Inhalt

1. Webdatenbankprojekt – ER-Modellierung

Projektwochenverwaltung

Als Webdatenbankprojekt nehmen wir eine Projektwochenverwaltung. Projekte haben eine Projektnummer (PNr), einen Titel, eine Beschreibung, eine minimale und maximale Teilnehmerzahl sowie die Anzahl der belegten Plätze. Für jedes Projekt muss angegeben werden für welche Klassenstufen es wählbar ist. Von Schülerinnen und Schülern werden der eindeutige Benutzername, das Passwort sowie Vorname, Nachname, Klasse und Klassenstufe gespeichert, von Lehrkräften das Lehrerkürzel, das Passwort sowie Vorname und Nachname. Lehrkräfte können Projekte leiten und ein Projekt muss von einer Lehrkraft geleitet werden. Schülerinnen und Schüler wählen drei Projekte als Erst-, Zweit- und Drittwahl. Ihnen wird nach dem Ende der Projektwahl ein Projekt zugeordnet.

Aufgaben

1. Modellieren Sie ein ER-Diagramm für die Projektwochenverwaltung und zeichnen Sie es mit *yEd Graph Editor*.

2. Geben Sie für die Beziehungen *wählt, zugeteilt* und *leitet* die KaMe- und MuMi-Fragen an und ergänzen Sie dementsprechend die Kardinalitäten und Optionalitäten im ER-Diagramm an.

   Die Entscheidung über die Kardinalität einer Beziehungstyprichtung wird über die sogenannte **KaMe-Frage** getroffen. **Kann eine** Entität des Typs A **mit mehreren** Entitäten des Typs B in Beziehung stehen?

   Ja → Kardinalität ist n
   Nein → Kardinalität ist 1

   Beispiele

Kann ein Mann mit mehreren Frauen verheiratet sein?	Nein, Kardinalität ist 1.
Kann ein Ort mehrere Schulen haben?	Ja, Kardinalität ist n.
Kann eine Schule in mehreren Orten liegen?	Nein, Kardinalität ist 1.

   Die Entscheidung über die Optionalität einer Beziehungstyprichtung wird über die sogenannte **MuMi-Frage** getroffen. **Muss eine** Entität des Typs A mit **mindestens einer** Entität des Typs B in Beziehung stehen?

   Ja → nicht optional, obligatorisch, muss-Beziehung
   Nein → optional, nicht obligatorisch, kann-Beziehung

   Beispiele:

Muss ein Mann mit mindestens einer Frau verheiratet sein?	Nein, optional, kann-Beziehung
Muss ein Ort mindestens eine Schule haben?	Nein, optional, kann-Beziehung
Muss eine Schule in mindestens einem Ort liegen?	Ja, obligatorisch, muss-Beziehung

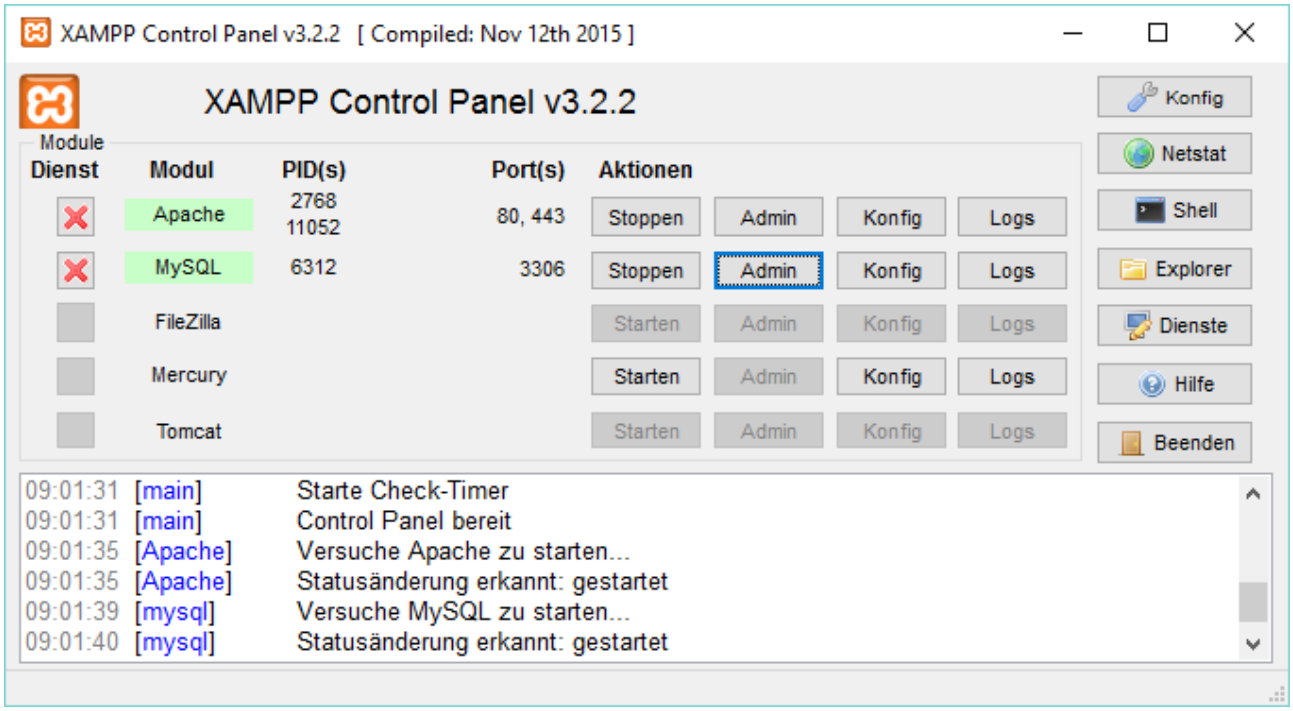
3. Überführen Sie das ER-Diagramm in das Relationenmodell.

2. Einen Web- und Datenbankserver mit XAMPP einrichten

XAMPP ist ein Softwarepaket, das den Webserver Apache, das Datenbankmanagementsystem MariaDB (vormals MySQL) sowie die Skriptsprache PHP enthält. Für die browserbasierte Verwaltung von MariaDB wird PhpMyAdmin mitgliedert. Zusätzliche Module sind der FTP-Server FileZilla, der Mailserver Mercury und Apache Tomcat zur Ausführung von Java Servlets und JavaServerPages.

2.1 Installation

Mit der PHP 7 Version von XAMPP habe ich auf Windows-Servern etliche Abstürze gehabt, weswegen ich zum Installieren einer PHP 5.6 Version rate. Rufen Sie das XAMPP-Control-Panel mit Administratorrechten auf. Sie können Apache und MySQL als Programm oder Dienst ausführen.



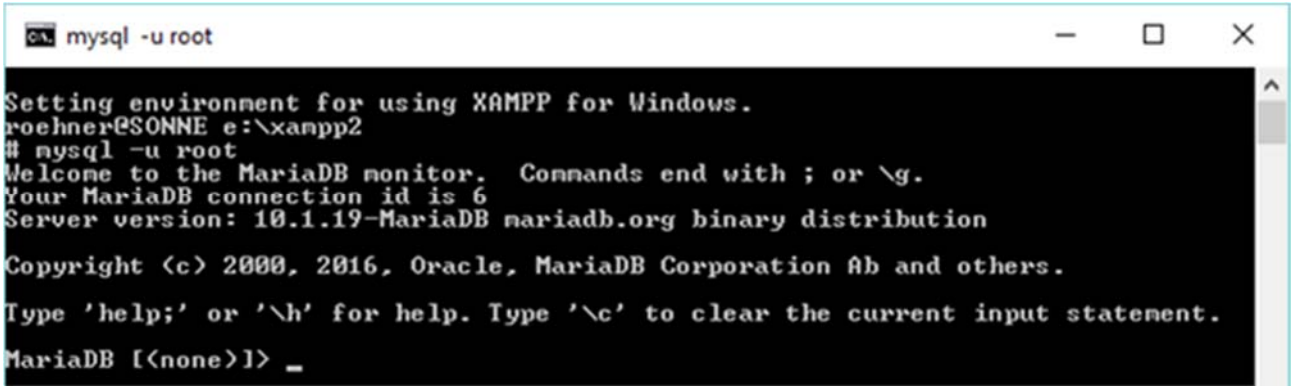
Richten Sie über 

Notepad++ als Editor für die Bearbeitung von Konfigurationsdateien ein, sowie ihren Standardbrowser ein.

2.2 MariaDB-Konsole

Einen schnellen Konsolenzugang zu MariaDB erhalten Sie über 

.



Nach der Anmeldung mit `mysql -u root` können Sie SQL-Befehle benutzen, z. B.

```
show databases;
use mysql;
show tables;
select host, user, password from user;
show character set;
show collation where charset = 'utf8';
```

SQL-Befehle können Sie über die Zwischenablage durch Anklicken mit der rechten Maustaste in das Konsolenfenster einfügen.

3. Verwaltung der Datenbank mit PhpMyAdmin

3.1 Mit PhpMyAdmin arbeiten

Zum Anlegen einer Datenbank für das Webdatenbankprojekt rufen Sie das Menü *Benutzerkonten* auf und fügen ein Konto hinzu: 

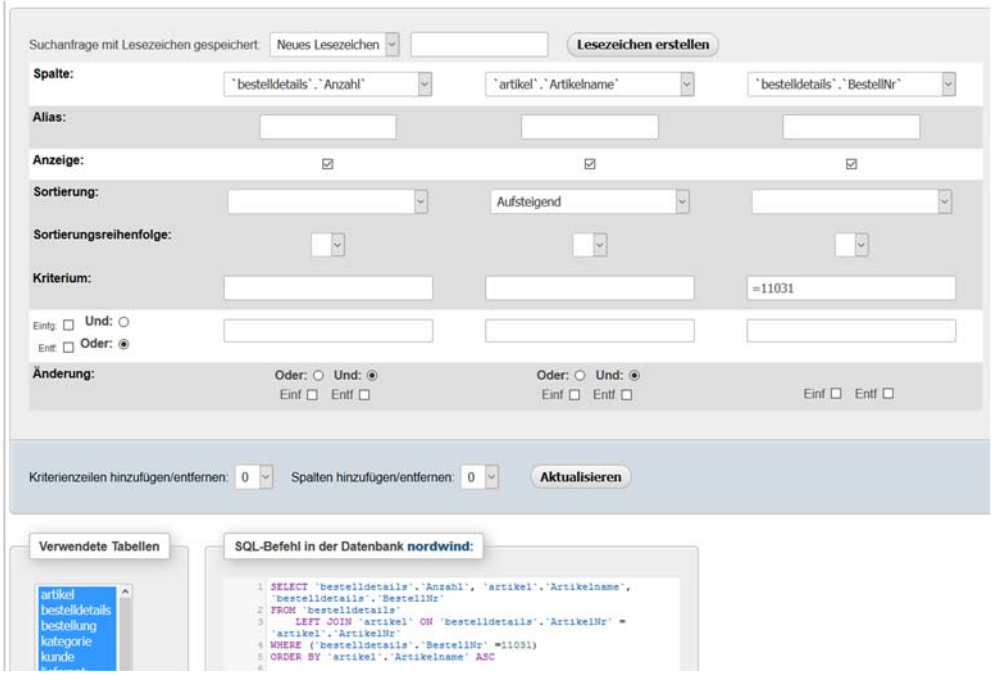
Benutzername: projektwoche, Hostname: localhost, Passwort generieren und aufschreiben. Für das Konto lassen Sie sich eine Datenbank anlegen: *Erstelle eine Datenbank mit gleichem Namen und gewähre alle Rechte*.

Erzeugen Sie in der Datenbank dann ihre Tabellen, kennzeichnen Sie die Primärschlüssel und fügen Sie Datensätze ein.

3.2 QBE – Query by Example

Achten Sie darauf, dass sich das Hauptmenü von PhpMyAdmin danach richtet, was sie im linken Baum ausgewählt haben: die Startseite  , eine Datenbank oder eine Tabelle. Wenn Sie eine Datenbank ausgewählt haben, zeigt das Menü *Abfrage*

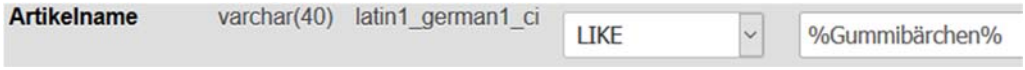
QBE finden Sie unter Abfrage. Im Bild sehen Sie die Abfrage von Anzahl, Artikelname und BestellNr für die die Bestellnummer 11031. Da die Abfrage die zwei Tabellen Artikel und Bestelldetails betrifft, wird automatisch ein Join erzeugt.



Man kann auch von der Abfrage in den Designer wechseln und dort die Abfrage erstellen.

3.3 Suche

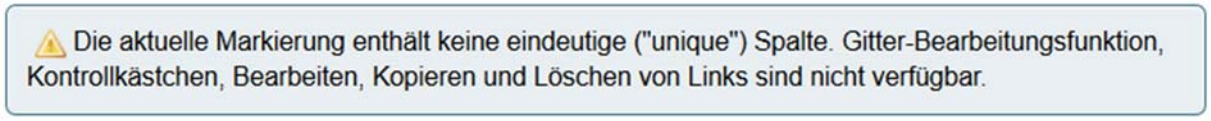
Die datenbankweite Suche können Sie einsetzen, wenn unklar ist, in welchen Tabellen ein Wert vorkommt, also insbesondere bei fremden Datenbanken mit vielen Tabellen. Bei der tabellenbezogenen Suche geben Sie den Vergleichsoperator und den Vergleichswert an. Achten Sie beim Operator *LIKE* auf die richtige Verwendung des %-Zeichens im Suchwert.



3.4 Datenbanken importieren

Legen Sie die Datenbanken *cia* und *nordwind* an und importieren Sie die jeweiligen Tabellen und Daten (cia.sql, nordwind.sql). Alle Nordwind-Tabellen haben Bearbeitungsmöglichkeiten:

Bei der cia-Tabelle fehlen diese. Ursache ist der fehlende Index.



Der Name wäre ein passender Index, der Primärschlüssel ist aber im Menü *Struktur* ausgegraut, weil der Datentyp *Text* ist.

Über Tabellenstruktur analysieren findet man heraus, dass der Name maximal 42 und die Region maximal 15 Zeichen lang ist. Ändern Sie den Datentyp von Name und Region in VARCHAR mit einer passenden Länge.

MySQL unterscheidet drei Index-Arten:

1. Primary Key – Jeder Eintrag in der Primary-Spalte muss eindeutig sein. Nur ein Primary Key je Tabelle.
2. Unique – Jeder Eintrag in der Unique-Spalte muss eindeutig sein. Mehrere Unique Keys möglich.
3. Index– Die Einträge in der Index-Spalte müssen nicht eindeutig sein.

Wählen Sie nun Name als Primärschlüssel aus. In der Anzeige erscheinen nun auch die Bearbeitungs-möglichkeiten.

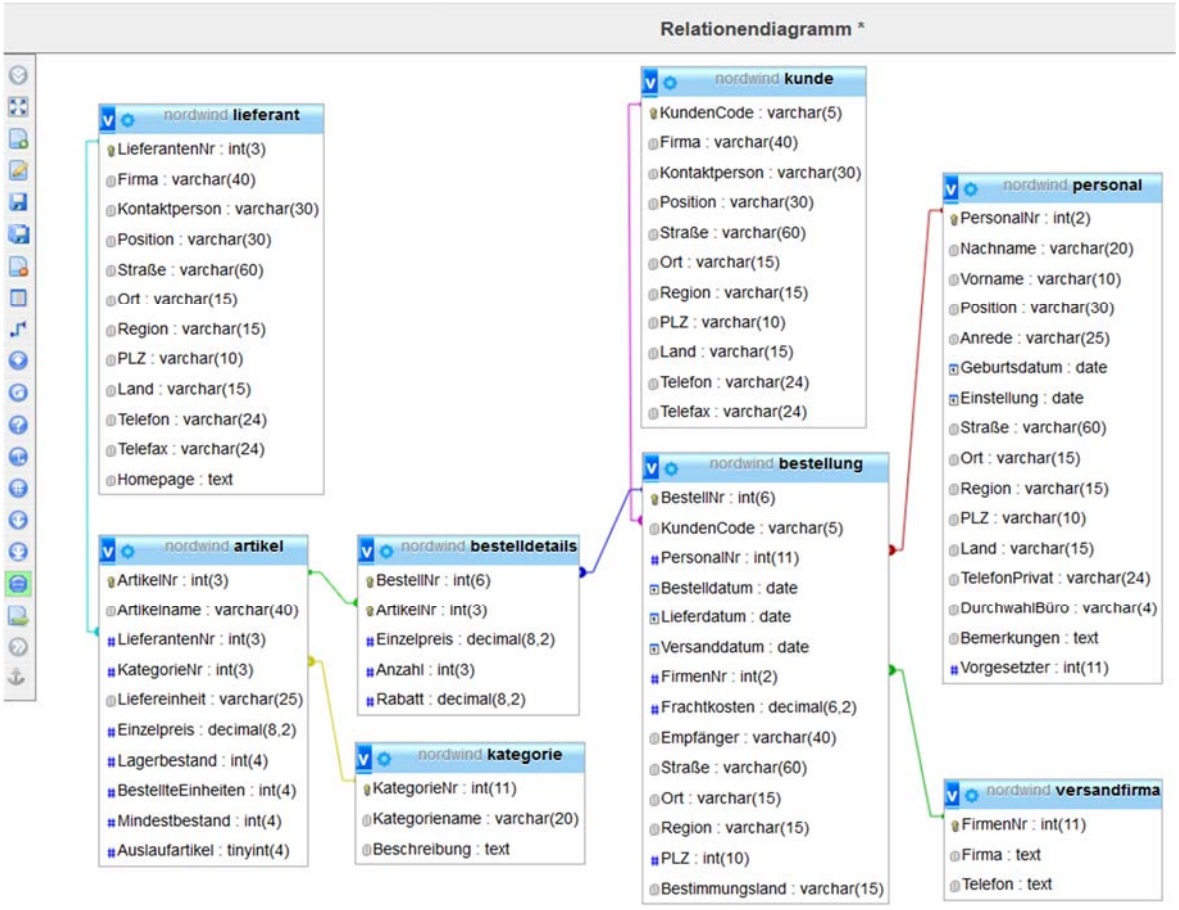
3.5 Fremdschlüssel

Lassen Sie sich die Struktur der Tabelle Bestelldetails aus der Nordwind-Datenbank anzeigen, so können Sie zur Beziehungsansicht wechseln. Hier können Sie Fremdschlüssel einer Relation eintragen. Da Fremdschlüssel für das Verständnis des Relationenmodells notwendig sind, sollte dies im Unterricht geschehen.

Außerdem wird das *Einfügen* von Datensätzen dadurch vereinfacht, dass Auswahlmöglichkeiten statt Eingabefelder angeboten werden.

3.6 Designer

Wählt man eine Datenbank aus, so steht rechts im Menü *Mehr* der Designer z  Designer zur Verfügung.



Sie können die Primärschüssel-Fremdschlüssel-Beziehungen hier anzeigen und bearbeiten.

3.7 Referentielle Integrität

Die Tabellen der Nordwind-Datenbank haben das Tabellenformat MyISAM. Um referentielle Integrität zur Verfügung zu haben, ändern wir zunächst mit SQL-Anweisungen für alle Tabellen das Format auf InnoDB:

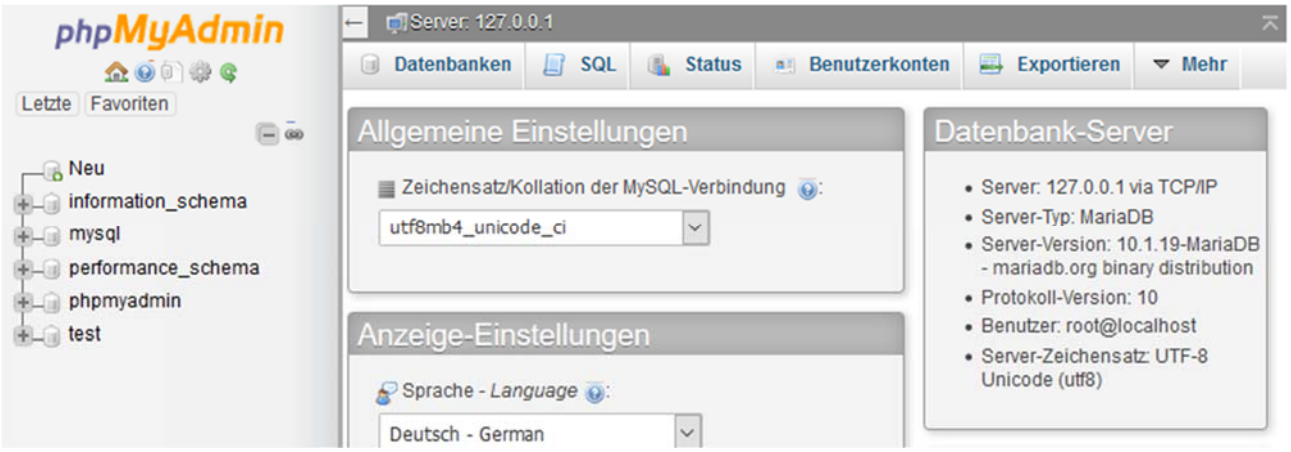
```
ALTER TABLE Artikel ENGINE=InnoDB;
```

Rufen Sie jetzt die Beziehungsansicht innerhalb des Menüs *Struktur* auf, so werden Beschränkungen durch Fremdschlüssel angezeigt. Die erste Beschränkung ON DELETE RESTRICT bedeutet, dass in der Lieferantentabelle ein Lieferant nicht gelöscht werden kann, wenn es einen Artikel mit diesem Lieferanten gibt. ON UPDATE CASCADE bedeutet, dass bei einer Änderung des Primärschlüssel LieferantenNr auch alle entsprechenden Fremdschlüssel in der Tabelle Artikel geändert werden.

Zum Herstellen dieser Beschränkungen müssen die Fremdschlüssel einen Index haben. Hierfür verwenden Sie in der Tabellenstruktur *Index.* Versuchen Sie einen Lieferanten zu löschen, von dem ein Artikel vorhanden ist. Ändern Sie in der Tabelle Kategorie eine Kategorienummer und sehen Sie sich die Auswirkungen auf die Tabelle Artikel an.

3.8 Konfiguration des Datenbankservers

Über Admin von MySQL rufen Sie PhpMyAdmin auf.



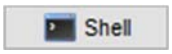
MariaDB unterstützt die Tabellenformate *MyISAM* und *InnoDB. MyISAM* war bis Version 5.5.4 das Standardformat von MySQL und führte Select-Anweisungen sehr schnell aus. Ab Version 5.5.5 ist InnoDB das Standard-Tabellenformat. Es unterstützt Transaktionen und referentielle Integrität.

Rufen Sie das Menü *Variablen* auf und stellen Sie die *storage engine* auf InnoDB ein.

Den Zeichensatz *character set server* stellen Sie auf *utf8* ein und die für die Sortierreihenfolge (ORDER BY) maßgebliche *collation (connection|database|server)* auf *utf8_unicode_ci.* (ci, engl. case insensitive)

PhpMyAdmin nutzt eine eigene Datenbank auf die es mit dem Benutzer *pma* zugreift.

3.9 Sicherheit

Rufen Sie über das XAMPP Control Panel eine Konsole 

auf und starten Sie darin einen MySQL-Client: mysql –u root. Sichern Sie nun den root-Zugang durch Vergabe eines Passworts ab. Notieren Sie sich das Passwort!

```
GRANT ALL PRIVILEGES ON * TO root@localhost IDENTIFIED BY '<passwort>';
```

Verlassen Sie mit *exit* die MySQL-Konsole und melden Sie sich erneut an, diesmal mit dem zusätzlichen Parameter –p, wodurch das Passwort abgefragt wird:

```
mysql –u root –p
```

Nach der Passwortvergabe funktioniert der Zugang mittels PhpMyAdmin nicht mehr. Im XAMPP Control Panel können Sie über den Button 

von Apache die Konfigurationsdatei phpMyAdmin (config.inc.php) öffnen. Wenn Sie die Authentifizierungsmethode von *config* auf *cookie* und *AllowNoPassword* auf *false* umstellen, erscheint beim Aufruf von PhpMyAdmin ein Login-Fenster.

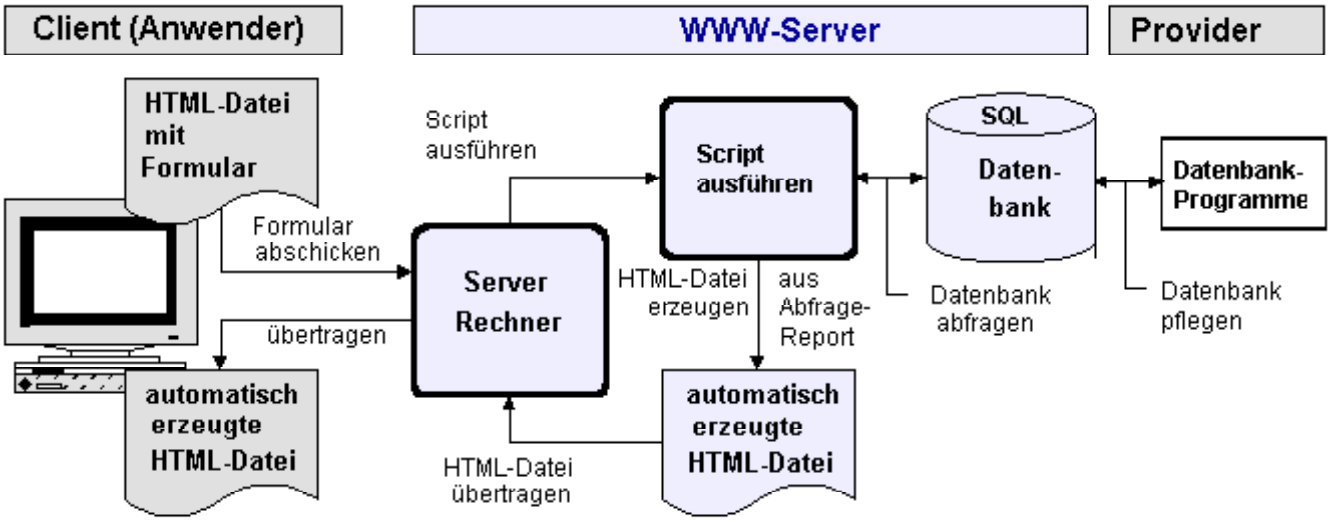

```
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['AllowNoPassword'] = false;
```

Vergleichen Sie den linken Datenbank-Baum, wenn Sie sich als *root* oder Benutzer *projektwoche* ihres Webdatenbankprojekts anmelden.

4. Mit PHP-Skripten eine Webdatenbankanwendung realisieren

PHP ist eine serverbasierte Skriptsprache, die zur Erstellung dynamischer Webseiten verwendet wird. Ca. 80 % aller Websites setzen PHP ein und bei den meisten Webhostern ist PHP vorinstalliert. Bekannte Webanwendungen wie *WordPress*, *Joomla* und *Moodle* sind mit PHP realisiert. Die Syntax von PHP ist an Java angelehnt. Ein wesentlicher Unterschied zu Java ist allerdings die schwache Typisierung. Variablen werden nicht deklariert, sondern bei der ersten Benutzung angelegt, weswegen der Datentyp implizit von PHP bestimmt wird. Da PHP auf Groß-/Kleinschreibung achtet, sind Tippfehler in Variablennamen manchmal schwer zu finden. Ein deutsches PHP-Handbuch steht unter http://de.php.net/manual/de/ zur Verfügung.

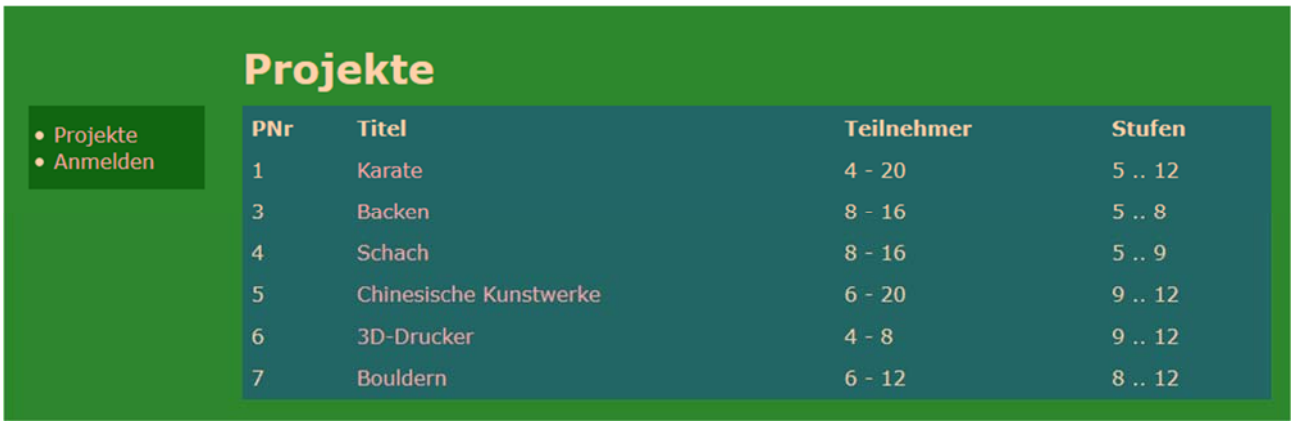
Das Prinzip dynamischer Webseiten ist im Bild dargestellt.



Der Anwender füllt ein HTML-Formular aus und schickt es an den Webserver. Dieser führt das vom Formular angegebene Skript (PHP, ASP, JSP, ...) aus. Im Skript kann mittels SELECT auf die Daten einer SQL-Datenbank zugegriffen werden. Mit der Antworttabelle baut das Skript die Ergebnisseite dynamisch auf. Diese wird vom Server als Antwort auf die Anfrage an den Anwender übertragen. Der Provider pflegt seine Datenbank mit Applikationen, welche aus Gründen der Datensicherheit nur intern zur Verfügung stehen.

4.1 Eine erste PHP-Seite

Die Startseite *index.php* der Projektwochenverwaltung soll eine Tabelle der Projekte anzeigen, wobei die Projekttitel Verweise auf eine detaillierte Darstellung des jeweiligen Projekts sind.



Die Datei index.php enthält folgenden Inhalt:

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04   <title>Projektwoche</title>
05   <meta charset="utf8">
06   <link rel="stylesheet" type="text/css" href="css/formate.css">
07 </head>
08 <body>
09   <div id="menue">
10     <ul>
11       <li><a href="index.php">Projekte</a></li>
12       <li><a href="anmelden.php">Anmelden</a></li>
13     </ul>
14   </div>
15   <div id="inhalt">
16   <h1>Projekte</h1>
17   <table><tr><th>PNr</th><th>Titel</th><th>Teilnehmer</th><th>Stufen</th></tr>
18 <?php
19   require('utils.php');
20   DatenbankAnmelden();
21
22   $sql = 'SELECT * FROM Projekt;';
23   $result = $mysqli->query($sql);
24
25   while ($row = $result->fetch_assoc()) {
26     echo '<tr><td>'.$row['PNr'].'</td>'.
27        '<td><a href="projekt.php?PNr='.$row['PNr'].'">'.$row['Titel'].'</a></td>'.
28        '<td>'.$row['Tmin'].' - '.$row['Tmax'].'</td>'.
29        '<td>'.$row['Kmin'].' .. '.$row['Kmax'].'</td></tr>'.PHP_EOL;
30   }
31 ?>
32   </table>
33   </div>
34 </body>
35 </html>
```

In den Zeilen 01 bis 17 steht statischer HTML-Code. Ab Zeile 18 beginnt mit <?php das eigentliche PHP-Skript, das in Zeile 31 mit ?> endet. In Zeile 19 wird die Datei *utils.php* mit allgemein nützlichen Funktionen eingebunden. Darin befindet sich die in Zeile 20 verwendete Funktion *DatenbankAnmelden().* In Zeile 22 erhält die Variable $sql den Wert 'SELECT * FROM Projekte'. Diese sql-Anweisung wird mittels `$mysqli->query($sql);` auf der Datenbank ausgeführt. Das Ergebnis ist ein mysqli_result-Objekt, das in der Variablen $result gespeichert wird. Dieses Objekt verfügt über die Methode *fetch_assoc*, mit der man den nächsten Datensatz des Ergebnisses als assoziatives Array erhält und in der Variablen $row speichert. Bei einem assoziativen Array erfolgt der Zugriff auf ein Element nicht über einen numerischen Index, sondern über einen String. Innerhalb der while-Schleife wird der Datensatz $row als HTML-Tabellenzeile ausgegeben. Der Titel wird als Verweis auf projekt.php?PNr=$row['PNr'] realisiert.

4.2 Die Datenbankverbindung

Die Skriptdatei utils.php enthält die Funktion *DatenbankAnmelden*(), mit welcher die Verbindung zur Datenbank hergestellt wird.

```
01 <?php
02   $mysqli = null;
03
04   function DatenbankAnmelden() {
05     global $mysqli;
06
07     error_reporting(E_ALL);
08     ini_set ('display_errors', 'On');
09
```

```
10      $host     = 'localhost';
11      $user     = 'projektwoche';
12      $password = '4HFQ70nwm1T7JM01';
13      $database = 'projektwoche';
14
15      $mysqli = new mysqli($host, $user, $password, $database);
16
17      if ($mysqli->connect_error)
18        die ('Keine Verbindung! '.$mysqli->connect_error);
19      $mysqli->set_charset('utf8');
20    }
```

Dazu wird in Zeile 15 mit dem Konstruktor der Klasse *mysqli* ein Verbindungsobjekt erstellt und in der Variablen $mysqli gespeichert. Schlägt die Verbindung fehl, wird eine Fehlermeldung ausgegeben und das Skript abgebrochen. Legt man eine Variable innerhalb einer Funktion an, so gilt sie lokal. Soll eine globale Variable genutzt werden, muss dies mit *global* angegeben werden.

4.3 Arrays und assoziative Arrays

Hat man mit `$result = $mysqli->query($sql);` eine Abfrage durchgeführt, so kann man die Datensätze in unterschiedlichen Formen abrufen.

$row = $result->fetch_row(); liefert einen Datensatz als Array:
Array ([0] => 1 [1] => Karate [2] => Der Karatekurs für Anfänger [3] => 4 [4] => 20 [5] => 0 [6] => 5 [7] => 12 [8] =>)

$row = $result->fetch_assoc(); liefert einen Datensatz als assoziatives Array**:**
Array ([PNr] => 3 [Titel] => Backen [Beschreibung] => Backe, backe Kuchen! [Tmin] => 8 [Tmax] => 16 [Plätze] => 0 [Kmin] => 5 [Kmax] => 8 [Kürzel] =>)

$row = $result->fetch_array(); liefert einen Datensatz als Array und assoziatives Array**:**
Array ([0] => 4 [PNr] => 4 [1] => Schach [Titel] => Schach [2] => Hol dir dein Schachdiplom [Beschreibung] => Hol dir dein Schachdiplom [3] => 8 [Tmin] => 8 [4] => 16 [Tmax] => 16 [5] => 0 [Plätze] => 0 [6] => 5 [Kmin] => 5 [7] => 9 [Kmax] => 9 [8] => [Kürzel] =>)

Bei der Verwendung eines assoziativen Arrays muss man zwar mehr schreiben, aber man macht sich unabhängig von späteren Erweiterungen einer Tabelle und produziert Code, der besser gewartet werden kann.

4.4 Einiges zu PHP

Ein Variablenname beginnt immer mit $. Strings können mit einfachen oder doppelten Anführungszeichen geschrieben werden. Innerhalb von doppelten Anführungszeichen werden Variablen ausgewertet, z. B.
`echo "Dein Name: $Vorname $Nachname".` Strings können mit dem Punkt verbunden werden, z. B.
`echo '<tr><td>'.$row['PNr'].'</td>'.`

Für die Fehlersuche kann man die Ausgabeanweisung *echo* benutzen. Beispiele:

```
echo $Anzahl;                  // gibt den Wert der Variablen $Anzahl aus.
echo "Anzahl: " . $Anzahl;     // beschriftet die Ausgabe der Anzahl
```

Detailliertere Informationen zu einer Variablen erhält man mit *var_dump()*: var_dump($Anzahl);

Eine Array-Variable wie z. B. $row kann man mit *print_r()* ausgeben. print_r($row);

PHP stellt derzeit drei Schnittstellen zum Zugriff auf MySQL/MariaDB-Datenbanken bereit:

- mysql - eine veraltete prozedurale Schnittstelle, nicht mehr in PHP 7 vorhanden
- mysqli - eine objektorientierte und prozedurale Schnittstelle ab PHP 5.0
- PDO_MySQL - eine streng objektorientierte Schnittstelle ab PHP 5.1

In dieser Ausarbeitung wird die mysqli-Schnittstelle verwendet.

4.5 Anmeldung

Möchte man Daten eingeben oder verändern, muss man sich bei der Projektwochenverwaltung anmelden.

Das Anmeldeformular kann als reines HTML-Dokument gestaltet werden:

```
<!DOCTYPE html>
<html>
<head>
  <title>Anmeldung</title>
  <meta charset="utf8">
  <link rel="stylesheet" type="text/css" href="css/formate.css">
</head>
<body>
  <div id="menue">
    <ul>
      <li><a href="index.php">Projekte</a></li>
      <li><a href="anmelden.php">Anmelden</a></li>
    </ul>
  </div>
  <div id="inhalt">
  <h1>Anmeldung</h1>
  <form action="anmelden.php" method="post" style="width: 400px">
    <p>
      <label>Benutzername</label>
      <input type="text" name="Benutzername" class="feld">
    </p>
    <p>
      <label>Passwort</label>
      <input type="password" name="Passwort" class="feld">
    </p>
    <p style="text-align: center">
      <button name="Action" value="Anmelden">Anmelden</button>
    </p>
  </form>
</div>
</body>
</html>
```

Im action-Attribut des form-Elements wird angegeben, wohin die Eingabedaten geschickt werden sollen. Auf dem Bildungsserver ist bei http://arbeitsplattform.bildung.hessen.de/fach/informatik/formulare.html angegeben, dass man mit action="http://www.schulserver.hessen.de/uebung/frmact.php" erkunden kann, welche Daten wie übertragen werden. Das Skript *frmact.php* gibt die Daten in aufbereiteter Form so aus:

Benutzername = admin
Passwort = Poli3452
Action = Anmelden

In Rohform werden die Daten im assoziativen Array $_POST übergeben, da wir method="post" angeben haben:

Array ([Benutzername] => admin [Passwort] => Poli3452 [Action] => Anmelden)

Man kann das Anmeldeformular *anmelden.html* nennen und das zugehörige Skript *anmelden.php*. Günstiger ist es jedoch, das Skript und das Formular in einer Datei zusammen zu fassen. In diesem Fall kommt das Skript vor das Formular:

```
01 <?php
02   require('utils.php');
03   DatenbankAnmelden();
04
05   if (isset($_POST['Action'])) {
06     $Benutzername = getPost('Benutzername');
07     $Passwort = getPost('Passwort');
08     $IstLehrer = strlen($Benutzername) < 6;
09
10     if ($IstLehrer) {
11       $sql = "SELECT * FROM Lehrer WHERE Kürzel = '$Benutzername'
12               AND Passwort = '$Passwort'";
13       $result = $mysqli->query($sql);
14       if ($result->num_rows == 1) {
15         session_start();
16         $_SESSION['Benutzername'] = $Benutzername;
17         $_SESSION['IstLehrer'] = true;
18         header('Location: lehrer.php');
19       }
20     } else {
21       $sql = "SELECT * FROM Schüler WHERE Benutzername = '$Benutzername'
22               AND Passwort = '$Passwort'";
23       $result = $mysqli->query($sql);
24       if ($result->num_rows == 1) {
25         session_start();
26         $_SESSION['Benutzername'] = $Benutzername;
27         $_SESSION['IstLehrer'] = false;
28         header('Location: schueler.php');
29       }
30     }
31   }
32 ?>
```

In Zeile 5 wird geprüft, ob die Variable $_POST['Action'] existiert. Dies ist beim ersten Aufruf des Formulars nicht der Fall, weswegen in diesem Fall das Skript abgearbeitet und das nachfolgende HTML-Formular angezeigt wird. Wenn man das Formular ausgefüllt und abschickt, so hat die Variable $_POST['Action'] den Wert 'Anmelden'. In den Zeilen 6 und 7 werden dann der Benutzername und das Passwort eingelesen. Dazu wird die Hilfsfunktion *getPost* aus der Datei *utils.php* verwendet:

```
function getPost($Name) {
  global $mysqli;
  if (isset($_POST[$Name]))
    return $mysqli->real_escape_string($_POST[$Name]);
  else
    return null;
}
```

getPost('Benutzername') holt sich den Benutzernamen aus dem assoziativen Array $_POST und wendet darauf die Funktion real_escape_string an, welche spezielle Zeichen so kodiert, sodass das Ergebnis gefahrlos in einer SQL-Anweisung genutzt werden kann. Aus der Länge des Benutzernamens ergibt sich, ob sich eine Lehrkraft oder ein Schüler anmeldet. Dementsprechend wird eine SQL-Anweisung zur

Überprüfung von Benutzernamen und Passwort für die Lehrer bzw. Schüler-Tabelle gebildet. Wird genau ein Datensatz gefunden, wird mittels `header('Location: lehrer.php');` das Lehrer-Skript aufgerufen.

4.6 Sessions und Cookies

Da Schüler und Lehrer unterschiedliche Rechte in der Projektwochenverwaltung haben, müssen wir uns merken, wer sich angemeldet hat und zwar über mehrere Aufrufe von Seiten der Projektwochenverwaltung. Für diesen Zweck verwendet man eine *Session*. Mit Sessions kann man Daten benutzerbezogen über mehrere Aufrufe von Seiten hinweg speichern. Dazu erzeugt PHP beim Anlegen einer Session eine Session-ID und übermittelt diese im Antwortkopf an den Browser:

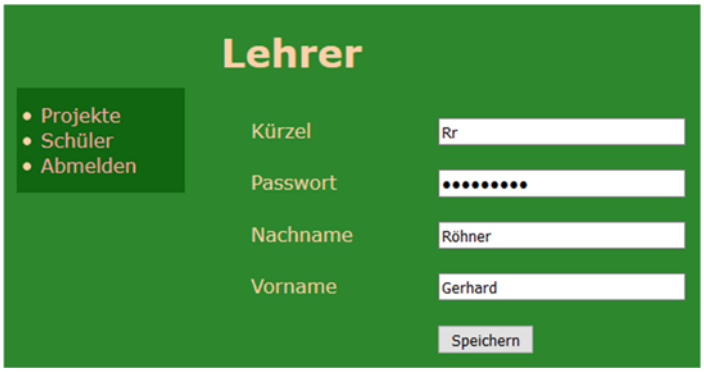
Set-Cookie PHPSESSID=oh742sipliqgefudkm;

Der Browser speichert die Session-ID unter der zugehörigen Domain und ihrem Namen. Ruft der Browser die nächste Webseite auf, so schickt er in der Anfrage die Session-ID mit. Der Server weiß dann, dass die Anfrage vom dem Benutzer kommt, dem diese Session-ID gehört. Über die globale $_SESSION-Variable kann man Werte, wie z.B. $Benutzernamen und $IstLehrer speichern und beim Aufruf des nächsten Skripts wieder abrufen.

session_start() erzeugt eine neue Session oder nimmt die aktuelle wieder auf, falls in der POST-Anfrage ein Cookie übermittelt wurde. Dabei werden die zur Session gehörenden Daten aus der Session-Datei in die $_SESSION-Variable geladen.

4.7 Lehrerdaten ändern

Wenn eine Lehrkraft sich anmeldet, sollen seine bzw. ihre Daten angezeigt und geändert werden können.



Das Skript *lehrer.php* kann wie folgt realisiert werden:

```php
<?php
  require('utils.php');
  DatenbankAnmelden();
  session_start();
  if (!$_SESSION['IstLehrer'])
    header('Location: index.php');
  if (getPost('Action') == 'Speichern') {
    $Passwort = getPost('Passwort');
    $Nachname = getPost('Nachname');
    $Vorname = getPost('Vorname');
    $sql = "UPDATE Lehrer
            SET Passwort = '".$Passwort."',
                Nachname = '".$Nachname."',
                Vorname  = '".$Vorname."'
            WHERE Kürzel = '".$_SESSION['Benutzername']."'";
    $mysqli->query($sql);
  }

  $sql = "SELECT * FROM Lehrer WHERE Kürzel = '".$_SESSION['Benutzername']."'";
  $result = $mysqli->query($sql);
  $row = $result->fetch_assoc();
?>

<!DOCTYPE html>
<html>
<head>
  <title>Lehrer</title>
  <meta charset="utf8">
  <link rel="stylesheet" type="text/css" href="css/formate.css">
</head>
<body>
  <div id="menue">
    <ul>
      <li><a href="index.php">Projekte</a></li>
      <li><a href="schueler.php">Schüler</a></li>
      <li><a href="abmelden.php">Abmelden</a></li>
    </ul>
  </div>
  <div id="inhalt">
  <h1>Lehrer</h1>
  <form action="lehrer.php" method="post" style="width: 400px">
    <p>
      <label>Kürzel</label>
    <input type="text" name="Kürzel" class="feld" value="<?php echo $row['Kürzel']; ?>" readonly>
    </p>
    <p>
      <label>Passwort</label>
    <input type="password" name="Passwort" class="feld" value="<?php echo $row['Passwort']; ?>">
    </p>
    <p>
      <label>Nachname</label>
     <input type="text" name="Nachname" class="feld" value="<?php echo $row['Nachname']; ?>">
    </p>
    <p>
      <label>Vorname</label>
     <input type="text" name="Vorname" class="feld" value="<?php echo $row['Vorname']; ?>">
    </p>
    <p style="text-align: center">
      <button name="Action" value="Speichern">Speichern</button>
    </p>
  </form>
</div>
</body>
</html>

```

In Zeile 4 wird die Session geöffnet. In Zeile 5 wird geprüft, ob die Session-Variable IstLehrer den Wert false hat. In diesem Fall wird die Startseite *index.php* aufgerufen. Die lehrer.php-Seite kann also nur aufgerufen werden, wenn man sich zuvor als Lehrkraft angemeldet hat.

Wenn im Formular von *lehrer.php* der Speichern-Button gedrückt wurde, werden in den Zeilen 8 bis 10 Passwort, Nachname und Vorname eingelesen, in eine SQL-Update-Anweisung eingebaut, die dann ausgeführt wird.

Ab Zeile 19 werden die Daten des angemeldeten Benutzers abgerufen und in der Variablen $row gespeichert. Die vorhandenen Werte für Kürzel, Passwort, Nachname und Vorname können darüber in das Formular eingetragen werden.

4.8 Projektdaten bearbeiten – Einsatz von $_GET

Die Projekte werden auf der Startseite als Liste aufgeführt. Die Projekttitel sind dabei als Verweise realisiert. Der Verweis *projekt.php?PNr=3* enthält neben dem aufzurufenden Skript *projekt.php* noch den Parameter *PNr=3*. Im Unterschied zur Post-Methode eines Formulars werden hier die an das Skript zu übertragenden Daten in der URL angegeben. Im Skript greift man nicht über den $_POST-Array sondern über den $_GET-Array auf solche Daten zu.

```
001 <?php
002   require('utils.php');
003   DatenbankAnmelden();
004   session_start();
005   if (getSession('Benutzername') == 'admin' && getPost('Action') == 'Speichern') {
006     $PNr = getPost('PNr');
007     $Titel = getPost('Titel');
008     $Beschreibung = getPost('Beschreibung');
009     $Tmin = getPost('Tmin');
010     $Tmax = getPost('Tmax');
011     $Plaetze = getPost('Plätze');
012     $Kmin = getPost('Kmin');
013     $Kmax = getPost('Kmax');
014     $Kuerzel = getPost('Kürzel');
015
016     $sql = "UPDATE Projekt
017             SET Titel = '".$Titel."',
018                 Beschreibung = '".$Beschreibung."',
019                 Tmin = ".$Tmin.",
020                 Tmax = ".$Tmax.",
021                 Plätze = ".$Plaetze.",
022                 Kmin = ".$Kmin.",
023                 Kmax = ".$Kmax.",
024                 Kürzel = '".$Kuerzel."'
025             WHERE PNr = '".$PNr."'";
026     $mysqli->query($sql);
027   }
028
029   if (getPost('Action') == 'Speichern')
030     $PNr = getPost('PNr');
031   else
032     $PNr = getGet('PNr');
033
034   $sql = "SELECT * FROM Projekt WHERE PNr = ".$PNr;
035   $result = $mysqli->query($sql);
036   $row = $result->fetch_assoc();
037 ?>
038
039 <!DOCTYPE html>
040 <html>
041 <head>
042   <title>Projekt</title>
043   <meta charset="utf8">
044   <link rel="stylesheet" type="text/css" href="css/formate.css">
```

```
045 </head>
046 <body>
047   <div id="menue">
048     <ul>
049       <li><a href="index.php">Projekte</a></li>
050       <li><a href="schueler.php">Schüler</a></li>
051       <?php
052         if (isset($_SESSION['IstLehrer']))
053           echo '<li><a href="abmelden.php">Abmelden</a></li>';
054         else
055           echo '<li><a href="anmelden.php">Anmelden</a></li>';
056       ?>
057     </ul>
058   </div>
059   <div id="inhalt">
060   <h1>Projekt</h1>
061   <form action="projekt.php" method="post" style="width: 400px">
062     <p>
063       <label>Projektnummer</label>
064      <input type="text" name="PNr" class="feld" value="<?php echo $row['PNr']; ?>" readonly>
065     </p>
066     <p>
067       <label>Titel</label>
068      <input type="text" name="Titel" class="feld" value="<?php echo $row['Titel']; ?>">
069     </p>
070     <p>
071       <label>Beschreibung</label>
072     <textarea cols="60" rows="5" name="Beschreibung"><?php echo $row['Beschreibung']; ?></textarea>
073     </p>
074     <p>
075       <label>minimale Teilnehmerzahl</label>
076     <input type="number" name="Tmin" class="zahl" min="4" max="30" value="<?php echo $row['Tmin']; ?>">
077     </p>
078     <p>
079       <label>maximale Teilnehmerzahl</label>
080     <input type="number" name="Tmax" class="zahl" min="4" max="30" value="<?php echo $row['Tmax']; ?>">
081     </p>
082     <p>
083       <label>Plätze</label>
084     <input type="text" name="Plätze" class="zahl" value="<?php echo $row['Plätze']; ?>" readonly>
085     </p>
086     <p>
087       <label>minimale Klassenstufe</label>
088     <input type="number" name="Kmin" class="zahl" min="5" max="13" value="<?php echo $row['Kmin']; ?>">
089     </p>
090     <p>
091       <label>maximale Klassenstufe</label>
092     <input type="number" name="Kmax" class="zahl" min="5" max="13" value="<?php echo $row['Kmax']; ?>">
093     </p>
094     <p>
095       <label>Leitung</label>
096      <input type="text" name="Kürzel" class="feld" value="<?php echo $row['Kürzel']; ?>">
097     </p>
098     <p style="text-align: center">
099       <?php if (getSession('Benutzername') == 'admin')
100         echo '<button name="Action" value="Speichern">Speichern</button>'; ?>
101     </p>
102   </form>
103 </div>
104 </body>
105 </html>
```

In den Zeilen 5 bis 27 werden Projektdaten aktualisiert, sofern der Benutzer *admin* auf *Speichern* geklickt hat. In Zeile 29 wird geprüft, ob das Skript mit *Speichern* aufgerufen wurde, dann wird die Projektnummer mit getPost ermittelt, oder mit nicht, dann wurde das Skript aus der Projektliste aufgerufen und die Projektnummer wird mit getGet() ermittelt. Zwischen den Zeilen 51 und 56 sehen Sie ein Beispiel, wie man das Menü in Abhängigkeit von einer Anmeldung gestalten kann. Die Zeilen 99 und 100 sorgen dafür, dass nur der *admin* Projektdaten ändern kann.

4.9 Aufgaben

Schüler

Wenn sich ein Schüler anmeldet

1. soll er analog zum Lehrer seine Daten bearbeiten können,
2. erhält er in der Projektliste alle für ihn zugelassenen Projekte angezeigt und kann eine 1., 2. und 3. Wahl durchführen
3. kann er sich für sein Projekt die Teilnehmer anzeigen lassen.

Lehrer

4. Bei der Anzeige eines Projekts soll auch eine Liste der zugeteilten Schüler angezeigt werden.
5. Das Menü *Schüler* zeigt eine Liste der Klassen. Wählt man eine Klasse aus, wird die Klassenliste angezeigt. Wählt man darin einen Schüler aus, kann man dessen Daten bearbeiten.
6. Eine Lehrkraft kann sich bequem ihr eigenes Projekt anzeigen lassen.

Admin

7. Das Menü für den Benutzer *admin* soll den Eintrag *Passwörter* haben. Das Skript passwort.php erzeugt Passwörter für die Lehrer und Schüler.
8. Das Menü für den Benutzer *admin* soll den Eintrag *Lehrer* haben. Das Skript lehrer.php zeigt eine Liste der Lehrkräfte an, aus der man eine Lehrkraft auswählen und bearbeiten kann.
9. Projekte sollen vom *admin* gelöscht und neu angelegt werden können.
10. Bei der Anzeige eines Projekts soll das Eingabefeld Leitung eine Auswahl der vorhandenen Lehrkräfte anbieten, so dass kein falscher Fremdschlüssel eingegeben werden kann.
11. Es soll eine Übersicht über die Projekte mit 1., 2. und 3. Wahl angezeigt werden.
12. Der *admin* kann sich für jede Klasse eine Liste mit den Zugangsdaten ausdrucken lassen.
13. In zuteilung.php liegt aus einem anderen Projekt ein Zuteilungsalgorithmus vor. Passen Sie diesen auf unser Webdatenbankprojekt an.

5. Quellen und Verweise

Luo-Wiki	http://www.luo-darmstadt.de/wiki2/doku.php
SQL-Tutorials	http://www.luo-darmstadt.de/wiki2/doku.php?id=db:sql
MySQL-Documentation	http://dev.mysql.com/doc/refman/5.7/en/multiple-tables.html
PHP-Handbuch	https://secure.php.net/manual/de/index.php
Bildungsserver	http://arbeitsplattform.bildung.hessen.de/fach/informatik/
SBO-Projekt	http://luo-darmstadt.de/users/~sbo/login.php Anmeldung: schueler/schueler