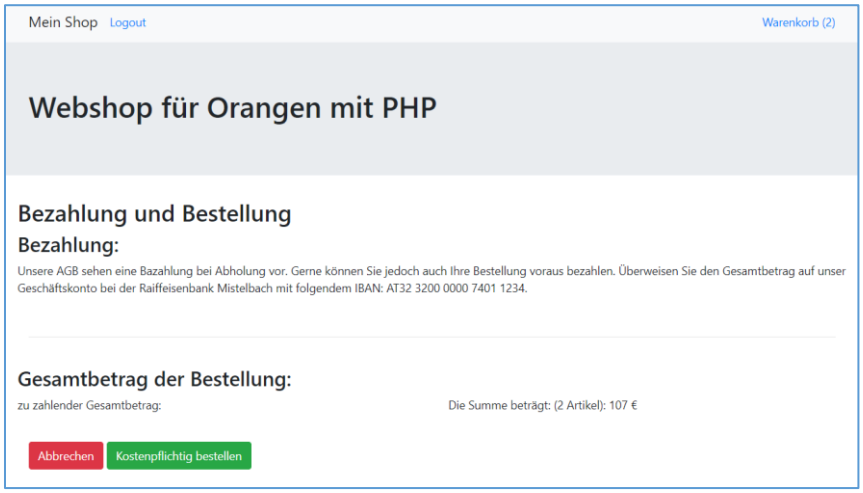
7)Webshop – Bezahlungsseite

Ziel:



1)bezahlung.php – Seite erstellen

Erstelle im Ordner „templates“ eine neue Datei mit dem Namen „bezahlung.php“. Kopiere dort hinein den Inhalt aus „main.php“. Entferne alles innerhalb vom Container und schreibe einfachen HTML Code mit Bootstrap:


```
<div class="row">
    <h2>Bezahlung und Bestellung</h2>
  </div>
  <div class="row">
    <h3>Bezahlung:</h3>
    <p>Unsere AGB sehen eine Barzahlung bei Abholung vor. Gerne können Sie jedoch auch Ihre Bestellung voraus bezahlen. Überweisen Sie den Gesamtbetrag auf unser Geschäftskonto bei der Raiffeisenbank Mistelbach mit folgendem IBAN: AT32 3200 0000 7401 1234.</p>
  </div>
  <br>
  <hr>
```

```
<br>
<div class="row">
  <h3>Gesamtbetrag der Bestellung:</h3>
</div>
<div class="row">
  <p>zu zahlender Gesamtbetrag:</p>
  <div class="col-4"></div>
  <div class="col-4">
  </div>
</div>
```

Für die Anzahl der Artikel und die Gesamtsumme in Euro muss man nun

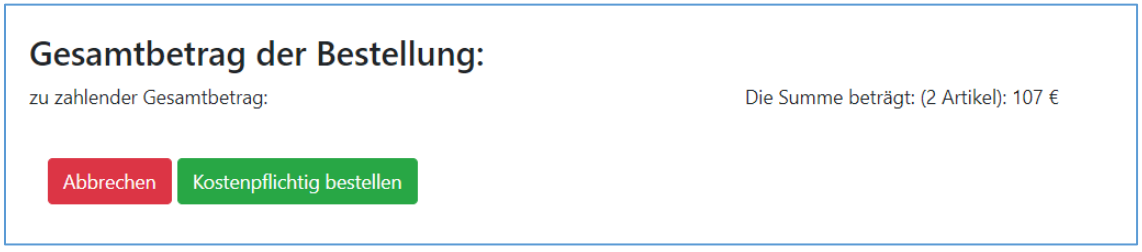
- deren beiden Funktionen suchen und diese anschreiben.
- Erst dann kann man diese Variablen ausgeben.

```
35          <div class="col-3">
36          <?php
37              //aufrufen der Funktion aus korb.php
38              $korbZahl = countProductInKorb($userId);
39              //aufrufen der Funktion aus routes.php
40              $korbSumme = getSummefuerUserId($userId);
41          ?>
42          Die Summe beträgt: (<?= $korbZahl ?> Artikel): <?= $korbSumme ?> €
43          </div>
44      </div>
45      </section>
```

Ergebnis:


```
<?php
        $korbSumme = getSummeFuerUserId($userId);
        $korbZahl = countProductInKorb($userId);
 ?>
        Summe (<?= $korbZahl ?> Artikel): <?= $korbSumme ?> €
```

<u>Nun folgen die Buttons unterhalb</u>




```
44      </div>
45      <br>
46      <a class="btn btn-danger" href="index.php">Abbrechen</a>
47      <a class="btn btn-success" href="index.php/danke">Kostenpflichtig bestellen</a>
48      </section>
```

Abbrechen
Kostenpflichtig bestellen

2)checkout in der Route anpassen

Da im „checkout“ noch nicht die Seite „bezahlung.php“ berücksichtig ist, müssen wir folgendes einfügen – nämlich wenn man bereits eingeloggt ist:

```
97 if(strpos($route,'/checkout') !== false) {
98     if(!isLoggedIn()){
99         $_SESSION['zielEingeloggt'] = "/shop4ck1/index.php/checkout";
100        header("Location: index.php/login");
101        exit();
102    }
103    require __DIR__.'/templates/bezahlung.php';
104    exit();
105 }
```

3)Neue Route anlegen für „Danke“-Seite

Damit man nach dem Klick auf den Button „Kostenpflichtig bestellen“ auf eine neue Seite kommt, muss man eine neue Route dorthin erstellen.

Öffne die „routes.php“ und erstelle (durch Kopie) diese neue Route.

```
135
136 if(strpos($route,'/danke') !== false) {
137     //hier folgt die Logik für die Danke Seite
138     require __DIR__.'/templates/danke.php';
139     exit();
140 }
```

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

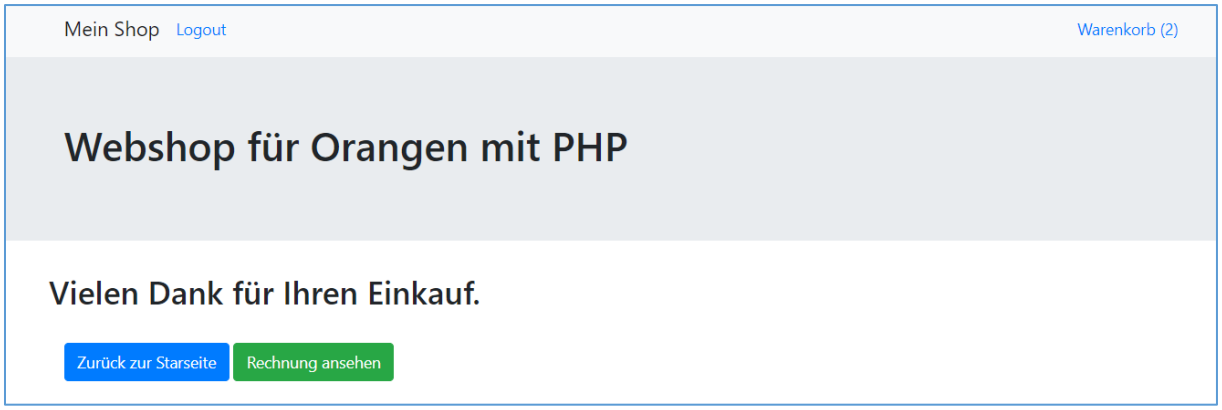
4)Danke Seite

Lege eine neue Seite „danke.php“ im Ordner „templates“ an:

Kopiere dort den Inhalt von „bezahlung.php“ hinein und lösche vieles, damit nur der Header und Footer übrig bleibt. Man benötigt nur den <section>-Bereich mit dem Container und einer Zeile Text:

```
  <body>
    <?php include __DIR__.'/navbar.php' ?>
    <header class="jumbotron" >
    <div class="container" >
    <h1>Webshop für Orangen mit PHP</h1>
    </div>
    </header>
    <section class="container" >
    <div class="row">
        <h2>Vielen Dank für Ihren Einkauf.</h2>
    </div>
    <br>
    <a class="btn btn-primary" href="index.php">Zurück zur Starseite</a>
    <a class="btn btn-success" href="index.php/rechnung">Rechnung ansehen</a>
    </section>
    <script src="assets/js/bootstrap.js"></script>
  </body>
</html>
```

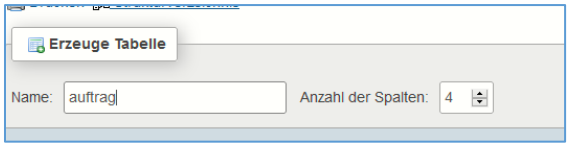
Ergebnis:



5)Datenbank:

5a)Tabelle „auftrag“ anlegen

Datenbank-Tabelle „auftrag“ soll beinhalten:



#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra
1	auftragId	int(11)			Nein	*kein(e)*		AUTO_INCREMENT
2	auftragsdatum	datetime			Nein	current_timestamp()		
3	status	text	utf8mb4_general_ci		Nein	*kein(e)*		
4	userId	int(11)			Nein	*kein(e)*		

- Bei dem Datum muss man „current_timestamp“ auswählen, damit er es auch laufend mitschreibt.

- Datum und Zeit der Anlage – bekommt einen „index“, da man später nach Datum suchen sollen können, z.B. welche Aufträge zu Weihnachten?

- userId – soll genauso heißen, wie die bereits verwendete Variable „userId“
  mit
  - einem eigenem Index, da später danach gesucht werden soll; also vorne mit Haken markieren und dann auf das Symbol „index“

  - dann die Verbindung (Fremdschlüssel) zu einem User (userId), der diese angelegt hat

Klicke auf das Kästchen am Zeilenkopf und „Beziehungsansicht“ und

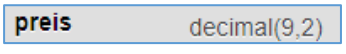
- gib einen Text an, der beide Felder betrifft – FS (für Fremdschlüssel) auftrag_zu_user
- wähle aus dieser Tabelle die „userId“ aus
- Datenbank „shop“
- dann die Referenztabelle „users“
- der Fremdspaltenname die „id“ in „users“
- für „delete“ – no action (wenn ein Auftrag gelöscht wird, soll beim User nichts passieren)
- für „update“ auch no action.

Info: eine Verbindung zu einem user, der die Bestellung ja aufgibt und somit zugeordnet bekommt.

5b)Tabelle „auftrag_produkte“ anlegen

- id_auftrag
- titel – muss wie in „products“ auch varchar 255 sein
- anzahl – muss wie in „products“ auch int(11) haben

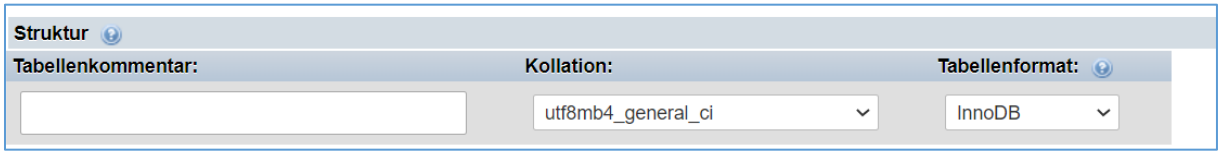
- preis - muss auch wie in „products“ decimal(9,2) haben

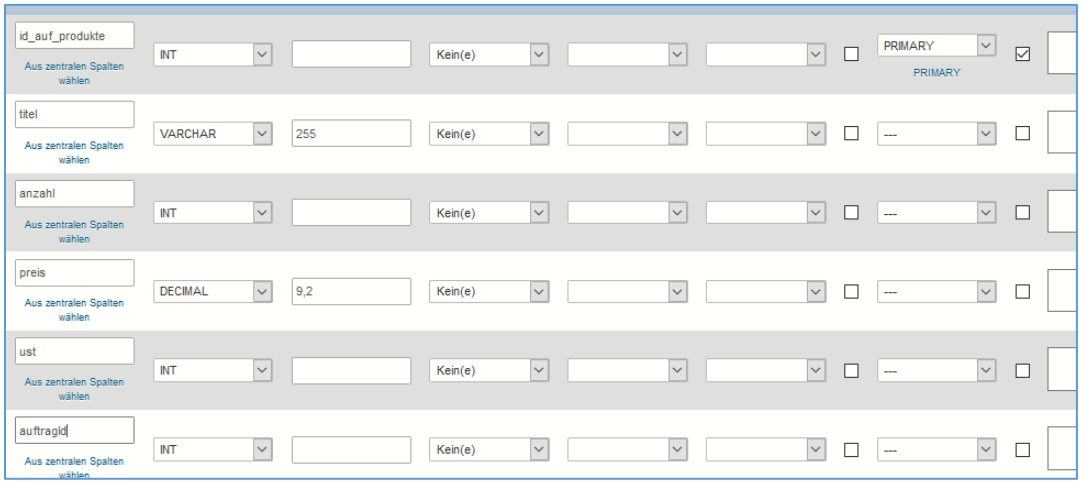


- ust – in Österreich wird hier 20% verwendet werden
- auftragId

Diese Verbindung ist wichtig, wenn man später eine Rechnung erstellt, die ein User haben will.

Mit der Kollation:





Beachte:

Sollen die paar, die sich überschneiden, die gleichen Bezeichnungen haben und Typen, wie in „products“, wobei man die Beschreibung auf einer Rechnung aber nicht benötigt – also

- titel und
- anzahl und
- preis

products:

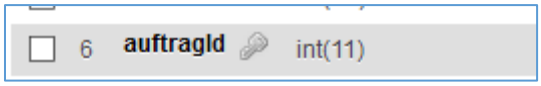
#	Name	Typ	Kollation	Att
1	id	int(11)		
2	titel	varchar(255)	utf8mb4_general_ci	
3	beschreibung	text	utf8mb4_general_ci	
4	preis	decimal(9,2)		
5	anzahl	int(11)		

auftrag_produkte:

#	Name	Typ	Kollation	Att
1	id_auftrag	int(10)		
2	titel	varchar(255)	utf8mb4_general_ci	
3	anzahl	int(11)		
4	preis	decimal(9,2)		
5	ust	int(10)		
6	auftragId	int(11)		

Nun noch eine Beziehung herstellen zwischen der nun angelegten „orderId“ und der ID von „auftrag“.

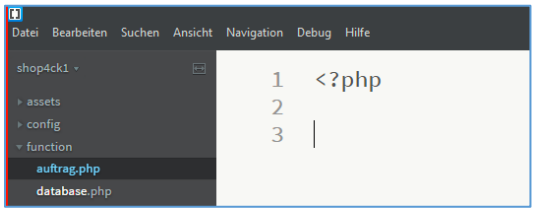
Ergebnis: Schlüssel wird grau dargestellt



Ende 4CK 2 19.3.

6)Datei „auftrag.php“ erstellen mit den Funktionen für den Auftrag

Diese Funktion soll einen Datensatz in der Datenbank anlegen, der einen neuen Auftrag schafft.
Daher in „function“ erstellen:



- Als erstes muss diese neue Datei in „includes.php“ eingebunden werden.

```
5  require_once __DIR__.'/function/korb.php';
6  require_once __DIR__.'/function/product.php';
7  require_once __DIR__.'/function/auftrag.php';
8  require_once __DIR__.'/routes.php';
9
```

- In der „auftrag.php“ erstelle eine neue Funktion namens „createAuftrag()“.
  - Diese ist eine „boolean“, die also ja oder nein liefert.
  - Zwei Parameter:
    - Sie erhält auch Bezug zur $userId, damit man den Auftrag dem eingeloggten User zugeordnet werden kann (in der Datenbank ja auch als Fremdschlüssel vorhanden)
    - Als Parameter wird auch der „Warenkorb“ als Array übergeben – die Variable lautete damals „$korbVonKunde“ und wurde ein „korb.php“ damals festgelegt:

```
33      $korbVonKunde = [];
34      while($row = $results->fetch()){
35          $korbVonKunde[]=$row;
36      }
37      return $korbVonKunde;
38  }
```

Der Status wird noch nicht wirklich verwendet, soll aber von vornherein auf „neu“ gestellt sein.
Daher dies gleich in die Funktion oben einbauen:

```
1  <?php
2
3  function createAuftrag(int $userId, array $korbVonKunde,string
   $status = 'neu'):bool{
4      $sql = "INSERT INTO auftrag SET
```

```
function createAuftrag(int $userId, array $korbVonKunde,string $status = 'neu'):bool{
   $sql = "INSERT INTO auftrag SET
}
```

6sa)Erste Speicherung von Daten mit der ersten SQL:

In der SQL wird nur die „userId“ gesetzt – wieder in der bewährten „prepaired statement“-Variante:

```
4       $sql = "INSERT INTO auftrag SET
5           status = :status,
6           userId = :userId
7           ";
8       $statement = getDB()->prepare($sql);
```

```
 8       $statement = getDB()->prepare($sql);
 9       $data = [
10           ':status'=>$status,
11           ':userId'=>$userId
12       ];
13       $statement->execute($data);
14       }
```

```
$statement = getDB()->prepare($sql);
$data = [
    ':status'=>$status,
    ':userId'=>$userId
];
$statement->execute($data);
```

Danach wird die letzte angelegte ID gesucht:

```
13       $statement->execute($data);
14       $auftragId = getDB()->lastInsertId();
```

```
$auftragId = getDB()->lastInsertId();
```

6b) Zweite SQL

Danach folgt die zweite SQL-Abfrage, um die einzelnen Auftrags-Details hinzu zu fügen.

```
14      $auftragId = getDB()->lastInsertId();
15
16      $sql = "INSERT INTO auftrag_produkte SET
17      titel=:titel,
18      anzahl=:anzahl,
19      preis=:preis,
20      ust=:ust,
21      auftragId=:auftragId
22      ";
23
24      $statement = getDB()->prepare($sql);
25      foreach($korbVonKunde as $korbVonKundeDetail){
26          $data = [
27          ':titel'=>$korbVonKundeDetail['titel'],
28          ':anzahl'=>$korbVonKundeDetail['anzahl'],
29          ':preis'=>$korbVonKundeDetail['preis'],
30          ':ust'=>20,
31          ':auftragId'=>$auftragId, //siehe Zeile 14
32      ];
33      $statement->execute($data);
34      }
```

```
$sql = "INSERT INTO auftrag_produkte SET
   titel=:titel,
   anzahl=:anzahl,
   preis=:preis,
   ust=:ust,
   auftragId=:auftragId
   ";

   $statement = getDB()->prepare($sql);
   foreach($korbVonKunde as $korbVonKundeDetail){
      $data = [
      ':titel'=>$korbVonKundeDetail['titel'],
      ':anzahl'=>$korbVonKundeDetail['anzahl'],
      ':preis'=>$korbVonKundeDetail['preis'],
      ':ust'=>20,
      ':auftragId'=>$auftragId, //siehe Zeile 14
   ];
   $statement->execute($data);
}
```

Und zum Schluss noch ein „true“, weil in der Zeile 3, beim Beginn der Funktion, ein „bool“, also ein true oder false verlangt wird:

```
33      $statement->execute($data);
34      }
35      return true; //weil oben ein bool verlangt wurde
36  }
37
```

return true; //weil oben ein bool verlangt wurde

7)Die Funktionen „createAuftrag“ und „clearWarenkorb“ ausführen

Klickt man auf „Kostenpflichtig bestellen“ soll der neue Auftrag nun in die Datenbank gesendet werden und dort gespeichert werden.

Daher öffne die Datei „routes.php“ und bearbeite die Route „danke“.

Hier hinein kommt jetzt die gerade angelegte Funktion „createAuftrag()“.

- Zuerst wird die Funktion geschrieben, davor aber die aktuelle „userId“ geholt
- Die Elemente des Warenkorbs muss man sich von früher, nämlich aus der Zeile 27 dieses Skripts, kopieren.

```
25
26 if(strpos($route,'/warenkorb') !== false) {
27     $korbVonKunde = getKorbVonKunde($userId);
28     $korbSumme = getSummefuerUserId($userId);
29     require __DIR__.'/templates/korbSeite.php';
30     exit();
```

Daher hat man dann beide Parameter für die „createAuftrag“-Funktion und kann diese einfügen.

```
136 if(strpos($route,'/danke') !== false) {
137
138     $userId = getCurrentUserId();
139     $korbVonKunde = getKorbVonKunde($userId);
140
141     createAuftrag($userId,$korbVonKunde);
```

$userId = getCurrentUserId();
$korbVonKunde = getKorbVonKunde($userId);

createAuftrag($userId,$korbVonKunde);

clearWarenkorb()

- Sinnvollerweise sollte hier auch der Warenkorb des Users gelöscht werden, da er ja schon kostenpflichtig bestellt hat. Somit wird der Warenkorb rechts oben wieder auf Null gestellt, wenn er z.B. bald wieder einkaufen will, soll er von null beginnen.
  Der Zeitpunkt ist hier ganz gut, weil die folgende Rechnungsseite müsste er ja nicht mehr unbedingt aufrufen und somit ist hier schon gesichert, dass der Warenkorb leer ist ab jetzt.

  Der Auftrag bleibt ja in der Datenbank bestehen, ist ja nicht direkt abhängig vom Warenkorb.

Die „createAuftrag“ kommt aber in eine kleine IF-Abfrage, nämlich

- Wenn der Auftrag angelegt wurde, dann soll der Warenkorb geleert werden und eine Dankesseite angezeigt werden:

```
159     if(createAuftrag($userId,$korbVonKunde)){
160         clearWarenkorb($userId);
161         require __DIR__.'/templates/danke.php';
162     }
```

```
if(createAuftrag($userId,$korbVonKunde)){
    clearWarenkorb($userId);
    require __DIR__.'/templates/danke.php';
}
```

8)Leeren des Warenkorbs - „clearWarenkorb()“-Funktion

Die neue Funktion „clearWarenkorb“ wird noch erstellt und zwar sinnvollerweise in der Datei „korb.php“, weil es ja den Warenkorb betrifft. Darin wird für den betreffenden User der Warenkorb geleert.

Öffne „korb.php“.

Diese Funktion erhält als Parameter die Variable userId und erstellt eine SQL-Funktion mit DELETE:

```
54 function clearWarenkorb(int $userId){
55     $sql = "DELETE FROM korb WHERE user_id = :userId";
56     $statement = getDB()->prepare($sql);
57     $statement->execute([
58         ':userId' => $userId
59     ]);
60 }
```

```
function clearWarenkorb(int $userId){
    $sql = "DELETE FROM korb WHERE user_id = :userId";
    $statement = getDB()->prepare($sql);
    $statement->execute([
        ':userId' => $userId
    ]);
}
```

Ergebnis testen:

Wird nun ein neuer Auftrag erstellt wird in der Tabelle „auftrag_produkte“ zu einer AuftragID eine oder eine Mehrzahl von Produkten zugeordnet. Somit kann man später zu dieser AuftragId alle Produkte abfragen.

Optionen

	id_auftrag	titel	anzahl	preis	ust	auftragId
Bearbeiten Kopieren Löschen	42	5kg Orangen	1	5.00	20	1
Bearbeiten Kopieren Löschen	43	10 kg Orangen	1	8.00	20	1

In der Tabelle „auftrag“ steht diese eine „auftragId“ natürlich drinnen und ist einem User zugeordnet. Das ermöglicht die perfekte Abfrage später bei der Rechnung, da man diese beiden kombinieren muss, ein User und die dazupassende „auftragId“.

Optionen

	auftragId	auftragsdatum	status	userId
Bearbeiten Kopieren Löschen	1	0000-00-00 00:00:00	neu	11

9)Die Route für die Rechnung anlegen

Erstelle folgende Route, die auf das noch nicht erstellte Template verweisen wird. Die Route kommt aus dem Button in der „danke.php“ – Zur Rechnung.

```
147
148 if(strpos($route,'/rechnung') !== false) {
149     //hier folgt die Logik für die Danke Seite
150     require __DIR__.'/templates/rechnung.php';
151     exit();
152 }
153
```

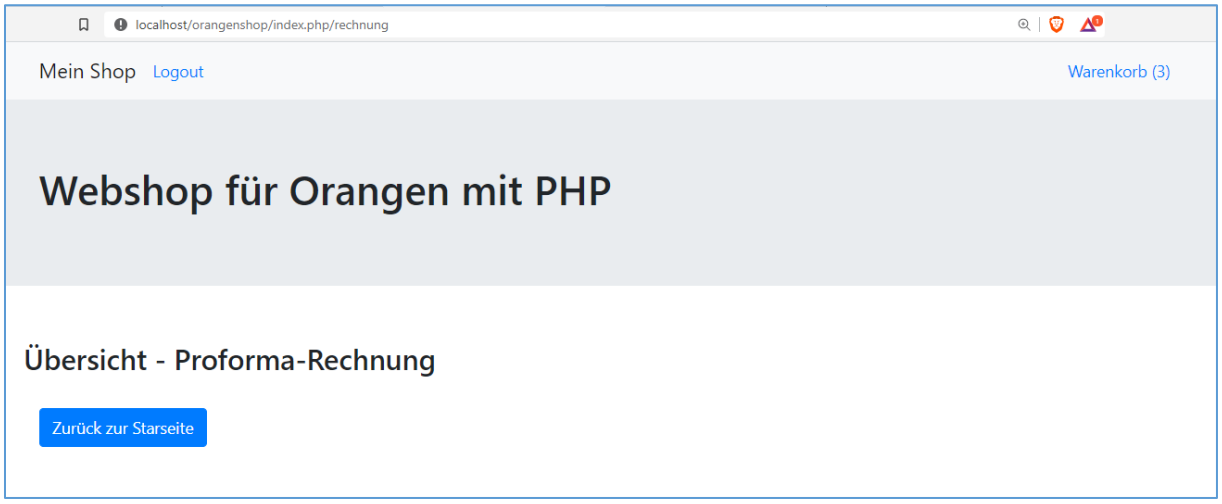
10)Die Seite „rechnung.php“ anlegen

Dies geschieht nun wieder im Ordner „templates“ indem man die „bezahlung.php“ kopiert und den Bereich innerhalb des Containers löscht oder umschreibt:

```
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     <br>
20     <div class="row">
21         <h3>Übersicht - Proforma-Rechnung</h3>
22     </div>
23     <br>
24     <a class="btn btn-primary" href="index.php">Zurück zur Starseite</a>
25     </section>
26     <script src="assets/js/bootstrap.js"></script>
27   </body>
28 </html>
```

```
<section class="container" id="produkte" >
  <br>
  <div class="row">
    <h3>Übersicht - Proforma-Rechnung</h3>
  </div>
  <br>
  <a class="btn btn-primary" href="index.php">Zurück zur Startseite</a>
  </section>
```

Ergebnis:



Quelle:
https://www.youtube.com/watch?v=Cvy4eJ1TQYQ&list=PLz858EFEcxiElOUr7b3Pql1CMHuVRYkTh&index=20

https://www.youtube.com/watch?v=qsZJ4xP85Us&list=PLz858EFEcxiElOUr7b3Pql1CMHuVRYkTh&index=21