Webshop mit PHP – PDO, Datenbank, 2022

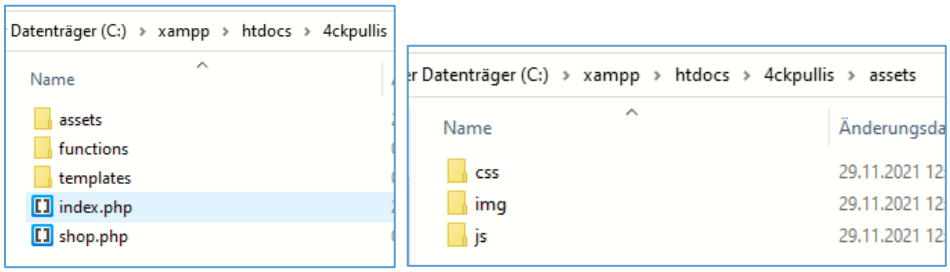
Inhalt:

1)Ordnerstruktur anlegen, Bootstrap einbinden

Erstelle in XAMPP / htdocs einen Ordner namens „4ckpulllis“ und darin eine „index.php“.

Bzw. nutze den Ordner weiter, der die „landigpage“ beinhaltet.

In diese soll Bootstrap eingebunden werden, aber als Download-Dateien mit „js-“ und „css-Ordnern“. Diese kommen in den allgemeinen „assets“-Ordner.



Kopiere das Startertemplate von Bootstrap in die index-Datei und stelle die CDN auf Lokal um.

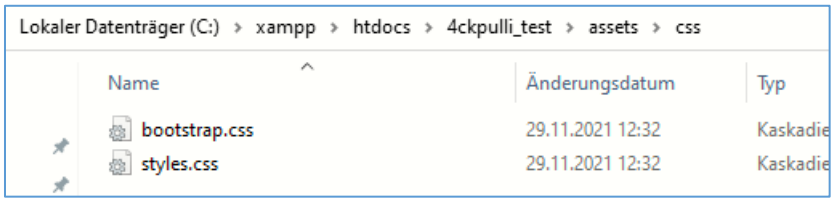
index.php

```
1   <!DOCTYPE html>
2   <html lang="de">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Orangenshop</title>
8       <link rel="stylesheet" href="assets/css/bootstrap.css">
9       <link rel="stylesheet" href="assets/css/styles.css">
10  </head>
11  <body>
12
13  <script src="assets/js/bootstrap.js"></script>
14  </body>
15  </html>
```

<!DOCTYPE html>
<html lang="de">

```
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Orangenshop</title>
   <link rel="stylesheet" href="assets/css/bootstrap.css">
   <link rel="stylesheet" href="assets/css/styles.css">
</head>
<body>

<script src="assets/js/bootstrap.js"></script>
</body>
</html>
```

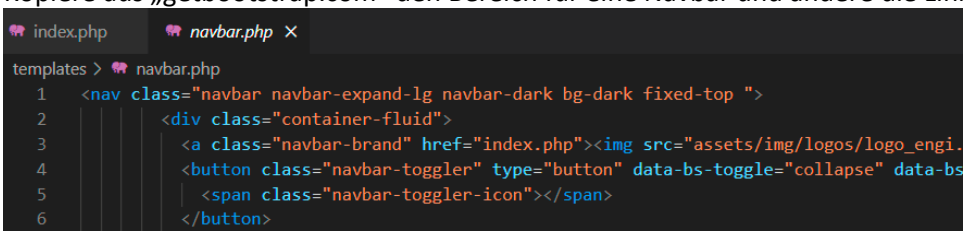
Erstelle im CSS-Ordner eine leere „style.css“ und verknüpfe sie unter der Bootstrap-CSS in der index-Datei. Sie muss danach stehen, damit sie private Änderungen nutzen kann und somit Bootstrap überschreiben kann.




```
10      <link rel="stylesheet" href="assets/css/bootstrap.css">
11      <link rel="stylesheet" href="assets/css/styles.css">
12      <title>Orangenshop</title>
```

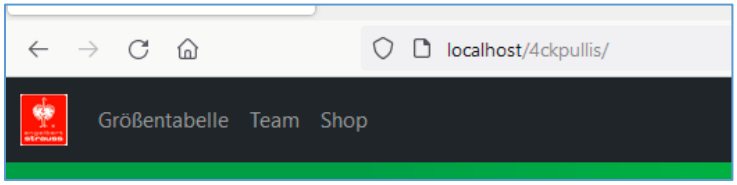
2)erstelle eine „navbar.php“ im Ordner „templates“

Kopiere aus „getbootstrap.com“ den Bereich für eine Navbar und ändere die Links.



Das Grundgerüst hole aus der Site „getbootstrap.com“ aus dem Bereich „docs“ und „navbar“.

Folgende Links sollen verwendet werden:



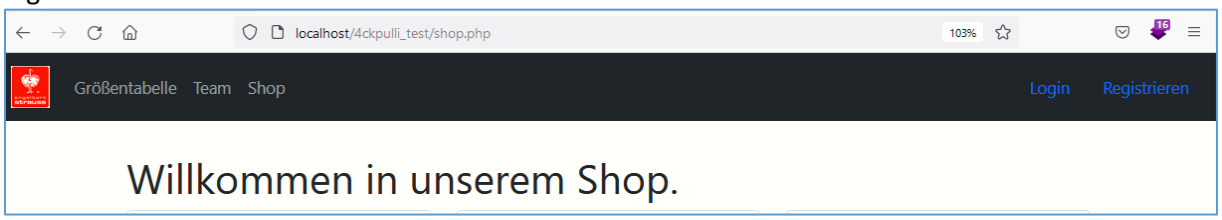
templates > navbar.php

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top ">
    <div class="container-fluid">
      <a class="navbar-brand" href="index.php"><img src="assets/img/logos/logo_engi.jpg" height="40" width="auto"></a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" a
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item">
            <a class="nav-link" href="#">Größentabelle</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Team</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="shop.php">Shop</a>
          </li>
        </ul>
```

Zusätzlich soll auch gleich auf der rechten Seite ein Link zum „Login“ erstellt werden und gleich darunter auch das „Registrieren“.

```
<!-- zweite ul -->
<ul class="navbar-nav ms-auto">
    <li>
        <a class="nav-link" href="templates/login.php">Login</a>
    </li>
    <li>
        <a class="nav-link" href="templates/registrieren.php">Registrieren</a>
    </li>
</ul>
```

Ergebnis:



3)erstelle eine „shop.php“ neben der index.php

Erstelle ganz oben in PHP den Code für ein sinnvolles Fehler-Reporting, damit bei Fehlern sinnvolle Anzeigen gesendet werden.

```php
<?php
error_reporting(-1);
ini_set('display_errors','On');
?>
<!DOCTYPE html>
<html lang="de">
```

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

Einbinden der Navbar mit Hilfe von PHP und „require“.

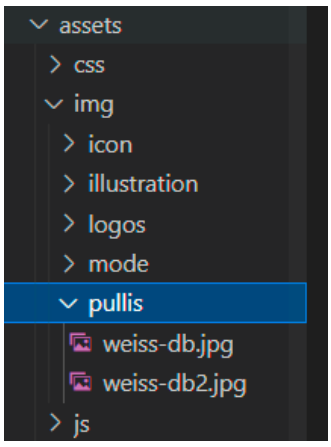
```php
<body>
  <?php
  require "templates/navbar.php";
  ?>

    <br><br><br><br>

<div class="container">
  <h1>Willkommen in unserem Shop.</h1>
```

Die „id“ wird später in der CSS „styles.css“ angesprochen und gestylt.

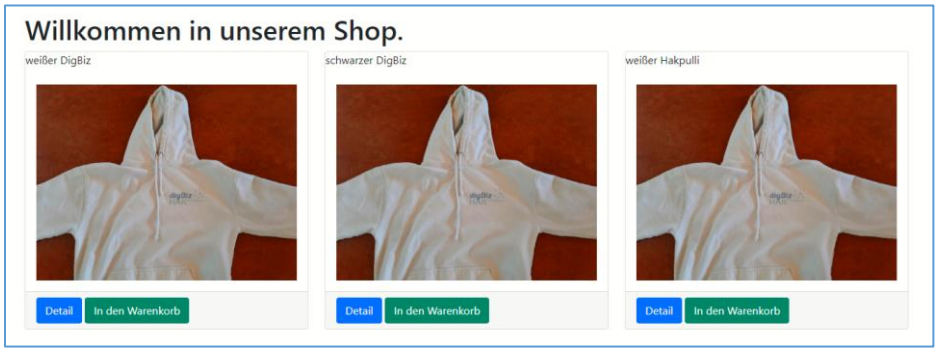
Unter der <h1> eine neue „row“ mit 3 Cards in jeweils einer „col“, die dann die Bilder mit den Bestellbuttons beinhalten soll:

shop.php

```php
  <h1>Willkommen in unserem Shop.</h1>

  <div class="row">
    <div class="col">
        <div class="card">

        </div>
    </div>
    <div class="col">
        <div class="card">

        </div>
    </div>
    <div class="col">
        <div class="card">

        </div>
    </div>
  </div>
```



Nun noch ohne Zugriff auf die Datenbank die 3 Bilder, Titel und jeweils 2 Buttons: die Bilder kommen noch aus dem Ordner „assets/img/pullis“ – wir haben versucht eine gute Ordnung im Bereich der Bilder zu schaffen.



Im <body> der Card soll das Bild eingefügt werden, im Footer dann 2 Buttons erstellt werden:

```
<div class="card-footer">
    <a href="#"><button class="btn btn-primary">Detail</button></a>
    <a href="#"><button class="btn btn-success">In den Warenkorb</button></a>
</div>
```

Hier der Code für eine Card:

```
29      <div class="card">
30          <div class="card-title">weißer DigBiz</div>
31          <div class="card-body">
32              <img src="assets/img/pullis/weiss-db.jpg" class="img-fluid">
33          </div>
34          <div class="card-footer">
35              <a href="#"><button class="btn btn-primary">Detail</button></a>
36              <a href="#"><button class="btn btn-success">In den Warenkorb</button></a>
37          </div>
38      </div>
```

```
<div class="card-title">weißer DigBiz</div>
        <div class="card-body">
          <img src="assets/img/pullis/weiss-db.jpg" class="img-fluid">
        </div>
        <div class="card-footer">
          <a href="#"><button class="btn btn-primary">Detail</button></a>
          <a href="#"><button class="btn btn-success">In den Warenkorb</button></a>
        </div>
</div>
```

Design: Zur Verbesserung kann man beide Buttons nebeneinander platzieren. Dafür erhalten sie die zusätzliche Klasse „small“ = btn-sm:

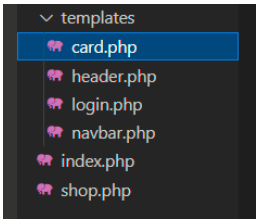
btn-sm

```
<a href="#"><button class="btn btn-primary btn-sm">Detail</button></a>
<a href="#"><button class="btn btn-success btn-sm">In den Warenkorb</but
```

4)Cards in PHP auslagern – erstelle eine „card.php“ im Ordner templates

Dieser Cards-Code soll nun in eine neue „card.php“ ausgelagert werden. Diese wird später die Daten aus der Datenbank aufnehmen.
Statt dem Code, der aus den <cols> kopiert wurde, wird nun mit „require“ auf diese card.php zugegriffen.

```
24  <div class="container">
25    <h1>Willkommen in unserem Shop.</h1>
26
27    <div class="row">
28        <?php
29        require "templates/card.php";
30        ?>
31    </div> <!--ende row-->
32  </div>
33  </body>
34  </html>
```



In der card.php sind nun alle drei „col“ drinnen:

```
shop.php   card.php
templates > card.php
1     <div class="col">
2         <div class="card">
3             <div class="card-title">weißer DigBiz</div>
```

Soll der Titel und das Bild im „body“ zentriert sein muss man in die entsprechende „class“ diese Klasse hinzufügen:

d-flex justify-content-center

```
2     <div class="card ">
3         <div class="card-title d-flex justify-content-center">weißer
```

5)Login erstellen

Erstelle im Ordner „templates“ die Datei „login.php“.
Darin ist die Grundstruktur gleich wie in der „shop.php“ wodurch du die <html>-Struktur übernehmen kannst.

Füge auch gleich die Verbindung zur „navbar.php“ her:

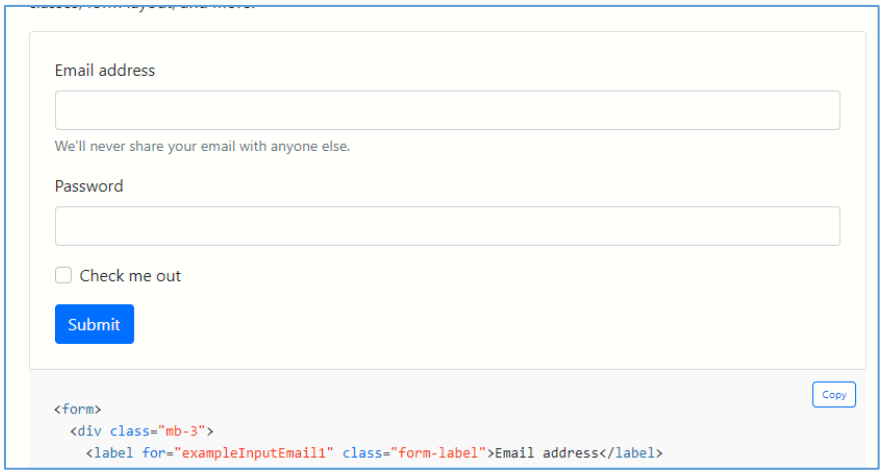
shop.php | card.php | login.php

templates > login.php

```
1   <!DOCTYPE html>
2   <html lang="de">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Shop Pullis</title>
8       <link rel="stylesheet" href="../assets/css/bootstrap.css">
9       <link rel="stylesheet" href="../assets/css/styles.css">
10  </head>
11  <body>
12    <?php
13    require "navbar.php";
14    ?>
```

```
15
16  <br><br><br><br>
17
18  <div class="container">
19
20  </div> <!--ende container-->
21
22  </body>
23  </html>
```

Im Container soll nun die HTML-Struktur für ein Formular eingefügt werden. Dafür gibt es bei „getbootstrap.com“ im Bereich „forms“ eine gute Vorlage. Diese kann kopiert und passend geändert werden:



Natürlich muss das Formular für die Benutzereingabe die erforderlichen Elemente enthalten:

- <form> Tag – auf und zu, dazwischen
- Submit-Button
- Input-Felder
- die Methode „post“

- eine action, damit man weiß, wohin nach dem Drücken verlinkt wird – dies geht hier auf die „login2.php“, die im Ordner „functions“ liegen soll. Dort sind die PHP-Dateien geparkt, die im Hintergrund die Abfragen usw. erstellen.
- Die method und action muss ganz oben im „form-Tag“ stehen.

Info: damit der Text in Großbuchstaben geschrieben wird, kann man die Klasse „text-uppercase“ nutzen:

```
<label for="email" class="form-label text-uppercase">E-Mail</label>
```

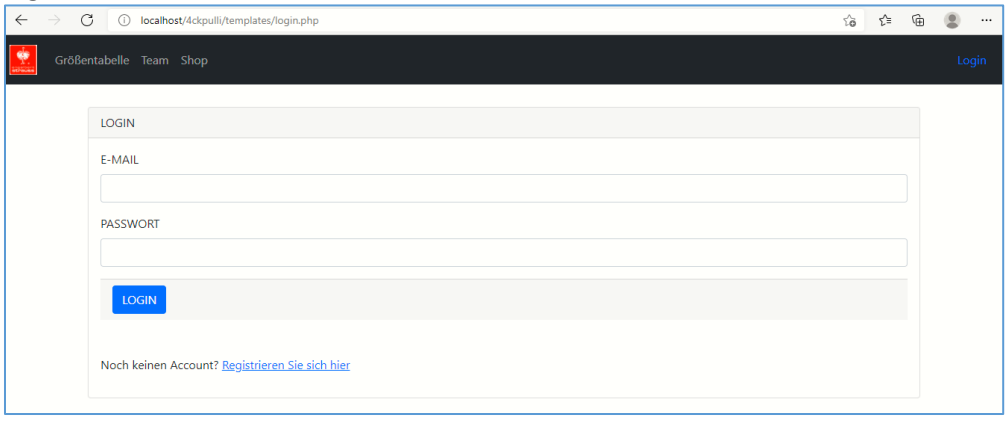
```
19  <div class="container">
20  <form method="post" action="../functions/login2.php">
21  <div class="card">
22    <div class="card-header">
23    LOGIN
24    </div>
```

```
25    <div class="card-body">
26      <div class="mb-3">
27      <label for="email" class="form-label text-uppercase">E-Mail</label>
28      <input type="email" class="form-control" id="email" name="email">
29    </div>
30    <div class="mb-3">
31      <label for="password" class="form-label text-uppercase"> Passwort</label>
32      <input type="password" class="form-control" id="password" name="password">
33    </div>
34
35    <div class="card-footer">
36      <button type="submit" class="btn btn-primary text-uppercase">Login</button>
37      <br>
38    </div>
```

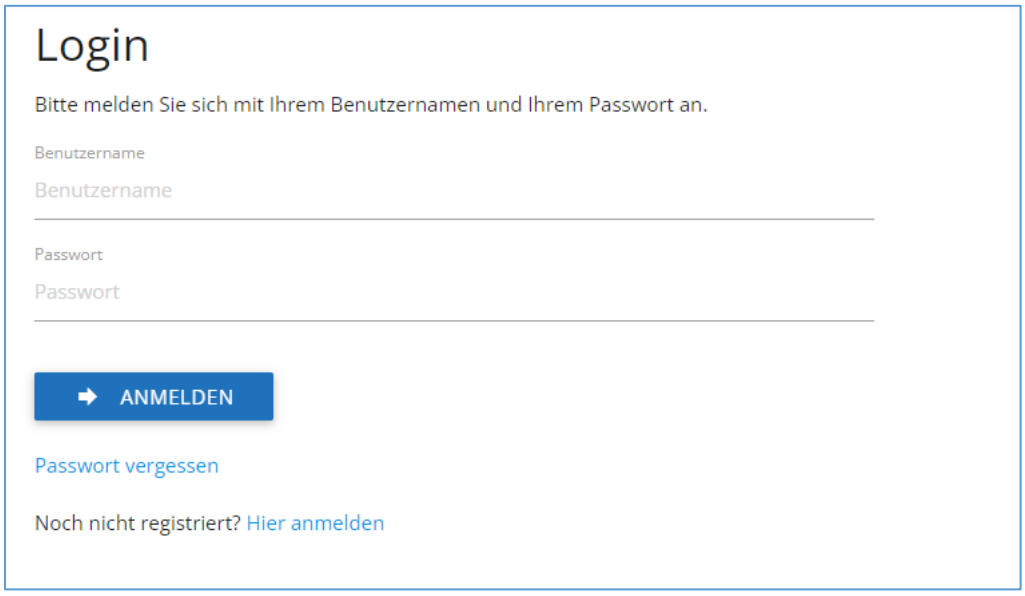
Ganz unten, noch in der Form kann man auch den Link zum Registrieren anführen:

```
39
40    <br><br>
41    <p>Noch keinen Account?
42      <a href="registrieren.php">Registrieren Sie sich hier</a></p>
43  </div>
44  </form>
45  </div> <!--ende container-->
```

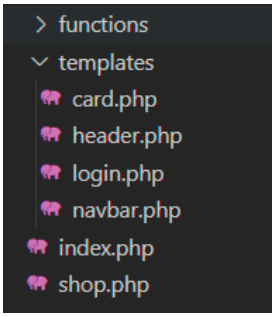
Ergebnis:



Schön wäre auch:



6)Base href



Problem: Da nun die „index.php“ und die „shop.php“ ganz außen liegen und die „navbar.php“ aber im Ordner „templates“ kommt es mit den anderen Dateien, die ebenfalls auf die „navbar.php“ zugreifen, aber im Ordner liegen, zu Problemen.
Die Links sind dann nicht passend.

Lösung: base href im Bereich <head>

```
<base href="/4ckpullis/">
```

Dadurch gilt dann die Basis ist immer die Ebene von „index.php“.

- Die Basis ist der übergeordnete Ordner „4ckpullis“
- Dieser muss in 2 „slash“ eingepackt angeführt sein, nämlich in allen Dateien, die im „templates“ Ordner liegen: <base href="/4ckpullis/">
- Nicht jedoch in der „navbar.php“ selbst und auch nicht in der „header.php“
- Und auch nicht in der „index.php“ oder „shop.php“

```
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Shop Pullis</title>
8      <base href="/4ckpulli/">
9      <link rel="stylesheet" href="assets/css/bootstrap.css">
10     <link rel="stylesheet" href="assets/css/styles.css">
11 </head>
```

Es kommt aber sogar auf die Position im „head“ an, denn ab der <base href> werden alle folgenden Links bereits davon abhängig:

- **VOR den Links ist anders als NACH den Links für die CSS usw., daher ist es vielleicht nötig die Punkte vor einem Link zu entfernen:**
  ../css/bootstrap.css -> css/bootstrap.css

Info:

- Das <base>Tag gibt die Basis-URL und/oder das Ziel für alle relativen URLs in einem Dokument an.
- Im <base>Tag muss entweder ein href- oder ein Zielattribut vorhanden sein oder beides.
- Es kann nur ein einzelnes <base>Element in einem Dokument geben und es muss sich innerhalb des <head>-Elements befinden.
- Die URL wird immer um das Verzeichnis aus dem href-Attribut des base-Tags ergänzt

7)Verbindung zur Datenbank

2 Möglichkeiten:

- mysqli – haben wir bis jetzt verwendet
- PDO - ist nicht nur auf mysql-Datenbanken fixiert

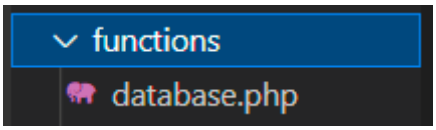
- **MySQLi** i steht für MySQL Improved Extension und ist die neuere Entwicklung. Sie erlaubt zwei Arten der Programmierung: prozedual und objektorientiert. Außerdem bietet sie attraktive Features wie Prepared Statements.
- **PDO** steht für „PHP Data Objects“ und ist nicht nur für MySQL ausgelegt. Diese Schnittstelle ist zwar leistungsfähiger, aber dafür auch etwas komplizierter als mysqli.

Die PDO Schnittstelle erwartet 3 Parameter:

- dsn
- username
- password

Um den Code sauber vom HTML-Code in der „index.php“ zu trennen soll eine eigene Datei erstellt werden, die den Zugang zur Datenbank beinhaltet.

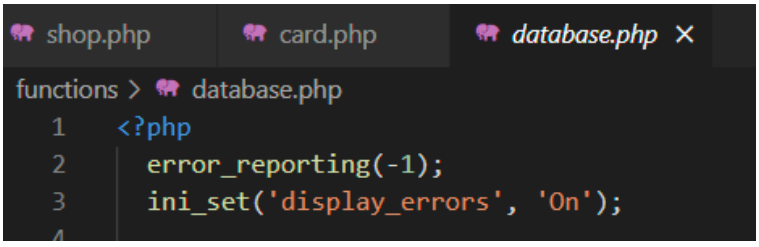
- Erstelle im Ordner „functions“ eine neue Datei „database.php“



Lege in der database.php folgende Zeilen an:

- in Xampp ist der user immer „root“,
- das Passwort leer und der
- host ist localhost.
- Der Name der Datenbank kann frei gewählt werden und muss mit dem Namen der Datenbank natürlich übereinstimmen: diese wird anschließend erstellt.

Am Beginn dieser Datei soll auch wieder die Möglichkeit geboten werden, dass ein Fehler gut dokumentiert angezeigt werden soll:




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

Darunter nun die Verbindung:

```
$host_name = 'localhost';
$database = '4ckpullis';
$user_name = 'root';
$password = '';
```

```
4
5   $host_name = 'localhost';
6   $database = '4ckpullis';
7   $user_name = 'root';
8   $password = '';
9
```

Darunter die echte Verbindung durch „new PDO“:

```
10    $dbh = null;
11
12    try {
13      $dbh = new PDO("mysql:host=$host_name; dbname=$database;", $user_name, $password);
14    } catch (PDOException $e) {
15      echo "Fehler!: " . $e->getMessage() . "<br/>";
16      die();
17    }
18  ?>
```

```
$dbh = null;

 try {
  $dbh = new PDO("mysql:host=$host_name; dbname=$database;", $user_name, $password);
 } catch (PDOException $e) {
  echo "Fehler!: " . $e->getMessage() . "<br/>";
  die();
 }
?>
```

shop.php:

Nun wird diese Datenbankverbindung in der shop.php benutzt und dann die Abfragen erstellt.

Füge in der bereits bestehenden Datei „shop.php“ ganz oben, vor dem HTML-Code, ein:

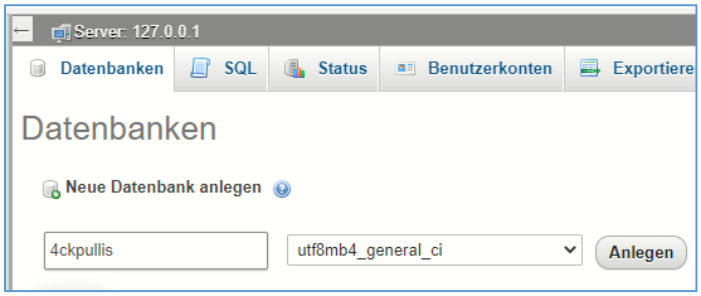
```
<?php
 error_reporting(-1);
 ini_set('display_errors', 'On');

require_once 'functions/database.php';    //damit wird die Verbindung hergestellt
```

shop.php

```
1   <?php
2     error_reporting(-1);
3     ini_set('display_errors', 'On');
4
5   require_once 'functions/database.php';   //damit wird die Verbindung hergestellt
6   ?>
7
8   <!doctype html>
```

8)Datenbank erstellen in PhpMyAdmin

Diese wird jetzt erstellt, mittels PhpMyAdmin in XAMPP:



a)Tabelle „produkte“ erstellen

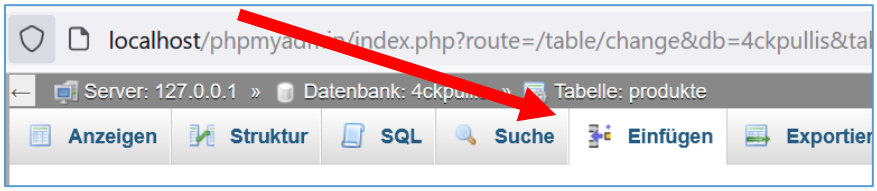
Nun erfolgt trotzdem noch die Erstellung der ersten Tabelle. Diese trägt den Namen „produkte“ und hat sechs Spalten:
Preis: hier wird Typ DECIMAL gewählt. Bei der Länge wird 5,2 eingegeben. D.h. dass der Betrag nicht länger als 5 Zeichen (der Dezimalpunkt nicht mitgerechnet) sein kann und dass es zwei Stellen hinter dem Dezimalpunkt gibt. Somit ist der höchste Preis bei 999,99 Euro.

Achtung: Hier muss ein BEISTRICH zwischen den Zahlen stehen (5,2) und kein Punkt. Der Punkt ist dann beim Befüllen zu verwenden.

Bild: hier wird „mediumblob“ ausgewählt

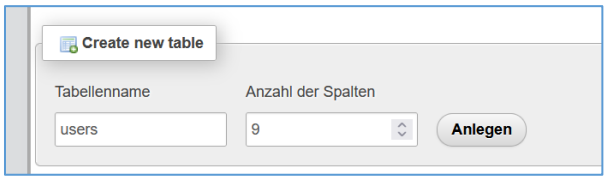
#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	**p_id**	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten
2	**name**	text	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
3	**beschreibung**	text	utf8_general_ci		Nein	*kein(e)*			Bearbeiten
4	**farbe**	text	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
5	**preis**	decimal(5,2)			Nein	*kein(e)*			Bearbeiten
6	**anzahl**	int(11)			Nein	*kein(e)*			Bearbeiten
7	**bild**	mediumblob			Ja	*NULL*			Bearbeiten

Zum Beginn sollen hier gleich die ersten Datensätze eingegeben werden:
Klicke auf „Einfügen“ und gib ein:



p_id	name	beschreibung	farbe	preis	bild
1	Flammen Hoodie	Stylischer Hoodie aus Baumwolle mit Flammenmotiv	bunt	39.90	[BLOB - 62,9 KiB]
2	Rudi das Renntier	roter Pulli mit Renntiermotiv, Material Baumwolle	rot	40.00	[BLOB - 56,8 KiB]
3	Schafpulli Dolli	Pulli aus echter Schafwolle	grau	59.90	[BLOB - 63,3 KiB]

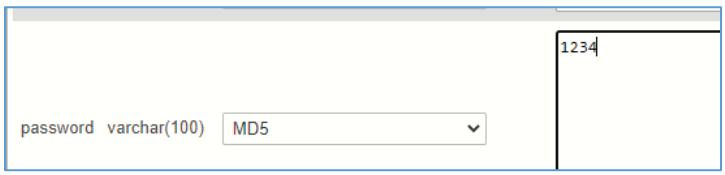
b)Tabelle „users“ erstellen



Diese soll folgende Parameter aufweisen:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	**u_id**	int(11)			Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten
2	**email**	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
3	**password**	varchar(100)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
4	**vorname**	text	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
5	**nachname**	text	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
6	**plz**	int(5)			Nein	*kein(e)*			Bearbeiten
7	**ort**	text	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
8	**strasse_nr**	varchar(50)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten
9	**telefon**	varchar(20)	utf8mb4_general_ci		Nein	*kein(e)*			Bearbeiten

Lege mithilfe des Buttons „Einfügen“ einen Benutzer an. Beim „password“ nutze die Möglichkeit, dieses verschlüsselt zu speichern, nämlich mit der Methode „MD5“. Im Linken Bereich auf den Pfeil nach unten und wähle „MD5“.



Ergebnis:

u_id	email	password	vorname	nachname	strasse_nr	plz	ort	telefon
1	4ck@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Franz	Berger	Brennerweg 8	2130	Mistelbach	025722305