Technische Projektdokumentation – Nokes

Usability der Applikation

Eine Clientseitige Fehlermeldung erscheint, da der User nicht eine Korrekte E-Mail-Adresse angegeben hat

Bei Angabe eines falschen alten Passworts wird dem User mit einer Fehlermeldung mitgeteilt, dass die Eingabe nicht korrekt war.


```
        if (strlen($error)) {
            echo "<div class=\"alert alert-
danger\" role=\"alert\">" . $error . "</div>";
        }
```

Bei Eingabe eines falschen Wertes wird mit diesem if Statement eine Fehlermeldung ausgegeben.

Client- und Serverseitige Validierung

```
<input type="text" name="userid" class="form-control" id="userid"
pattern="^[a-zA-Z]+$" minLength="3" maxLength="20" value="" required />
```

Die Benutzereingaben werden mit den Angaben pattern, minLength, maxLength und required im <input/> clientseitig validiert.

```
  $passwordPattern = "/^[ -~]+$/";
```

Hier bilden wir ein Pattern welches für das Passwort benötigt wird.

```
   if (
        !empty($user["password"]) &&
        strlen($user["password"]) >= 8 && strlen($user["password"]) <= 255 &&
        preg_match($passwordPattern, $user["password"])
    )
```

Serverseitig wird der mitgegebene Wert überprüft ob dieser die richtige Länge enthält. Danach wird der mitgegebene Wert mit dem Pattern geprüft.

```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| userid   | varchar(20)  | NO   | PRI | NULL    |       |
| name     | varchar(45)  | NO   |     | NULL    |       |
| email    | varchar(100) | NO   |     | NULL    |       |
| password | varchar(255) | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

Die Datentypen der Felder in der Datenbank sind sinnvoll und deren Zweck gemäss gewählt.

Bei dem Feld «password» werden gehashte Strings gespeichert. Da sich in der Zukunft der Hashingalgorithmus ändern könnte, haben wir zur Sicherheit eine maximale länge von 255 Zeichen genommen.

Sessionhandling

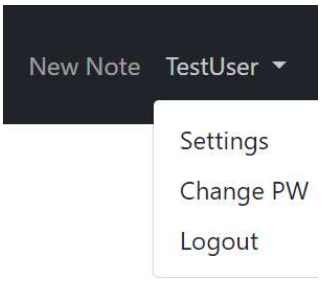
```
if (isset($_SESSION['loggedin']) and $_SESSION['loggedin'])
```

Die ganze Applikation ist erst vollumfänglich nutzbar wenn der User sich registriert und angemeldet hat.

```
session_start();
$_SESSION['loggedin'] = true;
$_SESSION['userid'] = $firstRow['userid'];
$_SESSION['name'] = $firstRow['name'];
$_SESSION['email'] = $firstRow['email'];
session_regenerate_id();
```

Nachdem sich der User eingeloggt hat, wird die Session gestartet.

Falls der Benutzer eingeloggt ist, kann er die Applikation nutzen und Notizen erstellen. Zudem erhält er die Möglichkeit sein Password, seinen Namen oder seine E-Mail-Adresse zu ändern. In Settings gibt es auch die Möglichkeit den eigenen Benutzer zu löschen. Wenn der Benutzer sich ausloggt, sind diese Möglichkeiten nicht mehr verfügbar, sondern nur noch die Möglicheiten zum Login oder zur Registrierung.



Sicherheit/Verschlüsselung

Für die Verbindung mit der Datenbank wird ein Datenbankbenutzer mit so wenigen Berechtigungen wie nötig verwendet. Er kann einzig und allein CRUD Operations auf der Datenbank «nokes» durchführen.

```
CREATE USER 'NokesUser' @'%' IDENTIFIED BY 'nokesUserP4ssw0rd';
GRANT SELECT,
    INSERT,
    UPDATE,
    DELETE ON nokes.* TO 'NokesUser' @'%';
```

```
$host_sql = 'localhost'; // host
$username_sql = 'NokesUser'; // username
$password_sql = 'nokesUserP4ssw0rd'; // password
$database_sql = 'nokes'; // database

// mit Datenbank verbinden
$mysqli = new mysqli($host_sql, $username_sql, $password_sql, $database_sql);
```

Passwörter der Benutzer der Applikation werden mittels einer Hashfunktion verschlüsselt und so auf der Datenbank abgelegt.

```
$hashedPW = password_hash($password, PASSWORD_BCRYPT);
```

```
+--------------------------------------------------------------+
| password                                                     |
+--------------------------------------------------------------+
| $2y$10$6Tl7dwLaIKoTfRu6lKvlNuZVcf2utEA3dRAwyTUyt9Z3NBYoCAEwO |
| $2y$10$YdCBZbjzu4oJMQNB.Bvlf.jxSOqeqmZCHAJNL.23bcSUQhe6lvE56 |
+--------------------------------------------------------------+
```

SQL-Injection wird verhindert, indem jede Query zuerst «vorbereit» an den Datenbank Server geschickt wird. Dadurch kann die Query nicht mit weiteren Statements erweitert werden.

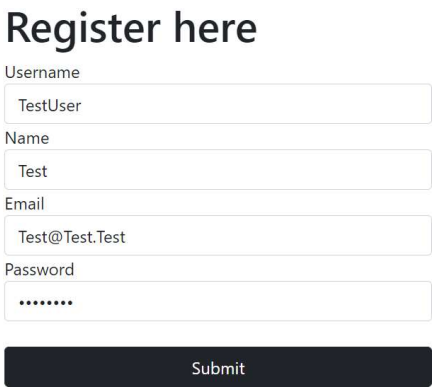
```
// INPUT Query erstellen

        $query = "INSERT INTO user(userid, name, email, password) VALUES (?,?,
?,?)";
        // Query vorbereiten mit prepare();
        $stmt = $mysqli->prepare($query);
        if ($stmt === false) {
            $errorString .= 'prepare() failed ' . $mysqli->error . '<br />';
        }
        // Parameter an Query binden mit bind_param();
        $hashedPW = password_hash($password, PASSWORD_BCRYPT);
        if (!$stmt->bind_param('ssss', $userid, $name, $email, $hashedPW)) {
            $errorString .= 'bind_param() failed ' . $mysqli-
>error . '<br />';
        }
        // query ausführen mit execute();
        if (!$stmt->execute()) {
            $errorString .= 'execute() failed ' . $mysqli->error . '<br />';
        }
        // Verbindung schliessen
        $stmt->close();
```

Damit über die Input Fields keine Scripts im Browser ausgeführt werden können, werden vor dem Speichern alle html entities ersetzt.

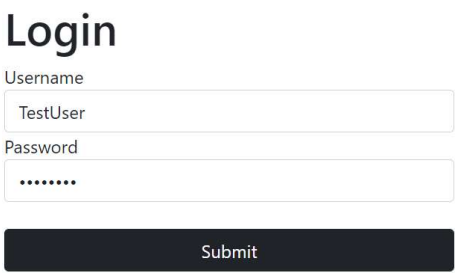
```
$name = htmlentities($trimmedName);
```

Registrierung und Login

Bei der Registrierung muss der User seinen Username, seinen Namen, seine E-Mail-Adresse und ein Passwort angeben.

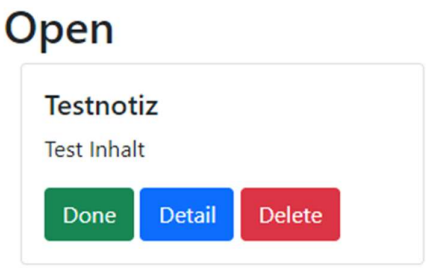


Beim Login benötigt der User nur noch seinen Username (userid) und sein Passwort um sich bei der Applikation anzumelden. Nach der Anmeldung wird die Session gestartet.



Notizverwaltung

Eingeloggte Benutzer können private Notizen erstellen. Diese besitzen einen Titel und einen Inhalt. Eine Notiz kann nach Vollendung auf «Done» gesetzt werden, bearbeitet oder gelöscht werden.



Validierung des HTML Codes

Document checking completed. No errors or warnings to show.

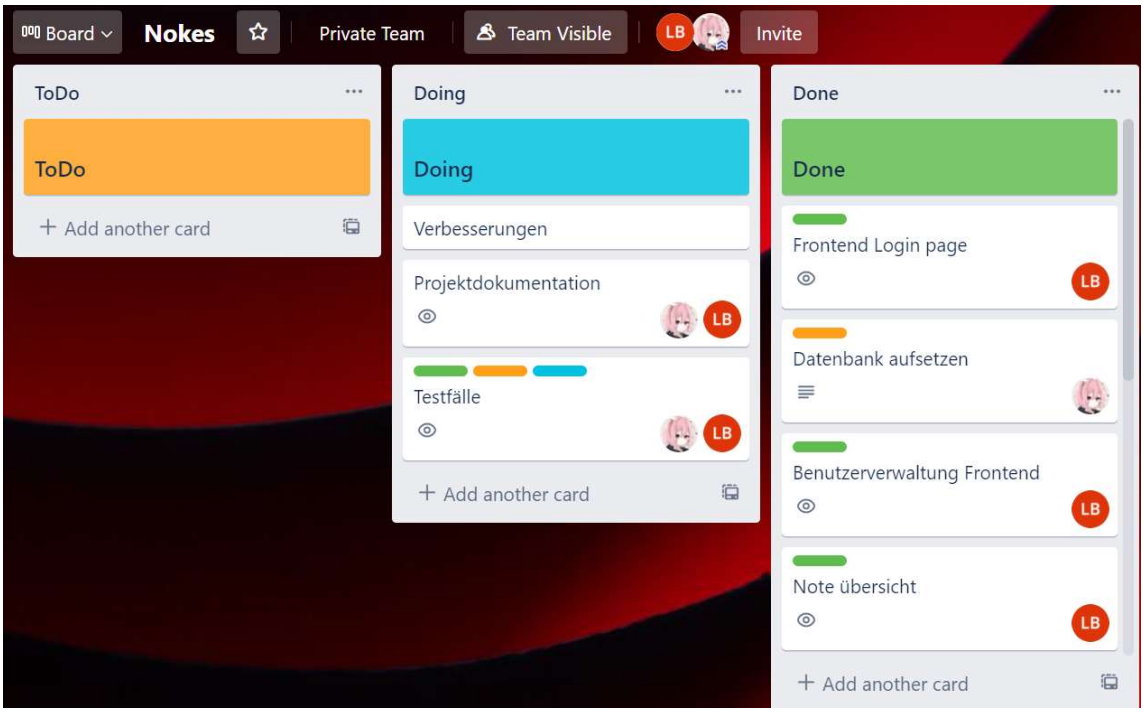
Source

```
1. ↩
2. <!DOCTYPE html>↩
3. <html lang="en">↩
4. ↩
5. <head>↩
6.     <meta charset="utf-8">↩
7.     <meta http-equiv="X-UA-Compatible" content="IE=edge">↩
8.     <meta name="viewport" content="width=device-width, initial-scale=1">↩
9.     <title>Nokes | Home</title>↩
10.    <!-- Icon für die Webseite -->↩
11.    <link rel="shortcut icon" href="./favicon.ico" type="image/x-icon">↩
12.    <!-- Bootstrap -->↩
13.    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet">↩
14. </head>↩
```

Der Code wurde mithilfe des HTML Validators von W3 (https://validator.w3.org/) validiert. Keine Fehlermeldungen oder Warnungen wurden erkannt.

CSS mussten wir nicht validieren, da wir Bootstrap nutzten.

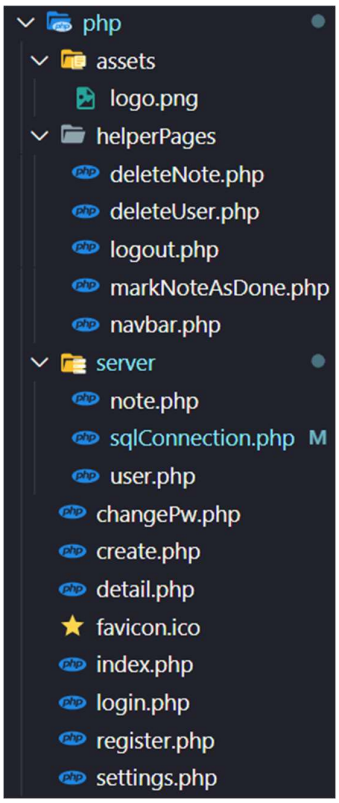
Projektplanung



Für die Projektplanung nutzen wir das Tool Trello, da wir beide auch schon miteinander Projekte mit diesem Tool durchgeführt haben. Durch Trello hatten wir beide die Möglichkeit unseren Fortschritt grob dem Anderen mitzuteilen.

Strukturierung des Codes

Wir versuchten, logische Funktionen ohne HTML in getrennte Dateien auszulagern.




```php
function loginUser($loginUser)
{
    // Database connection
    include('sqlConnection.php');

    $errorString = '';
```

Bei der Benennung der Variablen und Funktionen hielten wir uns an die gängigen Methoden. Alles sollte einen möglichst simplen und selbsterklärenden Namen bekommen.

Kommentiert seinen Code.

```php
    // Validation of userid
    // not empty
    // length: 3-20
    // only letters($letterOnly regex)
    $trimmedUserId = trim($user["userid"]);
```

Code wurde an den benötigten Stellen kommentiert.