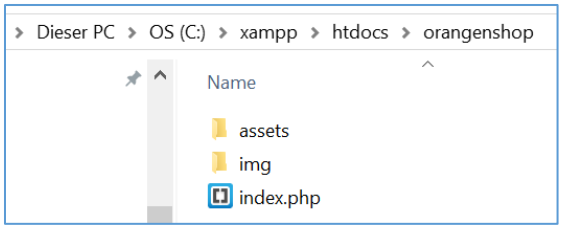
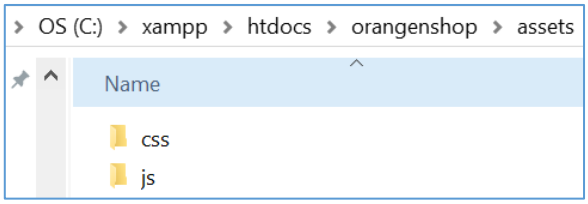
Webshop mit PHP

1)Ordnerstruktur anlegen, Bootstrap einbinden

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

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





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

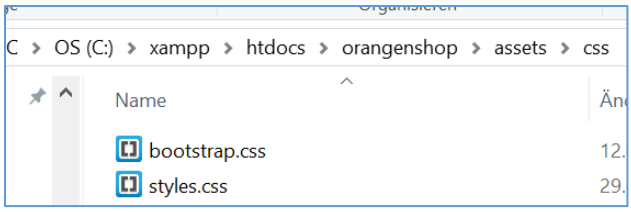
```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="assets/css/bootstrap.css">
    <title>Orangenshop</title>
  </head>
  <body>
    <h1>Webshop für Orangen mit PHP</h1>

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


```
<!doctype html>
<html lang="de">
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link rel="stylesheet" href="assets/css/bootstrap.css">
  <title>Orangenshop</title>
 </head>
 <body>
  <h1>Webshop für Orangen mit PHP</h1>

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


1.2)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)PHP Code

In der „index.php“ erstelle ganz oben in PHP den Code für ein sinnvolles Fehler-Reporting, damit bei Fehlern sinnvolle Anzeigen gesendet werden.

```
1  <?php
2  error_reporting(-1);
3  ini_set('display_errors','On');
4  ?>
5  <!DOCTYPE html>
6  <html lang="de">
```

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

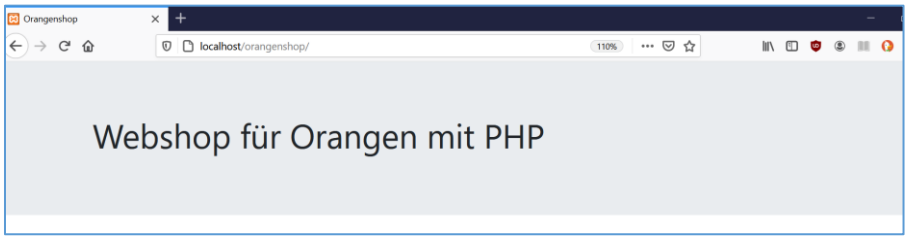
3)Startseite formen

Erstelle einen <header> mit einer H1-Überschrift.

```
14    <body>
15      <header class="jumbotron" >
16      <div class="container" >
17      <h1>Webshop für Orangen mit PHP</h1>
18      </div>
19      </header>
20      <script src="assets/js/bootstrap.js"></script>
21    </body>
```

```
<header class="jumbotron" >
  <div class="container" >
  <h1>Webshop für Orangen mit PHP</h1>
  </div>
  </header>
```

Ergebnis in „localhost“



Darunter kommt eine „section“, in der die Produkte in 4 umrahmten Feldern angezeigt werden sollen. Dafür werden „cards“ aus Bootstrap verwendet werden.
Später werden die Bilder und die Beschreibung mit dem Preis aus der Datenbank geholt werden.
Darunter dann ein Button für den Warenkorb.

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

```
19      </header>
20      <section class="container" id="produkte" >
21
22      </section>
```

Darin dann eine Klasse „card“:

```
20      <section class="container" id="produkte" >
21          <div class="card">
22              <div class="card-body"></div>
23          </div>
24      </section>
```

Da hier 4 Produkte nebeneinander angezeigt werden sollen, muss man dies kopieren und mit <row> und <col> aufbauen; es wird somit eine Art Tabelle erzeugt.

Erzeuge eine <row> und vier <col>:

```
20      <section class="container" id="produkte">
21          <div class="row">
22              <div class="col"></div>
23              <div class="col"></div>
24              <div class="col"></div>
25              <div class="col"></div>
26          </div>
27          <div class="card">
28              <div class="card-body"></div>
29          </div>
30      </section>
```

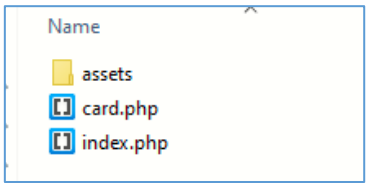
In diese „cols“ werden nun die Card-Konstrukte kopiert, und das somit 4 mal.

```
21        <div class="row">
22            <div class="col">
23                <div class="card">
24                <div class="card-body"></div>
25                </div>
26            </div>
27            <div class="col">
28                <div class="card">
29                <div class="card-body"></div>
30                </div>
31            </div>
32            <div class="col">
33                <div class="card">
34                <div class="card-body"></div>
35                </div>
36            </div>
37            <div class="col">
38                <div class="card">
39                <div class="card-body"></div>
40                </div>
41            </div>
42        </div>
```

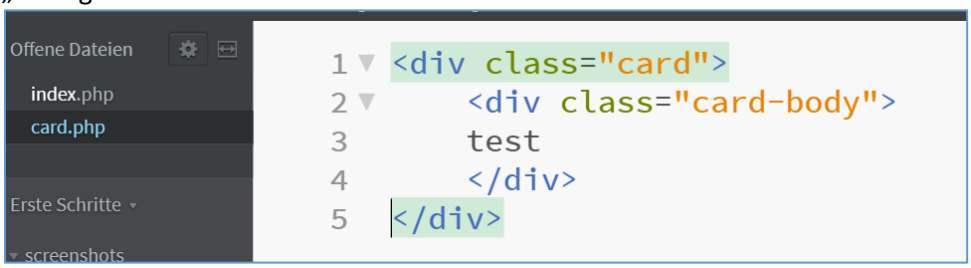
Das insgesamt 4-mal.

4)Cards in PHP auslagern

Dieser 4 fache 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.



Erstelle eine neue „card.php“, neben der 2index.php“ und kopiere den Code dort hinein, es sind nur diese 5 Zeilen, wobei in Zeile 3 „test“ geschrieben wird.



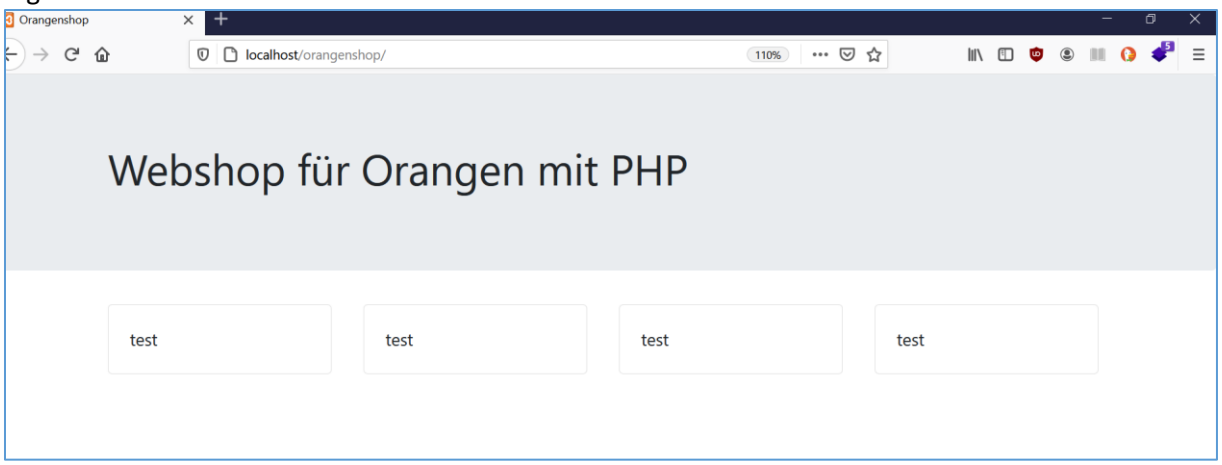
In der „index.php“ muss nun die Einbindung stattfinden. Das passiert mit der Kurzschreibweise des PHP-Tags:

```
21        <div class="row">
22            <div class="col">
23                <?php include 'card.php' ?>
24            </div>
25            <div class="col">
```

Und das natürlich 4-mal.

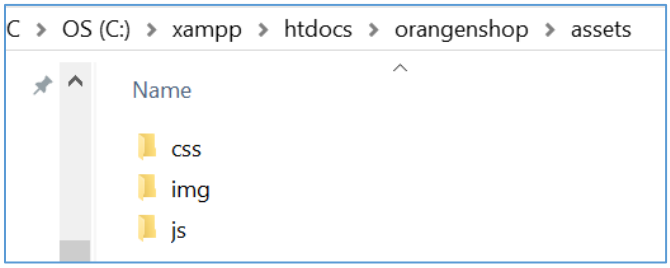
```
21      <div class="row">
22          <div class="col">
23              <?php include 'card.php' ?>
24          </div>
25          <div class="col">
26              <?php include 'card.php' ?>
27          </div>
28          <div class="col">
29              <?php include 'card.php' ?>
30          </div>
31          <div class="col">
32              <?php include 'card.php' ?>
33          </div>
34      </div>
```

Ergebnis:



5)Weitere Codeteile in der „card.php“.

Es soll auch ein Bild eingefügt werden, nämlich aus dem Ordner „assets/img“.



Füge über dem Body das Bild ein:

```
1  <div class="card">
2      <img src="assets/img/orangen.jpg" class="card-img-top" alt="orange">
3      <div class="card-body">
```

Die Klasse fügt das Bild oberhalb ein, da es die Erweiterung „img-top“ hat.

<img src="assets/img/orangen.jpg" class="card-img-top" alt="orange">

Produktbeschreibung:

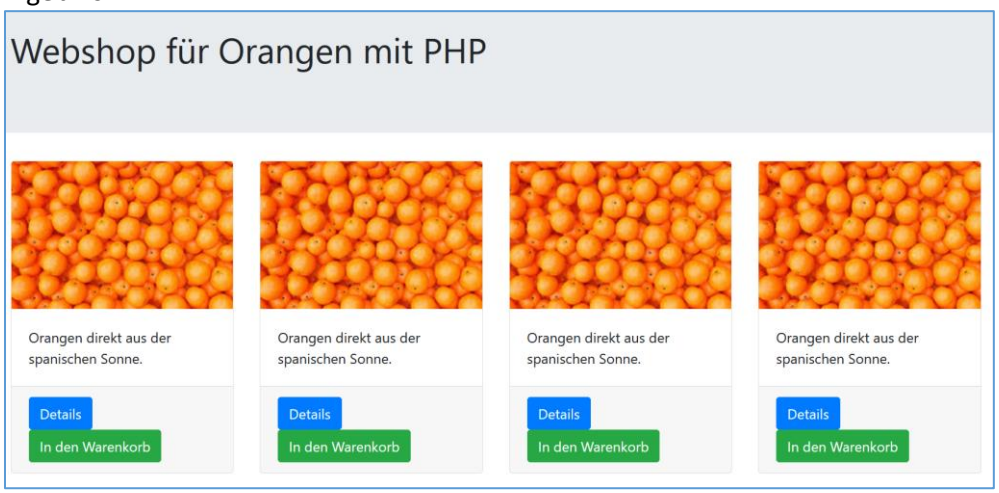
```
<div class="card-body">
Orangen direkt aus der spanischen Sonne.
</div>
```

Darunter erstelle zwei Buttons, für Details und eine Aufnahme in den Warenkorb. Auch hier wird typischer Bootstrap-Code verwendet.

```
    <div class="card-footer">
        <a href="" class="btn btn-primary">Details</a>
        <a href="" class="btn btn-success">In den Watenkorb</a>
    </div>
</div>
```

    Details
    In den Warenkorb

Ergebnis:



Zur Verbesserung kann man beide Buttons neben einander platzieren. Dafür erhalten sie den Code „small“:

```
<a href="" class="btn btn-primary btn-sm">Details</a>
<a href="" class="btn btn-success btn-sm">In den Warenkorb</a>
```

Ergebnis:

6)Verbindung zur Datenbank

2 Möglichkeiten:

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

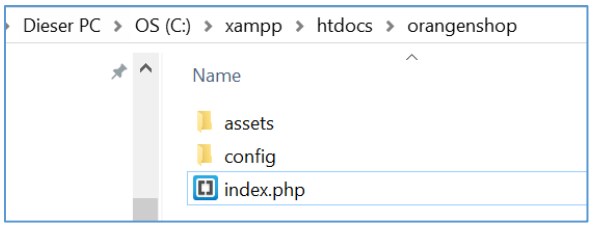
- **MySQLi** i steht für MySQL Improved Extension und ist die neuere Entwicklung. Sie erlaubt zwei Arten der Programmierung: prozedual und objektorientiert. Außerdem bietet sie attraktive Features wie Prepared Statements.
- **PDO** steht für „PHP Data Objects“ und ist nicht nur für MySQL ausgelegt. Diese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

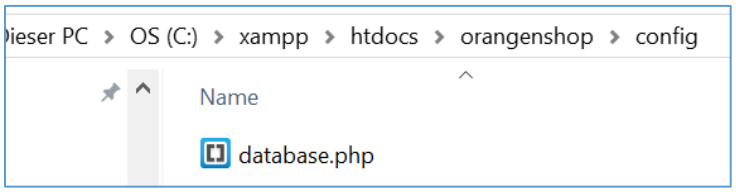
Die PDO Schnittstelle erwartet 3 Parameter:

- 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.

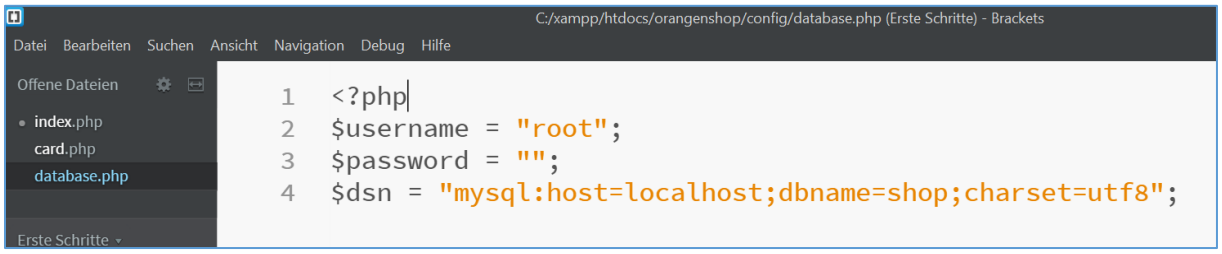
- Erstelle einen neuen Ordner „config“
- Darin eine neue Datei „database.php“





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.



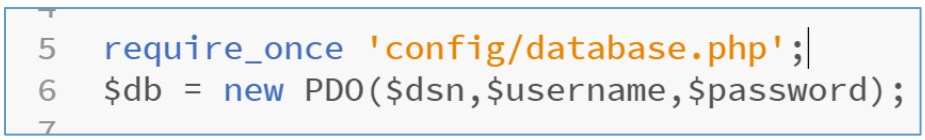
$username = "root";
$password = "";
$dsn = "mysql:host=localhost;dbname=shop;charset=utf8";

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

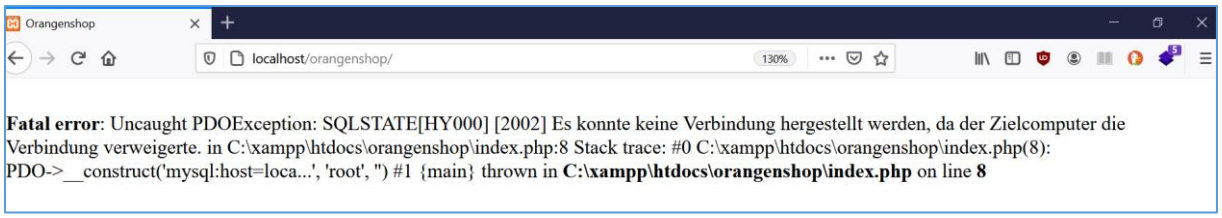
Füge in der „index.php“ ein:

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

```
$db = new PDO($dsn,$username,$password);
```
hier wird die Variable „db“ erstellt, die man gleich darunter benötigt



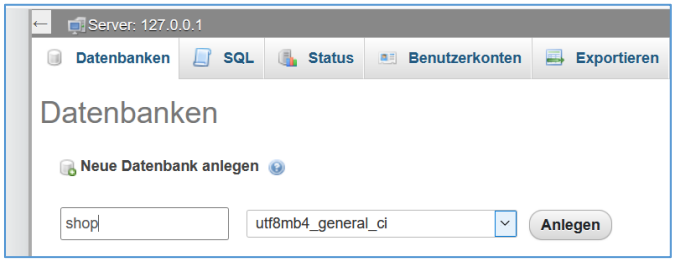
Ergebnis:



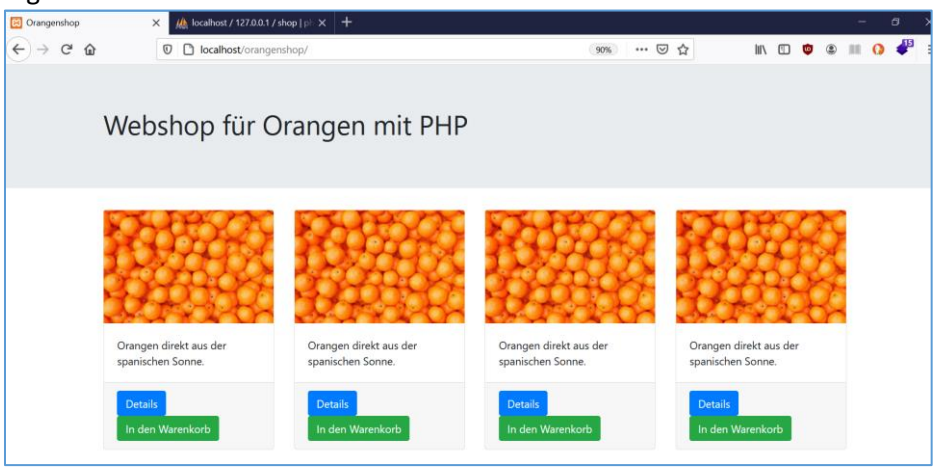
Klar, gibt ja noch keine Datenbank mit dem Namen.

7)Datenbank erstellen in PhpMyAdmin

Diese wird jetzt erstellt, mittels PhpMyAdmin in XAMPP:

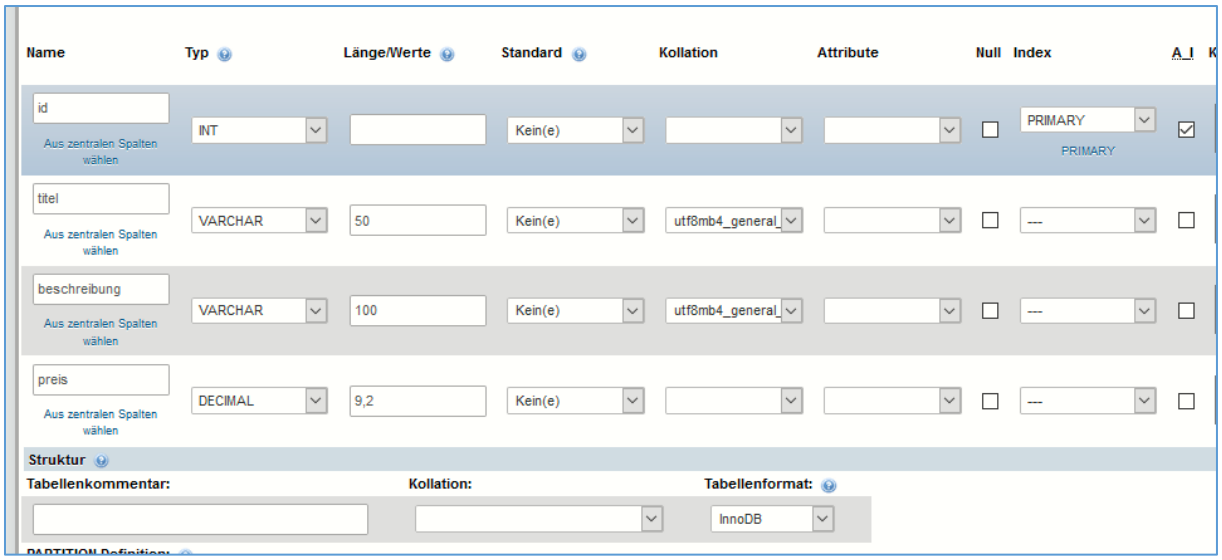


Ergebnis nun bereits wieder sichtbar:



Nun erfolgt trotzdem noch die Erstellung der ersten Tabelle. Diese trägt den Namen „products“ und hat vier Spalten:

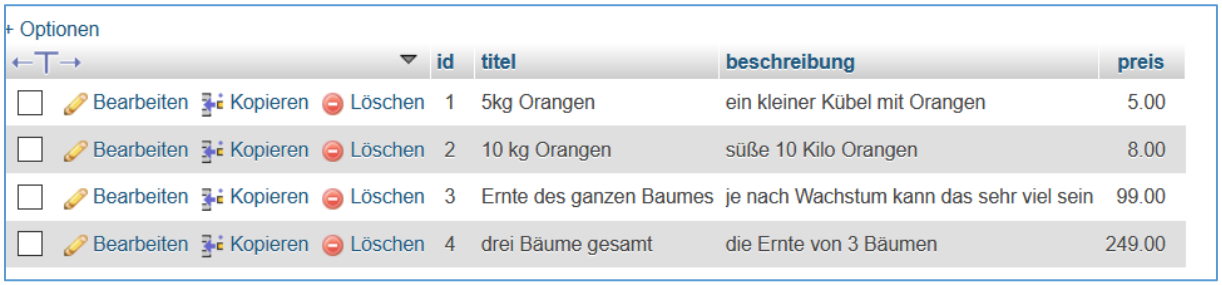
Preis: hier wird Typ DECIMAL gewählt. Bei der Länge wird 9,2 eingegeben. D.h. dass der Betrag nicht länger als 9 Zeichen (der Dezimalpunkt nicht mitgerechnet) sein kann und dass es zwei Stellen hinter dem Dezimalpunkt gibt.



Struktur:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommenta
1	id	int(11)			Nein	*kein(e)*	
2	titel	varchar(255)	utf8mb4_general_ci		Nein	*kein(e)*	
3	beschreibung	text	utf8mb4_general_ci		Nein	*kein(e)*	
4	preis	decimal(9,2)			Nein	*kein(e)*	

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



8)Produkte auslesen

Erstelle im PHP-Teil der „index.php“ die Abfrage „SELECT“:

```
6   $db = new PDO($dsn,$username,$password);
7
8   $sql = "SELECT id,titel,beschreibung,preis FROM products";
```

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

Wie üblich werden die SQL Befehle in Großbuchstaben notiert.
Die Verwendung vom Stern „*“ für alle Elemente sollte man aus Sicherheitsgründen vermeiden und lieber alle Element händisch anführen.

Danach wird in einer eigenen Variablen das Zwischenergebnis gespeichert.

```
 9
10  $result = $db->query($sql);
11
```

$result = $db->query($sql);

Nun wird das Ergebnis „result“ unten verwendet, um die Daten in einer „while-Schleife“ aus der Datenbank zu holen.
Dafür muss man die markierten Zeilen, die vier Cards, löschen und durch die „while-Schleife“ ersetzen:

```
30     <section class="container" id="produkte" >
31     <div class="row">
32         <div class="col">
33             <?php include 'card.php' ?>
34         </div>
35         <div class="col">
36             <?php include 'card.php' ?>
37         </div>
38         <div class="col">
39             <?php include 'card.php' ?>
40         </div>
41         <div class="col">
42             <?php include 'card.php' ?>
43         </div>
44     </div>
45     </section>
```

```
31     <div class="row">
32         <?php while($row = $result->fetch()):?> <!--hier der Doppelpunkt-->
33             <div class="col">
34                 <?php include 'card.php'?>
35             </div>
36         <?php endwhile;?>
37     </div>
38     </section>
```

In der Zeile der while-Schleife muss man zum Schluss den Doppelpunkt setzen, damit man ohne Probleme später mit einer „endwhile“ diese beenden kann.

Solange ein Wert aus dem $result kommt, wird unterhalb dieser Wert ausgegeben.
In der Variable $row befinden sich die Werte aus der Spalte <col>, die den Zeilen 33-35 angeführt ist.
Da hier die „card.php“ inkludiert wird, muss man nun diese bearbeiten.

Nun wechselt man in die „card.php“ um dort diese Werte auszugeben.

- Beschreibung

Der fixe Text in Zeile 4 wird nun durch den Wert aus der Datenbank ersetzt: mit Hilfe der PHP-Kurzvariante wird die Variable $row aufgerufen und daraus in eckigen Klammern die Beschreibung geholt, das ist das Element aus der Datenbank mit der gleichlautenden Bezeichnung. Diese muss natürlich vollkommen übereinstimmen und darf nicht falsch geschrieben sein:

WICHTIG: Kein Leerzeichen zwischen ? und dem Istgleichzeichen (=)

```
3       <div class="card-body">
4           <?= $row['beschreibung'] ?>
5       </div>
```

<?= $row['beschreibung'] ?>

- Titel

Auch der Titel soll aus der Datenbank geholt werden:
Dazu wird eine neue Zeile eingefügt und die Klasse „card-title“ verwendet.

```
1 <div class="card">
2       <div class="card-title"><?= $row['titel']?></div>
3       <img src="assets/img/orangen.jpg" class="card-img-to
```

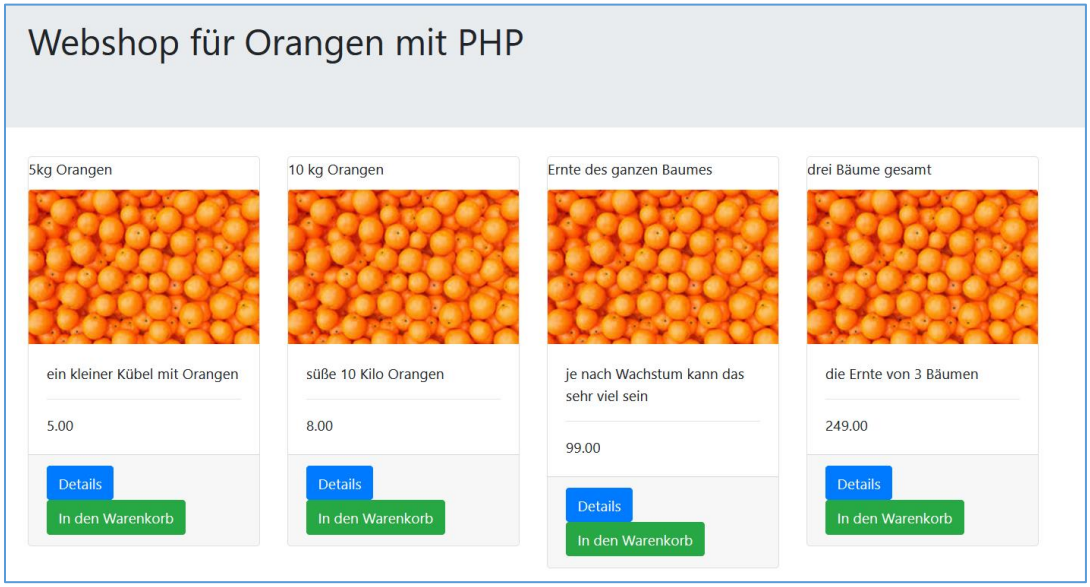
<?= $row['titel']?>

- Preis

Unter der Beschreibung soll nach einer Querlinie (<hr>) der Preis ausgegeben werden:

```
5           <?= $row['beschreibung'] ?>
6           <hr>
7           <?= $row['preis'] ?>
8       </div>
```

Ergebnis:

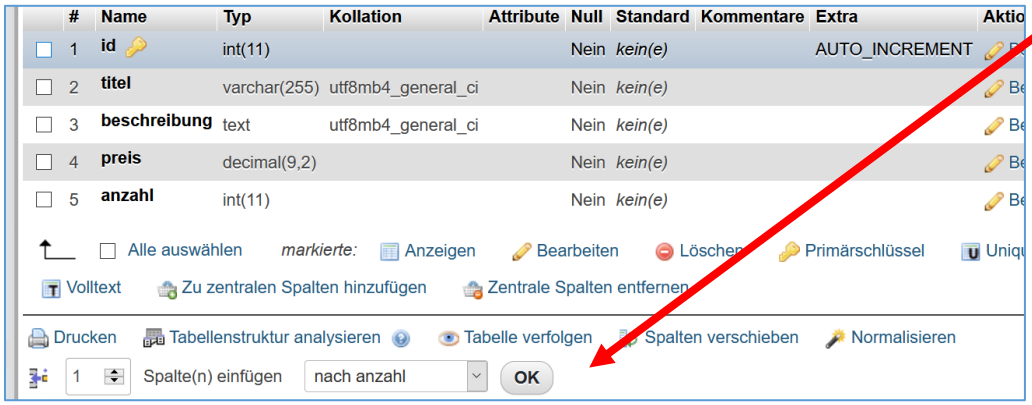


8a)Bilder einfügen und auslesen

8a1)Bilder in die Datenbank einfügen

Nun soll in der Datenbank auch das jeweilige Bild aufgenommen werden.
Öffne die Datenbank.

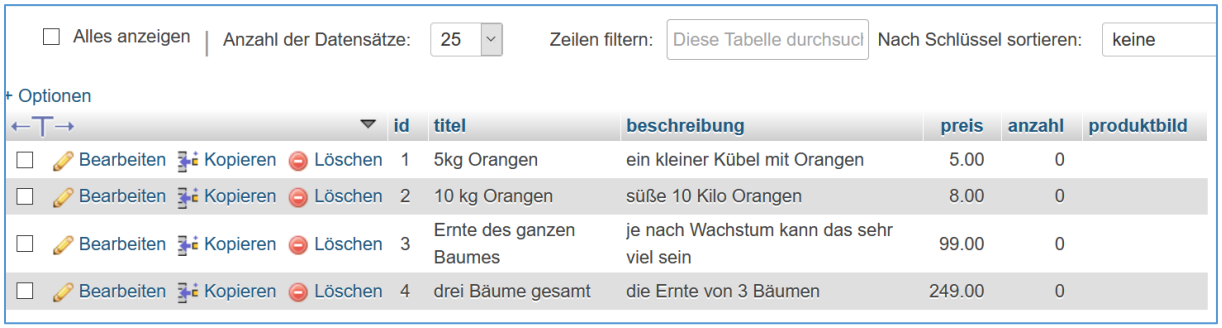
In der Tabelle „products“ öffne die Struktur und klicke auf „Ok“ bei „1 Spalte einfügen nach anzahl“.



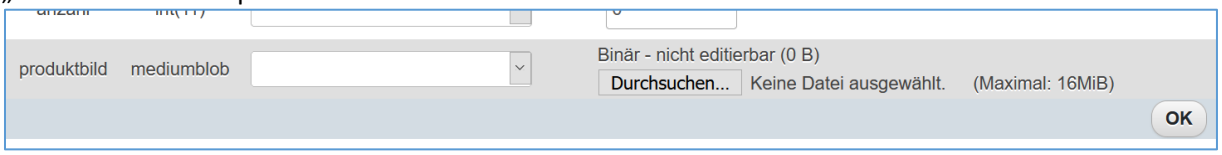
Wähle folgende Bezeichnungen:

- produktbild und
- mediumblob

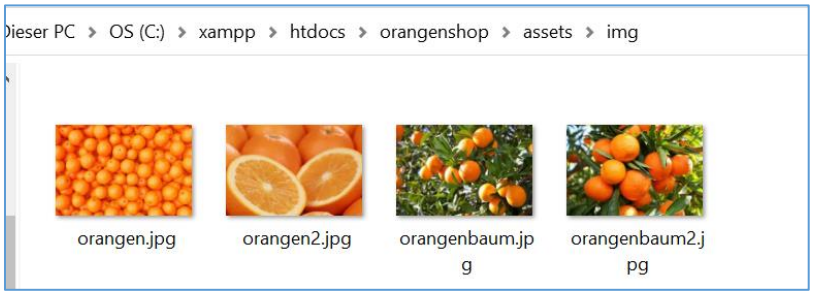
Nun ist jedoch in der „Ansicht“ kein Wert vorhanden:



Klicke daher nacheinander in jeder Spalte auf „Bearbeiten“ und wähle im Feld „produktbild“ mit „Durchsuchen“ das passende Bild aus:



Wähle dabei einfach nach der Reihenfolge aus:



Ergebnis:

	id	titel	beschreibung	preis	anzahl	produktbild
en	1	5kg Orangen	ein kleiner Kübel mit Orangen	5.00	0	[BLOB - 60,9 KiB]
en	2	10 kg Orangen	süße 10 Kilo Orangen	8.00	0	[BLOB - 44,9 KiB]
en	3	Ernte des ganzen Baumes	je nach Wachstum kann das sehr viel sein	99.00	0	[BLOB - 73,0 KiB]
en	4	drei Bäume gesamt	die Ernte von 3 Bäumen	249.00	0	[BLOB - 15,6 KiB]

8a2)Bilder anzeigen lassen

Nun müssen die Bilder nur noch im Code angesprochen werden, damit sie in der Darstellung angezeigt werden.

Öffne die Datei „card.php“ und bearbeite die Zeile 3:

```
1 <div class="card">
2       <div class="card-title"><?= $row['titel']?></div>
3       <img src="assets/img/orangen.jpg" class="card-img-top" alt="orange">
4       <div class="card-body">
5           <?= $row['beschreibung'] ?>
```

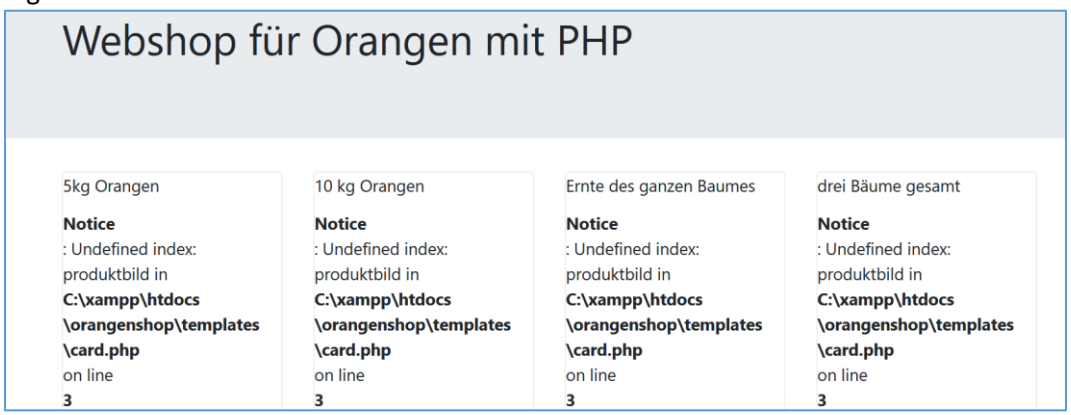
Diese Zeile soll durch folgende ersetzt werden:

<?php echo ''; ?>

```
1 <div class="card">
2       <div class="card-title"><?= $row['titel']?></div>
3       <?php echo '<img height="150" width="200"
        src="'.base64_encode( $row['produktbild'] ).'"/>'; ?>
4       <div class="card-body">
5           <?= $row['beschreibung'] ?>
```

Info:
Dabei wird auch eine PHP-Zeile eingefügt, aber nicht so einfach wie der Titel, sondern mit einer HTML-Größenangabe und der korrekten Quelle mit „src“ und der entsprechenden Formatierung.

Ergebnis:



Das heißt: es ist das neue Produktbild noch nicht aus der Datenbank geholt, daher

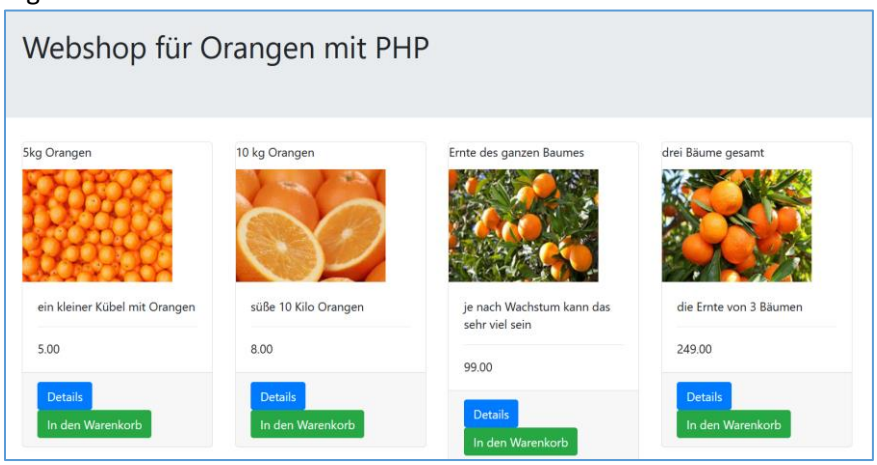
- öffne „index.php“ und

- füge das „produktbild“ hinzu.

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

Fertig.

Ergebnis:

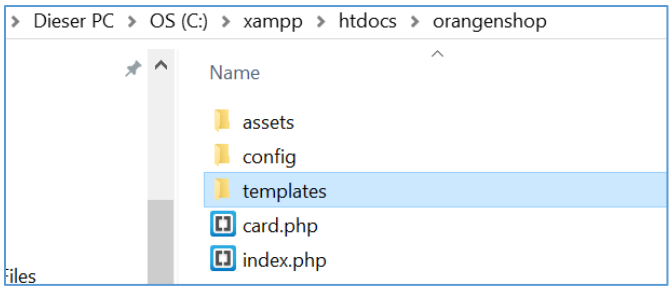


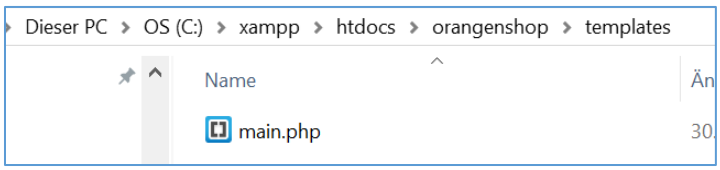
9)Code aufräumen und strukturieren – Ordner „templates“ erstellen

9.1)main.php mit Code aus index.php füllen

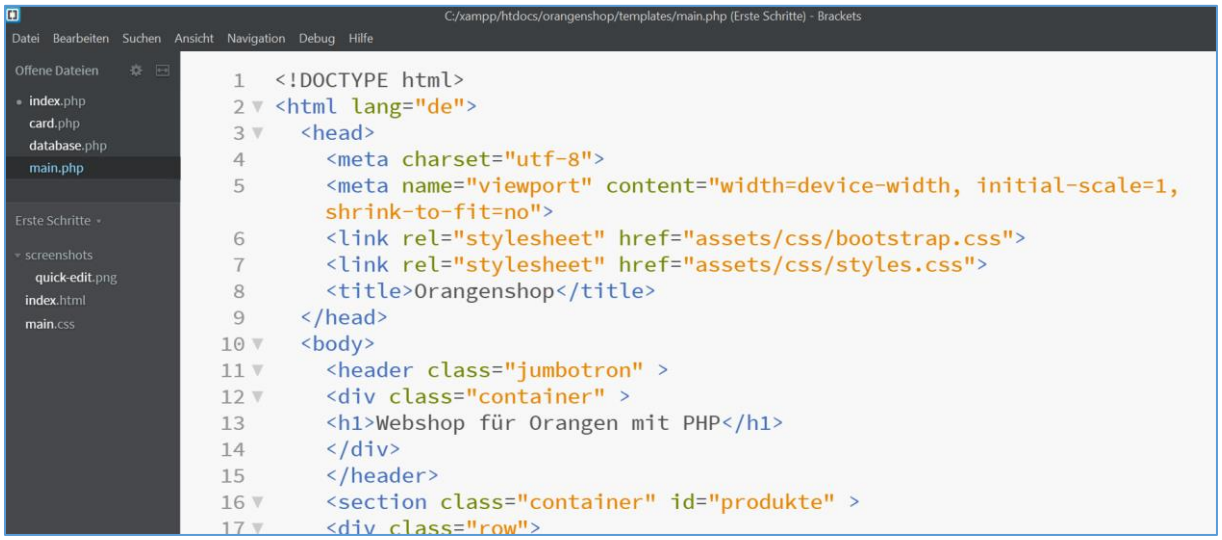
Die „index.php“ wird vom kompletten HTML-Code befreit. Dieser wird ausgeschnitten und in eine andere Datei gepackt:

- Erstelle einen Ordner „templates“
- Darin die Datei „main.php“
- Und die Datei „header.php“

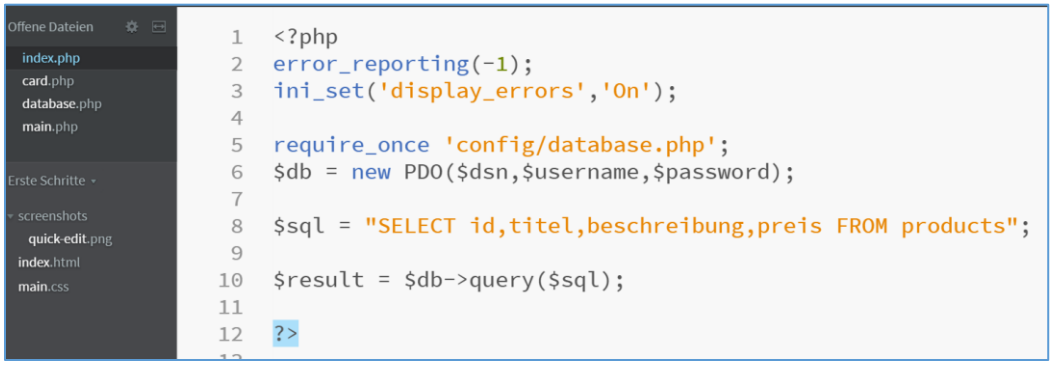




Füge nun den Code aus der index.php hier ein:



In der „index.php“ verbleibt nur mehr:



Das abschließende „php-End-Tag“ soll gelöscht werden, da hier nur reiner PHP-Code vorkommt. Sonst kann es zu einer Fehlermeldung kommen.

Dafür füge aber die Verbindung zur neuen „main.php“ hier in der letzten Zeile ein:

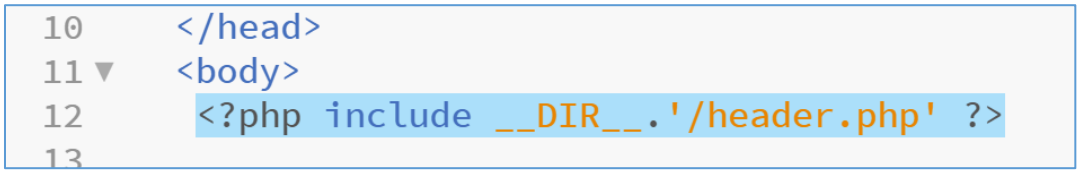
require __DIR__.'/templates/main.php';

<u>„header“ auslagern</u>

Zusätzlich wird der „header“ ausgelagert, um später dort zentral Änderungen vornehmen zu können. Schneide daher den Teil inklusive der <header>-Tags aus und füge ihn in der „header.php“ ein. Es muss kein PHP-Code dabei vorhanden sein, aber später könnte ja einer dazukommen, daher lieber php als html.

Dafür muss man aber in dem freien gewordenen Bereich im „main.php“ die Verbindung herstellen.

Verwende die Kurzschreibweise:

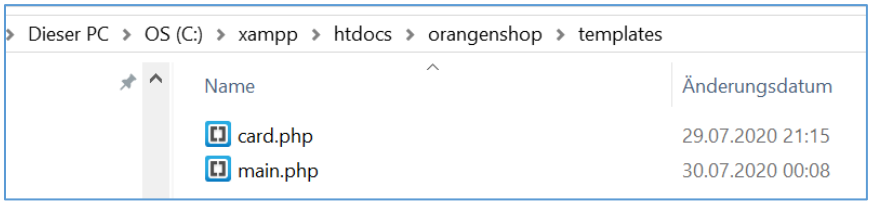


<?php include __DIR__.'/header.php' ?>

Test:
Ok die Website ist immer noch vorhanden und funktioniert.

9.2)card.php verschieben

Die card.php sollte auch in den neuen Ordner „templates“ verschoben werden, da sie auch HTML-Code anzeigt und das Layout somit festlegt.



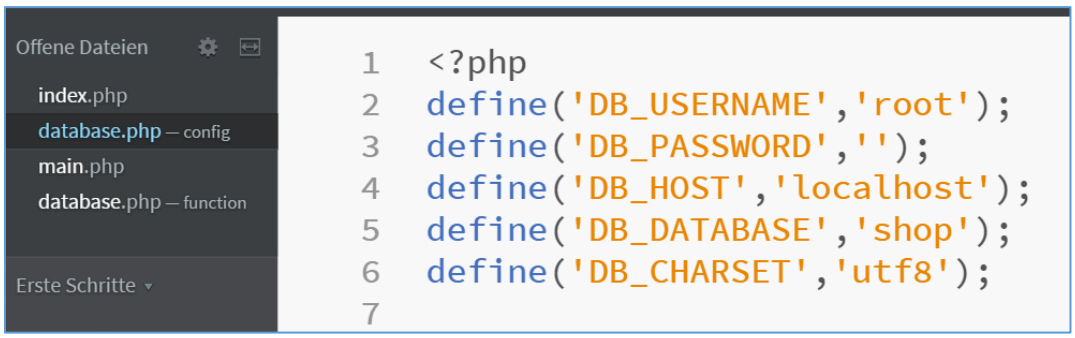
9.3)Zugriff auf die Datenbank, SQL-Abfragen in Funktionen kapseln

9.3.1)Konstante definieren

Nun werden die einzelnen Variablen aus der anderen „database.php“, die, welche in „config-Ordner“ liegt als Konstante definiert.
Diese Variablen sollen in Konstante umgewandelt werden:

Theorie:
Eine Konstante in PHP ist eine Variable, die nicht verändert werden kann. Deren Wert ist somit zur gesamten Laufzeit konstant.
Dies ist für Zugangsdaten eine gute Möglichkeit.

- Dazu benötigt man die Funktion „define“.
    - Darin wird der Name der Variable und der Wert eingesetzt.

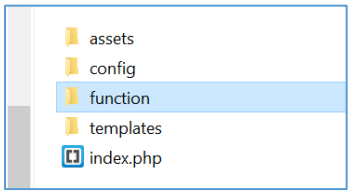



```
<?php
define('DB_USERNAME','root');
define('DB_PASSWORD','');
define('DB_HOST','localhost');
define('DB_DATABASE','shop');
define('DB_CHARSET','utf8');
```

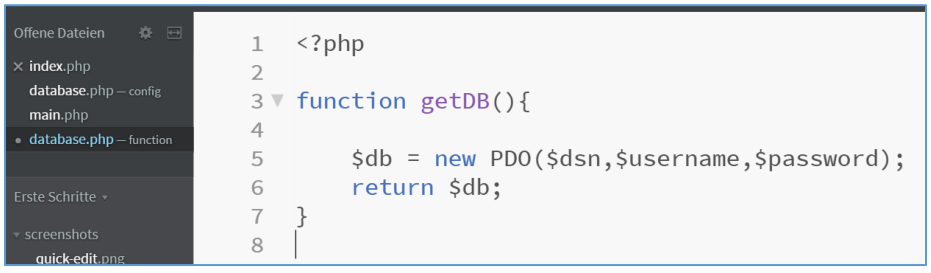
9.3.2)statische Variable erstellen

Damit kann man die Funktionen einfacher benutzen.

Dafür wird ein neuer Ordner erstellt, der alle Funktionen aufnehmen wird. Darin erfolgt anfangs hier eine Erstellung der neuen Datei „database.php“ in dem Ordner „function“



In der „database.php“ erstelle eine Funktion, die den Zugriff auf die Datenbank liefern soll.



Es wird nun jedes Mal, wenn die Datenbank aufgerufen wird eine neue Instanz angelegt. Um das zu verhindern, wird das Schlüsselwort „static“ verwendet.

Damit wird eine Variable definiert, die ihren Wert behält, auch wenn die Funktion mehrfach aufgerufen wurde, d.h. diese Variable wird nicht jedes Mal neu angelegt.

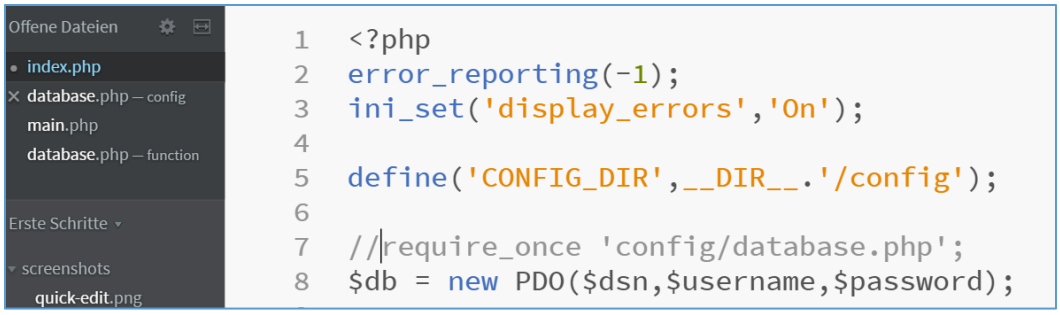
```
3  function getDB(){
4      static $db;
```

Darunter wird in einer IF-Entscheidung abgefragt, ob die Variable schon eine Instanz von PDO hat. Das hat sie nämlich beim ersten Aufrufen noch nicht und ist somit „Null“. Daher ist die IF-Abfrage ein „false“. Somit wird der Code weiter ausgeführt und eine neue Instanz angelegt und die „new PDO“ durchgeführt. Der Wert wird dann zurückgegeben und bleibt vorhanden, da die statische Variable belegt wurde. Beim zweiten Aufruf ist somit ein Wert vorhanden, der nun bleibt. Die IF-Abfrage ist jetzt ja „true“.

```
3  function getDB(){
4      static $db;
5      if($db instanceof PDO){
6          return $db;
7      }
```

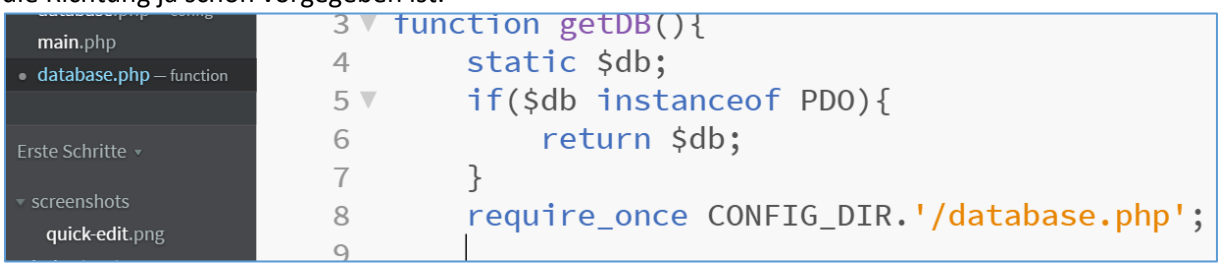
9.3.3)Die Konstanten verwenden

Um zugreifen zu können, muss man in der „index.php“ diesen Zugriff erstellen. Daher muss man eine weitere Konstante erstellen, wieder mit „define“ in Zeile 5:
Der Ordner „config“ befindet sich auf der gleichen Ebene wie die „index.php“.



Aber somit kann man die Zeile 7 mit „require“ löschen, da die Verbindung neu in Zeile 5 erstellt wird,

Zugriff auf die einzelnen Konstanten herstellen:
Diese Konstante verwendet man nun auch in der eben bearbeiteten „database.php“ im Ordner „function“: in Zeile 8
Nach dem „require_once“ genügt es den Slash und die Datei anzuführen, da durch die „config_dir“ die Richtung ja schon vorgegeben ist.



require_once CONFIG_DIR.'/database.php';

Hier werden nun auch die Variablen mit den Konstanten belegt, d.h. neu umgeschrieben:

```
10      $db = new PDO($dsn,$username,$password);
```

Statt dieser 3 Variablen erfolgt die Bezeichnung der Konstanten aus der „config/database.php“:

```
 9
10      $db = new PDO($dsn,DB_USERNAME,DB_PASSWORD);
11      return $db;
```

Und für die Variable „$dsn“ wird ein neuer STRING erstellt, eine Zeile darüber. Dabei werden mit „%s“ einzelne Platzhalter definiert.
Gleichzeitig wird die Funktion „sprintf“ verwendet: dabei werden nach dem Beistrich die einzelnen Parameter für die Platzhalter definiert, wie z.B. DB_HOST.

$dsn = sprintf("mysql:host=%s;dbname=%s, charset=%s",DB_HOST,DB_DATABASE,DB_CHARSET);

```
 8      require_once CONFIG_DIR.'/database.php';
 9      $dsn =
        sprintf("mysql:host=%s;dbname=%s;charset=%s",DB_HOST,DB_DATABASE,
        DB_CHARSET);
10      $db = new PDO($dsn,DB_USERNAME,DB_PASSWORD);
```

9.3.4)index.php

Nun muss man diese „function/database.php“ noch in der „index.php“ einbinden.

Löschen: Die Variable $db kann gelöscht werden, da sie ja bereits übernommen wurde.

```
5  define('CONFIG_DIR',__DIR__.'/config');
6  require_once __DIR__.'/function/database.php';
7
8  $sql = "SELECT id,titel,beschreibung,preis FROM products";
```

require_once __DIR__.'/function/database.php';

In Zeile 10 gibt es ja die $db nicht mehr und diese wird ersetzt durch „getDB()“:

Offene Dateien

index.php
database.php – config
main.php
database.php – function

Erste Schritte

screenshots
quick-edit.png
index.html
main.css

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

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

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

$result = getDB()->query($sql);

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

Ergebnis und Test:
Die Seite funktioniert weiterhin.