4)PDO-Shop – Warenkorb erstellen

1) Datenbank – neue Tabelle anlegen
2) Fremdschlüssel vergeben
3) Navbar anpassen
4) Anzahl der Produkte anzeigen
5) Waren in den Warenkorb legen

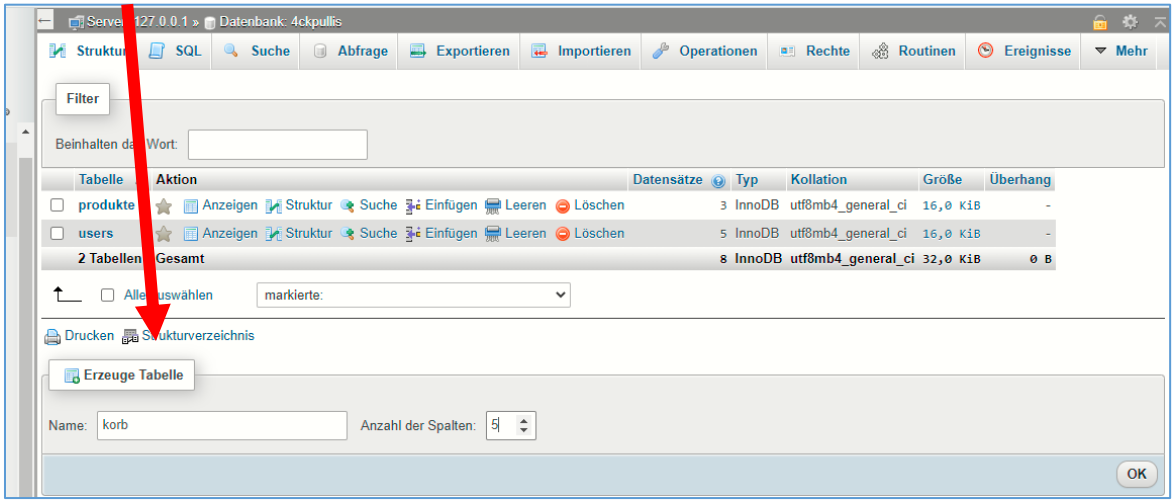
Überlegungen anhand vom Warenkorb bei amazon.de – wie sieht es dort aus?

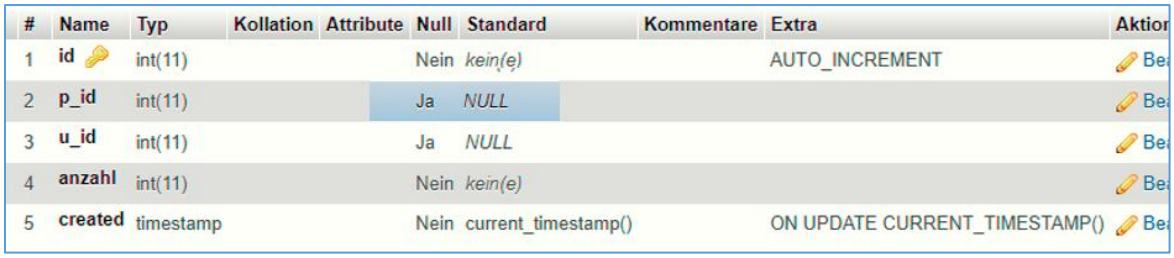
Analyse: Man kann Produkte in den Warenkorb legen, auch wenn man nicht eingeloggt ist. Das soll auch bei uns so sein.

1)Öffne die Datenbank.

Zuerst muss man eine neue Tabelle in der Datenbank anlegen, die den Warenkorb abbildet. Daher muss man eine neue Tabelle in der Datenbank anlegen. Diese soll den Namen „korb“ haben.

Das passiert hier unten: „Erzeuge Tabelle“ mit Spalten 5



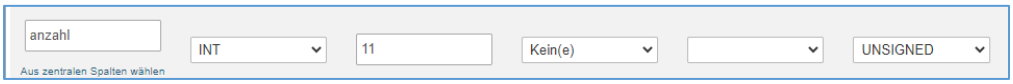


Folgende Elemente:

- id
- p_id (genauso wie der Primärschlüssel in „produkte“)
  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.
- u_id (genauso wie der Primärschlüssel in „users“)
  BEACHTE: hier ermögliche, dass dieser Datentyp auch NULL sein darf, sonst gibt es später

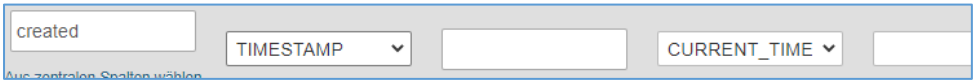
beim Einfügen des Fremdschlüssels Probleme. Dort wird nämlich „on_delete“ auf „set Null“ erlaubt, damit man im Warenkorb problemlos löschen kann.

- anzahl – damit man diese auch erhöhen kann – 3 mal das Produkt
  Beachte: damit man nicht weniger als NULL haben kann, muss man das Attribut auf „unsigned“ stellen:



„Mysql allows numeric column types to be UNSIGNED which basically means it cannot be negative.“

- 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



Hinweis:

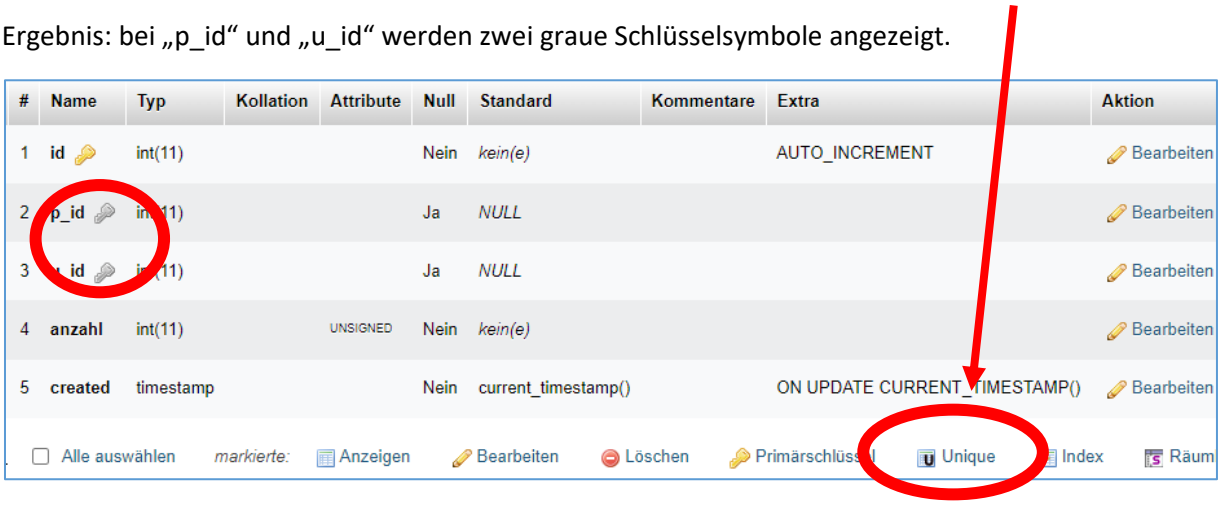
Für die Verwendung von einem Fremdschlüssel ist es nötig, dass die Tabellen die gleiche Kollation aufweisen. Hier ist das „utf8mb4_general_ci“.

Unique-Key erzeugen

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

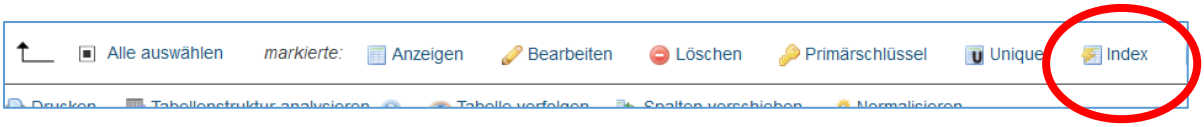
Dazu wähle beide im vorangestellten Kästchen aus und klicke dann auf den Button „Unique“.

Ergebnis: bei „p_id“ und „u_id“ werden zwei graue Schlüsselsymbole angezeigt.



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.



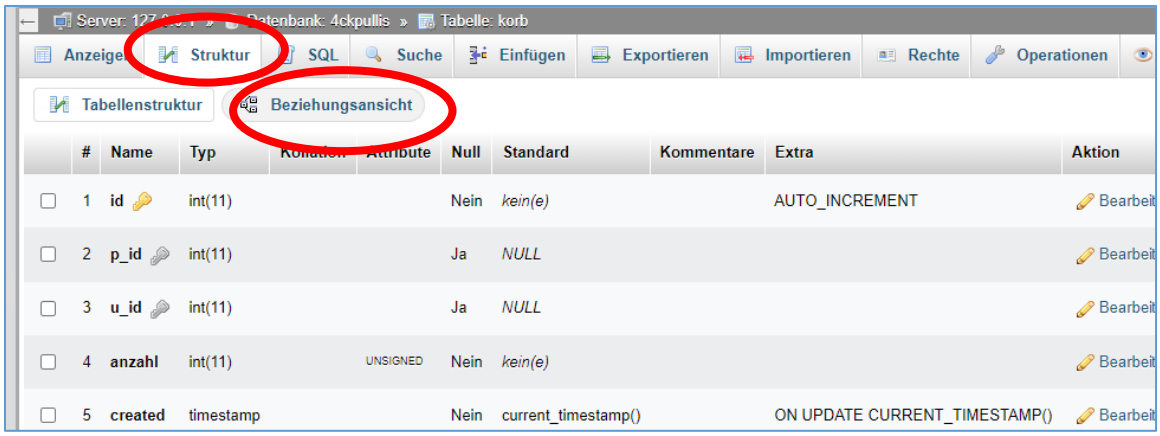
2. Variante: Man kann auch rechts in der Zeile auf „Mehr“ klicken und dort „Index“ auswählen.

2)Fremdschlüssel herstellen:

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

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

- Tabelle korb, weil hier wird die ID vom „produkte“ eingepflanzt als Fremdschlüssel
- Klicke auf die Struktur
- Öffne „Beziehungsansicht“



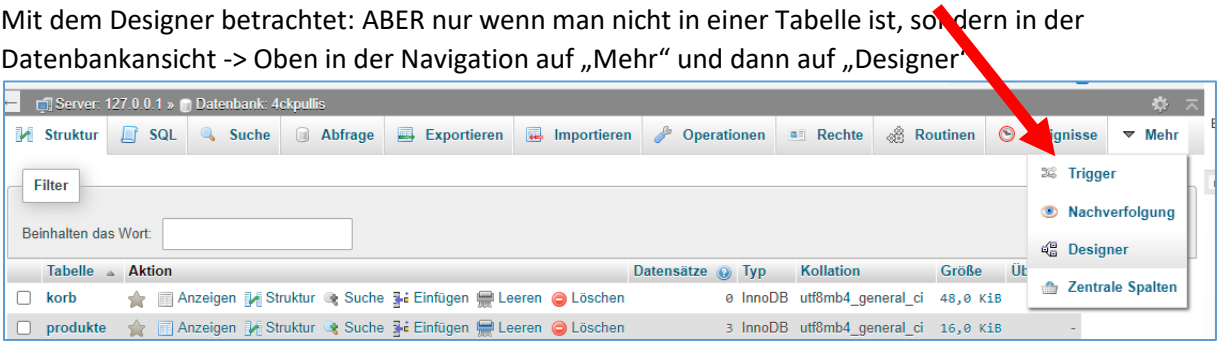
b) Vergib oben einen eindeutigen Namen, der sich am besten aus den beiden Tabellenamen zusammensetzt, z.B. fs_korb_produkt. (fs für Fremdschlüssel).

- Danach kann man bequem die Spalten usw. auswählen, wobei zu Beginn die Spalte gewählt wird, über die aus dieser Tabelle stammt und verknüpft wird, hier die „p_id“.
- Die Datenbank ist (leider) auch auszuwählen, wobei das eh klar ist. Es ist die Datenbank, in der man sich gerade befindet.
- Dann die andere Tabelle, mit der man sich verknüpft, hier die „produkte“.
- Als viertes wird eh nur die dazu passende ID vorgeschlagen.

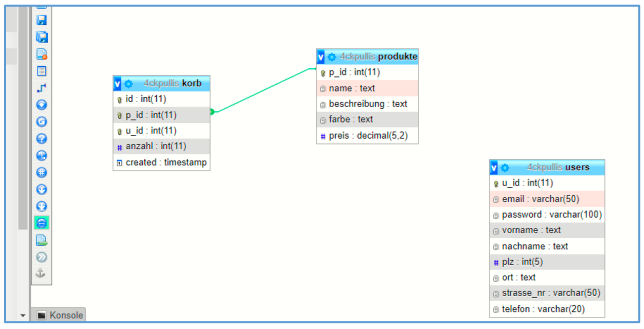
- Die zwei SPEZIAL-Auswahlen sind folgenermaßen zu belegen: (no action)
    - On delete: „set null“ – damit wird das Löschen im Warenkorb auch weitergeleitet und Null ermöglicht – Beachte: in der Tabelle „korb“ muss der Eintrag „p_id“ eine „NULL“ zulassen.
    - On update: „cascade“ – damit wird die „p_id“ im Warenkorb aktualisiert.

Dadurch erhält „p_id“ auch ein graues Schlüsselsymbol, weil es ja jetzt ein Fremdschlüssel ist. Aber da es schon so ein Symbol hat, sieht man das zweite nicht. ☺

Kontrolle:
Mit dem Designer betrachtet: ABER nur wenn man nicht in einer Tabelle ist, sondern in der Datenbankansicht -> Oben in der Navigation auf „Mehr“ und dann auf „Designer“



Zu sehen in der Verbindung in grüner Farbe.



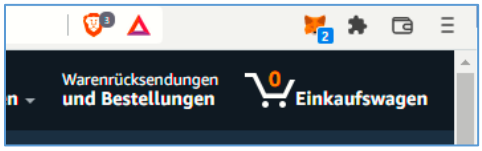
Was kann zu Problemen führen?

- Wenn die Namen vom Fremdschlüssel (p_id) nicht mit dem Namen des Primärschlüssels (p_id) übereinstimmen.
- Wenn der Typ nicht gleich ist – hier in beiden „int(11)
- Wenn im Fremdschlüssel schon Daten vorhanden sind. Dann sollte man diese löschen, und danach nochmals eingeben

3)Navbar ändern – Warenkorb

Neben Login soll der Warenkorb anklickbar sein, dafür benötigt es dort das Wort „Warenkorb“. Unmittelbar daneben soll in Klammern erkennbar sein, ob sich Waren darin befinden oder nicht.

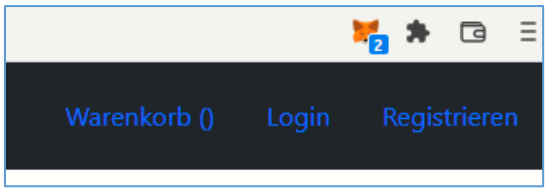
Beispiel mit Bild (das folgt später) von amazon:



Öffne die Site „navbar.php“. Kopiere den Code von „Login“, füge ihn darüber ein und ändere es zu „Warenkorb ()“.
In die Klammern kommt dann die Anzahl der Produkte.

```
18          </ul>
19          <form class="d-flex">
20            <a class="nav-link">Warenkorb ()</a>
21          </form>
22          <form class="d-flex">
23            <a class="nav-link" href="templates/login.php
```

Ergebnis:



4)Anzahl der Produkte im Warenkorb anzeigen – mit einer function()

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

Dafür benötigt man

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

4a)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

```
<form class="d-flex">
  <a class="nav-link">Warenkorb (<?php echo $korbZahl ?>)</a>
</form>
```

Besser und kürzer ist die moderne Schreibweise für das „echo“

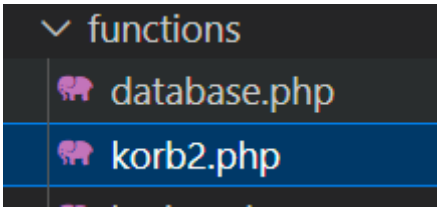
WICHTIG: Kein Leerzeichen zwischen ? und dem =
Dies ist nämlich die Kurzschreibweise von <?php echo

```
<a class="nav-link">Warenkorb (<?= $korbZahl ?>)</a>
```

4b)korb.php erstellen und mit „shop.php“ verbinden

Damit diese Variable auch benutzt werden kann, muss man in der Tabelle „korb“ die IDs holen, die für den angemeldeten User herumliegen.

Damit man diese SELECT nicht hier in der „navbar.php“ schreiben muss, erstelle eine neue Datei „korb2.php“ im Ordner „functions“.



Diese PHP-Datei kann ganz oben in der Navbar eingegliedert werden, aber da die „index.php“ schon die „navbar.php“ eingliedert (mit require) ist es sinnvoller (da auch dort schon PHP verwendet wird) die „shop.php“ zum Einbinden benutzten.
Damit ist die Variable „$korbZahl“ dann auch greifbar.

Öffne die „shop.php“.

Im shop.php: Füge vor dem Ende der PHP die Verbindung zum „korb2.php“ ein.

```php
$result = $dbh->query($sql);

require "functions/korb2.php";
?>

<!DOCTYPE html>
```

Damit es auch in der index.php funktioniert, muss man es auch dort einfügen

```php
<?php
require "functions/korb2.php";
require "templates/navbar.php";
?>
```

4c)SQL für die Anzahl im Korb mit „count“

- Erstelle eine Funktion mit dem Namen „countProdukte“ die das Argument des aktuellen Users bekommen. Hier nehmen wir den ersten User, vorerst.
- Mit SELECT wird die id geholt und mit COUNT diese gezählt
- aus der Tabelle „korb“ wobei dort die u_id benötigt wird:

#	Name	Typ
1	u_id	int(11)
2	email	varchar(50)

- ganz am Beginn muss eine Verbindung mit der „database.php“ hergestellt werden
- am Beginn wird die Variable „$korbZahl“ auf Null gesetzt, weil es ja noch keine Einträge gibt.

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

SQL erzeugen mit count. Da wir noch keine „userId“ haben, wähle einfach den User 1.
Am Beginn muss man noch die Datenbank und auch gleich die user.php mit einbinden.

navbar.php | korb2.php | user.php

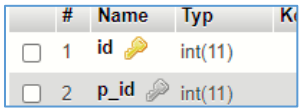
functions > korb2.php

```php
<?php
require_once __DIR__.'/database.php';
require_once __DIR__.'/user.php';

$userId = 1;

function countProdukte(int $userId){
    require "database.php";
    $korbZahl = 0;
    $sql ="SELECT COUNT(id) FROM korb WHERE u_id =" .$userId;
```

Die Funktion COUNT wird auf die „id“ angewendet, die in der Tabelle „korb“ liegt:



#	Name	Typ	K
1	id	int(11)	
2	p_id	int(11)	

Dann wird eine neue Variable erzeugt, die hier „$korbVorZahl“ heißen soll, die das Array aus der Datenbank holt.

```
10      $sql ="SELECT COUNT(id) FROM korb WHERE u_id =" .$userId;
11      $korbVorZahl = $dbh->query($sql);
```

Darunter wird nun die ursprüngliche Variabel $korbZahl, die ja mit Null begonnen hat, überschrieben, da die $korbVorZahl 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.
Diese Funktion muss das Ergebnis noch zurückgeben:

```
11      $korbVorZahl = $dbh->query($sql);
12      $korbZahl = $korbVorZahl->fetchColumn();
13      return $korbZahl;
14  }
15
```

Code:

```
function countProdukte(int $userId){
   require "database.php";
   $korbZahl = 0;
   $sql ="SELECT COUNT(id) FROM korb WHERE u_id =" .$userId;
   $korbVorZahl = $dbh->query($sql);
   $korbZahl = $korbVorZahl->fetchColumn();
   return $korbZahl;
}
```

4d) Funktion aufrufen

Die gerade erstellte Funktion muss nun, am besten gleich darunter, aufgerufen werden:

```
14  }
15
16  $korbZahl = countProdukte($userId);
```

5)Produkte in Warenkorb legen

5.1) Warenkorb Einträge vorbereiten

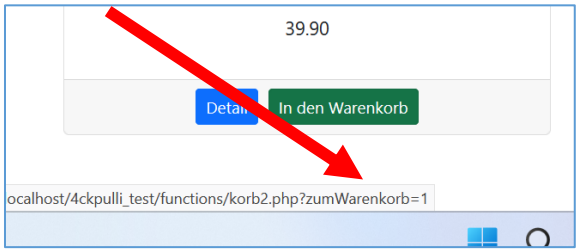
card.php

Hier wird mit einer GET-Variablen gearbeitet, um die Daten zu übergeben.
Dafür muss in der „card.php“ beim Button „Warenkorb“ erst einmal ein Link gesetzt werden.
Beachte: Die ID wird in PHP-Format der URL mitgegeben.

```
21              <!-- Abstand -->
22          <a href="functions/korb2.php?zumWarenkorb=<?= $row['p_id']?>">
23          <button class="btn btn-success btn-sm">In den Warenkorb</button></a>
24      </div>
```

- es wird auf die Datei „korb2.php“ geleitet und
- die Information angehängt, welches Produkt da gespeichert werden soll, nämlich das Produkt mit der angeklickten ID (p_id)

Ergebnis:
Wenn man mit der Maus über den Button „In den Warenkorb“ drüberfährt, wird links unten im Eck der Pfad sichtbar, die von der „row[‚id‘]“ stammt und hier angezeigt wird. Somit kann man die Nummer sehen, hier das Produkt 1:



Öffne die Datei „korb2.php“:

Hier wird die Speicherung in die Datenbank vorgenommen.

a)Den Klick aus der „card.php“ verarbeiten und die „p_id“ mit der GET übergeben:

Dies geschieht in einer IF, die überprüft, ob es eine GET-Information gibt. In dieser GET Info wird die Variable „productId“ geholt, die man ja dann benötigt: sie kommt aus der „card.php“ als übergebene „p_id“:

```
<a href="functions/korb2.php?zumWarenkorb=<?= $row['p_id']?>">
```

Daher folgender Code, der die „p_id“ in die „$productId“ überführt:

```
16  $korbZahl = countProdukte($userId);
17
18  if (isset($_GET['zumWarenkorb'])) {
19
20      $productId = $_GET['zumWarenkorb'];
21
22  }
```

Nun soll die Funktion aufgerufen werden, die wir gleich im Anschluss darunter erstellen werden, die nämlich die „productId“ benötigen wird. Sie wird heißen: „addProductToKorb“:

```php
if (isset($_GET['zumWarenkorb'])) {

    $productId = $_GET['zumWarenkorb'];
    addProductToKorb($userId, $productId);

}
```

Danach ein „Redirect“ zur Shop-Seite – man soll ja auf der Shop-Seite bleiben

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

header("Location: /4ckpulli_test/shop.php ");
exit();

```php
    addProductToKorb($userId, $productId);

    header("Location: /4ckpulli_test/shop.php");
    exit();
}
```

b)Funktion „addProductToKorb“ erstellen:

Nun wird wieder eine Funktion erzeugt.

```php
// zum Warenkorb hinzufügen
function addProductToKorb($userId, $productId)
{
```

Damit auch der User und das Produkt übergeben wird, sind diese beiden Argumente wichtig hier anzugeben.

INFO: Sicherheit

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 die Verbindung zur Datenbank herstellen, innerhalb der Funktion.

Zuerst muss man aber eine neue SQL-Query erzeugen mit „INSERT INTO“. Man kann, wie hier, statt „value“ auch „set“ verwenden.

- Die user-ID wird noch nicht betrachtet und daher mit User = 1 festgelegt. Diese kommt dann später dazu, wenn wer eigeloggt ist.
- Die Anzahl soll hier auch fix vergeben werden mit 1.

```
29  {
30      require "database.php";
31      $sql = "INSERT INTO korb SET
32              p_id = :productId,
33              u_id = :userId,
34              anzahl=1";
```

require "database.php";
$sql = "INSERT INTO korb SET
p_id = :productId,
u_id = :userId,
anzahl=1";

Es werden die u_id und die p_id 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.

```
35
36      $statement = $dbh->prepare($sql);
37
38      $statement->execute([
39       ':productId'=> $productId,
40       ':userId'=> $userId
41      ]);
42  }
```

$statement = $dbh->prepare($sql);

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

In Zeile 31 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: productId mit der Variablen „$productId“.

ABER: damit keine Fehlermeldung kommt, wenn man ein zweites Mahl auf den „In den Warenkorb“-Button klickt, muss man mit einer IF das so abfangen, dass der eben erstellte Code nur ausgeführt werden kann, wenn in der Datenbank das Produkt des jeweiligen Nutzers mit der Anzahl < 1 ist, das heißt somit, noch nicht vorhanden ist:
Füge unterhalb des „require“ zuerst

- die Abfrage – hier $sql1 (damit sie nicht mit der darunterliegenden $sql verwechselt werden kann) ein, die die Anzahl aus der Datenbank holt und
- dann die IF

```php
function addProductToKorb($userId, $productId)
{
    require "database.php";

    $sql1 ="SELECT anzahl FROM korb WHERE
    u_id =" .$userId." AND
    p_id =" .$productId;
    $row = $dbh->query($sql1);
    $anzahl = $row->fetchColumn();

    if($anzahl < 1){

    $sql = "INSERT INTO korb SET
```

Code:

```
$sql1 ="SELECT anzahl FROM korb WHERE
   u_id =" .$userId." AND
   p_id =" .$productId;
   $row = $dbh->query($sql1);
   $anzahl = $row->fetchColumn();

   if($anzahl < 1){
```

- nicht vergessen, dann am Ende die geschwungene Klammer der IF-Abfrage zu beenden:

```php

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

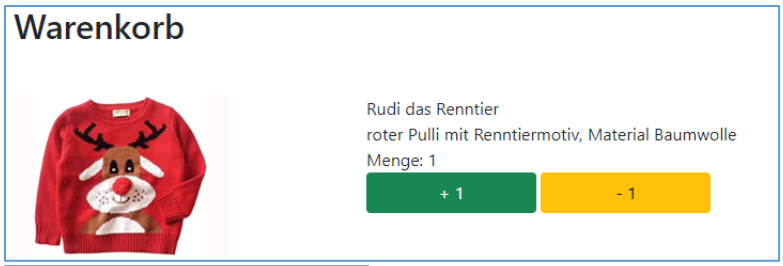
Ergebnis:

Der Warenkorb füllt sich

INFO

Ein Produkt kann **nur einmal** angelegt werden, weil wir in der Datenbank die „p_id“ auf „UNIQUE“ gestellt hatten.

Grund: Um zu vermeiden, dass ein Produkt mehrfach in den Warenkorb hinzugefügt – die Anzahl kann jederzeit danach noch erhöht werden, aber es soll nicht im Warenkorb mehrfach untereinanderstehen, sondern lieber einmal und daneben soll man dann mit einem Button die Anzahl erhöhen können.



#	Name	Typ	Kollation	At
1	id	int(11)		
2	p_id	int(11)		
3	u_id	int(11)		
4	anzahl	int(11)		
5	created	timestamp		

In der Datenbank wurde das Produkt aufgenommen:

	id	p_id	u_id	anzahl	created
Bearbeiten Kopieren Löschen	1	1	0	1	2022-10-22 15:25:36
Bearbeiten Kopieren Löschen	2	2	0	1	2022-10-22 15:25:37
Bearbeiten Kopieren Löschen	3	3	0	1	2022-10-22 15:25:39