ЗВІТ З ЛАБОРАТОРНИХ РОБІТ №3-4

«Обробка форм. Написання PHP-скрипта. Робота з базою даних»

з курсу «Програмування та підтримка web-застосувань»

студента групи ПА-18-3

Кондакова Гліба Владиславовича

кафедра комп’ютерних технологій, ДНУ

2020/2021 н.р.

**Опис завдання**

Створити гіпертекстову сторінку з формою, куди помістити не менше 10 доступних елементів, які дозволяють введення даних. Написати CGI-скрипт, за допомогою якого забезпечити контроль даних перед відправкою їх на сервер.

Для обробки даних, які переслані на сервер, створити CGI-скрипт на обраній мові. Розкодовані дані повернути клієнтові.

Написати PHP-скрипт для авторизації клієнта. Дані про клієнтів зберігати в таблиці MySQL або SQLITE. Для розробки бази даних можна використати PHPMyAdmin.

**Опис кроків виконання**

Для початку виконання лабораторного завдання необхідно мати доступ до веб-сервера, за допомогою якого ми зможемо запустити сайт в себе на комп'ютері, підключити до нього MySQL і відповідно PHP, а також працювати з базою даних. Задовільнити всі ці потреби може WampServer – спеціальне середовище для веб-розробки на Windows, яке постачається з PHPMyAdmin і SQLite Manager для того, щоб максимально легко керувати власними базами даних.

Для ще більшої ефективності завантажимо додаткову бібліотеку ReadBeanPHP, яка надасть можливість дистанційно створювати таблиці, поля та індекси у вже існуючій в PHPMyAdmin базі даних. Щоб не повторювати один і той самий код в різних виконавчих файлах, пропишемо підключення бібліотеки (вона знаходиться в папці libs) та її зв’язок з базою в окремому файлі db.php, який надалі будемо підключати за необхідності через вираз require:

<?php

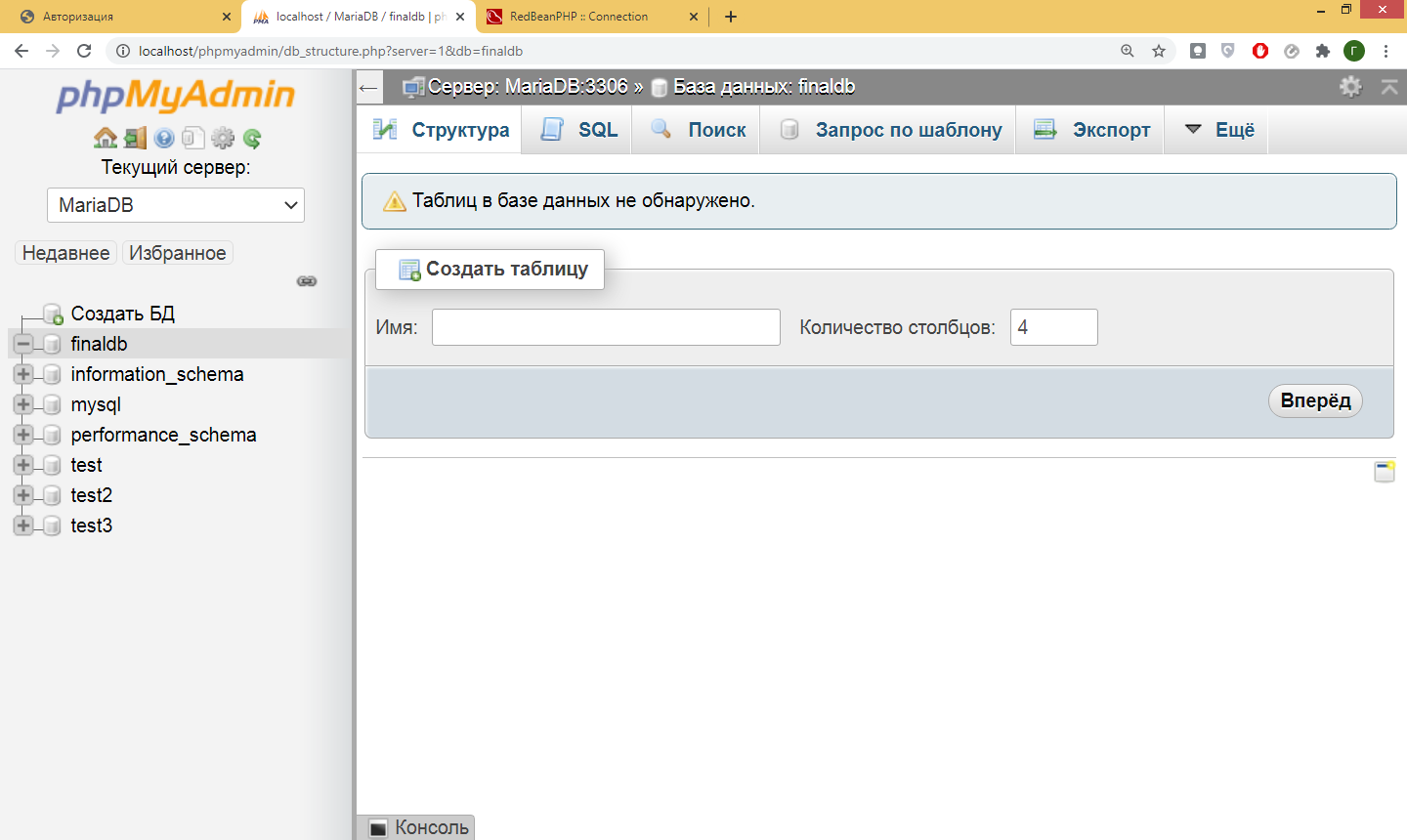
require "libs/rb.php"; // підключення зазделегідь завантаженої бібліотеки

R::setup( 'mysql:host=localhost;dbname=finaldb',

'root', '' ); // з’єднання с базою даних MariaDB

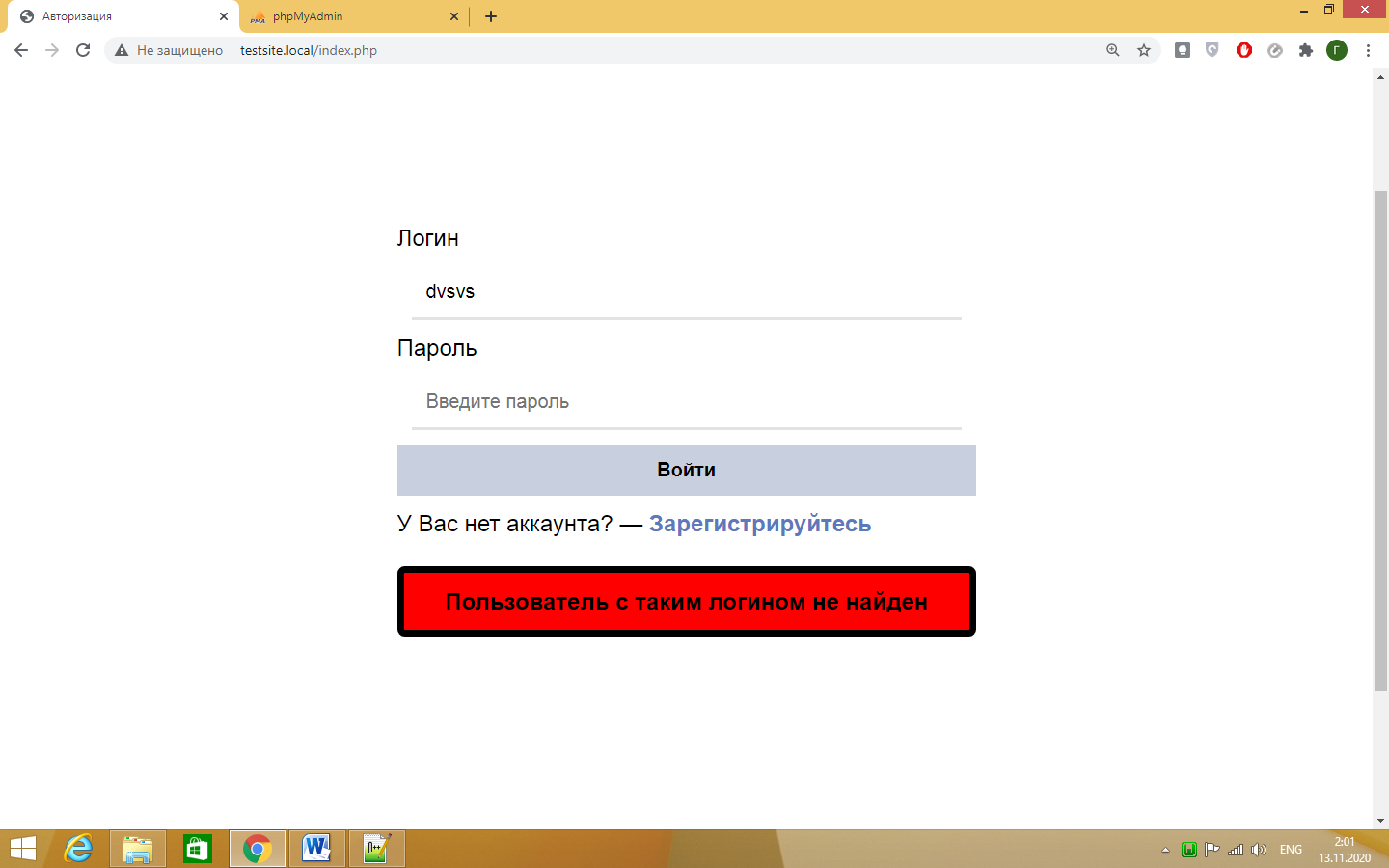
session\_start(); // починаємо сесію, щоб мати доступ до суперглобальної змінної $\_SESSION, яка насправді є асоціативним масивом

?>

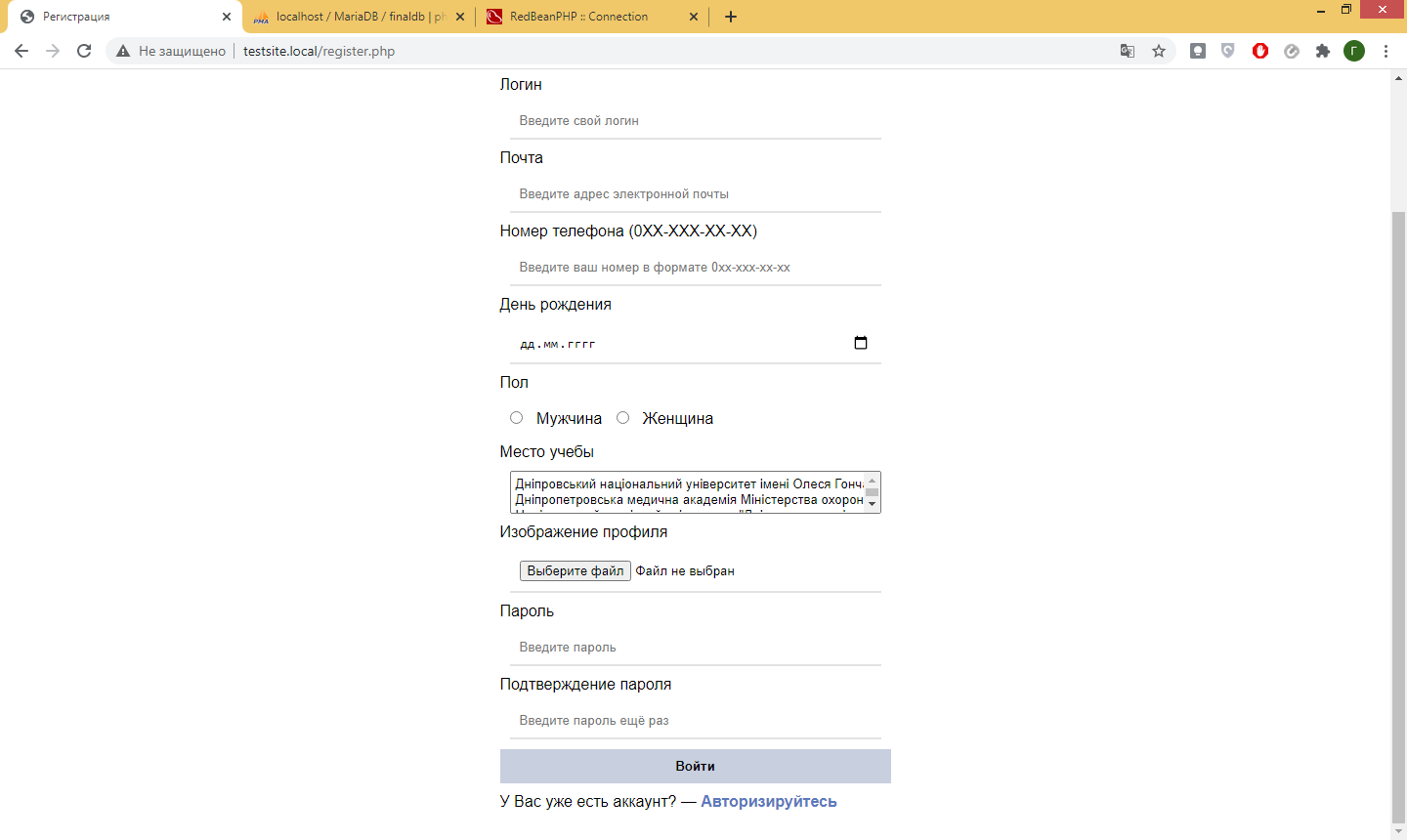


*Як бачимо, база даних створена, але таблиць в ній немає. Завдяки ReadBeanPHP нам не потрібно створювати їх в ручну (тобто безпосередньо через сайт)*

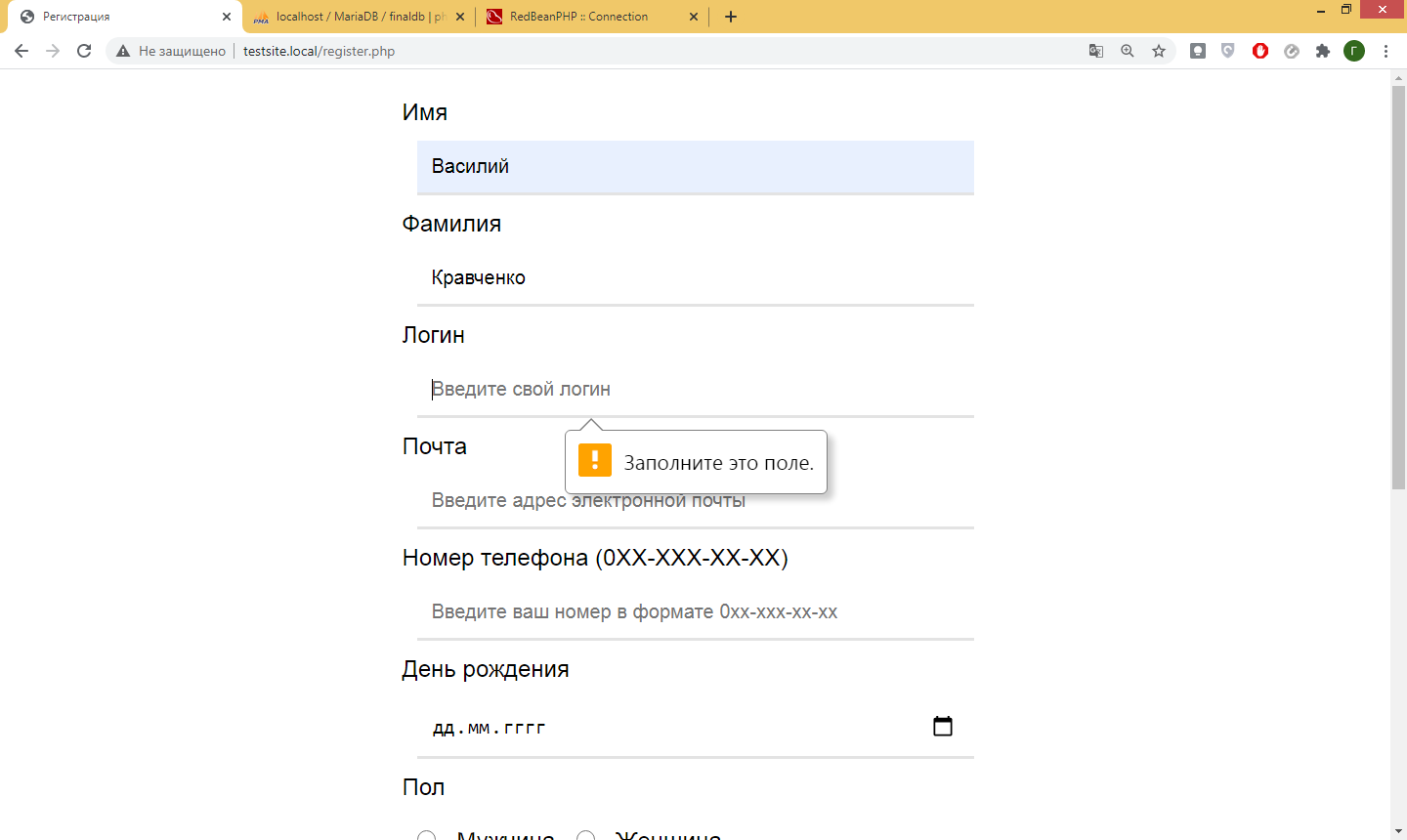
Переходимо до файлу index.php, де за допомогою тегу form створюємо сторінку авторизації користувача (йому необхідно ввести логін та пароль, які були вказані при реєстрації). Якщо в базі даних не знайдено користувача з введеним логіном або був вказаний хибний пароль, на екрані з’являється повідомлення про помилку.

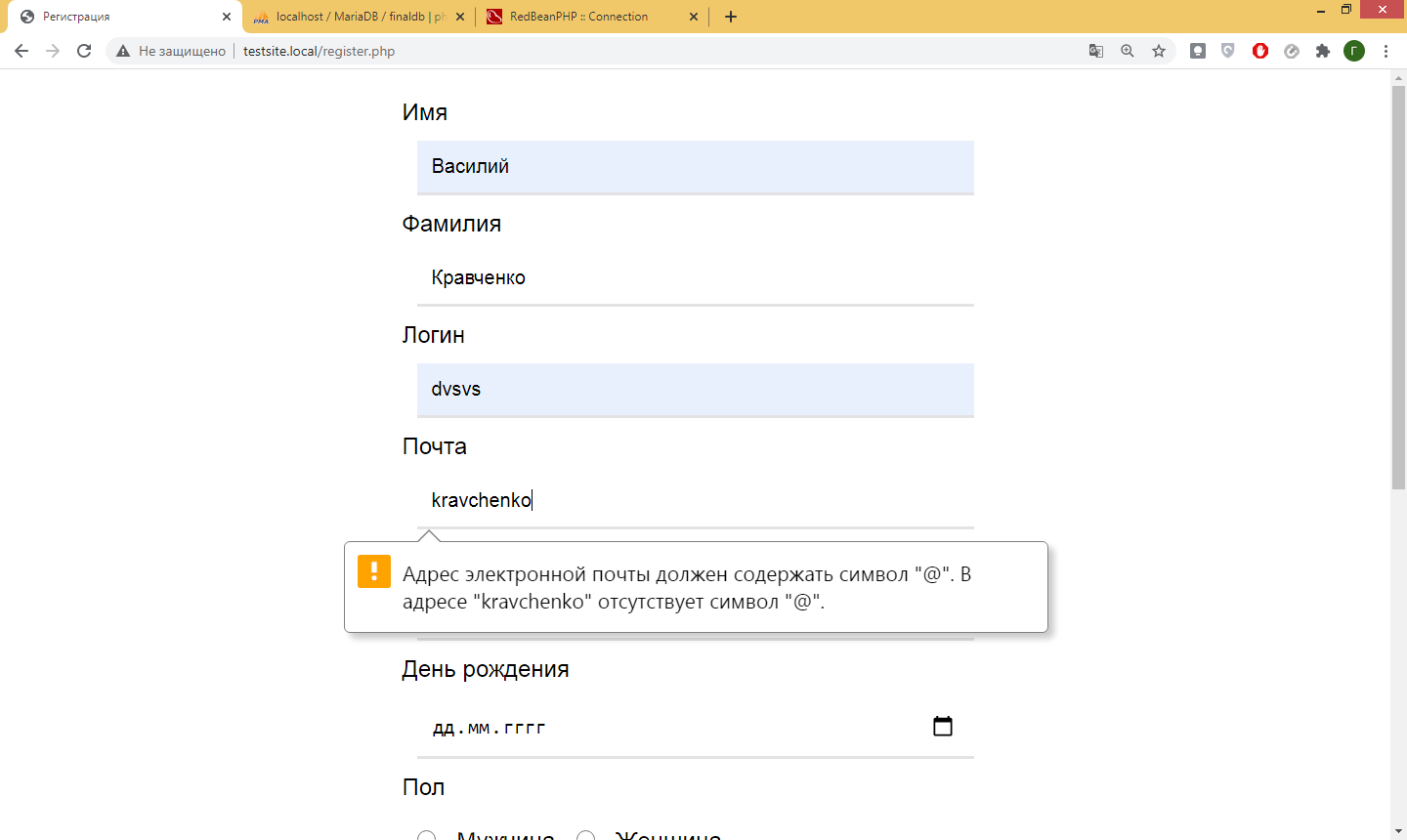


Авторизуватися не вдалося адже в базі даних ще немає жодного запису, тож починаємо процес реєстрації користувача, перейшовши за зручним посиланням зі сторінки index.php на сторінку register.php Тут нас зустрічає ще одна форма, але вже не така мінімалістична: користувачеві необхідно заповнити 11 полів вводу даних 8 різних типів (text, email, tel, date, file, password, radio та розкривний список).

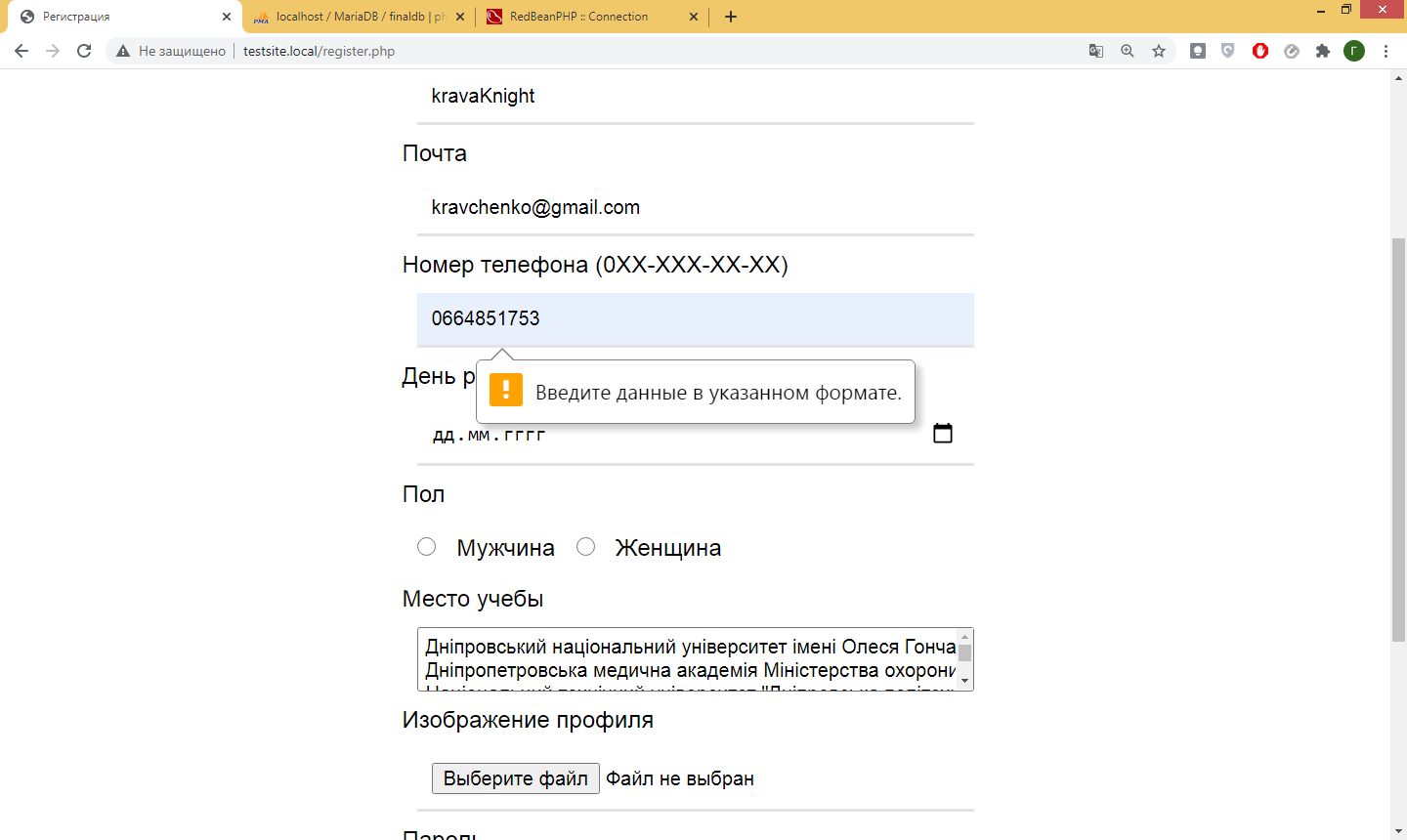


За наявності атрибуту required форма не буде надіслана на сервер доти, доки всі поля не будуть заповнені належним чином, про що користувачеві нагадає браузер у спливаючему повідомленні. Ось декілька прикладів:





*<input type = ‘email’ required>*



*Це повідомлення з’явилось, бо в атрибуті pattern був наведений шаблонний вираз, якому повинні відповідати введенні данні*

Але не за всі перевірки на коректність відповідає браузер. Пароль та його підтвердження порівнюємо самостійно в обробнику:

if ($data['password\_confirm'] != $data['password']) {

$\_SESSION['message'] = "Пароли не совпадают";

}

де $data – копія всіх даних, відправлених HTML-формою методом post після успішного натиснення на кнопку submit. До речі, якщо користувачеві з першої спроби не вдалось правильно повторити пароль, текстові поля непотрібно заповнювати ще раз, вони зберігають попередні дані через атрибут value = "<?php echo @$data['name'] ?>"

Після підтвердження паролю в обробнику ми створюємо змінну, яка буде безпосередньо пов’язана з автоматично створеною таблицею users: $user = R::dispense('users');

Зараз ми маємо порожню таблицю типу user з id 0. Додамо декілька методів, які будуть зберігати відповідні дані з масиву $data. Наприклад:

$user->name = $data['name'];

$user->surname = $data['surname'];

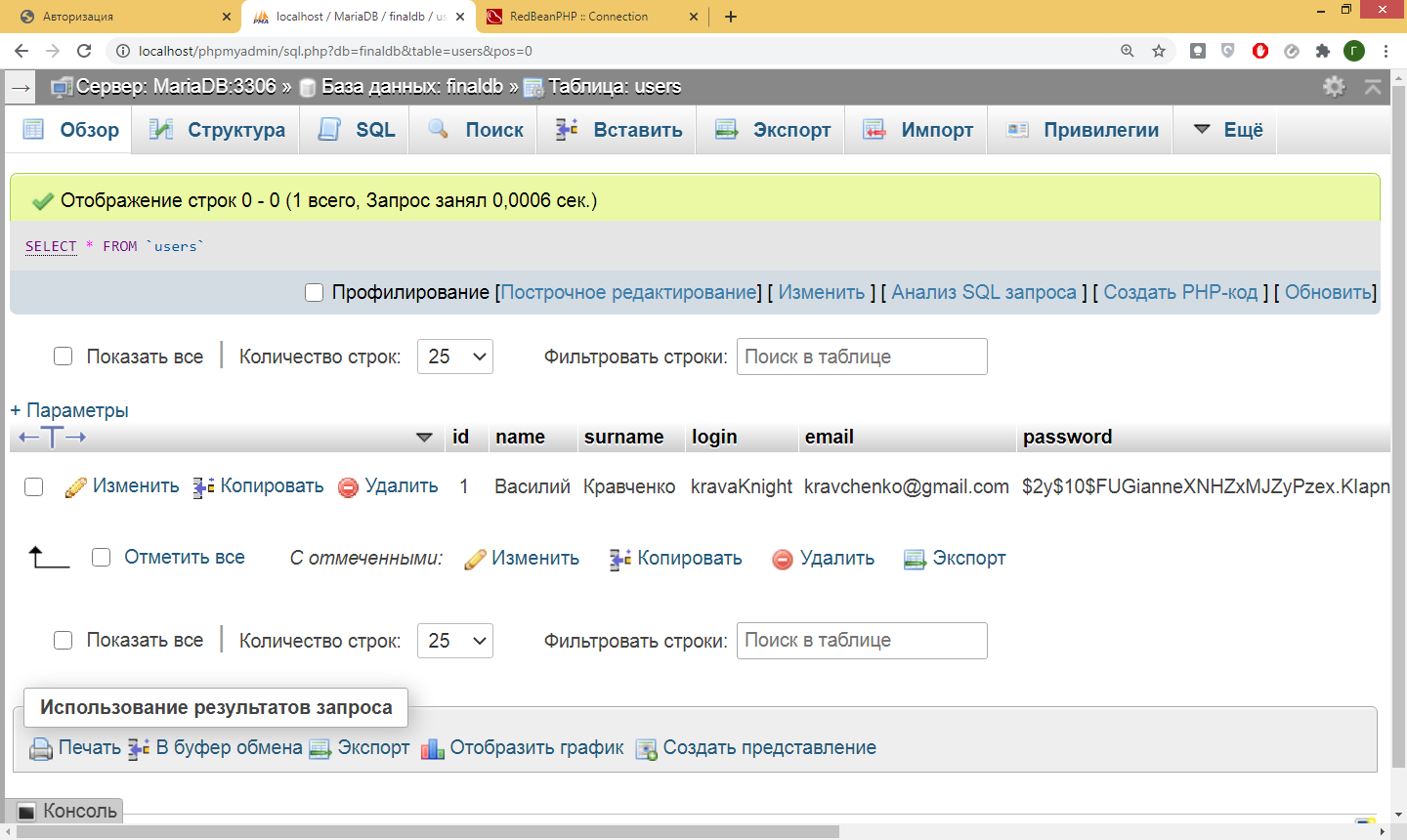
$user->login = $data['login'];

$user->email = $data['email'];

$user->password = password\_hash($data['password'], PASSWORD\_DEFAULT);

Збережемо внесені зміни: R::store($user);

Ось і все, в нашій таблиці з’явилися стовпці з ім’ям, введеним користувачем, його логіном, почтовою адресою і зашифрованим паролем (зберігати у відкритому вигляді не доцільно с точки зору безпеки). І для цього навіть не потрібно було залишати редактор коду, ReadBeanPHP самостійно створив потрібні записи в нашій базі, про що свідчить скріншот нижче.



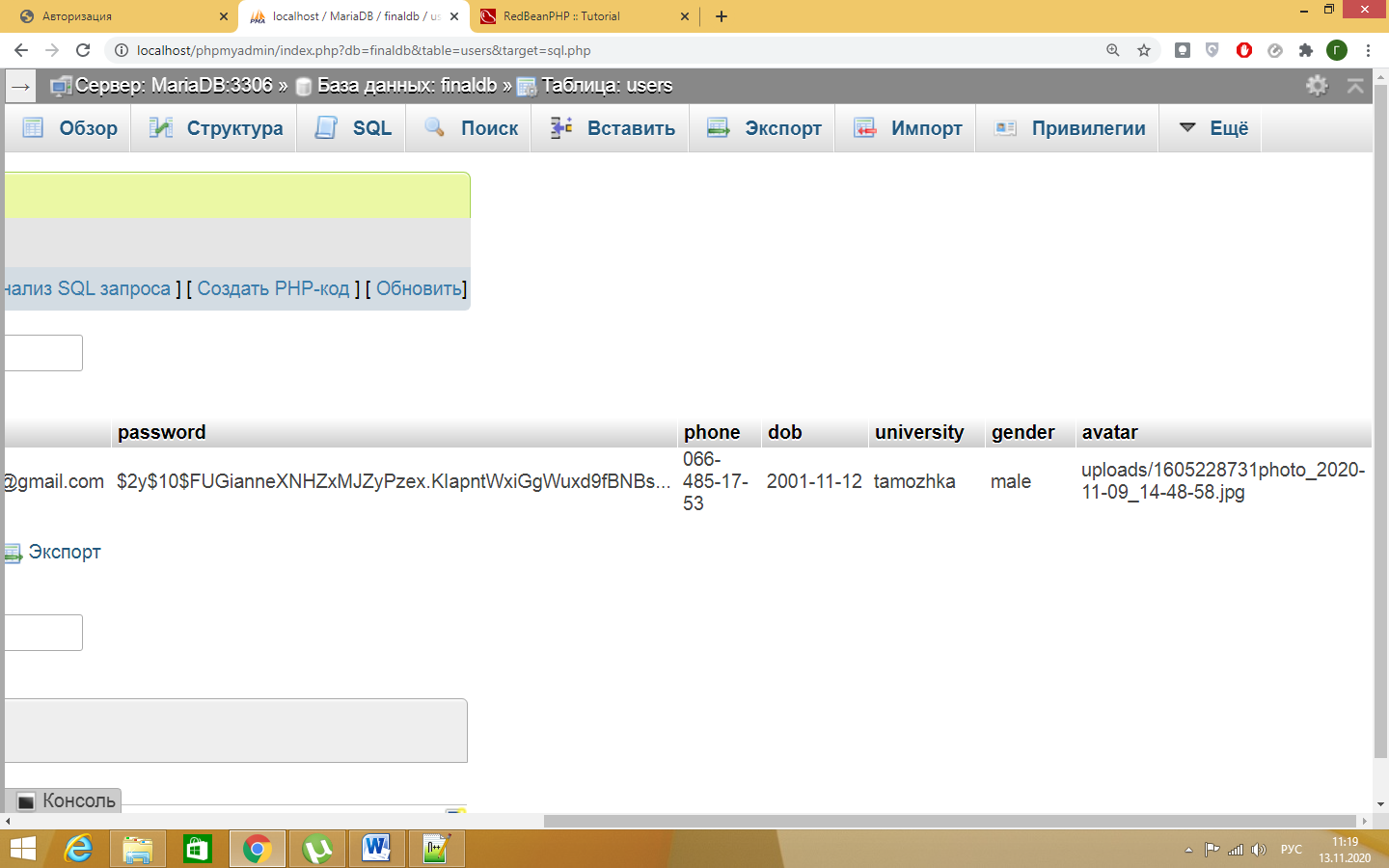
Також варто зазначити, що нові стовпчики додаються динамічно, якщо з’являються нові методи:

$user->phone = $data['phone'];

$user->dob = $data['dob'];

$user->university = $data['university'];

$user->gender = $data['gender'];



Отже, один користувач в нас точно зареєстрований, спробуємо авторизуватися. Для цього повертаємось на сторінку index.php Спочатку аналогічно сторінці реєстрації присвоюємо змінній $data відправлені на сервер дані форми, після чого вона стає асоціативним масивом з ключами, які відповідають стовпцям попередньо створеної та заповненої таблиці users.

За допомогою функції findOne перевіряємо, чи існує в таблиці запис з тим логіном, який користувач ввів під час спроби авторизуватися: $user = R::findOne('users', 'login = ?', array($data['login'])); Якщо такий запис існує, ініціалізуємо змінну user та порівнюємо вказаний в таблиці пароль (user->password) з нововведеним (($data['password']):

if (password\_verify($data['password'], $user->password)) {

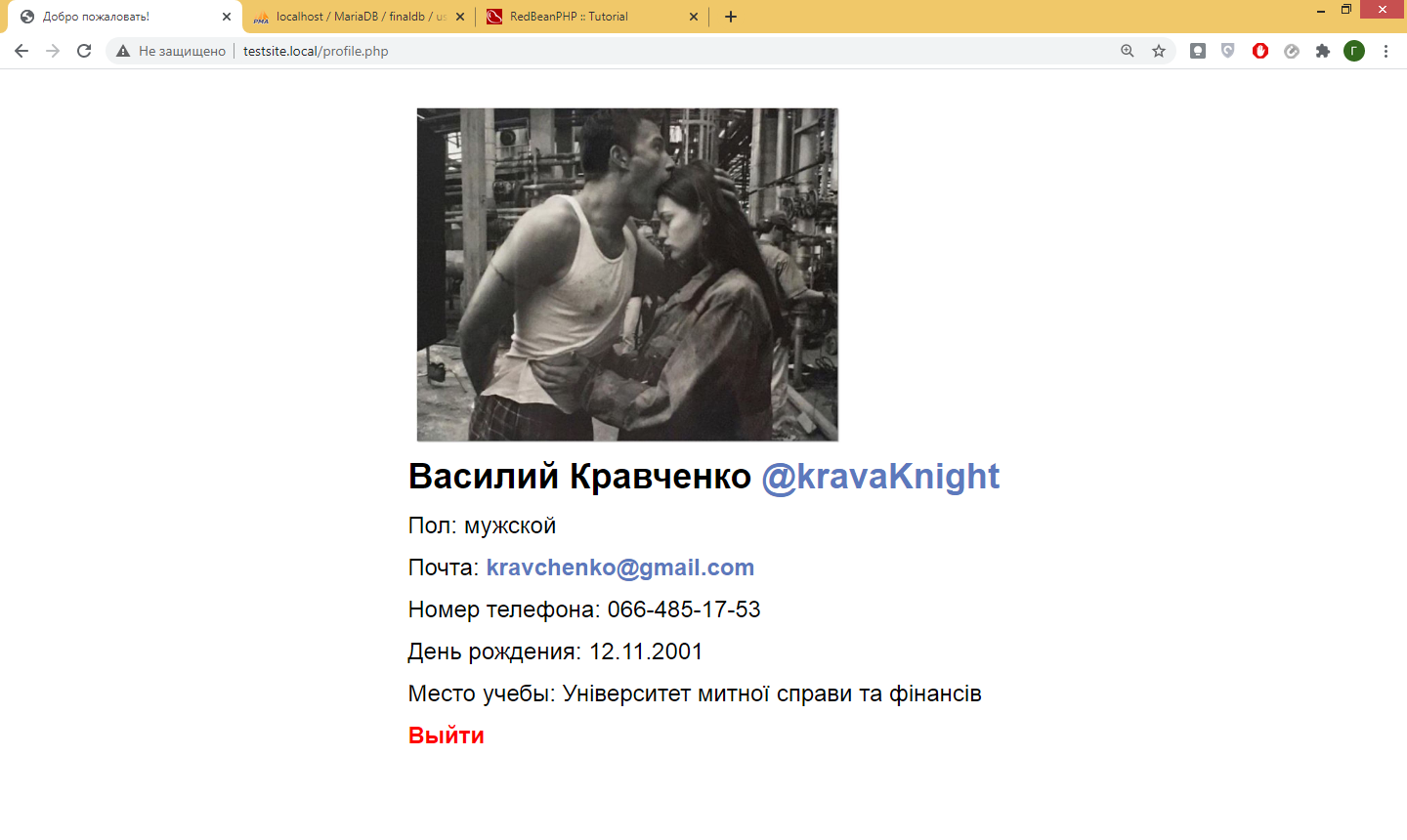
$\_SESSION['logged\_user'] = $user; // у разі збігу паролів, запам’ятовуємо нашого користувача у всіх поточних сесіях

header ('Location: profile.php'); // перенаправляємо на сторінку профіля, де нас чекатимуть розкодовані дані

}

**Результати виконання**

У профілі повертаємо всі введені під час реєстрації дані, окрім, зрозуміло, паролю:



Отримані з серверу значення полів dob (день народження), university та gender попередньо форматуємо в спеціальних функціях для коректного відображення інформації. Наприклад:

function usergender ($ug) {

switch ($ug) {

case 'male':

$ug = "мужской";

break;

case 'женский':

$ug = "женский";

}

return $ug;

}

При виклику функції передаємо їй одне з двох можливих значень поля radio button: male або female

<p> <?= "Пол: " . usergender($\_SESSION['logged\_user']['gender']) ?> </p>

Зазначимо, що після авторизації користувачеві недоступні сторінки index.php та register.php. При спробі зайти на них його автоматично перенаправляє на сторінку профілю, внизу якої є спеціальна кнопка для виходу з сесії.