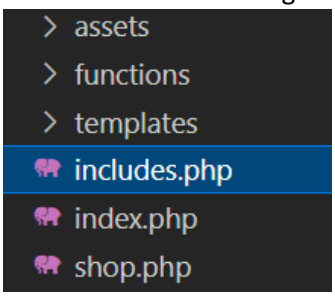
5)Login mit userId – PDO, Datenbank

1)Pfade richtig einstellen – includes.php erstellen

Erstelle neben der index.php (also im „root“-Pfad ganz außerhalb) eine Datei „includes.php“.

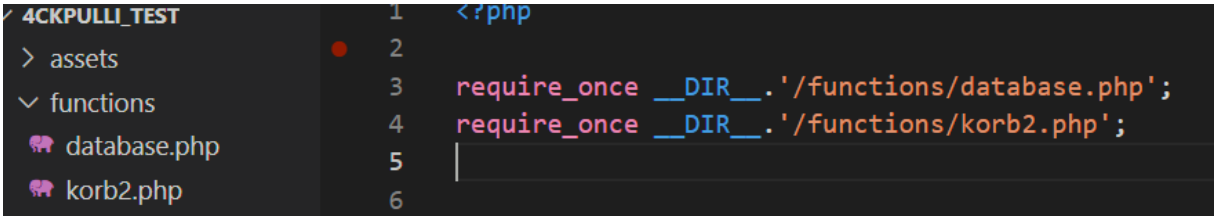
Diese soll alle Beziehungen zu den Dateien im Ordner „functions“ steuern.



Nutze die **Möglichkeit des absoluten Pfades**, den man verwendet, wenn ein relativer Pfad nicht funktioniert, sondern wenn die Datei von unterschiedlichen Ordnern benötigt wird (eingebunden werden soll):

__DIR__

Darin legen die beiden Pfade an:

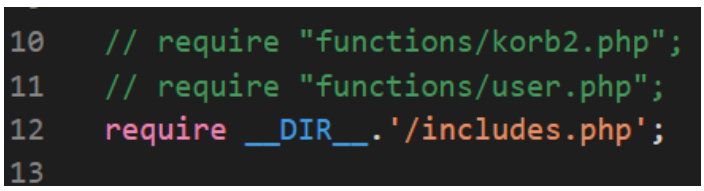



```
<?php
require_once __DIR__.'/functions/database.php';
require_once __DIR__.'/functions/korb2.php';
```

Ersetze danach in den beiden Dateien die Verbindungen zum Korb (functions/korb2.php) mit dieser neuen Verbindung zu dieser „includes.php“:

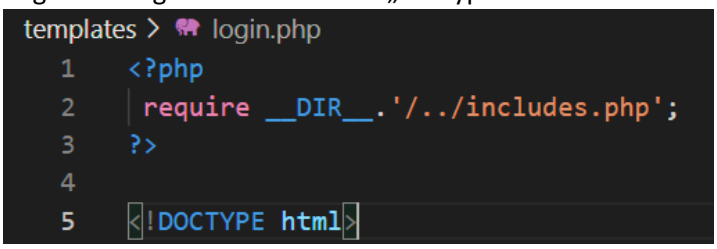
- In der index:

- In der shop:
  Dafür aber die beiden schon vorhandenen Verbindungen auskommentieren oder löschen:



In der „login.php“ und „registrieren.php“ muss ebenfalls diese Verbindungsdatei eingebaut werden. Da diese aber im Ordner „templates“ liegen, mit einem anderen Pfad.

Füge diesen ganz oben vor dem „doctype“ mittels PHP ein:




```
<?php
 require __DIR__.'/../includes.php';
?>
```

Info:
https://tutorials.supunkavinda.blog/php/include-require

include vs require

The `include` and `require` statements are identical except their error handling behavior when the file was not found.

- `include` produces a warning and continues execution of the script
- `require` produces a fatal error and stops the execution of the script

If the included script is needed to do the next processes, you should use `require`.

If you do

```
require(__DIR__ . '/file.php')
```

then you are requiring the file with the full pathname. If the file doing this require is required by another file in another directory, this require will always work. On the other hand, if you

```
require('file.php')
```

then if the file where this require statement is is required by another file in another directory, this statement will fail.

That is why it is generally good practice to include the `__DIR__`.

https://stackoverflow.com/questions/5371828/relative-path-in-require-once-doesnt-work

2)Erstelle eine neue Datei „user.php“ im Ordner „functions“

2a)Neue Funktion „isLoggedIn“ in der „user.php“

functions > user.php

```php
<?php

function isLoggedIn():bool{
    return isset($_SESSION['userId']);
}
```

function isLoggedIn():bool{
  return isset($_SESSION['userId']);
}

Nimm dann sofort diese Datei auch in die „includes.php“ auf:

```php
require_once __DIR__.'/functions/database.php';
require_once __DIR__.'/functions/korb2.php';
require_once __DIR__.'/functions/user.php';
```

Damit wird immer überprüft, ob der User eingelogt ist oder nicht.

Info:

- Diese Funktion gibt zurück ein „true“ oder „false“ – mit dem „bool“
- und ob die Session gesetzt ist, die nach dem Einloggen entsteht. Diese entsteht ja in der „login2.php“, sobald man korrekt eingeloggt ist.

2) navbar.php anpassen

Da jetzt die Funktion „isLoggedIn“ vorhanden ist, kann man das in der navbar.php berücksichtigen.

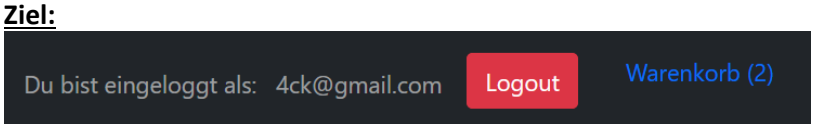
- Wenn man eingeloggt ist,
  - Soll die E-Mail angeführt werden, die in „login2.php“ als SESSION festgelegt wurde, und angezeigt werden (Bild: login2.php)

```php
$_SESSION['email'] = $row['email'];
```

  - soll der Link „Logout“ angezeigt werden, nämlich als Button in roter Farbe

- wenn man nicht eingeloggt ist, der Link „Login“ und „Registrieren“.

Ziel:



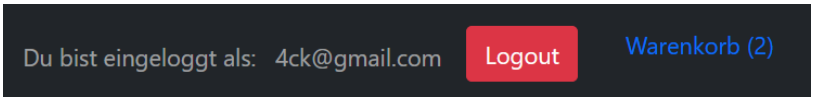
Das mit PHP-Code sieht folgendermaßen aus:

```
<ul class="navbar-nav">
<li>
<?php if(isLoggedIn()):?>
<a class="nav-link">Du bist eingeloggt als:
  <?php echo $_SESSION['email'] ?>  
</a>
</li>
<li>
<a href="templates/logout.php">
<button class="btn btn-danger">Logout</button>
</a>
<?php endif;?>
</li>
```

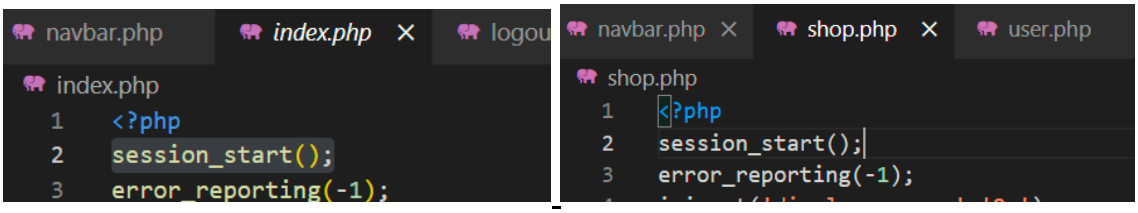
```
<li>
<?php if(isLoggedIn()):?>
<a class="nav-link">Du bist eingeloggt als:
  <?php echo $_SESSION['email'] ?>  
</a>
</li>
<li>
<a href="templates/logout.php">
<button class="btn btn-danger">Logout</button>
</a>
<?php endif;?>
</li>
```

```
</li>
<li class="nav-link">
<?php if(!isLoggedIn()):?>
<a href="templates/login.php">Login</a>
<a href="templates/registrieren.php">Registrieren</a>
<?php endif;?>
</li>
</ul>
<form class="d-flex">
```

```
<li class="nav-link">
<?php if(!isLoggedIn()):?>
<a href="templates/login.php">Login</a>
<a href="templates/registrieren.php">Registrieren</a>
<?php endif;?>
</li>
</ul>
```



Damit das funktioniert, muss die index.php und shop.php eine Funktion session_start() erhalten

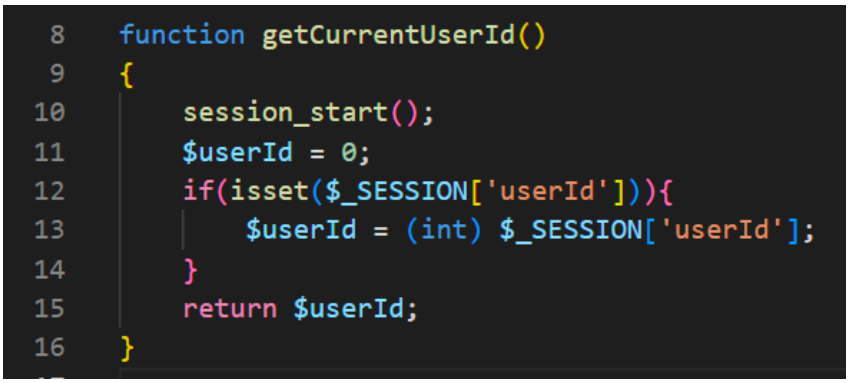


3)userId über SESSION und eine Funktion einbinden

Diese Variable „userId“ benötigt man dann laufend, weil man immer schauen muss, ob man noch beim richtigen, gerade geöffneten User ist. Z.B., wenn man ein Produkt in den Warenkorb legt, muss man die richtige User-Zuordnung durchführen.

Schreibe in die user.php unterhalb der Funktion „isLoggedIn“ folgenden Code weiter:

3a) function getCurrentUserId()



Zuerst muss man die Möglichkeit schaffen, auf die SESSION zugreifen zu können, daher „session_start()“. Dann wird 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.

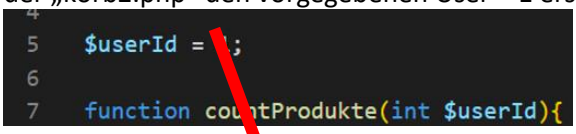
Vor dem Ende der Funktion muss man noch die „userId“ zurückliefern mit „return“:

Code:

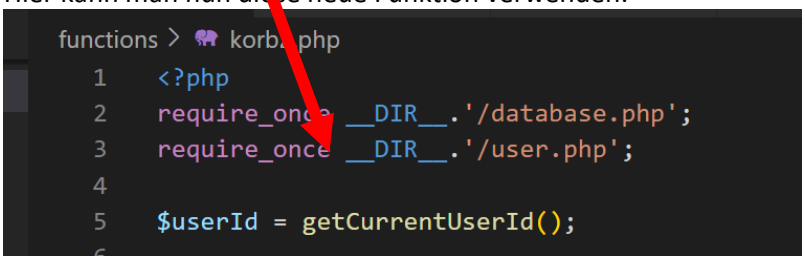
```
function getCurrentUserId()
{
   session_start();
   $userId = 0;
   if(isset($_SESSION['userId'])){
      $userId = (int) $_SESSION['userId'];
   }
   return $userId;
}
```

3b)korb2.php

Da man nun einen User hat und somit weiß, welcher user etwas in den Warenkorb legt, kann man in der „korb2.php“ den vorgegebenen User = 1 ersetzen:

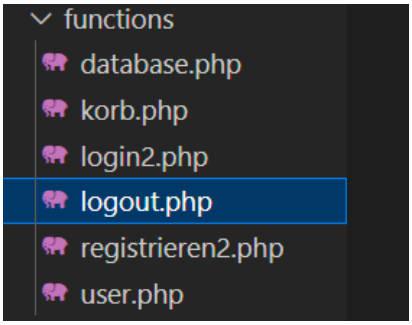


Hier kann man nun diese neue Funktion verwenden:



4)Logout

Dafür muss man eine neue Datei erstellen: im Ordner „functions“ erstelle „logout.php“



Darin wird zuerst die Session gestartet, wie in fast jeder anderen PHP-Datei auch. Dort wurde der Start in den Bereich „header.php“ gelegt, der fast überall benutzt wird.

Danach erfolgt das Löschen der Session. Damit werden alle offenen IDs oder User entfernt.


```
<?php
session_start();
session_destroy();
header("Location: ../index.php");
exit();
```

Danach sollte man zur „index.php“, also zur Startseite geleitet werden. Daher erfolgt nun die Umleitung mit „header-location“.

Testen:

Email: 4ck@gmail.com
Passwort: 1234

u_id	email	password	vorname
1	4ck@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	Franz