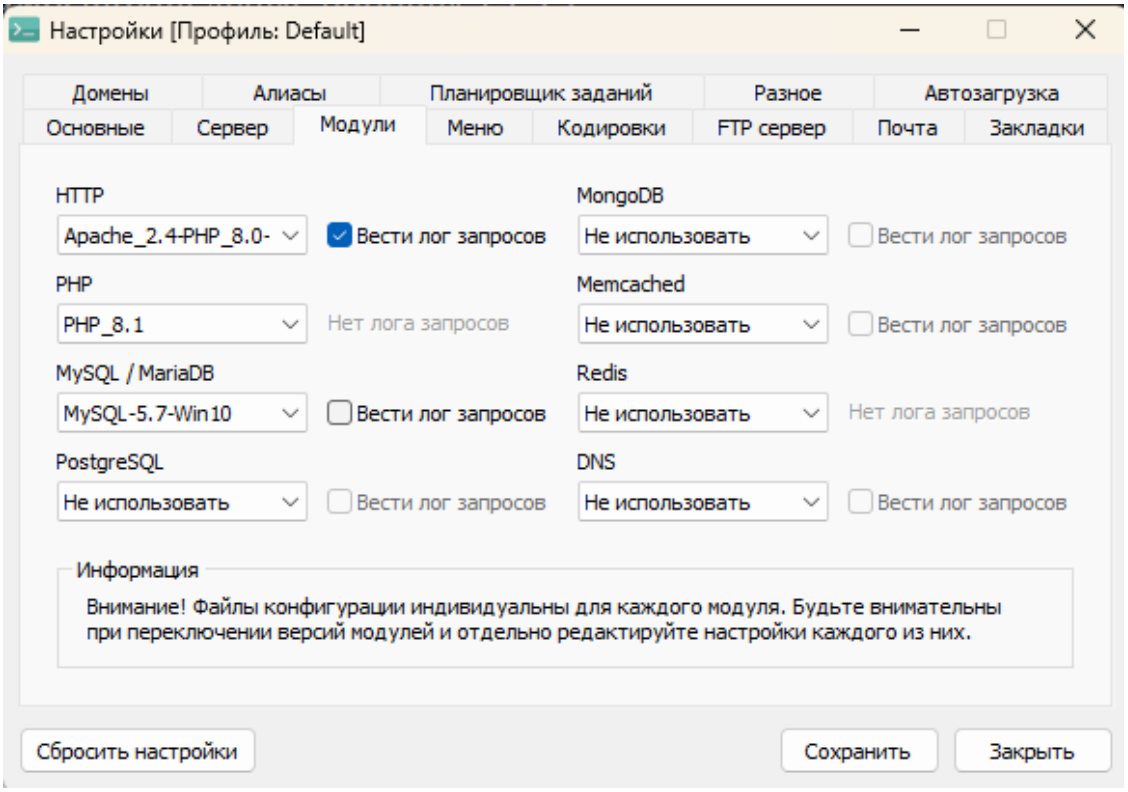
Пример выполнения задания

Используемые технологии: HTML, CSS, Bootstrap 5.3, PHP 8.X

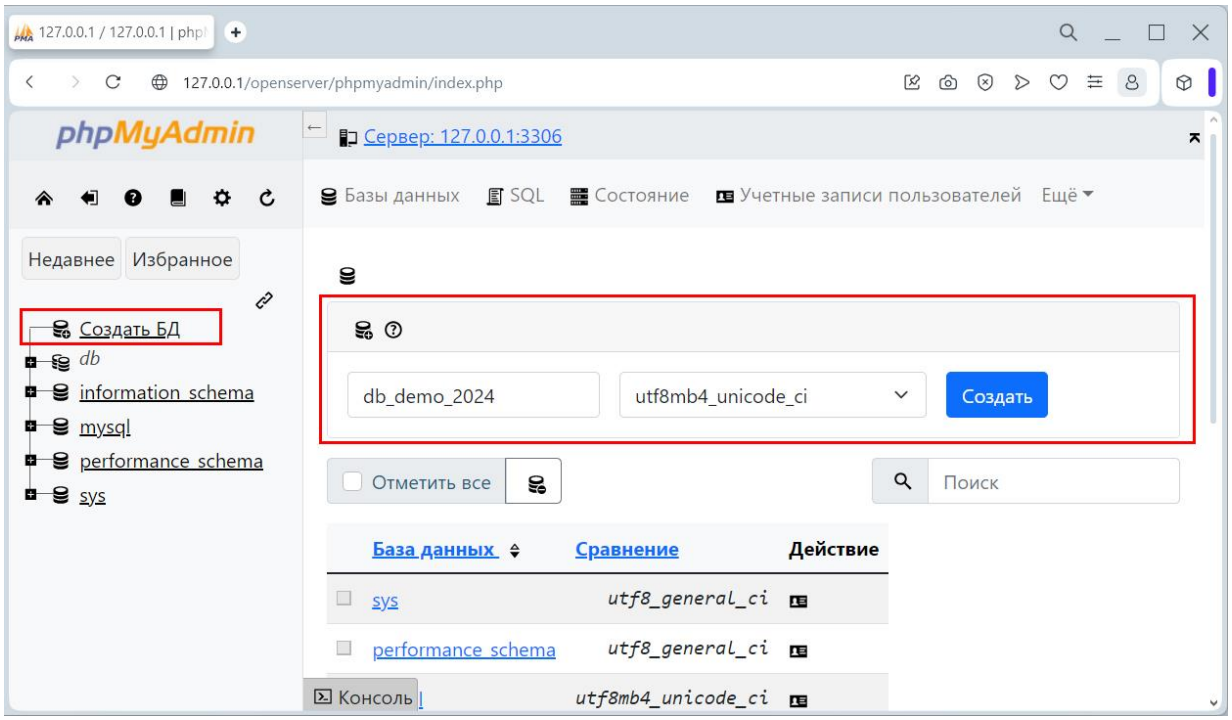
Важно! Рассматриваем ситуацию выполнения задания с использованием локального сервера OpenServer.



Этап 1. Подготовка. Создание базы данных

1. Создаем структуру проекта. Назначение и содержимое папок будем рассматривать подробнее в ходе выполнения работы.

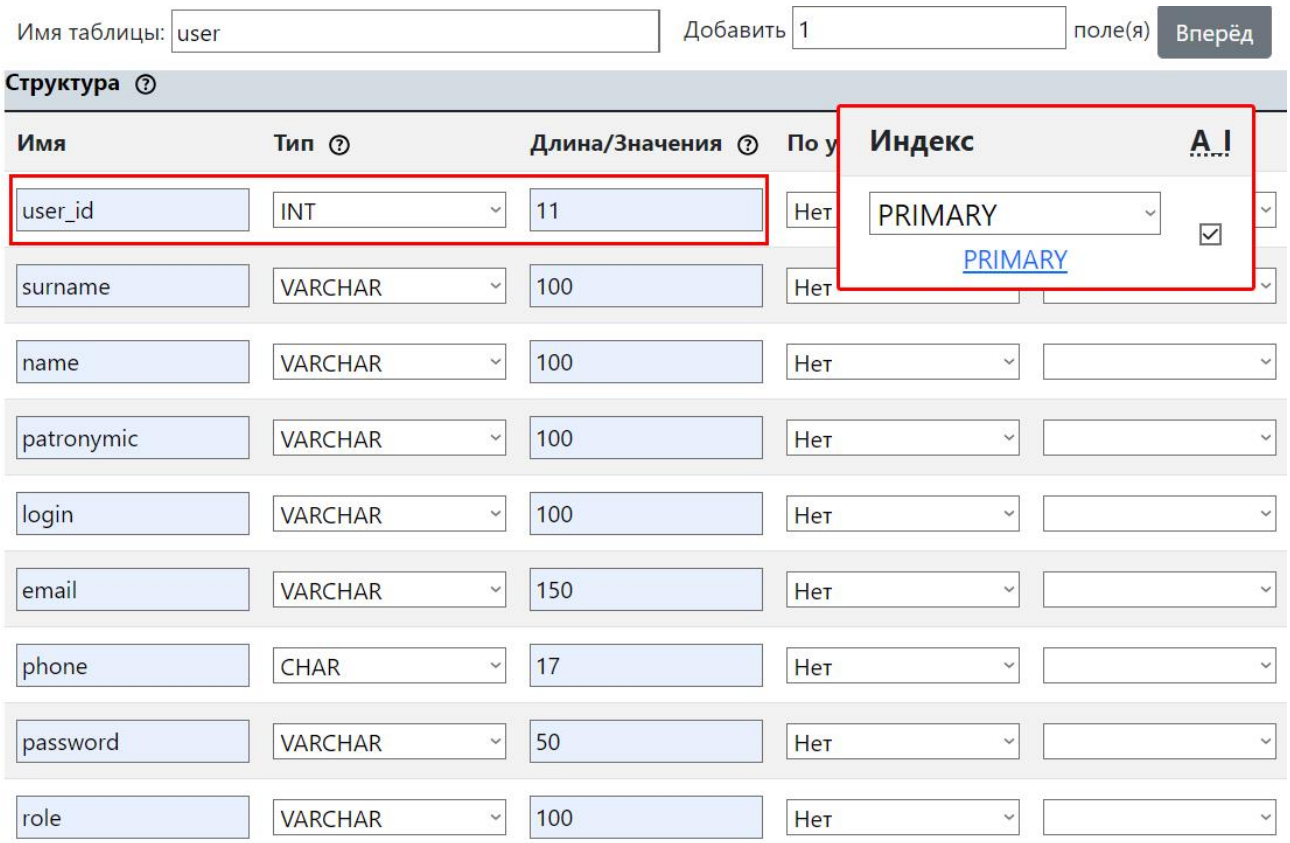
2. Создаем базу данных **db_demo_2024**.



База состоит из двух таблиц: **user** и **application** для хранения пользователей и их обращений соответственно.

Важно! Для **application_id** и **user_id** установите флажок **AI** (Auto Increment).

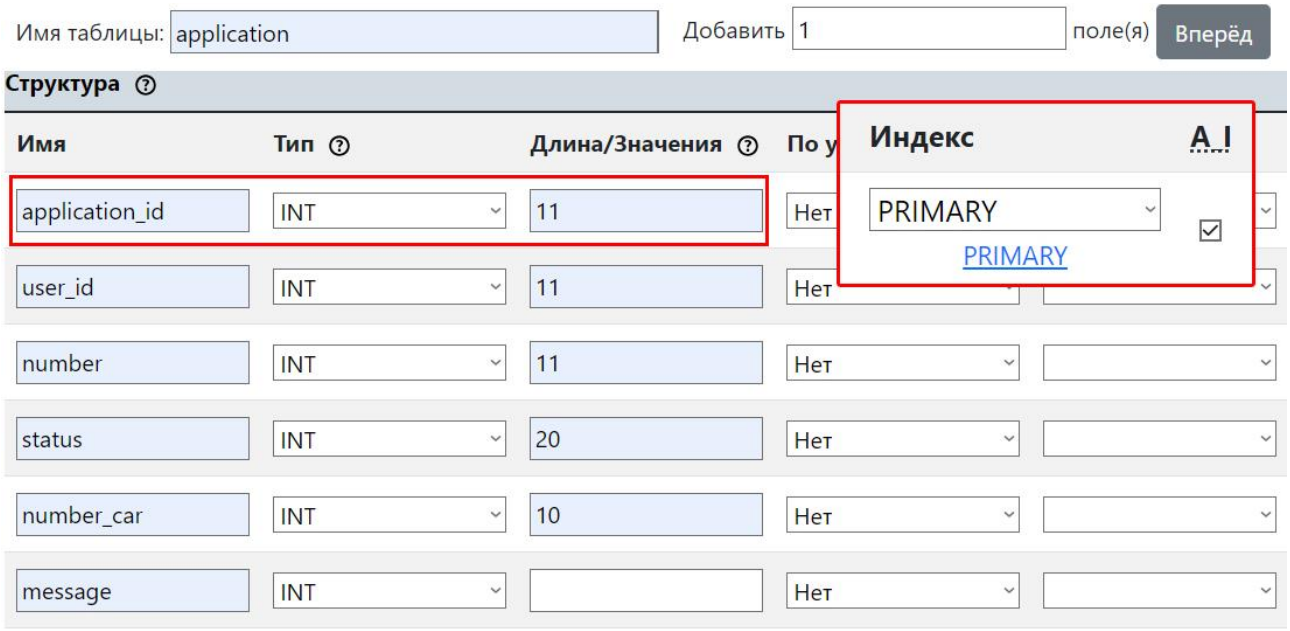
Таблица User



Можете не заполнять таблицу **user** сразу. Но не забудьте, что у одного из пользователей обязательно должны быть данные **login**: copp, **password**: password и **role**: Администратор.

Важно! В текущем задании требований к защищенному хранению пароля нет. Но в реальных проектах пароли хранятся не в открытом виде, а в виде хеша.

Таблица Application



3. Переносим файлы **bootstrap** и **шрифты** в папки **assets/bootstrap** и **assets/style/font** соответственно.

Этап 2. Настройка общих файлов проекта

Подключение к БД

1. Создаем файл **connect.php** в папке **function/**.
2. Настраиваем в нем подключение к базе данных.

Листинг 2.1 Подключение к БД в файле connect.php

```php
<?php
    $connect = new mysqli("localhost", "root", "", "db_demo_2024");

    if($connect->connect_error){
        die "Ошибка подключения к базе данных";
    }
?>
```

Важно! Ваша строка подключения может выглядеть по-другому в зависимости от версии PhpMyAdmin и названия файла базы. Например,

Листинг 2.2 Альтернативное подключение к БД

```
...
    $connect = new mysqli("localhost", "root", "root", "db_demo");
...
```

Настройка стилей

1. Создаем файл **style.css** в папке **style/**. Наполняем файл базовыми стилями.

Листинг 2.3 Собственные стили style.css

```
@font-face {
    font-family: Raleway;
    src: url("font/raleway/static/Raleway-Regular.ttf");
}

header, footer { font-family: Raleway; }

section.page { min-height: 83vh; }

div.cards_btn a { width: 48%; }
```

Важно! Ваши стили могут выглядеть по-другому в зависимости от выданного шрифта и личных предпочтений.

Файл header.php

1. Создаем файл **header.php** в папке **inc/**.
2. Начинаем со стандартной структуры страницы с подключением Bootstrap и собственных стилей.

Важно! Следите за порядком подключения стилей, чтобы избежать конфликтов.

Листинг 2.4 Подключение стилей в файле header.php

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Нарушениям.Нет</title>
    <link rel="stylesheet" href="/assets/bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="/assets/style/style.css">
</head>
<body>
    ...
```

3. В теле документа размещаем контейнеры для адаптивности контента и настройки внешнего вида страницы, а также элемент **nav** с классом **navbar**.

Важно! Здесь и далее, вы можете не писать классы bootstrap для оформления элементов, или заменить их на свои. Помните, что наша задача – получить приятный дизайн страниц наименьшим количеством кода.

Листинг 2.5 Шапка страниц

```
<body>
    <main class="text-success-emphasis">
        <header class="bg-success border-bottom border-success-emphasis">
            <div class="container-fluid d-flex align-items-center p-0">
                //здесь будет вставка элемента nav из Zeal
            </div>
        </header>
```

4. Копируем **nav.navbar** со всем содержимым из **Zeal**:
   Bootstrap5 – Guides – Navbar – раздел ***Supported content.***

Вносим небольшие изменения:

- Меняем класс `bg-body-tertiary` на `navbar-dark`.
- Меняем содержимое ссылки по умолчанию на собственное. Получаем:
  `<a class="navbar-brand" href="/">Нарушениям.Нет</a>`
- Содержимое списка **ul** удаляем, кроме одной конструкции.

Листинг 2.6 Элемент списка для динамического меню

```
<li class="nav-item">
   <a class="nav-link" href="#">Link</a>
</li>
```

- Переместим эту конструкцию в свободное пространство страницы и будем использовать при определении переменной **$menu.** Она будет формироваться в зависимости от роли пользователя (см. далее).

5. Вместо удаленного содержимого в списке указываем ссылку на переменную **$menu**.

Листинг 2.7 Меню сайта

```
<body>
    <main class="text-success-emphasis">
        <header class="bg-success border-bottom border-success-emphasis">
            <div class="container-fluid d-flex align-items-center p-0">

                //начало вставки из Zeal
                <nav class="navbar navbar-expand-lg w-100 p-3 navbar-dark">
                    <div class="container-fluid">
                        <a class="navbar-brand" href="/">Нарушениям.Нет</a>
                        <button class="navbar-toggler" type="button"
                                data-bs-toggle="collapse"
                                data-bs-target="#navbarSupportedContent"
                                aria-controls="navbarSupportedContent"
                                aria-expanded="false" aria-label="Toggle navigation">
                            <span class="navbar-toggler-icon"></span>
                        </button>
                        <div class="collapse navbar-collapse"
                             id="navbarSupportedContent">
                            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                                <?=$menu?>
                            </ul>
                        </div>
                    </div>
                </nav>

            </div>
        </header>
```

6. Вернемся к началу страницы и ДО объявления типа документа запустим механизм формирования переменной меню `$menu`. Старт или продолжение сессии `$_SESSION` будем определять в начале каждой страницы отдельно.

- Авторизованный пользователь увидит в меню пункты: **Личный кабинет**, **Выйти**.
- Пользователь, авторизованный с правами администратора: **Личный кабинет**, **Выйти**, **Админ Панель**.
- Неавторизованный пользователь: **Вход**, **Регистрация**.

Для записи в переменную используем, отложенную ранее конструкцию (см. Листинг 2.6), заменяя адреса ссылок и текст.
Код располагаем в самом начале страницы!

Листинг 2.8 Формирование переменной $menu

```
<?php
    $menu = "";
    if(isset($_SESSION['login'])){
        if($_SESSION['role'] == "Администратор"){
            $menu .= '<li class="nav-item">
                        <a class="nav-link" href="/admin/">Админ Панель</a>
                      </li>';
        }
        $menu .= '<li class="nav-item">
                    <a class="nav-link" href="/profile/">Личный кабинет</a>
                  </li>
                  <li class="nav-item">
                      <a class="nav-link" href="/admin/controllers/logout.php">
                      Выйти </a>
                  </li>';
    }else{
        $menu = '<li class="nav-item">
                   <a class="nav-link" href="/auth/">Вход</a>
                 </li>
                 <li class="nav-item">
                   <a class="nav-link" href="/register/">Регистрация</a>
                 </li>';
    }
?>
<!DOCTYPE html>
   ...
```

Файл footer.php

1. Создаем файл **footer.php** в папке **inc/**.

Листинг 2.9 Подвал сайта

```
        <footer class="sticky-bottom bg-dark text-light">
            <div class="container-fluid p-3 pt-3">
                <p>@Нарушениям.Нет<br>2023-2024</p>
            </div>
        </footer>
    </main>
    <script src="/assets/bootstrap/js/bootstrap.bundle.js"></script>
</body>
</html>
```

2. Также перед закрывающим тегом **body** разместим еще один скрипт. Он позволит автоматизировать процесс вывода сообщений при валидации полей форм. Код скрипта копируем из **Zeal**: *Bootstrap5 – Guides – Validation.*

Листинг 2.10 Скрипт валидации форм

```
    <script>
    // Example starter JavaScript for disabling form submissions if there are invalid
fields
      (() => {
        'use strict'

        // Fetch all the forms we want to apply custom Bootstrap validation styles to
        const forms = document.querySelectorAll('.needs-validation')
```

```
        // Loop over them and prevent submission
        Array.from(forms).forEach(form => {
          form.addEventListener('submit', event => {
            if (!form.checkValidity()) {
              event.preventDefault()
              event.stopPropagation()
            }

            form.classList.add('was-validated')
          }, false)
        })
      })()
    </script>

</body>
```

Этап 3. Вход | Регистрация | Заявление

Страница Вход

1. Создаем страницу **index.php** в папке **auth/**.

2. Стартуем сессию, подключаем «шапку» и «подвал» через **include**.

Листинг 3.1 Страница auth/index.php

```
<?php
    session_start();
    include "../inc/header.php";
?>
        <!--здесь будет контент-->

<?php
    include "../inc/footer.php";
?>
```

Важно! Перед скриптом с объявлением старта сессии не должно быть никаких пробельных символов (пустых строк или пробелов).

3. Устанавливаем блоки для размещения контента.

Листинг 3.2 Страница auth/index.php. Контент

```
<section class="page">
    <div class="container p-3">
        ...
    </div>
</section>
```

4. Добавляем заголовок страницы.

Листинг 3.3 Страница auth/index.php. Контент

```
...
<h1 class="text-center mb-3 text-success-emphasis display-1">Вход</h1>
...
```

5. Добавляем и настраиваем форму, которая содержит два поля с подписями и кнопку **Войти**. Не забывайте про bootstrap классы **.form-label** и **.form-control** для элементов формы.

    - Метод передачи данных: **post**.

- Обработчик формы: **/admin/controllers/login.php**.
- Поля обязательны для заполнения.

Листинг 3.4 Страница auth/index.php. Форма авторизации

```
<form action="/admin/controllers/login.php" method="post" class="m-auto">

   <div class="mb-3">
       <label for="login" class="form-label fs-5">Ваш логин</label>
       <input type="text" class="form-control shadow-sm p-3 rounded-pill"
              id="login" name="login" required>

   </div>

   <div class="mb-3">
       <label for="password" class="form-label fs-5">Ваш пароль</label>
       <input type="password" class="form-control shadow-sm p-3 rounded-pill"
              id="password" name="password" required>

   </div>

   <input type="submit" class="btn btn-success mb-3 mt-3 w-100 shadow-sm p-3
          fs-2 rounded-pill fw-bold" value="Войти">

</form>
```

Важно! Вы можете скопировать блок «подпись-поле» из **Zeal**:

Bootstrap5 – Guides – Form Controls – раздел ***Example.***

- Редактируйте значения атрибута **type** для **input**, а также значения атрибутов: **for**, **id**, **name**.
- Классы оформления кнопки **Войти** будут использоваться и на других страницах.

6. После старта сессии выполняем проверку. Авторизованный пользователь, вместо страницы **Вход** будет переходить на страницу **Личный кабинет**.

Листинг 3.5 Страница auth/index.php. Проверка логина

```
<?php
    session_start();

    if(isset($_SESSION['login'])){
        header("Location: /profile/");
        exit;
    }

    include "../inc/header.php";
?>
```

7. Настроим вывод сообщения пользователю при неудачной авторизации. Добавим блок с условием на страницу. Содержимое **message** передается из обработчика формы **admin/controllers/login.php**. Поработаем с ним на следующем этапе.

Листинг 3.6 Вывод сообщения об ошибке

```
<section class="page">
    <div class="container p-3">
```

```php
<?php
    if(isset($_GET['message'])){
        echo "<div class='border border-danger text-danger
                          text-center pt-4 pb-4 mb-3'>
        {$_GET['message']}
        </div>";
    }
?>

<h1 ...">Вход</h1>
```

Страница Регистрация

1. Копируем страницу **Вход** *(auth/index.php)* в папку **register/**.
2. Меняем заголовок страницы на **Регистрация**.
3. Редактируем форму:
   - Обработчик формы: /admin/controllers/registration.php.
   - Блоки с парами **«подпись-поле»** копируем необходимое количество раз, меняя текст подписи, значения атрибутов: **for**, **id**, **name** и **type** для **input**.
   - Добавляем проверки содержимого полей с помощью встроенных средств html.

Поле / подпись	Тип	Ключевое слово	Параметры проверки валидности поля
Фамилия	text	surname	обязательное
Имя	text	name	обязательное
Отчество	text	patronymic	обязательное
Логин	text	login	обязательное
Адрес электронной почты	email	email	обязательное
Телефон	tel	phone	обязательное количество символов 17, pattern="/+?[0-9/(/)/-]+" placeholder="+7(XXX)-XXX-XX-XX"
Пароль	password	password	обязательное длина не менее 6 символов
Повтор пароля	password	password-repeat	обязательное

Листинг 3.7 Страница register/index.php. Форма регистрации

```
<div class="container p-3">
   <h1 class="text-center mb-3 text-success-emphasis display-1">Регистрация</h1>
   <form action="/admin/controllers/registration.php" method="post" class="m-auto">
       <div class="mb-3">
           <label for="surname" class="form-label fs-5">Фамилия</label>
           <input type="text" class="form-control shadow-sm  p-3 rounded-pill"
id="surname" name="surname" required>
       </div>
       <div class="mb-3">
           <label for="name" class="form-label fs-5">Имя</label>
```

```
            <input type="text" class="form-control shadow-sm  p-3 rounded-pill"
id="name" name="name" required>
        </div>
        <div class="mb-3">
            <label for="patronymic" class="form-label fs-5">Отчество</label>
            <input type="text" class="form-control shadow-sm  p-3 rounded-pill"
id="patronymic" name="patronymic" required>
        </div>
        <div class="mb-3">
            <label for="login" class="form-label fs-5">Логин</label>
            <input type="text" class="form-control shadow-sm  p-3 rounded-pill"
id="login" name="login" required>
        </div>
        <div class="mb-3">
            <label for="email" class="form-label fs-5">Адрес электронной почты</label>
            <input type="email" class="form-control shadow-sm  p-3 rounded-pill"
id="email" name="email" required>
        </div>
        <div class="mb-3">
            <label for="phone" class="form-label fs-5">Телефон</label>
            <input type="tel" class="form-control shadow-sm  p-3 rounded-pill"
id="phone" name="phone" minlength="17" maxlength="17" pattern="/+?[0-9/(/)/-]+"
placeholder="+7(XXX)-XXX-XX-XX" required>
        </div>
        <div class="mb-3">
            <label for="password" class="form-label fs-5">Пароль</label>
            <input type="password" class="form-control shadow-sm  p-3 rounded-pill"
id="password" name="password" minlength="6" required>
        </div>
        <div class="mb-3">
            <label for="password-repeat" class="form-label fs-5">Повторите
пароль</label>
            <input type="password" class="form-control shadow-sm  p-3 rounded-pill"
id="password-repeat" name="password-repeat" minlength="6" required>
        </div>
        <input type="submit" class="btn btn-success mb-3 mt-3 w-100 shadow-sm  p-3 fs-2
rounded-pill fw-bold" value="Зарегистрироваться">
    </form>
</div>
```

4. Для запуска скрипта валидации (который прикрепили в подвале сайта **inc/footer.php**) и вывода сообщений об ошибках валидации добавим:

    - для формы: класс – .needs-validation и атрибут novalidate.
    - **для валидируемых полей**: блок с классом **.invalid-feedback** и текстом сообщения об ошибке.

Листинг 3.8 Вывод сообщений об ошибках валидации

```
<form action="/admin/controllers/registration.php" method="post"
      class="m-auto needs-validation" novalidate>
```

```
    <div class="mb-3">
        <label for="surname" class="form-label fs-5">Фамилия</label>
        <input type="text" class="form-control shadow-sm  p-3 rounded-pill"
id="surname" name="surname" required>
        <div class="invalid-feedback">
            Поле обязательно для заполнения.
        </div>
    </div>
      ...

</form>
```

Важно! Располагайте блок с текстом сообщения об ошибке сразу после валидируемого поля.

Классы и расположение элементов можно скопировать из **Zeal**: *Bootstrap5 – Guides – Validation.*

Страница Заявление

1. Копируем страницу **Вход** *(auth/index.php)* в папку **application/**.
2. Редактируем код проверки логина.

Листинг 3.9 Страница application/index.php. Проверка логина

```
if(!isset($_SESSION['login'])){
        header("Location: /auth/");
        exit;
    }
```

3. Меняем заголовок страницы на **Подача заявления**.
4. Удаляем вывод ошибки авторизации.
5. Редактируем форму:
    - Обработчик формы: **/admin/controllers/add_application.php** (будет реализован далее в процессе выполнения работы).
    - Меняем текст подписей и значения атрибутов: **for**, **id**, **name**.
    - Проверяем, чтобы все поля были **обязательны для заполнения**.
    - Второй элемент **input** меняем на **textarea**. Стили остаются те же.

Листинг 3.10 Страница application/index.php. Форма подачи заявления

```
<section class="page">
    <div class="container p-3">
        <h1 class="text-center mb-3 text-success-emphasis display-1">Подача
заявления</h1>
        <form action="/admin/controllers/add_application.php" method="post"
              class="m-auto needs-validation" novalidate>
            <div class="mb-3">
                <label for="number" class="form-label fs-5">
                        Государственный регистрационный номер автомобиля
                </label>
                <input type="text"
```

```
                        class="form-control shadow-sm p-3 rounded-pill"
                        id="number"
                        name="number"
                        required>
                    <div class="invalid-feedback">
                        Поле обязательно для заполнения.
                    </div>
                </div>
                <div class="mb-3">
                    <label for="message" class="form-label fs-5">Описание нарушения</label>
                    <textarea
                        class="form-control shadow-sm p-3 rounded-5"
                        id="message"
                        name="message"
                        rows="5"
                        required>
                   </textarea>
                    <div class="invalid-feedback">
                        Поле обязательно для заполнения.
                    </div>
                </div>
                <input type="submit" class="btn btn-success mb-3 mt-3 w-100 shadow-sm p-3
                                        fs-2 rounded-pill fw-bold"
                       value="Подать заявление">
            </form>
        </div>
    </section>
```

Этап 4. Функционал входа, выхода, регистрации

Функционал Вход

1. Создаем страницу **login.php** в папке **admin/controllers/**.
2. Начинаем сессию. Подключаемся к базе.
3. Сравниваем пары логин-пароль в базе и получаемые из формы.
   - Если результат **совпадает** – **передаем** в сессию значение **логина** и **роль** пользователя. Переходим на страницу **Личный кабинет**.
   - Если **не совпадает** – **передаем** в message **сообщение** «Некорректный логин или пароль». Переходим на страницу **Авторизация**.

Листинг 4.1 Страница admin/controllers/login.php. Обработка входа

```
<?php
    session_start();

    include '../../function/connect.php';

    $sql = sprintf("SELECT * FROM `user` WHERE `login` = '%s' AND `password` = '%s'",
                    $_POST['login'], $_POST['password']);

    $result = $connect->query($sql);

    if($result->num_rows){

        $row = $result->fetch_assoc();
        $_SESSION['login'] = $row['login'];
        $_SESSION['role'] = $row['role'];
```

```
        header("Location: /profile/");
        exit;
    }else{
        header("Location: /auth/index.php?message=Некорректный логин или пароль");
        exit;
    }
?>
```

Функционал Выход

1. Создаем страницу **logout.php** в папке **admin/controllers/**.
2. Возобновляем сессию.
3. Очищаем содержимое сессии. Переходим в корневой каталог проекта.

Листинг 4.2 Страница admin/controllers/logout.php. Обработка выхода

```
<?php
    session_start();

    unset($_SESSION['login']);
    unset($_SESSION['role']);
    header("Location: /");
    exit;
?>
```

Важно! Очистить содержимое сессии можно и другим способом, например функцией **session_unset()**, или удалить сессию полностью с помощью **session_destroy()**.

Функционал Регистрация

1. Создаем страницу **regisration.php** в папке **admin/controllers/**.
2. Начинаем сессию. Подключаемся к базе.
3. Проверяем логин на уникальность.
    a. Если логин повторяется, переходим на страницу **register/index.php** с отображением ошибки.
    b. Если совпадений не найдено, записываем в базу данные, получаемые из формы.
        - В сессию записываем логин и роль пользователя.
        - Переходим на страницу **Личный кабинет**.

Листинг 4.3 Страница admin/controllers/regisration.php. Обработка регистрации

```
<?php
    session_start();

    include '../../function/connect.php';

    //проверка уникальности логина
    $data = $connect->query(sprintf("SELECT `login` FROM `user` WHERE `login` = '%s'",
$_POST['login']));
```

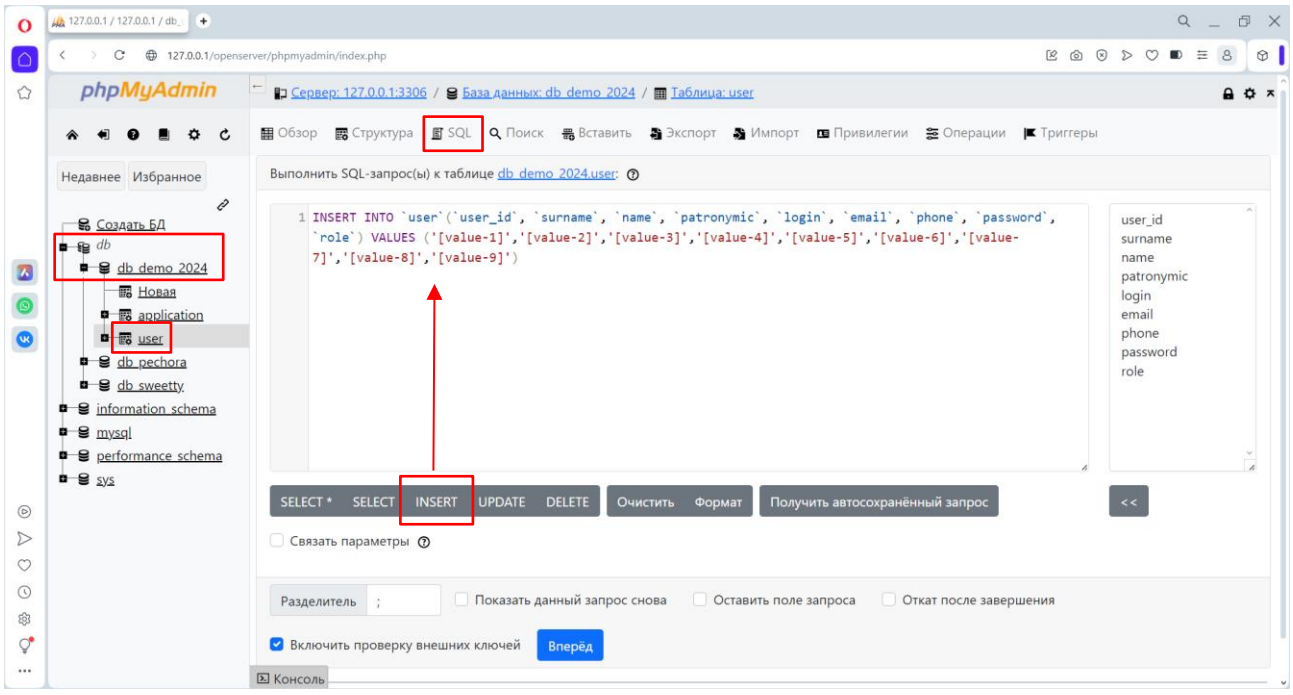
```php
    // передаем ошибку при совпадении логина
    if($data->num_rows){
        header("Location: /register/index.php?message=Пользователь с таким логином уже существует");
        exit;
    } else {
       //запись в базу данных о регистрации
        $sql = sprintf("INSERT INTO `user`(`user_id`, `surname`, `name`, `patronymic`, `login`, `email`, `phone`, `password`, `role`) VALUES
(NULL,'%s','%s','%s','%s','%s','%s','%s','Пользователь')",
            $_POST['surname'],
            $_POST['name'],
            $_POST['patronymic'],
            $_POST['login'],
            $_POST['email'],
            $_POST['phone'],
            $_POST['password']
        );

        if(!$connect -> query($sql)){
            echo "Ошибка добавления данных";
        }

        $_SESSION['login'] = $_POST['login'];
        $_SESSION['role'] = "Пользователь";

        header("Location: /profile/");
        exit;
    }
?>
```

Важно! Для ускорения работы, можно получить сроку `INSERT INTO...` через PhpMyAdmin. Выбираем нужную таблицу – переходим на вкладку **SQL** – нажимаем кнопку **Insert**.



Этап 5. Страница Личный кабинет. Файл function.php

Страница Личный кабинет

1. Копируем страницу **Подача заявления** *(application/index.php)* в папку **profile/**.

2. Подключаем файл **function.php**.

Листинг 5.1 Подключение function.php

```
<?php
    ...
    include "../inc/header.php";
    include "../function/function.php";
?>
```

3. Меняем заголовок страницы на **Личный кабинет**.

4. От формы оставляем только кнопку **Подать заявление**. Внесем в неё изменения.

Листинг 5.2 Ссылка Подать заявление

```
...
// было
<input type="submit"
   class="btn btn-success mb-3 mt-3 w-100 shadow-sm  p-3 fs-2 rounded-pill fw-bold"
   value="Подать заявление">

//стало
<a href="/application/"
   class="btn btn-success mb-3 mt-3 w-100 shadow-sm p-3 fs-2 rounded-pill fw-bold">
    Подать заявление
</a>
...
```

5. В контенте страницы вместо формы выводим результаты выполнения функций `fnGetProfile` и `fnGetCardProfile`.
   Сами функции описаны в файле **function.php** (см. далее).

Листинг 5.3 Вывод данных из функций

```
<div class="container p-3">
    <h1 ...> Личный кабинет</h1>

    <?php
        echo fnGetProfile($_SESSION['login']);
        echo fnGetCardProfile($_SESSION['login']);
    ?>

    <a ...>Подать заявление</a>
</div>
```

Файл function.php

1. Создаем файл **function.php** в папке **function/**.
2. Файл содержит подключение к базе данных и три функции:
   - `fnGetProfile` – используя логин, обращается к базе и получает данные пользователя (Фамилия, Имя, Отчество, телефон) и выводит их на странице в виде абзацев.
   - `fnGetCardProfile` – используя логин, получает id пользователя и выводит все его обращения в виде карточек.

- **fnGetCardAdmin** – для администратора выводит карточки обращений всех пользователей с дополнительными кнопками для подтверждения и отмены. *Рассмотрим ее подробнее на следующем этапе проектирования.*

Листинг 5.4 Структура function.php

```php
<?php
    include "connect.php";

    function fnGetProfile($login) {...}
    function fnGetCardProfile($login) {...}
    function fnGetCardAdmin(){...}
?>
```

fnGetProfile

Листинг 5.5 Функция fnGetProfile

```php
function fnGetProfile($login){

    global $connect;

    $sql = sprintf("SELECT `surname`, `name`, `patronymic`, `phone` FROM `user`
                    WHERE `login` = '%s'", $login);

    if(!$result = $connect->query($sql)){
        echo "Ошибка получения данных";
    }

    $row = $result->fetch_assoc();

    $data = sprintf('<p><span class="fw-semibold">Фамилия:</span> %s</p>
                     <p><span class="fw-semibold">Имя:</span> %s</p>
                     <p><span class="fw-semibold">Отчество:</span> %s</p>
                     <p><span class="fw-semibold">Телефон:</span> %s</p>',
                $row["surname"], $row["name"], $row["patronymic"], $row["phone"]);

    return $data;
}
```

fnGetCardProfile

3. Для всех обращений пользователя с текущим логином выбираем информацию: номер обращения, номер автомобиля, текст сообщения и статус.
4. Информация об обращении пользователя будет выводиться в блок с классом **cards**.

Листинг 5.6 Функция fnGetCardProfile

```php
function fnGetCardProfile($login){
    global $connect;

    $select = sprintf("SELECT `user_id` FROM `user` WHERE `login` = '%s'", $login);
    $select_result = $connect->query($select);
    $select_row = $select_result->fetch_assoc();
    $id = $select_row['user_id'];

    $data = '<div class="cards">';
```

```
    $sql = sprintf("SELECT `number`, `number_car`, `message`, `status` FROM
`application` INNER JOIN `user` ON `application`.`user_id` =  `user`.`user_id` WHERE
`application`.`user_id` = '%s' ORDER BY `application_id` DESC", $id);

    if(!$result = $connect->query($sql)){
        echo "Ошибка получения данных";
    }

        // место для вывода карточек Листинг 5.8

    $data .= "</div>";
    return $data;
}
```

5. Выводим информацию об обращениях.

Листинг 5.7 Код карточки обращения

```
<div class="card w-100 mb-3 mt-3">
   <div class="card-body">
     <h5 class="card-title">Нарушение №%s</h5>
     <p class="mb-1"><span class="fw-semibold">Статус:</span> %s</p>
     <p class="mb-1"><span class="fw-semibold">Гос.номер автомобиля:</span> %s</p>
     <p class="card-text">%s</p>
   </div>
</div>
```

Код карточки можно скопировать из Zeal, удалив ненужные элементы**:** атрибут style, изображение и ссылку: *Bootstrap5 – Guides – Cards – раздел* ***Example.***

6. Простой вариант вывода всех обращений пользователя реализуем через цикл **while**.

Листинг 5.8 Функция fnGetCardProfile. Простая реализация

```
while($row = $result->fetch_assoc())
    $data .= sprintf('<div class="card w-100 mb-3 mt-3">
                          <div class="card-body">
                              <h5 class="card-title">Нарушение №%s</h5>
                              <p class="mb-1"><span class="fw-semibold">Статус:</span>
%s</p>
                              <p class="mb-1"><span class="fw-semibold">Гос номер
автомобиля:</span> %s</p>
                              <p class="card-text">%s</p>
                          </div>
                      </div>',
                      $row['number'],
                      $row['status'],
                      $row['number_car'],
                      $row['message']);
}
```

Но мы предлагаем **еще один вариант**, который положительно повлияет на пользовательский опыт общения с вашим проектом.

Каждое обращение имеет дополнительный статус: Отменен или Подтвержден. С помощью проверки статуса будем менять цветовое решение карточки: **text-body-tertiary** для отмененных и **success** для подтвержденных.

Содержимое карточек неизменно (см. Листинг 5.7), редактируем только классы контейнера **.card.**

Листинг 5.9 Функция fnGetProfile. Изменение стилей карточек

```
while($row = $result->fetch_assoc()){
       if($row['status'] == 'Отменен'){
           $data .= sprintf('<div class="card w-100 mb-3 mt-3 text-body-tertiary">
                                    // см. код Листинг 5.7
                            </div>',
                            $row['number'],
                            $row['status'],
                            $row['number_car'],
                            $row['message']);
            } elseif($row['status'] == 'Подтвержден'){
           $data .= sprintf('<div class="card w-100 mb-3 mt-3 text-success">
                                    // см. код Листинг 5.7
                            </div>',
                            $row['number'],
                            $row['status'],
                            $row['number_car'],
                            $row['message']);
            } else{
           $data .= sprintf('<div class="card w-100 mb-3 mt-3">
                                    // см. код Листинг 5.7
                            </div>',
                            $row['number'],
                            $row['status'],
                            $row['number_car'],
                            $row['message']);
               }
}
```

Такое разделение карточек по статусам пригодится нам и при выводе карточек для администратора на следующем этапе.

Этап 6. Панель администратора

Файл index.php

1. Копируем страницу **Личный кабинет** *(profile/index.php)* в папку **admin/**.
2. Настраиваем проверку пользователя.

Листинг 6.1 Проверка пользователя

```
<?php
    session_start();
    if($_SESSION['role'] != "Администратор"){
        header("Location: /profile/");
        exit;
    }
    include "../inc/header.php";
    include "../function/function.php";
?>
```

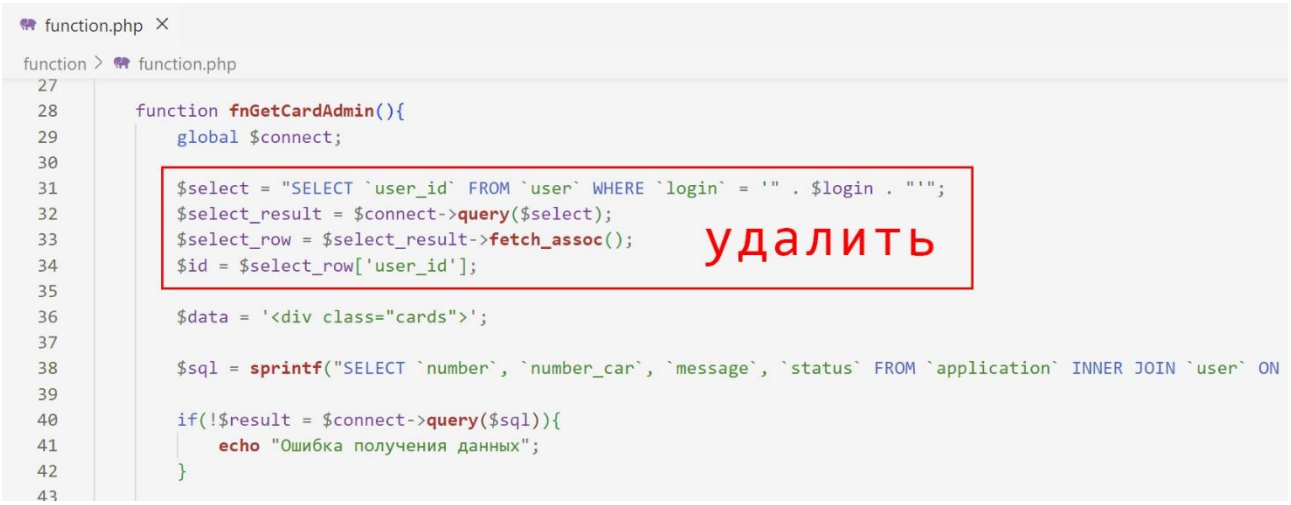
3. Меняем заголовок страницы.
4. Убираем функцию вывода информации и кнопку подачи заявления.
5. Функцию вывода заявлений переименовываем `fnGetCardAdmin()`.

Листинг 6.2 Страница admin/index.php

```
<section class="page">
    <div class="container p-3">
        <h1 class="text-center mb-3 text-success-emphasis display-1">
            Панель администратора
        </h1>
        <?php echo fnGetCardAdmin();?>
    </div>
</section>
```

Файл function.php

1. Переходим к файлу **function.php** в папке **function/**.
2. Дублируем код вывода обращений из профиля и меняем название функции с `fnGetCardProfile($login)` на `fnGetCardAdmin()`.
3. Редактируем функцию. Удаляем блок определения пользователя.



4. Дополняем перечень выбираемых данных информацией о Фамилии, Имени, Отчестве заявителя.

Листинг 6.3 Выборка данных

```
      $sql = "SELECT `application_id`, `surname`, `name`, `patronymic`, `number`,
`number_car`, `message`, `status` FROM `application` INNER JOIN `user` ON
`application`.`user_id` =  `user`.`user_id` ORDER BY `application_id` DESC";
```

5. Шаблон для карточек **всех статусов** дополняем строками Фамилия, Имя, Отчество.

Листинг 6.4 Вывод карточки

```
if($row['status'] == 'Отменен'){
    $data .= sprintf('<div class="card w-100 mb-3 mt-3 text-body-tertiary">
                        <div class="card-body">
                          <h5 class="card-title">Нарушение №%s</h5>
                          <p class="mb-1">
```

```
                        <span class="fw-semibold">Фамилия:</span> %s
                    </p>
                    <p class="mb-1">
                        <span class="fw-semibold">Имя:</span> %s
                    </p>
                    <p class="mb-1">
                        <span class="fw-semibold">Отчество:</span> %s
                    </p>
                    <p class="mb-1">
                        <span class="fw-semibold">Статус:</span> %s
                    </p>
                    <p class="mb-1">
                        <span class="fw-semibold">Гос номер авто:</span> %s
                    </p>
                    <p class="card-text">%s</p>
                </div>
            </div>',
            $row['number'],
            $row['surname'],
            $row['name'],
            $row['patronymic'],
            $row['status'],
            $row['number_car'],
            $row['message']);
    }
```

6. Для карточек обращений **без особого статуса** (отменен или подтвержден):

   - добавляем в вывод две кнопки: **Подтвердить** и **Отменить**. Нажатие по ним запустит обработчик в файле **update_applicate.php** с определенным параметром.

Ссылки можно скопировать из файла **profile/index.php**, за исключением `w-100` и адреса. Код стилей второй ссылки тот же + `btn-success-outline border border-success m-0`.

Листинг 6.5 Кнопки для смены статуса обращения

```
<div class="d-flex align-items-center justify-content-between cards_btn">

    <a href="controllers/update_applicate.php?id=%s&action=success" class="card-link
btn mb-3 mt-3 shadow-sm p-3 rounded-pill fw-bold btn-success">Подтвердить</a>

    <a href="controllers/update_applicate.php?id=%s&action=cancel" class="card-link btn
mb-3 mt-3 shadow-sm p-3 rounded-pill fw-bold btn-success-outline border border-success
m-0">Отменить</a>

</div>
```

   - добавляем в список переменных **application_id**.

Листинг 6.6 Данные для подстановки

```
            ...
            $row['number'],
```

```
                    $row['surname'],
                    $row['name'],
                    $row['patronymic'],
                    $row['status'],
                    $row['number_car'],
                    $row['message'],
                    $row['application_id'],
                    $row['application_id'] );
            }
```

Этап 7. Обновление статуса и подача заявления

1. Создаем файл **update_applicate.php** в папке **admin/controllers**.

В зависимости от переданного параметра **action** статус обращения будет сменяться **success** – **Подтвержден**, **cancel** – **Отменен**.

Листинг 7.1 Смена статуса обращения

```
<?php
    include "../../function/connect.php";

    if(isset($_GET['action'])){
        switch ($_GET['action']) {
            case 'success':
                $sql = sprintf("UPDATE `application` SET `status`='%s' WHERE
`application_id` = '%s'", 'Подтвержден', $_GET['id']);
                $connect -> query($sql);
                header("Location: /admin/");
                exit;

            case 'cancel':
                $sql = sprintf("UPDATE `application` SET `status`='%s' WHERE
`application_id` = '%s'", 'Отменен', $_GET['id']);
                $connect -> query($sql);
                header("Location: /admin/");
                exit;
        }
    }
?>
```

2. Создаем файл **add_application.php** в папке **admin/controllers**.

Листинг 7.2 Подача заявления

```
<?php
    session_start();

    include '../../function/connect.php';

    $select = sprintf("SELECT `user_id` FROM `user` WHERE `login` = '%s'",
$_SESSION['login']);

    $select_result = $connect->query($select);
    $select_row = $select_result->fetch_assoc();
    $id_user = $select_row['user_id'];

    $chars = '0123456789';
    $number = substr(str_shuffle($chars), 0, 5);
```

```
    $sql = sprintf("INSERT INTO `application`(`application_id`, `user_id`, `number`,
`status`, `number_car`, `message`) VALUES (NULL,'%s','%s','%s','%s','%s')",
    $id_user, $number, "Новый", $_POST['number'], $_POST['message']);

    $connect->query($sql);

    header("Location: /profile/");
    exit;
?>
```

Важно! Решение с формированием случайного номера заявления не оптимально и в задании никак не оговорено. Но позволяет получить похожий на настоящий номер в карточке личного кабинета.

Этап 8. Главная страница

Наличие, внешний вид и содержание главной страницы приложения не определены заданием. Но будет логичным ее создать и разместить на ней справочную информацию о проекте и кнопку перехода к процедуре подачи заявления.

1. Стартуем сессию.
2. Подключаем **header.php** и **footer.php**.

```
<?php
    session_start();
    include "inc/header.php";
?>

...

<?php
    include "inc/footer.php";
?>
```

3. Добавим текст и стилизованную ссылку **Подать заявление**.

```
<body class="bg-success">

<div class="container p-3 text-center">
    <h1 class="mt-3 mb-4 text-white display-1">
        Нарушениям.Нет
    </h1>
    <p class="p-1 pt-4 display-6 text-light">
        Наш портал представляет собой информационную систему для помощи полиции по
        своевременной фиксации нарушений правил дорожного движения.
    </p>
</div>

<div class="container p-2 mb-5 text-center">
    <a href="/auth/" class="btn btn-success text-success-emphasis bg-light
       w-75 p-3 fs-2 rounded-pill fw-bold shadow-lg">
         Подать заявление
    </a>
    <h2 class="display-4 mt-5 text-white">
        Будь ответственным гражданином!
    </h2>
</div>
```

Этап 9. Завершение работы

Завершающим этапом выполнения работы должны стать:

1. Проверка загрузки проекта на сервер!
2. Проверка кода на валидность.
3. Проверка наличия пользователя с правами администратора с логином и паролем из текста задания.
4. Создание скриншотов всех страниц.
    - Запускаем **Режим разработчика** браузера (**F12**).
    - В выпадающем меню выбираем пункт **Сделать полноразмерный скриншот**.
    - Переименовываем изображения в соответствии с заданием.