9)Webshop – Suche (PDO)

1)Datei „suche.php“ erstellen
2)Navbar erneuern
3)Route für Suche anlegen
4)“suche.php“ erweitern
5)Datei „sucheErgebnis.php“ erstellen inkl. SQL-Abfrage – keine „ordentliche“ Umleitung über eine Route. Dadurch ging es schneller ☺

1)Erstelle eine neue Datei namens „suche.php“ im Ordner „templates“.

Am besten man speichert die Datei „main.php“ unter dem Namen „suche.php“ um vieles zu übernehmen.

Öffne die neue „suche.php“ und lösche aber dann den Inhalt im „container“ um dann fortsetzen zu können.

```
11 ▼  <body>
12      <?php include __DIR__.'/navbar.php' ?>
13 ▼    <header class="jumbotron" >
14 ▼    <div class="container" >
15      <h1>Webshop für Orangen mit PHP</h1>
16      </div>
17      </header>
18 ▼    <section class="container" id="loginForm" >
19
20      </section>
21      <script src="assets/js/bootstrap.js"></script>
```

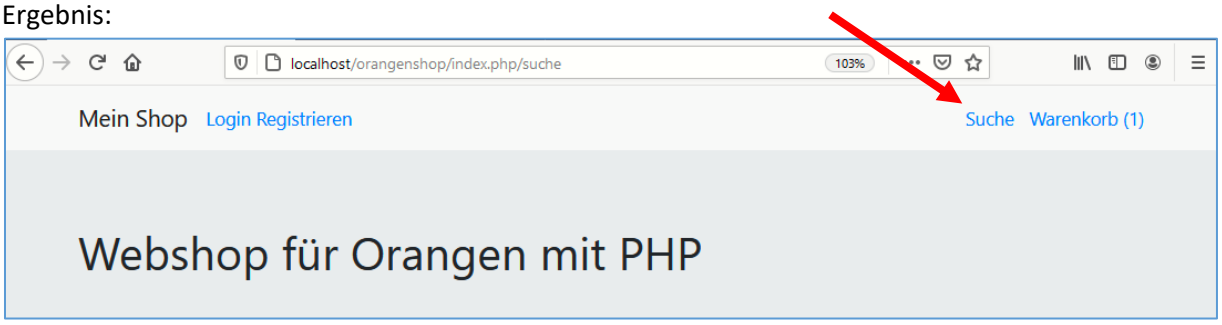
2)Navbar erneuern

Öffne die „navbar.php“.

Füge eine neue „li nav-item“ in der Navbar ein. Beachte auch die beiden geschützten Leerzeichen, damit es einen Abstand zum Warenkorb hat:


```
15 ▼  <ul class="navbar-nav ml-auto">
16 ▼      <li class="nav-item">
17        <a href="index.php/suche">Suche</a>
18      </li>
19 ▼      <li class="nav-item" >
20               
21        </li>
22 ▼    <li class="nav-item">
23        <a href="index.php/warenkorb">Warenkorb (<?
```

Ergebnis:



3)Route anlegen

Auch hier muss wieder eine Route angelegt werden:
Öffne die „routes.php“ und füge ganz unten ein:

```
136 if(strpos($route,'/suche') !== false) {
137     require __DIR__.'/templates/suche.php';
138     exit();
139 }
```

```
if(strpos($route,'/suche') !== false) {
   require __DIR__.'/templates/suche.php';
   exit();
}
```

4)Datei „suche.php“ erweitern

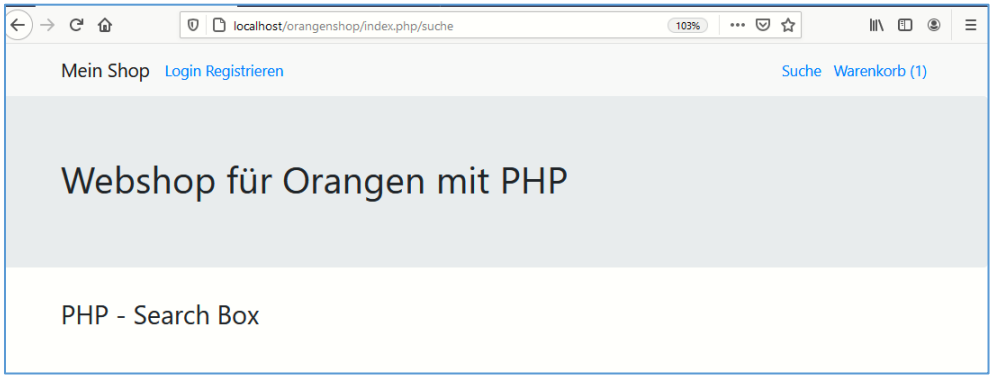
Füge unter der Überschrift folgenden Code ein:

```
18     <section class="container" id="loginForm" >
19     <div class="row">
20     <div class="col-md-6">
21         <h3 class="text">PHP - Search Box</h3>
22         </div>
23     </div> <!--ende row -->
24
25     </section>
```

Zuerst den oberen Teil:

```
<div class="row">
   <div class="col-md-6">
      <h3 class="text">PHP - Search Box</h3>
      </div>
   </div> <!--ende row -->
```

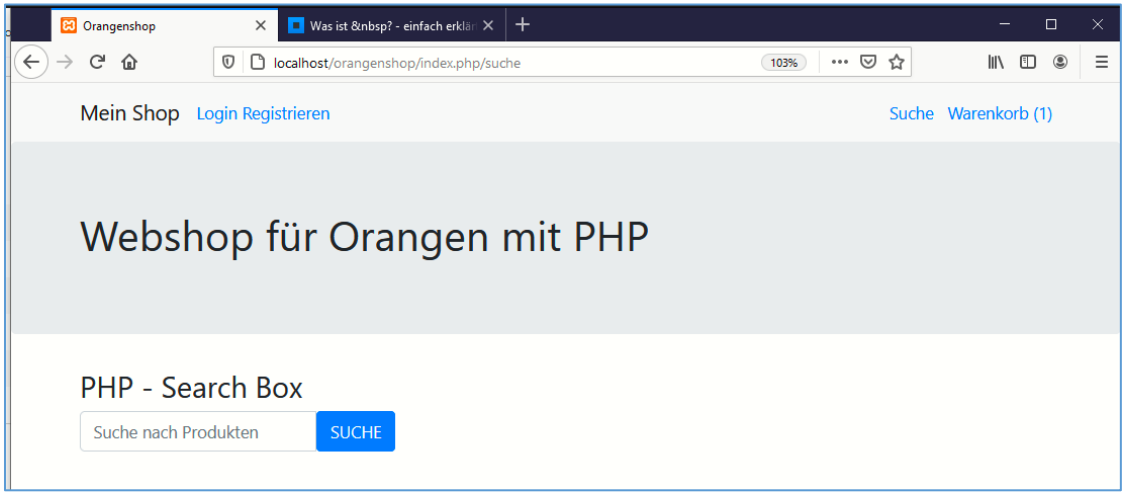
Dabei wird nur die Info über den späteren Eingabebereich erstellt:



Darunter nun der Eingabebereich, der auch gleichzeitig ein Formular ist, weil der Klick auf die „SUCHE“ ja die Eingabe weiterleiten soll:

```
<form action="templates/sucheErgebnis.php" method="POST" class="form-inline">
  <div class="input-group">
  <input type="text" name="suchbegriff" placeholder="Suche nach Produkten" class="form-control">
  <button type="submit" class="btn btn-primary" name="suchbutton">SUCHE</button>
  </div>
  </form>
```

Ergebnis:



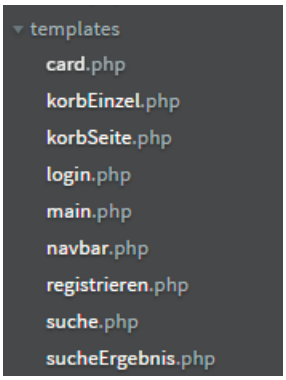
Info:

- In der „action“ steht die Datei, die dann den PHP-Code mit der Datenbankabfrage enthält.
- Die Klasse „form-inline“ zwingt dazu, dass es in einer Zeile bleibt
- Input-group ist zur Verschönerung mittels bootstrap
- Die „SUCHE“ wird im Button integriert

5)Erstellung der Datei „sucheErgebnis.php“ im Ordner „templates“ neben „suche.php“

Im Prinzip ist es genauso aufgebaut wie die bereits vorhandene Datei „main.php“, nämlich

- Navbar ist bereits eingebaut
- Darunter werden die Produkte dargestellt (in der main.php alle die mit „index.php“ auch gezeigt werden.
- In unserem jetzigen Fall sollen diejenigen Produkte genauso angezeigt werden, aber abhängig von der Auswahl durch das Suchfeld.



Daher kopiere die „main.php“ als „sucheErgebnis.php“ im gleichen Verzeichnis.

Öffne dann die neue „sucheErgebnis.php“.

```
11    <body>
12      <?php include __DIR__.'/navbar.php' ?>
13      <header class="jumbotron" >
14      <div class="container" >
15      <h1>Webshop für Orangen mit PHP</h1>
16      </div>
17      </header>
18      <section class="container" id="produkte" >
19      <div class="row">
20          <?php while($row = $result->fetch()):?> <!--hier der Doppelpunkt-
21              <div class="col">
22                  <?php include 'card.php'?>
23              </div>
24          <?php endwhile;?>
25      </div>
26      </section>
27      <script src="assets/js/bootstrap.js"></script>
28    </body>
```

Eine H1 zur besseren Übersicht könnte man einfügen – gleich nach dem „container“:

```
  </header>
  <section class="container" id="produkte" >
  <h1>Ihr Suchergebnis:</h1>
  <div class="row">
      <?php while($row = $result->fetch()):?> <!
```

5a)Korrekte Einbindung der navbar:

In der PHP der „navbar.php“ muss man nun noch die Verbindung herstellen, da wir hier nicht über die Route arbeiten.

- Zuerst die „user.php“ einbinden, da sonst die „userId“ nicht vorhanden ist und eine Fehlermeldung bezüglich „isLoggedIn“ kommt.
- Darunter die Zeile 8 aus der „route.php“ – nämlich, dass die „userId“ übernommen wird.

- Erst dann die „korb.php“ einbinden, weil wir in der nächsten Zeile die „userId“ benötigen in der Funktion, die man für den Button ganz rechts benötigt, den mit dem Warenkorb:
- Die Variable korbZahl aus der routes.php Zeile 9.

```
10    <body>
11      <?php
12          //weil wir hier nicht über die routes.php gehen:
13          //für die navbar - loggedIn und Warenkorb diese 4 Zeilen:
14          include '../function/user.php';
15          $userId = getCurrentUserId();
16          include '../function/korb.php';
17          $korbZahl = countProductInKorb($userId);
18
19      include __DIR__.'/navbar.php';
20      ?>
21      <header class="jumbotron" >
```

//weil wir hier nicht über die routes.php gehen:
//für die navbar - loggedIn und Warenkorb diese 4 Zeilen:
include '../function/user.php';
$userId = getCurrentUserId();
include '../function/korb.php';
$korbZahl = countProductInKorb($userId);

5b)Nun folgt noch die SQL-Abfrage

Über den gerade eingebundenen PHP-Code kann man die SQL-Abfrage erstellen.
Zuerst muss man auf die Datenbank hinkommen.

```
12      <?php
13      //SQL-Abfrage erstellen
14      //zuerst den Datenbankzugriff auf eine neue database1
15      include '../function/database1.php';
```

Dazu kann aber wegen einer Fehlermeldung die alte „database.php“ in dem Ordner „function“ nicht so ohne weiteres genommen werden. Um die Fehlermeldung zu vermeiden kopiere die „database.php“ in eine „**database1.php**“.

Dort ändere dann eine Zeile:

- Alt (database.php)

```
7       }
8       require_once CONFIG_DIR.'/database.php';
9       $dsn =
```

- Neu (database1.php)

```
7       }
8       require_once '../config/database.php';
9       $dsn =
```

Nun folgt die typische Sicherheits - IF-Abfrage, ob der Button überhaupt geklickt wurde in der Datei „suche.php“. Natürlich ist er das, daher

```
12      <?php
13      //SQL-Abfrage erstellen
14      //zuerst den Datenbankzugriff auf eine neue database1
15      include '../function/database1.php';
16
17      if (isset($_POST['suchbutton'])) {
18          $suchbegriff= $_POST['suchbegriff'];
19
```

Code:

```
if (isset($_POST['suchbutton'])) {
    $suchbegriff= $_POST['suchbegriff'];
  }
```

Dabei wird auch die Übergabe aus dem Formular geregelt, und mit $_POST übergeben (der betreffende Inhalt mit dem: name=“suchbegriff“).

ABFRAGE SQL – mit LIKE:

In der Abfragevariante „PDO“ (etwas geringfügig anders als bei „mySQLi“) muss man die Möglichkeit so gestalten, dass man ungefähre Daten eingibt, und dann aus den Datenbankfeldern die passenden Daten geholt werden. Man verwendet dazu üblicherweise das „LIKE“.
Hier muss man den Suchbegriff mit „bindValue“ an den SQL-Befehl binden:

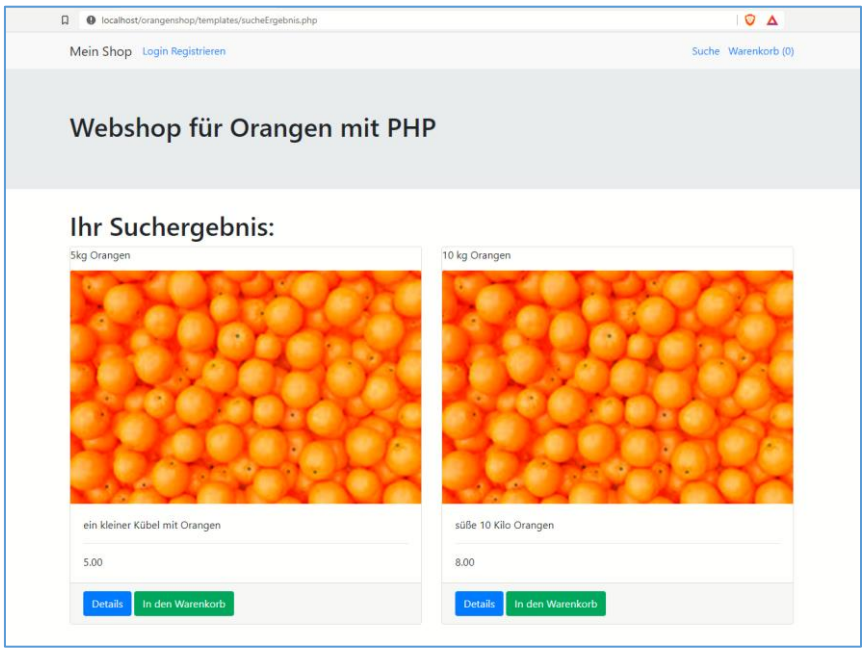
```
19
20      //query in PDO mit LIKE inkl. bindValue
21      $sql ="SELECT * FROM products
22              WHERE titel LIKE :suchbegriff OR beschreibung LIKE
                :suchbegriff";
23
24      $result = getDB()->prepare($sql);
25      $result->bindValue(':suchbegriff','%'.$suchbegriff.'%');
26      $result->execute();
27      }
28
29      //weil wir hier nicht über die routes.php gehen:
```

Code:

```
//query in PDO mit LIKE inkl. bindValue
  $sql ="SELECT * FROM products
        WHERE titel LIKE :suchbegriff OR beschreibung LIKE :suchbegriff";

  $result = getDB()->prepare($sql);
  $result->bindValue(':suchbegriff','%'.$suchbegriff.'%');
  $result->execute();
  } //Klammer von der IF
```

Ergebnis:



TIPP: wenn z.B. in zwei Formular-Inputfelder Werte eingegeben werden können, dann kann statt dem „OR“ auch ein „AND“ sinnvoll sein:

```php
<?php
session_start();

    //SQL-Abfrage erstellen
    //zuerst den Datenbankzugriff auf eine neue database1
    include 'includes/verbindung.php';

    if (isset($_POST['suche_submit'])) {
        $autor= $_POST['autor'];
        $titel= $_POST['titel'];

    //query in PDO mit LIKE inkl. bindValue

    $result   $pdo >prepare("SELECT * FROM medium WHERE Autor LIKE :Autor AND Titel LIKE :Titel");

    $result->bindValue(':Autor','%'.$autor.'%');
    $result->bindValue(':Titel','%'.$titel.'%');
    $result->execute();
    } //Klammer von der IF
?>
```