2)Webshop in PHP – Warenkorb erstellen

1)Datenbank anpassen
2)Website verändern – Navbar mit Warenkorb-Symbol und Anzahl
3)Produkte in Warenkorb legen

1)Datenbank anpassen

Überlegungen anhand vom Warenkorb bei amazon.de

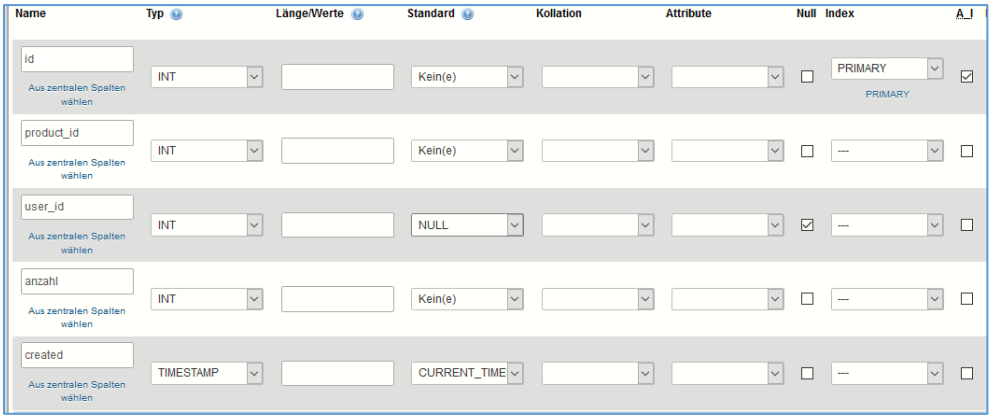
Analyse:

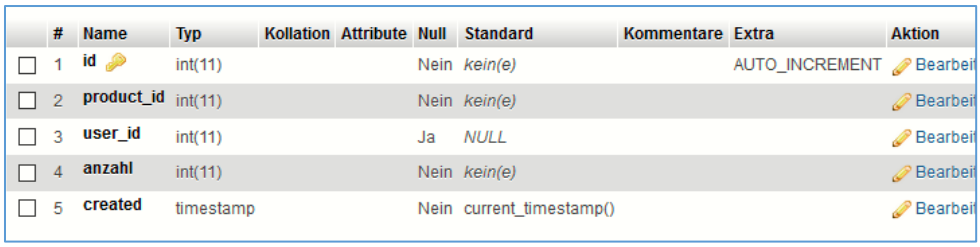
- Man kann Produkte in den Warenkorb legen, auch wenn man nicht eingeloggt ist
- Dort ist es so: wenn man einkauft bekommt man eine Session-ID zugewiesen, die dann in den Cookies gespeichert wird. Wird etwas in den Warenkorb gelegt, wird diese Session-ID auch in die Datenbank eingetragen.
  Info an Minute 1:10
  https://www.youtube.com/watch?v=iyk7ePdeX7w

Daher muss man eine neue Tabelle in der Datenbank anlegen. Diese soll den Namen „korb“ haben.

Folgende Elemente:

- id
- product_id
- user_id – BEACHTE: hier ermögliche, dass dieser Datentyp auch NULL sein darf, sonst gibt es später beim Einfügen des Fremdschlüssels Probleme. Dort wird nämlich „on_delete“ auf „set Null“ erlaubt, damit man im Warenkorb problemlos löschen kann.
- anzahl – damit man diese auch erhöhen kann – 3 mal das Produkt
- created – timestamp – damit kann man absteigend oder aufsteigend durchsuchen, wann das Produkt zum Warenkorb hinzugefügt wurde. Beachte, dass der Standard auf „CURRENT_TIME eingestellt ist





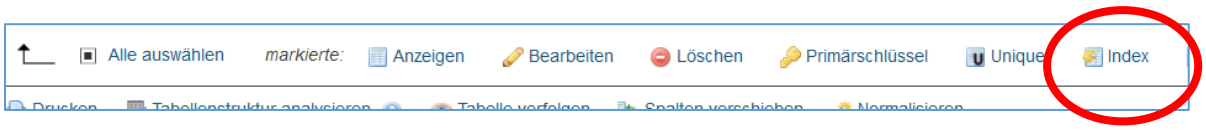
#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	id	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten
2	product_id	int(11)			Nein	*kein(e)*			Bearbeiten
3	user_id	int(11)			Ja	*NULL*			Bearbeiten
4	anzahl	int(11)			Nein	*kein(e)*			Bearbeiten
5	created	timestamp			Nein	current_timestamp()			Bearbeiten

Aber: es soll ein Produkt NICHT mehrfach hinzugefügt werden können, nur die Anzahl soll verändert werden können. Das muss hier in der Datenbank geregelt werden:

- Die Kombination von „user_id“ und „product_id“ muss einzigartig sein.
- Die id der Tabelle „products“ wird als Fremdschlüssel in der Tabelle „korb“ verwendet, damit hier eine Beziehung vorhanden ist.
  Für die Verwendung von einem Fremdschlüssel ist es nötig, dass die Tabellen die gleiche Kollation aufweisen. Hier ist das „utf8mb4_general_ci“.

Index erzeugen

Da man später oft nach der „user_id“ suchen wird, bzw. diese abfragen wird, sollte man dieses Element zu einem „index“ machen. Damit wird eine schneller Suche bzw. Abfrage ermöglicht. Wähle daher diese Zeile aus und klicke auf den Button „Index“. Diese Möglichkeiten werden unterhalb der Tabelle in der „Struktur“ angeboten.



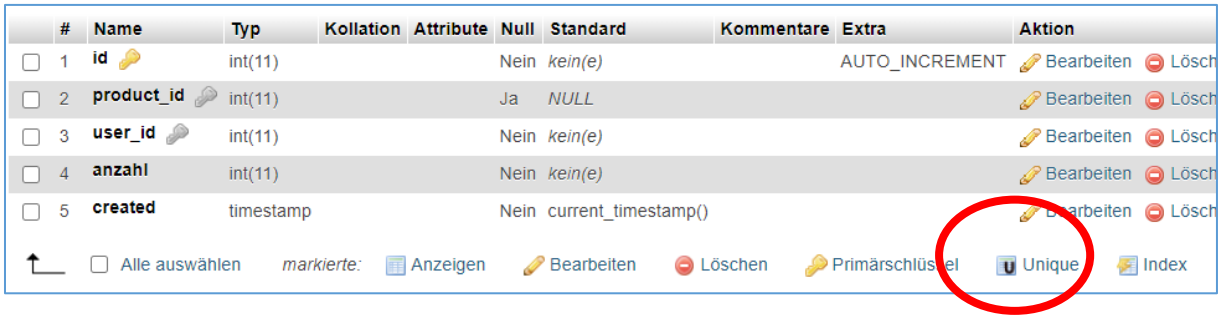
2. Variante: Man kann auch rechts in der Zeile auf „Mehr“ klicken und dort „Index“ auswählen.

Unique-Key erzeugen

Um zu vermeiden, dass ein Produkt mehrfach in den Warenkorb hinzugefügt wird, muss man die „product_id“ und „user_id“ verknüpfen und als „unique“ bezeichnen. Würde ein User dann ein Produkt ein zweites Mal hinzufügen, würde eine Info (Fehlermeldung) aus der Datenbank erfolgen.

Dazu wähle beide im vorangestellten Kästchen aus und klicke dann auf den Button „Unique“.

Ergebnis: bei „product_id“ und „user_id“ werden zwei graue Schlüsselsymbole angezeigt.



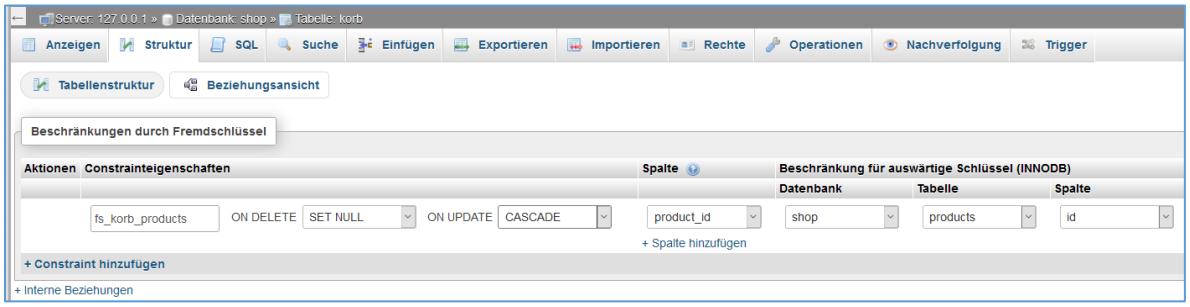
#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	id	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten Lösch
2	product_id	int(11)			Ja	*NULL*			Bearbeiten Lösch
3	user_id	int(11)			Nein	*kein(e)*			Bearbeiten Lösch
4	anzahl	int(11)			Nein	*kein(e)*			Bearbeiten Lösch
5	created	timestamp			Nein	current_timestamp()			Bearbeiten Lösch

Fremdschlüssel herstellen:

https://www.youtube.com/watch?v=7HTAX0dfJqg

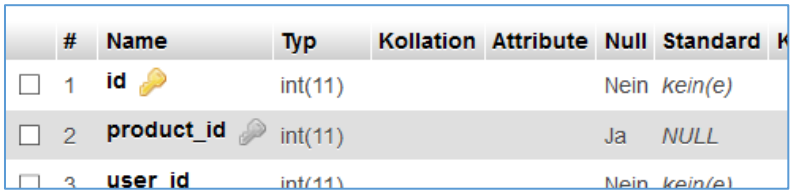
a)In der Tabelle, in der der Fremdschlüssel vergeben werden soll, kicke auf „Beziehungsansicht“:

b)vergib oben einen eindeutigen Namen, der sich am Besten aus den beiden Tabellenamen zusammensetzt, z.B. fs_korb_products. (fs für Fremdschlüssel).

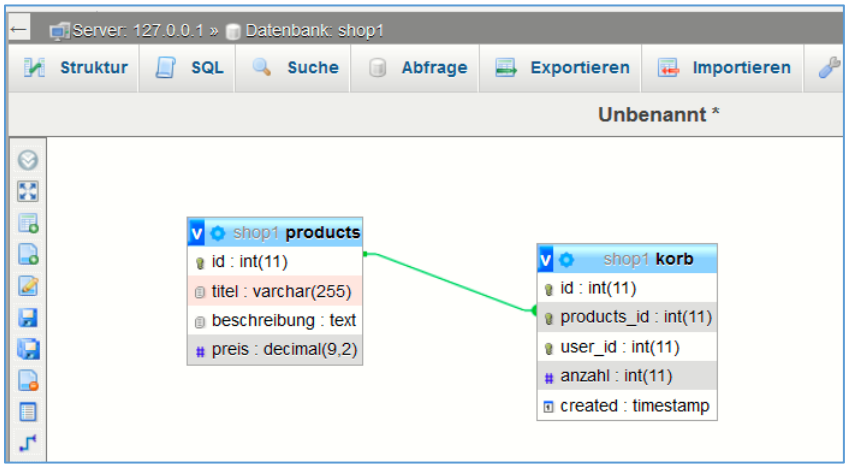


- Danach kann man bequem die Spalten usw. auswählen, wobei zu Beginn die Spalte gewählt wird, über die verknüpft wird, hier die „product_id“.
- Die zwei Auswahlen sind folgenermaßen zu belegen:
  - On delete: „set null“ – damit wird das Löschen im Warenkorb auch weitergeleitet und Null ermöglicht – Beachte: in der Tabelle „korb“ muss der Eintrag „product_id“ eine „NULL“ zulassen.
  - On update: „cascade“ – damit wird die „product_id“ im Warenkorb aktualisiert.

Ergebnis: KEY-Symbol bei „product_id“:



Mit dem Designer betrachtet:

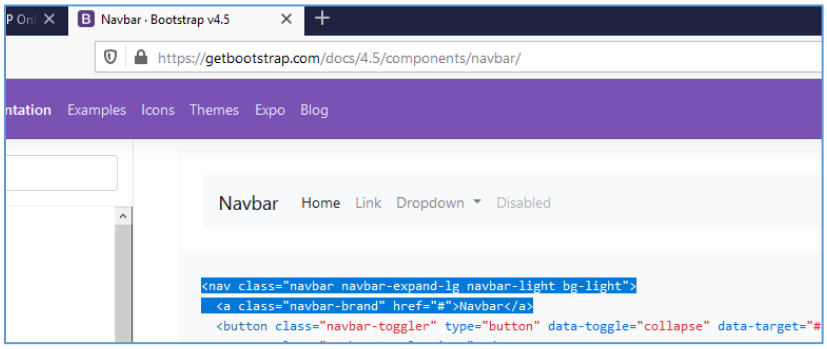


2)Website verändern – Navbar mit Warenkorb-Symbol und Anzahl

- **NavBar erstellen**

Oberhalb der Produkte soll eine Zeile (NavBar) angezeigt werden, in der das Warenkorbsymbol und später die Anzahl der aufgenommenen Produkte abgelegt ist.
Dafür verwende einen Teil der NavBar aus www.getbootstrap.com

Aber der Besuch der Site ist nicht nötig, da der Code hier unter dem Bild kopiert werden kann.



Kopiere diese ersten 2 Zeilen und füge sie in eine neue Datei ein.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 <a class="navbar-brand" href="#">Navbar</a>
```

Diese neue Datei „navbar.php“ soll im Ordner „templates“ liegen. Diese NavBar soll später in mehreren Seiten angezeigt werden und daher wird man dann darauf zugreifen.

Ändere die Bezeichnung auf „Mein Shop“ und füge als Link href=“index.php“ ein.

Darunter wird diese Bezeichnung mit „ml-auto“ ganz nach rechts geschoben. („ml“ steht für margin left, „auto“ sorgt dafür, dass diese „ml“, also links, automatisch so groß wird. Das schiebt alles nach rechts.)

```
1 <nav class="navbar navbar-expand-lg navbar-light bg-light">
2   <a class="navbar-brand" href="index.php">Mein Shop</a>
3   <ul class="navbar-nav ml-auto">
4     <li class="nav-item">
5       Warenkorb ()
6     </li>
7   </ul>
8 </nav>
```

```
<ul class="navbar-nav ml-auto">
   <li class="nav-item">
     Warenkorb ()
   </li>
 </ul>
</nav>
```

Damit es zentrierter aussieht, erstelle noch einen div-Container:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
<div class="container">
  <a class="navbar-brand" href="index.php">Mein Shop</a>
  <ul class="navbar-nav ml-auto">
    <li class="nav-item">
      Warenkorb ()
    </li>
  </ul>
</div>
</nav>
```

Speichern nicht vergessen.

- **NavBar-Datei einbinden**

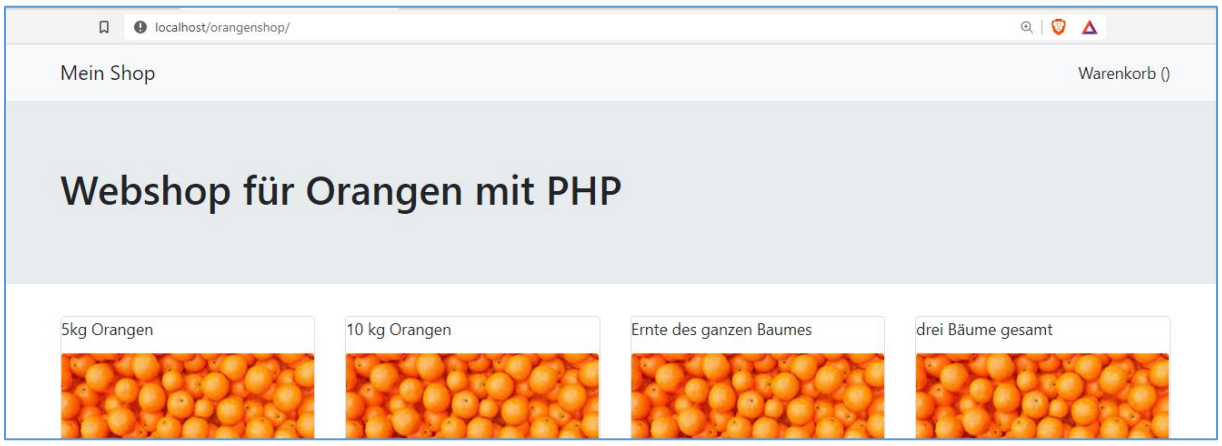
Damit diese NavBar nun auch in der Seite angezeigt werden kann, muss sie eingebunden werden.

Füge daher in der „main.php“ gleich am Beginn des <body> diese Verknüpfung ein:

```
<body>
  <?php include __DIR__.'/navbar.php' ?>
  <?php include __DIR__.'/header.php' ?>
```

<?php include __DIR__.'/navbar.php' ?>

Ergebnis:



- **Anzahl der Produkte im Warenkorb anzeigen**

Die in den Warenkorb gelegten Produkte sollen als Zahl angezeigt werden.

Dafür benötigt man

a) eine Variabel in der Datei „navbar.php“
b) eine SQL-Query in der index.php (mit SELECT und COUNT) für diesen User
c) SESSION starten

a)Variable erstellen - $korbZahl

Öffne die „navbar.php“.

In der NavBar soll neben dem Wort „Warenkorb“ in Klammer die Anzahl der geklickten Produkte angezeigt werden. Dafür erstelle in einem PHP-Umfeld die neue Variable

```
<li class="nav-item">
  Warenkorb (<?= $korbZahl ?>)
</li>
```

Warenkorb (<?= $korbZahl ?>)

WICHTIG: Kein Leerzeichen zwischen ? und dem =

b)SQL erstellen und Produkte in den Warenkorb legen

Ab nun wird in der „index.php“ gearbeitet. Hier am Beginn wird der komplette Code der nun entstehenden „index.php“ abgebildet. Dieser kann kopiert werden. Will man die Einzelheiten wissen, kann man darunter die Details und den langsamen Fortschritt selbst durchführen.

```
<?php
session_start();
error_reporting(-1);
ini_set('display_errors','On');

define('CONFIG_DIR',__DIR__.'/config');
require_once __DIR__.'/function/database.php';

$userId = 0;
$korbZahl = 0;

if(isset($_SESSION['userId'])){
   $userId = (int) $_SESSION['userId'];
}

$sql = "SELECT id, titel, beschreibung, preis, anzahl, produktbild FROM products";
$result = getDB()->query($sql);

$sql ="SELECT COUNT(id) FROM korb WHERE user_id =" .$userId;
$korbResult = getDb()->query($sql);

$korbZahl = $korbResult->fetchColumn();

$url = $_SERVER['REQUEST_URI'];
$indexPosition = strpos($url, 'index.php');
$route = substr($url,$indexPosition);
$route = str_replace('index.php', '',$route);
```

```
if(strpos($route,'/cart/add/') !== false) {
   $routeTeile = explode("/",$route);
   $productId = (int)$routeTeile[3];
$sql = "INSERT INTO korb SET
      anzahl=1,
      user_id = :userId,
      product_id = :productId";
   $statement = getDB()->prepare($sql);

   $statement->execute([
      ':userId'=>$userId,
      ':productId'=> $productId
   ]);
   header("Location: /shop/index.php");
   exit();
}

require __DIR__.'/templates/main.php';
?>
```

Zusätzlich muss in der „card.php“ noch folgendes eingefügt werden.

Dafür muss in der „card.php“ beim Button „Warenkorb“ erst einmal ein Link gesetzt werden.

Beachte die ID wird in PHP-Form der URL mitgegeben

```
11          <a href="index.php/cart/add/<?= $row['id']?>" class="btn btn-
            success ">In den Warenkorb</a>
```

href="index.php/cart/add/<?= $row['id']?>"

BEACHTE NUR:

Der Ordner in XAMPP/htdocs hat hier den Namen „shop“. Ist er bei dir anders, muss du in der „index“ die letzte Umleitung auch anders wählen:

```
    header("Location: /shop/index.php");
    exit();
}
```

FERTIG.

AB hier ist alles für die Erklärungen und den langsamen Fortschritt in Eigenproduktion:

Öffne die „index.php“.

- Mit SELECT wird die id geholt und mit COUNT diese gezählt
- Aus der Tabelle „korb“
- Wobei die user_id noch definiert werden muss

Schreibe diesen Code aus Zeile 12 in der „index.php“:

```
10  $result = getDB()->query($sql);
11
12  $sql ="SELECT COUNT(id) FROM korb WHERE user_id =" .$userId;
13
```

$sql ="SELECT COUNT(id) FROM korb WHERE user_id =" .$userId;

Diese Variable $userId soll aus der SESSION (bzw. wenn das nicht funktioniert aus der den Cookies) ausgelesen werden.

Dafür muss man oben gleich nach dem PHP-Tag die Session starten:

session_start();

Zuerst wird dann die Variable $userId mit Null definiert und anschließend überschrieben, wenn es einen anderen Wert gibt, der aus der Session geholt wird, wenn man eingeloggt ist. Aber mit Null soll sie starten:
Die IF fragt, wenn die Session gesetzt ist, dann soll diese genommen werden.

```
13  $userId = 0;
14  if(isset($_SESSION['userId'])){
15      $userId = (int) $_SESSION['userId'];
16  }
```

$userId = 0;
if(isset($_SESSION['userId'])){
  $userId = (int) $_SESSION['userId'];
}

- Die Variable „$korbZahl“

Gleich unter der Definition der $userId wird diese ebenfalls auf Null gestellt, weil es ja noch keine Einträge gibt.

```
13  $userId = 0;
14  $korbZahl = 0;
15
```

$korbZahl = 0;

Wenn ein Benutzer Waren auswählt und in den Warenkorb legt, landen diese ja in der Datenbank in unserer Tabelle „korb“. Siehe oben. Nun kann man den Wert aus der Datenbank auslesen und hier anzeigen lassen, der dann die Null überschreibt.

Zuerst wird eine neue „result“ erzeugt, die hier „$korbResult“ heißen soll.
Zeile 24: hier unter dieser $sql, da man das COUNT(id) benötigen wird

```
23  $sql ="SELECT COUNT(id) FROM korb WHERE user_id =" .$userId;
24  $korbResult = getDb()->query($sql);
25
```

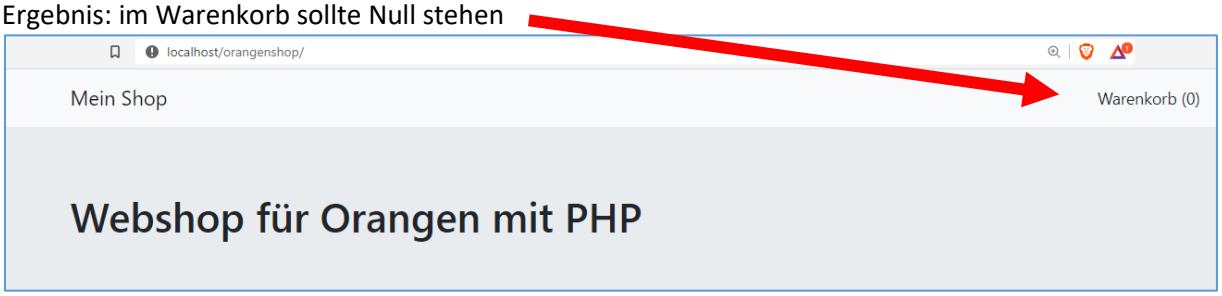
$korbResult = getDb()->query($sql);

Darunter wird nun die ursprüngliche Variabel $korbZahl, die ja mit Null begonnen hat, überschrieben, da die $korbResult eine neue Zahl liefern wird, die aus dem COUNT(id) kommt. Diese wird mit „fetchColumn()“ geholt. Da in der $sql ja nur diese eine Spalte abgefragt wird, nämlich COUNT(id) ist das die beste Wahl:

```
23  $sql ="SELECT COUNT(id) FROM korb WHERE user_id =" .$userId;
24  $korbResult = getDb()->query($sql);
25
26  $korbZahl = $korbResult->fetchColumn();
```

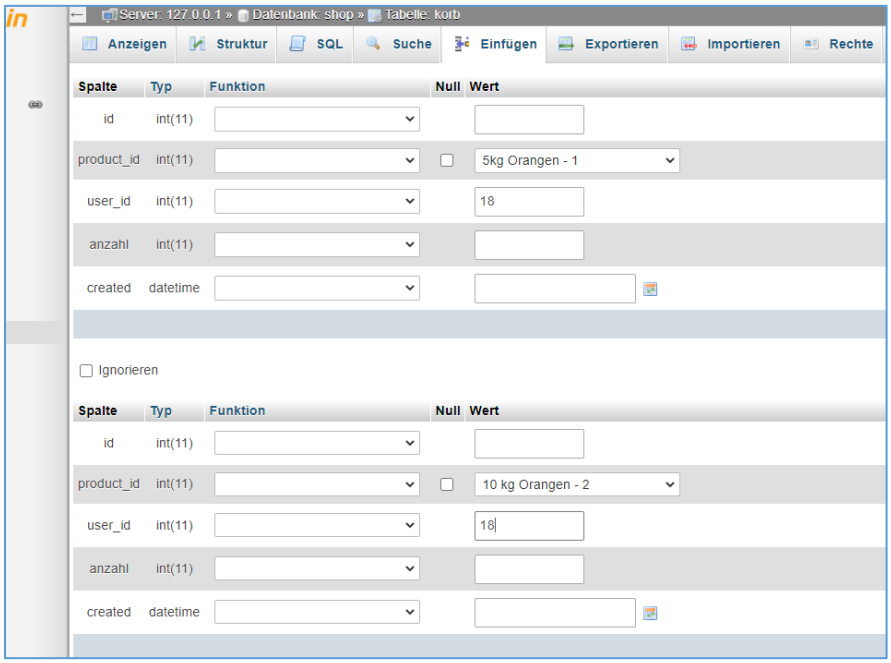
$korbZahl = $korbResult->fetchColumn();

Ergebnis: im Warenkorb sollte Null stehen

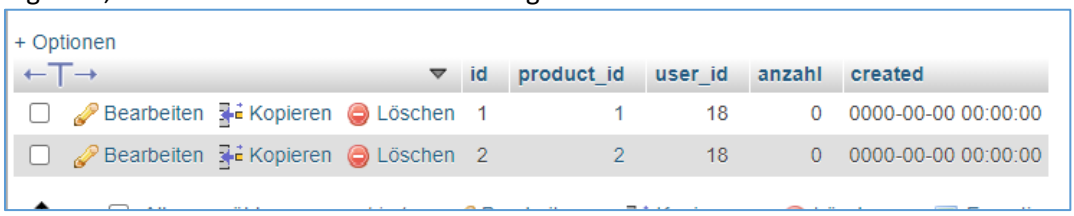


<u>Testen:</u>
Nun sollen in der Datenbank und der Tabelle „korb“ zwei Produkte angelegt werden und auch ein User.
Öffne die Tabelle „korb“ und füge mit dem Reiter „Einfügen“ folgende Produkte und folgenden user ein:

Die Produkte kann man einfach auswählen, da ja schon welche vorhanden sind:



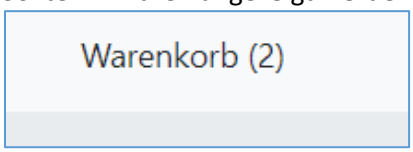
Ergebnis, auch wenn eventuell rote Warnungen erschienen sind:



Wenn man nun im Code in der Dateien „index.php“ die Zahl 18 in die Variable „$userId“ schreibt:

```
12
13   $userId = 18;
14   $korbZahl = 0;
```

Sollten 2 Waren angezeigt werden:



Passt.

Lösche die „18“ wieder und auch die beiden Einträge in der Datenbank.

3)Produkte in Warenkorb legen

3.1)Warenkorb Einträge vorbereiten

Hier wird mit einer Server-Variable gearbeitet und nicht mit einer GET-Variablen.

3.1.1)Link setzen

Öffne die „card.php“.
Dafür muss in der „card.php“ beim Button „Warenkorb“ erst einmal ein Link gesetzt werden.

Beachte die ID wird in PHP-Form der URL mitgegeben

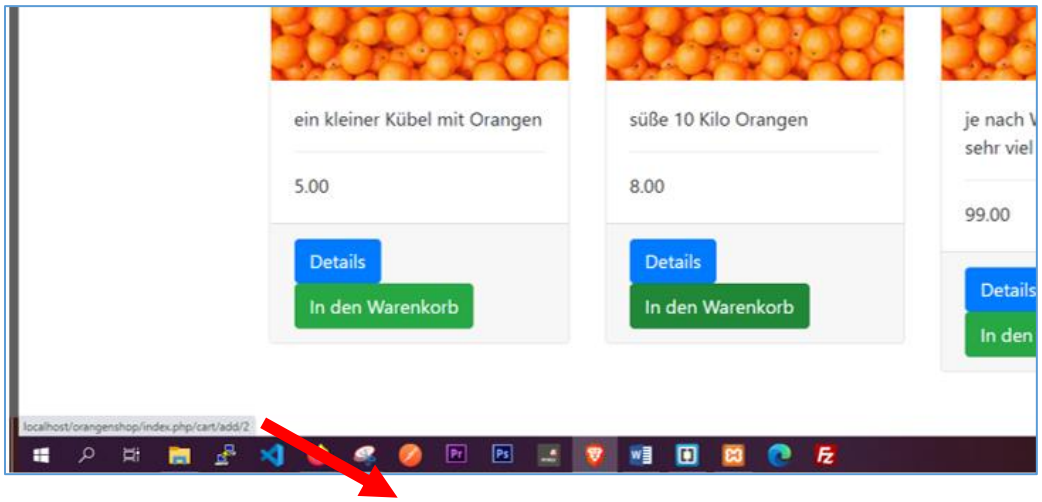
```
11          <a href="index.php/cart/add/<?= $row['id']?>" class="btn btn-
            success ">In den Warenkorb</a>
12      </div>
```

href="index.php/cart/add/<?= $row['id']?>"

Ergebnis:
Wenn man auf einen der grünen Buttons „In den Warenkorb“ klickt, wird als letzte Position im Link ganz unten links im Eck immer eine Zahl ausgegeben, die von der „$row['id']“ stammt und somit aus der Datenbank und der „id“.
Diese kommt über den Umweg der „index.php“ aus dieser Abfrage:

```
$sql = "SELECT id,titel,beschreibung,preis,anzahl,produktbild FROM products";
```

Das ist diese „id“:



Somit kann man mit dieser ID (Zahl) arbeiten.

3.1.2. Server Variable verwenden

Problem: welches Produkt soll hinzugefügt werden?
Herausfinden, bei welches Produkt aus dem ganzen Warenkorb auf den Button geklickt wurde und somit hinzugefügt werden soll.
Dafür werden folgende Elemente erstellt:

- Eine SERVER_URI erstellt

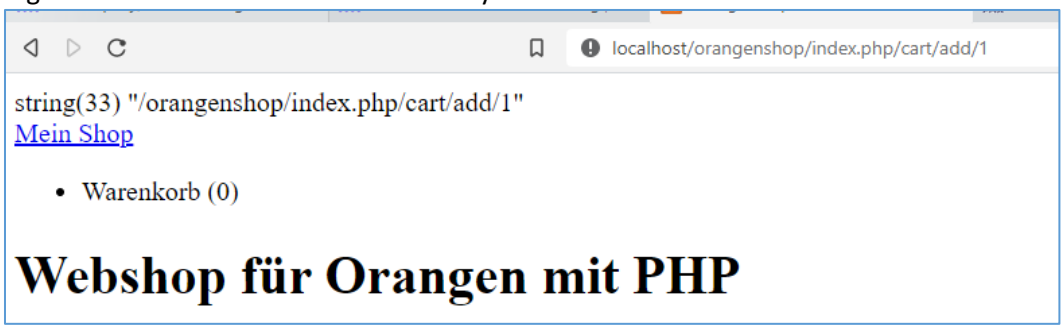
- Die aktuelle URL wird mit „strpos“ und „substr“ beschnitten
- Routen erstellt mit einer Variablen „route“
- Die Position für die Zahl (ganz hinten) herausgenommen (explode()

Erklärung mit „var_dump“ in der „index.php“:

```
23  $korbZahl = $korbResult->fetchColumn();
24
25  var_dump($_SERVER['REQUEST_URI']);
26
```

var_dump($_SERVER['REQUEST_URI']);

Ergebnis bei Klick auf das 1. Produkt: Layout ist hier noch nicht ok ☺



In Wirklichkeit interessiert uns aber nicht der ganze Pfad, sondern nur die letzte zahl, nämlich hier der „1“, damit man genau dieses Produkt in die Tabelle „korb“ der Datenbank senden kann.

Wie kommt man dahin?

- Den „var_dump“ in eine Variable „$url“ umwandeln
  Die Zeile gleich nutzen aber umschreiben – Achte auf die runden Klammern, die verschwinden

```
26  $url = $_SERVER['REQUEST_URI'];
```

- Ersten Teil wegschneiden mit „strpos“ (string position)
  Erstelle eine Variable $indexPosition und bestimme die Startposition von dem Wort „index“ aus dieser „$url“. Die Antwort ist somit eine Zahl, wo „index“ beginnt:

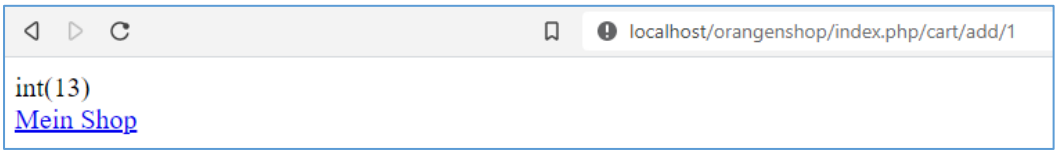
```
26  $url = $_SERVER['REQUEST_URI'];
27  $indexPosition = strpos($url,'index.php');
28
```

  $indexPosition = strpos($uri, 'index.php');

  Wenn man dies ausgibt, sieht man, dass das „i“ von „index.php“ an der Position beginn:

```
27  $indexPosition = strpos($url,'index.php');
28
29  var_dump($indexPosition);
30
```

Ergebnis der Position: 13



Stimmt das?
ja, orangenshop hat 11 Buchstaben, der Slash danach auch einen, daher steht das „i“ an Position 13.
Lösche den „var_dump“ wieder.

- Um die „url“ aufzuteilen benutzt man die Funktion „substr()“ – substring
  Daher erstelle eine Variable „route“.
  Danach wird sofort der unnötige Bereich „index.php“ entfernt mit „str_replace()“ – string replace. Man möchte später nur mit dem hinter Teil ab „cart“ weiterarbeiten.

```
28  $route = substr($url,$indexPosition);
29  $route = str_replace('index.php', '',$route);
```

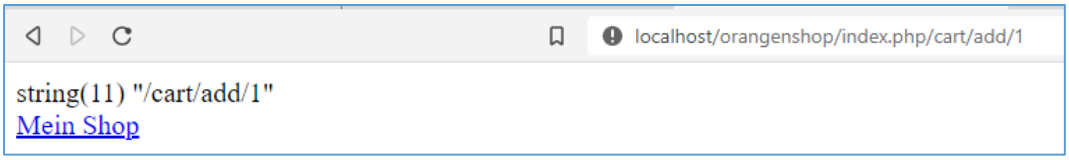
$route = substr($url,$indexPosition);
$route = str_replace('index.php', '',$route);

D.h.: beim „replace“ wird die „index.php“ mit „nichts“ (= ' ',) ersetzt. Da ist ja nichts zwischen den Anführungszeichen. Das innerhalb der Variable „route“.

Ergebnis mit „var_dump“ zum Prüfen:

```
28  $route = substr($url,$indexPosition);
29  $route = str_replace('index.php', '',$route);
30
31  var_dump($route);
```

Der String ist kürzer geworden:



Somit wurde nun eine eigene ROUTE definiert. Diese ist eine eindeutige URL, mit der man nun bequemer weiterarbeiten kann. Mit Routen kann man hier besser arbeiten als mit GET-Parameter.

- Nun wird mit einer IF-Abfrage geklärt, ob in der URL „cart/add“ steht
- Die „strpos“ liefert ein „false“ wenn der Text zwischen den Anführungszeichen nicht gefunden wird, also macht man dann das Gegenteil mit dem Rufzeichen vor den beiden „==“.
- Wenn ja, dann soll die URL weiter geteilt werden
- Dies passiert mit „explode“ – damit kann man auf die einzelnen Elemente der URL zugreifen und hier nach den „Slashes“ aufteilen – die Variabel „routeTeile“ enthält somit alle Teile aus der URL
- Uns interessiert aber nur die letzte Position, dort wo die Zahl für die ID des Produktes steht

Lösche den „var_dump“ wieder.

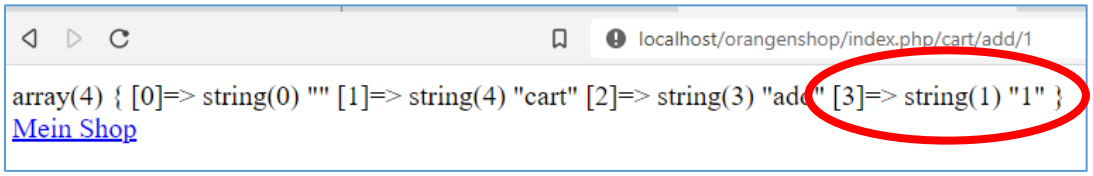
```
30
31 if(strpos($route,'/cart/add/') !== false) {
32     $routeTeile = explode("/",$route);
33 }
```

```
if(strpos($route,'/cart/add/') !== false) {
   $routeTeile = explode("/",$route);
}
```

Testen:
Ansehen und Prüfen mit „var_dump“:

```
31 if(strpos($route,'/cart/add/') !== false) {
32     $routeTeile = explode("/",$route);
33 }
34
35 var_dump($routeTeile);
```

Ergebnis:



d.h. unsere wichtige ID steht an Position „3“.

3.1.2.1) $productID festlegen

Nun kann man diese ID in eine neue Variable platzieren.
Da es ein String ist, siehe oberes Bild, muss das noch in ein Integer gewandelt werden.
Man weiß ja jetzt, dass es an Position 3 sich befindet. Daher wird diese in die eckigen Klammern geschrieben.

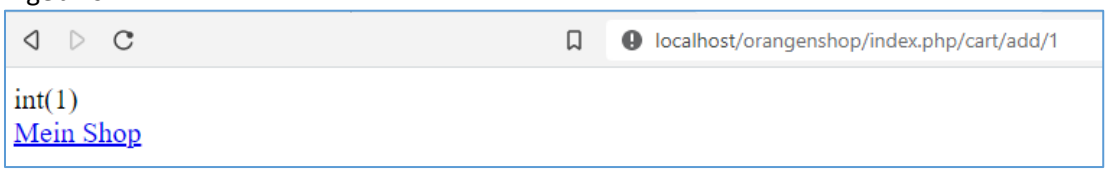
```
31 if(strpos($route,'/cart/add/') !== false) {
32     $routeTeile = explode("/",$route);
33     $productId = (int)$routeTeile[3];
34 }
```

Ergebnis:
Die ID aus der URL wurde korrekt ausgelesen und kann nun in einem SQL-Query weiterverwendet werden.

Testen: Gib mit „var_dump“ diese Variable aus.

```
34  }
35
36  var_dump($productId);
```

Ergebnis:



Korrekt, da ja der Button vom ersten Produkt geklickt wurde.

3.3) Warenkorb Eintrag in die Datenbank schreiben

Dies soll um die Sicherheit zu erhöhen mit „prepaired statements“ erfolgen. Daher muss man Platzhalter (mit Doppelpunkt am Beginn) arbeiten. Das verhindert schädliche SQL-Injections.

Zuerst muss man aber eine neue SQL-Query erzeugen mit „INSERT INTO“.

3.3.1) INSERT INTO

Erstelle in der gerade erzeugten „IF“-Anweisung eine Variabel „$sql“ mit einem SQL-Befehl INSERT INTO

```
33      $productId = (int)$routeTeile[3];
34      $sql = "INSERT INTO korb SET
35          anzahl=1,
36          user_id = :userId,
37          product_id = :productId";
38  }
```

```
$sql = "INSERT INTO korb SET
   anzahl=1,
   user_id = :userId,
   product_id = :productId";
```

Man kann, wie hier, statt „value“ auch „set“ verwenden.

Es werden die userId und die productId in die Tabelle „korb“ der Datenbank geschrieben.
Aber auch die Anzahl, die hier sofort auf 1 festgelegt wird, da ja ein Klick bedeutet, dass 1 Produkt hinzugefügt wird.

Zur Erinnerung hier die Tabelle „korb“:

Optionen

	id	product_id	user_id	anzahl	created
Bearbeiten Kopieren Löschen	78	2	5	1	2020-08-21 16:10:57

Für das „prepaired statement“ verwende folgenden Code:

```
35          user_id = :userId,
36          product_id = :productId";
37      $statement = getDB()->prepare($sql);
38
39      $statement->execute([]);
40  }
```

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

In Zeile 39 muss das Statement ausgeführt werden. Die Details folgen in den Klammern: dabei wird der Datenbank mitgeteilt, wodurch die Platzhalter ersetzt werden sollen, so z.B. der Platzhalter :userId mit der Variablen „$userId“.

```
39      $statement->execute([
40          ':userId'=> $userId,
41          ':productId'=> $productId
42      ]);
```

3.3.2) Redirect zur Startseite

Danach wird eine Umleitung zur Startseite vorgenommen, damit sich die Seite aktualisiert. Dies erfolgt mit der Funktion „header()“.

```
42      ]);
43      header("Location: /orangenshop/index.php");
44      exit();
45  }
46
```

```
header("Location: /orangenshop/index.php");
exit();
```

Testen:

Klicke 1 und dann noch ein Produkt an. Beobachte die Veränderung bei der Zahl im Warenkorb rechts oben.

Beachte:
Wird ein Produkt doppelt geklickt, **ändert sich die Anzahl NICHT**. Dafür wurde ja vorgesorgt, da später die Anzahl direkt geändert werden soll. Dafür wurde vorgesorgt, als wir die „user_id“ und „product_id“ in der Datenbank gemeinsam zu „unique“ gemacht haben.

Veränderung in der Datenbank: die Produkte wurden aufgenommen: