Forum

Assignment zum Modul:

Datenbanksysteme (DBA20)

27. Dezember 2018

Vladimir Zhelezarov

.....

Studiengang: Digital Engineering und angewandte Informatik - Bachelor of Engineering

AKAD University

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Ziel dieser Arbeit ist ein Internet-Forum aufzubauen, basierend auf MySQL und PHP. Dazu stellen wir folgende Anforderungen:

- Das Forum arbeitet ausschließlich mit registrierten Benutzern. Nicht registrierte Benutzer können sich allerdings als „Gast“ das Forum anschauen;
- Voreingestellt ist der System-Benutzer mit unbegrenztem Zugriff „admin“;
- Neue Benutzerkonten können nur durch Administratoren erstellt werden;
- Registrierte Benutzer können eigene Themen eröffnen und eigene oder andere vorhandene Themen kommentieren, solange das jeweilige Thema nicht gesperrt ist;
- Der Administrator hat ein Web-Interface um die Benutzer zu steuern. Das beinhaltet: Passwörter zurücksetzen, Benutzer sperren, Administrator-Rechte erteilen, neue Benutzerkonten erstellen oder Benutzer komplett löschen;
- Der Administrator hat erweiterten Zugriff auf Themen und Antworten. Dazu gehört: Themen sperren, wodurch keine weiteren Einträge möglich sind, Themen löschen, alle Nachrichten bearbeiten oder löschen;
- Jeder normale Benutzer kann die eigene Nachrichten bearbeiten.

1.2 Benutzte Software, Bibliotheken und Werkzeuge

- Die lokale Testumgebung ist XAMPP[1] - ein Software-Packet, was unter anderem aus einem Web-Server (Apache) und einem Datenbank-Server besteht. Der enthaltene Datenbank-Server ist MariaDB, der für unsere Zwecke gleich zu setzen mit MySQL ist;
- Aufzustellen ist das Forum in reale Umgebung im Internet. Für dieses Projekt haben wir GearHost[2] ausgewählt - ein Hosting-Anbieter aus der USA;
- Das ER-Modell wird im draw.io[3] gezeichnet; das relationale Modell - mit MySQL-Workbench[4];

[1] verfügbar unter: https://www.apachefriends.org/de/index.html
[2] verfügbar unter: https://www.gearhost.com/
[3] verfügbar unter: https://www.draw.io/
[4] verfügbar unter: https://www.mysql.com/products/workbench/

- Als Quellcode-Editor wird Atom[1] benutzt. Alle Grafiken erstellen wir, wenn nichts anderes erwähnt, mit Gimp[2];
- Um das visuelle Aussehen des Forums zu verbessern, benutzen wir CSS von Bootstrap[3];
- Maßgeblich für die Arbeit mit PHP und MySQL ist die offizielle Dokumentation verfügbar online[4] [5];
- Alle Meldungen und Benutzeroberflächen werden auf Deutsch angezeigt; interne Parameter, Variablen, Funktionen etc. - auf Englisch.

2 Grundlagen

2.1 HTTP GET/POST

Die Informationen, die vom Benutzer eingelesen werden, müssen irgendwie zum Server gelangen, um bearbeitet zu werden. Zwei verbreitete Methoden, womit das gemacht wird und mit denen auch PHP arbeitet, sind *GET* und *POST* - Teil der HTTP Spezifikation[6]. Bei beiden Methoden werden Tupel wie *Parameter=Wert* zum Server übertragen, wobei bei *GET* die Tupel als Teil der Webadresse (URL) übertragen werden, und bei *POST* - im Körper vom HTTP-request.

2.2 PHP Session

Der Server muss von Benutzer zu Benutzer unterscheiden und dementsprechend korrekt die Daten anpassen. Das passiert mittels die sogenannte *Session* - jedem Benutzer wird eine eigene eindeutige ID zugeordnet, die auf dem Server und beim Benutzer (Cookie) gespeichert wird. Dadurch wird er erkannt und muss sich nicht bei jedem Aufruf der Seite neu authentifizieren. Die benutzereigenen Daten können in dem Array *$_SESSION* gespeichert werden[7].

2.3 Benutzereingaben und mysqli

Als gute Praxis gilt es immer den Eingaben vom Benutzer nicht zu vertrauen, seien die Fehler aus Versehen oder gezielte Angriffe, damit es nicht zu Fehlfunktion oder Datenverlust

[1] verfügbar unter: https://atom.io/
[2] verfügbar unter: https://www.gimp.org/downloads/
[3] verfügbar unter: https://getbootstrap.com/
[4] für PHP siehe https://secure.php.net/docs.php
[5] für MySQL siehe https://dev.mysql.com/doc/
[6] https://www.w3.org/Protocols/ (Zugriff am 22.12.2018)
[7] https://secure.php.net/manual/en/intro.session.php, (Zugriff am 22.12.2018)

kommen kann. Normalerweise wollen wir SQL-Befehle durch PHP vorbereiten und vom Datenbank-Server ausführen lassen, wobei die Parameter oftmals von der Benutzereingabe kommen. Wenn wir diese nicht kontrollieren, kann der Benutzer sogar seine eigenen SQL-Befehle inserieren. Um dieses Problem vorzubeugen und allgemein für die Kommunikation mit der Datenbank wird ***mysqli*** : *die MySQL Improved Extension*[1] angeboten, eine PHP-Klasse, die sich von der Vorbereitung über Ausführung bis zum Ergebnis eines SQL-Befehls kümmern kann. Bei dem in dieser Arbeit gebautem Internet-Forum wird ausschließlich *mysqli* benutzt.

3 Modellieren

3.1 ER-Modell

Nach der vorhandenen Problemstellung lässt sich das ER-Modell wie folgt erzeugen:

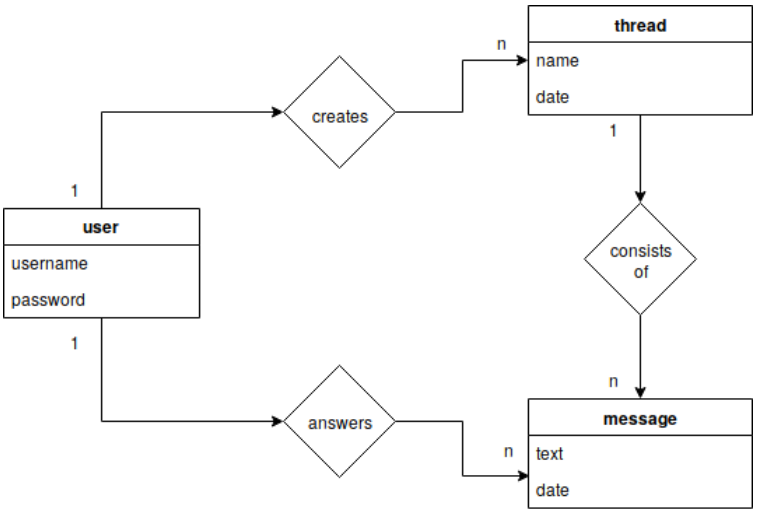

Abbildung 1: ER-Diagramm

3.2 Relationales Modell

Bei der Umsetzung von ER auf ein relationales Modell müssen wir noch ein Paar Besonderheiten beachten und entsprechend das Modell ergänzen:

- Beim **user** und beim **thread** lässt sich erkennen, dass der **user.username** und der **thread.name** nur einmalig in der jeweiligen Tabelle vorkommen dürfen. Also können sie als Primärschlüssel dienen. Allerdings bringt das den Nachteil mit sich, dass der Benutzer seinen Namen nicht ändern kann, gleich so gilt es für die Themennamen.

[1] https://secure.php.net/manual/en/book.mysqli.php, (Zugriff am 22.12.2018)

Obwohl das in unserem Fall nicht unbedingt notwendig ist, können wir mit allen drei Tabellen gleich umgehen, und überall als Primärschlüssel **id** bevorzugen;

- Der Benutzer kann vom Administrator gesperrt werden, wie auch ein Thema. Dazu führen wir die Felder **user.locked** und **thread.locked** ein;
- Ob ein Benutzer ein Administrator ist, zeigt das Parameter **user.is_admin** an;
- Bei allen drei Tabellen ist zusätzlich ein Zeitstempel notwendig, um später das Sortieren zu erleichtern - seit wann der Benutzer existiert, wann das Thema oder die Nachricht erstellt wurde;
- Um mehr Kontrolle über das Thema zu haben, wäre es gut zu wissen aus wieviel Nachrichten es besteht. Dafür das Feld **thread.msg_count**;
- Die Relationen zwischen den Tabellen erzeugen wir mit der Einführung der entsprechenden Fremd-Schlüssel;
- Die Auswahl von Datentypen der verschiedenen Felder ergibt sich selbstverständlich aus dem Zweck des Feldes. Bei **user.password** ist es allerdings nie eine gute Idee das Passwort im Klartext zu speichern. Dafür speichern wir nicht das Passwort, sondern sein Hash. Nach der PHP-Dokumentation eignet sich dabei eine Feld-Länge von 255 Zeichen[1].

[1] https://secure.php.net/manual/en/function.password-hash.php, (Zugriff am 22.12.2018)

Somit sieht das relationale Modell wie folgt aus:

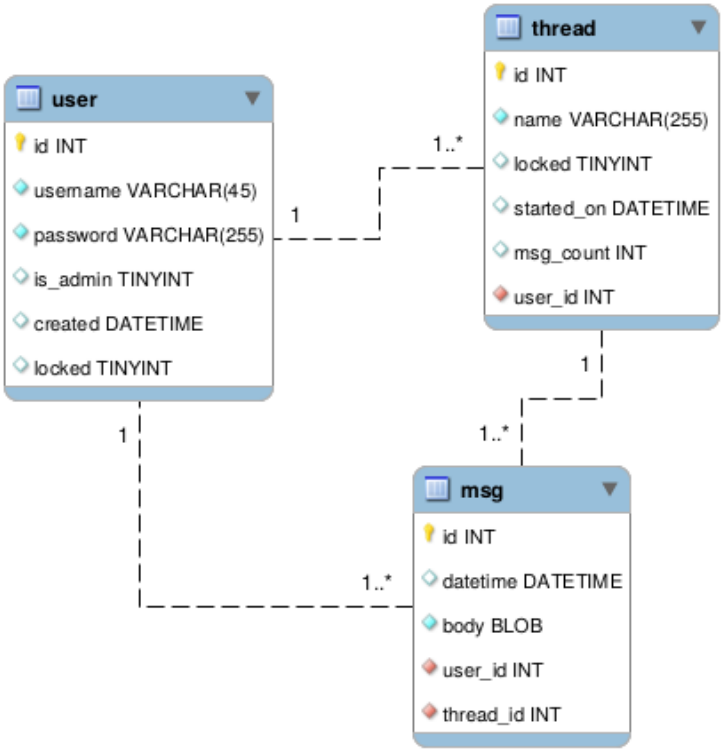

Abbildung 2: relationales Modell

3.2.1 Normalisierung

Um später Probleme bei der Arbeit mit den Daten zu vermeiden - die sogenannten Anomalien[1], die beim Löschen, Zufügen usw. entstehen, richten wir uns nach der „obersten Regel des Datenbankentwurfs“[2], wonach jede Information in der Datenbank nur ein Mal vorkommen darf. Wie es sich leicht erkennen lässt ist dies in dem oben angezeigten Modell schon der Fall.

Weil es keine mehrfachen Einträge gibt (1NF)[3], alle Nichtschlüsselattribute vom gesamten Primärschlüssel abhängig sind (2NF)[4] und nur davon (3NF)[5], befindet sich die Relation schon in der dritten Normalform.

[1] Staud, J. (o.J.), S.53
[2] Staud, J. (2015), Kap. 7.1
[3] Staud, J. (o.J.), S.52
[4] Staud, J. (o.J.), S.58
[5] Staud, J. (o.J.), S.60

4 Datenbank erstellen

Nachdem wir nichts mehr am Modell optimieren wollen, können wir es direkt in SQL-Befehle umsetzen.

1. Datenbank mit Voreinstellung für Unicode erstellen:

```
CREATE DATABASE IF NOT EXISTS `forum`
    DEFAULT CHARACTER SET utf8;
```

2. Wechseln zur (neu erstellten) Datenbank:

```
USE forum;
```

3. Tabelle "Benutzer" nach dem Modell erstellen:

```
CREATE TABLE IF NOT EXISTS `forum`.`user` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `username` VARCHAR(45) NOT NULL UNIQUE,
    `password` VARCHAR(255) NOT NULL,
    `is_admin` TINYINT DEFAULT 0,
    `created` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `locked` TINYINT DEFAULT 0,     PRIMARY KEY (`id`)
);
```

4. System-Benutzer *admin* - der Haupt-Administrator. Voreinstellung ist *admin / admin*

```
INSERT INTO `forum`.`user` (`id`, `username`, `password`, `is_admin`)
    VALUES (1, 'admin',
        '$2y$10$z9EN942BCKzoPFwKQCmAdOkFedHEQFkjCYhU9DgtRWuL76zblWG82', 1
);
```

5. Noch ein System-Benutzer. Dieser wird benötigt für die Situation, wenn ein Benutzer gelöscht wird, der schon Themen erstellt hat, die andere kommentiert haben. Voreinstellung ist *[entfernt]/zufälliges Passwort*, diese werden allerdings nicht benötigt / direkt adressiert, wie wir später beim PHP-Code sehen werden.

```
INSERT INTO `forum`.`user` (`id`, `username`, `password`, `locked`)
    VALUES (2, '[entfernt]',
        '$2y$10$Hm8d/Mj6wQwBTAbbwBHihOFqH3sTfqGwbnoaoRw635fkA01Rpbupu', 1
);
```

6. Tabelle "Themen" nach dem Modell:

```
CREATE TABLE IF NOT EXISTS `forum`.`thread` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL UNIQUE,
```

```
    `locked` TINYINT DEFAULT 0,
    `started_on` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `msg_count` INT DEFAULT 1,
    `user_id` INT NOT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`user_id`) REFERENCES `forum`.`user` (`id`)
);
```

7. Tabelle "Nachrichten" nach dem Modell:

```
CREATE TABLE IF NOT EXISTS `forum`.`msg` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `datetime` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `body` BLOB NOT NULL,
    `thread_id` INT NOT NULL,
    `user_id` INT NOT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`thread_id`) REFERENCES `forum`.`thread` (`id`),
    FOREIGN KEY (`user_id`) REFERENCES `forum`.`user` (`id`)
);
```

Ob in der lokalen Umgebung oder auf dem Hosting-Server, nach der Ausführung des SQL Skriptes steht uns schon die Datenbank zur Verfügung. Die Schnittstelle zwischen ihr und dem Benutzer stellen wir durch PHP-Skripten her.

5 PHP, SQL und Web-Interface

5.1 Server-Einstellungen

Bevor wir mit PHP etwas anfangen wollen, müssen wir dafür sorgen, dass PHP unsere Datenbank findet und sich in ihr einloggen kann. Diese und andere vordefinierte Einstellungen lassen sich in der **config.php** eintragen. Diese Datei werden wir bei allen anderen Skripten einschließen (über **helpers.php**). Außer der Login-Daten sind dabei auch die Adresse vom Server, Hilfspfad, Adresse von der *BOOTSTRAP* - CSS Datei und die Zeitverschiebung. Der letzte Parameter wird benötigt, wenn der Server nicht in Deutschland ist, um die angezeigte Zeit und das Datum bei den *SELECT* Befehlen korrekt anzuzeigen. Der Vorteil dieser Datei ist, dass hier der einzige Platz ist, wo man etwas ändern muss, wenn es z.B. zum Provider-Wechsel kommt.

5.2 Hilfsfunktionen

Die Datei **helpers.php** kümmert sich um das Einschließen von der *config*-Datei, den Aufbau der Verbindung zur Datenbank und stellt ein Paar oftmals benötigte Funktionen zur Verfügung.

- ***exec_sql:***

Ein typisches Szenario einer möglichen SQL-Abfrage sieht wie folgt aus:

1. Es werden Parameter vom Benutzer abgefragt und in Variablen gespeichert;
2. Ein SQL-Befehl wird aufgebaut, wobei die eingefügten Eingaben kontrolliert werden;
3. Der Skript wird ausgeführt. Vom Ergebnis interessiert uns nur ob es geklappt hat, oder nicht - wie etwa bei *INSERT*, *UPDATE*, *DELETE* usw. Daten von der Datenbank kommen nicht in Frage.

Weil das relativ oft vorkommt, bietet sich an, das Vorgehen als eine Funktion bereitzustellen.

- ***boilerplate_head***:

Weil alle *.html* Dateien ein *<head></head>* Teil haben werden, der überall gleich bis auf dem Titel ist, kann man das auch als Funktion bauen, wo der einzige Parameter der Titel der Webseite ist.

- ***console_log*** ist nützlich beim Debugging;
- ***err_msg*** ist die eigene Fehlermeldung, die bei Problemen von überall aufgerufen wird;
- **check_user** wird von überall aufgerufen um die aktuelle Benutzerdaten abzufragen, weil es sein kann, dass sich der Status eines Benutzer geändert hat (gesperrt, gelöscht oder Admin-Rechte geändert), während er noch in der selben Session angemeldet ist. In so einer Situation würde die Session nicht mehr aktuelle Daten beinhalten, deswegen wird in *$_SESSION* nur die ID gespeichert und dann immer mit *check_user* alles anderes nachgeholt.

5.3 Homepage, Themenansicht, Nachrichten

Die Homepage-Datei **index.php**, wie auch alle anderen PHP-Dateien von unserem Forum, fängt mit Initialisierung der Session an. Daraus wird der benutzerbezogenen Parameter „*id*“ abgelesen, der beim Einloggen eingestellt wurde (siehe dazu *login.php*). Außer etwas Formatieren mittels html-Tabelle und die übliche Header und Footer, ist hier der SQL-Befehl zum Ablesen der Daten, den wir etwas mehr beschreiben:

Um Übersicht in unserer Homepage zu bekommen, wollen wir die Tabelle mit Themenname, Themen-Starter und letzter Eintrag einfüllen. Dazu brauchen wir Daten von allen drei SQL-Tabellen *user*, *thread* und *msg*. Hierfür eignet sich der SQL-Befehl *JOIN*, der Tabellen nach vorgegebenen Kriterien (.. *ON* ..) kombiniert[1]. Zusätzlich wollen wir die Tabelle *user* praktisch zwei Mal abfragen - einmal nach dem Starter, und einmal nach dem Benutzer, der die letzte Nachricht erstellt hat. Dafür schließen wir *user* jeweils mit verschiedenen Namen ein - *user_a* und *user_b*. *CONVERT_TZ* dient das Datum und die Zeit richtig umzurechnen[2].

Zuerst SELECT-ieren wir alle Felder, wonach wir uns interessieren, inklusive Pfad - sprich die Tabelle wo sie sich befinden:

```
SELECT thread.id, thread.name, thread.locked, thread.msg_count,
    CONVERT_TZ(thread.started_on,'+00:00','".TIMEOFFSET."') AS 'started_on',
    CONVERT_TZ(msg.datetime,'+00:00','".TIMEOFFSET."') AS 'last_entry',
    user_a.username AS 'starter', user_b.username AS 'last_user',
    user_a.locked AS 'starter_lock', user_b.locked AS 'last_user_lock'
```

Wir müssen jetzt mit der einen Tabelle anfangen und alle anderen benötigten Tabellen mit *JOIN* einschließen:

```
FROM thread
```

Bei den *JOIN*s passen wir auf Kriterien und Benennung auf:

```
INNER JOIN user AS user_a ON thread.user_id=user_a.id
INNER JOIN msg ON thread.id=msg.thread_id
INNER JOIN user AS user_b ON msg.user_id=user_b.id
```

Soweit sind alle Themen ausgewählt, mit dem entsprechenden Themenstarter, alle anderen abgefragten Felder und *alle* Nachrichten. Wir wollen aber nur die Zeit und den Autor der letzten Nachricht des jeweiligen Themas anzeigen. Dafür kommt die *WHERE* Klausel - Abfrage nach Zeit, wobei diese Zeit die letzte Zeit in dem angesprochen Thema ist (Unterklausel):

```
WHERE msg.datetime = (SELECT MAX(msg.datetime) FROM msg
    WHERE msg.thread_id=thread.id)
```

Üblich bei Online-Foren ist oftmals nach den zuletzt kommentierten Themen zu sortieren:

```
ORDER BY msg.datetime DESC
```

Die SQL-Abfrage beinhaltet keine vom Benutzer eingegebenen Daten, weshalb wir nicht viel vorbereiten müssen, sondern gleich mit *mysqli->query*[3] abfragen und die Ergebnisse,

[1] https://dev.mysql.com/doc/refman/5.7/en/join.html, (Zugriff am 22.12.2018)

[2] https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_convert-tz, (Zugriff am 22.12.2018)

[3] https://secure.php.net/manual/en/mysqli.query.php, (Zugriff am 22.12.2018)

wenn vorhanden, vom *$result* Objekt in einer *while* Schleife ablesen. Dieses Vorgehen wird auch in den anderen Skripten bei ähnlichen Bedingungen benutzt.

Der Administrator sieht etwas mehr auf der Homepage, nämlich die Links zum Sperren oder Löschen von Themen.

Ähnlich so sieht es aus bei dem Themenansicht - dem Administrator stehen extra die Links zum Löschen oder Bearbeiten einzelne Nachrichten zur Verfügung. Für den normalen Benutzer ist es nur möglich die eigene Nachrichten zu bearbeiten.

Beim Erstellen einer neuen Antwort oder eines neuen Themas wird genau so zuerst nach Benutzerrechte überprüft, auch zusätzlich ob der Benutzer oder das Thema gesperrt sind und ein Paar übliche Sicherheitschecks - wie z.B. bei neue Antworten ob deren Thema überhaupt existiert.

5.4 Ein- und Ausloggen

Für das Ein- und Ausloggen sorgen jeweils **login.php** und **logout.php**

Beim Einloggen, nachdem logischerweise überprüft wird ob der Benutzer existiert und ob sein Passwort richtig ist, werden die Daten von der $_SESSION eingestellt, wonach wir unserem Benutzer in allen anderen Skripten erkennen können.

Beim Ausloggen werden diese Daten gelöscht und die Session zerstört[1].

5.5 Benutzerverwaltung

Der **users.php** Skript beschäftigt sich mit der Benutzerverwaltung. Diese Seite wird nur Administratoren angeboten. Außer der Übersicht der vorhandenen Benutzerkonten und Formular zur Erstellung von neuen, werden hier auch die Benutzer einzeln verwaltet.

Nach der Problemstellung und den üblichen Praktiken für Forum-Administratoren setzen wir folgende Verfahren ein:

- Ein Benutzer kann zu einem Administrator gemacht werden und umgekehrt. Dafür wird das Feld *is_admin* gesetzt;
- Ein Benutzer kann gesperrt oder wieder entsperrt werden. Dafür wird das Feld *locked* gesetzt. Nach diesem Feld wird in den anderen Skripten abgefragt und danach entsprechend wird der Benutzer eingegrenzt oder nicht;
- Benutzerpasswörter können geändert werden. Das neue Passwort wird gehascht und so in der Datenbank gespeichert;

[1] https://secure.php.net/manual/en/function.session-destroy.php, (Zugriff am 22.12.2018)

- Ein Benutzerkonto kann komplett gelöscht werden. Hier kommt das System-Konto *[entfernt]* ins Spiel - alle vom gelöschten Benutzer erstellten Themen oder Nachrichten werden aktualisiert, auf *[entfernt]* zu zeigen.

5.6 Interne PHP Skripten

Die Skripten **manage*.php**, sind nicht dafür gedacht, direkt vom Benutzer aufgerufen zu werden. Sie bekommen die benötigten Parameter mit *GET*, prüfen diese, und führen damit SQL-Befehle durch.

5.7 Redirects

Es kommt immer wieder zu der Situation, in der der Benutzer zu einer anderen Seite geleitet werden muss, sei es wenn bei der aktuellen alles gut gelaufen ist, oder wenn schwere Fehler vorliegen. Um eventuelle Fehler bei inkorrekter Interpretation von relativen Pfaden zu vermeiden, setzen wir auf volle Pfade, wie „*http://webseite.com/subseite.php/?param=value*“. Die Server-Adresse kann in der *config*-Datei eingestellt werden und daraus wird bei der Umleitung die volle Adresse aufgebaut. Dazu ein Beispiel aus **managemsg.php** :

```
header("location:␣" .HOMEDIR. "viewthread.php/?id=$thread_id");
exit;
```

exit stellt sicher, dass selbst wenn der Browser unsere Redirect-Anweisung ignoriert, keine weiteren Skripten oder Inhalten geladen werden.

6 Zusammenfassung

Die Webseite „Mein Forum“ ist aufgebaut, funktionsfähig[1] und so gut wie möglich nach Bugs getestet. Wegen der unsicheren Verbindung ist sie nicht richtig als reales Forum geeignet, als Test-Zwecke dieser Arbeit aber schon.

Es wurde eine Datenbank entwickelt, die den gegebenen Kriterien entspricht und die Arbeitsszenarien unterstützen kann. Durch PHP-Skripten werden die Benutzereingaben bearbeitet und entsprechende Abfragen an der Datenbank gestellt. Über Session wird der Benutzer auf den verschiedenen Seiten verfolgt und seine Freigaben und Optionen kontrolliert. Einfache Webseiten werden auf dem Server erstellt und dem Benutzer angeboten. Es wird zwischen Administratoren und normalen Benutzer unterschieden, wobei ein Administrator mehrere Rechte und Optionen hat, dazu auch die separate Seite zur Benutzerverwaltung.

Um den Umfang dieser Arbeit in den vorgegebenen Grenzen zu halten, wurde auf verschiedenen nützlichen Funktionen verzichtet, womit ein Forum verfügen kann. Das optische Aussehen basiert praktisch nur auf *BOOTSTRAP* und Interaktivität bezieht sich nur auf server-side-Skripten. Sicherheit wurde nur sehr oberflächlich behandelt um die einfachsten Gefahren zu vermeiden, allerdings werden z.B. die *POST* Eingaben so gut wie gar nicht kontrolliert. Beschrieben wurden nur die Schlüsselkomponenten vom Quellcode. Formatierung, Aufbau der Befehle und sprachspezifische Themen wurden übersprungen.

[1] Erreichbar unter http://forum-dba20.gearhostpreview.com und Test-Konto „*stefan*“ / „*st9denT*“

Literaturverzeichnis

The PHP Group (o.J.)
PHP Manual,
https://secure.php.net/docs.php (Zugriff am 22.12.2018)

Oracle Corporation (o.J.)
MySQL Documentation,
https://dev.mysql.com/doc/ (Zugriff am 22.12.2018)

World Wide Web Consortium (o.J.)
HTTP - Hypertext Transfer Protocol,
https://www.w3.org/Protocols/ (Zugriff am 22.12.2018)

Staud, J. (2015)
Relationale Datenbanken. Grundlagen, Modellierung, Speicherung, Alternativen., o.O.

Staud, J. (o.J.)
Vom Datenmodell zur Speicherung in Dateien, AKAD-Studienbrief DAO101, o.O.