**Форма за регистрация**

Проектът - форма за регистрация (уеб приложение - client-server) - съдържа следните файлве и папки – images, PHPMailer-master, styles\_and\_script, config.php, crud\_index.php, delete.php, index.php, login.php, logout.php, mail.php, players\_index.php, registration.php, update.php, verify.php. В папката 'images', съм сложил изображенията, които съм използвал за конструиране и задаване на икона на уебстраниците. Папката 'styles\_and\_script' съдържа 'css' и 'js' файлове, за стилизацията на уебстраниците като в тях съм сложил коментари, за по - лесна ориентация в кода.

В 'config.php' установявам връзка между PHP скрипта и MySQL сървърa на базата данни и я запазвам в променливата '$conn', която ще използвам при подаване на заявки към базата данни. Създавам база данни 'user\_accounts' и в нея създавам таблица 'users\_with\_verification', която съдържа следните данни – ид (аутоинкрементира се и епървичен ключ), име, презиме, имейл, телефонен номер, парола и верифициран имейл (по подразбиране е 'NULL', но след успешна верификация стойността му става същата като на имейла на потребителя). Сега ще ви разясня за файловете от тип PHP, за функционалностите, които съм направил, как съм ги реализирал и какви готови функции съм използвал. В началото говоря повече за функциите, които използвам, докато не ги спомена всичките.

'login.php' представлява страница за вход на потребителите в системата. Но ако потребителя вече е логнат не може да влезе директно в нея. В 'login.php' се позволява на потребителите да въведат своя имейл и парола за вход в системата. Файлът се състои от 2 части:

PHP частта се състои от:

* Сесии (session\_start()): Начало на PHP сесията.
* Променлива ($\_SESSION["user"]): - тази променлива е налична във всички обхвати в скрипта и в случаят е масив, който съдържа данните на потребителя.
* Проверка за съществуваща сесия (if (isset($\_SESSION["user"]))): Ако потребителят вече е влезнал, пренасочва към 'index.php'.
* Функцията header("Location: index.php") - препраща потребителя към уебстраницата 'index.php'.
* Проверки и съобщения (<div class="errors">): Показва съобщения, свързани с логин операцията. Включва съобщения за изтриване на профил (изтриването се прави в 'crud.php',но след изтриване на профила потребителя се връща в login.php и се показва съобщение), успешна промяна на имейла (промяната може да се направи в 'crud.php', но потребителя отново се ръша в 'login.php' и се показва съобщение) и за неправилно въведени данни.
* Функция (mysqli\_query($conn, $sql)): изпълнява статична SQL заявка без никакви променливи данни.
* Функция (mysqli\_fetch\_array($result, MYSQLI\_ASSOC)): извлича на редове от базата данни и съхраняването им като асоциативен масив(индексите са имената на отделните колони).
* Извличане на потребител от базата данни: По въведеният имейл се изпълнява SQL заявка, и ако имейлът съществува и паролата съвпада, потребителят се счита за логнат и му се създава PHP сесия.В противен случай се извеждат съответните съобщения за грешка.

HTML частта се състои от:

* Форма за въвеждане на имейл и парола
* Елементи за въвеждане на данни (<input>): Използват се за въвеждане на имейл и парола
* Линк към регистрационна страница (<div class="register\_message">): Показва линк, който пренасочва към страницата за регистрация.

'register.php' представлява страница за регистрация на потребителите. Но ако потребителя вече е логнат не може да влезе в нея, освен ако не цъкне на 'Logout'. В тази страница се позволява на потребителите да въведат необходимата информация за регистрация, като име, фамилия, имейл, телефонен номер и парола, за да се регистрират в системата. Файлът се състои от 2 части:

PHP частта се състои :

* Проверки и съобщения (<div class="errors">): Показва съобщения, свързани с регистрацията. Включва проверки за дължина на паролата, наличие на цифри в имена, валидация на телефонния номер и проверка за уникален имейл.
* Функция (strlen($password)) - връща дължината на низа.
* Функция (preg\_match('~[0-9]+~', $lastName)): търси низа за шаблон и връща true, ако моделът съществува(низа не съдържа нито едно число), в противен случай връща false().
* Функция (preg\_replace("/[^0-9]/", '', $phoneNumber)): връща низ, който е без излишните символи(според шаблона), в случая остават само числата в низа.
* Функция (mysqli\_num\_rows($result)) - връща броя на редовете, присъстващи в резултата.
* Функция (mysqli\_stmt\_init($conn)): инициализира израз и връща обект, подходящ за mysqli\_stmt\_prepare().
* Функция (mysqli\_stmt\_prepare($stmt, $sql)): тази функцията за разлика от mysqli\_query, само подготвя SQL заявка за изпълнение, като може да използвате маркери за параметри ("?"),а по-късно
* да посочите стойности за тях и да я изпълните.
* Функция (mysqli\_stmt\_bind\_param()): за свързване на променливи към маркерите за параметри ("?").
* Функция (mysqli\_stmt\_execute($stmt)): изпълнява само заявка, която е била подготвена с помощта на функцията mysqli\_stmt\_prepare(). Когато се изпълни, всички маркери на параметри, които съществуват, автоматично ще бъдат заменени със съответните данни.
* Извличане на потребител от базата данни: По въведеният имейл се изпълнява SQL заявка, и ако имейлът вече съществува в базата данни, се добавя съобщение за грешка. Ако няма грешки, потребителят се вписва в базата данни.

HTML частта се състои от:

* HTML форма (<form>): Форма за въвеждане на информация за регистрация.
* Елементи за въвеждане на данни (<input>): Използват се за въвеждане на име, фамилия, имейл, телефонен номер и парола.
* Линк към страницата за вход (<div class="login\_message">): Показва линк, който пренасочва към страницата за вход.

В 'index.php' се намира главната страница на сайта. Съдържанието се отнася за Манчестър Юнайтед, а дизайна на страницата е сравнително прост и включва различни елементи . Като този файл и файла 'players\_index.php' съм ги разработвал преди време (тогава се упражнявах да правя уебстраница, без flexbox и затова съм използвал flexbox само където съм променял кода наскоро), за проекта добавих навигационното меню в заглавието, с линкове към други страници (като линка за верификация се скрива, след като потребителя верифицира имейла си). Но, за да може потребителя да влезе в главната страница, първо трябва да се логне ,а ако няма регистрация, следователно трябва да се регистрира и след това да се логне. Използвал съм 'JavaScript' за таблицата и за функционалността на бутоните - 'button\_stadium' и 'button\_back'.Нова функция, която съм използвал PHP частта е:

* Функцията check\_verified(): Проверява дали имейлът на потребителя е потвърден в базата данни. Връща True, ако е потвърден, и False в противен случай.

В HTML частта се състои от:

* Хедър (<header>): Съдържа логото на клуба, заглавието на страницата и навигационните линкове.
* Секция с колони (<div class="row">): Разделя страницата на три колони.
* Лява колона (<div class="col-1 menu">): Съдържа меню с линкове и допълнителни елементи като таблица с мачове за месеца, списък с играчи, бутон за стадион и секция за трофеи.
* Централна колона (<div class="col-2">): Съдържа секция с историята на отбора.
* Дясна колона (<div class="col-3">): Съдържа информация за лични предпочитанията - защо съм избрал този отбор, кой е любимият ми играч и колко фенове има отборът.
* Футър (<footer>): Включва бутон за връщане, който е скрит, но се показва при натискане на бутона за стадион(в менюто), и линк за закупуване на билети.

'players\_index.php' представлява страница с информация за футболистите на Манчестър Юнайтед. Показва снимки и имената на футболистите, разделени в категории: "Вратари", "Защитници", "Полузащитници" и "Нападатели". Като и тази страница не е достъпна за потребителя, ако не е логнат, а HTML частта съдържа:

* HTML заглавие и хедър (<header>): Съдържа заглавие и информация за сезонът.
* HTML секция (<main>): Съдържа информация за различните категории на футболистите, разделени в колони с картинки и имена.
* HTML картинки и текст (<div class="column">): Колони, представляващи различни футболисти с техните снимки и имена, разделени в категории.

'crud.php' представлява CRUD страница, където потребителите могат да видят, да редактират и да изтрият своите данни в таблица (данните са извлечени от базата данни и се показват в таблата). PHP частта се състои от:

* Извличане на данните от базата данни за текущия потребител (SELECT \* FROM users\_with\_verification WHERE id = '$id'). После запаметява тези данни в масив и създава таблицата с тези стойности.
* Показване на съобщенията при редактиране на данни.

HTML часта съдържа:

* Заглавие (<header>): Заглавие и бутони за редактиране и изтриване.
* Модален прозорец за редактиране (<div id="myModal" class="modal">): Форма, която позволява на потребителя да редактира своите данни. Тя се появява при натискане на бутона 'Edit Data' като минава пред останалите елементи от страницата и затъмнява фона извън формата. За да се затвори трябва да се натисне символа 'x' в горния десен ъгъл. При изпращане на формата, данните се изпращат към 'update.php'.
* Таблица с данни (<table>): Показва данните на потребителя в табличен формат..
* Футър (<footer>): Показва съобщения за успешно обновяване и възможни грешки при опит за редактиране или изтриване на данни.

В 'update.php' се извършва редактиране на потребителските данни от таблицата 'users\_with\_verification' в базата данни, в отговор на формата за редактиране от 'crud\_index.php'. Състои се само от PHP част:

* Обработка на формата за редактиране, новите данни се валидират и ако не са въведени правилно се добавят като съобшения (низове в масива ('$\_SESSION["errors"]')) и потребителят се пренасочва към страницата за редактиране (crud\_index.php),където се показват съобщения с грешките(низове запаметени в масива $\_SESSION["errors"]).
* Използване на ID за потребителя от текущата сесия ($id = $\_SESSION["user"]["id"];)
* Ако не дава грешка има два варианта:

1. имейлът е променен, потребителската сесия се унищожава и потребителя се пренасочва към страницата за вход със съобщение, че имейла е редактиран.
2. Променените данни, не включват имейла и потребителят се пренасочва към страницата с данни 'crud\_index.php' със съобщение за успешно редактиране.

В 'delete.php' се извършва изтриване на потребителския акаунт от таблицата 'users\_with\_verification' в базата данни и се унищожава сесията. Този файл се извиква пи натискане на бутона за изтриване в CRUD страницата. Файлът се състои само от php код:

* Използване на ID потребителя от текущата сесия ($id = $\_SESSION["user"]["id"];)
* потребителската сесия се унищожава, потребителя се изтрива от базата с данни, пренасочване към 'login.php' и се показва съобщение, че акаунта е изтрит.

За верифицирането на акаунта създавам нова таблица 'verify' в базата данни, съдържаща 4 данни – ид(ауто инкрементира се и е първичен ключ),код за верификация(случайно генерирано 5 цифрено число и ), времетраене на кода(3 минути) и имейл на потребителя. В 'verify.php' кода реализира система за потвърждаване на имейл адреса, като изпраща верификационен код на потребителя по имейл от моя личен имейл(трябва да е 'gmail'). Но преди да изпратя кода, го записвам в таблицата 'verify'. Изпращането на кода чрез имейл става чрез функцията 'send\_mail($recipient,$subject,$message)', която е описана в 'mail.php', като в файла са включени файловете от папката 'PHP-Mailer-master', взех файловете от интернет като в коментар в 'mail.php' съм оставил линк към видеото и часа и минутите ,в които се обяснява за тези файлове. Без тази функция потребителя не може да види своя код. След това описвам условия за валидиране на кода. Ако той е валиден, не е изтекъл и имейлът не е верифициран, се отбелязва имейлът на потребителя като потвърден(в таблицата 'users\_with\_verification' в даната 'email\_verified' се записва текущия имейл на потребителя,). Ако ли не, се извежда съобщение за грешка и при рефрешване на страницата се изпраща нов код. Ако потребителя смени имейла, трябва отново да го верифицира.HTML частта на файла включва: форма за въвеждане на верификационен код, елементи за извеждане на информационни съобщения и допълнителни HTML елементи (бутон за верифициране и линк към 'index.php').