Technische Projektdokumentation

Beschreibt die technische Umsetzung der im Kompetenzraster geforderten Kompetenzen welche mit Projekt gekennzeichnet sind.

Modul 151

Elias Mattern und Raquel Lima

25. Oktober 2021

Inhaltsverzeichnis

Kompetenzbereich: Handlungsziel 2 (133)

Kompetenz: In meinem Projekt werden alle Benutzereingaben client- und serverseitig validiert.

Beispiel: include/registration.php

Input Feld wird clientseitig validiert:

```
<div class="form-floating mb-3">
    <input type="text" name="firstname" class="form-control rounded-4" id="firstname" value="<?php echo $firstname ?>"
    placeholder="First name" maxlength="30" required="true">
    <label for="firstname">First name</label>
</div>
```

Eingabe wird serverseitig validiert:

```
// Wurden Daten mit "POST" gesendet?
if ($_SERVER['REQUEST_METHOD'] == "POST") {

    // Vorname ausgefüllt?
    if (isset($_POST['firstname'])) {
        //trim and sanitize
        $firstname = htmlspecialchars(trim($_POST['firstname']));

        //mindestens 1 Zeichen und maximal 30 Zeichen lang
        if (empty($firstname) || strlen($firstname) > 30) {
            $error .= "Geben Sie bitte einen korrekten Vornamen ein.<br />";
        }
    } else {
        $error .= "Geben Sie bitte einen Vornamen ein.<br />";
    }
```

Kompetenz: In meinem Projekt wird das Session-Handling korrekt eingesetzt. Ein angemeldeter Benutzer hat Zugriff auf weitere Funktionen, welche nicht angemeldeten Benutzern verwehrt bleiben.

Beispiel: index.php

```php
// Sessionhandling starten
session_start();
```

Wir prüfen mit einer if-Schleife, ob der Benutzer angemeldet ist. Wenn der Benutzer eingeloggt ist, sieht er ein Begrüssungslabel mit seinem Benutzernamen. Wenn nicht, sieht er die Website-Beschreibung und die Buttons "Sign-up" und "Login".

```php
//wenn Session personalisiert
if (isset($_SESSION['loggedin']) && $_SESSION['loggedin']) {
    echo '<p class="lead text-muted">Welcome ', $_SESSION['username'], '!</p>';

    if (isset($_SESSION['isAdmin']) and $_SESSION['isAdmin']) {

        include('include/addRestaurantModal.php');
        echo "<a href='' class='btn btn-dark my-2 text-warning' data-toggle='modal' data-target='#addRestaurantModal'>Create Restaurant</
        a>";
    }
} else {
    //wenn Session nicht personalisiert
    echo '<p class="lead text-muted">Find the best restaurants that deliver.<br>Get contactless delivery for restaurant takeout,
    groceries, and more! Order food online at home!</p>';
    echo '<p>
        <a href="" class="btn btn-warning my-2" data-toggle="modal" data-target="#modalSignup">Sign-up</a>
        <a href="" class="btn btn-dark my-2" data-toggle="modal" data-target="#modalSignin">Login</a>
    </p>';
}
```

Kompetenz: Ein angemeldeter Benutzer kann sich wieder abmelden. Die Session wird dabei korrekt beendet.

Beispiel: include/nav.php -> include/loggedinNav.php -> include/logout.php

In der Datei nav.php wird mit einer if-Schleife geprüft, ob der Benutzer eingeloggt ist. Wenn das der Fall ist, wird die Datei loggedinNav.php angezeigt.


```php
//wenn Session personalisiert
if (isset($_SESSION['loggedin'])) {
    include('include/loggedinNav.php');

} else {
    //wenn Session nicht personalisiert
    echo '<button type="button" class="btn btn-warning" data-toggle="modal" data-target="#modalSignup">Sign-up</button>';
    echo '<button id="loginBtn" type="button" class="btn btn-outline-light me-2" data-toggle="modal" data-target="#modalSignin"
    style="margin-left: 15px;">Login</button>';
}
```

In der Datei loggedinNav.php haben wir ein Dropdown-Menü. Eine der Optionen ist Sign out. Wenn man darauf klickt, wird die Datei logout.php ausgeführt.

In der Datei logout.php wird die Session korrekt beendet und der Benutzer wird zurück zu index.php weitergeleitet, wo nav.php wieder angezeigt wird

```php
<?php
session_start();
$_SESSION = array();
session_destroy();
header('Location: ../index.php');
?>
```

Kompetenz:In meinen Projekt werden Script-Injection, Session-Fixation und Session-Hijacking konsequent verhindert.

Beispiel: include/login.php

Script-Injection: In diesem Beispiel prüfen wir mit einer if-Schleife, ob der username in der POST-Methode gesetzt ist. Wenn das der Fall ist, entfernen wir die Leerzeichen und auch die vordefinierten Zeichen von beiden Seiten des Strings mit der Methode trim(). Dann wandeln wir vom Benutzer eingegeben HTML-Code mit der Methode htmlspecialchars() in HTML-Entities um. Diese String wird dann in einer Variablen gespeichert, die angezeigt oder in der Datenbank gespeichert wird. Und so verhindern wir Script-Injection in allen Eingabefeldern, die wir anzeigen oder in der Datenbank speichern möchten.

```php
if (isset($_POST['username'])) {
    //trim and sanitize
    $username = htmlspecialchars(trim($_POST['username']));

    // Prüfung username
    if (empty($username) || !preg_match("/(?=.*[a-z])(?=.*[A-Z])[a-zA-Z]{6,30}/", $username)) {
        $error .= "Der Benutzername entspricht nicht dem geforderten Format.<br />";
    }
} else {
    $error .= "Geben Sie bitte den Benutzername an.<br />";
}
```

Session-fixation und Session-Hijacking: In diesem Beispiel verändern wir die Session-ID der aktiven Session nach dem personalisieren der Session mit der methode session_regenerate_id(). Dabei wird die Session-ID im Client-Cookie und der Dateiname der Session auf dem Server durch eine neue ID ersetzt. Und so können wir den Aufwand für diese Angriffsszenarien einfach erhöhen.

```php
if (password_verify($password, $row['password'])) {

    // Session personifizieren
    $_SESSION['id'] = $row['id'];
    $_SESSION['username'] = $username;
    $_SESSION['loggedin'] = true;
    $_SESSION['isAdmin'] = $row['admin'];
    $_SESSION['products'] = array();


    // Session ID regenerieren
    session_regenerate_id(true);

    // weiterleiten auf index.php
    header("location: index.php");

    // Script beenden
    die();
```

Kompetenzbereich: Handlungsziel 2 (151)

Kompetenz: In meinem Projekt werden sensible Daten wie das Passwort mit sicheren und aktuellen Methoden gehasht und gesaltet.

Beispiel: include/registration.php

In diesem Beispiel wird nach der Validierung der Benutzereingabe und der Speicherung unter der Variablen password das Passwort mit der Methode password_hash() gehasht und gesaltet und anschliessend unter der Variablen password_hash in der Datenbank gespeichert.

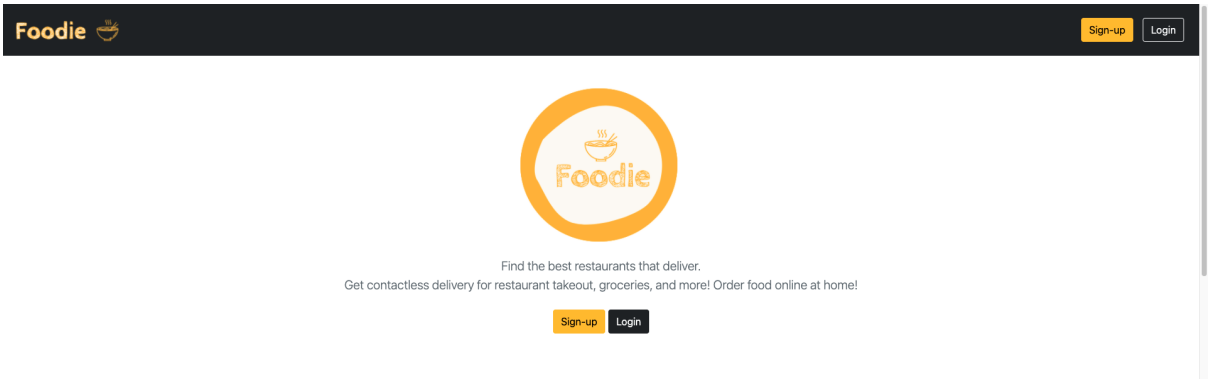
```php
// Password haschen
$password_hash = password_hash($password, PASSWORD_DEFAULT);

// Query erstellen
$query = "Insert into users (firstname, lastname, username, password, email) values (?,?,?,?,?)";

// Query vorbereiten
$stmt = $mysqli->prepare($query);
if ($stmt === false) {
    $error .= 'prepare() failed ' . $mysqli->error . '<br />';
}

// Parameter an Query binden
if (!$stmt->bind_param('sssss', $firstname, $lastname, $username, $password_hash, $email)) {
    $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
}

// Query ausführen
if (!$stmt->execute()) {
    $error .= 'execute() failed ' . $mysqli->error . '<br />';
}
```

Kompetenzbereich: Handlungsziel 3 (151)

Kompetenz: Ein Benutzer kann sich an meinem Projekt registrieren. Dazu erfasse ich sinnvolle Daten des Benutzers.

Beispiel: index.php -> include/registration.php

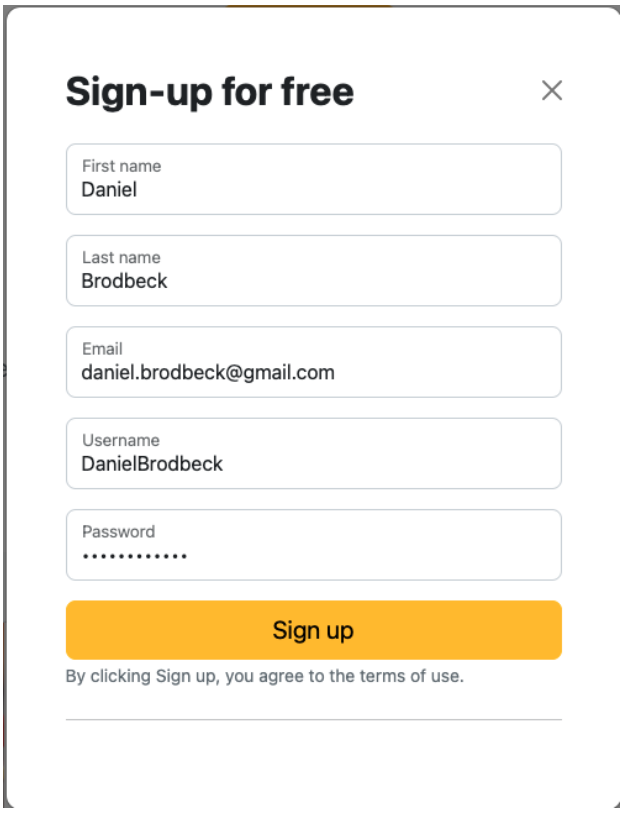
Der Benutzer kann sich über den Button "Sign-up" registrieren, der sich in der Navigation und in der Mitte der Seite index.php befindet. Der Button wird nur angezeigt, wenn man nicht eingeloggt ist.



Beim Anklicken wird das Modal in der Datei include/registration.php aktiviert.

```
<a href="" class="btn btn-warning my-2" data-toggle="modal" data-target="#modalSignup">Sign-up</a>
```

Der Benutzer kann dann seine persönlichen Daten eingeben und mit einem Klick auf "Sign up" werden diese Daten im Array $_POST gespeichert.



```html
<form action="" method="POST">
    <div class="form-floating mb-3">
        <input type="text" name="firstname" class="form-control rounded-4" id="firstname"  placeholder="First name" maxlength="30"
        required="true">
        <label for="firstname">First name</label>
    </div>
    <div class="form-floating mb-3">
        <input type="text" name="lastname" class="form-control rounded-4" id="lastname"  placeholder="Last name" maxlength="30"
        required="true">
        <label for="lastname">Last name</label>
    </div>
    <div class="form-floating mb-3">
        <input type="email" name="email" class="form-control rounded-4" id="email"  placeholder="name@example.com" maxlength="100"
        required="true">
        <label for="email">Email</label>
    </div>
    <div class="form-floating mb-3">
        <input type="text" name="username" class="form-control rounded-4" id="username"  pattern="(?=.*[a-z])(?=.*[A-Z])[a-zA-Z]{6,}"
        placeholder="Password" maxlength="30" required="true">
        <label for="username">Username</label>
    </div>
    <div class="form-floating mb-3">
        <input type="password" name="password" class="form-control rounded-4" id="password" pattern="(?=^.{8,}$)((?=.*\d+)(?=.*\W+))(?![.
        \n])(?=.*[A-Z])(?=.*[a-z]).*$" placeholder="Password" maxlength="255" required="true">
        <label for="password">Password</label>
    </div>
    <button class="w-100 btn btn-lg rounded-4 btn-warning" name="button" value="submit" type="submit">Sign up</button>
    <small class="text-muted">By clicking Sign up, you agree to the terms of use.</small>
    <hr class="my-4">
</form>
```

Die Angaben werden dann serverseitig validiert und unter Variabeln gespeichert.

```php
// Wurden Daten mit "POST" gesendet?
if ($_SERVER['REQUEST_METHOD'] == "POST") {

    // Vorname ausgefüllt?
    if (isset($_POST['firstname'])) {
        //trim and sanitize
        $firstname = htmlspecialchars(trim($_POST['firstname']));

        //mindestens 1 Zeichen und maximal 30 Zeichen lang
        if (empty($firstname) || strlen($firstname) > 30) {
            $error .= "Geben Sie bitte einen korrekten Vornamen ein.<br />";
        }
    } else {
        $error .= "Geben Sie bitte einen Vornamen ein.<br />";
    }

    // Nachname ausgefüllt?
    if (isset($_POST['lastname'])) {
        //trim and sanitize
        $lastname = htmlspecialchars(trim($_POST['lastname']));

        //mindestens 1 Zeichen und maximal 30 Zeichen lang
        if (empty($lastname) || strlen($lastname) > 30) {
            $error .= "Geben Sie bitte einen korrekten Nachname ein.<br />";
        }
    } else {
        $error .= "Geben Sie bitte einen Nachname ein.<br />";
    }
```

Falls alle Eingaben in Ordnung sind, wird das Passwort gehasht und gesaltet und die Daten werden in die Datenbank geschrieben.

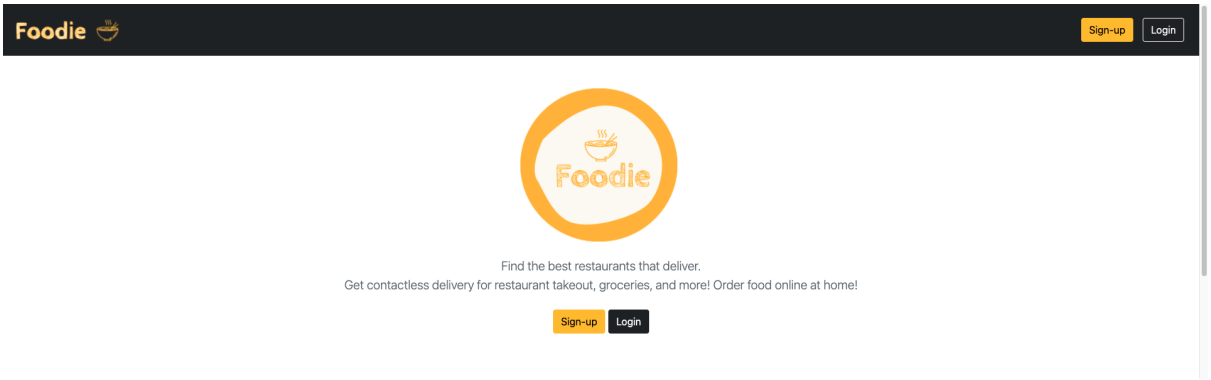
```php
// wenn kein Fehler vorhanden ist, schreiben der Daten in die Datenbank
if (empty($error)) {

    // Password haschen
    $password_hash = password_hash($password, PASSWORD_DEFAULT);

    // Query erstellen
    $query = "Insert into users (firstname, lastname, username, password, email) values (?,?,?,?,?)";

    // Query vorbereiten
    $stmt = $mysqli->prepare($query);
    if ($stmt === false) {
        $error .= 'prepare() failed ' . $mysqli->error . '<br />';
    }

    // Parameter an Query binden
    if (!$stmt->bind_param('sssss', $firstname, $lastname, $username, $password_hash, $email)) {
        $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
    }

    // Query ausführen
    if (!$stmt->execute()) {
        $error .= 'execute() failed ' . $mysqli->error . '<br />';
    }
```

id	firstname	lastname	username	password	email
20	Daniel	Brodbeck	DanielBrodbeck	$2y$10$MrIfPE1on0rKQz144tQ5keYvh2/.ioEWXCWc36r4bFo...	daniel.brodbeck@gmail.com

Kompetenz: Ein Benutzer kann sich an meinem Projekt anmelden. Nach der Anmeldung stehen dem Benutzer weitere Funktionen zur Verfügung.

Beispiel: index.php -> include/login.php -> index.php

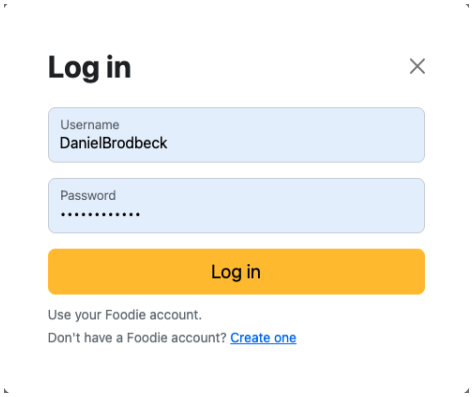
Nach der Registrierung landet der Benutzer wieder in index.php. Er kann sich über den Button "Login" einloggen, der sich in der Navigation und in der Mitte der Seite befindet.



Beim Anklicken wird das Modal in der Datei include/login.php aktiviert.

```
<a href="" class="btn btn-dark my-2" data-toggle="modal" data-target="#modalSignin">Login</a>
```

Der Benutzer kann dann seinen Benutzernamen und sein Passwort eingeben. Mit einem Klick auf "Log in" werden diese Daten im Array $_POST gespeichert.



```
<form action="" method="POST">
    <div class="form-floating mb-3">
        <input type="text" name="username" class="form-control rounded-4" id="username" placeholder="Password" pattern="(?=.*[a-z])(?=.*
        [A-Z])[a-zA-Z]{6,}" maxlength="30" required="true">
        <label for="username">Username</label>
    </div>
    <div class="form-floating mb-3">
        <input type="password" name="password" class="form-control rounded-4" id="password" placeholder="Password" pattern="(?=^.{8,}$)((?
        =.*\d+)(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$" maxlength="255" required="true">
        <label for="floatingPassword">Password</label>
    </div>
    <button class="w-100 mb-2 btn btn-lg rounded-4 btn-warning btn btn-info" name="button" value="submit" type="submit">Log in</button>
    <small class="text-muted">Use your Foodie account.<br>Don't have a Foodie account? <a href="" class="" data-toggle="modal"
    data-target="#modalSignup">Create one</a></small>
</form>
```

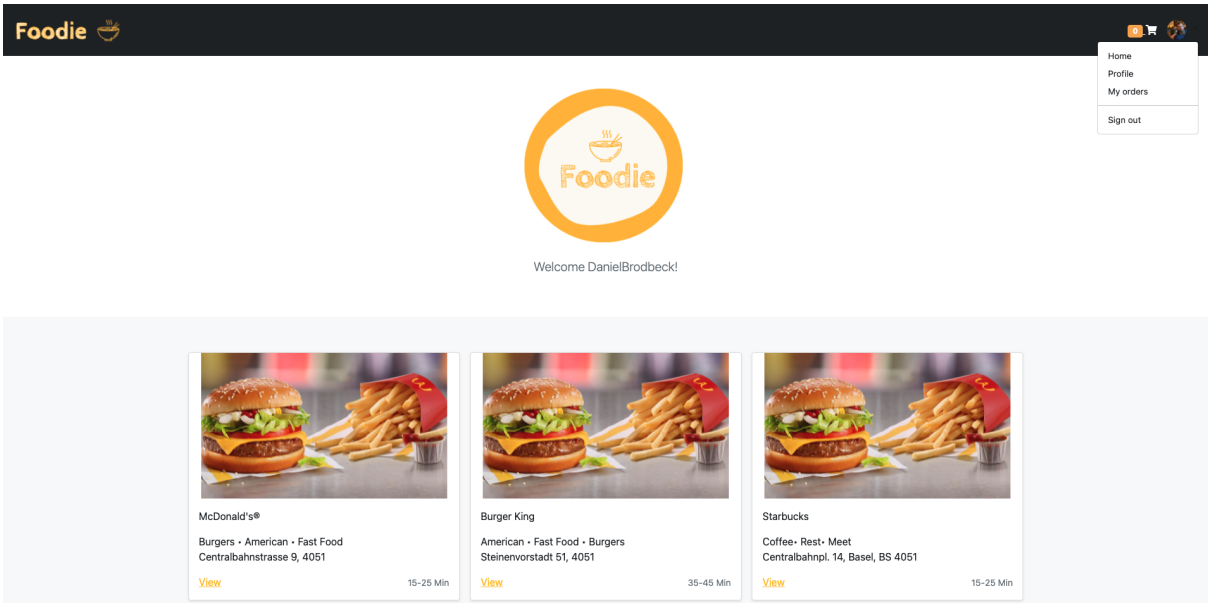
Die Angaben werden dann serverseitig validiert und unter Variabeln gespeichert.

```
// Formular wurde gesendet und Besucher ist noch nicht angemeldet.
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // username
    if (isset($_POST['username'])) {
        //trim and sanitize
        $username = htmlspecialchars(trim($_POST['username']));

        // Prüfung username
        if (empty($username) || !preg_match("/(?=.*[a-z])(?=.*[A-Z])[a-zA-Z]{6,30}/", $username)) {
            $error .= "Der Benutzername entspricht nicht dem geforderten Format.<br />";
        }
    } else {
        $error .= "Geben Sie bitte den Benutzername an.<br />";
    }
    // password
    if (isset($_POST['password'])) {
        //trim and sanitize
        $password = htmlspecialchars(trim($_POST['password']));
        // passwort gültig?
        if (empty($password) || !preg_match("/(?=^.{8,255}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/", $password)) {
            $error .= "Das Passwort entspricht nicht dem geforderten Format.<br />";
        }
    } else {
        $error .= "Geben Sie bitte das Passwort an.<br />";
    }
```

Wir vergleichen dann das eingegebene Passwort mit dem Passwort in der Datenbank mit password_verify(). Wenn es korrekt ist, wird die Session personifiziert, die Session-ID neu generiert und der Benutzer wird wieder zur index.php geleitet.

```
// kein Fehler
if (empty($error)) {
    // Query erstellen
    $query = "SELECT id, username, password, admin from users where username = ?";

    // Query vorbereiten
    $stmt = $mysqli->prepare($query);
    if ($stmt === false) {
        $error .= 'prepare() failed ' . $mysqli->error . '<br />';
    }
    // Parameter an Query binden
    if (!$stmt->bind_param("s", $username)) {
        $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
    }
    // Query ausführen
    if (!$stmt->execute()) {
        $error .= 'execute() failed ' . $mysqli->error . '<br />';
    }
    // Daten auslesen
    $result = $stmt->get_result();

    // Userdaten lesen
    if ($row = $result->fetch_assoc()) {

        // Passwort ok?
        if (password_verify($password, $row['password'])) {

            // Session personifizieren
            $_SESSION['id'] = $row['id'];
            $_SESSION['username'] = $username;
            $_SESSION['loggedin'] = true;
            $_SESSION['isAdmin'] = $row['admin'];
            $_SESSION['products'] = array();


            // Session ID regenerieren
            session_regenerate_id(true);

            // weiterleiten auf index.php
            header("location: index.php");

            // Script beenden
            die();
```

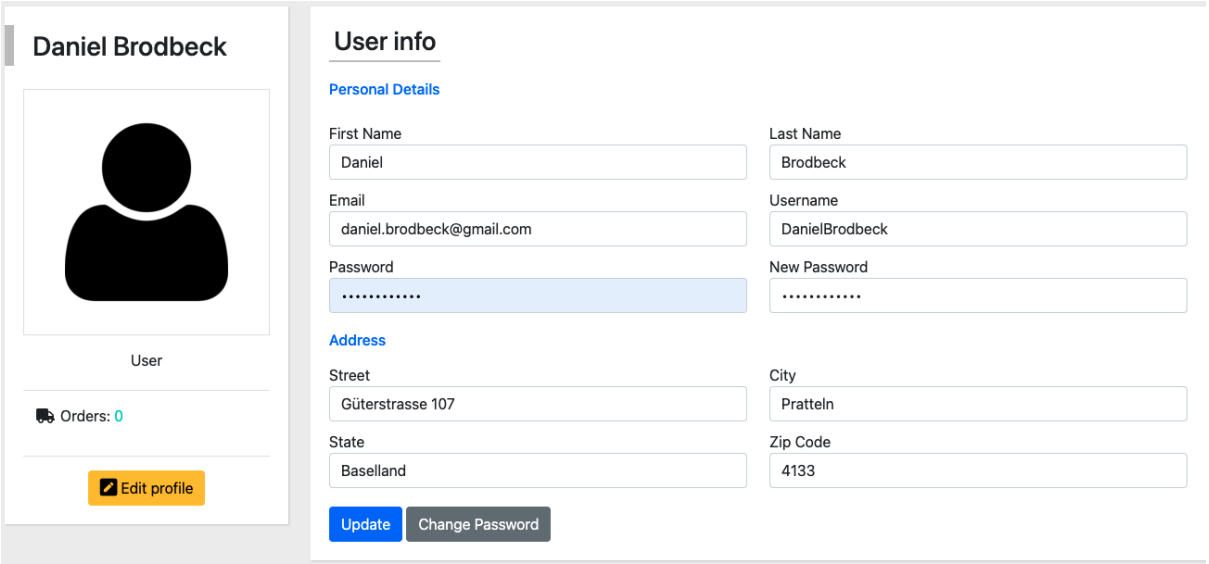
Da der Benutzer nun eingeloggt ist, kann er die loggedinNav.php mit seinem Profilbild, dem Warenkorb und einem Dropdown-Menü sehen, über das er sein Profil und seine Bestellungen aufrufen kann. Der Benutzer kann sich nun auch abmelden, auf die Restaurantkarten klicken und Essen kaufen.



Kompetenz: In meinem Projekt kann ein angemeldeter Benutzer sein Passwort ändern.

Beispiel: profile.php

Der Benutzer kann sein Passwort unter profile.php ändern, indem er auf den Button "Edit profile" und dann auf "Change Password" klickt. Um das Passwort zu ändern, muss der Benutzer alle Eingabefelder ausfüllen, das korrekte aktuelle Passwort und das neue Passwort eingeben und auf "Update" klicken.



Nach dem man auf Update gedrückt hat, werden alle Eingaben validiert und bei Fehlern kommt eine Fehler Nachricht. Das Feld New Password ist nicht Required, falls man sein Passwort nicht ändern möchte und wird deshalb nicht geprüft ob es empty ist.

```php
if (isset($_POST['newPassword'])) {
    $newPassword = htmlspecialchars(trim($_POST['newPassword']));

}
```

```php
$query = "SELECT * FROM users WHERE {$_SESSION['id']}=id;";

$stmt = $mysqli->prepare($query);

$stmt->execute();

$result = $stmt->get_result();

if ($row = $result->fetch_assoc()) {
```

```php
// wenn kein Fehler vorhanden ist, schreiben der Daten in die Datenbank
if (empty($error) && password_verify($password, $row['password'])) {
    if (!empty($newPassword)) {
        // Password haschen
        $password_hash = password_hash($newPassword, PASSWORD_DEFAULT);
    } else {
        $password_hash = password_hash($password, PASSWORD_DEFAULT);
    }
```

Die User ID um den Nutzer in der Datenbank zu finden holen wir aus der Session. Somit können wir jetzt falls, die Validierung ohne Fehler durchgelaufen ist, dass eingegebene Passwort mit dem Passwort aus der Datenbank vergleichen. Falls die Passwörter übereinstimmen prüfen wir ob ein neues Passwort eingegeben wurde. Wenn nicht setzen wir die Variable passwort_hash auf das alte Passwort. Falls ein neues eingegeben wurde setzen wir die Variable auf das neue Passwort. Natürlich wird das Passwort hierbei gehashed und gesalted.

```php
// Query erstellen
$query = "UPDATE  users  SET firstname = ?, lastname = ?, username = ?, password = ?, email = ?, street = ?, city = ?, state = ?, zip = ? WHERE users.id = {$_SESSION['id']};";

// Query vorbereiten
$stmt = $mysqli->prepare($query);
if ($stmt === false) {
    $error .= 'prepare() failed ' . $mysqli->error . '<br />';
}

// Parameter an Query binden
if (!$stmt->bind_param('sssssssss', $firstname, $lastname, $username, $password_hash, $email, $street, $city, $state, $zip)) {
    $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
}

// Query ausführen
if (!$stmt->execute()) {
    $error .= 'execute() failed ' . $mysqli->error . '<br />';
}
```

Danach wird alles in der Datenbank geupdatet. Je nachdem mit dem neuen Passwort oder mit dem alten Passwort.

Kompetenz: In meinem Projekt können angemeldete Benutzer weitere Informationen in der Datenbank erfassen. Diese Datensätze können ausschliesslich vom Ersteller dieser Datensätze einzeln geändert und gelöscht werden.

Beispiel: index.php -> include/addRestaurantModal.php -> restaurant.php -> editRestaurant.php -> delereRestaurant

In unserem Projekt haben wir einen einzigen Admin-Benutzer angelegt, und nur er kann Restaurants anlegen, bearbeiten und löschen sowie Lebensmittel hinzufügen. Die Benutzer-Tabelle in unserer Datenbank hat eine Admin-Spalte und für diesen Benutzer ist diese auf true gesetzt.

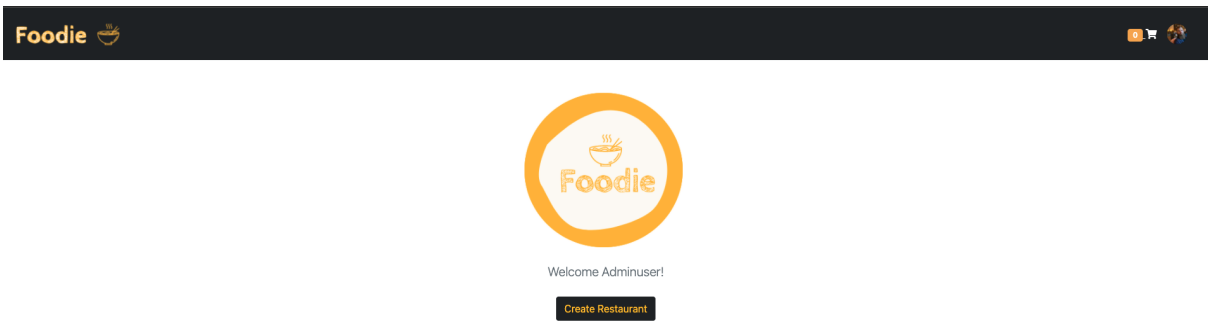
Um sicherzustellen, dass es sich um einen Admin-Benutzer handelt, personifizieren wir die Session in login.php, indem wir $_SESSION['isAdmin'] setzen. Wie im folgenden Beispiel zu sehen ist.

```php
// Session personifizieren
$_SESSION['id'] = $row['id'];
$_SESSION['username'] = $username;
$_SESSION['loggedin'] = true;
$_SESSION['isAdmin'] = $row['admin'];
$_SESSION['products'] = array();
```

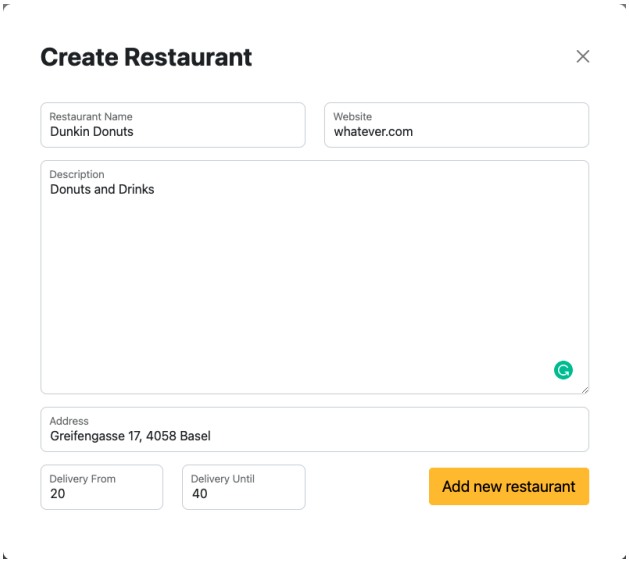
Um Funktionen hinzuzufügen, die nur der Admin ausführen kann, testen wir einfach mit einer if-Schleife ob $_SESSION[isAdmin] gesetzt ist und true ist.

```php
if (isset($_SESSION['isAdmin']) and $_SESSION['isAdmin']) {

    include('include/addRestaurantModal.php');
    echo "<a href='' class='btn btn-dark my-2 text-warning' data-toggle='modal' data-target='#addRestaurantModal'>Create Restaurant</a>";
}
```

Nachdem man sich als Admin angemeldet hat, kann man den Button "Create Restaurant" sehen und Restaurants erstellen. Der Button aktiviert das Modal in der Datei include/addRestaurantModal.php.



Hier kann man die Daten des neuen Restaurants eingeben, das man erstellen möchte



Wenn man auf "Add new restaurant" klickt, werden die eingegebenen Daten an das Array $POST gesendet.

```
<form class="row" action="" method="POST" id="addRestaurantForm">
    <div class="form-floating col-md-6 mb-3">
        <input type="text" name="restaurantName" class="form-control rounded-4" id="restaurantName" placeholder="Restaurant Name"
        maxlength="60" required="true">
        <label class="px-4" for="restaurantName">Restaurant Name</label>
    </div>
    <div class="form-floating col-md-6 mb-3">
        <input type="text" name="website" class="form-control rounded-4" id="website" placeholder="Website (https://www.myRestaurant.com)
        " maxlength="256" required="true">
        <label class="px-4">Website</label>
    </div>
    <div class="form-floating mb-3">
        <textarea name="description" style="height: 300px;" class="form-control rounded-4" id="description" placeholder="Description"
        maxlength="256" required="true" cols="30" rows="10" required="true"></textarea>
        <label class="px-4" for="description">Description</label>
    </div>
    <div class="form-floating mb-3">
        <input type="text" name="address" class="form-control rounded-4" id="address" placeholder="Address" maxlength="256"
        required="true">
        <label class="px-4">Address</label>
    </div>
    <div class="form-floating col-md-3 mb-3">
        <input type="number" name="from" class="form-control rounded-4" id="from" placeholder="From" min="0" max="500" required="true">
        <label class="px-4">Delivery From</label>
    </div>
    <div class="form-floating col-md-3 mb-3">
        <input type="number" name="until" class="form-control rounded-4" id="until" placeholder="Until" required="true">
        <label class="px-4">Delivery Until</label>
    </div>
    <div class="form-floating col-md-6" style="text-align: right;">
        <button class="btn btn-lg btn-warning mt-1" name="addRestaurantForm" form="addRestaurantForm" type="submit">Add new restaurant</
        button>
    </div>
</form>
```

Dann werden die Eingaben validiert und unter Variablen gespeichert.

```php
if (isset($_POST['restaurantName'])) {
    $restaurantName = trim(htmlspecialchars($_POST['restaurantName']));
    if (empty($restaurantName) || strlen($restaurantName) > 60) {
        $error .= " Invalid Restaurant Name";
    }
} else {
    $error .= " Invalid Restaurant Name";
}
if (isset($_POST['website'])) {
    $website = trim(htmlspecialchars($_POST['website']));
    if (empty($website) || strlen($website) > 256) {
        $error .= " Invalid website";
    }
} else {
    $error .= " Invalid website";
}
if (isset($_POST['description'])) {
    $description = trim(htmlspecialchars($_POST['description']));
    if (empty($description) || strlen($description) > 256) {
        $error .= " Invalid description";
    }
} else {
    $error .= " Invalid description ";
}
if (isset($_POST['address'])) {
    $address = trim(htmlspecialchars($_POST['address']));
    if (empty($address) || strlen($address) > 256) {
        $error .= " Invalid address";
    }
} else {
    $error .= " Invalid address";
}
```

Wenn es keine Fehler gibt, werden die Daten in die Datenbank geschrieben.

```php
if (empty($error)) {

    $query = "INSERT INTO `restaurants` (`name`, `description`, `place`, `website`, `delivery-from`, `delivery-until`) VALUES (?, ?, ?, ?, ?, ?)";

    // Query vorbereiten
    $stmt = $mysqli->prepare($query);
    if ($stmt === false) {
        $error .= 'prepare() failed ' . $mysqli->error . '<br />';
    }
    // Parameter an Query binden
    if (!$stmt->bind_param('ssssii', $restaurantName, $description, $address, $website, $from, $until)) {
        $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
    }
    // Query ausführen
    if (!$stmt->execute()) {
        $error .= 'execute() failed ' . $mysqli->error . '<br />';
    }
    echo '<script>
        window.onload = function(){
            window.location.href = "index.php";

        }
    </script>';
}
```

Die erstellte Restaurants werden dann auf der index.php Seite angezeigt. Wir erstellen ein Query, das alle erstellte Restaurants von der Datenbank holt. In der foreach-Schleife erstellen wir dann für jedes Restaurant eine card. Wenn der Benutzer eingeloggt ist, kann er auf die Restaurant cards klicken, und durch Anklicken wird der Benutzer zum restaurant.php geleitet. Jedes Restaurant hat seine eigene Seite, dass mit seiner ID aufgerufen wird. Wenn man nicht eingeloggt ist, kommt ein Error.

Dunkin Donuts

Donuts and Drinks
Greifengasse 17, 4058 Basel

View 20-40 Min

```php
<?php
$query = "SELECT * FROM restaurants";

$stmt = $mysqli->prepare($query);

$stmt->execute();

$result = $stmt->get_result();

foreach ($result as $value) {
    if (isset($_SESSION['loggedin']) && $_SESSION['loggedin']) {
        $btn = "<a href='restaurant.php?id={$value['id']}' class='text-warning stretched-link'>";
    } else {
        $btn = " <a class='text-warning stretched-link' onclick='errorLogin()'>";
    }

    echo "<div class='col'>
    <div class='card shadow-sm'>
        <svg class='bd-placeholder-img card-img-top' width='100%' height='230' xmlns='http://www.w3.org/2000/svg' ro
            <title>Placeholder</title>
            <image href='images/mc.jpg' height='100%' width='100%' />
        </svg>

        <div class='card-body'>
            <p class='card-text'>", $value['name'], "</p>
            <p class='card-text'>", $value["description"], "<br>", $value["place"], "</p>
            <div class='d-flex justify-content-between align-items-center'>
            {$btn}View</a>
            <small class='text-muted'>", $value["delivery-from"], "-", $value["delivery-until"], " Min</small>
            </div>
        </div>
    </div>
</div>";
}
```

In restaurant.php gibt es 2 Buttons in der Mitte, einen zum Bearbeiten und einen zum Löschen. Sie aktivieren die Modals in den Dateien include/editRestaurant.php und include/deleteRestaurantModal.php

Edit Restaurant Delete Restaurant

include/editRestaurant.php funktioniert gleich wie include/addRestaurantModal.php. Nur das Query wurde geändert und die Eingabefelder sind schon ausgefüllt.

```
$query = "UPDATE `restaurants` SET `name` = ?, `description` = ?, `place` = ?, `website` = ?, `delivery-from` = ?, `delivery-until` = ? WHERE
`restaurants`.`id` = ?;";
```

Bei include/deleteRestaurantModal.php sieht das Query so aus:

```
$query = "DELETE FROM restaurants WHERE id = {$_POST['deleteRestaurant']};";
```

Kompetenz: In meinem Projekt können diese zusätzlichen Informationen von nicht angemeldeten Benutzern angesehen werden.

Alle erstellten Restaurants können von auch nicht eingeloggten Benutzern auf der index.php Seite gesehen werden. Auf die Restaurants können nur Benutzer gehen. Diese können dort alle Änderungen, sowie neue Gerichte die vom Admin erstellt wurden sehen.

Kompetenz: In meinem Projekt wird SQL-Injection konsequent verhindert.

Beispiel: include/registration.php

Wenn wir etwas in unserer Datenbank Speichern/Ändern wollen besteht die Gefahr von SQL Injection. In unserem Projekt verhindern wir das folgender Massen.

```php
// Query erstellen
$query = "Insert into users (firstname, lastname, username, password, email) values (?,?,?,?,?)";

// Query vorbereiten
$stmt = $mysqli->prepare($query);
if ($stmt === false) {
    $error .= 'prepare() failed ' . $mysqli->error . '<br />';
}

// Parameter an Query binden
if (!$stmt->bind_param('sssss', $firstname, $lastname, $username, $password_hash, $email)) {
    $error .= 'bind_param() failed ' . $mysqli->error . '<br />';
}

// Query ausführen
if (!$stmt->execute()) {
    $error .= 'execute() failed ' . $mysqli->error . '<br />';
}
```

Wir erstellen ein Query. Dort setzen wir jedoch noch nicht die Daten ein. Wir ersetzen diese mit “?”. Danach wird das Query vorbereitet. Danach setzen wir für alle “?” unsere Daten ein, indem wir bei bind_param() zuerst angeben, welche Datentypen unsere Daten haben. In diesem Fall 5 mal s. Das bedeutet 5 Strings. Danach geben wir die Daten in der richtigen Reihenfolge ein. Diese werden als Variablen in das Statement gesetzt und somit können keine weiter befehle ausgeführt werden.
