Erstellung des Login- und Registrierungsformulars

Als erster Schritt wurden eine ganze Reihe von Layouts für das Login- und Registrierungsformular vorskizziert, um uns darauf zu einigen, welches Design und welche Benutzer-Führung unseren Vorstellungen am besten entsprechen. Es kamen dabei mehrere interessante Ideen in Frage: Es boten sich die Möglichkeiten, das Loginformular unabhängig vom Registrierungsformular als separate Webseiten zu realisieren, sowohl das Login- als auch das Registrierungsformular parallel auf einer einzigen Webseite anzuzeigen oder eine einzige Webseite dafür zu nutzen, je nach Wahl des Benutzers, entweder das Login- oder das Registrierungsformular anzuzeigen. Es wurden zu Testzwecken für jede dieser Möglichkeiten einfache Webseiten realisiert und zuletzt fiel die Wahl auf die scheinbar kompakteste, simpelste und zugleich benutzerfreundlichste Variante: Die, in der dem Benutzer per JavaScript, je nach Wahl, auf ein- und derselben Seite entweder die Login- oder die Registrierungsfunktionalität geboten wurde.

Die eben beschriebene Webseite wurde mittels Notepad++ erstellt, einem kleinen Texteditor vom Software-Entwickler Don Ho, ähnlich dem gewöhnlichen, mit Windows vorinstallierten Notepad, jedoch einen deutlich größerer Funktionalitätsumfang bietend: Drag 'n' Drop, Auto-Vervollständigung, Syntax-Hervorhebung entsprechend der Dateiendung der editierten Datei, Multi-Ansicht und Plug-Ins. Dieses Programm ist einer der kleinsten und gleichzeitig beliebtesten Editoren zum Verfassen von Code in diversen Programmiersprachen oder um Webseiten von Grund auf zu erstellen. Obwohl prinzipiell leicht in der Handhabung, bietet Notepad+

+ keine Webseiten-Templates zur leichten Überarbeitung und so erforderte fast jeder Schritt in der Umsetzung des Designs oder der Funktionalität eine eigene Internet-Recherche.

Es wurde eine Div-Box [Abb.1], die zwei weiter Div-Boxen [Abb.2] enthält, erstellt, eine für das Loginformular und eine für das

```
<body>
    <div class="last">
        <div class="form">
            <ul class="tab-group">
                <li class="tab active"><a href="#login">Anmeldung</a></li>
                <li class="tab"><a href="#signup">Registrierung</a></li>
            </ul>

            <div class="tab-content">

                <div id="login">
                </div>


                <div id="signup">
                </div>

            </div><!-- tab-content -->
        </div> <!-- /form -->

        <div class="img-container">
            <img src="gallery/favicon.svg" alt="logo">
        </div>
    </div>

    <script src='js/jquery.min.js'></script>
    <script  src="js/index.js"></script>

</body>
```

Registrierungsformular. Beide Div-Boxen erhielten dabei eigene IDs, damit sie über die angehängte CSS-Datei bearbeitet werden können.

[Abb.1]

Abbildung 2 zeigt die Anmeldungs-Div-Box ( ), die das Anmeldungs-Formular (<form [...]>) enthält. Das Anmeldungs-Formular

ist zu diesem Zeitpunkt noch nicht mit einer php-Datei verknüpft, welche zukünftig den Zugriff auf die Datenbank durchführen sollte. Es sind in Abbildung 2 auch die Definitionen der in den Formularen enthaltenen Eingabefelder und Buttons zu sehen, über der Benutzer seine Eingaben tätigen und bestätigen kann – je nach Wahl des Benutzers die des Anmeldungs- oder die des Registrierungs-Formulars.

```
<div class="tab-content">

    <div id="login">

        <form action="/" method="post">

            <div class="field-wrap">
                <label>Email<span class="req">*</span></label>
                <input type="email"required autocomplete="off"/>
            </div>

            <div class="field-wrap">
                <label>Passwort<span class="req">*</span></label>
                <input type="password"required autocomplete="off"/>
            </div>

            <p class="forgot">
                <a href="#">Passwort vergessen?</a>
            </p>

            <button class="button button-block"/>Anmelden</button>

        </form>
    </div>

    <div id="signup">
        <form action="/" method="post">

            <div class="top-row">
                <div class="field-wrap">
                    <label>Vorname<span class="req">*</span></label>
                    <input type="text" required autocomplete="off" />
                </div>

                <div class="field-wrap">
                    <label>Nachname<span class="req">*</span></label>
                    <input type="text"required autocomplete="off"/>
                </div>

            </div>

            <div class="field-wrap">
                <label>Email<span class="req">*</span></label>
                <input type="email"required autocomplete="off"/>
            </div>

            <div class="field-wrap">
                <label>Passwort<span class="req">*</span></label>
                <input type="password"required autocomplete="off"/>
            </div>

            <button type="submit" class="button button-block"/>Registrieren</button>

        </form>
    </div>

</div><!-- tab-content -->
```

Das CSS-File enthält Informationen, die die graphischen Eigenschaften der Elemente der Oberfläche bestimmen. Um die Box auf die gewünschte Art darstellen zu können wird eine Webkit-Erweiterung mit Präfix verwendet. Mit Hilfe der Definition „list-style: none;“ wird aus der Tabelle „.tab-group“ eine waagrechte Liste gemacht und mit Hilfe der Code-Blöcke in „.tab-group:after“ und „.tab-group li a“ wird diese Liste so eingestellt, dass entweder der Button „Anmeldung“ oder der Button „Registrierung“ farblich betont sind, je nach dem welches Formular der Benutzer gerade vor sich hat.

Der Code-Block in „.form“ bezieht sich auf die äußerste Div-Box und legt für diesen die gewünschte Farbe und die gewünschten Abstände fest.

Der form Tag ist ja eigentlich auch die Div-Box, in der sich alles befindet. die formtags anmeldung und registrierung sind fixer bestandteil der benutzeroberfläche. (damit man die div-boxen mit den form tags genauer sieht) Damit man nun diese Form genauer sieht, wurden ein paar Justierungen hinzugefügt. (.form)

[Abb.2]

[Abb.3]

Zur Anwendung kommen zwei JavaScript-Dateien, ein selbsterstellts JavaScript [Abb. 4] und ein angefügtes jquery.min JavaScript, um so die eventuelle Nutzung von jQuery-Codes zu ermöglichen.

```
$('.form').find('input, textarea').on('keyup blur focus', function (e) {

  var $this = $(this),
      label = $this.prev('label');

      if (e.type === 'keyup') {
            if ($this.val() === '') {
          label.removeClass('active highlight');
        } else {
          label.addClass('active highlight');
        }
    } else if (e.type === 'blur') {
        if( $this.val() === '' ) {
            label.removeClass('active highlight');
            } else {
            label.removeClass('highlight');
            }
    } else if (e.type === 'focus') {

      if( $this.val() === '' ) {
            label.removeClass('highlight');
            }
      else if( $this.val() !== '' ) {
            label.addClass('highlight');
            }
    }

});

$('.tab a').on('click', function (e) {

  e.preventDefault();

  $(this).parent().addClass('active');
  $(this).parent().siblings().removeClass('active');

  target = $(this).attr('href');

  $('.tab-content > div').not(target).hide();

  $(target).fadeIn(600);
```

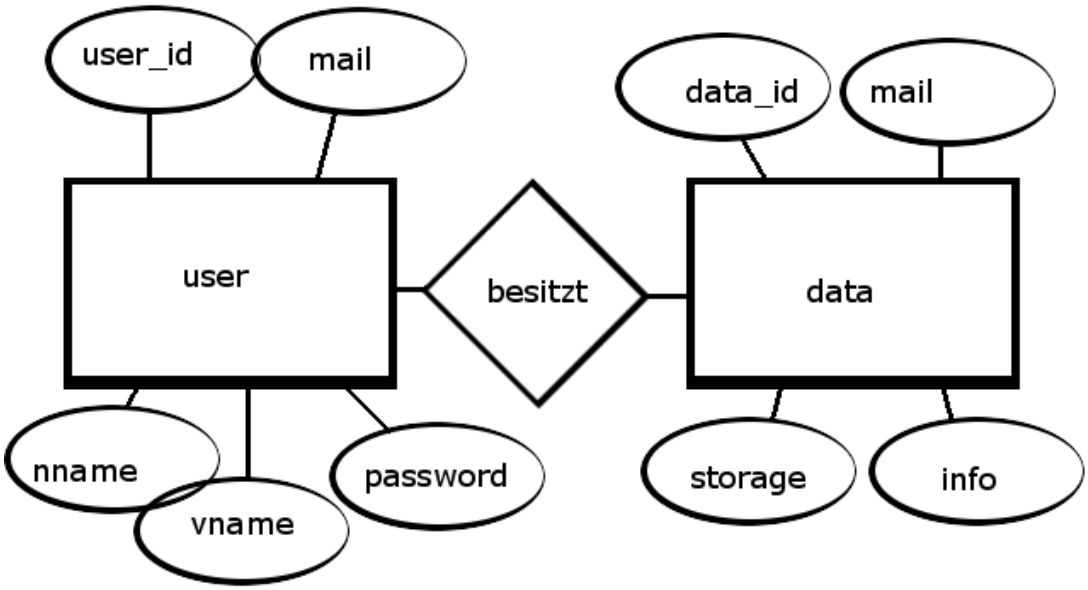
[Abb.4]

Der obere Abschnitt des Codes definiert die Eingabefelder für den Benutzer. In diesen Eingabefeldern steht jeweils bereits ein Wort, das dem Benutzer als Hinweis dient, welche Art von Eingabe von ihm erwünscht wird. Sobald der Benutzer in ein Eingabefeld klickt, bewegt sich dieses Hinweis-Wort unter das Eingabefeld und verkleinert sich etwas. Verlässt der Benutzer das Eingabefeld, ohne etwas eingegeben zu haben, so bewegt sich das Hinweis-Wort wieder zurück an seinen ursprünglichen Platz und nimmt seine ursprüngliche Größe an; außerdem wird der Rahmen des Eingabefeldes rot gefärbt, um dem Benutzer zu

signalisieren, dass noch eine Eingabe von ihm erwartet wird. Der untere Abschnitt des Codes führt den Wechsel vom Anmeldungs- zum Resgistrierungs-Formular und umgekehrt durch: Befiehlt der Benutzer den Wechsel auf das andere Formular wird das zuvor angezeigte Formular zerstört und das andere Formular an seiner Stelle erzeugt.

Erstellung einer Datenbank

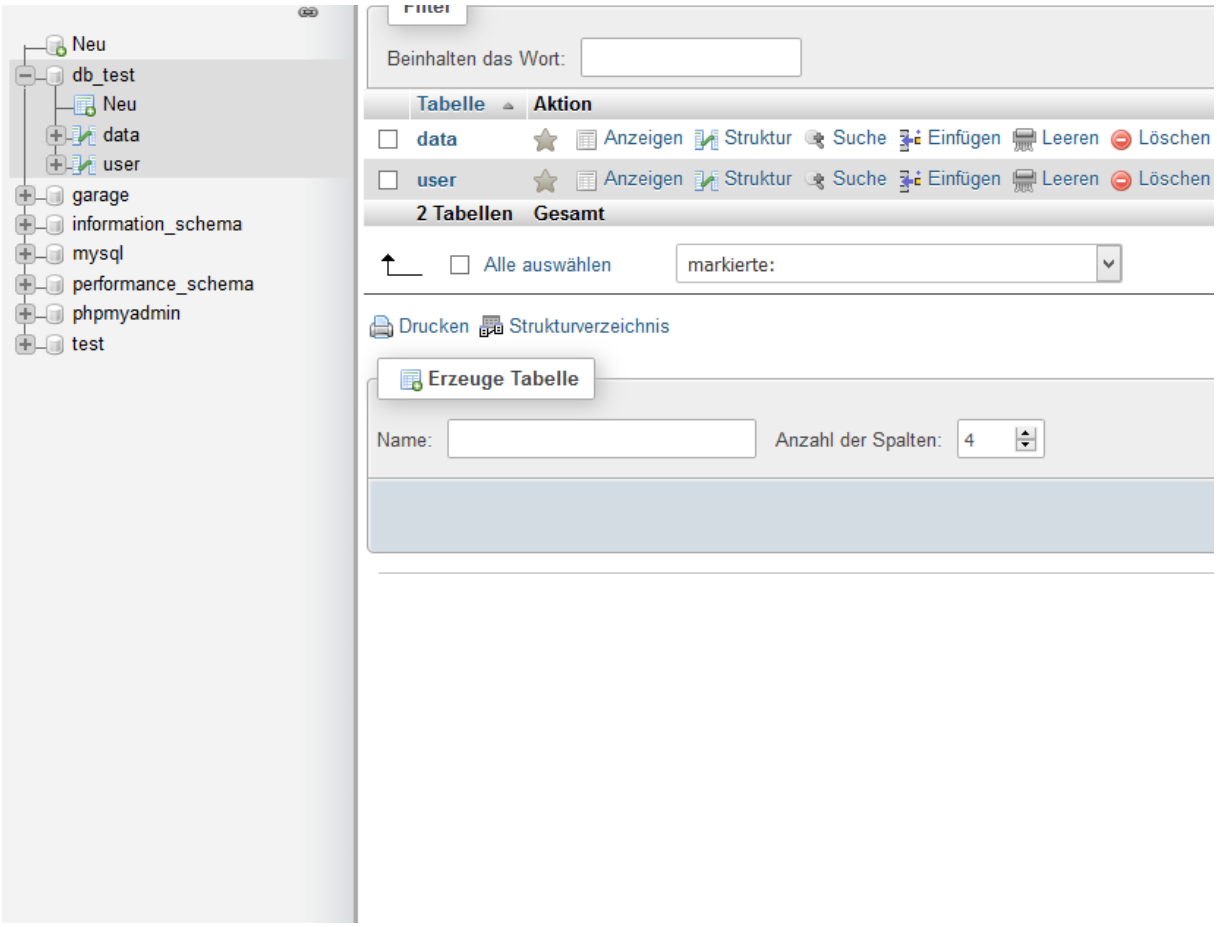
Nach erfolgreicher Erstellung der Webseite zur Registrierung und zum Login war der nächste Schritt die Erstellung einer Datenbank, in der die Anmeldungsdaten der Benutzer gespeichert und abgefragt werden können. Um einen Überblick zu gewinnen wurde zunächst ein Entity-Relationship-Modell skizziert. [Abb. 5] Dann wurde überlegt, ob eine SQL-Datenbank genutzt werden sollte, oder eine Datenbank ohne SQL und die Wahl fiel vorerst auf eine SQL-Datenbank. Die Idee war, diese Datenbank so lange zu nutzen, bis eine bessere Lösung zur Speicherung und Abfrage der Benutzerdaten gefunden würde.



[Abb.5]

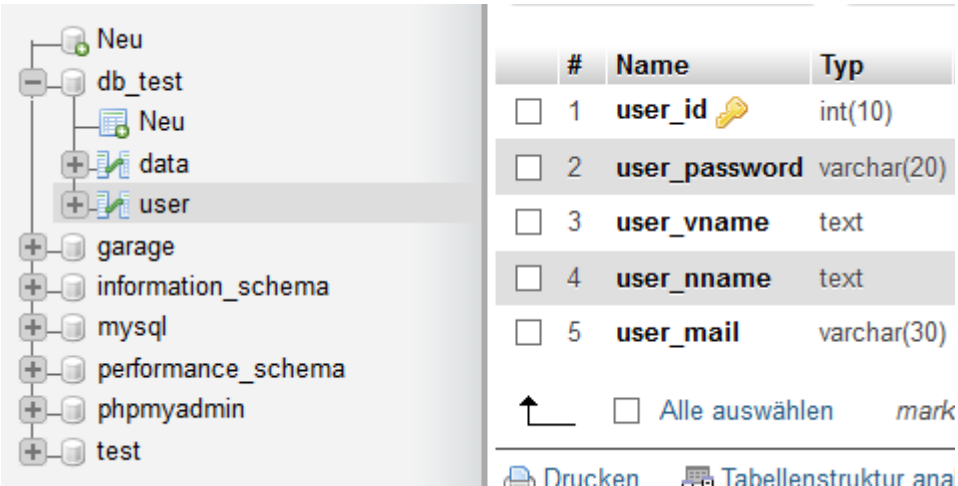
Da wir im Zuge des Unterrichts bereits Erfahrungen „XAMPP" gesammelt hatten fiel zur Realisierung der MySQL-Datenbank unsere Wahl auf dieses Software-Paket, welches den Webserver „Apache", die Datenbanklösung „MariaDB", und die Skriptsprachen „PHP" und „Perl" enthält. Mit Hilfe von „XAMPP" wurde zunächst ein Datenbank-Server lokal in Betrieb genommen, wozu sich dieses Software-Paket besonders gut eignet. Zur Nutzung von „XAMPP" in einem Produktivsystem empfiehlt sich eine umfassende Änderung der „XAMPP"-Konfiguration, um einen einigermaßen sicheren Betrieb zu ermöglichen.

Als Benutzer-Interface bietet „XAMPP" das „Control Panel". Über dieses wurden zuerst „Apache" und anschließend „MySQL" gestartet. Mittels MySQL wurde eine Test-Datenbank „db_test" erstellt, in der, dem Entity-Relationship-Modell [Abb. 6] entsprechend, zwei Tabellen erzeugt wurden: die Tabelle „user" [Abb. 7], in der die die Benutzer betreffenden Informationen gespeichert werden und die Tabelle „data" [Abb. 8] in der Dateien gespeichert werden.



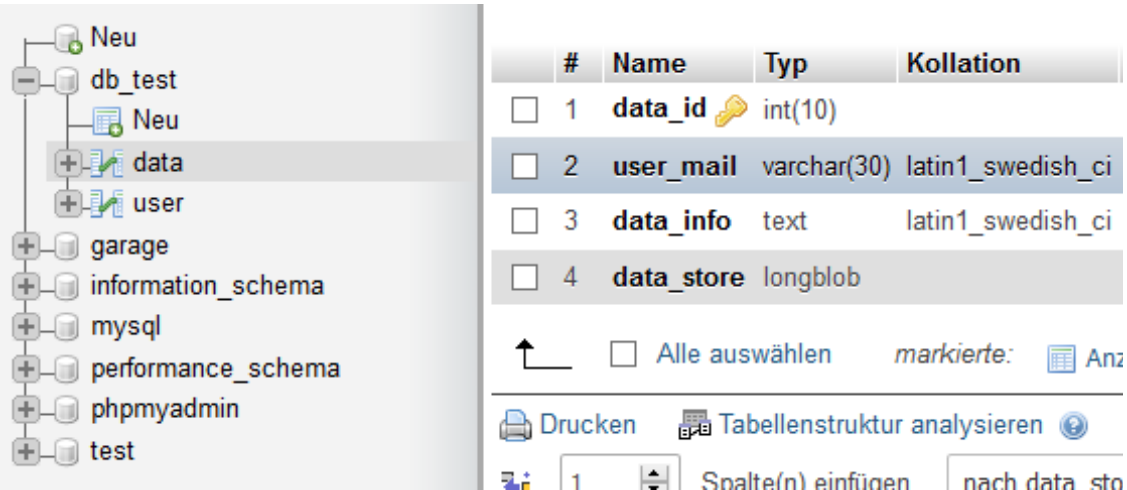
[Abb.6]

Die Tabelle „user" wurde gemäß dem Entity-Relationship-Modell erstellt:



[Abb.7]

Das Feld „user_id“ in der „user“ Tabelle dient der eindeutigen Identifikation jedes Benutzers. Die Option „auto-increment“ bewirkt hier, dass jeder neu registrierte Benutzer eine „user_id“ bekommt, die um 1 höher als der zuletzt vergebene „user_id“-Wert ist. Um einen problemlosen Ablauf des Loginvorganges zu gewährleisten sollte für jeden registrierten Benutzer auch tatsächlich nur ein Datensatz in der Datenbank existieren – redundante Datensätze sollen hier unbedingt vermieden werden. In den Feldern „user_mail“ und „user_password“ werden die Strings gespeichert, die der Benutzer bei der Anmeldung als E-Mail-Adresse und Passwort angegeben hat und mit Hilfe derer er sich am Webinterface als registrierter Benutzer anmelden kann. Die Felder „user_vname“ und „user_nname“ dienen der Speicherung von Informationen für ein Benutzer-Profil, das eventuell noch realisiert wird.



[Abb.8]

Auch die Tabelle „data" wurde gemäß dem Entity-Relationship-Modell erstellt: In diesem Fall dient das Feld „data_id" als Primärschlüssel während das Feld „user_mail" der Verknüpfung mit der Tabelle „user" dient. Das Feld „data_info" enthält eine Textdatei mit zusammenfassenden Informationen über die im Feld „data_store" gespeicherten Dateien. „data_store" ist als „longblob" definiert, einem Datentyp, der es ermöglicht, in einen Datenstrom umgewandelte Dateien abzuspeichern. In umgekehrter Richtung wandelt das außenstehende Empfangsprogramm den Datenstrom wieder in seine Ursprungsdateien um. Neben dem „longblob" existieren auch noch die Datentypen „blob" und „tinyblob", welche die gleiche Funktion erfüllen, jedoch weniger Speicherplatz bieten. Das Speichern der Dateien hätte noch auf andere Wege realisiert werden können, doch die eben beschriebene Methode erschien als ausreichend.

Verknüpfung von Formular mit der Testdatenbank

Um die Verbindung zwischen Webseite und Datenbank testen zu können wurden in die Tabelle „user" zunächst manuell einige Testwerte eingetragen.

Logo und Icons designen

Erstellen einer Webinterface

Erstellen der Server-App