <label for="password">Passwort</label>
    <input type="password" name="password" class="form-control">
</div>
```

Ergebnis: direkt Aufruf, da noch keine Route besteht – nur zur Kontrolle



Da keine Route besteht kann die Navbar auch keinen Warenkorb anzeigen, es fehlt ja z.B. die userId usw.

5.2)Route für login erstellen

Bevor man zur Kassa gehen kann, muss man sich einloggen oder registrieren.

- **Neue Funktion „isLoggedIn“ in der „user.php“**
  Damit wird immer überprüft, ob der User eingelogt ist oder nicht.

```
13
14 ▼ function isLoggedIn():bool{
15       return isset($_SESSION['userId']);
16   }
```

function isLoggedIn():bool{
 return isset($_SESSION['userId']);
}

Info:

  - Diese Funktion gibt zurück ein „true“ oder „false“ – mit dem „bool“
  - und ob die Session gesetzt ist, die nach dem Einloggen entsteht.

- **Neue Route „login“ anlegen**
  erstelle in der “routes.php“ ganz unten folgende neue Route, inkl. „exit“.
  Kopiere das von oben und ändere es:

```
33 ▼ if(strpos($route,'/login') !== false) {
34         exit();
35   }
```

Hier muss nun auch ein Template ausgegeben werden. Kopiere die „require“ von oben und ändere es:

```
33 ▼ if(strpos($route,'/login') !== false) {
34       require __DIR__.'/templates/login.php';
35         exit();
36   }
```

Nun wird unterschieden, ob bereits ein Formular abgeschickt wurde und somit bereits eine Methode „POST“ vorhanden ist, oder ob nur auf die Seite geklickt wurde. Dafür wird eine neue Variable erstellt:

- $isPost

Dies wird aus der zentralen Servervariablen ausgelesen ($_SERVER). Da es aber auch vorkommen kann, das diese intern groß oder klein geschrieben wird, sichert man sich hier ab mit der Funktion „strtoupper“. Heißt „string to upper“ und wandelt alles in Großbuchstaben um. Daher füge ein:

```
33 ▼ if(strpos($route,'/login') !== false) {
34       $isPost = strtoupper($_SERVER['REQUEST_METHOD']) ==='POST';
35       require __DIR__.'/templates/login.php';
```

$isPost = strtoupper($_SERVER['REQUEST_METHOD']) ==='POST';

- **Variablen (Values) für username und passwort erstellen**

In dieser Route sollen nun 2 Variablen erstellt werden, nämlich für „username“ und „password“. Diese werdern im Formular übernommen, damit sie als <value> angezeigt werden. Damit sieht der Benutzer, was man im Formular ausfüllt.

```
34      $isPost = strtoupper($_SERVER['REQUEST_METHOD'])
35      $username ="";
36      $password = "";
37      require __DIR__.'/templates/login.php';
```

Entsprechend fügt man die „value“ auch in das Formular ein. Öffne wieder „login.php“ und füge in den beiden Input-Felder dies ein:

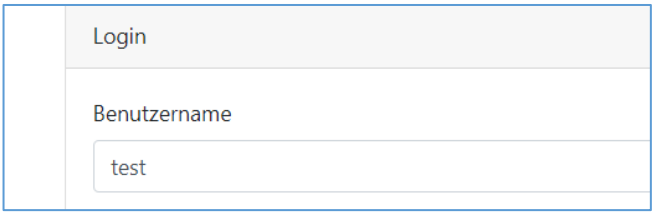
```
24              <label for="username">Benutzername</label>
25              <input type="text" value="<?= $username ?>"
                 name="username" class="form-control">
26          </div>
27          <div class="form-group">
28              <label for="password">Passwort</label>
29              <input type="password" value="<?= $password ?>"
                name="password" class="form-control">
```

value="<?= $username ?>"

Testen:
Trage in „routes.php“ im username „test“ ein, öffne das Formular und aktualisiere dieses.

```
34      $isPost = strtoupper($_SERVER['R
35      $username ="test";
36      $password = "";
```

Ergebnis:



5.3)Link „Zur Kassa gehen“

Füge in der „korbSeite.php“ den Link ein:

```
33          <br>
34          <a href="index.php/checkout" class="btn btn-primary col-12">Zur
            Kasse gehen</a>
35          </div>
```

Dieser Link muss auch in der „routes.php“ bekannt gemacht werden.

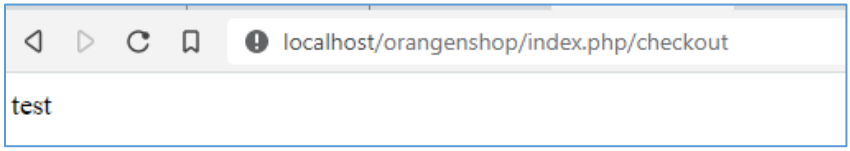
Öffne **routes.php** und erstelle ganz unten einen weiteren IF-ELSE Block, also eine neue Route. Mit einem „exit“.

```
33 ▼ if(strpos($route,'/checkout') !== false) {
34       |
35       exit();
36   }
```

Diese Route sofort kurz testen, indem man ein kleines „echo“ verwendet und die Warenkorb-Seite aktualisiert und den Button „Zur Kassa“ betätigt.

```
33 ▼ if(strpos($route,'/checkout') !== false) {
34       echo "test";
35       exit();
36   }
```

Ergebnis:



Passt. Echo wieder löschen.

Nun kommt hier die IF-Entscheidung: wenn der User NICHT eingeloggt ist, soll was passieren, daher verwende diese Funktion aus der „user.php“ hier in der IF:

```
33 ▼ if(strpos($route,'/checkout') !== false) {
34 ▼     if(!isLoggedIn()){
35           |
36       }
37       exit();
38   }
```

Auf jeden Fall soll auf die „Login-Seite“ weitergeleitet werden.

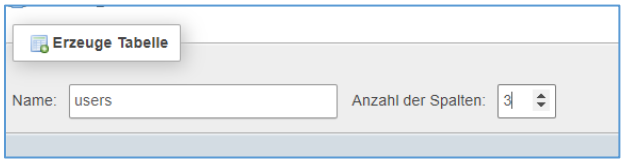
header("Location: index.php/login");

```
41 ▼ if(strpos($route,'/checkout') !== false) {
42       if(!isLoggedIn())
43 ▼     {
44           header("Location: index.php/login");
45       }
46       exit();
47   }
```

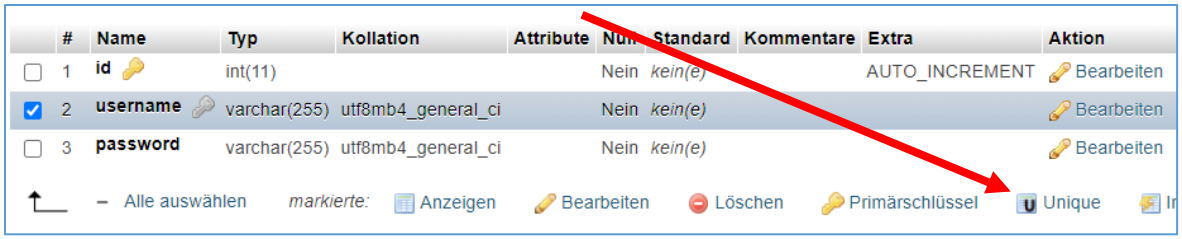
Was „ansonsten“ passiert folgt später.

5.4)Login-Tabelle in Datenbank anlegen

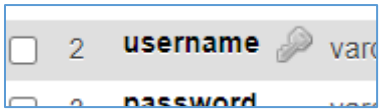
Öffne mit phpMyAdmin die Datenbank und lege eine neue Tabelle an namens „users“.



Nachdem alle 3 Elemente angelegt sind, kann man den „username“ eindeutig machen: Dabei soll der „username“ eindeutig sein und sich nicht wiederholen. Dafür muss man den „username“ vorne am Zeilenkopf auswählen und dann den Button „Unique“ anklicken:

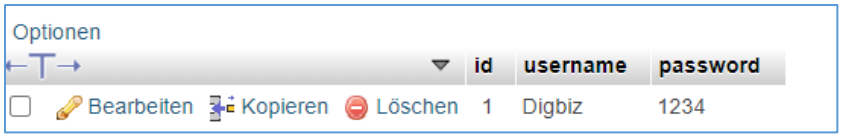


Als Ergebnis wird dann ein grauer Schlüsselsymbol neben dem Namen angezeigt.



- **User eingeben**

Zum Testen gib einen neuen User und Passwort ein mit „einfügen“, z.B. Digbiz und 1234



Hier wird eine spätere Verschlüsselung des Passwortes noch nicht berücksichtigt.
Quelle: https://www.youtube.com/watch?v=cXZG4yJhLsQ 10:10

5.5)User und Passwort abfragen

Da ein User vorhanden ist, kann man nun darauf eingehen.
Öffne wieder die „routes.php“, um bei der Route vom Login weiter zu arbeiten.

Zuerst entferne den „test“ in username wieder.

Nun muss man prüfen, ob eine „POST“-Methode abgesendet wurde: Zeile 38

```
33 ▼ if(strpos($route,'/login') !== false) {
34       $isPost = strtoupper($_SERVER['REQUEST_MET
35       $username ="";
36       $password = "";
37
38       if($isPost){}
39       require __DIR__.'/templates/login.php';
```

if($isPost){ }

- **Testen, ob etwas im Formular eingegeben wurde**

Da drinnen wird nun noch eine IF erstellt , die überprüft, ob in der „username“-Variable etwas drinnen steht. Dieser Wert ist mit „bool“ zu einem Bolean umgewandelt worden, der „true“ oder „false“ ist. Steht nichts drinnen, also ist „false2 dann soll eine Fehlermeldung erfolgen.

```
38      if($isPost){
39          if((bool)$username === false){
40              $errors[]="Benutzername ist leer";
41          }
42      }
```

```
if((bool)$username === false){
        $errors[]="Benutzername ist leer";
    }
```

Das gleiche macht man dann auch für das Passwort:

```
39          if((bool)$username === false){
40              $errors[]="Benutzername ist leer";
41          }
42          if((bool)$password === false){
43              $errors[]="Passwort ist leer";
44          }
45      }
```

```
if((bool)$password === false){
        $errors[]="Passwort ist leer";
    }
```

- **Testen, ob ein Benutzer in der Datenbank ist**

Erstelle gleich darunter eine Variable, die auf eine noch zu erstellende Funktion zurgreift und vergleicht mit IF, ob der username gesetzt ist (mit (bool) und man keine Daten aus der Datenbank bekommt. Dann soll eine Fehlermeldung erfolgen:
Info:
Die Funktion „getUsername“ wird danach noch in der datei „routes.php“ von uns angelegt.

```
43              $errors[]="Passwort ist leer";
44          }
45          $userData = getUsername($username);
46          if((bool)$username && 0 === count($userData)){
47              $errors[]="Benutzername existiert nicht";
48          }
49      }
```

```
$userData = getUsername($username);
    if((bool)$username && 0 === count($userData)){
        $errors[]="Benutzername existiert nicht";
    }
```

- **Testen, ob das Passwort in der Datenbank stimmt**
  - Zuerst wird gefragt, ob das Passwort überhaupt übergeben wurde - (bool)$password.
  - Dann wird überprüft, ob in den Userdaten ($userData), die aus der Funktion „getUsername“ und deren SQL-Abfrage kommt, ein Passwort vorhanden ist.
  - Wenn das also falsch ist, dann wird eine Fehlermeldung ausgegeben.

```
49          //Passwort checken mit Datenbank-Vergleich
50          if((bool)$password && isset($userData['password']) && false ===
            $userData['password']){
51              $errors[]="Passwort stimmt nicht";
52          }
```

```
if((bool)$password && isset($userData['password']) && false === $userData['password']){
        $errors[]="Passwort stimmt nicht";
    }
```

Achtung: hier in „userData“ muss der Name aus der Datenbank gewählt werden, der kann manchmal auch anders lauten als wie hier zufällig ebenfalls „password“, nämlich z.B. „Passwort“. Dann **würde** es so aussehen:

würde:

```
//Passwort checken mit Datenbank-Vergleich
if((bool)$password && isset($userData['PasswortGeheim']) && false ===
$userData['password']){
    $errors[]="Passwort stimmt nicht";
}
```

- **Sinnvolles Ende: erfolgreich eingeloggt sein**

Ergibt nun die Zählung aller Fehler ($errors) eine Null, dann passt ja alles und der Login ist erfolgreich.

Daher kann man sich aus den $userData die „userId“ in die SESSION übergeben und sie somit hineinschreiben in die Session-Variable.

```
54          }
55          //erfolgreich alles geschafft
56          if(0 === count($errors)){
57              $_SESSION['userId'] = (int)$userData['id'];
58          }
59      }
60      require __DIR__.'/templates/login.php';
```

```
if(0 === count($errors)){
        $_SESSION['userId'] = (int)$userData['id'];
    }
```

- **Die <input>-Felder aus dem Formular holen**

Nun muss man noch die Inputfelder, die ein User in das Formular befüllt, in diese Ansammlung von IF-Verzweigungen überhaupt mal hereinholen, und das am Anfang.

Diese nun folgenden zwei Variablen sollen gleich ganz oben angeführt sein, noch vor dem ersten direkten Vergleich, ob die Variable $username eingegeben wurde. Daher setze dich auf die Zeile 40

und 41, um die zwei Variablen festzulegen: Achtung – icht mit den gleichen verwechslen, die in Zeile 32 und 33 stehen!

```
38      if($isPost){
39
40          $username =
41          $password =
42
43          if((bool)$username === false){
44              $errors[]="Benutzername ist leer";
```

Man verwendet hier eine „gefilterte“ Übergabe, die sicherer die Werte aus Formularfeldern übergibt. Dabei wird absichtlich eingegebener HTML-Code oder JavaScript-Code weggefiltert.

- Der erste Parameter gibt an, woher die Daten ausgelesen werden – INPUT_POST
- Der Name des Eingabefeldes – ist hier gleich
- Der Filter – hier werden Zeichen wie Kleiner- oder Größerzeichen, Hochkomma und & entfernt. Diese werden dann nicht im username vorkommen.

```
30  if(strpos($route,'/login') !== false) {
31      $isPost = strtoupper($_SERVER['REQUEST_METHOD']) ==='POST';
32       $username ="";
33       $password = "";
34
35       $errors = [];
36       $hasErrors = false;
37
38       if($isPost){
39       $username = filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPECIAL_CHARS);
40       $password = filter_input(INPUT_POST,'password');
```

$username = filter_input(INPUT_POST,'username',FILTER_SANITIZE_SPECIAL_CHARS);

Beim Passwort möchte man schon Sonderzeichen und wird daher diesen Filter nicht verwenden.

```
41          $password = filter_input(INPUT_POST,'password');
```

$password = filter_input(INPUT_POST,'password');

Speichern.

Die Funktion für die Abfrage aus der DB erstellen.

Nun benötigt man diese Funktion „getUsername“ in der Datei „user.php“.

Die Funktion liefert ein Array aus der SQL-Abfrage und ermöglicht wiederum die sichere Variante mittels „prepairded statements“:

```php
function getUsername(string $username):array{
    $sql ="SELECT id,password
            FROM users
            WHERE username=:username";

    $statement = getDB()->prepare($sql);
    if($statement === false){
        return[]; //ein leeres array wird zurückgegeben
    }
    $statement->execute([
        ':username'=>$username
    ]);

}
```


```php
function getUsername(string $username):array{
    $sql ="SELECT id,password
            FROM users
            WHERE username=:username";

    $statement = getDB()->prepare($sql);
    if($statement === false){
        return[]; //ein leeres array wird zurückgegeben
    }
    $statement->execute([
        ':username'=>$username
    ]);
}
```


Darunter folgt aber noch eine IF-Abfrage, wo geschaut wird, ob überhaupt ein Wert in der Datenbank vorhanden ist, wenn der Wert Null ist, dann wird nur ein leeres Array zurückgegeben.
Ansonsten wird das Array aus der Datenbank zurückgegeben (return $row).

```php
    ]);
    if(0 === $statement->rowCount()){
        return [];
    }
    $row = $statement->fetch();
    return $row;
}
```

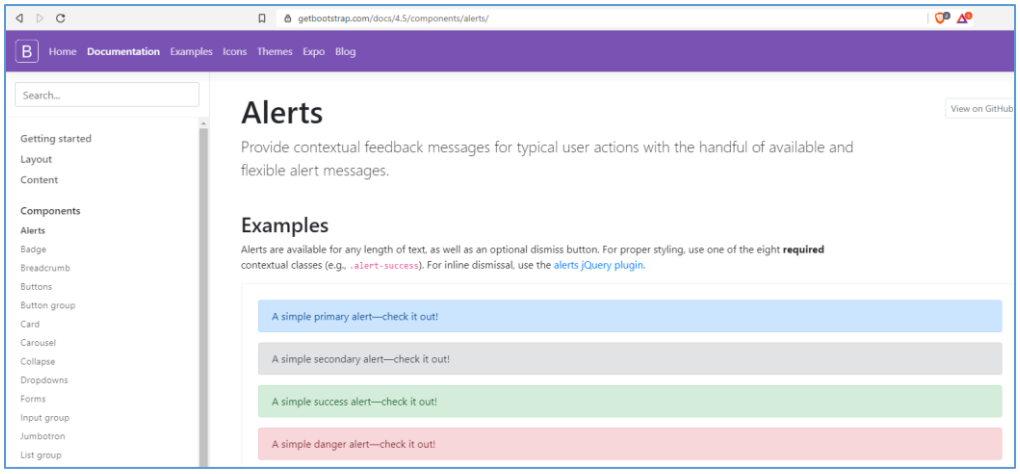

```php
if(0 === $statement->rowCount()){
    return [];
}
$row = $statement->fetch();
return $row;
```


Nun ist die Funktion fertig.

5.6)Fehlermeldungen ausgeben lassen

Damit der Benutzer die Information bekommt, wenn er Fehler macht, wie z.B. ein falsches Passwort eingibt, sollten diese ihm auch angezeigt werden.

Dafür gibt es in Bootstrap die „alert“ – Möglichkeit. Zu finden auf
https://getbootstrap.com/docs/4.5/components/alerts/



Das beste ist das mit der roten Farbe, da fällt am besten auf. Der Code sieht so aus:

```
<div class="alert alert-danger" role="alert">
A simple danger alert—check it out!
</div>
```

Das muss in der Datei des Formular eingetragen werden, weil der Benutzer dann sofort sieht, wenn es zu einer Fehlermeldung aufgrund seiner unkorrekten Eingaben kommt:
Öffne daher wieder:

login.php

Mit Verwendung des obigen Codes arbeitet man un und davor kommt eine IF-Abfrage,

- Ob es eine Fehlermeldung gibt
- Dann wird durch alle Fehlermeldungen mit „foreach“ durchgegangen und
- Darunter mit einem einfachen p-Tag diese ausgegeben
- Die IF muss aber wieder geschlossen werden – „endif“

```
22          <div class="card-body">
23          <?php if($hasErrors):?>
24          <div class="alert alert-danger" role="alert">
25              <?php foreach($errors as $errorMessage):?>
26                  <p><?= $errorMessage ?></p>
27              <?php endforeach?>
28          </div>
29          <?php endif;?>
30              <div class="form-group">
```

<?php if($hasErrors):?>

```
    <div class="alert alert-danger" role="alert">
      <?php foreach($errors as $errorMessage):?>
        <p><?= $errorMessage ?></p>
      <?php endforeach?>
       </div>
<?php endif;?>
```

In „routes.php“

Erstelle ein Array:

```
54      $username = filter_input(INPU
55      $password = filter_input(INPU
56
57      $errors = [];
58      $hasErrors = false;
```

$errors = [];
$hasErrors = false;
Nun erfolgt eine Variable, die festhält, ob es Fehler gibt, sie wird von vornherein als „false“ definiert.

```
38      $errors = [];
39      $hasErrors = false;
40
```

Ganz unten, vor dem „require“ des Templates überschreibe diese Variable, wenn die Anzahl größer als Null sein sollte:

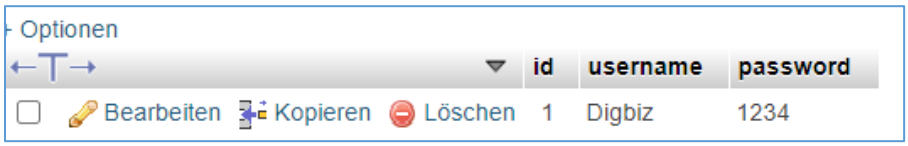
```
66      }
67
68      $hasErrors = count($errors) > 0;
69      require __DIR__.'/templates/login.php';
70        exit();
```

$hasErrors = count($errors) > 0;

Ergebnisse:

Testen:

Mit folgenden Eintrag:



WEITERENTWICKLUNG: HASH für das Login erfolgt in der Beschreibung von „registrieren“ später.

Aber lassen wir es so stehen, weil jetzt ein Skript für die Ver-hash-ung von einem Passwort zu schreiben muss nicht sein. ☺

5.7)Weiterleitung nach erfolgreichem Login – checkout-Prozess

Das Weiterleitungsziel ist die Route „checkout“. Diese besteht schon in „routes.php“.

Ein „checkout“-Prozess ist, wenn man nach dem Ausfüllen eines Formulars anschließend eine Bestellung mit Bestellnummer erzeugt.

Öffne die „routes.php“ und stelle den Cursor in die Route „checkout“.
Nachdem man eingeloggt ist, soll hier eine Variable in der SESSION definiert werden:

```
73 ▼ if(strpos($route,'/checkout') !== false) {
74 ▼     if(!isLoggedIn()){
75       //nach erfolgreichem Login
76       $_SESSION['zielEingeloggt'] = "/orangenshop/index.php/checkout";
77
78       header("Location: /orangenshop/index.php/login");
79       }
```

$_SESSION['zielEingeloggt'] = "/orangenshop/index.php/checkout";

Diese Session wird nun benutzt in dem Teil, wo festgestellt wurde, dass keine Fehler sind und alles passt, um den Benutzer erfolgreich einloggen zu lassen, nämlich in dieser „if“:
Dies ist auch in der „routes.php“ nur einige Zeilen darüber aber in der Route „login“ die nun vervollständigt wird:

- Man holt sich aus der Session die „zielEingeloggt“ und
- wenn es diese nicht geben sollte, wird auf die „index.php“ weitergeleitet
- man definiert ein „Default“-zielEingeloggt auf „index.php“ – Zeile 66
- dann eine IF: wenn in der Session ein zielEingeloogt gesetzt ist, dann wird die obige Variable „zielEingeloggt“ mit der aus der Session überschrieben.
- Anschließend wird auf das Ziel des „zielEingeloggt“ weitergeleitet mit „header“.

Füge dies nun in der **„login“-Route** ein:

```
63 ▼         if(0 === count($errors)){
64               $_SESSION['userId'] = (int)$userData['id'];
65 ▼         //weiterleiten nach erfolgreichem Login
66               $zielEingeloggt = '/orangenshop/index.php';
67 ▼             if(isset($_SESSION['zielEingeloggt'])){
68                   $zielEingeloggt = $_SESSION['zielEingeloggt'];
69               }
70               header("Location: ".$zielEingeloggt);
71               exit();
72           }
73       }
74
```

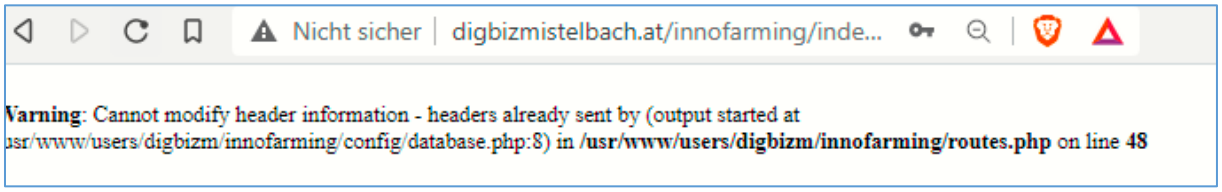
```
//weiterleiten nach erfolgreichem Login
        $zielEingeloggt = '/shop/index.php';
        if(isset($_SESSION['zielEingeloggt'])){
          $zielEingeloggt = $_SESSION['zielEingeloggt'];
        }
        header("Location: ".$zielEingeloggt);
        exit();
```

}

Zum Testen sollte man auch den Browser schließen und neu öffnen, weil man ausgeloggt sein sollte. Durch Klick auf „Zur Kassa“ wird der „zielEingeloggt“ gesetzt und ist dann vorhanden.

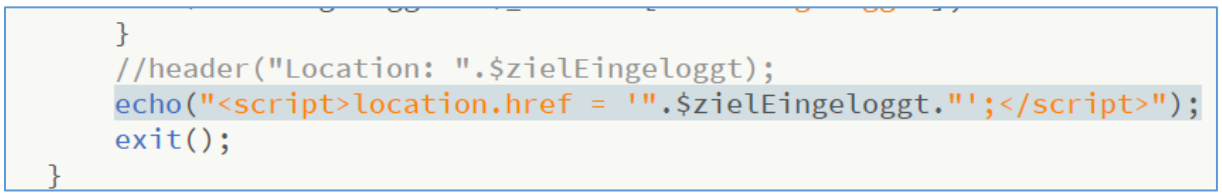
BEACHTE:

In „localhost“ ist folgendes KEIN Problem. Aber z.B. auf einem Domian-Server wie www.hetzner.de kommt es zu einer Fehlermeldung bezüglich dieser Weiterleitung in Zeile 70. Es wird durch die „session_start“ der „header“ dann nochmals beansprucht, was aber nicht möglich ist. Folgende Fehlermeldung KANN kommen:



Daher sollte man die Zeile mit der „header("Location: ".$zielEingeloggt);“ ersetzen durch folgenden Code, der ebenfalls weiterleitet

echo("<script>location.href = '".$zielEingeloggt."';</script>");



```
        }
        //header("Location: ".$zielEingeloggt);
        echo("<script>location.href = '".$zielEingeloggt."';</script>");
        exit();
    }
```

Lösung gefunden bei:

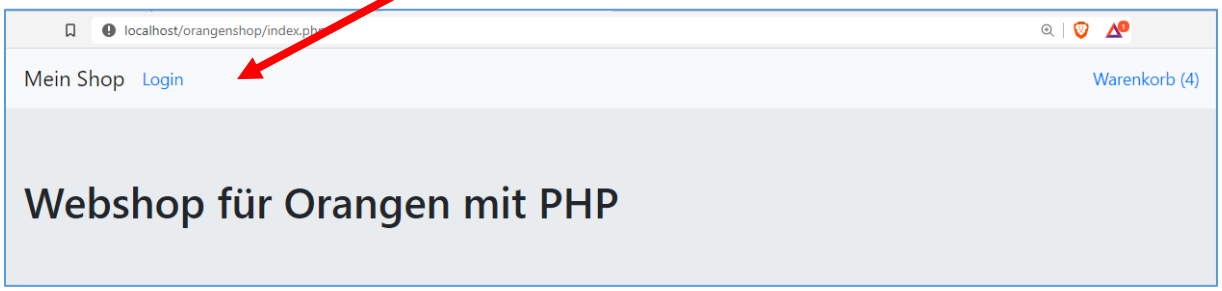
https://stackoverflow.com/questions/8028957/how-to-fix-headers-already-sent-error-in-php

It'll definitely solve your problem. I faced the same problem but I solved through writing header location in the above way.

// Lösung Ende. ☺

5.8)Login- und Logout-Button

Damit man auch in der Navbar einloggen kann, soll hier auch eine Möglichkeit entstehen, um in das Login-Formular zu kommen.



Öffne die „navbar.php“ und füge nach der Zeile 3 eine neue <ul> ein:

```
1  <nav class="navbar navbar-expand-lg navbar-light bg-light">
2  <div class="container">
3    <a class="navbar-brand" href="#">Mein Shop</a>
4      <ul class="navbar-nav">
5      <li class="nav-item">
6
7      </li>
8      </ul>
9    <ul class="navbar-nav ml-auto">
```

```
<ul class="navbar-nav">
    <li class="nav-item">
  </li>
  </ul>
```

Innerhalb des „item“ erfolgt nun die Logik: wenn man eingeloggt ist, soll der Link „Logout“ angezeigt werden, wenn man nicht eingeloggt ist, der Link „Login“. Das mit PHP-Code sieht folgendermaßen aus:

```
5      <li class="nav-item">
6          <?php if(isLoggedIn()):?>
7              <a href="index.php/logout">Logout</a>
8          <?php endif;?>
9          <?php if(!isLoggedIn()):?>
10             <a href="index.php/login">Login</a>
11         <?php endif;?>
12     </li>
```

```
<?php if(isLoggedIn()):?>
    <a href="index.php/logout">Logout</a>
  <?php endif;?>
  <?php if(!isLoggedIn()):?>
    <a href="index.php/login">Login</a>
  <?php endif;?>
```

5.9)Logout erstellen

Zuerst öffne „routes.php“ und erstelle ganz unten eine neue Route:

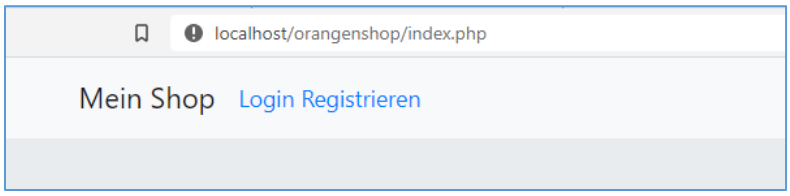
```
90 if(strpos($route,'/logout') !== false) {
91
92 }
93
```

- Zuerst wird die Session neugeneriert mit „regenerate“
- Mit „destroy“ wird die Session komplett geleert.
- Dann erfolgt die Weiterleitung

```
95 if(strpos($route,'/logout') !== false) {
96     session_regenerate_id(true);
97     session_destroy();
98     header("Location: /orangenshop/index.php");
99     exit();
100 }
```

```
if(strpos($route,'/logout') !== false) {
   session_regenerate_id(true);
   session_destroy();
   header("Location:  /orangenshop/index.php");
   exit();
}
```

Klickt man auf „Logout“ kommt man zurück und kann sich wieder anmelden:



Quelle:
https://www.youtube.com/watch?v=cXZG4yJhLsQ und
https://www.youtube.com/watch?v=2NDq7tojttc bei 18:50 (Login- und Logout-Button)