МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Волжский политехнический институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Волгоградский государственный технический университет»

(ВПИ (филиал) ВолгГТУ)

|  |  |
| --- | --- |
| Факультет | Инженерно-экономический факультет |
| Кафедра | Информатика и технология программирования |

##### **Контрольная работа**

|  |  |
| --- | --- |
| по дисциплине: | Основы проектирования WEB приложений |

|  |  |
| --- | --- |
| Студент | Сергеев Андрей Игоревич |
|  | (фамилия, имя, отчество) |

|  |  |
| --- | --- |
| Группа | ВВТ-406 |
|  |  |

|  |  |
| --- | --- |
| Оценка |  |
|  | (баллы) |

|  |  |  |  |
| --- | --- | --- | --- |
| Проверил |  |  | Белоусов И.С. |
|  | (подпись и дата подписания) |  | (долж., фамилия и инициалы) |

|  |  |  |  |
| --- | --- | --- | --- |
| Специалист по УМР |  |  | Инкарбекова М.Р. |
|  | (подпись и дата подписания) |  | (фамилия и инициалы) |

Волжский, 2024 г

***Тема:*** Программирование интерфейсов работы с данными с применением паттерна TableModule.

***Задание:*** реализовать интерфейсы для таблицы с данными из ЛР 2.

***Интерфейсы:***

* Добавление записи в таблицу
* Редактирование записи (в том числе замена изображения, старое обязательно удалять)
* Валидация в полном объеме - и клиентская, и серверная
* Удаление записи, изображение обязательно удалять

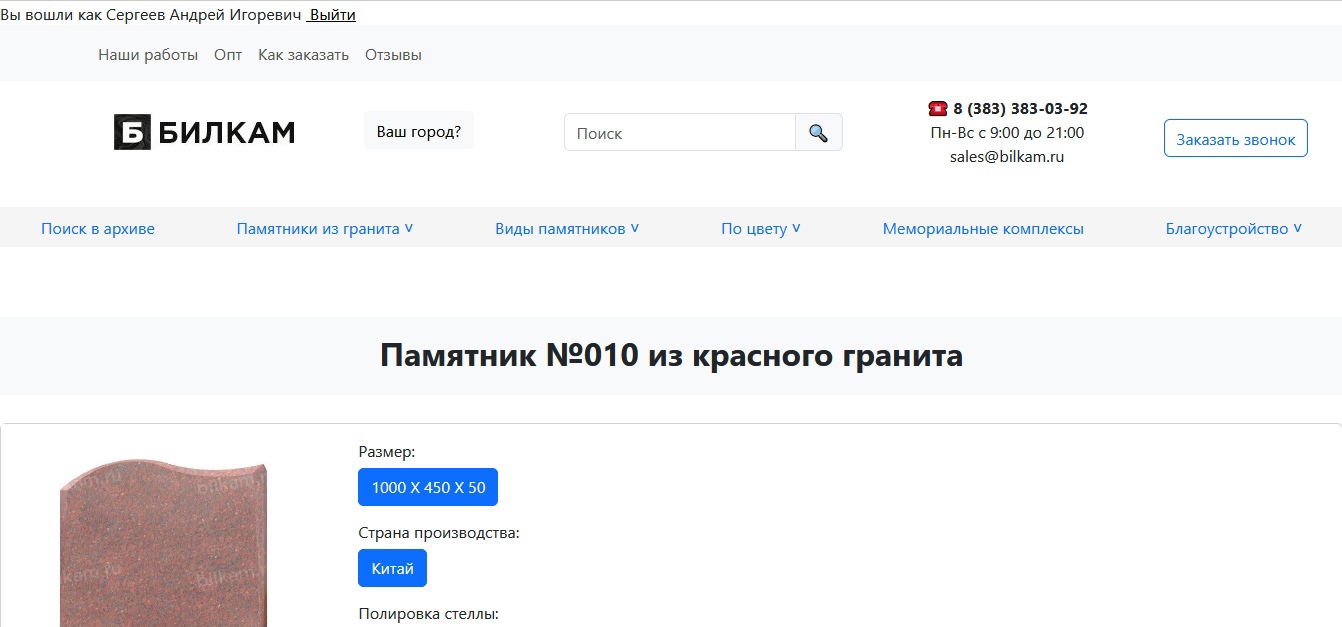


Рис.1: Главная страница сайта

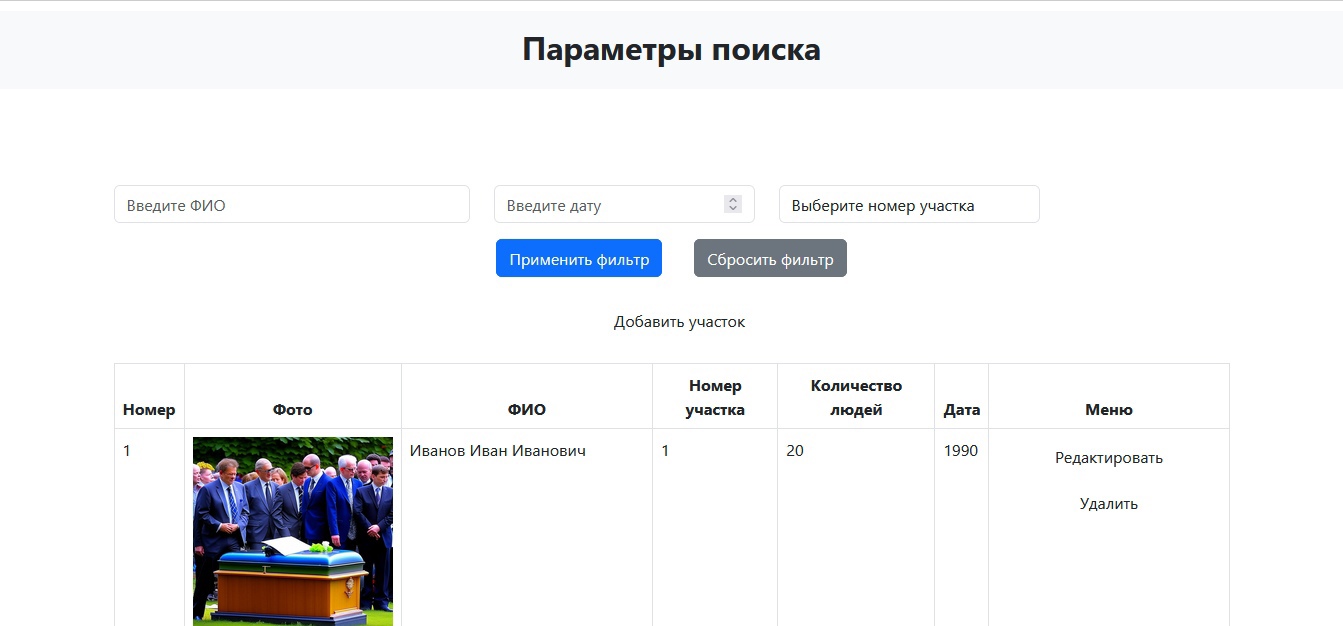


Рис.2: Страница с данными из базы данных

***Подпункты, необходимые для реализации:***

Кладбища, участки, виды памятников, похороны, бригады могильщиков, залы для поминок.

***ТЗ:***

Поля для похорон:

* фото с похорон
* фио усопшего
* участок кладбища (хранить номер в таблице участков)
* дата и время похорон текстом
* число присутствовавших

Требования к паролю: длиннее 6 символов, обязательно содержит большие латинские буквы, маленькие латинские буквы и цифры. Русские буквы запрещены. Дополнительные поля пользователя: ФИО, группа крови, резус-фактор.

***Листинг программ:***

User\_action.php

<?php

class UserActions

{

    // метод для авторизации

    public static function signIn() {

        if ('POST' != $\_SERVER['REQUEST\_METHOD']){

            return [];

        }

        $action = isset($\_POST['action']) ? $\_POST['action'] : '';

        if('signIn' != $action){

            return[];

        }

        $errors = UserLogic::signIn($\_POST['email'] ?? '', $\_POST['password'] ?? '');

        if (!count($errors)){

            header('Location: ' . $\_SERVER['PHP\_SELF'] . '?success=y');

            die();

        }

        return $errors;

    }

    // метод для регистрации

    public static function signUp()

    {

        if ('POST' != $\_SERVER['REQUEST\_METHOD']){

            return [];

        }

        $action = isset($\_POST['action']) ? $\_POST['action'] : '';

        if('signup' != $action){

            return[];

        }

        $errors = UserLogic::signUP(

                $\_POST['email'] ?? '', $\_POST['password'] ?? '', $\_POST['twoPassword'] ?? '', $\_POST['full\_name'] ?? '', $\_POST['date'] ?? '',

                $\_POST['address'] ?? '', $\_POST['gender'] ?? '', $\_POST['interests'] ?? '', $\_POST['link'] ?? '', $\_POST['blood\_group'] ?? '',

                $\_POST['rh\_factor'] ?? '');

            if (!count($errors)){

                header('Location: ' . $\_SERVER['PHP\_SELF'] . '?success=y');

                die();

            }

        return $errors;

    }

}

?>

User\_logic.php

<?php

class UserLogic

{

    public static function signUp( ?string $email, ?string $password, ?string $twoPassword, ?string $full\_name, ?string $date, string $address, string $gender, string $interests, string $link, string $blood\_group, string $rh\_factor, ) : array {

        $errorMessage = ['success' => false, 'message' => ' '];

        // Проверка email на валидность

        if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

            $errorMessage['message'] .= 'Введенный Email не является верным';

        }

        $user = UserTable::get\_by\_email($email);

        if (!null == $user){

            $errorMessage['message'] .= '<br>' . 'Пользователь с таким Email уже существует';

        }

        // Проверка на требования к паролю

        if (!(

            strlen($password) > 6 &&

            preg\_match('/[A-Z]/', $password) && // Большие латинские буквы

            preg\_match('/[a-z]/', $password) && // Маленькие латинские буквы

            preg\_match('/[!@#$%^&\*()\-\_+=\[\]{};:\'",.<>\/?\\| ]/', $password) && // Спецсимволы и пробел

            preg\_match('/[0-9]/', $password) && // Цифры

            !preg\_match('/[А-Яа-я]/u', $password) // Русские буквы запрещены

            )) {

                $errorMessage['message'] .= "<br>" .  'Пароль должен соответствовать требованиям: <br>

                \* длиннее 6 символов, <br>

                \* большие латинские буквы, <br>

                \* маленькие латинские буквы, <br>

                \* спецсимволы (знаки препинания, арифметические действия и тп),  <br>

                \* пробел,  <br>

                \* дефис,  <br>

                \* подчеркивание и цифры,  <br>

                \* pусские буквы запрещены. <br> ';

            }

        // Проверка на совпадение паролей

        if ($password !== $twoPassword) {

            $errorMessage['message'] .= "<br>" . 'Пароли не совпадают.';

        }

        // Проверка ФИО

        if(!isset($full\_name) || empty($full\_name)){

            $errorMessage['message'] .= "<br>" . 'Введите ФИО';

        }

        // Проверка даты

        if(!isset($date) || empty($date)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода даты';

        }

        // Проверка адреса

        if(!isset($address) || empty($address)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода адреса';

        }

        // Проверка гендера

        if(!isset($gender) || empty($gender)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода гендера';

        }

        // Проверка интересов

        if(!isset($interests) || empty($interests)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода интересов';

        }

        // Проверка социальной сети

        if(!isset($link) || empty($link)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода ссылки на социальную сеть';

        }

        // Проверка группы крови

        if(!isset($blood\_group) || empty($blood\_group)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода группы крови';

        }

        // Проверка резус фактора

        if(!isset($rh\_factor) || empty($rh\_factor)){

            $errorMessage['message'] .= "<br>" . 'Ошибка ввода резус факторв';

        }

        if($errorMessage['message'] !== ' '){

            return $errorMessage;

        }

        // Защита от SQL-инъекций

        $email = htmlspecialchars($email);

        $password = htmlspecialchars($password);

        $full\_name = htmlspecialchars($full\_name);

        $address = htmlspecialchars($address);

        $interests = htmlspecialchars($interests);

        $link = htmlspecialchars($link);

        // Вызов метода create из UserTable

        $result = UserTable::create($email, $full\_name, $password, $date, $address, $gender, $interests, $link, $blood\_group, $rh\_factor);

        $query = $result->fetchAll();

        if(!count($query)){

            return [];

        }

        // Возвращение результата

        return $query;

    }

    public static function signIn(string $email, string $password) : array

    {

        $errorMessage = ['success' => false, 'message' => ' '];

        if (static::isAuthorized()){

            $errorMessage['message'] .= "<br>" . 'Вы уже авторизованы';

        }

        $userIN = UserTable::get\_by\_email($email);

        if (null == $userIN){

            $errorMessage['message'] .= "<br>" . 'Логина нет в системе';

        }

        if(null !== $password && null !== $userIN){

            if(password\_verify($password, $userIN['Password']) != $userIN['Password']){

                $errorMessage['message'] .= "<br>" . 'Неверный пароль';

            }

        }

        else {

            if(null !== $userIN)

                $errorMessage['message'] .= "<br>" . 'Пароль не введен';

        }

        if($errorMessage['message'] !== ' '){

            return $errorMessage;

        }

        $\_SESSION['USER\_NAME'] = $userIN['FIO'];

        $\_SESSION['USER\_ID'] = $userIN['id'];

        return [];

    }

    public static function signOut()

    {

        unset($\_SESSION['USER\_ID']);

    }

    public static function isAuthorized() : bool {

        return isset($\_SESSION['USER\_ID']) && intval($\_SESSION['USER\_ID']) > 0;

    }

}

?>

User\_table.php

<?php

class UserTable{

    public static function create(string $email, string $full\_name, string $password, string $date\_of\_birth, string $address, string $gender, string $interests, string $link, string $blood\_group, string $rh\_factor) {

        // Хэшируем пароль

        $hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

        $query = Database::prepare('INSERT INTO users (`Login`,`FIO`,`Password`,`DateOfBirth`,`Address`,`Gender`,`Interests`,`Link`,`BloodType`,`RhFactor`) VALUES (:email, :full\_name, :password, :date\_of\_birth, :address, :gender, :interests, :link, :blood\_group, :rh\_factor );');

        //привязываем параметры

        $query->bindParam(':email', $email);

        $query->bindParam(':full\_name', $full\_name);

        $query->bindParam(':password', $hashedPassword);

        $query->bindParam(':date\_of\_birth', $date\_of\_birth);

        $query->bindParam(':address', $address);

        $query->bindParam(':gender', $gender);

        $query->bindParam(':interests', $interests);

        $query->bindParam(':link', $link);

        $query->bindParam(':blood\_group', $blood\_group);

        $query->bindParam(':rh\_factor', $rh\_factor);

        if(!$query->execute()){

            throw new PDOException('При добавлении пользователя возникла ошибка');

        }

        $id = UserTable::get\_by\_email($email);

        $\_SESSION['USER\_NAME'] = $id['FIO'];

        $\_SESSION['USER\_ID'] = $id['id'];

        return $query;

    }

    public static function get\_by\_email(string $email) : ?array{

        $query = Database::prepare('SELECT \* FROM `users` WHERE `Login` = :email LIMIT 1');

        $query->bindParam(':email', $email);

        $query->execute();

        $users = $query->fetchAll(PDO::FETCH\_ASSOC);

        if(!count($users)){

            return null;

        }

        return $users[0];

    }

}

?>