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

2. November 2013

AKAD Hochschule Stuttgart

Inhaltsverzeichnis

Abbildungsverzeichnis

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 Besuchern 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, Entwurfsmuster verwenden und die HTML–Ausgabe per CSS formatieren.

1.2 Gemeinschaftsarbeit

Die Aufgabe war arbeitsteilig in Teamarbeit zu lösen. Das der Anwendung zugrunde liegende Datenmodell wurde gemeinsam in einer Teambesprechung erarbeitet und festgelegt. Anschließend wurde ein Mockup[1] der HTML–Seiten und die benötigten SQL–Abfragen, gefolgt von einem prozedural programmierten Prototypen, erstellt. Hierbei verantwortete der Autor die für die Benutzerverwaltung und Frageneingabe notwendigen 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 Autor 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 werden im Grundlagenkapitel 2 die verwendeten Konzepte, Schnittstellen und Frameworks beschrieben. Anschließend wird das Datenbankschema im Kapitel 3 und die erstellte PHP–Klassenhierarchie in Kapitel 4 ausführlich dargestellt. Erläuterungen zur darauf aufbauenden Implementierung der Anwendung mit Hilfe von PHP und HTML schließen in Kapitel 5 den Hauptteil dieser Arbeit ab.

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

1.4 Abgrenzung

Da eine datenbankgestützte Web–Anwendung in der Regel einem großen Personenkreis[2] 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.

[2] allen 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. Der Nutzer der Fassadenklasse benötigt kein Wissen über die Funktionsweise der Klassenhierarchie hinter der Fassade, kann jedoch auf diese zugreifen, falls die bereitgestellte Funktionalität nicht ausreicht[3].

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 sie in einem Programm 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 werden[4].

Somit bildet das Singleton–Entwurfsmuster eine passende Grundlage für die Nutzerverwaltung, da immer nur ein Benutzer angemeldet sein kann[5].

[3] vgl. [Balzert, 2005], Seite 367ff
[4] vgl. [Balzert, 2005], Seite 361ff
[5] Dies 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“ kritisiert[6].

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 weitergeben werden[7].

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–Klasse[8] 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 können[9]. Die Überprüfung der Eingabewerte ist somit in jedem Falle empfehlenswert und aufgrund des öffentlichen Charakters der Webanwendung hier besonders wichtig.

[6] vgl. [Hauer, 2010]
[7] vgl. [Theis, 2013], S. 417ff
[8] PDO: PHP Data Object
[9] vgl. [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 responsives Layout, welches sich an die unterschiedlichen Bildschirmgrößen vom Smartphone bis zum wandfüllenden HD–TV anpassen kann, anbietet[10].

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–Frontendramework: 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]

[10] Lizenz: Apache Licence V2.0, Download unter `http://getbootstrap.com/`
[11] Lizenz: MIT-Licence, Download unter `http://jquery.com/`

3 Datenbank–Schema

3.1 Konzeptuelles Datenbankschema: Entity–Relationship Diagramm

Hier werden die Gegenstände der realen Welt wie in Abbildung 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. Eine 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.

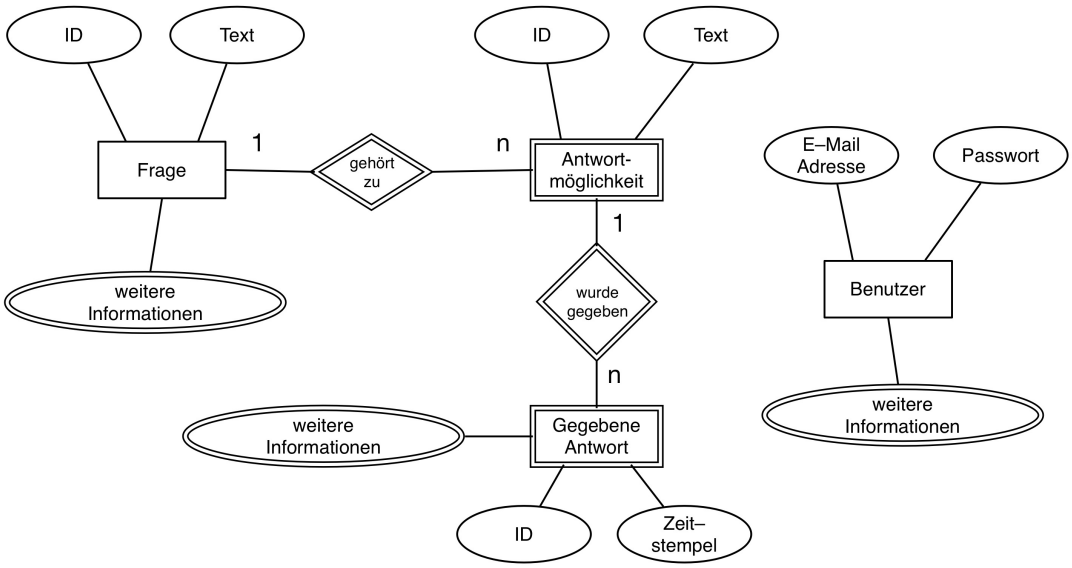


Abbildung 1: Entity Relationship Diagram

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–Fingerprinting[12] 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 1 auf Seite 7 dargestellten Modell ergibt sich die in Abbildung 2 dargestellten Relationen. Dieses Modell entspricht der dritten Normalform[13].

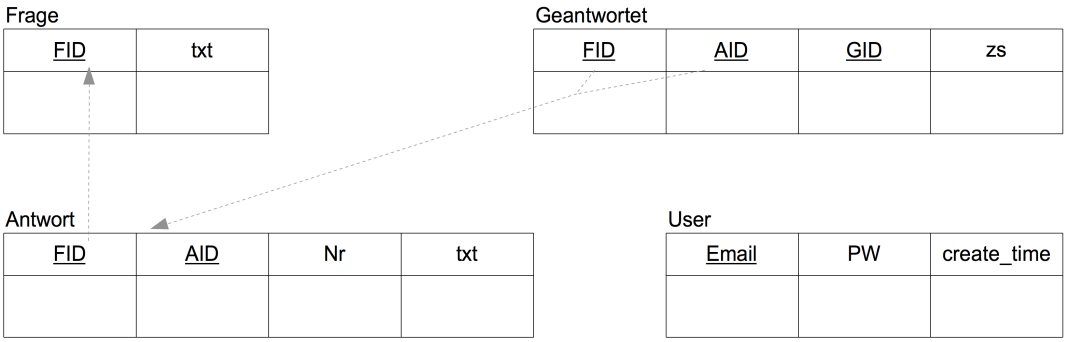


Abbildung 2: Relationales Modell

[12] Die Identifikation eines speziellen Nutzers durch Konfigurationsdetails des Webbrowsers, Bildschirmgröße, Betriebssystemversion, etc. (siehe auch `https://panopticlick.eff.org/`)
[13] Kriterien 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 seperaten 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 selbst[15] oder durch Angreifer.
Die hier reservierten 32 Byte sind für den in der MySQL–Dokumentation[17] empfohlenen MD5-Hash ausreichend. Da die entsprechende PHP–Dokumentation[18] hier allerdings eine genau entgegengesetzte Empfehlung gibt, ist dieser Sicherheitsaspekt für ein Produktivsystem nochmals genauer zu prüfen.

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

[14] Also der Hashwert des Passworts, welches zuvor mit applikationsspezifischen Zusatzdaten ergänzt wurde
[15] Spätestens seit dem Datenklau bei Vodafone[16] eine dokumentierte Gefahr
[17] [Oracle, 2013]
[18] [Olson (Hrsg.), 2013]

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:

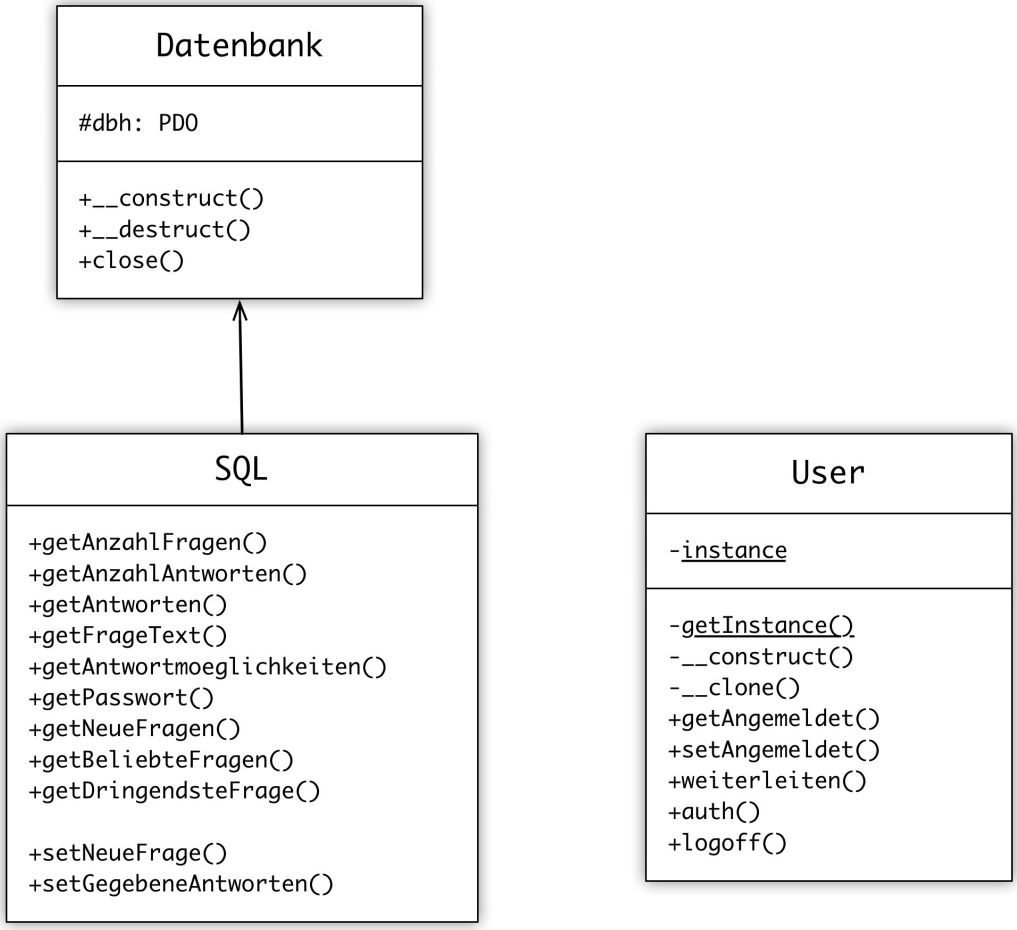


Abbildung 7: UML–Klassendiagramm

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 Attibut 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 zugrunde 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 Fassaden-Entwurfsmuster realisiert. Der aufrufende Code benötigt keinerlei Informationen über die Datenstruktur, die Datenbank oder Transaktionen. Änderungen an diesen technischen Details wirken sich lediglich auf die SQL–Klasse aus. Solange 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. 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 sicherzustellen, 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 `logoff()` meldet den momentan angemeldeten Nutzer ab, in dem sie 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 Anwendung

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 zusammengesetzt[19]. Der eigentliche Funktionsbereich[20] wird von den PHP–Dateien im Verzeichniss `.../pages/` übernommen.

```
<?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

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

[19] Darstellung vereinfacht

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

auf Desktoprechnern, Tabletts oder Smartphones zu kümmern. Jede Bildschirmgröße erhält durch Bootstrap automatisch ein speziell abgestimmtes Layout.

5.2 Layoutanpassungen mit CSS

Die Gestaltung des Layouts mit „Cascading Stylesheets“ (kurz: CSS) ermöglicht es, das durch das Bootstrap–Framework vorgegebene Aussehen durch Hinzufügen eines eigenen Stylesheets anzupassen.

So wurden bei der besprochenen Anwendung die Überschriften in der Seitenleiste individuell farbig unterstrichen. Die thematischen Blöcke werden hierzu in `DIV`–Blöcken mit den entsprechenden CSS–IDs eingeschlossen. Wie in Abbildung 9 auf Seite 18 zu sehen, werden die Links beim Berühren mit dem Mauszeiger in der gleichen Farbe dargestellt.

Bei der Darstellung der Auswertung werden die vom Benutzer selbst gegebenen Antworten durch Auszeichnung mit einer CSS-Klasse in fetter Schrift hervorgehoben.

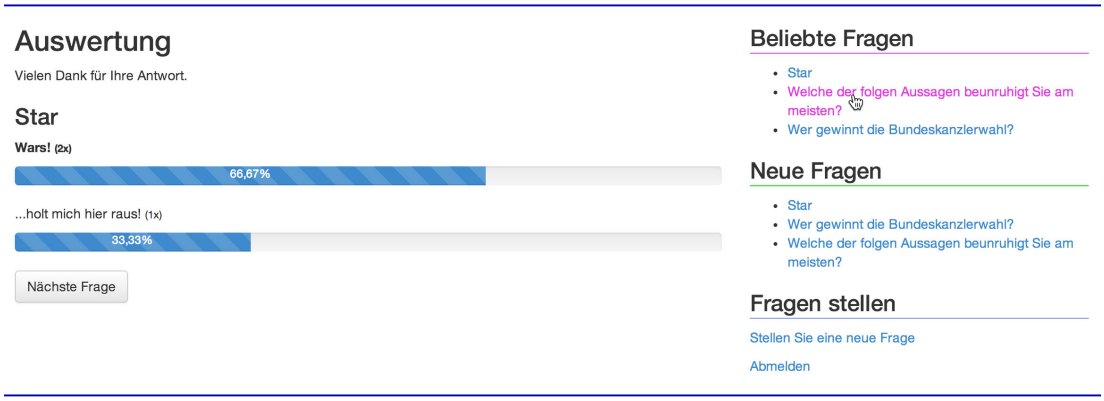


Abbildung 9: Unterstreichungen und Hervorhebungen per CSS

5.3 Mögliche Benutzerpfade durch die Anwendung

Abbildung 10 auf Seite 19 zeigt die Pfade, auf denen der Nutzer durch die Anwendung navigieren kann. Hierbei wurde die Darstellung wegen der besseren Übersichtlichkeit auf den jeweiligen Erfolgsfall reduziert. Bei einem Anmeldefehler erscheint direkt wieder das Anmeldeformular. Bei Datenbankfehlern (z.B. beim Abspeichern einer neuen Frage) erscheint die entsprechende PHP–Fehlermeldung.

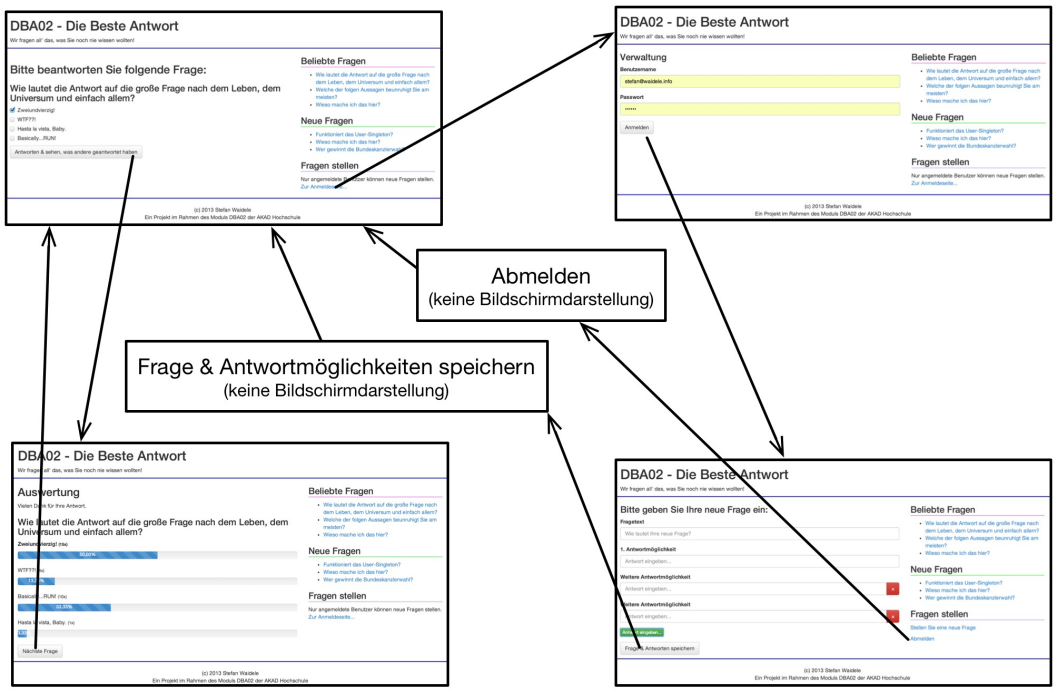


Abbildung 10: HTML–Seiten und Verlinkungen

5.4 Eingabeformular für neue Fragen

Zunächst stellt das Eingabeformular für neue Fragen und Antwortmöglichkeiten ein HTML–Formular dar, dessen Grundstruktur in Abbildung 11 gezeigt wird.

Hierbei ist zu beachten, dass lediglich ein Feld zur Eingabe einer einzigen Antwort zur Verfügung steht. Der HTML–Button mit der Aufschrift „Antwort eingeben...“ wird daher mit einer JavaScript–Funktion belegt, die ein weiteres Antwortfeld an

```
<form action="/index.php?show=speichern" method="post">
  <input type="text" name="frage" >
  <input type="text" name='awm[]' >
  <button type="button">
    Antwort eingeben...
  </button>
  <button type="submit">
    Frage &amp;amp; Antworten speichern
  </button>
</form>
```

Abbildung 11: HTML: Eingabeformular (vereinfacht)

der entsprechenden Stelle[21] im DOM-Baum[22] einfügt. Hierzu werden die vom jQuery–Framework bereitgestellten Funktionen genutzt. Diese übernehmen auch das optisch ansprechende Einblenden des neuen Felds.

```
$("#newanswer").click(function() {
  var na = $("HTML-Code: Weiteres Eingabefeld mit Button");
  na.hide();
  na.appendTo("#answers");
  na.slideDown();
});
```

Abbildung 12: jQuery: Weiteres Eingabefeld (vereinfacht)

Im Gegensatz zur ersten Antwortmöglichkeit sind alle weiteren Antwortmöglichkeiten mit einem HTML–Button versehen, der es erlaubt diese wieder zu löschen. Auch hierzu wird dem Button eine JavaScript–Funktion zugewiesen, die mithilfe von jQuery für das Ausblenden und Entfernen des Eingabefelds aus dem DOM–Baum sorgt:

```
$(".delanswer").click(function(){
 $(this).parent().parent().parent().slideUp("normal",
 function() {$(this).remove();}
);
```

Abbildung 13: jQuery: Eingabefeld löschen (vereinfacht)

[21] d.h. am Ende des Elements mit der ID „answers“

[22] DOM: Document Object Model, Browserinternes Interface zur Darstellung und Manipulation des HTML–Dokuments. Siehe auch `http://www.w3.org/DOM/`

Durch Klicken des Buttons mit der Aufschrift „Frage & Antworten speichern“ sendet der Nutzer die eingegebenen Werte an das Script `.../pages/speichern.php`. Da alle Antwort–Eingabefelder ein gleichlautendes `name`–Attribut besitzen, werden die Antwortmöglichkeiten dem Script in einem Array übergeben, welches der in Abschnitt 4.2.1 beschriebenen Methode der Klasse „SQL“ übergeben wird.

5.5 Benutzerverwaltung

Wie in Abschnitt 4.3 beschrieben, stellt die Klasse `user` alle Funktionen für die Authentifizierung der Seitenadministratoren bereit. Das PHP–Script unter `.../pages/anmeldung.php` nimmt daher lediglich den Benutzernamen und das Passwort über ein HTML–Formular entgegen und übergibt diese der Methode `User::auth()`.

```php
<?php
// Sessionverwaltung ist in die Klasse User integriert
$user = User::getInstance();
?>
```

Abbildung 14: PHP: Einbinden der Nutzerverwaltung

```php
<?php if (!$user->getAngemeldet()) { ?>
  Inhalte: Nicht angemeldete Besucher
<?php } else { ?>
  Inhalte: Angemeldete Nutzer
<?php } ?>
```

Abbildung 15: PHP: Einbinden der Nutzerverwaltung (vereinfacht)

Alle Teile der Anwendung erhalten über den in Abbildung 14 gezeigten Aufruf eine Referenz auf die Instanz der Benutzerverwaltung. Diese Code–Zeilen sind in der Datei `.../includes/header.php` eingefügt, welche von allen Seiten der Anwendung eingebunden wird.

An den Stellen, bei dem die Seitendarstellung für angemeldete Nutzer und nicht angemeldete Besucher unterschiedlich sein soll, wird der Anmeldestatus über

die Methode `User::getAngemeldet()` abgefragt und die Ausführung des PHP–Scripts entsprechend verzweigt. Als Beispiel wird hier die individualisierte Seitenleiste aus der Datei `.../includes/sidebar.php` in Abbildung 15 gezeigt.

6 Bewertung & Ausblick

Die vorliegende Anwendung implementiert eine Reihe von Funktionen, die nicht nur für die geforderte Funktionalität genutzt werden kann. So sind die Klassen „Datenbank“ und „User“ für andere Datenbankbasierte Anwendungen mit Benutzerverwaltung wiederverwendbar. Die Klasse „SQL“ kann als Vorlage für andere Abfragen genutzt werden.

Mit dem Apache–Module `mod_rewrite` und der im Quellcode enthaltene Konfigurationsdatei `.../.htaccess` sind besser lesbare URLs für die Seiten realisierbar. So könnte Frage Nummer 3 unter der Adresse `http://dba02.10100.de/frage/3` angezeigt werden. Da die Anwendung zur Beurteilung auch auf Systemen ohne `mod_rewrite` lauffähig sein soll, wurde hierauf bewusst verzichtet. Alle Links der Anwendung nutzen das interne Format mit sichtbarer Parameterübergabe (z.B. `http://dba02.10100.de/index.php?show=frage&fid=3`).

Vor einem Einsatz in einem Produktivsystem ist allerdings noch eine wirksame Verschlüsselung der Benutzerpasswörter zu implementieren[23].

Für die gegebene Aufgabenstellung war die Nutzung von PHP–Includes zur Sicherstellung eines einheitlichen Designs ausreichend. Für komplexere Aufgaben wäre eine Implementierung nach dem MVC–Musters mit der Nutzung eines Templating–Frameworks aufgrund der höheren Übersichtlichkeit und Robustheit ratsam.

Durch die Nutzung von PDO für den Datenbankzugriff ist die Anwendung für verschiedene Einsatzumgebungen gerüstet. Ein Wechsel der genutzten Datenbank ist durch die Anpassung weniger Zeilen der Konfigurationsdatei möglich.

Aufgrund der Modellierung von zwei schwachen Entities wird die Frage–ID und die Antwort–ID mehrfach in den Tabellen gespeichert. Bei den SQL–Joins muss nicht nur ein Element sondern alle Elemente des zusammengesetzten Schlüssels

[23] Vgl. Abschnitt 3.3.1

verglichen werden. Eine weiterführende Arbeit könnte untersuchen, welche Auswirkungen eine Modellierung mit starken Entities auf den Speicherverbrauch und auf die Geschwindigkeit der Datenbankanfragen hätte.

A Bildschirmfotos

Abbildung 16: Screenshot: Frage

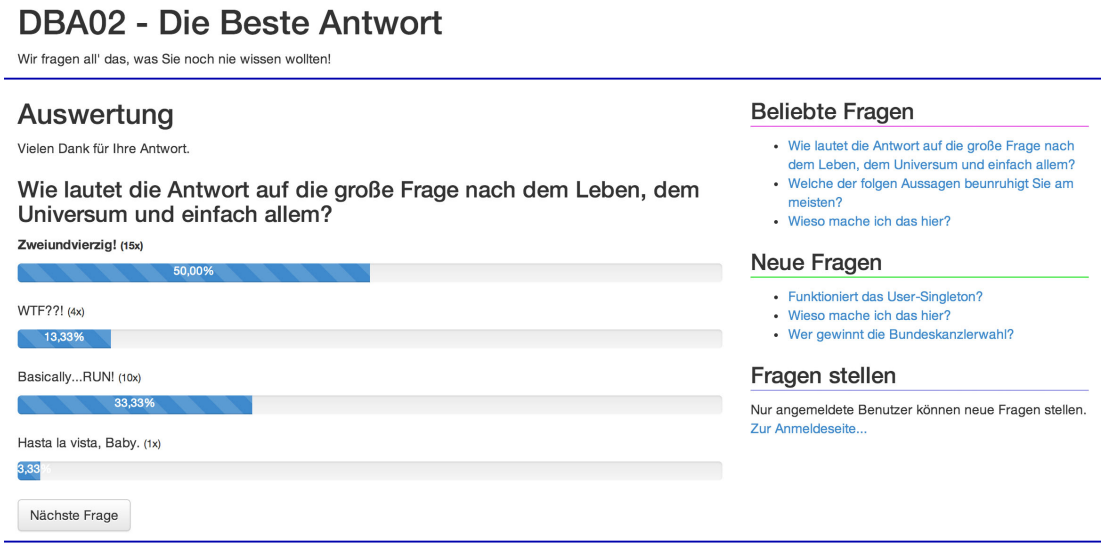


Abbildung 17: Screenshot: Auswertung

Abbildung 18: Screenshot: Anmeldung

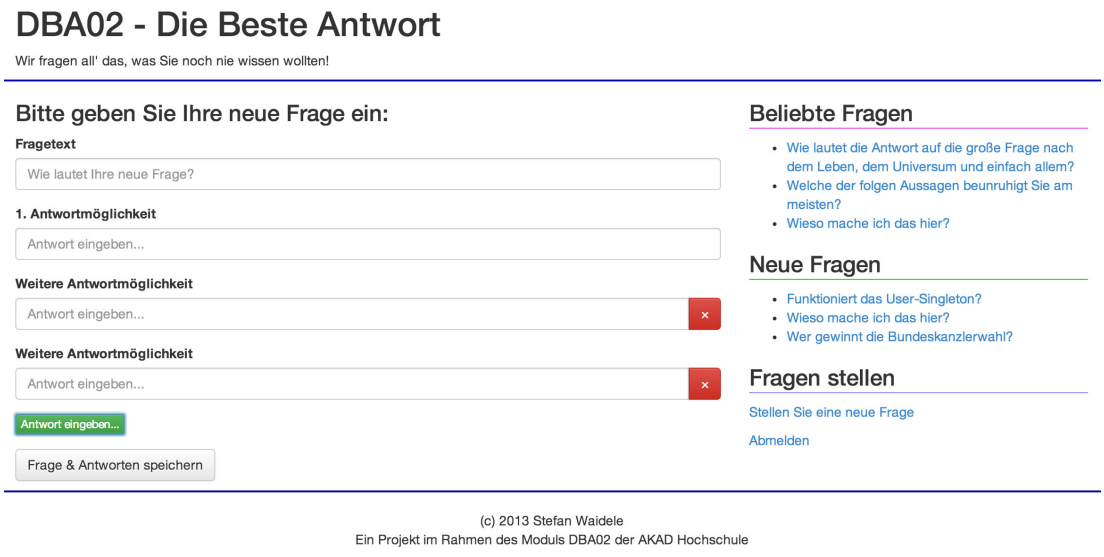


Abbildung 19: Screenshot: Neue Frage

B Verzeichnisstruktur

```
\- bootstrap/:
|  |
|  \-- ...
|
\- class/:
|  |
|  \-- Datenbank.php
|  \-- SQL.php
|  \-- User.php
|
\- includes/:
|  |
|  \-- dbconf.ini
|  \-- footer.php
|  \-- header.php
|  \-- sidebar.php
|
\- pages/:
|  |
|  \-- abmeldung.php
|  \-- anmeldung.php
|  \-- auswertung.php
|  \-- frage.php
|  \-- neuefrage.php
|  \-- speichern.php
|
\- sql/:
|  |
|  \-- createdatabase.sql
|  \-- queries.sql
|
\-- .htaccess
\-- index.php
\-- style.css
```

Abbildung 20: Verzeichnisstruktur

C Quellcode

Der Quellcode der erstellten Anwendung befindet sich auf der diesem Assignment beigefügten CD–ROM. Außerdem steht er in der Modulspezifischen Arbeitsgruppe sowie unter `https://github.com/stwaidele/DBA02-Site` zur Verfügung.

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.philipphauer.de/study/se/design-pattern/singleton.php#nachteile`.

[Olson (Hrsg.), 2013] Olson (Hrsg.) (2013). PHP Manual: Safe Password Hashing, Abruf am 25.10.2013. `http://www.php.net/manual/de/faq.passwords.php#faq.passwords.fasthash`.

[Oracle, 2013] Oracle (2013). MySQL 5.5 Manual: Encryption and Compression Functions, Abruf am 25.10.2013. `https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_md5`.

[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.

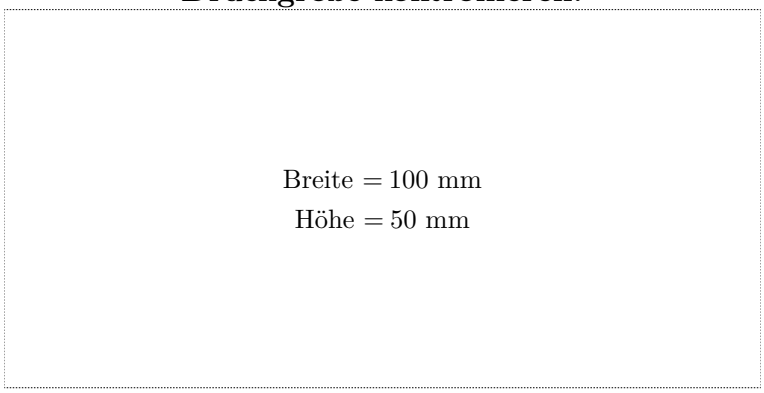
[Theis, 2013] Theis (2013). *Einstieg in PHP 5.5 und MySQL 5.6*. Galileo Computing, Bonn.

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.

(Datum, Ort) (Unterschrift)

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —