4)Webshop in PHP – Warenkorb-Übersicht anzeigen

INFO:

Bestellen geht jetzt noch ganz normal, aber später nur, wenn man eingeloggt ist. Hat man den Warenkorb vor dem Login (der wird erst später erstellt) befüllt, und loggt sich dann erst ein, ist der Warenkorb wieder leer.

1.1)Link erstellen

Der Text „Warenkorb()“ rechts oben muss verlinkt werden.

Dies passiert in der Datei in den Templates, nämlich in „navbar.php“. Öffne diese.

Nun wird hier kein komplett neuer Dateiname angegeben, sondern eine Route zu index:

```
5     <li class="nav-item">
6       <a href="index.php/warenkorb">Warenkorb (<?= $korbZahl ?>)</a>
7     </li>
```

1.2) auch „routes.php“ erweitern - eine neue Route anlegen – reine Startseite

Damit das auf einen Klick reagieren kann, muss man die Datei „routes.php“ ebenfalls erweitern. Öffne „routes.php“.

Nun erstellt man eine Route, die die reine „index-Seite“ beinhalten soll. Das wäre somit die Startseite, oder anders gesagt, das ist, wenn man hinten keine weiterführenden Slash anhängt, wie z.B. „korb“.

Daher legen ober der ersten „if“-Route eine neue an.

- Diese sagt aus, dass keine Route vorhanden ist, d.h. man gibt ein „!$route“.
- Diese erhält aus der „index.php“ 3 Zeilen, welche die „index.php“ beschrieben haben, nämlich folgende, die man aus der „index“ ausschneidet und hier einfügt.
  Beachte: die Zeile mit der Variable „$korbZahl“ wird oberhalb der „if“ hineinkopiert, da sie global wirken soll, nicht nur in dieser Seite!

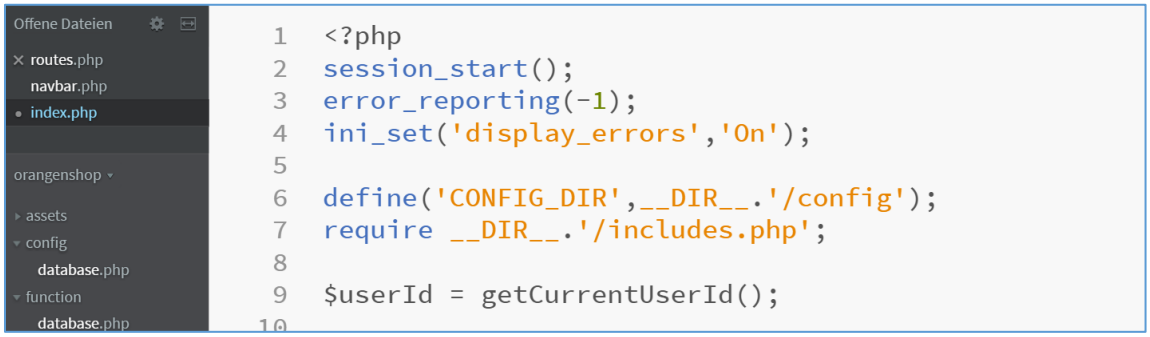
  ```
  $korbZahl = countProductInKorb($userId);
  $result = getAllProducts();
  require __DIR__.'/templates/main.php';
  ```

- Füge noch vor der Klammer ein „exit()“ hinzu.

```
 8  $userId = getCurrentUserId();
 9  $korbZahl = countProductInKorb($userId);
10
11  if(!$route){
12      $result = getAllProducts();
13      require __DIR__.'/templates/main.php';
14      exit();
15  }
```

Hier werden somit die Inhalte geladen und das Template gerendert.

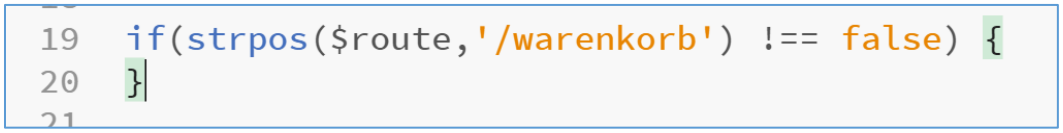
Zum Vergleich und Kontrolle: die „index.php“ ist noch kleiner geworden:



1.3.)eine weitere Route anlegen – für /warenkorb

Damit das auf einen Klick reagieren kann, muss man die Datei „routes.php“ ebenfalls erweitern. Öffne „routes.php“.

Kopiere die Zeile 17 und füge sie ganz unten an. Ändere nach dem Beistrich den Pfad auf „warenkorb“:



Klickt man nun bereits auf den neuen Link, wird die Seite zwar angezeigt, aber noch nicht korrekt. Die Layouts passen nicht hinein, da deren Pfade (CSS) noch nicht korrekt sind.

Daher muss die Route nun erweitert werden. Es muss auch hier ein Template ausgegeben werden, und zwar von einer Seite, die man erst erzeugen muss.

Kopiere die Zeile von oben mit dem „require“ inkl. „exit“ und ändere auf die neue Datei. Der Name „korbSeite.php“ klingt gut.

```php

if(strpos($route,'/warenkorb') !== false) {

    require __DIR__.'/templates/korbSeite.php';
    exit();
}
```

Um diese nicht neu erstellen zu müssen, dupliziere die „main.php“ und benenne diese dann um in „korbSeite.php“.

- Öffne danach diese neue „korbSeite.php“ und entferne die „while-Schleife“. Die wird hier nicht benötigt.

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

Speichern.

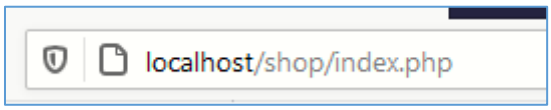
Ergebnis:

Verwendet man die Vorlage des Shops aus der ZIP-Datei, arbeitet man ab hier weiter. Bis hier her ist alles bereits inkludiert.

ABER: in der Datei „routes.php“ muss man in

Zeile 22 die Umleitung anpassen. Statt der „/shopzip/index.php“ muss man den eigenen Namen, den man dem Ordner in XAMPP nach dem entzippen gegeben hat, geben. Z.B. „/shop/index.php“.

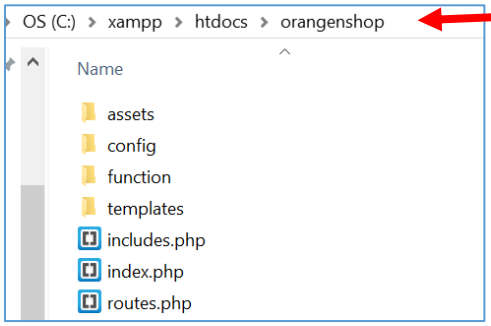
```
21        addProductToKorb($userId,$productId);
22        header("Location: /shopzip/index.php");
23        exit();
24    }
```



2)CSS-Dateien können noch nicht geladen werden – <base href>

In HTML kann man eine globale URL definieren.

Dafür verwendet man <base href>. Füge daher in der „korbSeite.php“ im <head> dies ein: Dabei wird der Ordnername verwendet, der die komplette Website beinhaltet, hier ist das „orangenshop“

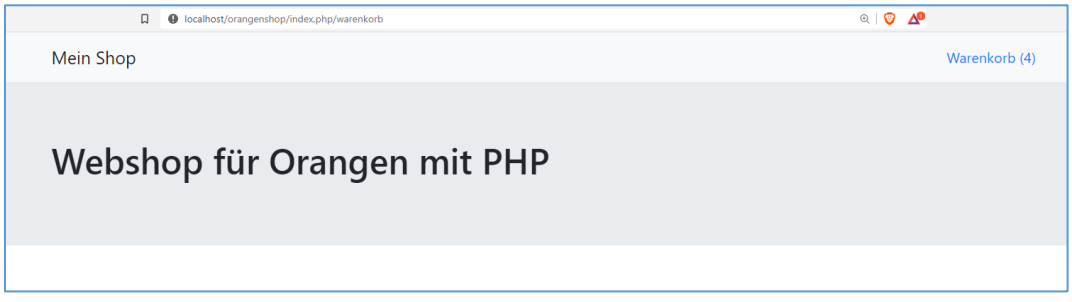


<base href="/orangeshop/">

```
5        <meta name="viewport" content="width=device-width, initia
         shrink-to-fit=no">
6        <base href="/orangenshop/">
7        <link rel="stylesheet" href="assets/css/bootstrap.css">
```

Nun werden alle weiteren „href“ damit versehen, d.h. diese „base“ wird vorangestellt an alle Links.

Ergebnis:



Aber Achtung:
wird diese Website eventuelle in einen anderen Space (z.B. einem Hoster wie hetzner.de) kopiert, muss dieser Name gleichbleiben oder hier angepasst werden.

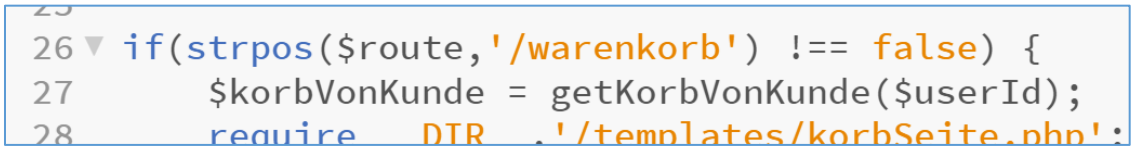
Auch bei „main.php“:
Übernimm diese „base-Zeile“ auch sofort für „main.php“ bevor man vergisst:

https://www.youtube.com/watch?v=by3rO8Wm6zQ&list=PLz858EFEcxiElOUr7b3Pql1CMHuVRYkTh&index=7 6:50

3)Elemente auf einer neuen Warenkorb-Seite anzeigen

3a)routes.php

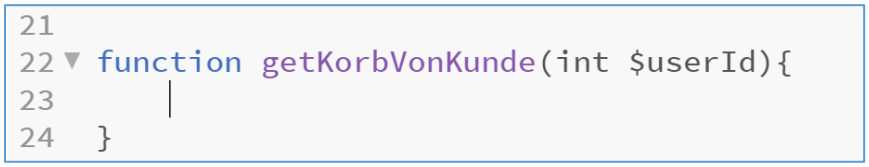
Dafür erstelle in der Route des Warenkorbs eine Variable mit einem neuen Funktionsnamen, der auf den Inhalt hindeutet. Quetsche diese zwischen die eben erstellen Zeilen. Diese Funktion wird anschießend in der „korb.php“ erst erzeugt:



$korbVonKunde = getKorbVonKunde($userId);

3b)öffne die „korb.php“.

Lege ganz unten eine neue Funktion an, die den eben erstellten Namen hat. Darin wird die „userId“ übergeben.



Nun folgt die SQL-Abfrage, die einen Join beinhaltet, um auf zwei Tabellen gleichzeitig zugreifen zu können.
BEACHTE: bei einem JOIN muss man die Elemente genau zuordnen. Da die ID in unterschiedlichen Tabellen vorkommt, so wie auch die „anzahl“ muss man ganz genau davor mit einem Punkt getrennt die genaue Tabelle dazuschreiben, wie z.B. products.id und korb.anzahl
BEACHTE:
Hier muss man die „korb.anzahl“ verwenden, nicht die aus „products“, weil sie im Korb später auch hochgezählt werden kann, mit einem button „add“.

Tipp: Daher ist zumindest bei der ID es besser, diese schon in der Datenbank unterscheidbarer zu machen, wie z.B. user_id und products_id zu nennen, statt zweimal „id“.

```
function getKorbVonKunde(int $userId):array{
    $sql = "SELECT products.id,titel,beschreibung,preis,korb.anzahl,produktbild
            FROM products
            JOIN korb
            ON korb.product_id = products.id
            WHERE korb.user_id = ". $userId;
```

$sql = "SELECT products.id,titel,beschreibung,preis,korb.anzahl,produktbild FROM products
JOIN korb
ON korb.product_id = products.id
WHERE korb.user_id = ". $userId;

Nun wird gleich in der Funktion festgelegt, dass es sich um ein Array handelt. Gibt daher :array vor der Klammer ein:

```
21
22 function getKorbVonKunde(int $userId):array{
```

- Darunter wird in Zeile 28 und 29 mit IF geklärt, dass ein leeres Array ausgegeben wird, wenn entweder die Abfrage falsch ist oder kein Eintrag vorhanden ist.

```
27              WHERE korb.user_id = ". $userId;
28
29      $results = getDB()->query($sql);
30      if($results === false){
31          return [];
32      }
```

$results = getDB()->query($sql);
if($results === false){
return [];

- Ansonsten wird ein Array „$korbVonKunden“ mit den gefundenen Produkten erstellt

```
33      $korbVonKunde = [];
```

$korbVonKunde = [];

- Danach wird durch das gesamte „results“ gearbeitet und die gefundenen Werte in die Variable gelegt

```
34      while($row = $results->fetch()){
35          $korbVonKunde[]=$row;
36      }
```

while($row = $results->fetch()){
$korbVonKunde[]=$row;
}

- Diese werden dann mit „return“ zur Verfügung gestellt

```
36      }
37      return $korbVonKunde;
38  }
```

return $korbVonKunde;

3c)korbSeite.php

Wechsle wieder in die „korbSeite.php“.
Es sollen nun die gefundenen Produkte aufgelistet werden.

Dafür füge ein „foreach“ ein.

```
18    <section class="container" id="produkte" >
19          <?php foreach($korbVonKunde as $korbVonKundeAnzeige):?>
20            <div class="row">
21                <?php include __DIR__.'/korbEinzel.php';?>
22            </div>
23        <?php endforeach;?>
24    </section>
```

```
<?php foreach($korbVonKunde as $korbVonKundeAnzeige):?>
    <div class="row">
     <?php include __DIR__.'/korbEinzel.php';?>
  </div> <!-- Ende row -->
    <?php endforeach;?>
```

Das „foreach“-Statement erfolgt vor der <row> und bezieht sich auf die Variable aus den eben erstellten „korb.php“ mit dem Namen „$korbVonKunden“ (bzw. ist gleich mit „routes.php“). Diese wird etwas anderslautende Variable nach dem „as“ verwendet. (Dies benötigt man dann in der „korbEinzel.php“).

Das Ende der „foreach“ wird ebenfalls angeschrieben.

Dazwischen wird eine Klasse <row> eingefügt, damit nach jedem gefundenen Produkt eine neue Zeile angefangen wird.

Das Layout soll die einzelnen Produkte, die ein User in den Warenkorb gelegt hat mit Bild und Text untereinander angezeigt. Das regelt im Detail eine neue Datei, die hier mittels PHP eingebunden wird. Hier werden mit „include“ die einzelnen Produkte hereingeladen.

Diese Datei wird nun erzeugt im Ordner „templates“ nämlich „korbEinzel.php“

3d)Neue Datei „korbEinzel.php“

Diese wird die einzelnen Produkte anzeigen.

Erstelle ebenfalls im Ordner „templates“ eine neue Datei namens „korbEinzel.php“.

Diese bekommt ein einfaches Layout mit Bootstrap-DIV. So ähnlich wie die „card.php“, die ja auch alle Produkte anzeigt.

- Dabei soll das Bild links und rechts daneben die Beschreibung stehen.
- Für das Bild kopiere den Code aus der Zeile 3 aus „card.php“ und ändere
    - Die col – damit sie nebeneinander sind und
    - nur den Namen der Variablen von $row auf $korbVonKundeAnzeige:

```php
<div class="col-4">
   <?php echo '<img height="150" width="200"
   src="'.base64_encode( $korbVonKundeAnzeige['produktbild']
   ).'"/>'; ?>
</div>

<div class="col-6">
     <div><?= $korbVonKundeAnzeige['titel']?></div>
     <div><?= $korbVonKundeAnzeige['beschreibung'] ?></div>
     <div>Menge: <?= $korbVonKundeAnzeige['anzahl'] ?></div>
     <div><?= $korbVonKundeAnzeige['preis'] ?></div>
</div>
```

 <?php echo ''; ?>

  <?= $korbVonKundeAnzeige['titel']?>
  <?= $korbVonKundeAnzeige['beschreibung'] ?>
  Menge: <?= $korbVonKundeAnzeige['anzahl'] ?>
  <?= $korbVonKundeAnzeige['preis'] ?>

Ergebnis:

4)Seite verschönern

Öffne die „korbSeite.php“.
Hier ändere den Namen der „id“ in der <section> auf „seiteWarenkorb“ und um alle Elemente darin bearbeiten zu können erstelle mit auch gleich eine Klasse neben der „row“. Damit kann man alle Elemente in der ID bearbeiten.

```php
    <section class="container" id="seiteWarenkorb" >
          <?php foreach($korbVonKunde as $korbVonKundeAnzeige):?>
            <div class="row seiteWarenkorb">
                <?php include __DIR__.'/korbEinzel.php';?>
```

Danach kann in der bereits vorhandenen CSS-Datei „style.css“:

Beachte: kein Leerzeichen vor der geschwungenen Klammer

- Es soll unter jedem Eintrag unterhalb eine Linie gezogen werden

```
#seiteWarenkorb .seiteWarenkorb{
    border-bottom: 1px solid #DDD;
}
```

- Abstand oben und unten:

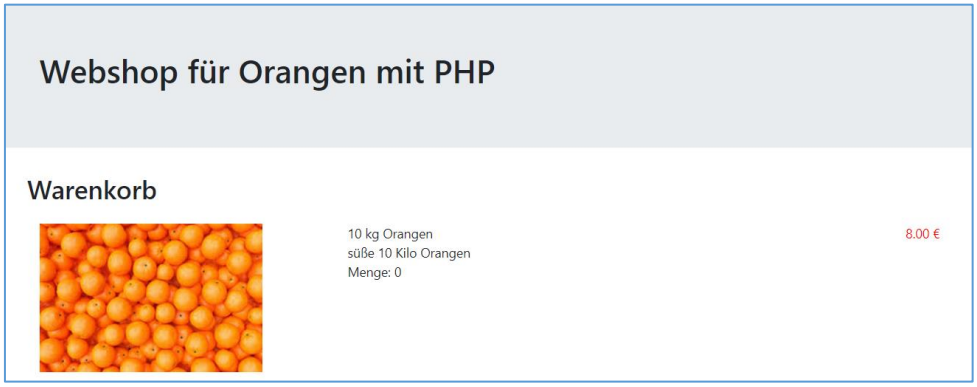
```
#seiteWarenkorb .seiteWarenkorb{
    border-bottom: 1px solid #DDD;
    padding-top: 15px;
    padding-bottom: 15px;
}
```

Ergebnis ist nun schon schöner:

Die Bootstrap-Grid Verteilung soll sich noch zu Gunsten einer eignen Spalte für den Preis ändern, damit dieser extra steht und besser gesehen wird:
Öffne die „korbEinzel.php“ und

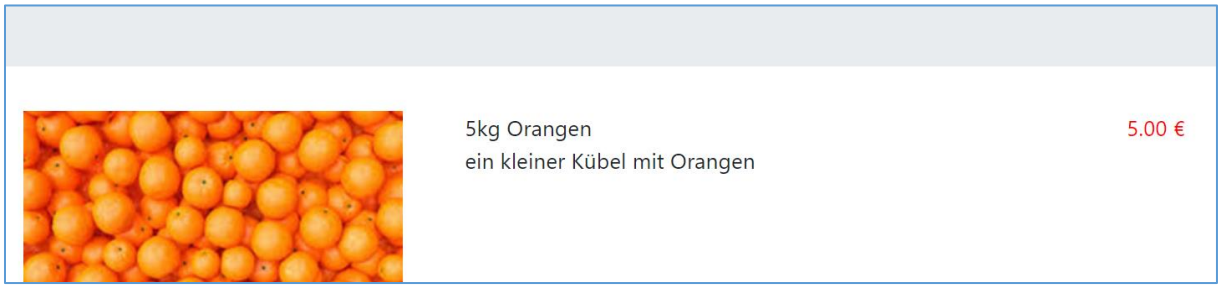
- erstelle ein neue Spalte mit „col-2“,
- schneide die Zeile mit dem Preis aus um sie hier einzufügen und
- reduziere die mittlere Spalte von 8 auf 6

Ergebnis:



- Damit der Preis rechtsbündig angezeigt wird, verwende die Bootstrap-Klasse „text-right“.
- Farbe soll rot sein.
- Ein Eurozeichen am Ende

```
11 ▼ <div class="col-2 text-right">
12       <div style="color:red;"><?= $korbVonKundeAnzeige['preis'] ?> €
13       </div>
14   </div>
```



https://www.youtube.com/watch?v=qOt-k7o7hS0&list=PLz858EFEcxiElOUr7b3Pql1CMHuVRYkTh&index=8 7:30

4a)Überschrift „Warenkorb“ einfügen

Füge eine neue „Klasse“ container ein und dazwischen eine Überschrift „Warenkorb“:

<section class="container" id="seiteDesign" >

```
18 ▼     <section class="container" id="seiteDesign" >
19 ▼         <div class="row">
20               <h2>Warenkorb</h2>
21           </div>
22             <?php foreach($korbVonKunde as $korbVonK
```

Ergebnis:

Abstand zwischen den Produkten mit einem zusätzlichen „echo“ und einem geschütztem Leerzeichen in HTML: d.h. no break space – achte auf den Strichpunkt hinten (ist nötig)

```
22          <?php foreach($korbVonKunde as $korbVonKundeAnzeige):?>
23 ▼            <div class="row ">
24                  <?php include __DIR__.'/korbEinzel.php';
25                  echo " "?>
26              </div>
27          <?php endforeach;?>
```

4b)Unterhalb ein Button mit „Zur Kasse gehen“

Nach dem Ende der Schleife, aber noch innerhalb der <section> erstelle einen neue <row> und einen Link mit Button-Style.

```
26          <?php endforeach;?>
27          <div class="row">
28          <br>
29          <a href="#" class="btn btn-primary col-12">Zur Kasse gehen</a>
30          </div>
31      </section>
```

Zur Kassa gehen

Darüber soll die Gesamtsumme der Waren im Warenkorb angezeigt werden.

Dafür wird wieder eine <row> erstellt mit 12 Spalten und dem Text rechtsbündig:

```
26          <?php endforeach;?>
27          <div class="row">
28              <div class="col-12 text-right"></div>
29          </div>
30          <div class="row">
```

Summe (<?= $korbZahl ?> Artikel)

4c)Summe Artikelanzahl

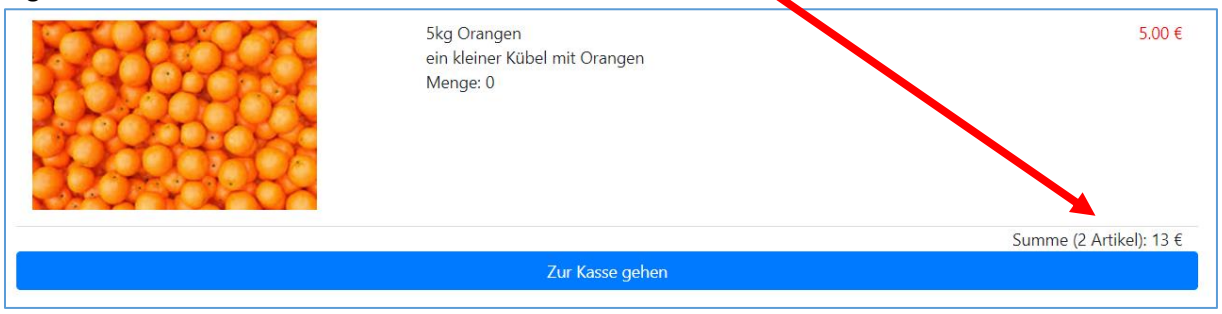
Für die Anzahl der Artikel kann man auf eine alte Variable zurückgreifen, die schon in der Datei „routes.php“, nämlich

```
9   $korbZahl = countProductInKorb($userId);
```

Daher schreibe:

```
28              <div class="col-12 text-right">
29              Summe (<?= $korbZahl ?> Artikel):
30              </div>
```

Ergebnis:



4d)Summe Gesamtpreis

Die Summe des Gesamtpreises soll in der Datei „korb.php“ mit einer neu zu erstellenden Variablen erstellt werden, die man sich dann holen und hier einbauen kann.

Öffne „korb.php“.
Erstelle ganz unten eine neue Funktion, nämlich

```
37        return $korbVonKunde;
38    }
39
40    function getSummefuerUserId{
41        
42    }
```

Dabei muss man die „userId“ übergeben und man erwartet eine Zahl als Ergebnis. In der Funktion wird man auf die Datenbank zugreifen müssen und benötigt daher eine „sql“-Abfrage:

```
40  function getSummeFuerUserId(float $userId): float{
41      $sql = "";
42  }
```

In SQL muss man die Funktion SUM() nutzen.

Einfacherweise kann man das letzte SQL-Statement kopieren und statt den einzelnen Elementen die Summe dieser errechnen lassen:

```
23  function getKorbVonKunde(int $userId):array{
24      $sql = "SELECT products.id,titel,beschreibung,pr
25              FROM products
26              JOIN korb
27              ON korb.product_id = products.id
28              WHERE korb.user_id = ". $userId;
```

Bearbeite nun die Zeile wo „SELECT“ steht:

```
40  function getSummefuerUserId(int $userId): float
41  {
42      $sql = "SELECT SUM(preis)
43      FROM products
44      JOIN korb
45      ON korb.product_id = products.id
46      WHERE korb.user_id = ".$userId;
```

Darunter wird wieder die Variable $results gesetzt. Das kann man von oben kopieren.

Darunter wieder eine **IF-Verzweigung**,

- nämlich, wenn das Resultat falsch ist, z.B. keine Ausgabe aus der Datenbankabfrage soll eine Null zurückgegeben werden.

```
46
47          $results = getDB()->query($sql);
48          if($results === false){
49              return 0;
50      }
```

- **Ansonsten** soll eine Ausgabe erfolgen. Hier genügt die einfache Form mit „fetchColumn“:

```
51      return $results->fetchColumn();
52
```

Das Problem, dass man den Wert „Null“ zurückbekommen würde, wenn keine Daten eingetragen sind, löst man mit der Konvertierung in einem Datentyp „float“. Nun wir eine Zahl zurückgegeben, die auch Kommawerte anzeigen kann..

```
50              return 0;
51          }
52          return (float)$result->fetchColumn();
53      }
```

return (float)$result->fetchColumn();

Summe des Eurobetrages ausgeben

Definition einer **neuen Variable**, die diese neue Funktion aufruft. Damit kann sie später ausgegeben werden. Wechsle in die Datei „**routes.php**“, nämlich in die Route des Warenkorbs. Dort lege diese Variable an, die sich die Daten aus der eben erstellten Funktion holt:

$korbSumme = getSummeFuerUserId($userId);

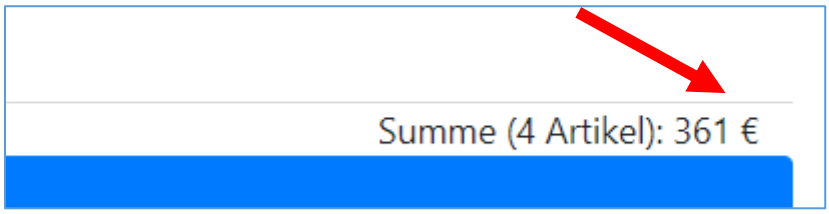
```
26  if(strpos($route,'/warenkorb') !== false) {
27          $korbVonKunde = getKorbVonKunde($userId);
28          $korbSumme = getSummefuerUserId($userId);
29          require __DIR__.'/templates/korbSeite.php';
```

Dieser Wert wird nun in die Seite des Warenkorbes übertragen, daher wechsle in die „korbSeite.php“ um diese Variable einzutragen, die nun die Summe beinhaltet. Schreibe dahinter einfach das Eurozeichen.

```
29              Summe (<?= $korbZahl ?> Artikel): <?= $korbSumme ?> €
30              </div>
```

Nach einem Speichern und Aktualisieren der Warenkorbseite sollte man nun die Summe des Einkaufes ablesen können:

Ergebnis:



5)Menge der einzelnen Produkte erhöhen

Nun soll diese Anzahl erhöht werden, wenn man ein weiteres Stück dieses Produkts addiert. Nebenbei muss sich auch die Summe in Euro erhöhen, wenn ein Produkt dazukommt.

Wie:

- Übergabe der richtigen „Produkte-ID“ durch URL Verlängerung
- Auslesen der URL und dritte Position – wie vorher – wird hier nur kopiert
- Die neue Route bietet eine neue Funktion, die um eins erhöht

5.1)Anzeige mit Button

Dazu ertelle in der „korbEinzel.php“ in der gleichen Zeile wie die Menge zwei Buttons:


```
6       <div><?= $korbVonKundeAnzeige['titel']?></div>
7       <div><?= $korbVonKundeAnzeige['beschreibung']?></div>
8       <div>Menge: <?= $korbVonKundeAnzeige['anzahl']?></div>
9       <a href="index.php/cart/addEins/<?= $korbVonKundeAnzeige['id']?>"
        class="btn btn-success col-3">+ 1</a>
10      <a class="btn btn-warning col-3">- 1</a>
11  </div>
```

<a href="index.php/cart/addEins/<?= $korbVonKundeAnzeige['id']?>" class="btn btn-success col-3">+ 1
- 1

Farbe unterschiedlich, einmal „success“ und dann „warning“.

Der PHP-Teil mit der ID dient als URL-Verlängerung, die genauso wie bei „in den Warenkorb“ auf der Seite „card.php“-Zeile 11, dazu dient, die richtige Auswahl des Produktes weiterzugeben.

Diese ID wird nämlich benötigt, um das richtige Produkt zu erhöhen, und zwar in der neu zu erstellenden Route und dort in der Funktion:

5.2)Neue Route anlegen

Öffne „routes.php“ und kopiere die Route „card/add“, um es darunter einzufügen.

Die neue Route soll nun, wie oben in „korbEinzel.php“ in Zeile 11 festgelegt heißen: „cart/addEins“:

- Neben dem Namen der Route ändere auch
- Den Namen der Funktion in „addEinsToKorb“ und
- Die Weiterleitung, die nämlich zurück zur gerade geöffneten Seite führen muss, da durch die Erhöhung um ein Stück die Menge ja neu ist.
- Die Übernahme der Variable „productId“ erfolgt genau gleich wie in der alten Route und ist somit auch nicht zu verändern

```
26 ▼ if(strpos($route,'/cart/addEins/') !== false) {
27       $routeTeile = explode("/",$route);
28       $productId = (int)$routeTeile[3];
29
30       addEinsToKorb($userId,$productId);
31       header("Location: /shop/index.php/warenkorb");
32       exit();
33   }
```

Beide untereinander: zum Vergleich - weil ja kopiert und gering verändert

```
17 ▼ if(strpos($route,'/cart/add/') !== false) {
18       $routeTeile = explode("/",$route);
19       $productId = (int)$routeTeile[3];
20
21       addProductToKorb($userId,$productId);
22       header("Location: /orangenshop/index.php");
23       exit();
24   }
25
26 ▼ if(strpos($route,'/cart/addEins/') !== false) {
27       $routeTeile = explode("/",$route);
28       $productId = (int)$routeTeile[3];
29
30       addEinsToKorb($userId,$productId);
31       header("Location: /orangenshop/index.php/warenkorb");
32       exit();
33   }
```

5.3)Neue Funktion erstellen

Öffne die „korb.php“.

Da die Funktion „addProductToKorb“ als Grundlage dienen soll, kopiere diese und füge sie gleich darunter ein.

- Prinzipiell muss nur die Erhöhung um +1 eingebaut werden. Sonst ist alles gleich, außer natürlich der Name, der sich ändert in „addEinsToKorb“
- Die Erhöhung erfolgt nur durch folgenden einfachen Code:

```
ON DUPLICATE KEY UPDATE anzahl = anzahl +1";
```

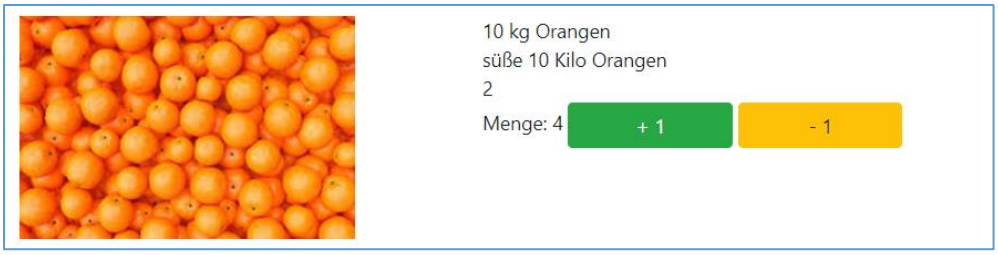
ON DUPLICATE KEY UPDATE anzahl = anzahl +1

Änderungen sind hervorgehoben: nämlich der Name der Funktion und die Erhöhung der anzahl +1:

```
15 function addEinsToKorb(int $userId, int $productId){
16       $sql = "INSERT INTO korb SET
17          anzahl=1,
18          user_id = :userId,
19          products_id = :productId
20          ON DUPLICATE KEY UPDATE anzahl = anzahl +1";
21      $statement = getDB()->prepare($sql);
22
23      $statement->execute([
24          ':userId'=>$userId,
25          ':productId'=> $productId
26      ]);
27  }
```

Testen:
auch in der Datenbank erfolgt natürlich die Erhöhung

id	product_id	user_id	anzahl	created
101	2	8	4	2020-08-22 10:34:18



https://www.youtube.com/watch?v=qOt-k7o7hS0 14:10

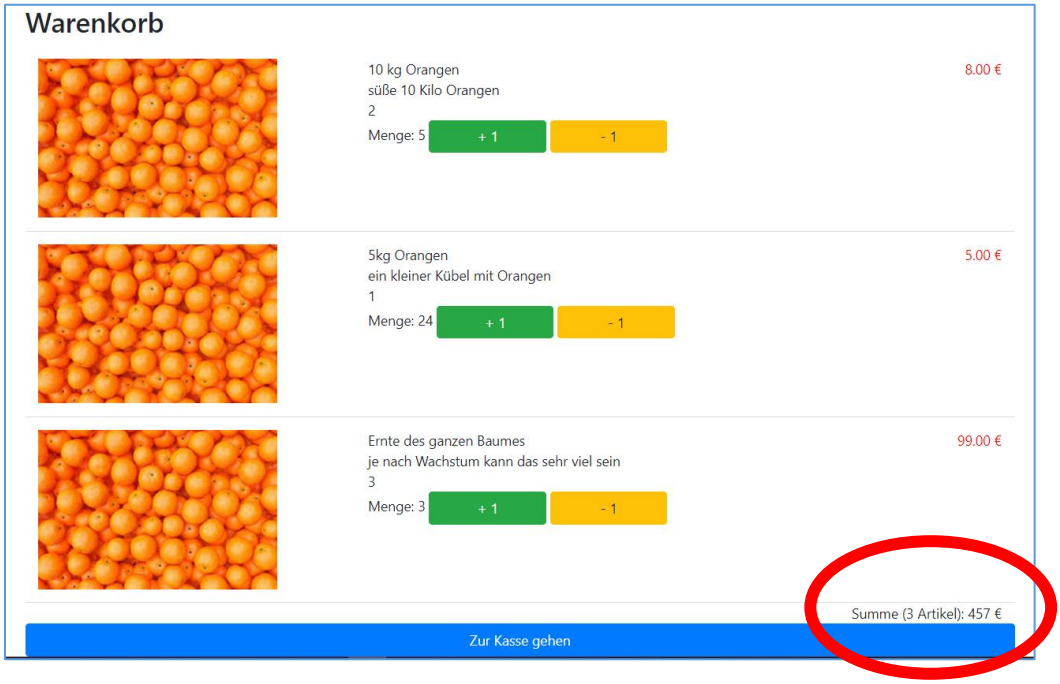
6)Summe ebenfalls erhöhen

Wenn sich die Menge erhöht, muss sich natürlich auch die Summe erhöhen.

Dafür muss man nur in der datei „korb.php“ bei der „SUM“ die richtige Anzahl, also die im Korb, mit dem Preis multiplizieren:

```
39
40  function getSummefuerUserId(int $userId): float
41  {
42      $sql = "SELECT SUM(preis * korb.anzahl)
43      FROM products
44      JOIN korb
45      ON korb.product_id = products.id
46      WHERE korb.user_id = ".$userId;
```

Die Gesamtsumme ganz unten, wo alle Produkte als Endpreis zusammengezählt werden, hat sich nun angepasst.

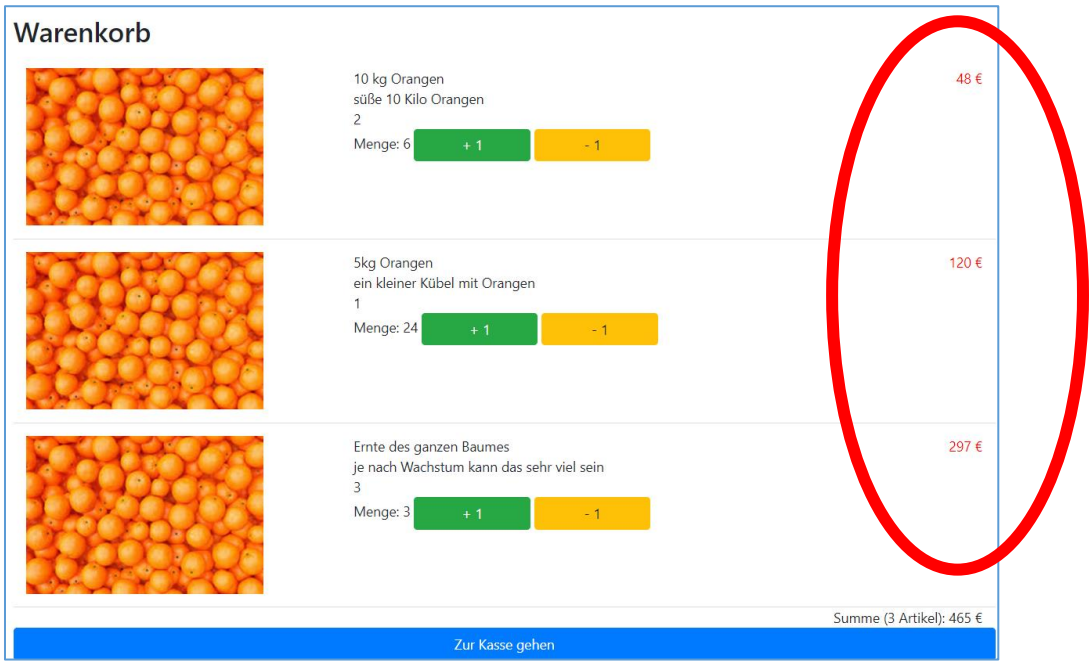


korbEinzel.php:

Damit sich auch die Preise der einzelnen Produktzeilen ändert, wenn mehr als eines bestellt wird, muss man noch in der Datei „korbEinzel.php“ die Anzeige des Preises mit der Menge multiplizieren:

```
12 <div class="col-2 text-right">
13     <div style="color:red;"><?= $korbVonKundeAnzeige['preis'] *
       $korbVonKundeAnzeige['anzahl']?> €</div>
14 </div>
```

Ergebnis:



7)Menge reduzieren

Hier das gleiche wie in Punkt 5, nur mit anderen Namen und die Subtraktion statt Addition in der Funktion „addWenigerToKorb“

„korbEinzel.php“:

```
10      <div>Menge: <?= $korbVonKundeAnzeige['anzahl'] ?>
11      <a href="index.php/cart/addEins/<?= $korbVonKundeAnzeige['id']?>"
12         class="btn btn-success col-3">+ 1</a>
13      <a  href="index.php/cart/addWeniger/<?= $korbVonKundeAnzeige['id']?>"
14         class="btn btn-warning col-3">- 1</a>
15      </div>
16  </div>
```

„routes.php“: Kopiere den Bereich von oben „addEins“, füge alles darunter ein und änderen nur auf „addWeniger“ und auch den Funktionsnamen weiter unten auf „addWenigerToKorb“:

```
35 if(strpos($route,'/cart/addWeniger/') !== false) {
36      $routeTeile = explode("/",$route);
37      $productId = (int)$routeTeile[3];
38
39      addWenigerToKorb($userId,$productId);
40      header("Location: /shop/index.php/warenkorb");
41      exit();
42  }
```

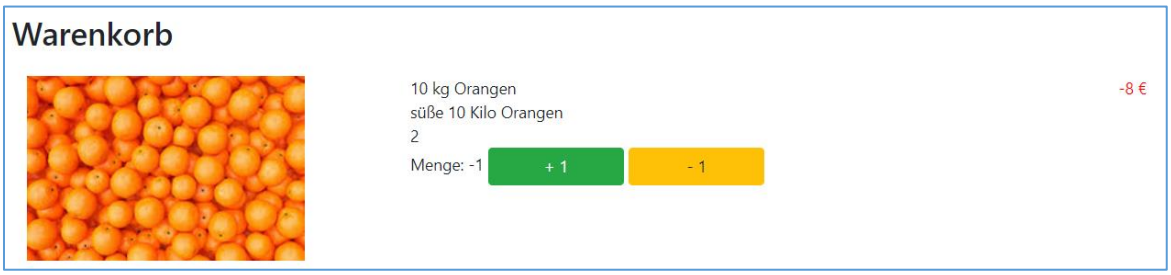
„korb.php“: Kopiere die Funktion von oben „addEinsToKorb“, füge alles darunter ein und änderen nur auf „addWenigerToKorb“ und ändere das PLUS zu MINUS.

```php
function addWenigerToKorb(int $userId, int $productId){
    $sql = "INSERT INTO korb SET
        anzahl=1,
        user_id = :userId,
        product_id = :productId
        ON DUPLICATE KEY UPDATE anzahl = anzahl -1";
    $statement = getDB()->prepare($sql);

    $statement->execute([
        ':userId'=> $userId,
        ':productId'=> $productId
    ]);
}
```

Testen:

Leider kann man noch ins Minus kommen



Das muss man noch ausschließen.