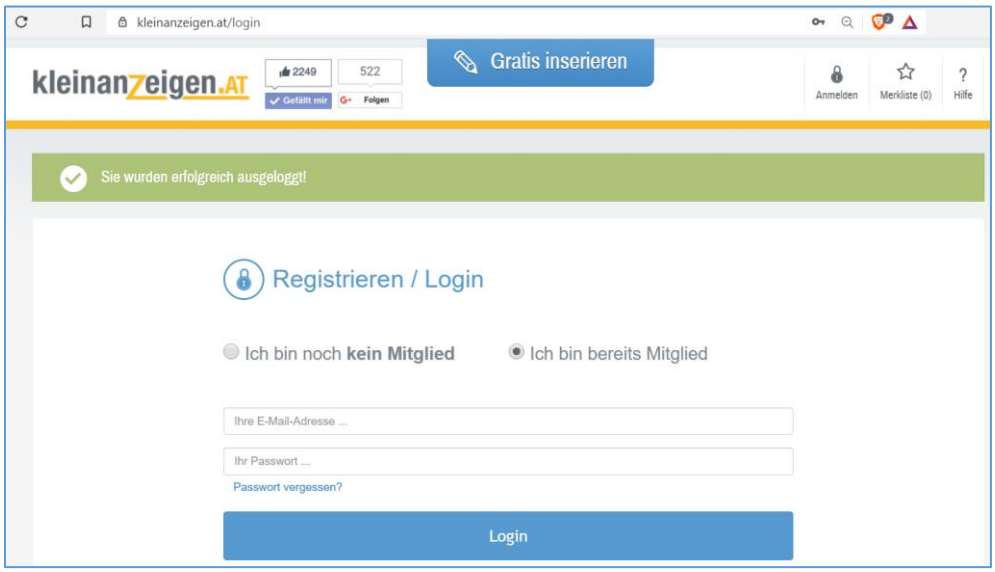
Schueler - Login in PHP mit Datenbank – Teil 5:

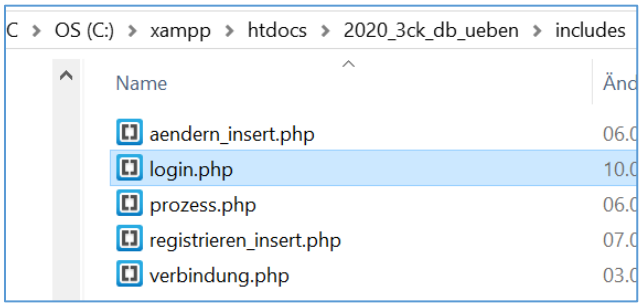
1. erstellen von login.php
2. Datenbank Abfrage
3. Passwort überprüfen
4. Login
5. Freigeben von Inhalt, wenn eingeloggt
6. Adminbereich absichern
7. Buttons abhängig von Login anzeigen
8. Logout

Login ist ein wichtiger Bestandteil und kann auch sehr schön aussehen:



1)Login.php erstellen, inkl. Fehler- bzw. Erfolgsmeldung

Erstelle im Ordner „includes“ die Datei „login.php“.



Diese ist vom Aufbau und Code der „registrieren_insert.php“ ähnlich.

Es startet ebenfalls mit der Überprüfung, ob der Submit-Button geklickt wurde.

Daher gib ein:

Vergleiche, ob der Name des Button in der Datei „header.php“ auch genauso lautet:

```
39 ▼        <form action="includes/login.php" method="post">
40          <input type="text" name="mailuid" placeholder="Benutzer/E-Mail">
41          <input type="password" name="pwd" placeholder="Passwort">
42          <button type="submit" name="login-submit">Login</button>
43          </form>
```

In der „else“ soll nun auf die Seite geleitet werden, auf der man sich eigentlich befindet, nämlich die „index.php“. Die „../“ müssen sein, da wir uns im Ordner „includes“ befinden und zuerst diesen verlassen müssen. Die Umleitung erfolgt wiederum mit der typischen PHP-Umleitung:

```
 7 ▼ else {
 8       header("Location: ../index.php");
 9       exit;
10   }
11
12   ?>
```

Im Gegenteil zur „else“ wird nun in der IF zuerst die Verbindung hergestellt. Dies erfolgt wiederum mit einem „require“:

```
 3   //wenn Button geklickt
 4   if (isset($_POST['login-submit']))
 5 ▼ {
 6       require 'verbindung.php';
```

Danach werden 2 Variablen erstellt, die die Eingabe der Logindaten (durch einen Benutzer auf der Starseite) erhalten. Deren Input-Namen sind „mailuid“ und „pwd“ – siehe header.php.

$mailuid = $_POST['mailuid'];
$password = $_POST['pwd'];

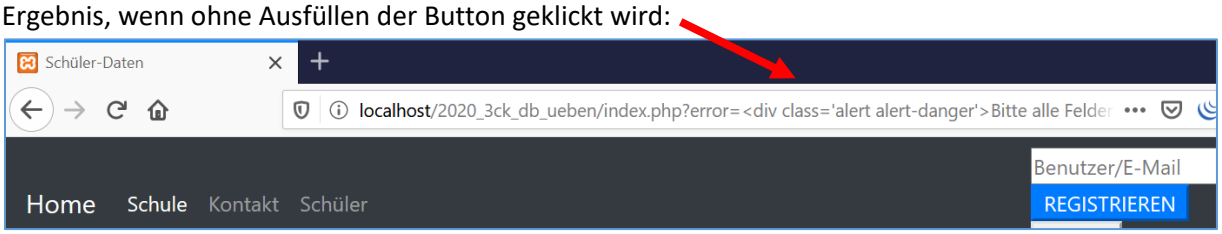
```
6       require 'verbindung.php';
7
8   $login_user = $_POST['mailuid'];
9   $login_pwd = $_POST['pwd'];
```

Wichtig ist es, wie im „registrieren_insert.php“ die beiden Felder zu prüfen, ob sie leer gelassen wurden, oder nicht. Wenn das der Fall ist, soll eine Weiterleitung zurück zur Startseite stattfinden. Diese wird hier nur mit einer einfachen error-message gesendet.

```
if(empty($login_user) || empty($login_pwd) ) {
   header("Location: ../index.php?error=<div class='alert alert-danger'>Bitte alle Felder
ausfüllen</div>");
   exit();   }
```

```php
if(empty($login_user) || empty($login_pwd) )
{
    header("Location: ../index.php?error=<div class='alert alert-danger'>Bitte alle Felder ausfüllen</div>");
    exit();
}
```

Ergebnis, wenn ohne Ausfüllen der Button geklickt wird:



Fehlermeldung schön anzeigen lassen:

Jetzt muss man sich überlegen, wo man auf der Seite „index.php“ die Fehlermeldung ausgeben will. Eigentlich kann man festhalten, dass es nicht im „header“ sein wird. Eher im „body“ der Hauptseite.

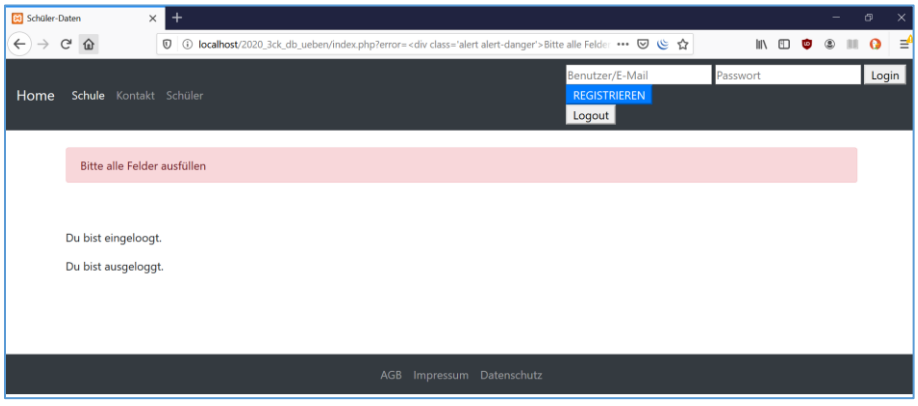
Öffne die „index.php“.

Füge mit entsprechendem Abstand unterhalb des „header“ und oberhalb des Containers den Code ein:

```php
    <div class="container">
        <br>
        <?php
            if(isset($_GET['error'])) {
            $error = $_GET['error'];
            echo $error;
            }
          ?>
        <br><br>
```

```
<?php
    if(isset($_GET['error'])) {
    $error = $_GET['error'];
    echo $error;
    }
  ?>
```

Ergebnis:



2)Datenbankabfrage:

Nun wird in der Datenbank nachgesehen, ob der User vorhanden ist **ODER** eine passende E-Mail dazu. Hier wird absichtlich **beides ermöglicht**, was ein wenig Mehraufwand ist, aber oft von Benutzer mit Freude angenommen wird.

Hier werden nun die Daten des eingebenden Benutzers mit den Daten der Datenbank auf Übereinstimmung geprüft, nämlich

a) zuerst ob Benutzer ODER E-Mail passen und erst dann
b) wird das Passwort verglichen

Sicherheit erhöhen:

Dabei werden Placeholder „prepared statements“ zur Sicherheit verwendet. Hier sind das die Fragezeichen.

Benutzer oder E-Mail ok?

In der folgenden Abfrage werden aus der Tabelle der Darenbank die Elemente für den Benutzername und für die E-Mail gesucht.

☐	8	**benutzername**	varchar(50)	utf8_general_ci
☐	9	**email**	varchar(50)	utf8_general_ci
☐	10	**pass**	longtext	utf8_general_ci

Beachte den Strichpunkt vor und nach dem schließenden Anführungszeichen.

$sql = "SELECT * FROM schueler WHERE benutzername=? OR email=?;";

```
14        exit();
15    }
16 ▼ else {
17        $sql = "SELECT * FROM schueler WHERE benutzername=? OR email=?;";
18    }
19    }
20 ▼ else {
21        header("Location: ../index.php");
22        exit;
23    }
```

Nun erfolgt eine neue Variable bzw. ein neues Statement:

Die Initialisierung erfolgt bezüglich der korrekten Verbindung $con.

```
$stmt = mysqli_stmt_init($con);
```

$stmt = mysqli_stmt_init($con);
mysqli_stmt_prepare($stmt, $sql)

```
16 else {
17       $sql = "SELECT * FROM schueler WHERE benutzername=? OR email=?;";
18       $stmt = mysqli_stmt_init($con);
19       mysqli_stmt_prepare($stmt, $sql);
20  }
```

Nun erfolgt erst die Überprüfung der Daten ob sie zusammenpassen (Vergleich):

Dann werden die Informationen, die man durch das Ausfüllen bekommt, verwendet. Diese werden nun gebunden (bind) und überprüft, ob sie mit der Datenbank zusammenpassen. Wenn ja, erfolgt eine Meldung „Success“.

Zeile 21: Hier werden die Daten des Benutzers in die Datenbank gegeben, um zu schauen, ob die Eingabe (das Statement) des Users zu einem Resultat aus der Abfrage in Zeile 17 führen:

Die beiden Daten, die mit den Daten aus der Datenbank kontrolliert werden, werden als zwei Strings mittels „ss“ übergeben. Dafür steht ein „s“ für einen „string“. Hier werden danach zweimal, sind ja zwei Strings, die aktuellen Daten genommen, die oben in der Zeile 8 als Variable erstellt sind, nämlich „$login_user“. Zweimal, da es ja der Benutzer aber auch das E-Mail sein kann, siehe Zeile 17.

mysqli_stmt_bind_param($stmt, "ss", $login_user, $login_user);

```
17       $sql = "SELECT * FROM schueler WHERE benutzername=? OR email=?;";
18       $stmt = mysqli_stmt_init($con);
19       mysqli_stmt_prepare($stmt, $sql);
20
21       mysqli_stmt_bind_param($stmt, "ss", $login_user, $login_user);
22       mysqli_stmt_execute($stmt);
23       $result = mysqli_stmt_get_result($stmt);
```

Damit man ein Resultat aus der Datenbank erhält, muss man dieses nun ausführen (execute):
Zeile 22:

mysqli_stmt_execute($stmt);

Das daraus erhaltene Ergebnis wird nun in eine Variable ($result) gebunden, die man danach benötigt:
Zeile 23

$result = mysqli_stmt_get_result($stmt);

Nun muss man checken, ob dieses Ergebnis in $result auch sinnvoll ist, d.h. ob es gefüllt oder leer ist. Das Resultat kann ja eigentlich auch leer sein, weil keine Übereinstimmung gefunden wurde.

Geklärt wird das nun mit einem IF.

Hier wird überprüft, ob man ein Resultat bekommen hat und wenn nicht, gibt’s wieder eine Fehlermeldung.

In der IF wird gleich in der Bedingung mit der neuen Variablen $row mittels „fetch_assoc“ das Resultat aus der Zeile darüber geholt. Es wird hier somit in ein assoziiertes Array gegeben, um damit danach was zu tun. Das Ergebnis in $result ist nämlich eine „rohe Datenlösung“, die nun mit dem assoziierten Array verwendet werden kann.

Siehe Zeile 25:

```
24              $result = mysqli_stmt_get_result($stmt);
25              if ($row = mysqli_fetch_assoc($result)) {
```

Erstelle gleich die „else“, wenn es zu einem Error kommen sollte: mit der Fehlermeldung „Kein User/E-Mail gefunden“

```
23      $result = mysqli_stmt_get_result($stmt);
24
25      if ($row = mysqli_fetch_assoc($result)) {
26
27          }else {
28              header("Location: ../index.php?error=<div class='alert alert-
                danger'>Kein User/E-Mail gefunden </div>");
29              exit();
30          }
31      }
```

3)Passwort überprüfen

Erst jetzt wird auch das Passwort überprüft, ob das eingegebene mit dem aus der Datenbank übereinstimmt. Noch vor der „else“, da es ja „true“ ist, dass die $row was gefunden hat.

Passwort vergleichen: password_verify()“

Nun folgt im nächsten IF-Vergleich die Maßnahme, dass, nachdem ja ein Benutzer gefunden wurde, das Passwort, dass der Benutzer eingetippt hat gehasht wird und dann mit dem in der Datenbank, beim User, verglichen wird.

- Daher wird eine Variable erstellt „pwdCheck“.
- Die Funktion „password_verify()“ vergleicht die Passwörter – als erster Parameter das vom User eingegebene, als zweiter das von der Datenbank. Hier ist das zweite so zu verwenden, als es aus dem eben erstellten $row genommen wird, nämlich aus der Datenbank das Feld „pass“.

$pwdCheck = password_verify($login_pwd, $row['pass']);

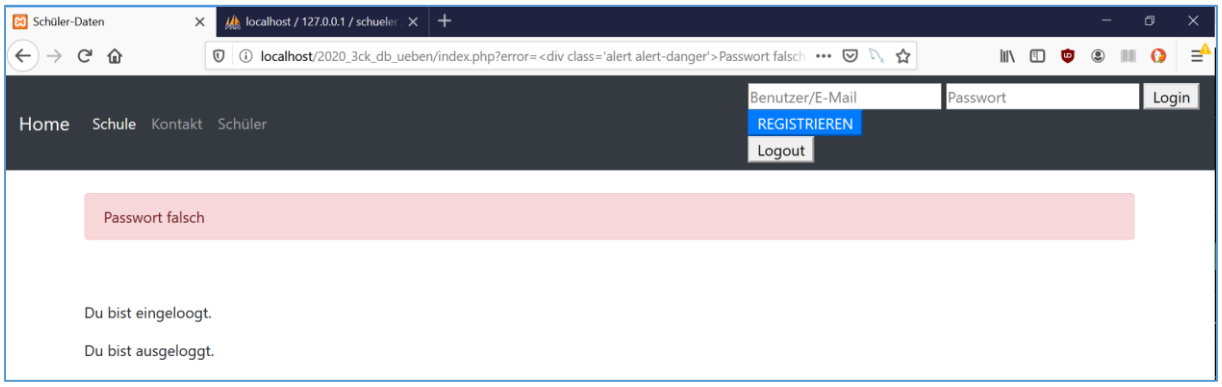
Da das jetzt wahr oder falsch sein kann, folgt wieder eine IF-Abfrage. Ist es nämlich „false“, dann sollte man den user nicht einloggen lassen und ihn zurücksenden zur Startseite mit einer Fehlermeldung.

```
25      if ($row = mysqli_fetch_assoc($result)) {
26          $pwdCheck = password_verify($login_pwd, $row['pass']);
27          if ($pwdCheck == false) {
28              header("Location: ../index.php?error=<div class='alert alert-
                danger'>Passwort falsch</div>");
29              exit();
30          }else {
31              header("Location: ../index.php?error=<div class='alert alert-
                danger'>Kein User/E-Mail gefunden </div>");
32              exit();
33          }
34      }
```

Code:

```
if ($pwdCheck == false) {
    header("Location: ../index.php?error=<div class='alert alert-danger'>Passwort falsch</div>");
    exit();
```

Ergebnis:



Endlich alles richtig eingegeben:

Wenn er aber der richtige Benutzer ist, dann ist es das „true“-Statement und dann soll zur Sicherheit noch eine **Session gestartet** werden, um den User dann auch durch zu lassen auf die versperrte Seite. Dies wird hier mit einer „else if“ erstellt, die auf die geheime Seite, hier „adminbereich.php“ weiterleitet. Die „else if“ ist nötig, da danach ja auch eine „else“ kommt, sonst gäbe es ein Problem.

```
25      if ($row = mysqli_fetch_assoc($result)) {
26          $pwdCheck = password_verify($login_pwd, $row['pass']);
27          if ($pwdCheck == false) {
28              header("Location: ../index.php?error=<div class='alert alert-
                danger'>Password falsch</div>");
29              exit();
30              }
31              //endlich alles richtig
32              else if ($pwdCheck == true) {
33
34                  header("Location: ../adminbereich.php?error=<div
                    class='alert alert-success'>Willkommen. </div>");
35                  exit();
36              }
37
38          } else {
39              header("Location: ../index.php?error=<div class='alert alert-
                danger'>Kein User/E-Mail gefunden </div>");
40              exit();
41
42      }
```

4)Hier erfolgt endlich der LOGIN:

Er hat nun das richtige E-Mail oder den richtigen Benutzer und dann auch das korrekte Passwort eingegeben und darf somit weiter geleitet werden. Somit wird in der gerade eingefügten „else“, aber noch vor der „header“-Weiterleitung eine SESSION erzeugt.

- Zuerst wird die SESSION gestartet.
- Dann wird eine session-Variable (der Name ist frei aber passend zu erstellen) mit den Daten der Datenbank zugeordnet. Hier muss man die genaue Bezeichnung aus der Datenbank verwenden. Es werden 2 Daten genommen, die id und der Benutzername.

Danach erfolgt die erfolgreiche Weiterleitung auf eine Seite, die nun freigegeben wird. Bei uns soll das die Seite „adminbereich.php“ sein. Zusätzlich kann auch die Erfolgsmeldung mitgegeben werden.

```
31              //endlich alles richtig
32              else if ($pwdCheck == true) {
33                  session_start();
34                      $_SESSION['userId'] = $row['s_id'];
35                      $_SESSION['userName'] = $row['benutzername'];
36
37                  header("Location: ../adminbereich.php?error=<div
                    class='alert alert-success'>Willkommen. </div>");
38                  exit();
39              }
```

```
session_start();
    $_SESSION['userId'] = $row['s_id'];
    $_SESSION['userName'] = $row['benutzername'];

header("Location: ../adminbereich.php?error=<div class='alert alert-success'>Willkommen.
</div>");
exit();
```

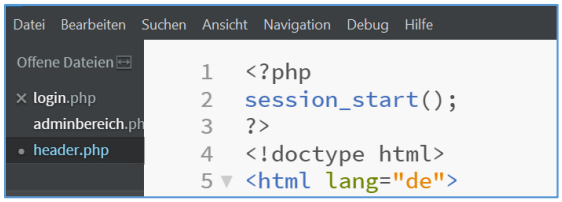
SESSIONs in allen Seiten starten lassen:

Damit man mit Sessions hier nun arbeiten kann, muss man in alle Seiten der Website diese ganz am Beginn der Seiten starten lassen.

Ohne diesem Starten der Session würde man nicht auf die durch das LOGIN erstellten und nun vorhandenen Session-Variablen zugreifen können, um zu sehen ob sie verfügbar sind.

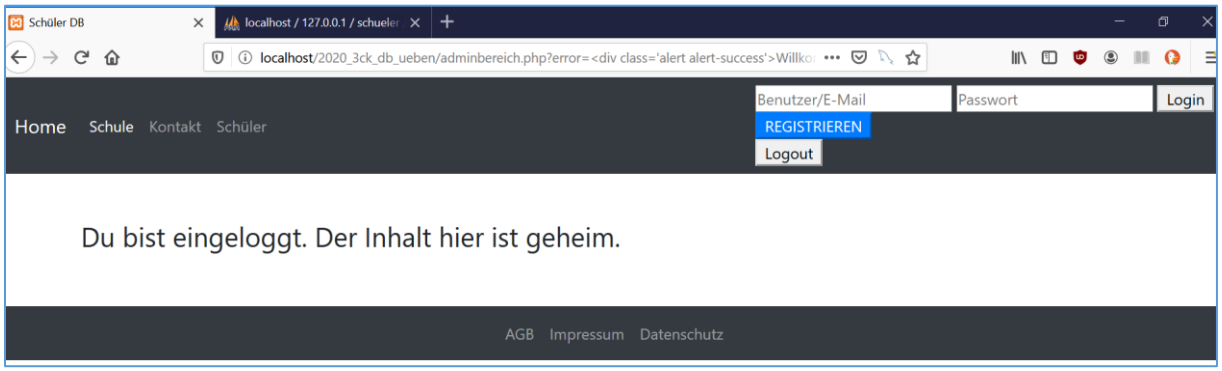
Da unsere „index.php“ durch „require“ auf die „header.php“ zugreift, muss man das nun in der „header.php“ durchführen. Der Vorteil dieser „require“-Varainte ist, dass bei mehreren Seiten, die alle diese Teile „einbeziehen (=require)“, das nur einmal gemacht werden muss. In unserem Fall dann nicht mehr extra in „adminbereich.php“, da diese ja auch auf die „header.php“ zugreift.

Schreibe vor der ersten Zeile noch diesen Code: Zeile 1-3:



Fertig und testen.

Wenn man sich korrekt einloggt, kommt man in den Adminbereich hinein und hat auch noch eine positive „willkommen“-Meldung in der URL und als grüne Alert-Meldung.



Beachte:

Wurde in der Datenbank mittels „einfügen“ händisch was eingetragen, passt das nicht mit den gehashten Passwörtern zusammen. Diese daher löschen.

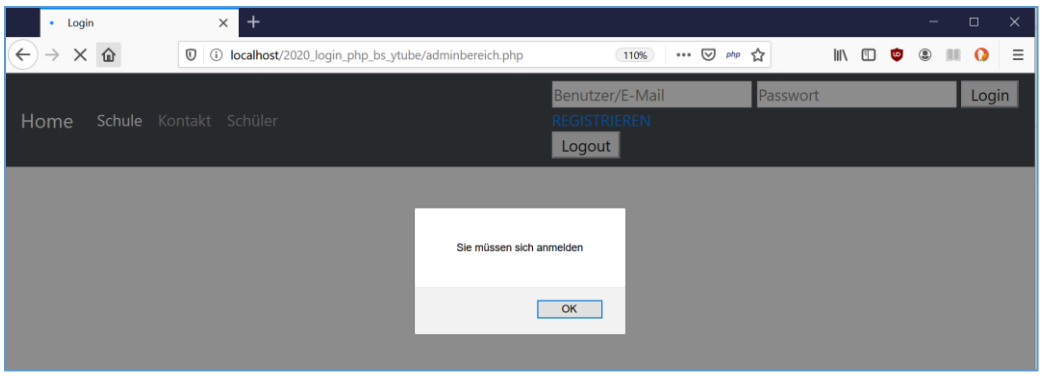
5)NUR eingeloggte User sehen alles:

5a)Adminbereich absichern bzw. alle „internen“-Seiten absichern

Man könnte aber die „adminbereich.php“-Seite auch so öffnen, indem man direkt „adminbereich.php“ eingibt. Daher muss man die Session hier mit einbeziehen und mit einer IF-Anfrage gleich am Beginn der Seite checken, ob eine erstellte „Session-Variable“ existiert, und die gibt es nur, wenn man erfolgreich eingeloggt hat. Wenn nicht wird man sofort wieder aus der Seite geschmissen.

Das betrifft auch alle anderen Seiten, die nur zu sehen sein sollen, wenn man „eingeloggt“ ist.

Ziel:
Alert-Fenster als Warnung und nach dem Kick auf „OK“ kommt man zurück zur Index-Seite.



Öffne die Seite „adminbereich.php“.

Dafür muss man im Hauptbereich die im header gestartete SESSION mit einer IF weiterführen und festlegen, was passiert, wenn eine der im Login angelegten Session-Variablen vorhanden ist (hier z.B. „userId“):

- Bei „true“: man sieht den kompletten Inhalt des Bereiches
- Bei „false“ (else) wird ein JavaScript Alert-Fenster angezeigt und man wird auf die Index-Seite zurück gesendet.

```
15          <?php
16            require "header.php";
17          ?>
18
19      <?php
20      if (isset($_SESSION['userId'])) {
21      ?>
22          <div class="container">
23          <br><br>
24          <h3>Du bist eingeloggt. Der Inhalt hier ist geheim.</h3>
25          </div>
26      <?php
27      }
28      else {
29              //hier folgt JavaScript im echo-Code
30              echo '<script>';
31              echo 'alert("Sie müssen sich anmelden");';
32              echo 'window.location = "index.php"';
33              echo '</script>';
34          }
35      ?>
```

Erklärung:

Der „else“-Bereich ermöglich im „echo“ den Betrieb nicht nur von HTML, sondern auch von JavaScript.
Der Code für die Umleitung auf die „index.php“ lautet hier „window.location“. Beachte dabei den Wechsel von einfachen und doppelten Anführungszeichen.
Der direkte Code, wie wir es bis jetzt in PHP durchgeführt haben, nämlich „Location:“ würde hier, direkt nach JavaScript nicht durchkommen und somit nicht ausgeführt. Daher muss man nicht nur das „alert“ hier ausführen, sondern eben auch die Umleitung.

Code für die else:

```
<?php
}
else {
    //hier folgt JavaScript im echo-Code
    echo '<script>';
    echo 'alert("Sie müssen sich anmelden");';
    echo 'window.location = "index.php"';
    echo '</script>';
  }
?>
```

5b)Nur wenn man eingeloggt ist: Test auf der index.php

Daher kann man das erfolgreiche Einloggen auch auf der „index.php“ zeigen bzw. überprüfen. Nur wenn man eingeloggt ist, wird etwas gezeigt, sonst nicht. Das ist zur Sicherheit von der SESSION abhängig.

Wenn man eingeloggt ist, soll eine andere Nachricht gezeigt werden, als wenn man nicht eingeloggt ist.

Öffne die index.php.

Füge noch im „container“ einen neuen PHP-Bereich ein, der wie beim Adminbereich oben, eine IF-Abfrage mit der SESSION zeigt. Die Ausgabe ist einerseits das eine, andererseits das andere „echo“.

```
29          <br><br>
30          <?php
31              if (isset($_SESSION['userId']))
32              {
33                  echo '<p>Du bist eingeloggt.</p>';
34              }
35              else
36              {
37                  echo '<p>Du bist nicht eingeloggt.
                    </p>';
38              }
39          ?>
40      </div>  <!--ende container-->
```

6)Buttons je nach Login-Zustand anzeigen lassen

Dafür kann man den PHP-Code nutzen, der gerade in der index.php gezeigt worden ist. Er unterscheidet mittels IF-Anweisung, ob jemand eingeloggt ist oder nicht.

Öffne die header.php und die index.php.

- Suche die passende Position für die IF-Verzweigung, nämlich direkt vor der <form> vom Login.
- Dann kopiere den kompletten PHP-Teil mit der IF-Abfrage aus der index und füge sie hier ein
- Danach sind noch ein paar Änderungen zu erstellen

```
35      </ul>
36      <div class="ml-auto">
37
38          <?php
39              if (isset($_SESSION['userId']))
40              {
41                  echo '<p>Du bist eingeloggt.</p>';
42              }
43              else
44              {
45                  echo '<p>Du bist nicht eingeloggt.</p>';
46              }
47          ?>
48
49          <form action="includes/login.php" method="post">
50          <input type="text" name="mailuid" placeholder="Benutzer/E-Mail">
```

In die obere IF-Anweisung, wenn true, soll der Logout Button hineingeschoben werden. Denn ist man eingeloggt, soll der LOGOUT-Button sichtbar sein. In das **„echo“** wird die ganze <form> vom Logout geschoben: echo – nicht vergessen!

```
37      <div class="ml-auto">
38      <?php
39          if (isset($_SESSION['userId']))
40          {
41          echo '<form action="includes/logout.php" method="post">
42          <button type="submit" name="logout-submit">Logout</button>
43          </form>';
44          }
45          else
```

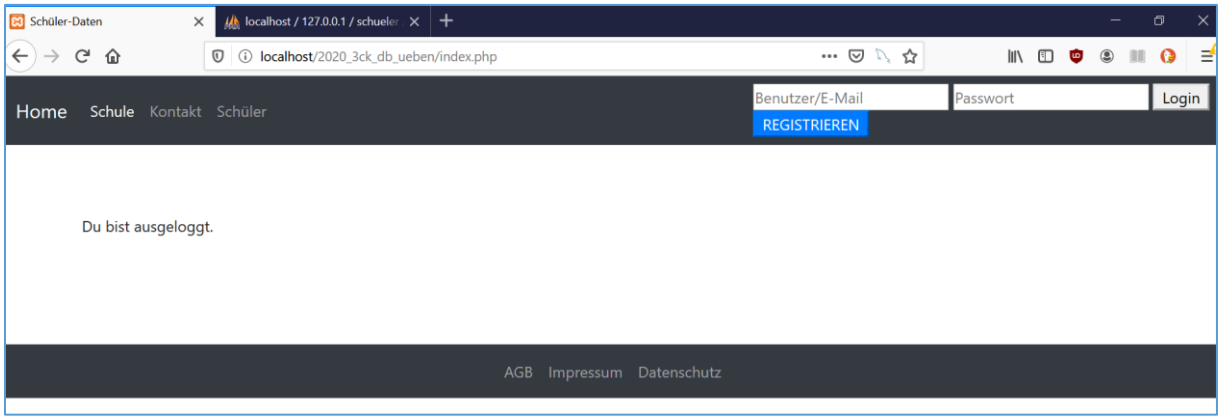
Ist man NICHT eingeloggt, also im „else“-Bereich, dann soll dort alles vom LOGIN und REGISTRIEREN sichtbar sein. Daher müssen in deren „echo“ diese beiden Bereiche geschoben sein:

```
45              else
46              {
47                  echo '<form action="includes/login.php" method="post">
48                      <input type="text" name="mailuid" placeholder="Benutzer/E-Mail">
49                      <input type="password" name="pwd" placeholder="Passwort">
50                      <button type="submit" name="login-submit">Login</button>
51                      </form>
52                      <a href="registrieren.php"><button class="button btn-
                        primary">REGISTRIEREN</button></a>';
53              }
54          ?>
```

Sieh zu, dass die Darstellung auch gut aussieht und nicht irgendwie schief eingerückt usw. ist.

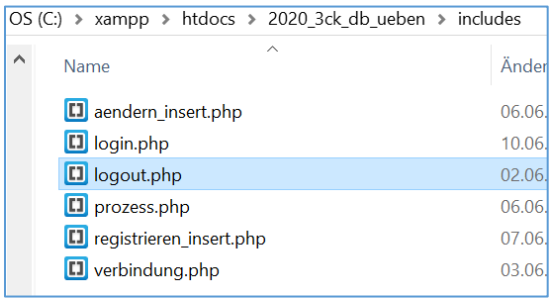
Testen:

Passt, wenn man draußen ist: kein Logout-Button sichtbar



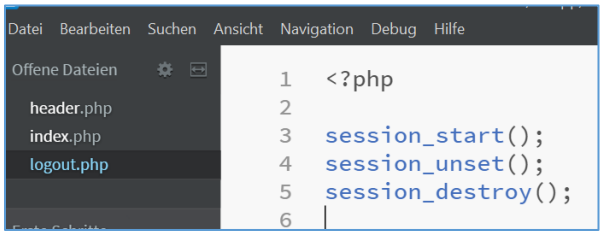
7)Logout

Dafür muss man eine neue Datei erstellen: im Ordner „includes“ 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.



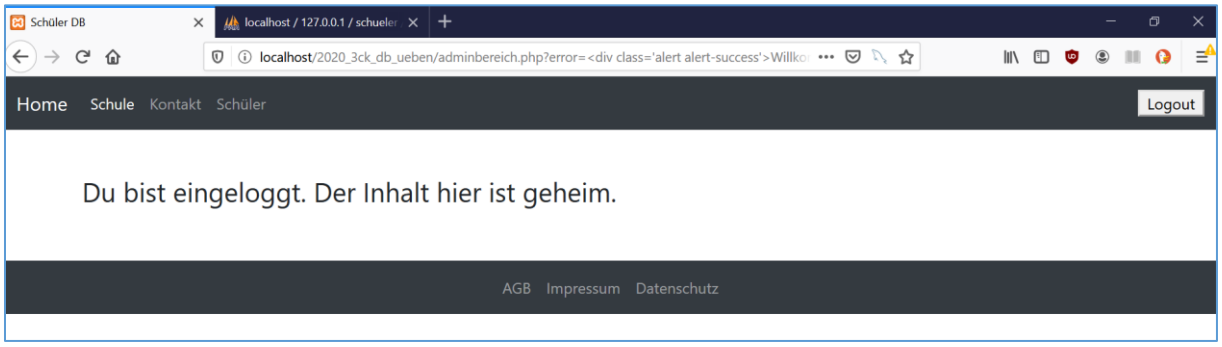
Danach sollte man zur „index.php“, also zur Startseite geleitet werden. Daher erfolgt nun die Umleitung mit „header-location“.

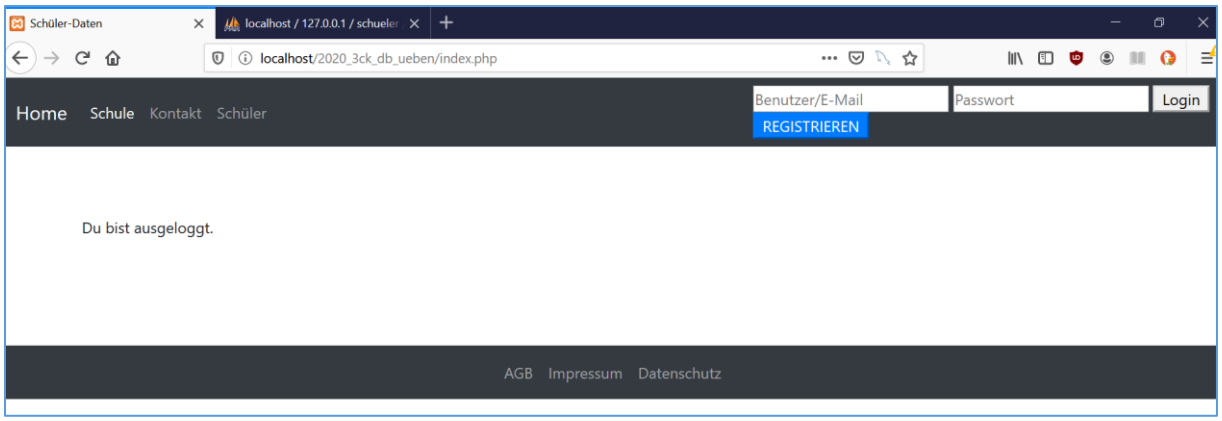
```php
<?php

session_start();
session_unset();
session_destroy();
header("Location: ../index.php");

```

Testen: passt ☺





Quelle: https://www.youtube.com/watch?v=LC9GaXkdxF8