8)Ausgabe einer Rechnung

1)In „routes.php“ die Route und die Logik anlegen:

Die Rechnung soll nach der ID des Auftrags erstellt werden.

In der Datenbank sieht es z.B. gerade so aus:

	id_auftrag	auftragsdatum	userId	titel	anzahl	preis	ust
Löschen	1	2020-08-21 16:00:59	10	5kg Orangen	0	5.00	20
Löschen	2	2020-08-21 16:00:59	10	10 kg Orangen	0	8.00	20
Löschen	5	2020-08-21 16:16:02	8	10 kg Orangen	0	8.00	20
Löschen	8	2020-08-21 17:04:48	2	5kg Orangen	0	5.00	20

In „routes.php“ wurde bereits eine Route erstellt. Dort wird nun die Logik codiert.

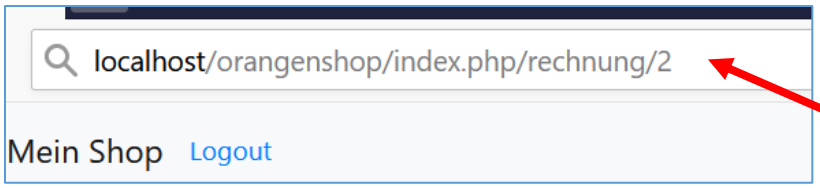
- Zuerst wird geprüft, ob man überhaupt eingeloggt ist, damit man nicht auf gut Glück dazu kommen kann. Wenn nicht, wird man zur „Login“-Seite gesendet.

```
166 if(strpos($route,'/rechnung') !== false) {
167     if(false === isLoggedIn()){
168         header("Location: /orangenshop/index.php/login");
169         exit();
170      }
171     require __DIR__.'/templates/rechnung.php';
172     exit();
173 }
```

```
if(strpos($route,'/rechnung') !== false) {
   if(false === isLoggedIn()){
      header("Location: /orangenshop/index.php/login");
      exit();
    }
   require __DIR__.'/templates/rechnung.php';
   exit();
}
```

Funktion „explode“ verwenden:

Die Rechnungsnummer soll aus der URL ausgelesen werden. Dafür wird die Route aufgeteilt in Einzelteile und zwar mit „explode“. Zum jetzigen Zeitpunkt haben wir zwar noch keine Rechnungsnummer in der URL, da die dazu passende „SQL-SELECT-Abfrage“ erst später in der „auftrag.php“ geschrieben wird, aber die Vorbereitungen laufen hier bereits für das Auslesen der Id aus der URL:



den 2 gibt es noch nicht, nur händisch hin geschrieben, um das zu zeigen ☺

- Dafür wird zuerst eine Variable „routeParts“ für die einzelnen Teile definiert.
- Die wird gleichgesetzt mit der Funktion „explode“ zum Aufteilen des Textes (string) in der URL. (Die URL besteht immer als Text.)
  Parameter: teile nach dem Slash und nimm die URL (=die Route)

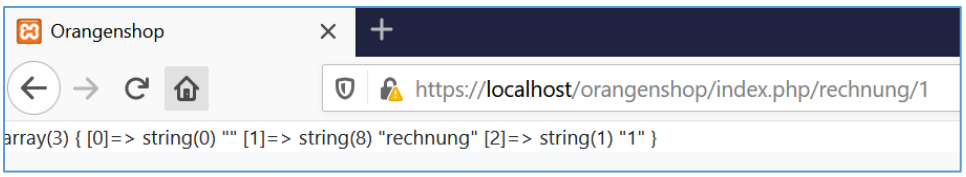
```
169             exit();
170         }
171         $routeParts = explode('/', $route);
172         
173         require __DIR__.'/templates/rechnung.php';
```

$routeParts = explode('/', $route);

Zur Überprüfung kann man die Route ausgeben und sehen, welche Positionen vorhanden sind. Dafür verwende ein „var_dump()“:
Man muss jedoch eine Nummer eingeben, damit man es sieht:

```
170         $routeParts = explode('/', $route);
171
172         var_dump($routeParts);
```

Ergebnis:



d.h. bei einer Nummer wird diese an 2 Position ausgelesen.

Lösche das var_dump wieder.

- Danach wird die Variable $rechnungId als leerer Wert angelegt.

```
171         $routeParts = explode('/', $route);
172         $rechnungId = null;
```

$rechnungId = null;

- Dann wird die Id des Orders aus der URL ausgelesen, und zwar mit einer IF isset, ob der Array-Key mit der Nummer 2 auch gesetzt ist:

```
170     $routeParts = explode('/', $route);
171     $rechnungId = null;
172     if(isset($routeParts[2])){
173     }
```

- Wenn das der Fall ist, soll die $rechnungId diesen neuen Wert aus dem Array erhalten: aber es soll als Integer verwendet werden, nicht als string, wie aus der URL entnommen.

```
172     if(isset($routeParts[2])){
173         $rechnungId = (int)$routeParts[2];
174     }
```

```
if(isset($routeParts[2])){
    $rechnungId = (int)$routeParts[2];
}
```

BEACHTE: keine Anführungszeichen in diesem Fall in den eckigen Klammern!!!

- Damit man nicht von außen einfach eine URL erfindet und abzufragen versucht, sollte man das mit einer Fehlermeldung (IF-Abfrage) berücksichtigen und einen Text ausgeben lassen, wie z.B. Rechnung bitte korrekt abfragen.

```
175     }
176     if(!$rechnungId){
177         echo "Bitte Rechnung korrekt abfragen.";
178         exit();
179     }
```

```
if(!$rechnungId){
    echo "Bitte Rechnung korrekt abfragen.";
    exit();
}
```

- Nun wird endlich eine Kombination zwischen Rechnungsnummer und User gesucht, um die richtige Rechnung zu finden. Dazu wird euch eine eigene Funktion im Ordner „functions“ erstellt werden, nämlich in der „auftrag.php“, die bald folgen wird.
  Zuerst wird aber wieder die aktuelle UserId geholt:
-

```
179     }
180     $userId = getCurrentUserId();
```

```
$userId = getCurrentUserId();
```

- Dann wird die Variabel „orderData“ geladen und mit der Funktion, die bald erstellt werden wird, nämlich „getAuftragForUser“ mit den Parametern der „rechnungId“ und der „userId“:

```
180       $userId = getCurrentUserId();
181
182       $orderData = getAuftragForUser($rechnungId,$userId);
```

$orderData = getAuftragForUser($rechnungId,$userId);

Da aber die Variabel „orderData“ auch null sein könnte, wie von uns auch definiert wurde, soll eine IF das abfragen und eine Meldung bereitstellen:

```
182       $orderData = getAuftragForUser($rechnungId,$userId);
183
184       if(!$orderData){
185           echo "Daten wurden nicht gefunden.";
186           exit();
187       }
```

if(!$orderData){
    echo "Daten wurden nicht gefunden.";
    exit();
  }

2)Funktion in „auftrag.php“ erstellen – getAuftragForUser()

Diese Funktion soll 2 Parameter haben, nämlich $rechnungId und $userId, wobei ein „optionales array“ zurückgegeben werden soll – optional bedeutet, dass später die beiden „return“ nicht ein leeres array haben (return []) sondern eine „null“ (return null).

```
38
39  function getAuftragForUser(int $rechnungId,int $userId):?array{
```

function getAuftragForUser(int $ rechnungId,int $userId):?array{}

Dann sollen die Aufträge ausgegeben werden, bei denen der betreffende User gemeint ist, der gerade angemeldet ist und der betreffende Auftrag und das aus der Tabelle „auftrag“:

#	Name	Typ	Kollation	Attr
1	auftragId	int(11)		
2	auftragsdatum	datetime		
3	status	text	utf8mb4_general_ci	
4	userId	int(11)		

Dann kommt die SQL-Query mit der entsprechenden SELECT-Abfrage: mit einem LIMIT 1, damit maximal ein Datensatz ausgegeben werden kann.

```
41      $sql = "SELECT auftragId,auftragsdatum,status,userId
42      FROM auftrag
43      WHERE auftragId = :rechnungId AND userId = :userId
44      LIMIT 1
45      ";
```

```
$sql = "SELECT auftragId,auftragsdatum,status,userId
  FROM auftrag
  WHERE auftragId = :rechnungId AND userId = :userId
  LIMIT 1
  ";
```

Darunter natürlich die Vorbereitung mit „prepare“ als übliches Element von prepared statements.

Darunter eine Fehlermeldung, wenn das Ergebnis fehlerhaft wäre, nun mit einer speziellen Fehlermeldungs-Funktion im echo: die „printDBErrorMessage()“ hilft sehr, wenn in der Datenbank eine Fehlermeldung gut angezeigt werden soll, damit man den Fehler besser findet.

```
46
47      $statement = getDB()->prepare($sql);
48      if(false === $statement){
49          echo printDBErrorMessage();
50          return null;
51      }
```

```
$statement = getDB()->prepare($sql);
  if(false === $statement){
    echo printDBErrorMessage();
    return null;
  }
```

Im nächsten Schritt wird das statement ausgeführt, mit zwei Parameter:

```
38      $statement->execute([
39          ':rechnungId'=>$rechnungId,
40          ':userId'=>$userId
41      ]);
```

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

Zuerst wird überprüft, ob überhaupt ein Datensatz gefunden wird: wenn keine gefunden werden, bricht man ab mit „return null“

```
56
57      if(0 === $statement->rowCount()){
58          return null;
59      }
```

```
if(0 === $statement->rowCount()){
    return null;
}
```

Nun wird der Datensatz geholt und gleich darunter ein Array mit den Produkten.

```
59
60      $orderData =$statement->fetch();
61      $orderData['products'] = [];
62
63      return $orderData;
```

```
$orderData =$statement->fetch();
  $orderData['products'] = [];

  return $orderData;
```

Vor dem return kommt nun die zweite Abfrage, die die Produkte aus der betreffenden Tabelle holt, wobei die „auftragId“ in der WHERE gefragt ist:

id_auftrag	titel	anzahl	preis	ust	auftragId
42	5kg Orangen	1	5.00	20	1

```
64      $sql = "SELECT id_auftrag,titel,anzahl,preis,ust
65      FROM auftrag_produkte
66      WHERE auftragId = :rechnungId
67      ";
```

```
$sql = "SELECT id_auftrag,titel,anzahl,preis,ust
  FROM auftrag_produkte
  WHERE auftragId = :rechnungId
  ";
```

Darunter muss man natürlich die Elemente für das prepared statement ebenfalls ausführen. Dazu kann man es von der ersten Abfrage kopieren und nur leicht verändern, da man nur eine Variable benötigt.

```
69      $statement = getDB()->prepare($sql);
70      if(false === $statement){
71          echo printDBErrorMessage();
72          return null;
73      }
74       $statement->execute([
75          ':rechnungId'=>$rechnungId
76       ]);
77
78       if(0 === $statement->rowCount()){
79          return null;
80      }
```

Nun können die einzelnen Produkte geholt werden und die „orderData[‘products‘] befüllt werden und zwar mit einer WHILE-Schleife werden die Daten geholt und die Products mit den Datensatz

erweitert ($row). Und anschließend mit dem schon vorhandenen return ausgegeben:

```
79      }
80
81      while($row = $statement->fetch()){
82          $orderData['products'][]=$row;
83      }
84
85      return $orderData;
86  }
```

```
while($row = $statement->fetch()){
    $orderData['products'][]=$row;
}
```

3)Die ID der Rechnung in die URL übergeben

Damit die Rechnung aufgerufen werden kann muss die passende NR hinten angefügt sein, damit sie in der route.php auch ausgelesen werden kann:
Dafür wird eine Session angelegt, die die „auftragId“ beinhaltet. Das kann gleich direkt unter der Erstellung dieser AuftragId passieren, weil die Session dann eh länger gilt.

Öffne die „auftag.php“.

```
$_SESSION['auftragId'] = $auftragId;
```

Man holt sich in eine SESSION die nötige „auftragId“ in der Datei „auftrag.php“

```
14      $auftragId = getDB()->lastInsertId();
15      $_SESSION['auftragId'] = $auftragId;
16
17      $sql = "INSERT INTO auftrag_produkte SET
```

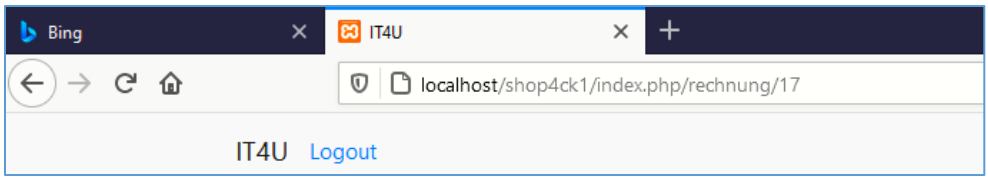
Öffne die „danke.php“

In der „danke.php“ wird sie dann angehängt an den Button (die URL) diese betreffende Session mit einem Slash nach der Route „rechnung“:

/<?= $_SESSION['auftragId'] ?>

```
24      <a class="btn btn-primary" href="index.php">Zurück zur Starseite</a>
25      <a class="btn btn-success" href="index.php/rechnung/<?=
        $_SESSION['auftragId']?>">Rechnung ansehen</a>
26      </section>
```

Ergebnis:
Nun wird die passende Rechnungsnummer automatisch aufgerufen und sofort die Rechnung angezeigt:



4)die Datei „rechnung.php“ in templates für die Anzeige der Rechnung herrichten

Öffne die Datei „rechnung.php“ im Ordner „templates“. Der PHP-Code soll unter der Überschrift ausgegeben werden.

Hier soll eine „foreach“-Schleife angelegt werden, die die Variable „$orderData“ aus der Funktions-Datei „auftrag.php“ holt: Erstelle aber auch gleich das Ende der „foreach“.

```
21          <h3>Übersicht - Proforma-Rechnung</h3>
22
23          <?php foreach($orderData['products'] as $order):?>
24          <?php endforeach; ?>
25      </div>
```

<?php foreach($orderData['products'] as $order):?>
<?php endforeach; ?>

Dazwischen werden die Daten ausgegeben:

```
22
23          <?php foreach($orderData['products'] as $order):?>
24              <?=$order['titel']?>
25              <?=$order['anzahl']?>
26              <?=$order['preis']?>
27              <?=$order['ust']?>
28          <?php endforeach; ?>
29      </div>
```

<?=$order['titel']?>
<?=$order['anzahl']?>
<?=$order['preis']?>
<?=$order['ust']?>

Ergebnis und Testen:

Führt man eine Bestellung durch, kann man dann die betreffende Pfad-Nummer in die URL eingeben, hier die 2 und sich somit die Rechnung ansehen:

Die 2 sieht man hier in der Tabelle:

Ausgabe:

Das Ergebnis ist noch nicht schön aber es kommt.

Elemente einbauen

Die Rechnung soll, wie in Bootstrap üblich, in Zeilen und Spalten aufgeteilt werden. Dafür werden auch mehrere „section“ erstellt werden.

Öffne die „rechnung.php“.

Am besten man erstellt am Beginn den Container mit einer eigenen Klasse, nämlich „pdf-container“ und macht aus dieser section eine :

Darunter nun die erste section für das Logo.

```
11  <body>
12    <?php include __DIR__.'/navbar.php' ?>
13    <header class="jumbotron" >
14    <div class="container" >
15     <div class="row">
16        <h3>Übersicht - Proforma-Rechnung</h3>
17    </div>
18    </div>
19    </header>
20    <div class="pdf-container" >
21    <section class="row" id="companyLogo">
22    </section>
23    <section class="row" id="companyDetails">
24    </section>
```

```
25      <section class="row" id="empfaenger">
26      </section>
27      <section class="row" id="rechnungsdatum">
28      </section>
29      <section class="row" id="rechnungsnummer">
30      </section>
31      <section class="row" id="rechnungsheader">
32      </section>
```

```
33      <section id="produkte">
34      <!--hier folgen die Produkte-->
35          <?php foreach($orderData['products'] as $order):?>
36          <div>
37              <?=$order['titel']?>
38              <?=$order['anzahl']?>
39              <?=$order['preis']?>
40              <?=$order['ust']?>
41          </div>
42          <?php endforeach; ?>
43      </section>
```

```
44      <section class="row" id="summe">
45      </section>
46      <section class="row" id="rechnungsfooter">
47      </section>
48
49      <br>
50      <br>
51      <a class="btn btn-primary" href="index.php">Zurück zur Starseite</a>
52      </div> <!--ende container-->
53      <script src="assets/js/bootstrap.js"></script>
54    </body>
55  </html>
```

4a)Daten eintragen

Firmendaten:

Die einzelnen Daten für die Firma werden nicht in die Datenbank gelegt, sondern hier nur direkt eingetragen.

```
20      <div class="container" >
21      <section class="row" id="companyLogo">
22      </section>
23      <section class="row" id="companyDetails">
24          <h3>IT4U GmbH, Brennerweg 8 2130 Mistelbach, Tel.: 02572/2305</h3>
25      </section>
```

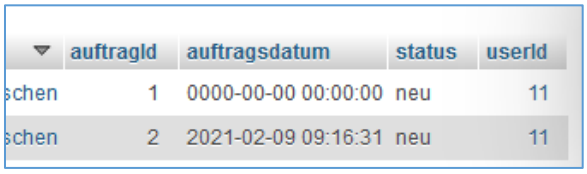
Empfänger:

Da kein Formular vorhanden ist, wo der Kunde seine Adresse eintragen kann, wird dies nur angedeutet und nur „Empfänger“ angeschrieben. Dabei wird aber nach oben ein Innenabstand (padding) verwendet, damit es schöner aussieht und nicht an der Kopfzeile klebt:

```
26      <section class="row" id="empfaenger">
27          <h4 style="padding-top:60px">Empfänger</h4>
28      </section>
```

Rechnungsdatum:

aus der Tabelle „auftrag“:



	auftragId	auftragsdatum	status	userId
schen	1	0000-00-00 00:00:00	neu	11
schen	2	2021-02-09 09:16:31	neu	11

In der Datei „auftrag.php“ findet man bereits dessen Ausgabe aus der Datenbank, siehe Zeile 41

```
39 function getAuftragForUser(int $rechnungId,int $userId):?array{
40
41       $sql = "SELECT auftragId,auftragsdatum,status,userId
42       FROM auftrag
43       WHERE auftragId = :rechnungId AND userId = :userId
44       LIMIT 1
45       ";
```

Leider ist das Format, wie oben in der DB sichtbar, nicht ganz ideal für die Rechnungsadresse. Man benötigt ja nicht die Uhrzeit.

Da in der Variable $orderData diese Daten ausgelesen werden, sollte man hier auch diese Umwandlung durchführen.

Öffne daher die „auftrag.php“ und erstelle unter der Erzeugung dieser Variable diesen Code:

```
61       $orderData =$statement->fetch();
62       $orderData['auftragsdatumFormatiert'] =
63
64       $orderData['products'] = [];
```

$orderData['auftragsdatumFormatiert'] =

Nun muss man ein Objekt für das Datum erzeugen, um es dann formatieren zu können. Dafür erstelle eine neue Variable davor, welche die NEUE Funktion „date_create()“ hat, wobei diese auf den Inhalt des Feldes in der Datenbank zugreift:

```
61       $orderData =$statement->fetch();
62       $neuesDatum = date_create($orderData['auftragsdatum']);
63       $orderData['auftragsdatumFormatiert'] =
```

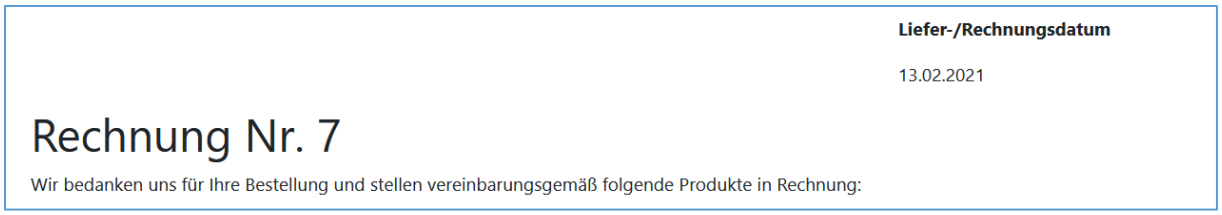
$neuesDatum = date_create($orderData['auftragsdatum']);

Nach dem Istgleichzeichen von vorher wird nun die Funktion „date_format()“ zum Umformatieren angewendet: die Attributen sind zuerst die erstellte Variable und dahinter das gewünschte Format:

```
$orderData['auftragsdatumFormatiert'] = date_format($neuesDatum,'d.m.Y');
```

Nun kann man dies in der „rechnung.php“ ausgeben:

Dabei soll es ca. so aussehen: nur eine Spalte ganz rechts, wobei die anderen davor leer sind. Das Datum dann in der rechten Spalte. Dies soll mit Bootstrap umgesetzt werden: wobei wir aber Lieferdatum und Rechnungsdatum in eine Spalte zusammen geben, da das bei uns immer gleich ist



Umsetzung in Bootstrap: dabei wird mit einem „offset“ gearbeitet.

Und darunter das neu formatierte Datum, wobei die obere Zeile fett (strong) geschrieben sein soll:

```
29      <section class="row" id="rechnungsdatum">
30          <div class="col-4 offset-8">
31              <p><strong>Liefer-/Rechnungsdatum</strong></p>
32              <p><?=$orderData['auftragsdatumFormatiert']?></p>
33          </div>
```

Quelle: Part 26: https://www.youtube.com/watch?v=ppL8RpkEIN0

Als nächstes:

Rechnungsnummer, Ausgabe der Produkte und Summe inkl. USt

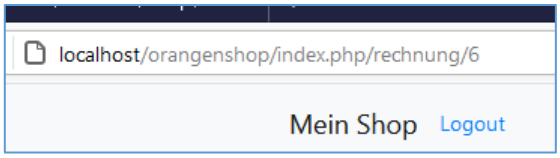
Quelle: NR 27 https://www.youtube.com/watch?v=5xsBC6ZRN_c

Rechnungsnummer:

```
34      <section class="row" id="rechnungsnummer">
35          <h1 class="col-12"></h1>
36      </section>
```

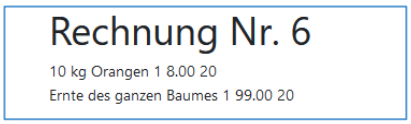
Darin nun der Text und die Nummer aus der Datenbank, nämlich die „auftragId“ aus der Tabelle „auftrag“. Eigentlich die gleiche wie oben in der URL:

auftragId	auftragsdatum	status	userId
1	0000-00-00 00:00:00	neu	11
2	2021-02-09 09:16:31	neu	11



```
34 ▼    <section class="row" id="rechnungsnummer">
35          <h1 class="col-12">Rechnung Nr. <?=$orderData['auftragId']?></h1>
36      </section>
```

Ergebnis:



„Wir bedanken uns“:

```
37 ▼    <section class="row" id="rechnungsheader">
38          <p class="col-12">Wir bedanken uns für Ihre Bestellung und stellen
            vereinbarungsgemäß folgende Produkte in Rechnung:</p>
39      </section>
```

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

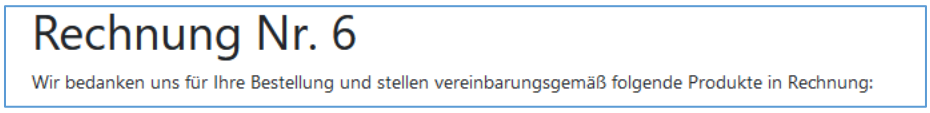
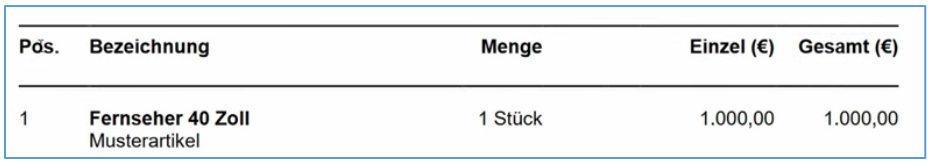


Tabelle mit den Produkten:

Ziel: 5 Spalten



Pos.	Bezeichnung	Menge	Einzel (€)	Gesamt (€)
1	**Fernseher 40 Zoll** Musterartikel	1 Stück	1.000,00	1.000,00

Daher wird eine Tabelle eingebaut, inkl. head, body und footer:

```
40 ▼    <section id="produkte">
41      <!--hier folgen die Produkte-->
42 ▼        <table>
43          <thead></thead>
44 ▼        <tbody>
45          <?php foreach($orderData['products'] as $order):?>
```

```
52          <?php endforeach; ?>
53          </tbody>
54          <tfoot></tfoot>
55          </table>
56      </section>
```

Im Head dann die Zeile mit den 5 Überschriften:

```
43 ▼        <thead>
44 ▼            <tr>
45                  <th>Pos.</th>
46                  <th>Bezeichnung</th>
47                  <th>Menge</th>
48                  <th>Einzel</th>
49                  <th>Gesamt</th>
50              </tr>
51          </thead>
```

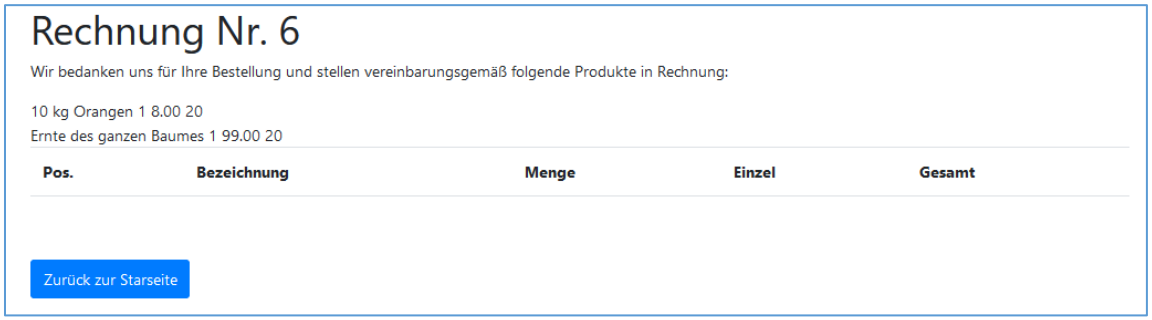
<tr>
    <th>Pos.</th>
    <th>Bezeichnung</th>

```
    <th>Menge</th>
    <th>Einzel</th>
    <th>Gesamt</th>
  </tr>
```

Zur schönen Aufteilung der Elemente benutze die class="table" aus Bootstrap:

```
41      <!--hier folgen die Produkte-->
42          <table class="table">
43          <thead>
```

Ergebnis:



Nun muss man die Produkte in die Tabelle hinunter bringen. Dafür muss man das entfernen und dafür eine <tr> verwenden. Die einzelnen Elemente der Produkte dann in Zellen, und daher in „td".

```
53          <?php foreach($orderData['products'] as $order):?>
54          <tr>
55              <td> <?=$order['titel']?></td>
56              <td> <?=$order['anzahl']?> </td>
57              <td> <?=$order['preis']?> </td>
58              <td> <?=$order['ust']?> </td>
59          </tr>
60          <?php endforeach; ?>
```

Aber es stimmen die Daten nicht mit der Überschrift überein, z.B. Bezeichnung usw. Grund ist die erste Spalte, wo „Pos." steht. Dort sollte eigentlich aufsteigend ab 1 eine Zahl stehen. Wenn hier eine Zahl eingefügt wird, dann rückt alles nach rechts und es passt. Diese benötigte Zahl haben wir aber nicht in der Datenbank.

Daher soll diese aufsteigende Zahl nun hier erstellt werden, nämlich aus der „foreach" heraus:

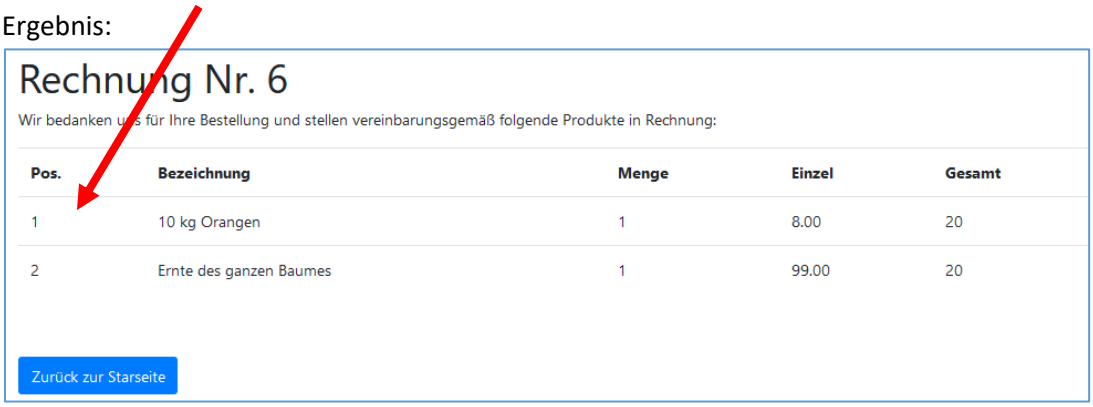
Es wird hier nach dem „as“ ein index eingefügt, bei dem jedes Element heruntergezählt wird, beginnend aber bei „null“.

```
<?php foreach($orderData['products'] as $index=> $order):?>
<tr>
```

Dafür wird nun auch eine eigene Zelle erstellt, die aber auch gleich den Wert um eins erhöht, damit er nicht bei Null beginnt, sondern bei eins. Dafür muss man „inkrement“ zwei Plus davor stellen:

```
54        <tr>
55            <td> <?= ++$index?> </td>
56            <td> <?=$order['titel']?></td>
```

Ergebnis:



Rechnung Nr. 6

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	10 kg Orangen	1	8.00	20
2	Ernte des ganzen Baumes	1	99.00	20

Zurück zur Starseite

Der Wert bei „Gesamt“ muss korrigiert werden. Es muss der Preis mal Anzahl gerechnet werden. Natürlich ist dort das Zeigen der „Ust“ nicht korrekt.

```
  <td> <?=$order['anzahl']?> </td>
  <td> <?=$order['preis'] ?> </td>
  <td> <?=$order['preis'] * $order['anzahl'] ?> </td>
</tr>
<?php endforeach; ?>
```

Ergebnis:

Rechnung Nr. 6

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	10 kg Orangen	1	8.00	8
2	Ernte des ganzen Baumes	1	99.00	99

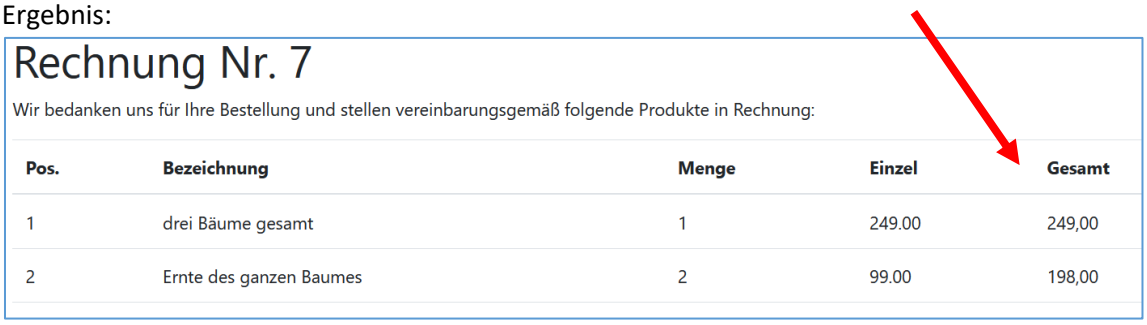
Problem: der Gesamtpreis ist ohne Centbeträge.

Dafür verwende nun die Funktion „number_format()“. Es gibt 4 Attribute in der Klammer, nämlich am Beginn die Variable, dann die Anzahl der Kommastellen, dann der Beistrich für das Komma, dann der Punkt für den Tausender:

<?=number_format($order['preis'] * $order['anzahl'],2,",",".") ?>

```
59                  <td> <?=$order['preis'] ?> </td>
60                  <td> <?=number_format($order['preis'] *
                    $order['anzahl'],2,",",".") ?> </td>
61              </tr>
```

Ergebnis:



Rechnung Nr. 7

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	drei Bäume gesamt	1	249.00	249,00
2	Ernte des ganzen Baumes	2	99.00	198,00

Bitte auch für den Einzelbetrag übernehmen:

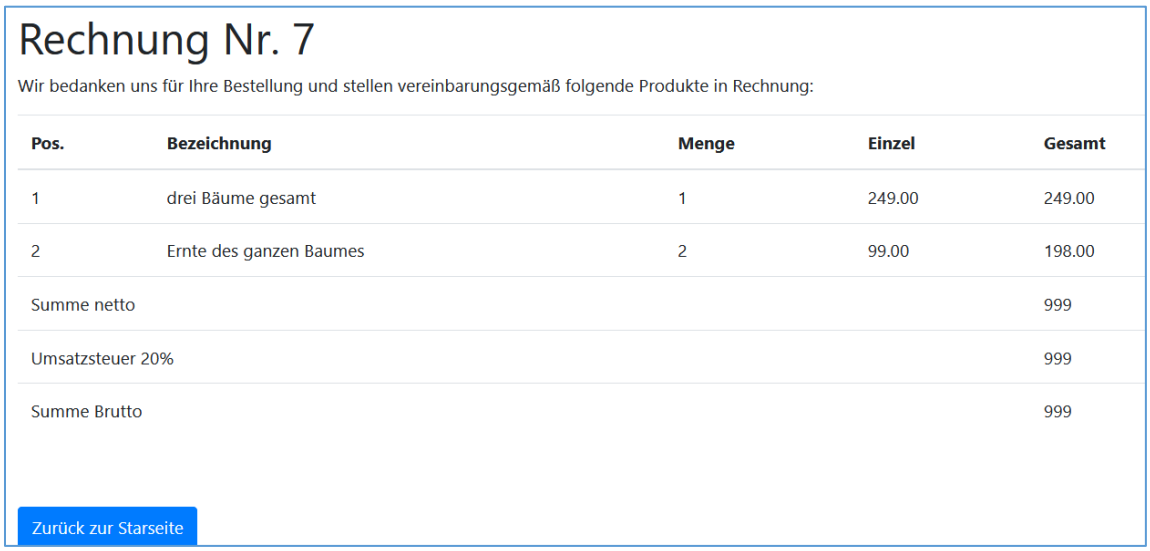
```
<td> <?=number_format($order['preis'],2,',','.')?></td>
<td> <?=number_format($order['anzahl'] *
$order['preis'],2,',','.')?></td>
```

Footer mit Netto, USt und Brutto-Betrag

Im Footer sollen nun ebenfalls mit einer Tabelle die drei Zeilen eingefügt werden, die das Ergebnis anzeigen sollen. Dabei muss man aber ein <colspan> benutzen, damit man die Zeile breit genug macht, nämlich hier mit dem Parameter 4, da man die Spalten addieren muss, wie man sie haben will.. In unserem Fall 4, da diese 4 Spalten breit sein soll und danach, quasi in der 5. Spalte was anderes kommt.

```
64          <tfoot>
65              <tr>
66                  <td colspan="4">Summe netto</td>
67                  <td>999</td>
68              </tr>
69              <tr>
70                  <td colspan="4">Umsatzsteuer 20%</td>
71                  <td>999</td>
72              </tr>
73              <tr>
74                  <td colspan="4">Summe Brutto</td>
75                  <td>999</td>
76              </tr>
77          </tfoot>
78          </table>
```

Ergebnis:



Rechnung Nr. 7

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	drei Bäume gesamt	1	249.00	249.00
2	Ernte des ganzen Baumes	2	99.00	198.00
Summe netto				999
Umsatzsteuer 20%				999
Summe Brutto				999

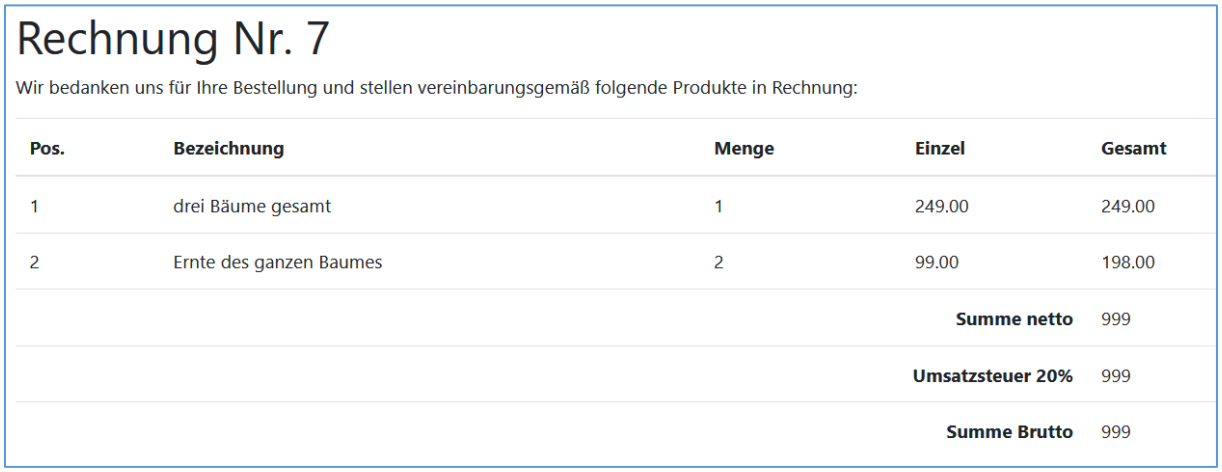
Zurück zur Starseite

Die Beträge werden nun rechtsbündig gemacht und auch mit fetter Schrift:

style="text-align: right;"

```
64          <tfoot>
65          <tr>
66              <td colspan="4" style="text-align: right;"><strong>Summe netto</strong></td>
67              <td>999</td>
68          </tr>
69           <tr>
70              <td colspan="4" style="text-align: right;"><strong>Umsatzsteuer 20%</strong>
                </td>
71              <td>999</td>
72          </tr>
73          <tr>
74              <td colspan="4" style="text-align: right;"><strong>Summe Brutto</strong></td>
75              <td>999</td>
76          </tr>
77          </tfoot>
```

Ergebnis:



Rechnung Nr. 7

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	drei Bäume gesamt	1	249.00	249.00
2	Ernte des ganzen Baumes	2	99.00	198.00
			Summe netto	999
			Umsatzsteuer 20%	999
			Summe Brutto	999

5)Beträge errechnen in auftrag.php (functions)

Dies soll wieder in der Datei „auftrag.php“ in einer neuer Funktion erfolgen.

Diese Funktion bekommt die gleichen Parameter wie die Funktion davor, nämlich:
int $rechnungId,int $userId

```
90  }
91
92  function rechnungBetrag(int $rechnungId,int $userId):?array{
93
94  }
```

function rechnungBetrag(int $rechnungId,int $userId):?array{}

Nun folgen die SELECT und die Berechnung:

Zuerst einmal die allgemeine Überlegung ohne der genauen Werte-Berechnung: man benötigt auf jeden Fall zwei Tabellen daher einen JOIN: wobei man achten muss, dass bei WHERE und AND die richtigen Elemente aus der richtigen Tabelle kommen, daher vorher den Tabellennamen „auftrag.“ mit einem Punkt einfügen.

```
92  function rechnungBetrag(int $rechnungId,int $userId):?array{
93      $sql = "SELECT
94          FROM auftrag_produkte
95          JOIN auftrag ON (auftrag_produkte.auftragId = auftrag.auftragId)
96          WHERE auftrag.userId = :userId
97          AND auftrag.auftragId = :rechnungId
98          ";
```

$sql = "SELECT
FROM auftrag_produkte
JOIN auftrag ON (auftrag_produkte.auftragId = auftrag.auftragId)
WHERE auftrag.userId = :userId
AND auftrag.auftragId = :rechnungId
";

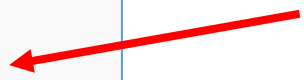
Nun die Berechnung:

```
93      $sql = "SELECT SUM(anzahl*preis) AS sumNetto,
94      SUM(anzahl*preis)*1.2 AS sumBrutto,
95      SUM(anzahl*preis)*0.2 AS betragSteuer
96          FROM auftrag_produkte
```

Mithilfe von „AS“ wird jede Berechnung extra verwendbar, weil sie einen Namen hat, z.B. AS sumNetto.

Danach wieder die Elemente von „prepared statements“ einbauen, am besten man kopiert das von der Funktion darüber, aber beachte, dass es sich hier wieder um 2 Parameter handelt und daher in der „execute“ rechnungId und userId verwendet werden muss:

```
100           ";
101
102         $statement = getDB()->prepare($sql);
103            if(false === $statement){
104            echo printDBErrorMessage();
105            return null;
106         }
107           $statement->execute([
108            ':rechnungId'=>$rechnungId,
109            ':userId'=>$userId
110         ]);
111
112         if(0 === $statement->rowCount()){
113            return null;
114         }
115
116     }
```



Aber zum Abschluss muss man das Array ausgeben und zwar mit der Methode „fetch“, welche die 3 Werte als Array liefert, die in der SELECT als „AS“ erstellt worden sind.

```
113            return null;
114         }
115         return $statement->fetch();
116     }
```

return $statement->fetch();

6)route.php bearbeiten

Die eben erstellten Werte in der SELECT können nun in der route.php verwendet werden, da sie ja wunderbar leicht verfügbar sind in der „auftrag.php“.

Öffne die routes.php.

Füge die eben erstellte Funktion hier ein, damit sie dann in der rechnung.php auch verwendet werden kann:

```
181         $orderData = getAuftragForUser($rechnungId,$userId);
182         $orderSum = rechnungBetrag($rechnungId,$userId);
183
184         if(!$rechnungId){
```

Am besten unter den Variablen orderData als neue Variable „orderSum“, die die Funktion aus „auftrag.php“ beinhaltet:

$orderSum = rechnungBetrag($rechnungId,$userId);

7)rechnung.php – die 3 Werte für die Berechnung verwenden:

Öffne die rechnung.php.

Die 3 Werte, die in der SELECT mit „AS“ bezeichnet worden sind, werden nun hier statt den Dummy-Werten eingesetzt – dies wieder mit der PHP-Abkürzung für den „echo“-Befehl: <?= ?>

Verwende auch die Möglichkeit, das Format korrekt anzuzeigen mit „number_format“:

```
65          <tr>
66              <td colspan="4" style="text-align: right;"><strong>Summe
                netto</strong></td>
67              <td><?= number_format($orderSum['sumNetto'],2,",",".")?></td>
68          </tr>
```

<?= number_format($orderSum['sumNetto'],2,",",".")?>

Nun auch die beiden anderen Werte:

```
70              <td colspan="4" style="text-align: right;"><strong>Umsatzsteuer
                20%</strong></td>
71              <td><?= number_format($orderSum['betragSteuer'],2,",",".")?>
                </td>
72          </tr>
73          <tr>
74              <td colspan="4" style="text-align: right;"><strong>Summe
                Brutto</strong></td>
75              <td><?= number_format($orderSum['sumBrutto'],2,",",".")?></td>
76          </tr>
77          </tfoot>
```

Ergebnis:

Rechnung Nr. 7

Wir bedanken uns für Ihre Bestellung und stellen vereinbarungsgemäß folgende Produkte in Rechnung:

Pos.	Bezeichnung	Menge	Einzel	Gesamt
1	drei Bäume gesamt	1	249.00	249,00
2	Ernte des ganzen Baumes	2	99.00	198,00
			Summe netto	447,00
			Umsatzsteuer 20%	89.40
			Summe Brutto	536.40

8)Rechnungsfußzeile noch darstellen

Die <section> für die „summe“ benötigt man nun nicht mehr. Bitte löschen.

```
78          </table>
79      </section>
80      <section class="row" id="summe">
81      </section>
82      <section class="row" id="rechnungsfooter">
83      </section>
```

Nun wird der „rechnungsfooter“ befüllt.

Hier wird dies eingefügt, am besten mit eine CSS-Style, damit nach oben genug Platz erstellt wird, zwecks Abstand.

```
79      </section>
80      <section class="row" id="rechnungsfooter">
81          <p class="col-12" style="padding-top:120px">
82          Zahlungen innerhalb von 14 Tagen ohne Abzug an die angegebene
            Bankverbindung. IBAN AT60 6000 1000 1234 5678.
83          </p>
84      </section>
```

Zahlungen innerhalb von 14 Tagen ohne Abzug an die angegebene Bankverbindung. IBAN AT60 6000 1000 1234 5678.

Ergebnis:

9)Rechnung ausdrucken

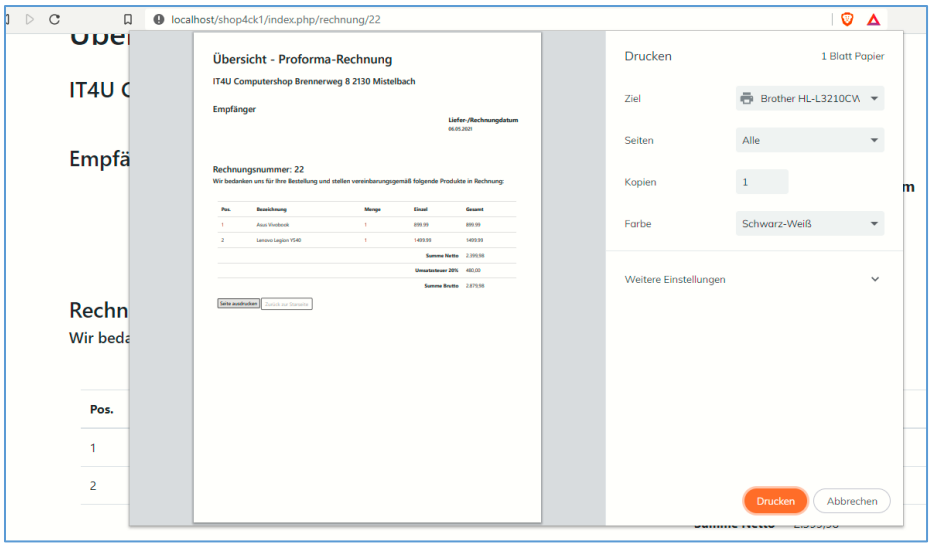
Um die aktuelle Seite auszudrucken gibt es einen einfachen Druckbefehl: onclick="window.print()

https://www.w3schools.com/jsref/met_win_print.asp

Nutze die einfache Print-Funktion. Natürlich gibt es andere Möglichkeiten, vor allem, um das Layout zu verbessern, aber das ist schon ganz gut:

<input type="button" onclick="window.print();" value="Seite ausdrucken" />

```
76          </table>
77
78      </section>
79
80      <input type="button" onclick="window.print();" value="Seite ausdrucken" />
81
82      <a class="btn btn-primary" href="index.php">Zurück zur Starseite</a>
83      </section>
```



Somit kann man es auch als PDF abspeichern, wenn man den Drucker umstellt:

Quelle: https://www.youtube.com/watch?v=jHT56XqSX9I 4:25
Da „createOrder“ ja nur true oder false ausgibt, kann man, außer mit der oben angeführten SESSION-Lösung, keine passende „auftragId“ erhalten, die eine Zahl ist.
Daher muss man eine neue Funktion erstellen:

PDF erzeugen hier: https://www.youtube.com/watch?v=jHT56XqSX9I

Info:
Rechnung als PDF erzeugen

mit wkhtmltopdf

https://www.youtube.com/watch?v=jHT56XqSX9I (Part 39)

Quelle:
https://www.youtube.com/watch?v=5xsBC6ZRN_c (Teil 27)

https://www.youtube.com/watch?v=OUZ4BGfzXR8 (Teil 22, rechnung.php und routes anlegen)

Rechnung erstellen: Teil 24: https://www.youtube.com/watch?v=P3Tk0w5JRYA

Rechnungsadresse sollte man eingeben können, eventuell nach dem „kostenpflichtig“ bestellen in ein Formular, das in eine Datenbank speichert. Dann das in die RG schreiben – ab ca. 13:30