

Stefan Waidele
Ensisheimer Straße 2
79395 Neuenburg am Rhein
Stefan.Waidele@AKAD.de

AKAD Hochschule Stuttgart
Immatrikulationsnummer: 102 81 71

Modul DBA02 — Praktisches Arbeiten mit Datenbanken
Assignment

DATENBANKGESTÜTZTE PHP–ANWENDUNG FÜR EINE UMFRAGE-WEBSITE

Betreuer: Prof. Dr. Franz–Karl Schmatzer

1. November 2013



AKAD Hochschule Stuttgart

Inhaltsverzeichnis

| | |
|---|------------|
| Abbildungsverzeichnis | ii |
| 1 Einleitung | 1 |
| 1.1 Aufgabenstellung | 1 |
| 1.2 Gemeinschaftsarbeit | 2 |
| 1.3 Aufbau der Arbeit | 2 |
| 1.4 Abgrenzung | 3 |
| 2 Grundlagen | 4 |
| 2.1 Entwurfsmuster: Fassade | 4 |
| 2.2 Entwurfsmuster: Singleton | 4 |
| 2.3 PHP-Schnittstelle: Session-Cookies | 5 |
| 2.4 PHP-Schnittstelle: PDO | 5 |
| 2.5 HTML-Designframework: Bootstrap | 6 |
| 2.6 JavaScript-Frontendframework: jQuery | 6 |
| 3 Datenbank-Schema | 7 |
| 3.1 Konzeptuelles Datenbankschema: Entity-Relationship Diagramm | 7 |
| 3.2 Logisches Datenbankschema: Relationales Datenmodell | 8 |
| 3.3 Umsetzung in SQL | 9 |
| 3.3.1 Tabelle: user — Benutzerverwaltung | 9 |
| 3.3.2 Tabelle: frage — Fragestellungen | 10 |
| 3.3.3 Tabelle: antwort — Antwortmöglichkeiten | 10 |
| 3.3.4 Tabelle: geantwortet — Gegebene Antworten | 11 |
| 4 Klassenhierarchie | 11 |
| 4.1 Klasse: Datenbank — Low-Level Zugriff | 12 |
| 4.2 Klasse: SQL — High-Level Zugriff | 13 |
| 4.2.1 Speichern von neuen Fragen und Antwortmöglichkeiten | 14 |
| 4.3 Klasse: User — Benutzerverwaltung | 15 |
| 5 Umsetzung in HTML & PHP | 16 |
| 5.1 Seitenstruktur | 16 |
| 5.2 Layoutanpassungen mit CSS | 16 |
| 6 Bewertung | 18 |
| 6.1 Zusammenfassung | 18 |
| 6.2 kritische Würdigung | 18 |
| 6.3 Ausblick | 18 |
| 6.4 Erfolgsfaktoren | 18 |
| Literatur | iii |

Abbildungsverzeichnis

| | | |
|---|---|----|
| 1 | Entity Relationship Diagram | 7 |
| 2 | Relationales Modell | 8 |
| 3 | SQL: CREATE TABLE user | 9 |
| 4 | SQL: CREATE TABLE frage | 10 |
| 5 | SQL: CREATE TABLE antwort | 10 |
| 6 | SQL: CREATE TABLE geantwortet | 11 |
| 7 | UML-Klassendiagramm | 12 |
| 8 | HTML: Seitenstruktur und PHP-Includes | 17 |
| 9 | HTML-Seiten und Verlinkungen | 17 |

1 Einleitung

1.1 Aufgabenstellung

Im Rahmen des Moduls DBA02 war eine datenbankgestützte PHP–Anwendung für eine Website zu erstellen, welche die folgenden Kriterien erfüllt:

- **Frage stellen:** Besuchern der Website soll eine Frage gestellt werden, auf die sie mit einer oder mehreren vorgegebenen Möglichkeiten antworten können.
- **Auswertung:** Nach der Beantwortung der Frage soll dem Besucher eine Auswertung der bisher gegebenen Antworten (Angaben in Prozent) gezeigt werden.
- **Benutzerverwaltung:** Ein Administrator soll sich bei der Anwendung anmelden können. Hierzu soll ein Benutzername und Passwort abgefragt und geprüft werden.
- **Neue Fragen eingeben:** Dem Seitenadministrator soll es über ein Formular möglich sein, neue Fragen mit den zugehörigen Antwortmöglichkeiten einzugeben. Normalen Besucher der Website ist diese Möglichkeit zu verwehren.
- **Datenbank:** Alle benötigten Daten werden in einer MySQL–Datenbank gespeichert.
- **Echtzeitstatistiken:** Die Auswertung der gegebenen Antworten soll unmittelbar vor der Anzeige berechnet werden.
- **XAMP:** Die Anwendung soll mit der Kombination von Apache–Webserver, MySQL–Datenbank und PHP als Programmiersprache lauffähig sein. Das Betriebssystem kann frei gewählt werden.

Desweiteren sollte die Anwendung objektorientiert programmiert werden, Enturfsmuster verwenden und die HTML-Ausgabe per CSS formtiert werden.

1.2 Gemeinschaftsarbeit

Die Aufgabe war arbeitsteilig in Teamarbeit zu lösen. Das der Anwendung zu Grunde liegende Datenmodell wurde gemeinsam in einer Teambesprechung erarbeitet und festgelegt. Anschließend wurde ein Mockup¹ der HTML-Seiten und die benötigten SQL-Abfragen, gefolgt von einem prozedural programmierten Prototypen erstellt. Hierbei erstellte der Autor die für die Benutzerverwaltung und Frageneingabe notwendige Programmteile. Die Abfrage- und Auswertungsseiten wurden von Yvonne Frezel gefertigt.

Da die im Seminar DBA02 begonnene Umsetzung des Programmcodes in Klassen unterschiedliche Richtungen verfolgte, wurde die enge Teamarbeit anschließend nicht mehr weitergeführt. Aufgrund der gemeinsamen Datenbasis sind die vom Author und von Frenzel erstellten PHP-Dateien miteinander kombinierbar, auch wenn sie intern andere Klassen und Zugriffsmethoden nutzen.

Dieses Assignment geht hauptsächlich auf die vom Autor konzipierten Programmteile „Benutzerverwaltung“ und „Neue Fragen hinzufügen“ ein.

1.3 Aufbau der Arbeit

Zunächst wurden die für die Anwendungen relevanten Konzepte, Techniken, und Frameworks beschrieben. Anschließend erfolgte die Beschreibung der Implementierungsdetails und der vom Autor gewählten Lösungsmöglichkeiten

¹engl. für Attrappe. (to mock: nachahmen)

1.4 Abgrenzung

Da eine datenbankgestützte Web-Anwendung in der Regel einem großen Personenkreis² zur Verfügung steht sind hier unbedingt Sicherheitsaspekte zu beachten. Da eine ausführliche Betrachtung dieser Maßnahmen den Rahmen dieses Dokuments sprengen würde, werden nur entsprechende Hinweise auf weiterführende Informationen gegeben. Auch Performance-Überlegungen gehen nur in sehr beschränktem Maß in die Implementation ein.

²allen Internet- oder zumindest Intranetnutzern

2 Grundlagen

2.1 Entwurfsmuster: Fassade

Beim Fassaden-Entwurfsmuster gewährt eine Klasse einen einfachen Zugriff auf ein beliebig komplexes System weiterer Klassen. Den Nutzer der Fassadenklasse benötigt kein Wissen über die Funktionsweise der Klassenhierarchie hinter der Fassade, kann jedoch auf diese zugreifen, falls die bereitgestellte Funktionalität nicht ausreicht³.

In der hier erstellten Anwendung wird der Zugriff auf die Datenbank über die Fassaden-Klasse `SQL` realisiert. Diese erstellt das Low-Level Objekt der Klasse `Datenbank`, bereitet die notwendigen SQL-Abfragen vor und gibt die Resultate dann als String oder als Array von Strings zurück. Die Aufrufenden Routinen benötigen kein Wissen über die verwendete Datenbankschnittstelle oder über die Details der Abfragen. Sollten die in der Klassendefinition vorgesehenen Abfragen allerdings nicht ausreichen, kann auch direkt auf die Klasse `Datenbank` zugegriffen werden.

2.2 Entwurfsmuster: Singleton

Das Klasse nach dem Singleton-Entwurfsmuster stellt sicher, dass es in einem Programm von einer Klasse nur ein einziges Mal instanziiert wird. Allen Nutzern der Klasse wird dann eine Referenz auf ebendiese Instanz übergeben, der Zugriff erfolgt jeweils auf die gleichen Daten, die somit global zur Verfügung gestellt werden⁴.

Somit bildet das Singleton-Designpattern eine passende Grundlage für die Nutzerverwaltung, da immer nur ein Benutzer angemeldet sein kann⁵.

³vgl. [Balzert, 2005], Seite 367ff

⁴vgl. [Balzert, 2005], Seite 361ff

⁵Dies gilt jeweils pro Browser-Instanz. In einem weiteren Browserfenster mit eigenen Cookies kann sich ein weiterer Nutzer anmelden, jedoch auch wieder nur einer

Das Singleton-Entwurfsmuster kann aufgrund seiner Eigenschaften als objektorientierte Umsetzung von globalen Variablen mit all deren Vor- und Nachteilen gesehen werden und wird daher auch als „Anti-Pattern“ kritisiert⁶

2.3 PHP-Schnittstelle: Session-Cookies

Da HTTP ein zustandsloses Protokoll ist, wird ein Mechanismus benötigt, mit dem gespeichert werden kann, ob es sich beim Besucher der Website um einen angemeldeten Benutzer handelt oder nicht. Die von PHP bereitgestellten Session-Cookies können eine begrenzte Menge Daten (ca. 4kB), die im Browser gespeichert wird, von Seitenaufruf zu Seitenaufruf weitergeben⁷.

Die hier besprochene Anwendung verwendet diese Möglichkeit um den Anmeldestatus (`angemeldet==TRUE` bzw. `angemeldet==FALSE`) und den Benutzernamen zu speichern.

2.4 PHP-Schnittstelle: PDO

Der Datenbankzugriff wird in der beschriebenen Anwendung über die PDO-Klasse⁸ realisiert. Diese sorgt für den Verbindungsaufbau zur MySQL-Datenbank und stellt Schutzmechanismen gegen SQL-Injection Angriffe bereit. Für einen solchen Angriff müssen die Benutzereingaben ungeprüft in die SQL-Abfrage (z.B. durch einfache Variablen-Substitution) übernommen werden. Diese wird dadurch dann so verändert, dass sensible Daten ausgespäht oder die Daten verändert bzw. gelöscht werden können⁹. Die Überprüfung der Eingabewerte ist somit in jedem Falle empfehlenswert und aufgrund des öffentlichen Charakters der Webanwendung hier besonders wichtig.

⁶vgl. [Hauer, 2010]

⁷vgl. [Theis, 2013], S. 417ff

⁸PDO: PHP Data Object

⁹vgl. [Friedl, 2007]

2.5 HTML–Designframework: Bootstrap

Die hier vorgestellte Anwendung realisiert ihr Aussehen zum großen Teil mit dem CSS–Framework „Bootstrap“, welches moderne Designprinzipien wie variables, mehrspaltiges Layout oder gar Responsive Layout, welches sich an die unterschiedlichen Bildschirmgrößen vom Smartphone bis zum wandfüllenden HD–TV anpassen kann, anbietet¹⁰.

Die Anwendung selbst muss den entsprechenden Elementen lediglich die gewünschten CSS–Klassen zuweisen. Der Entwickler wird somit nicht mit den Design–Details belastet.

Trotz allem ist eine eigene Anpassung des Designs wie in der Datei `.../style.css` zu sehen ist möglich.

2.6 JavaScript–Frontendframework: jQuery

Bei der Realisierung der Eingabeseite für neue Fragen und Antwortmöglichkeiten wird die JavaScript Bibliothek „jQuery“ verwendet, um weitere Antwortmöglichkeiten einfügen oder diese auch wieder löschen zu können. jQuery bietet auch noch weitere Funktionen für interaktive Benutzeroberflächen, die allerdings in der hier beschriebenen Anwendung nicht genutzt werden.¹¹

¹⁰Lizenz: Apache Licence V2.0, Download unter <http://getbootstrap.com/>

¹¹Lizenz: MIT-Licence, Download unter <http://jquery.com/>

3 Datenbank–Schema

3.1 Konzeptuelles Datenbankschema: Entity–Relationship Diagramm

Hier werden die Gegenstände der realen Welt modelliert wie in Abbildung 3.1 gezeigt modelliert. Hierbei ist zu beachten, dass die Entität „Frage“ das zentrale Element des Datenmodells darstellt. Die weiteren Entitäten „Antwortmöglichkeit“ bzw. „gegebene Antwort“ sind existenzabhängige Entities. Ohne Antwortmöglichkeit kann ohne zugehörige Frage nicht existieren, eine gegebene Antwort macht nur Sinn, wenn es eine entsprechende Antwortmöglichkeit und Frage gibt.

Die Entität „Benutzer“ steht in keiner Beziehung zu den anderen Entitäten, sie wird auch nur für die Eingabe neuer Fragen und Antwortmöglichkeiten, bzw. zur Authentifizierung benötigt.

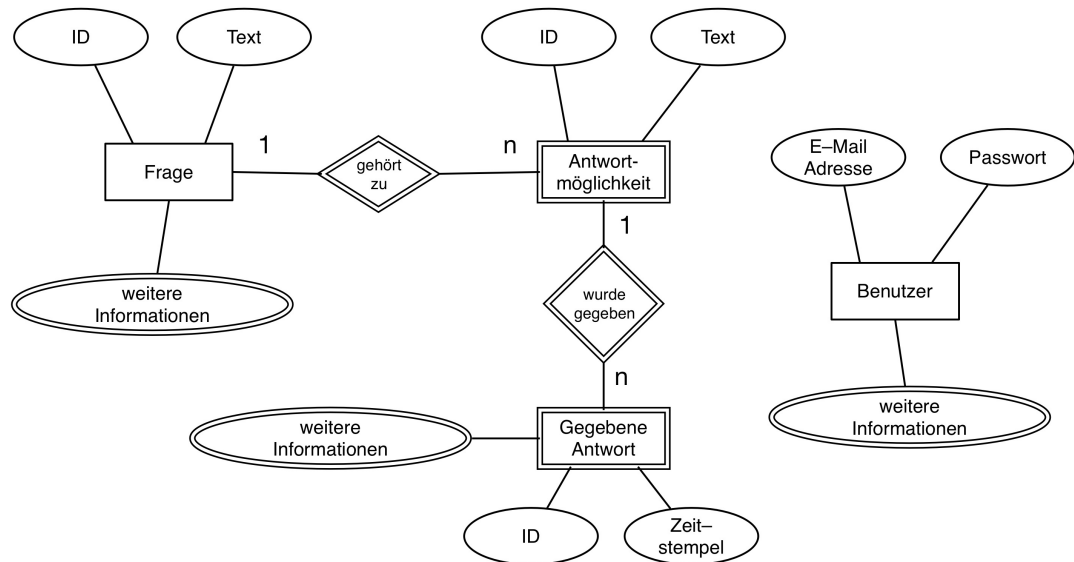


Abbildung 1: Entity Relationship Diagram

Die als zusammengesetzte Attribute dargestellten „weiteren Informationen“ sind Attribute der jeweiligen Entitäten, die für die gestellten Anforderungen nicht not-

wendig sind, jedoch im Produktivbetrieb großen Zusatznutzen bieten könnten. So wäre es möglich die gegebenen Antworten durch die Speicherung der IP-Adresse, Browser-Fingerprinting¹² oder andere Techniken genauer zu identifizieren und somit einem bestimmten Nutzer zuzuordnen. Hierbei sind dann die jeweiligen Datenschutzbestimmungen zu beachten.

Die Datensätze der Entity „Frage“ könnten Zeitangaben enthalten, die festlegen wann bzw. wie lange eine Frage auf der Website angezeigt wird. Die Antwortmöglichkeiten könnten durch entsprechende Angaben zur Sortierreihenfolge geordnet werden.

3.2 Logisches Datenbankschema: Relationales Datenmodell

Aufgrund des in Abbildung 3.1 auf Seite 7 dargestellten Modell ergibt sich die in Abbildung 3.2 dargestellten Relationen. Dieses Modell entspricht der 3. Normalform¹³.

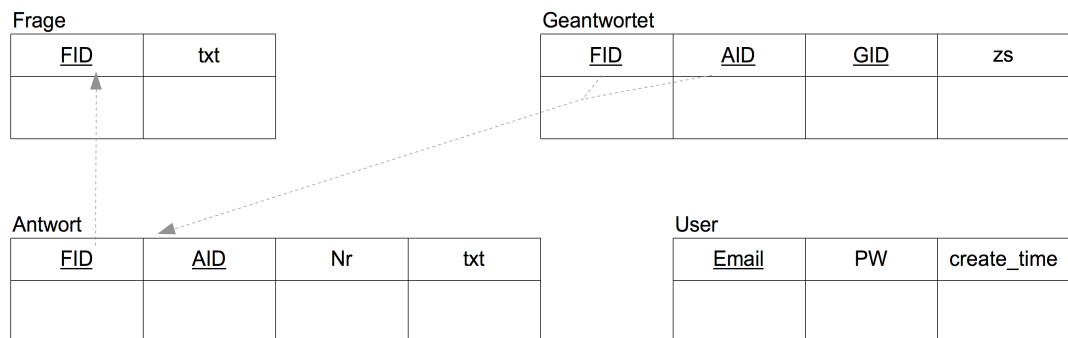


Abbildung 2: Relationales Modell

¹²Die Identifikation eines speziellen Nutzers durch Konfigurationsdetails des Webbrowsers, Bildschirmgröße, Betriebssystemversion, etc. Siehe auch <https://panopticlick.eff.org/>

¹³Kriterien laut [Staud, 2010], Kapitel 3.4

3.3 Umsetzung in SQL

3.3.1 Tabelle: user — Benutzerverwaltung

```
CREATE TABLE user (
    email VARCHAR(255) NOT NULL,
    pw CHAR(32) NOT NULL,
    create_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY ('email'));
```

Abbildung 3: SQL: CREATE TABLE user

Als Benutzername und auch als Primärschlüssel wird die E-Mail Adresse des Administrators genutzt. Da diese den Nutzer eindeutig identifiziert wird auf einen separaten Nutzernamen und auch auf eine generierte ID als Primärschlüssel verzichtet.

Das Passwort sollte nicht im Klartext in der Datenbank gespeichert werden. Ein `salted hash`¹⁴ schützt hier das Passwort vor dem Ausspähen durch den Administrator selbst¹⁵ oder durch Angreifer.

Die hier reservierten 32 Byte sind für den in der MySQL-Dokumentation¹⁷ empfohlenen MD5-Hash ausreichend. Da die entsprechende PHP-Dokumentation¹⁸ hier allerdings eine genau entgegengesetzte Empfehlung gibt, ist dieser Sicherheitsaspekt für ein Produktivsystem nochmals genauer zu prüfen.

¹⁴Also der Hashwert des Passworts, welches zuvor mit Applikationsspezifischen Zusatzdaten ergänzt wurde

¹⁵Spätestens seit dem Datenklau bei Vodafone¹⁶ eine dokumentierte Gefahr

¹⁷[?]

¹⁸[?]

3.3.2 Tabelle: frage — Fragestellungen

```
CREATE TABLE frage (  
    fid INT NOT NULL AUTO_INCREMENT,  
    txt VARCHAR(1024) NOT NULL,  
    CONSTRAINT pk_frage PRIMARY KEY ('fid'));
```

Abbildung 4: SQL: CREATE TABLE frage

In der Tabelle „Frage“ wird lediglich der Text der Frage sowie die eindeutige Frage-ID gespeichert. Letztere dient als Primärschlüssel.

3.3.3 Tabelle: antwort — Antwortmöglichkeiten

```
CREATE TABLE antwort (  
    fid INT NOT NULL,  
    aid INT NOT NULL AUTO_INCREMENT,  
    nr INT NULL,  
    txt VARCHAR(1024) NOT NULL,  
    CONSTRAINT pk_antwort PRIMARY KEY (fid, aid),  
    CONSTRAINT fk_antwort_frage_fid FOREIGN KEY (fid)  
    REFERENCES frage(fid)  
    ON UPDATE CASCADE  
    ON DELETE CASCADE );
```

Abbildung 5: SQL: CREATE TABLE antwort

Aufgrund der Modellierung als schwache Entity setzt sich der Primärschlüssel der Tabelle „Antwort“ aus der Frage-ID sowie der Antwort-ID zusammen. Zusätzlich wird der Antworttext sowie eine frei zu vergebende Nummer gespeichert, welche für die Sortierreihenfolge bei der Anzeige genutzt werden kann.

Die Integritätsbedingung wird so definiert, dass beim Löschen einer Frage auch die zugehörigen Antwortmöglichkeiten gelöscht werden.

3.3.4 Tabelle: geantwortet — Gegebene Antworten

```
CREATE TABLE geantwortet (  
    fid INT NOT NULL,  
    aid INT NOT NULL,  
    gid INT NOT NULL AUTO_INCREMENT,  
    zs TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    CONSTRAINT pk_geantwortet PRIMARY KEY (fid, aid, gid),  
    CONSTRAINT fk_geantwortet_antwort_aid FOREIGN KEY (fid, aid)  
    REFERENCES antwort(fid, aid)  
    ON UPDATE CASCADE  
    ON DELETE CASCADE );
```

Abbildung 6: SQL: CREATE TABLE geantwortet

Für die Speicherung der gegebenen Antworten in der Tabelle „geantwortet“ genügt der aus Frage-ID, Antwort-ID und Geantwortet-ID zusammengesetzte Primärschlüssel. Zusätzlich wird noch der jeweilige Zeitstempel für spätere Auswertungen erfasst.

Auch hier stellt die Integritätsbedingung sicher, dass beim Wegfall der entsprechenden Antwortmöglichkeit keine undefinierten gegebene Antworten zurückbleiben.

4 Klassenhierarchie

Der Zugriff auf die in der bislang beschriebenen Datenbank gespeicherten Daten erfolgt über die folgende Klassenhierarchie:

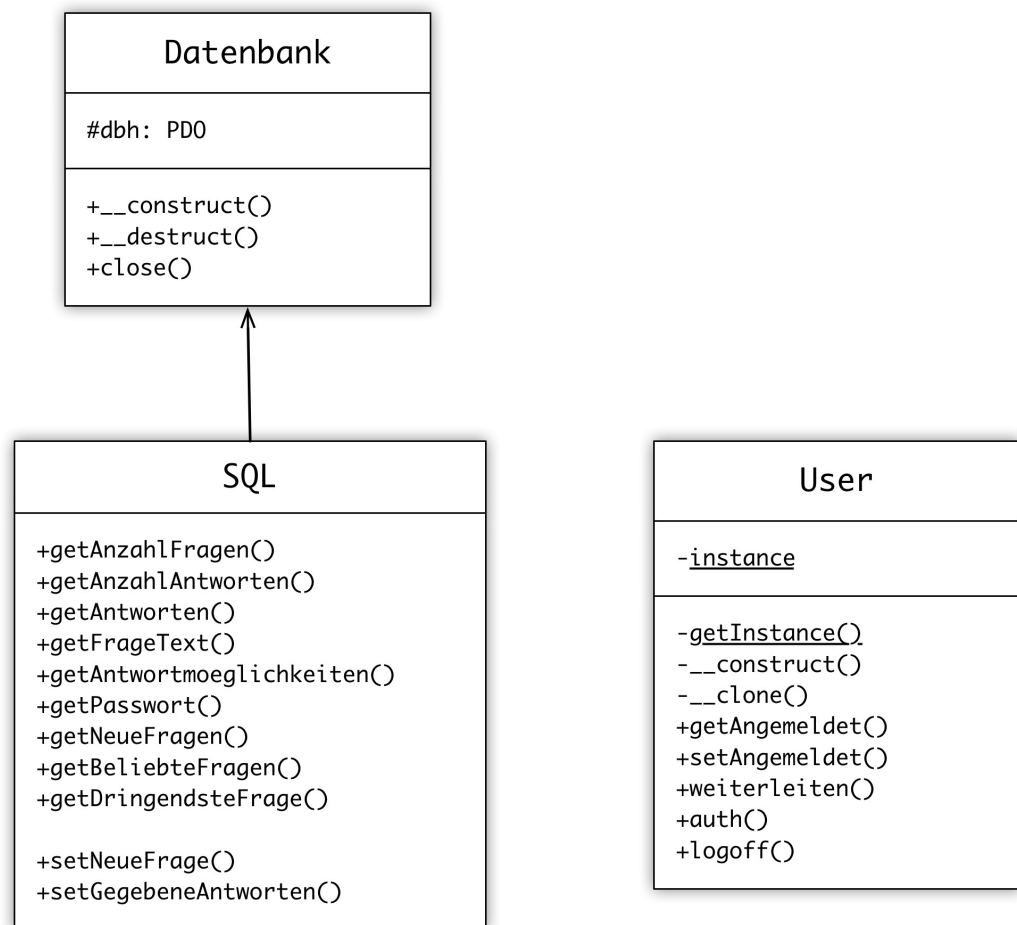


Abbildung 7: UML-Klassendiagramm

Die Implementation der Klassen ist im beigefügten Quelltext in den Dateien `.../class/Datenbank.php`, `.../class/SQL.php` und `.../class/User.php` zu betrachten. Auch die Klasse „User“ nutzt für den Datenbankzugriff die Klasse „SQL“. Dies geschieht allerdings nur innerhalb der Methode `auth()` über eine lokale Variable und ist daher im UML-Diagramm nicht zu erkennen.

4.1 Klasse: Datenbank — Low-Level Zugriff

Die Klasse „Datenbank“ stellt die unterste Ebene des Datenzugriffs der Anwendung dar. Aufgrund der von PHP zur Verfügung gestellten PDO-Klasse greift

jedoch auch diese Klasse nicht direkt auf die Datenbank zu. Das PDO-Objekt ist als Attribut in der Klasse vorhanden. Die für den Zugriff notwendigen Angaben werden im Konstruktor aus der Konfigurationsdatei `.../includes/dbconf.ini` gelesen und anschließend wird die Verbindung zur Datenbank hergestellt.

Die Datenbankverbindung bleibt so lange bestehen, wie die Instanz existiert. Ein Objekt mit geschlossener Datenbankverbindung würde die Notwendigkeit von zusätzlichen Fehlerprüfungen bzw. der erneuten Herstellung der Verbindung mit sich bringen. Um hierauf ausdrücklich hinzuweisen ist die Methode `close()` ohne Funktion definiert.

4.2 Klasse: SQL — High-Level Zugriff

In der Klasse „SQL“ werden die SQL-Abfragen gekapselt. Der aufrufende PHP-Code benötigt keinerlei Wissen über die zu Grunde liegende Datenbank. Es müssen lediglich die Get- und Set-Methoden eines Objektes vom Typ SQL aufgerufen werden. Innerhalb dieser Methoden werden dann die SQL-Abfragen mit Hilfe des geerbten PDO-Objekts vorbereitet und dann mit den geforderten Parametern aufgerufen. Die Ergebnisse werden dann als einzelner Wert oder als Objekt mit mehreren Werten bzw. Array zurückgegeben. Hierbei wird die jeweils geeignete Form gewählt.

Zum Speichern von neuen Fragen und Antwortmöglichkeiten bzw. von gegebenen Antworten werden diese an die entsprechenden Set-Methoden übergeben. Hierin werden die zum Abspeichern benötigten SQL-INSERTs in eine Transaktion verpackt. Somit ist auch bei gleichzeitigem Zugriff mehrerer Nutzer oder im Fehlerfall ein konsistenter Zustand des Datenbestandes gesichert.

Durch die Reduzierung der Aufrufe auf die passenden Get- und Set-Methoden wird in der SQL-Klasse das Fassade-Entwurfsmuster realisiert. Der Aufrufende Code benötigt keinerlei Informationen über die Datenstruktur, die Datenbank oder Transaktionen. Änderungen an diesen technischen Details wirken sich ledig-

lich auf die SQL-Klasse aus. So lange diese laut Spezifikation Werte entgegennimmt und zurückliefert, muss kein weiterer Programmcode geändert werden.

4.2.1 Speichern von neuen Fragen und Antwortmöglichkeiten

In der Methode `setNeueFrage()` werden neue Fragen und die zugehörigen Antwortmöglichkeiten in der Datenbank abgespeichert. Hierzu werden die entsprechenden Texte in den Parametern `$frageText` und `$antworten` übergeben werden. Bei letzterem handelt es sich um ein Array, da es pro Frage mehrere Antwortmöglichkeiten gibt.

Das Datenbankschema verlangt es, dass zu jeder Antwortmöglichkeit die ID der zugehörigen Frage gespeichert wird. Somit wird zunächst die Frage in die Datenbank eingefügt. Die automatisch generierte ID kann dann mit der PDO-Methode `lastInsertId()` ausgelesen werden. Anschließend werden die Antwortmöglichkeiten in einer `foreach`-Schleife in die Datenbank eingefügt.

Dieser Vorgang besteht somit aus vielen einzelnen Schritten. Fehlermöglichkeiten bestehen im PHP-Code selbst sowie beim Zugriff auf die Datenbank. An jeder Stelle sind Unterbrechungen durch weitere Instanzen der Anwendung möglich, was im schlimmsten Fall zu einer inkorrekten Frage-ID führen könnte. Es ist sicher zu stellen, dass entweder die gesamte Kombination von Frage mit allen zugehörigen Antwortmöglichkeiten gespeichert wird, oder das Speichern gar nicht statt findet und mit einer entsprechenden Fehlermeldung abbricht. Im Fehlerfall darf keine Frage ohne den kompletten Antwortsatz in der Datenbank sein.

Daher wird der gesamte Vorgang in einer SQL-Transaktion eingebettet und im Fehlerfall werden schon getätigte Änderungen mit einem Rollback zurückgenommen. Somit ist Konsistenz der Datenbank zu jedem Zeitpunkt sichergestellt.

4.3 Klasse: User — Benutzerverwaltung

Die Benutzerverwaltung nutzt das Singleton Entwurfsmuster. Dieses wird durch die folgenden Maßnahmen implementiert: Die private Deklaration des Konstruktors sowie der `clone()` Methode wird die direkte Instanziierung der Klasse unterbunden. Eine Referenz auf die Instanz wird in der ebenfalls privaten statischen Variable `$instance` gespeichert und über die öffentliche Funktion `getInstance()` bekannt gemacht.

Die Funktion `auth()` erwartet als Parameter den Benutzernamen und das Passwort sowie optional eine URL zu der im Falle der erfolgreichen Authentifizierung umgeleitet werden soll. Der Datenbankzugriff erfolgt über eine lokale Instanz der Klasse `SQL`. Stimmen die Anmeldedetails mit den in der Datenbank gespeicherten überein werden in der von PHP zur Verfügung gestellten `$_SESSION`-Variable das Feld „angemeldet“ mit `TRUE` und das Feld „benutzer“ mit dem Benutzernamen gefüllt. Stimmen die Anmeldedaten nicht mit den gespeicherten Werten überein, so erhält das Feld „angemeldet“ den Wert `FALSE`.

Die Funktion `logout()` meldet den momentan angemeldeten Nutzer ab, in dem die Session-Variable zerstört und damit auch die Felder „angemeldet“ und „benutzer“ löscht.

Die Funktion `setAngemeldet()` ist eine private Hilfsmethode, mit der das Feld „angemeldet“ der `$_SESSION`-Variable gesetzt werden kann. Da sie aus der Methode `getInstance()` heraus aufgerufen wird, ist sie wie diese auch statisch deklariert.

Die Funktion `getAngemeldet()` dient zur Anfrage des Anmeldestatus. Sie wird immer dann aus der Anwendung heraus aufgerufen, wenn der Seitenaufbau für angemeldete Benutzer anders sein soll, wie für nicht angemeldete. Am auffälligsten ist dies beim Aufruf der Seite zur Eingabe von neuen Fragen. Hier wird dem nicht angemeldeten Nutzer das Login-Formular angezeigt, während dem angemeldeten Nutzer die gewünschte Funktion direkt zur Verfügung steht.

Aber auch in Details können sich die Seiten unterscheiden: Die in der Seitenleiste angebotenen Links können schon entsprechend angepasst sein. Auf diese Weise werden dem Nutzer nur die Möglichkeiten gezeigt, die für ihn tatsächlich relevant sind.

5 Umsetzung in HTML & PHP

5.1 Seitenstruktur

Alle Seiten der Anwendung sind nach dem gleichen Muster aufgebaut. Sie werden von der Datei `index.php` erzeugt. Hier werden wie in Abbildung 8 auf Seite 17 die Seitenelemente aus den verschiedenen Include-Dateien für den Kopfbereich, die Navigationsleiste am rechten Rand und dem Fußbereich zusammengesetzt¹⁹. Der eigentliche Funktionsbereich²⁰ werden von den PHP-Dateien im Verzeichniss `.../pages/` übernommen.

Diese Seitenelemente und Programmteile werden in einer Struktur aus `DIV`-Elementen angeordnet, welche durch die CSS-Klassen des Bootstrap-Frameworks zu einem modernen, responsiven Layout ausgezeichnet werden. Der Anwendungsentwickler braucht sich somit nicht um die optimale Darstellung der HTML-Seiten auf Desktoprechnern, Tablets oder Smartphones zu kümmern. Jede Bildschirmgröße erhält durch Bootstrap automatisch ein speziell abgestimmtes Layout.

5.2 Layoutanpassungen mit CSS

¹⁹Darstellung vereinfacht

²⁰Frage anzeigen, Antworten speichern und auswerten, anmelden, abmelden, Eingabemaske für neue Fragen sowie das Abspeichern der neuen Fragen

```

<?php
include($_SERVER['DOCUMENT_ROOT'].'/includes/header.php');
?>
<div class="row">
  <div class="col-md-8">
    <?php
    include($_SERVER['DOCUMENT_ROOT'].'/pages/'. $show . '.php');
    ?>
  </div>
  <div class="col-md-4">
    <?php
    include($_SERVER['DOCUMENT_ROOT'].'/includes/sidebar.php');
    ?>
  </div>
</div>
<?php
include($_SERVER['DOCUMENT_ROOT'].'/includes/footer.php');
?>

```

Abbildung 8: HTML: Seitenstruktur und PHP-Includes

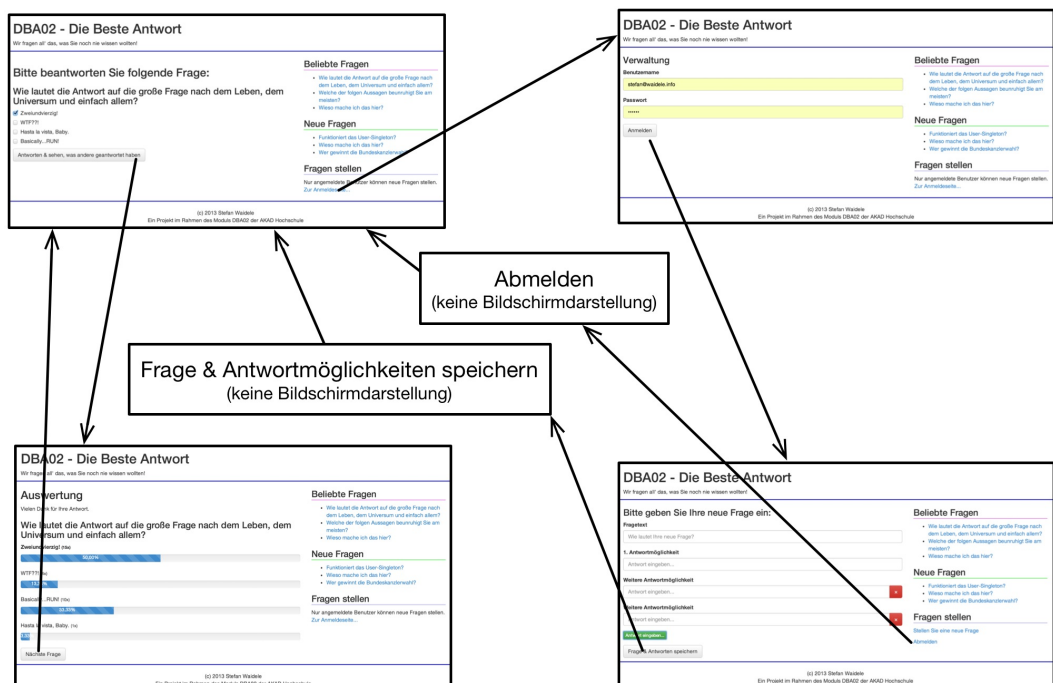


Abbildung 9: HTML-Seiten und Verlinkungen

6 Bewertung

6.1 Zusammenfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

6.2 kritische Würdigung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

6.3 Ausblick

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

6.4 Erfolgsfaktoren

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Literatur

- [Balzert, 2005] Balzert (2005). *Lehrbuch der Objektmodellierung*. Spektrum Akademischer Verlag, München.
- [Friedl, 2007] Friedl (2007). SQL Injection Attacks by Example, Abruf am 25.10.2013. <http://www.unixwiz.net/techtips/sql-injection.html>.
- [Fuest, 2013] Fuest (2013). Fall Vodafone zeigt die wahren Sicherheitslücken, Abruf am 25.10.2013. <http://www.welt.de/wirtschaft/webwelt/article119967954/Fall-Vodafone-zeigt-die-wahren-Sicherheitsluecken.html>.
- [Hauer, 2010] Hauer (2010). Das Singleton Design Pattern, Abruf am 11.10.2013. <http://www.philippbauer.de/study/se/design-pattern/singleton.php#nachteile>.
- [Staud, 2010] Staud (2010). *Grundlagen der Datenorganisation: Vom Datenmodell zur Speicherung in Dateien*. AKAD. Die Privathochschulen GmbH, Stuttgart.
- [Strickel, 1991] Strickel (1991). *Datenbankdesign: Methoden und Übungen*. Gabler, Wiesbaden.
- [Theis, 2013] Theis (2013). *Einstieg in PHP 5.5 und MySQL 5.6*. Gabler, Wiesbaden.

Eidesstattliche Erklärung

Ich versichere, dass ich das beiliegende Assignment selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.

tum, Ort)

(Unterschrift) (Da-

— Druckgröße kontrollieren! —

Breite = 100 mm

Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —