Internetserver-programmierung

Projektbericht im Rahmen der Portfolio-Prüfung SoSe 2020

**2. Zeitraum: 06.07.2020 – 02.08.2020**

**Gruppe 16**

**Gruppenmitglied: Felicitas Yeboah**

**Gruppenmitglied: Christian Till**

**SEITE FÜR FREITEXT - MUSS VOR ABGABE GELÖSCHT WERDEN!**

**Issues/Fragen: -> Meinungen und Hinweise dazu gerne kundtun! 😉**

* **Soll ein Termin mehrere Kategorien haben können?**

**wenn ja -> zusätzliche Datenbanktabelle für m:n-Beziehung nötig**

**23.07.2020: nachdem ich nun etwas drüber nachgedacht habe, fällt mir keine schöne Oberfläche für die Pflege ein. Ich denke, ein Termin sollte nur eine Kategorie jaben können, dann kann man das einfach über eine DropDown-Liste machen. So ist es auch beim Google-Kalender… -> Nee, würde auch sagen, dass nur eine Kategorie pro Termin zugelassen ist.**

* **Sind eigene Klassen zur Kommunikation mit der Datenbank sinnvoll?**

**Dann könnte man das Entwurfsmuster „Fassade“ verwenden. Dies entspricht im Grunde einer Klasse „Datenbank“, welche dann abhängig von der Art der Datenbank agieren kann … allerdings haben wir ja nur eine Datenbank und dann ist das wohl zu viel. Trotzdem sollte es eine Klasse für ein Datenbank-Objekt geben, welche dann entsprechend die DB-Zugriffe übernimmt. Denke, wenn wir uns für Factory und Observer entscheiden, reicht das aus.**

* **Nutzen von PDO oder MySQLi?**

**Im Skript nur MySQLi, aber im Kurs-Forum hat sie auch auf PDO verlinkt. Daher scheint beides in Ordnung zu sein.**

[**https://websitebeaver.com/php-pdo-vs-mysqli**](https://websitebeaver.com/php-pdo-vs-mysqli)**:**

**Aufgrund des dort genannten Vorteils für PDO:**

**„**Allow to pass variables and values directly into execute“

**würde ich mich dafür entscheiden.**

* **Genau deswegen würde ich auch PDO verwenden. Bisher verwenden wir aber auch PDO. Sollte so bleiben.**

**Ich stelle mir den Ablauf nämlich folgendermaßen vor.**

**Es gibt eine Klasse für das Objekt „Termin“ und eine Klasse für die Datenbankanbindung.-> Die Klasse für die Datenbankverbindung ist ja quasi bereits die PDO-Klasse (Die gibt die Verbindung zur Datenbank zurück) Wenn wir das PDO-Objekt nochmal in einer weiteren Klasse unterbringen, wird es schwer Funktionen der PDO-Klasse aufzurufen (z.B. exec() Siehe auch meine Anmerkung in der index.php Datei. Oder? Wir müssten das PDO-Objekt dann immer über Getter abrufen.**

**Wir könnten den Verbindungsaufbau auch als Rückgabewert einer Methode einer Klasse zurückgeben lassen (Siehe DB\_Conf.php).**

**Die Klasse Datenbank dann komplett in die Klasse Kalender implementieren**

**Die Klasse Kalender würde dann alle bisherigen Funktionen der Klasse Datenbank beinhalten und Funktionen wie „lese Termin“, „schreibe Termin“, etc.**

**Die Datenbank-Klasse implementiert so eine Art REST-Server und beinhaltet alle Funktionen, welche für die Interaktion mit der Datenbank benötigt wird, z.B.**

* + **Lese Termine für Monat**
  + **Lese Termine für Kategorie xy**
  + **Schreibe Termin**

**Das könnte die Klasse Kalender ebenfalls tun. Der Kalender ist quasi die Datenbank, in der die Termine gespeichert werden.**

**Die Klasse Termin hält wiederum in den Membern die Daten des Termins (bis diese an die Datenbank übergeben werden). Und bietet unter anderem eine Funktion, welche den Termin, aufbereitet in HTML-Code, ausgibt. Ja, sehe ich genau so.**

**Ein paar Links -> eventuell später als Quelle angeben:**

[**https://webdeasy.de/json-in-javascript-und-php-benutzen/**](https://webdeasy.de/json-in-javascript-und-php-benutzen/)

[**https://www.mediaevent.de/javascript/json.html**](https://www.mediaevent.de/javascript/json.html)

[**https://phpdelusions.net/pdo/objects**](https://phpdelusions.net/pdo/objects)

**https://dzone.com/articles/practical-php/practical-php-patterns-data**

1. **Spezifikation**
   1. **Benutzergruppen**
   2. **Funktionale und nicht-funktionale Anforderungen**
2. **Entwurf**
   1. **Klassendiagramm**
   2. **Entwurfsmustereinsatz**

**Observer**

Bei Änderung eines Termins wird die Anzeige des Kalenders aktualisiert.

**Fabrik**

Implementiert einen Termin, der sich in der Kategorie unterscheidet oder Erzeugung eines Objekts für die Datenbank.

**Das klingt als wären wir uns reinig. Ich hatte zu Beginn mal folgendes notiert:**

**Verhaltensmuster - Strategie (Strategy Pattern) für -> Erstellen eines Termins und hinzufügen einer Kategorie ---🡪 Factory Method**

**und**

**Observer Pattern (Änderungen an Terminen, hinzufügen von Terminen) -> update(Terminkalender)**

1. **Implementierung**
   1. **Screenshots**
   2. **Genutzte Frameworks**
2. **Zusammenfassung**
   1. **Aufgabenteilung**
   2. **Ausblick**