



Erstellen des Login- und Registry- Formular

Es wurden zunächst viel Skizzen vorgefertigt, um ungefähr festzulegen, welches Layout, welche Struktur und welche Methoden unserer Vorstellung am meisten entsprechen würden. Dabei kamen mehrere interessante Ideen zur Erzeugung der Webseite und des Logins- sowie Registrierungsformular in Frage. Es gab die Möglichkeit, das Anmeldeformular und das Registrierungsformular in zwei separate Webseiten zu halten, dann ergab sich die Möglichkeit, alles in einer Seite verfügbar zu machen und schlussendlich bietet sich auch die Alternative an, die Anmeldung und die Registrierung in einem Fenster zu erzeugen, wobei diese sich immer nur abwechselnd zeigen würden, je nachdem, welches Formular man gerade bräuchte. Ich erzeugte, testete und probierte zunächst die einfachsten Methoden, so dass wir letzten Endes doch noch fündig geworden sind. Da ich fand, es sollte benutzerfreundlich, simple und kompakt gestaltet sein, entschieden wir uns für die dritte Variante, nämlich die, in der Anmeldung und Registrierung mittels JavaScript schön zu einem dynamisch je nach Wunsch reagierenden Fenster verpackt wurde.

Die Software, welche für das Erstellen der Anmeldung und Registrierung zum Einsatz kam, nennt sich Notepad++, ein kleiner Texteditor geschrieben vom Software-Entwickler Don Ho, der dem vorinstallierten und gewöhnlichem Notepad, was Zusatzoptionen angeht, um Längen voraus ist. Mit den Möglichkeiten wie Drag n Drop, Auto-Vervollständigung, Syntax-Hervorhebung (unter Verwendung der



jeweiligen Programmiersprache mit der dazugehörigen Dateieindung), einer Multi-Ansicht und der Verwendung von Plug-ins wurde dieses Programm leicht zur einer der beliebtesten und einfachsten Tools, mit denen man zwar mehrere Programmiersprachen-orientierte Applikationen, aber vor allem Web-basierte Seiten oder Projekte von null aus erstellen kann. Notepad ++ ist zwar leicht zu handhaben, dennoch war es nicht wirklich gut zu mir, da, anders als bei vorgefertigten Webseiten-Templates, die man einfach überarbeiten kann, fast jeder Schritt, betreffend des Designs, der Funktionsweise und der Umsetzung eine Internet-Recherche erforderte.

Erstellt wurde eine Div-Box [Abb.1], in der zwei weitere Div-Boxen [Abb.2] integriert wurden, wobei eines für das Anmelde-Formular während das andere für das Registrierungsformular verantwortlich ist. Identifiziert

```
<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">
      
    </div>

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

wurde jede Div-Box mit separaten IDs, um diese im angehängte CSS-File bearbeiten zu können.

[Abb.1]



In Abbild 2 sieht man, wie eine form per Div-Box erzeugt wurde, jedoch noch nicht mit der php-Datei verankert ist, die später mit der Datenbank verbunden werden soll. Desweiteren werde die üblichen Elemente wie Inputs & Buttons erstellt, welche bei der virtuellen Darstellung als eine Anmeldungs- oder als eine Registrierungs-Box mit Informationen zum Eingeben angezeigt wird.

```
<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 -->
```

[Abb.2]



```

*{
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}

html {
  overflow-y: scroll;
}

body {
  background: #D5EACD;
  font-family: 'Titillium Web', sans-serif;
}

a {
  text-decoration: none;
  color: #89c400;
  -webkit-transition: .5s ease;
  transition: .5s ease;
}

a:hover {
  color: #179b77;
}

.last {
  position: relative;
  background: #305400;
  max-width: 600px;
  height: 800px;
  margin: auto;
  margin-top: 100px;
}

.form {
  background: #305400;
  padding: 40px;
}

.tab-group {
  list-style: none;
  padding: 0;
  margin: 0 0 40px 0;
}

.tab-group:after {
  content: "";
  display: table;
  clear: both;
}

.tab-group li a {
  display: block;
  text-decoration: none;
  padding: 15px;
  background: rgba(160, 179, 176, 0.25);
  color: #a0b3b0;
  font-size: 20px;
  float: left;
  width: 50%;
  text-align: center;
  cursor: pointer;
  -webkit-transition: .5s ease;
  transition: .5s ease;
}

```

[Abb.3]

Im CSS-File [Abb.3] stehen die Funktionen für die graphische Oberfläche, die man auf einer Webseite sieht. Es wurde einer Webkit-Erweiterung mit Präfix verwendet, um eine bestimmte Box-Darstellung zu erzielen. Die Liste "tab-group" wurde so eingestellt, dass sie zu einer waagrechten Liste, anstelle der standardisierten senkrechten Liste wird, die Punkte und Unterpunkte, werden dabei auch versteckt und mit :after wurde die Liste so eingerichtet, dass man mittels einer bestimmter Farbeveränderung schnell merkt, dass die Anzeige Login oder Registrierung leuchtet, je nachdem in welchen man sich gerade befindet.

Der form Tag ist ja eigentlich auch die Div-Box, in der sich alles befindet. Damit man nun diese Form genauer sieht, wurden ein paar Justierungen hinzugefügt.



Verwendet wurden dabei zwei JavaScript-Formate, ein selbsterstelltes JavaScript [Abb.4] und ein angefügtes jquery.min JavaScript, um so das Verwenden einiger jQuery-Codes zu ermöglichen, falls es benötigt werden

```
$('.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);  
});
```

sollte.

Das erste Code-Stück umfasst die Input-Felder, was dann von Usern verwendet werden soll, in denen sich bereits ein Wort befindet, welches kurzübergreifend andeutet, was genau eingetragen gehört. Beim anklicken soll der Text im Input kleiner unter dem

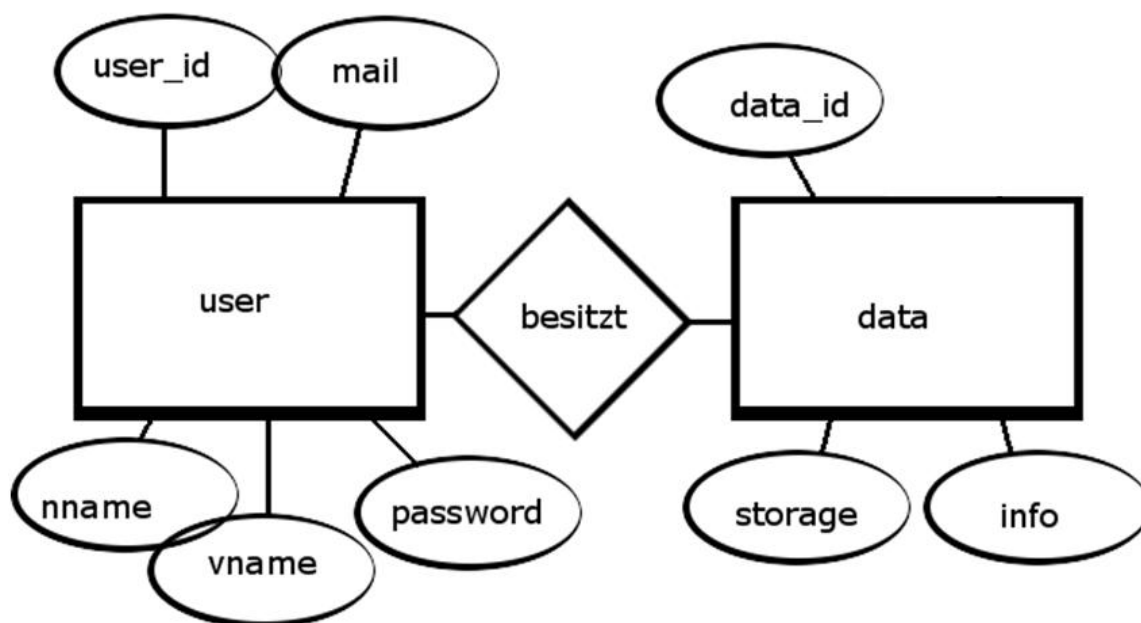
[Abb.4]

Input stehen, damit die Benutzer immer noch lesen können, worüber es bei den jeweilige Input-Felder geht. Beim Verlassen dieser Input-Felder mit leerem Kontext soll der ursprüngliche Text wieder hinein, dabei wird der Feld-Rand rot markiert, um zu signalisieren das ein Wert hineingehört. Beim zweiten Code-Stück handelt es sich dann um den aktiven Wechsel zwischen dem Anmeldungs-Format und dem Registrierungs-Format, bei dem eine Klasse immer erzeugt wird, wenn man auf den entsprechenden Label der tab-group klickt.



Erstellen einer Datenbank

Da das Webinterface für die Anmeldung und die Registrierung erfolgreich war, wurde anschließend mit der Überlegung einer Prototyp-Datenbank begonnen, denn ohne eine Datenbank wäre ein Anmeldungs- und Registrierungsformular sinnlos. Zunächst wurde wieder eine Skizze für die Datenbank, das sogenannte Entity-Relationship-Modell erstellt. [Abb.5] Dies dient zu einem orientierungsvollerem Überblick über die Datenbank, seiner Tabellen und dessen Funktionsrolle. Dannach wurde überlegt, welche Datenbank man benutzen sollte, nur war diesmal die Auswahl nicht besonders groß. Es gab die Variante mit SQL und es gab die Alternative ohne SQL. Wir entschieden uns vorübergehend für eine SQL-Datenbank. Diese Datenbank soll dann so lang benutzt werden, bis wir eine bessere oder einfachere Lösung unserer Datenverarbeitung finden würden.



[Abb.5]

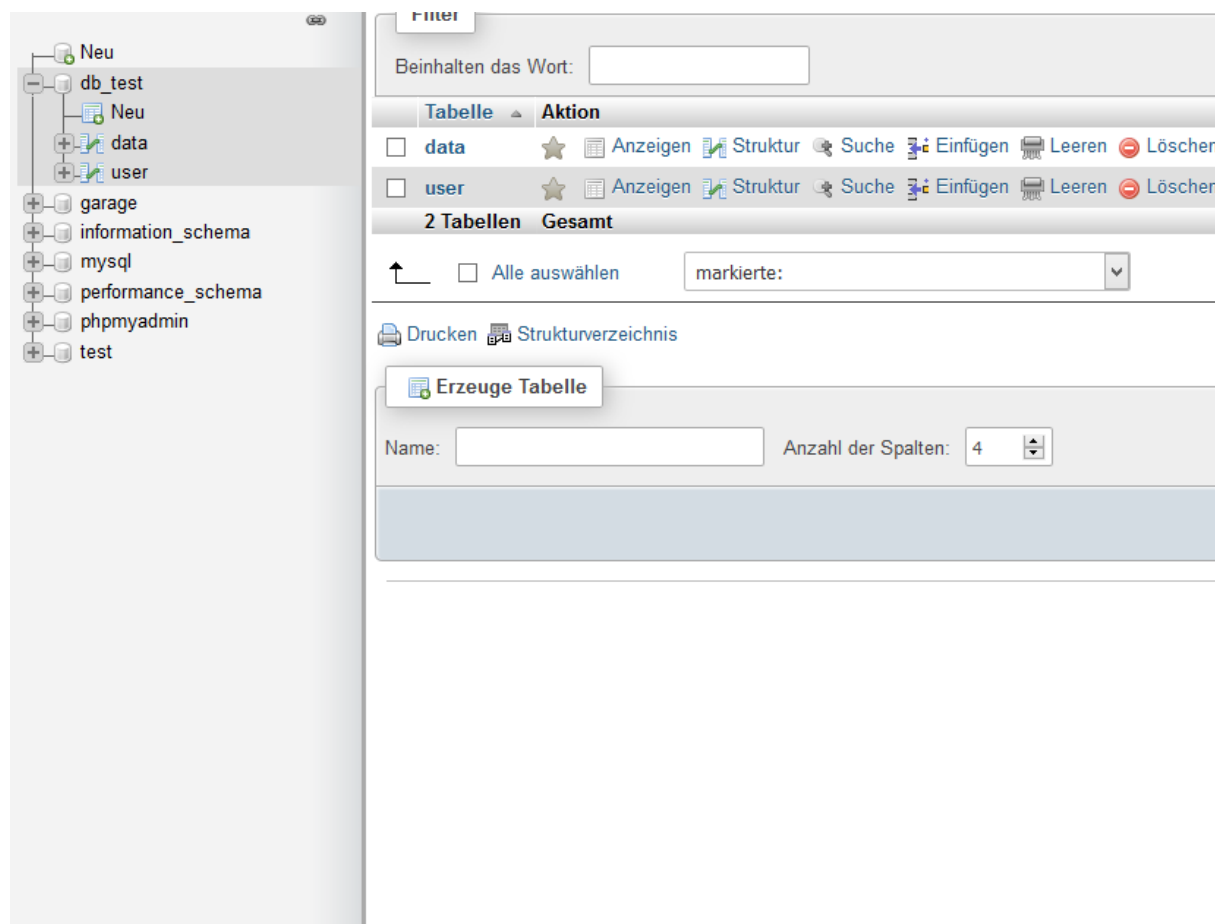


Als Anwendungsprogramm hierfür konnte nur das `mySQL` von XAMPP in Frage kommen, da dies in unserer Schulausbildung schon des Öfteren benutzt wurde. Genauer ist `MySQL` eine Anwendung von dem Software-Paket XAMPP, welches von dem gemeinnützigen Projekt Apache Friends entwickelt und bereitgestellt wurde. Die Software XAMPP ist wiederum eine kostenlose und einfache Apache-Distribution, welches `MariaDB`, `PHP` und `Perl` mit einschließt. Jedoch sollte es noch erwähnt werden, dass für XAMPP in einem überaus schnellen Tempo immer wieder neue Versionen auftreten, die man zwar nicht benutzen muss, aber wenn man diese braucht, man die alte Version dann aufgeben muss. `MySQL` soll als Simulation eines Datenbank-Servers auf den lokalen Rechner dienen, kann jedoch auch auf der globalen Ebene erweitert werden, wobei man beachten sollte, dass die Sicherheit der übertragenen Daten hierbei nicht gewährleistet wird. XAMPP hat das Ziel, die Datenbankverwaltung und das Datenbankverständnis an Neuankömmlinge einfach und schnell zu übermitteln. Außerdem ist XAMPP eher an Entwickler gerichtet, die möglichst schnell ein kompaktes Testsystem aufsetzen wollen. Sie ist nicht zum Gebrauch als Produktivsystem (z.B. öffentlicher Webserver) gedacht und es ist daher weniger wünschenswert es als ein solches anzusehen.

Über dem Control Panel, ein Benutzer-Interface zum Auswählen einer oder mehrerer Applikationen, wurde zuerst Apache und anschließend `MySQL` gestartet. In `MySQL` wurde die Datenbank `db_test` erstellt, da es sich hierbei ja auch um eine Test-Datenbank handelt. Hinzugefügt wurden zwei Tabellen, die jeweils dem ER-Modell entsprechen sollen [Abb.6]. Eine Tabelle trägt den Namen `user` [Abb.7], wo alle Benutzer ihre



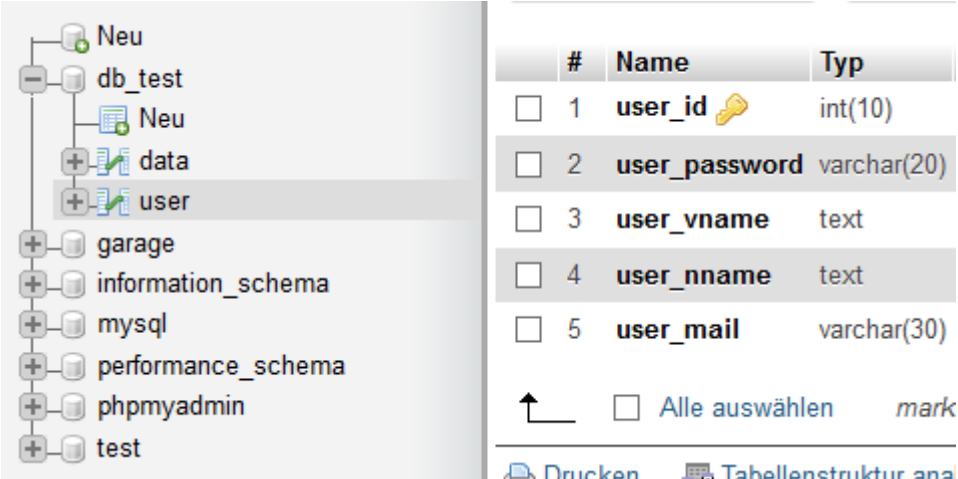
eingeegebenen Konto-Information aufbewahren und das andere wird als data [Abb.8] bezeichnet, da dort auch die Dateien gelagert werden sollen.




[Abb.6]



Gemäß dem Entity-Relationship-Modell wurde die notwendigen Einträge hinzugefügt:



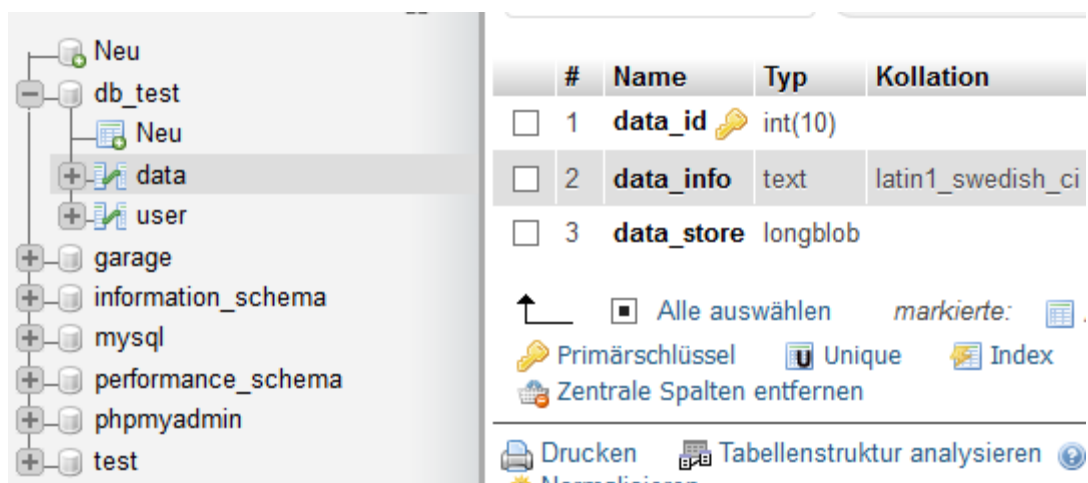
#	Name	Typ
<input type="checkbox"/> 1	user_id 	int(10)
<input type="checkbox"/> 2	user_password	varchar(20)
<input type="checkbox"/> 3	user_vname	text
<input type="checkbox"/> 4	user_nname	text
<input type="checkbox"/> 5	user_mail	varchar(30)

☐ Alle auswählen *mark*

[Drucken](#) [Tabellenstruktur analysieren](#)

[Abb.7]

Bei der user Tabelle gibt es eine user_id zur eindeutigen Identifizierung von jedem Benutzer, der sich neu registrieren oder anmelden will. Deshalb wird dem auch ein Primärschlüssel hinzugefügt, der automatisch aufzählen soll, falls ein neuer Benutzer sich registriert. Die Verbindung sollte auch redundantfrei sein, da es sonst zu Datenbankverbindungsfehler kommen kann. Hinzugefügt wird dann sowohl ein user_password als auch eine user_mail, welches wieder in der Registrierung vorhanden ist und dann zum Anmelden immer benötigt wird. Zuletzt noch ein user_vname und user n_name für das eigene Profil. Ob es ein Profil geben wird, lässt sich streiten, da das mehr oder weniger zum Schluss zusätzlich verarbeitet werden kann, falls die Notwendigkeit und die Zeit dafür besteht. Im Falle eines Profils werden diese zwei Datensätze sehr wichtig sein.



[Abb.8]

Dasselbe gilt auch für die Tabelle data und seine Einträge. data_id wird wieder als Schlüssel verwendet, user_mail wird zur Verknüpfung mit user benötigt. In der data_info wird eine Zusammenfassung über die gelagerten Daten in Form einer Textdatei erstellt. Schlussendlich kommt dann ein data_store definiert als longblob. Ein blob ist ein variablentyp, welcher Dateien in einem Datenstrom umwandelt und diese einlagert. Um dies rückgängig zu machen, muss das außenstehende Empfangsprogramm den überlieferten Datenstrom wieder in seine Ursprungsdatei umwandeln. Dabei unterscheidet man zwischen tinyblob, einem normalen blob und einem longblob, wobei sie der Reihe nach die Größe des Platzspeichers ausmachen. Neben anderen Alternativen, die die Software zu bieten hätte, reicht der Typ blob für das lagern einiger Files aus.



Verknüpfung von Formular mit der Testdatenbank

Vorerst wurden manuell Werte eingetragen, damit zumindest die Kontodaten getestet werden können. Es wurde für je ein Formular eine php-Datei erstellt, eine mit dem Namen login.php [Abb.9] und die andere register.php [Abb.10], welche die Verbindung und Datenübertragung zur Test-Datenbank erstellen soll.

Die Datenbank-Verbindung wurde in einer zusätzlichen php-Datei erzeugt, die zur HTML-Seite verlinkt wurde. Wichtig ist dabei zu beachten, dass das php-Tag nicht geschlossen wird. Es wurden PHP-Variablen erstellt, in denen die Werte der Datenbank eingetragen wurden. Die Verbindung zum Host und zur Datenbank wurden dann mit new mysqli geöffnet.

```
<?php
$host = 'localhost';
$user = 'root';
$pass = '';
$db = 'db_test';
$mysqli = new mysqli($host,$user,$pass,$db) or die($mysqli->error);
```

[Abb.9]



Im login.php [Abb.10] wurden zunächst in die erstellten Variablen die Werte der Input-Boxen aus der HTML-Seite eingefügt. Beim Einfügen des user_mail in die \$user_mail Variable und beim Vergleichen dessen Werte mit der in der Datenbank, wurde auch escape_string verwendet. Dies muss nicht dazugeschrieben werden, bietet aber zusätzlich Sicherheit und Schutz gegen mögliche SQL-Injections. SQL-Injections sind Angriffe auf eine Datenbank durch Einfügen von SQL-Codestücken in die Input-Boxen. Dadurch könnte beispielsweise eine Datenbank komplett gelöscht werden oder der Zugang zu einer Seite könnte ohne Passwort erfolgen. Im \$result wird nach den Informationen für die gewünschten user_mail gesucht. Sollte diese nicht gefunden werden, wird eine Nachricht zurückgeliefert, dass diese Mail noch nicht in der Datenbank existiert. Falls diese jedoch existiert, wird mit fetch_assoc() der ganze Array des gesuchten Wortes hergeholt. Anschließend wird das Passwort überprüft, sollte diese Barriere auch überwunden sein, werden die restlichen Werte; Vorname, Nachname und Mail vom Datenarray übernommen.

```
<?php

$user_mail = $mysqli->escape_string($_POST['user_mail']);
$result = $mysqli->query("SELECT * FROM users WHERE user_mail='$user_mail'");

if ( $result->num_rows == 0 )
{
    echo "User with that user_mail doesn't exist!";
}
else
{
    $user = $result->fetch_assoc();

    if ( $_POST['user_password'], $user['user_password'] )
    {
        $_SESSION['user_mail'] = $user['user_mail'];
        $_SESSION['user_vname'] = $user['user_vname'];
        $_SESSION['user_nname'] = $user['user_nname'];
    }
}
```

[Abb.10]



[Abb.11] zeigt die Funktionsweise für die Registrierung und somit auch der Eintragung der Daten in die Test-Datenbank. Hier werden in den erstellten php-Variablen die Werte aus den Eingabefeldern der HTML übernommen. Es wurde wieder `escape_string` verwendet, um SQL-Injections zu vermeiden. Dann wird mit `$result` analysiert, ob die `user_mail` bereits vorhanden ist. Das wissen wir, falls die Nummer der Reihe größer als 0 zurückliefert, was wiederum bedeutet, dass sie zweimal enthalten ist. Sollte dies zutreffen, wird eine Fehler-Nachricht entsendet. Andernfalls wird von der `$sql`-Variable aus in die Reihe `users` die jeweiligen Werte eingetragen. Sollte dabei ein Fehler bezüglich des falschen Formats oder ein anderer SQL-Fehler auftreten, wird wieder eine Fehler-Nachricht entsendet.

```
<?php
$user_vname = $mysqli->escape_string($_POST['user_vname']);
$user_nname = $mysqli->escape_string($_POST['user_nname']);
$user_mail = $mysqli->escape_string($_POST['user_mail']);
$user_password = $mysqli->escape_string($_POST['user_password']);

$result = $mysqli->query("SELECT * FROM users WHERE user_mail='$user_mail'") or die($mysqli->error());

if ( $result->num_rows > 0 )
{
    echo 'User with this user_mail already exists!';
}

else
{
    $sql = "INSERT INTO users (user_vname, user_nname, user_mail, user_password) "
        . "VALUES ('$user_vname', '$user_nname', '$user_mail', '$user_password')";

    else
    {
        echo 'Registration failed!';
    }
}
}
```

[Abb.11]



Zum Schluss wurde das dann mit dem Registrieren [Abb.12] und Anmelden [Abb.13] überprüft [Abb.14].

The registration form (Abb.12) is a dark-themed interface. At the top, there are two buttons: 'Sign Up' (green) and 'Log In' (grey). Below these is the heading 'Sign Up for Free'. The form contains several input fields: 'First Name' with the value 'Alex', 'Last Name' with the value 'Rus', 'Email Address' with the value 'test@mail.com', and a password field with 10 dots. Below the password field is the label 'Set A Password'. At the bottom is a large green button labeled 'REGISTER'.

[Abb.12]

The login form (Abb.13) is a dark-themed interface. At the top, there are two buttons: 'Sign Up' (grey) and 'Log In' (green). Below these is the heading 'Welcome Back!'. The form contains two input fields: 'Email Address' with the value 'test@mail.com' and a password field with 10 dots. Below the password field is the label 'Password'. To the right of the password field is a link 'Forgot Password?'. At the bottom is a large green button labeled 'LOG IN'.

[Abb.13]

The user profile page (Abb.14) is a dark-themed interface. At the top is the heading 'Welcome'. Below it is the user's name 'Alex Rus' in green, followed by the email address 'test@mail.com'. At the bottom is a large green button labeled 'LOG OUT'.

[Abb.14]



Logo und Icons designen

Für die Erstellung des Logos war sehr viel Überlegung und Kreativität gefragt. Zum einen sollte es zeigen, dass es sich hierbei um ein Backup handeln soll, zum anderen sollte der Anblick des Symbols auch die Idee einer Cloud vermitteln. Somit kam eine Wolke mit mehreren Verbindungen zustande[Abb.15].



[Abb.15]

Es fehlte jedoch die Farbeneinheitlichkeit und das Design stellte sich eher als altmodisch als modern dar, weshalb die Inspiration in Programmen mit ähnlichen Zwecken gesucht wurde. Als bestes Beispiel eines Backup-Clouds wurde dann die Dropbox genommen, da diese sehr benutzerfreundlich gestaltet wurde und dessen Emblem auch nicht unbemerkt blieb. Fokussiert wurde eher auf deren Design, das es simple



jedoch bemerkenswert aus der Masse herausstach. Von daher kam die Inspiration, das Design leicht zu verändern und es so moderner darstellen zu lassen. Grundsätzlich wurde am Hauptdesign nichts verändert, jedoch wurde die Farbe einheitlich gestaltet und die Darstellung anstelle verschiedener Farben mit weißen Konturen verziert. Es hat noch sehr viele Versuche und Kombinationen gebraucht, vor allem im Farbbereich, doch nach unzähligen Abstimmungen kamen wir schlussendlich zum gewünschten Ergebnis [Abb.16]. Lustigerweise kam beim Bild nicht nur eine Wolke mit Verbindungen als Darstellung heraus, das ja bekanntlich eine Cloud symbolisiert, sondern sie stellt auch einen Baum mit verankerten Wurzeln dar, welches für ein sicheres Backup stehen könnte.

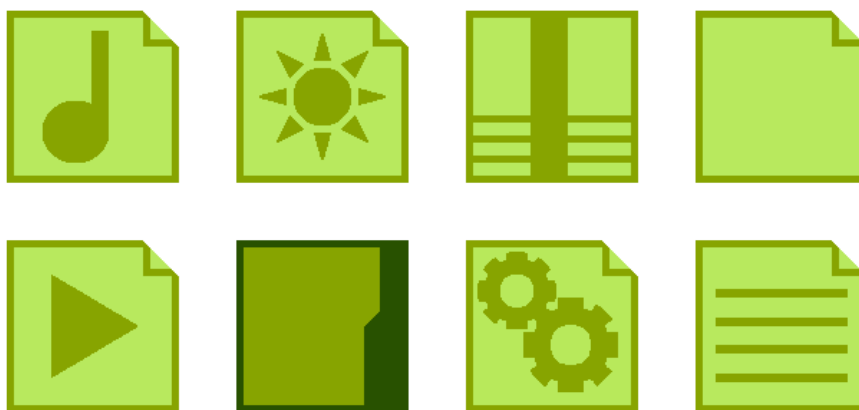


[Abb.16]

Verglichen mit dem Logo, welches das Projekt ja symbolisiert, waren die Icons eher was experimentelles und flüchtiges. Noch ungewiss, ob wir überhaupt Icons für die Dateien im Programm verwenden oder sie einfach im standardmäßigen Format lassen, kamen wir zum Entschluss, das es zumindest eigene Icons gäben müsste, wenn nicht für den Moment,

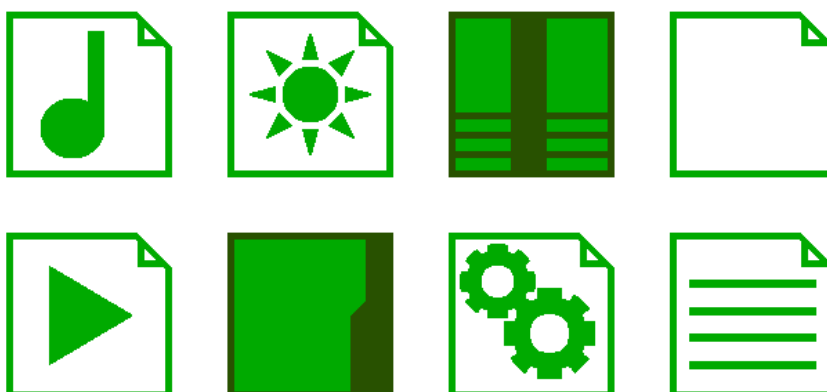


aber dann in naher Zukunft. Obwohl diese Arbeit leichter fiel als die Erstellung des Logos, war sie nicht gerade von kurzer Dauer. Trotz des immer verwendeten Grundrisses, dem File, mussten die Symbole für die jeweiligen Formate genauestens überlegt werden. Sie mussten original von null aus designt werden und durften nicht einfach schlampig dargestellt werden. Für den Anfang sind wir der Meinung, das zwischen Musik, Bilder, Videos, Text, Programm, Archiv und Ordner unterscheidet werden soll, alles andere wird als Leer-File dargestellt[Abb.17].



[Abb.17]

Eventuell wurden auch die Icons überarbeitet, da diese zum Design des Emblems dazugehören müssen [Abb.18].



[Abb.18]