

E-Bibliothek

Login/Logout

Mirza Kuljancic



Inhaltsverzeichnis

1. Aufgabenstellung.....	2
2. Aufwandschätzung	2
3. Designüberlegung.....	4
4. Datenbankdesign.....	5
5. Designmöglichkeiten für Implementierung des Logins.....	8
6. Implementierung.....	9
6.3.1 Login	10
6.3.2 Auf Benutzereingaben reagieren	10
6.3.3 Logout.....	10
6.4 Auf Benutzereingaben reagieren	10
7. Testfälle (registrieren)	11
8. Smarty	14
8.1 Installation	14
9. Quellen	16

1. Aufgabenstellung

Um die in den verschiedenen fachspezifischen Fächern erworbenen Kenntnisse zu demonstrieren und zu festigen soll ein gemeinsames Abschlussprojekt fächerübergreifend realisiert werden. Insbesondere soll dabei Augenmerk auf die Kompetenzen „User-Interface-Design“, „verteilte Systeme“, „Web Services“ sowie „Softwareentwicklung“ gelegt werden. Es soll ein Anwendung für eine Online-Bibliothek und Lese-Seite für quelloffene Bücher erstellt werden. Der Zugriff auf die Inhalte dieser Bibliothek soll sowohl von Desktop-Systemen (Webseite) als auch von mobilen Geräten (Webseite oder App) möglich sein.

2. Aufwandschätzung

Task	Beschreibung	Geschätzte Zeit in Stunden
Erzeugung des XML-Schemas	Schema wurde erstellt und die Datenbank in Propel erstellt	3 h
Recherche nach bestehenden Produkten	Recherche nach Logins und Logouts	1,5 h
Erstellung des UMLs	UML für das Login, Logout und für die gesamten Benutzerfunktionen erstellen	3 h
Erstellung des Aktivitätsdiagramms	Aktivitätsdiagramm erstellen	2 h
Programmierung des Logins	Das Login programmieren + Codedokumentation	6 h
Programmierung des Logouts	Das Logout programmieren + Codedokumentation	3 h
Erstellung der Tests	Erstellung der Tests für das Registrieren	2,5 h
Einbindung in Smarty	In Smarty einbinden	3,5h

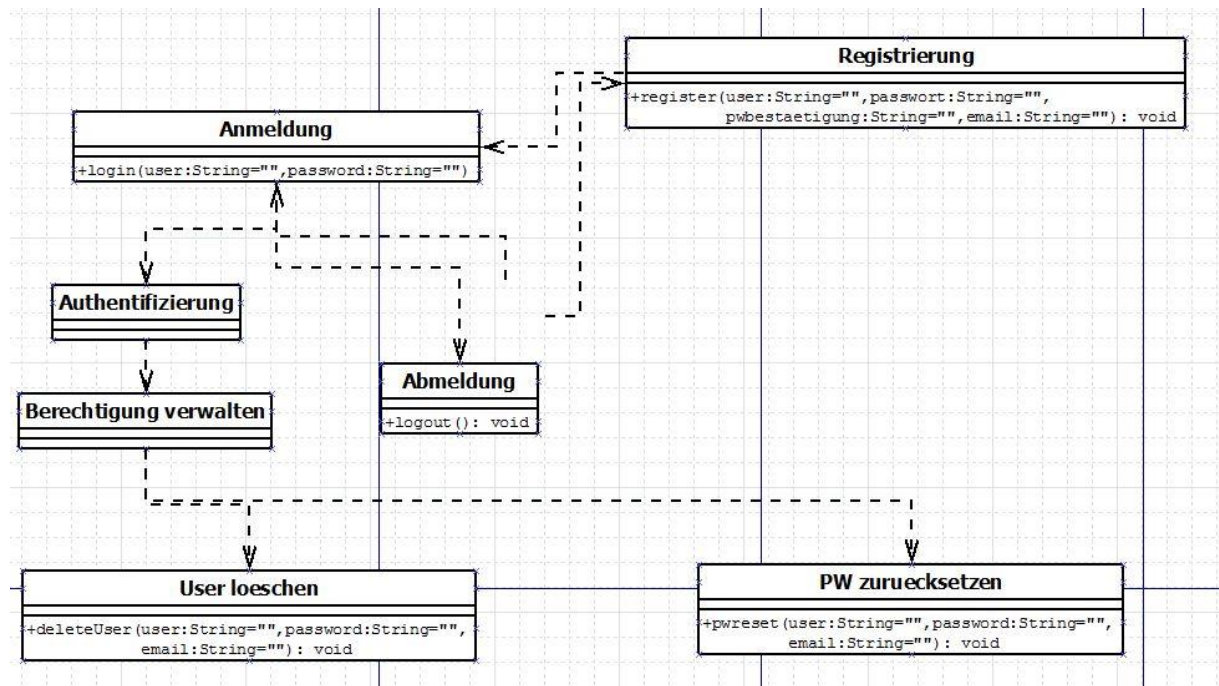
Geschätzter Gesamtaufwand des Logins und Logouts: 24,5 h

Task	Beschreibung	Tatsächliche Zeit in Stunden
Erzeugung des XML-Schemas	Schema wurde erstellt und die Datenbank in Propel erstellt	2,5 h
Recherche nach bestehenden Produkten	Recherche nach Logins und Logouts	0,5 h
Erstellung des UMLs	UML für das Login, Logout und für die gesamten Benutzerfunktionen erstellen	2,5 h
Erstellung des Aktivitätsdiagramms	Aktivitätsdiagramm erstellen	2,5 h
Programmierung des Logins	Das Login programmieren + Codedokumentation	5 h
Programmierung des Logouts	Das Logout programmieren + Codedokumentation	2,5 h
Erstellung der Tests	Erstellung der Tests für das Registrieren	2,5 h
Einbindung in Smarty	In SMarty einbinden	2,5 h

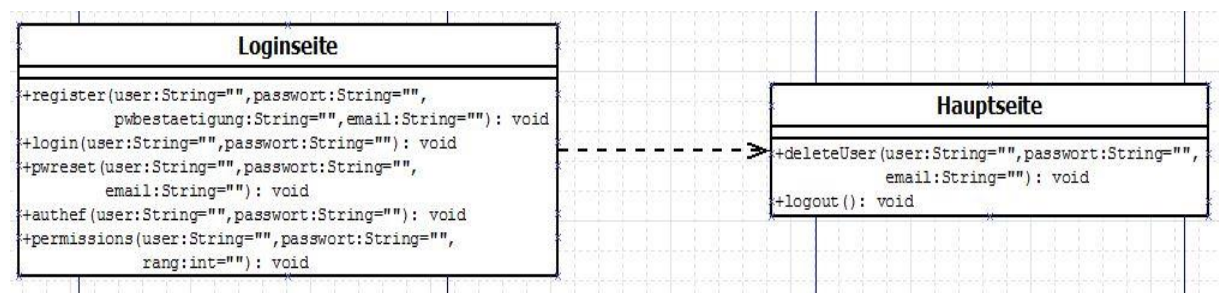
Tatsächlicher Gesamtaufwand des Logins und Logouts: 20,5 h

3. Designüberlegung

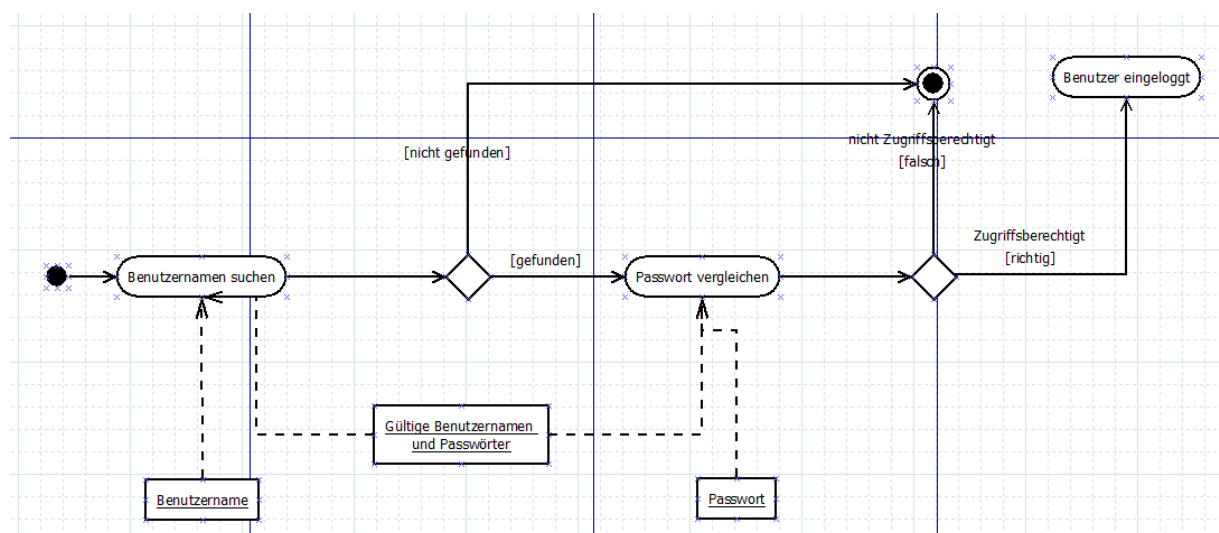
Das UML Diagramm (Gesamt):



Das UML Diagramm (Login):



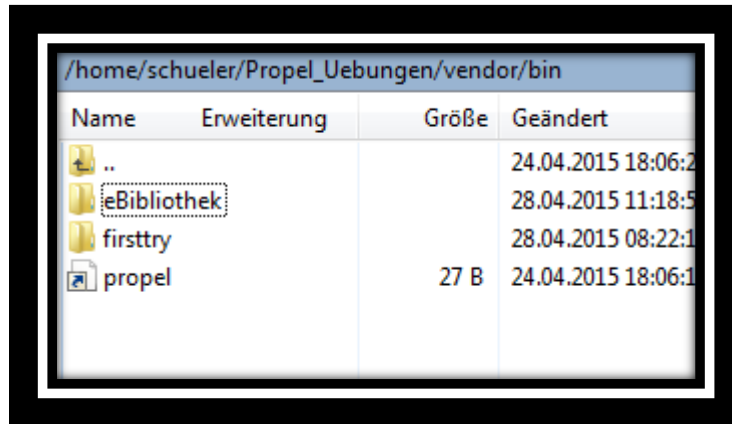
Das Aktivitätsdiagramm:



4. Datenbankdesign

4.1 Erstellung des XML-Schema und Vergleich mit `propel sql:build`

Verzeichnis anlegen



Schema.xml:

Man muss das Schema erstellen und als „schema.xml“ abspeichern. Das schema.xml habe ich zunächst mittels Notepad++ persönlich am Rechner erstellt, dann mittels WinSCP in den Ordner verschoben. [1]

Vergleich des Schemas mit `propel sql:build`:

Nach dem ich das schema.xml erzeugt habe, habe ich in der VMware im Ordner den Befehl `$propel sql:build` eingegeben. Es wurde ein Ordner „generated-sql“ mit dem dazugehörigen sql-File erstellt. Wie man in den folgenden Teilabschnitten sehen kann, sind sowohl das sql-File als auch das Schema-File ziemlich ähnlich: [2]

Schema-File:

```
<?xml version="1.0" encoding="UTF-8"?>
<database name="ebibliothek" defaultIdMethod="native">
  <table name="user" phpName="Book">
    <column name="UserID" type="integer" required="true" primaryKey="true" autoIncrement="true"/>
    <column name="benutzername" type="varchar" size="255" required="true" />
    <column name="passwort" type="varchar" size="24" required="true"/>
    <column name="rolle" type="integer" required="true"/>
  </table>
</database>
```

SQL-File:

```
# This is a fix for InnoDB in MySQL >= 4.1.x
# It "suspends judgement" for fkey relationships until are tables are set.
SET FOREIGN_KEY_CHECKS = 0;

-----
-- user
-----

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user`
(
  `UserID` INTEGER NOT NULL AUTO_INCREMENT,
  `benutzername` VARCHAR(255) NOT NULL,
  `passwort` VARCHAR(24) NOT NULL,
  `rolle` INTEGER NOT NULL,
  PRIMARY KEY (`UserID`)
) ENGINE=InnoDB;

# This restores the fkey checks, after having unset them earlier
SET FOREIGN_KEY_CHECKS = 1;
```

4.2 Generieren der PHP-Klassen für das Model

In der VMware habe ich den Befehl `$propel model:build` eingegeben. Es wurde ein Ordner „generated-classes“ mit dem dazugehörigen PHP-Files erstellt. Es wurde für diese eine Tabelle aus dem Schema eine PHP Klasse erstellt.

4.3 Composer.json

Dieser Teilabschnitt muss in den composer.json eingefügt werden:

```
"autoload": {
  "classmap": ["generated-classes/"]
}
```

Anschließend muss folgender Befehl eingefügt werden:

`php /composer.phar dump-autoload`

und danach:

propel config:convert

Es sind noch folgende 2 Zeilen notwendig, um auf die erzeugten Propel Verzeichnisse zugreifen zu können:

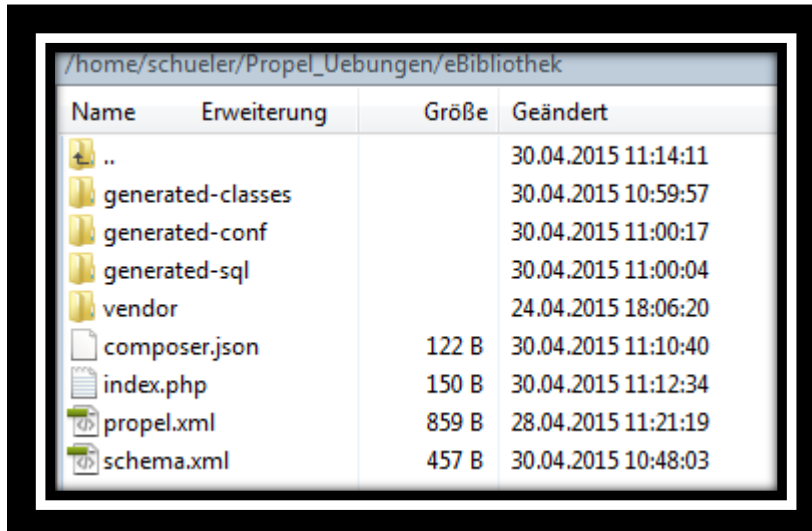
```
// setup the autoloading
require_once '/path/to/vendor/autoload.php';

// setup Propel
require_once '/generated-conf/config.php';
```

require_once: PHP prüft hier allerdings ob die gewünschte Datei bereits eingebunden wurde und wird sie in diesem Fall nicht ein weiteres Mal einbinden.

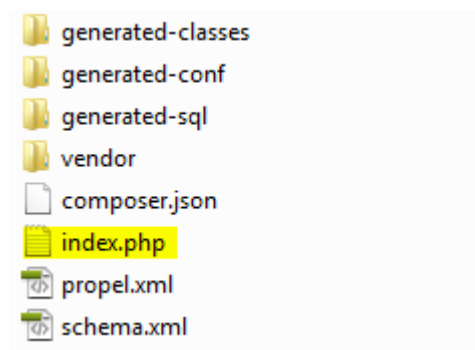
4.4 Fertigstellung

Wenn man alles fertiggestellt hat, muss das Gesamtergebnis so aussehen:

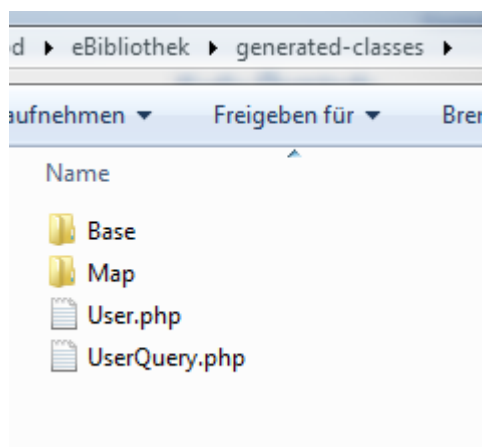


Name	Erweiterung	Größe	Geändert
..			30.04.2015 11:14:11
generated-classes			30.04.2015 10:59:57
generated-conf			30.04.2015 11:00:17
generated-sql			30.04.2015 11:00:04
vendor			24.04.2015 18:06:20
composer.json		122 B	30.04.2015 11:10:40
index.php		150 B	30.04.2015 11:12:34
propel.xml		859 B	28.04.2015 11:21:19
schema.xml		457 B	30.04.2015 10:48:03

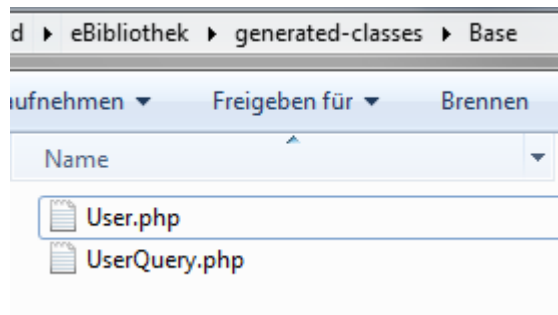
index.php ist unser Ausgangsfile, aus diesem navigieren wir uns zu unseren Daten.



Es wird ausschließlich auf den Base Ordner zugegriffen.



In dem Ordner befinden sich für uns zwei wichtige Files. Nur auf diese Files darf und soll zugegriffen werden.



4.5 Sicherheitsmechanismus in Propel

Propel überprüft das intern. Es überprüft gleich für uns die Benutzereingabe. Propel hat uns schon Daten vorgefertigt, wenn wir die Methode `findOnebyBenutzer`, weiß es schon, dass es eine Zeichenkette mit bestimmten Vorschriften ist.

5. Designmöglichkeiten für Implementierung des Logins

Variante 1:

Implementierung des Logins direkt in mitten des HTML Codes.

- + Code ist direkt dort wo er verwendet wird
- Kann nicht getrennt getestet werden
- Kann nicht wiederverwendet werden

Variante 2:

Eigene Klasse mit Methoden zur Durchführung des Logins

- + Trennung von Logik und Design
- + Kann wiederverwendet werden
- + Kann getrennt getestet werden

Variante 3:

Funktionen zur Durchführung des Logins

- + Trennung von Logik und Design
- + Kann wiederverwendet werden
- + Kann getrennt getestet werden

Entscheidung:

Aufgrund der Anzahl - wird klar gegen Variante 1 entschieden.

Variante 2 ist nicht wirklich notwendig da wir nie mehrere Instanzen(das was mit `new` angelegt wird) des Objektes "Login" haben werden.

Variante 3 ist etwas weniger Aufwand als Variante 2, und gerade ausreichend für unsere Zwecke.

-> deshalb wurde Variante 3 umgesetzt.

Beschreibung der Umsetzung:

Auslagerung der implementierten Funktionen in eine separate Datei

*) damit sie nur dann eingebunden werden müssen wenn wir sie brauchen

*) sie in den Tests eingebunden werden kann

*) definieren von eigenen Konstanten damit im Design-teil das Ergebnis der Authentifizierung geprüft werden kann

**) Separater Status für: Passwort falsch, Username Falsch

Diese werden im Design unterschieden um dem Benutzer Feedback über das Ergebnis zu geben.

**) Rückgabe der Propel Userklasse wenn der Login erfolgreich war, damit diese in die Session geschrieben werden kann.

Prüfen ob eingeloggt:

Zum Prüfen ob ein Benutzer eingeloggt ist wurde den Kollegen ein Codesnippet zur Verfügung gestellt.

















Dieses Snippet prüft ob ein Benutzer (Propel Objekt) in der Session gespeichert ist, wie es vom Loginprozess bei einer erfolgreichen Authentifizierung angelegt wird.

Logout:

Umsetzung des logouts in eine separate Datei (logout.php) weil es die einfachste Variante ist. Hier wird die Session beendet was zur Löschung des Benutzerobjektes in der Session führt.

6. Implementierung

Es sind einige Files drinnen. Als erstes das Login.php, das Login.fnc.php und das Logout.php File, das sind die Files wo sich die Funktionalität verbirgt. Anschließend gibt es noch das form File und das form_logout File, das die GUI beinhaltet.

	.gitignore	04.06.2015 11:18	Textdokument
	_buildpath	29.04.2015 03:56	Datei
	_project	29.04.2015 03:56	Datei
	composer.json	30.04.2015 11:10	JSON-Datei
	EBiblio_Hauptseite.php	14.05.2015 15:47	PHP-Datei
	form.html	14.05.2015 15:36	Chrome HTML D
	form_logout.html	10.06.2015 11:22	Chrome HTML D
	login.fnc.php	10.06.2015 11:20	PHP-Datei
	login.php	10.06.2015 11:22	PHP-Datei
	login_test.php	09.06.2015 11:22	PHP-Datei
	logo.gif	29.04.2015 03:56	GIF-Bild
	logout.php	10.06.2015 11:20	PHP-Datei
	phpunit.phar	09.06.2015 10:56	PHAR-Datei
	propel.xml	30.04.2015 13:01	XML-Dokument
	README.md	29.04.2015 03:57	MD-Datei
	register_test.php	15.06.2015 09:15	PHP-Datei

6.3.1 Login

```
function Benutzer_login($username,$password) {
//Wir suchen den Benutzer mit dem Namen $username aus der Datenbank
    $userQ = new UserQuery();
    //username muss nicht ueberprueft werden, da Propel die Sicherheitsmechanismen ueberprueft.
    $user = $userQ->findOneByBenutzername($username);
    if(!is_null($user)){
        if ($user->getPasswort() == $password){
            //Daten korrekt, Login wird ausgefuehrt
            return $user;
        }else{
            //Das Passwort passt nicht zum Benutzer
            return LOGIN_PWFALSE;
        }
    }

    //Es wurde kein Benutzer mit dem Namen $user gefunden
    return LOGIN_USERFALSE;
}
```

6.3.2 Auf Benutzereingaben reagieren

Benutzer Eingabe	Funktioniert	Funktioniert nicht
1.) Leer	X	
2.) Pw stimmen nicht überein	X	
3.) Benutzername ist falsch	X	
4.) Richtig	X	

6.3.2.1 Leere Eingabe

Sie haben einen falschen Benutzer! ?

6.3.2.2 Passwörter stimmen nicht überein

Sie haben ein falsches Passwort eingegeben! ?

6.3.2.3 Benutzername stimmt nicht überein

Sie haben einen falschen Benutzer! ?

6.3.2.4 Richtige Eingabe

6.3.3 Logout

```
//damit das PHP am Server eine Session anlegt damit wir was einspeichern koennen
session_start();

//loescht die Session vom Server, somit ist der Benutzer nicht mehr eingeloggt
session_destroy();
```

6.4 Auf Benutzereingaben reagieren

Benutzer Eingabe	Funktioniert	Funktioniert nicht
1.) Ausloggen richtig	X	

6.4.1 Ausloggen

testuser Log out

e-Library testuser ... Sign in

Registrierung

Benutzername

Passwort

Passwort

Registrieren

© TGM, Wexstrasse 19-23

7. Testfälle (registrieren)

7.3.1.1 testerfolg

Rwg_richtigkeit_überprüfen, ob eine gültige Eingabe auch als gültig erkannt wird.

```
/**
 * Testen einer erfolgreichen
 * Registrierung
 */
public function testerfolg(){
    $bname = "asdf112";
    $pw1 = "asdfg1";
    $pw2 = "asdfg1";

    $status = Reg_richtigkeit_ueberprufen($bname, $pw1, $pw2);
    $this->assertEquals(REG_ERFOLGREICH, $status);
}
```

7.3.1.2 testleer

leerer String überprüft, ob er erkennt, ob ein Feld leer gelassen wird.

```
/**
 * Testen einer leeren Eingabes
 */
public function testleer(){
    $bname = "asdf112";
    $pw1 = "asdfg1";
    $pw2 = "asdfg2";

    $status = Reg_richtigkeit_ueberprufen("", $pw1, $pw2);
    $this->assertEquals(REG_FEHLER_LEER, $status);

    $status = Reg_richtigkeit_ueberprufen($bname, "", $pw2);
    $this->assertEquals(REG_FEHLER_LEER, $status);

    $status = Reg_richtigkeit_ueberprufen($bname, $pw1, "");
    $this->assertEquals(REG_FEHLER_LEER, $status);
}
```

7.3.1.3 testpwgleich

Prüft ob er erkennt, dass die Passwörter nicht übereinstimmen.

```
/**
 * Testen der gleichheit der PW
 */
public function testpwgleich(){
    $bname = "asdf112";
    $pw1 = "asdfg1";
    $pw2 = "asdfg2";

    $status = Reg_richtigkeit_ueberprufen($bname, $pw1, $pw2);

    $this->assertEquals(REG_FEHLER_UNGLEICHPW, $status);
}
```

7.3.1.4 testzeichen

Überprüft ob er erkennt, dass Sonderzeichen eingefügt wurden. Haben auch ein gültiges eingegeben, das er überprüft, ob nicht fehlerhafte Zeichen eingegeben wurden,

```
/**
 * Testen der gueltigen Zeichen fuer die
 * Eingabe
 */
public function testzeichen(){
    $bname = "asdf112";
    $pw1 = "asdfg1";
    $pw2 = "asdfg2";

    $status = Reg_richtigkeit_ueberprufen($bname,$pw1,$pw2);
    $this->assertNotEquals(REG_FEHLER_ZEICHENUNG, $status);

    $status = Reg_richtigkeit_ueberprufen("asdfasdf??!", $pw1,$pw2);
    $this->assertEquals(REG_FEHLER_ZEICHENUNG, $status);
}
```

7.3.1.5 testzeichen

Länge muss stimmen. 1 muss er sagen das alles passt. 2 kurzer Benutzername Fehler, Passwort zu kurz Problem mit dem Passwort.

```
/**
 * Testen der gueltigen Lange der Eingaben
 */
public function testlengte(){
    $bname = "asdf";
    $pw = "asdfg1";

    $status = Reg_richtigkeit_ueberprufen($bname,$pw,$pw);
    $this->assertNotEquals(REG_FEHLER_LAENGE, $status);

    $status = Reg_richtigkeit_ueberprufen("123", $pw,$pw);
    $this->assertEquals(REG_FEHLER_LAENGE, $status);

    $status = Reg_richtigkeit_ueberprufen($bname,"12345","12345");
    $this->assertEquals(REG_FEHLER_LAENGE, $status);
}
```

Um den Test erfolgreich auszuführen, wechselt man in den Ordner wo sich die „test.bat“ Datei befindet und man führt im CMD dann die Datei test.bat aus.

```
C:\Program Files (x86)\Server\Apache2.2\htdocs\E-BIBLIOTHEK>cd E_Biblio
C:\Program Files (x86)\Server\Apache2.2\htdocs\E-BIBLIOTHEK\E_Biblio>test.bat
C:\Program Files (x86)\Server\Apache2.2\htdocs\E-BIBLIOTHEK\E_Biblio>php phpunit.phar --bootstrap vendor/autoload.p
hp login_test.php
PHPUnit 4.7.2 by Sebastian Bergmann and contributors.

...
Time: 5.38 seconds, Memory: 10.00Mb
OK (3 tests, 6 assertions)
C:\Program Files (x86)\Server\Apache2.2\htdocs\E-BIBLIOTHEK\E_Biblio>
```

8. Smarty

Vorteil: Design und Code getrennt und es ist sehr praktisch für Designer. **[3]**

8.1 Installation

Zuerst habe ich Smarty runtergeladen und den Ordner „Smarty“ in htdocs Ordner eingefügt.

smarty	15.06.2015 15:23	Dateiordner
templates_c	16.06.2015 16:49	Dateiordner
vendor	08.05.2015 14:17	Dateiordner
.gitignore	04.06.2015 11:18	Textdokument
_buildpath	29.04.2015 03:56	Datei
_project	29.04.2015 03:56	Datei
composer.json	30.04.2015 11:10	JSON-Datei
EBiblio_Hauptseite.php	16.06.2015 16:55	PHP-Datei
form.tpl	16.06.2015 16:38	TPL-Datei
form_logout.tpl	16.06.2015 16:47	TPL-Datei
login.fnc.php	10.06.2015 11:20	PHP-Datei
login.php	16.06.2015 16:52	PHP-Datei
login_test.php	09.06.2015 11:22	PHP-Datei
logo.gif	29.04.2015 03:56	GIF-Bild
logout.php	10.06.2015 11:20	PHP-Datei

Da wir die Smarty.class.php einbinden müssen, habe ich die libs Datei aus dem Ordner Smarty genommen und es dann eingefügt.

js	08.05.2015 14:21	Dateiordner
libs	16.06.2015 16:29	Dateiordner
smarty	15.06.2015 15:23	Dateiordner
templates_c	16.06.2015 16:49	Dateiordner
vendor	08.05.2015 14:17	Dateiordner
.gitignore	04.06.2015 11:18	Textdokument
_buildpath	29.04.2015 03:56	Datei
_project	29.04.2015 03:56	Datei
composer.json	30.04.2015 11:10	JSON-Datei
EBiblio_Hauptseite.php	16.06.2015 16:55	PHP-Datei
form.tpl	16.06.2015 16:38	TPL-Datei

Anschließend habe ich das File, in die Files Login.php und EBiblio_Hauptseite eingebunden und ein Smarty Objekt erzeugt.

```
//Einbindung des Files
require_once 'libs/Smarty.class.php';
//Legen uns das Smarty Objekt an
$smarty = new Smarty();
```

Danach muss man die HTML Files umändern auf tpl. Anschließend verwendet man in den TPL Files statt dem üblichen PHP-Code eine eigene Smarty-Syntax.

```

</div>
<div id="navbar" class="navbar-collapse collapse">
  <form class="navbar-form navbar-right" method="POST">
    <!--Fehlermeldung ausgeben wenn bname oder passw falsch sind-->
    {if isset($usernotfound)}

      <div class="form-group has-error">
        <!--span einfach das der Text nicht frei im div ist-->
        <span>Sie haben einen falschen Benutzer!</span>
      </div>
    {elseif isset($pwfalsch)}

      <div class="form-group has-error">
        <span>Sie haben ein falsches Passwort eingegeben!</span>
      </div>
    {/if}
  </form>
</div>

```

Dies habe ich in der form.tpl und form_logout.tpl Klasse gemacht. Das heißt ich habe den PHP Teil nur auf die Smarty-Syntax angepasst. Nach dem das gemacht wurde, habe ich im Login.php File und in der EBiblio_Hauptseite einige Sachen auf das Smarty angepasst.

Im Login.php File, habe ich folgende Sachen geändert:

```

//Das Passwort passt nicht zum Benutzer
//Wir stellen dem Template die Variable pwfalsch zur Verfügung
$smarty->assign('passwfalsch', true);

}else{
  //Es wurde kein Benutzer mit dem Namen $user gefunden usernotfound
  //Wir stellen dem Template die Variable pwfalsch zur Verfügung
  $smarty->assign('usernotfound', true);
}

//Wir sagen der Engine das sie das Template form.tpl anzeigen soll
$smarty->display('form.tpl');

```

Im EBiblio_Hauptseite.php File, habe ich folgende Sachen geändert:

```

//Stellen dem Template die Variable User zur Verfügung
$smarty->assign('user', $user);
//Wir sagen der Engine das sie das Template form.tpl anzeigen soll
$smarty->display('form_logout.tpl');

```


9. Quellen

[1] -, Database Schema, Abgerufen am 01.06.2015 von
<http://propelorm.org/documentation/reference/schema.html>

[2] -, The Build Time, Abgerufen am 01.06.2015 von
<http://propelorm.org/documentation/02-buildtime.html>

[3] -, Smarty Template Engine, Abgerufen am 16.06.2015 von
<http://www.smarty.net/>