

Projektdokumentation

Projektgesamtziel:

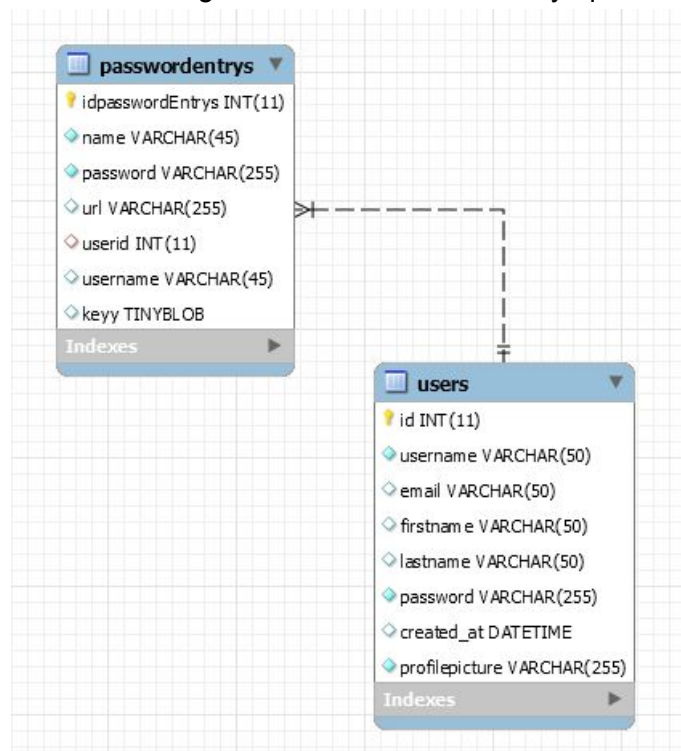
Das Ziel dieses Projekts war es, eine Plattform zu schaffen, auf der sich ein Benutzer registrieren/anmelden kann. Dann zur Hauptanwendung weiterleiten.

Innerhalb der Hauptanwendung kann der Benutzer persönliche Informationen einschliesslich eines Profilbildes erstellen/bearbeiten/löschen.

Die Anwendung bietet auch einen Passwort-Generator für den Benutzer, um Zeichenfolgen und sichere Passwörter zu erzeugen. Ein Passwort-Manager sollte erstellt werden, um Passworteingaben zu verwalten.

Technische Aspekte

Die Anwendung kommuniziert mit einer mysql-Datenbank. Dies sind die erstellten Tabellen:



Versionierung

Github wurde als Versionskontrolle verwendet. Die meiste Arbeit wurde im Master-Branch erledigt, die hCaptcha-Implementierung wurde jedoch in dev/captcha erledigt

<https://github.com/cstringer17/Hashedpotatoes>

Installationsanleitung

Die installationsanleitung ist auf dem Readme.md zu finden

Verwendete Technologien

PHP 7.2.34 (cli)

PHP ist die Haupt Programmiersprache, die innerhalb der Anwendung verwendet wird

font-awesome 5.15.1

Font-awesome wird verwendet, um kleine Icons zu rendern, die für den Benutzer gut sichtbar sind

Bootswatch 4.5.2

Bootswatch Lux wird als Bootstrap-Theme verwendet.

Clearbit Logos

Clearbit ist eine API, die Firmenlogos zur Verfügung stellt. Wenn man z.B. "spotify.com" abschickt, erhältet man das Spotify-Logo

jquery 3.5.1

Jquery wird für einige kleine Funktionen verwendet, die z.B. in bestimmten Bootstrap-Funktionen benötigt werden,

hCaptcha

Das hCaptcha-Widget schützt die Anwendungen vor Bots, Spam und anderen Formen des automatischen Missbrauchs

Testen

Siehe das Testskript für die Tests, die mit der Anwendung durchgeführt werden

[Testprotokoll](#)

Kompetenzen

1. Kann eine Projektantrag zu einem eigenen Projekt nach Vorlage erstellen.

Vor dem Projektstart wurde ein Projektantrag erstellt, indem das Ziel, Organisation und Ressourcen beschrieben sind. Es wurden auch eventuelle unsicherheiten Bezüglich des Projekts aufgelistet.

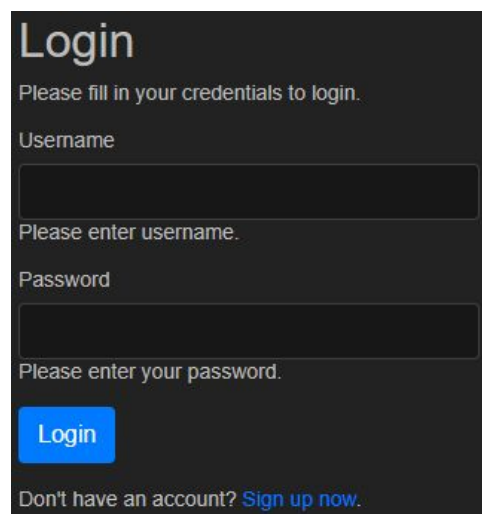
[Projektantrag](#)

Der Projektantrag wurde angenommen und von allen Beteiligten signiert.

2. Kann den Projektantrag in eine Projekt-Dokumentation überführen. In der Projektdokumentation werden die für dieses Modul relevanten Funktionen der Applikation beschrieben und erklärt

[Projektdokumentation](#)

3. Achtet auf Usability der Applikation, leitet den Benutzer mit sinnvollen Meldungen durch alle Prozesse.



The image shows a dark-themed login form. At the top, it says 'Login' in a large font. Below it, a message reads 'Please fill in your credentials to login.' There are two input fields: one for 'Username' and one for 'Password'. Below the 'Username' field, a message says 'Please enter username.' Below the 'Password' field, a message says 'Please enter your password.' There is a blue 'Login' button. At the bottom, it says 'Don't have an account? [Sign up now.](#)'

Wie im Beispiel zu sehen ist gibt es bei jedem Error ein sinnvollen Meldung an den Benutzer, auch es werden auch Alerts von Bootstrap eingesetzt.

COPY TEXT

Password Copied! ✕

4. Prüft alle Benutzereingaben client- und serverseitig (Pflichtfelder, Länge, Datentyp). Verwendet sinnvolle Datentypen und Speichergrößen zur Speicherung der Informationen.

Im Code wurde darauf geachtet, dass die Serverseitige und Clientseitige Benutzereingabe überprüft wird. Dies wurde mit input type, required und minlength für die Clientseite und mit trim(), strlen() und isset() umgesetzt.

Clientside:

```
<input type="text" name="username" required="true" minlength="5"
maxlength="50" class="form-control" value="<?php echo $username; ?>">
```

Serverside:

```
// Validate username
if (empty(trim($_POST["username"]))) {
    $username_err = "Please enter a username.";
} elseif (strlen(trim($_POST["username"])) > 50) {
    $username_err = "Username must be less than 50 characters.";
```

5. Verwendet Session Handling zur Sitzungsverfolgung und stellt für autorisierte Benutzer neben der Möglichkeit das Passwort zu ändern, zusätzliche Funktionen zur Verfügung.

```
<?php
// Initialize the session
session_start();

// Check session for user
if (isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true) {
    header("location: welcome.php");
    exit;
}
```

```
if (password_verify($password, $hashed_password)) {
    // Password is correct, so start a new session
    session_start();
    // Store data in session variables
    $_SESSION["loggedin"] = true;
    $_SESSION["id"] = $id;
    $_SESSION["username"] = $username;

    // Redirect user to welcome page
    header("location: welcome.php");
```

Beim logout wird die session gelöscht

```
// Unset all of the session variables
$_SESSION = array();

// Redirect to login page
header("location: login.php");
exit;

// remove all session variables
session_unset();

// destroy the session
session_destroy();
```

- 6. Speichert schutzwürdige Informationen sicher und nach neustem Stand der Technik in der Datenbank. Achtet darauf, das der Datenbankbenutzer die für die Kommunikation zwischen Web-Applikation und Datenbank minimal nötigen Berechtigungen besitzt.**

register.php

```
// Set parameters
$params['username'] = $username;
$params['password'] = password_hash($password,
PASSWORD_DEFAULT); // Creates a password hash
```

sodiumtest.php (ist zu finden in uploadpassword.php und passwordmanager.php)

```
<?php

$key = random_bytes(SODIUM_CRYPTO_SECRETBOX_KEYBYTES);
$nonce = random_bytes(SODIUM_CRYPTO_SECRETBOX_NONCEBYTES);
$ciphertext = sodium_crypto_secretbox("password", $nonce, $key);
$encoded = base64_encode($nonce . $ciphertext);

echo $encoded . "<br>";
echo $key . "<br>";

$decoded = base64_decode($encoded);
$nonce = mb_substr($decoded, 0, SODIUM_CRYPTO_SECRETBOX_NONCEBYTES,
'8bit');
```

```
$ciphertext = mb_substr($decoded, SODIUM_CRYPTO_SECRETBOX_NONCEBYTES,
null, '8bit');
$plaintext = sodium_crypto_secretbox_open($ciphertext, $nonce, $key);
echo $plaintext;
```

Datenbankbenutzer:

```
CREATE USER 'test'@'%' IDENTIFIED BY 'qwert_1337';
```

```
GRANT SELECT ON hashedpotatoes.users TO 'test'@'%' WITH GRANT OPTION;
GRANT INSERT ON hashedpotatoes.users TO 'test'@'%' WITH GRANT OPTION;
GRANT DELETE ON hashedpotatoes.users TO 'test'@'%' WITH GRANT OPTION;
GRANT UPDATE ON hashedpotatoes.users TO 'test'@'%' WITH GRANT OPTION;
```

```
GRANT SELECT ON hashedpotatoes.passwordentrys TO 'test'@'%' WITH GRANT
OPTION;
GRANT INSERT ON hashedpotatoes.passwordentrys TO 'test'@'%' WITH GRANT
OPTION;
GRANT DELETE ON hashedpotatoes.passwordentrys TO 'test'@'%' WITH GRANT
OPTION;
GRANT UPDATE ON hashedpotatoes.passwordentrys TO 'test'@'%' WITH GRANT
OPTION;
```

7. Verhindert SQL-Injection, Script-Injection und Session-Hijacking.

In login.php und register.php zu finden.

```
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>"
method="post">
```

Bei jede SQL Query wird der Statement zuerst prepared um SQL Injections zu verhindern

```
if ($stmt = $mysqli->prepare($sql)) {
    // Bind variables to the prepared statement as parameters
    $stmt->bind_param("sssss", $param_name, $param_password,
    $param_url, $param_username, $param_keyy);
```

Die parameters werden nach der prepare gesetzt

Session Hijacking wird bei der Kompetenz Session-Handling verhindert indem die session beim logout gelöscht wird.

Bietet die Möglichkeit zur Registrierung und Login an der Web-Applikation an. Dazu werden sinnvolle Informationen zum Benutzer erfasst.

Username, Password und Email werden von Benutzer erfasst. Der Vorname und Nachname sowie das Profilbild kann nachher bei profile.php hinzugefügt werden.

- 8. Ermöglicht das Erfassen, Ändern und Löschen zusätzlicher Daten über die Webseite in der Datenbank. Dieses Daten können von nicht autorisierten Benutzern angesehen werden.**

HI, ASTERIX. DO YOU WANT TO UPLOAD A PASSWORD?

BACK TO PWMANAGER

Username		Url	
Password		Name	

SAVE

☐ Dark Mode
Generate a new Password!

COPY TEXT

8

☐ A-Z

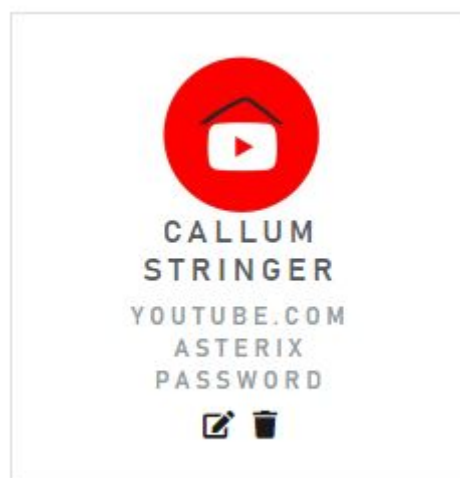
☐ 0-9

☐ !@#\$%^&*

GENERATE PASSWORD

Jeder Benutzer kann einen Passworteintrag hochladen, der Benutzer kann nur seine eigenen Passwörter einsehen.

Nach dem hochladen, kann der Benutzer all seine Passwörter in der Passwort-Manager-Ansicht einsehen.



Jeder Eintrag kann durch Klicken auf die Symbole gelöscht und bearbeitet werden.

Nebenbei kann der Benutzer Informationen über sich selbst speichern und ändern. Ein Profilbild kann auch gesetzt werden.

In der Profilbild Übersicht kann jeder Benutzer sein Passwort zurücksetzen und sein Konto löschen. Beim Löschen eines Kontos wird auch jeder zum Benutzer gehörende Passworteintrag gelöscht

9. Setzt für die Projektplanung ein geeignetes Planungstool ein und verwendet dieses aktiv während der Entwicklung des Projektes.

Für die Planung und Durchführung des Projekts nutzten wir das Online-Tool trello, um offene Aufgaben zu verfolgen und sicherzustellen, dass nichts vergessen geht.

<https://trello.com/b/ygnttmtY/m151>

10. Kennt Möglichkeiten zur Strukturierung von Quellcode und der Umsetzung unter Berücksichtigung von Codierungsrichtlinien und wendet diese in seinem Projekt an.

Alle Variablen und Dateien verwenden Namen, die die Bedeutung erklären. Jeder wiederholte Code wurde in eine eigene Datei geschrieben und mit `require_once` oder `include()` eingebunden.

11. Kommentiert seinen Code.

Der Quellcode wurde kommentiert.

Validierung

Der Code wurde validiert, allerdings blieben einige Fehler übrig, die vom W3C-Validator gemeldet wurden, die bei der Behebung Fehler verursachten und die Website kaputt machten. Daher wurden diese ignoriert. Alle Fehler, die behebbar waren und die Funktionalität der Website nicht veränderten, wurden korrigiert.