Задание № 1

==========

Необходимо разработать веб-приложение для соответствующей предметной области.

Инструкция к выполнению практической части:

Используйте все знания в области языков разметки и программирования, но не забывайте что избыточность кода-моветон, а кодовое разнообразие – признак хорошего вкуса. Каждый раздел сайта должен иметь свою отличительную черту. Не перегружайте вашу разработку обилием цветов, оставьте это для живописцев. При разработке поставьте себя на место потенциального клиента и постарайтесь создать идеальное веб-приложение.

Вам необходимо также разработать дизайн всех страниц для использования со смартфоном с разрешением 390x844 px. Дизайн можно представить в виде файлов изображений .png (отдельное изображение для каждой страницы), либо в виде .html файлов (отдельный файл для каждой страницы).

Критерии оценивания задания находятся в прилагаемом файле в формате pdf.

Описание предметной области:

Портал клининговых услуг «Мой Не Сам» представляет собой информационную систему для формирования заявок на уборку жилых и производственных помещений. Перед началом использования портала пользователю необходимо пройти процедуру регистрации. В ходе регистрации он указывает данные о себе (ФИО, телефон, адрес электронной почты, логин и пароль).

Войдя в систему, заказчик может сформировать заявку на получение услуг, указав свой адрес, контактные данные и конкретный вид услуги из списка предоставляемых. Также заказчик может указать желаемую дату и время получения услуги, а также предпочтительный тип оплаты (наличными или с помощью банковской карты).

Заявки заказчиков хранятся в системе. В каждой заявке находятся данные заказчика, статус оказания услуги (новая заявка, услуга оказана или услуга отменена).

После подачи заявки администратор может подтвердить заявку,

поставить отметку о выполнении или отклонить заявку с указанием причины отклонения.

Основной функционал информационной системы:

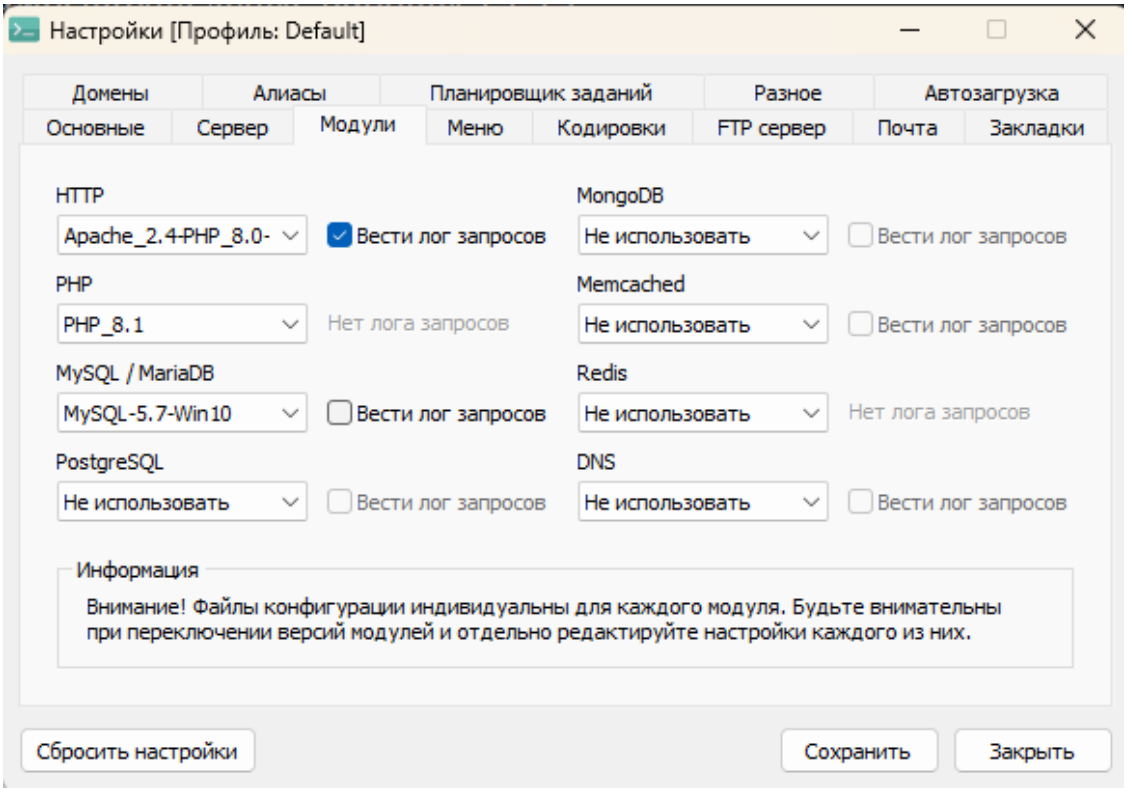
1. Страница регистрации. На данной странице необходимо предусмотреть добавление пользователя в систему. Пользователю необходимо предоставить возможность ввести уникальный логин, пароль, ФИО, телефон, адрес электронной почты. Все поля обязательны для заполнения. По кнопке «Зарегистрироваться» пользователь должен заноситься в базу. 2. Страница авторизации. На данной странице необходимо предусмотреть возможность ввода логина и пароля для зарегистрированных пользователей. Попытки некорректного ввода логина и пароля должны сопровождаться сообщениями.

3. Страница создания заявки. На данной странице авторизованный пользователь имеет возможность просмотреть историю своих заявок, а также оставить новую заявку.

4. Страница формирования заявки. Пользователь указывает: свой адрес, контактные данные, желаемую дату и время получения услуги. Помимо этого, пользователь выбирает вид услуги из предоставленного списка (общий клининг, генеральная уборка, послестроительная уборка, химчистка ковров и мебели). Также пользователь указывает предпочтительный тип оплаты (наличные или банковская карта). Все поля обязательны для заполнения.

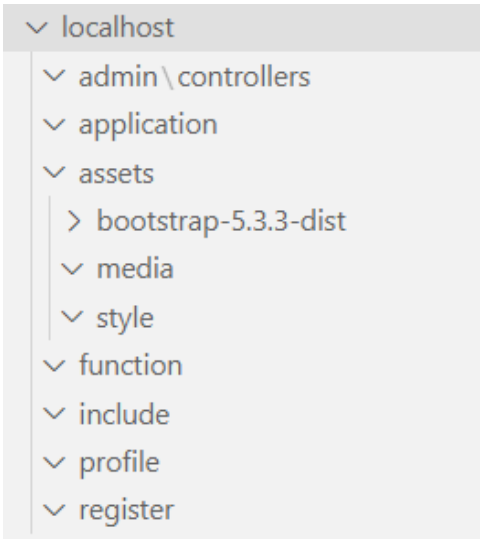
5. Панель администратора. Доступ в панель администратора осуществляется по логину adminka и паролю password. В панели администратора видны все заявки (ФИО заявителя, контактные данные и иные свойства заявки). Администратор может сменить статус на «в работе», «выполнено» или «отменено» (обязательно с указанием причины отмены).

Настройки Open Server Panel:

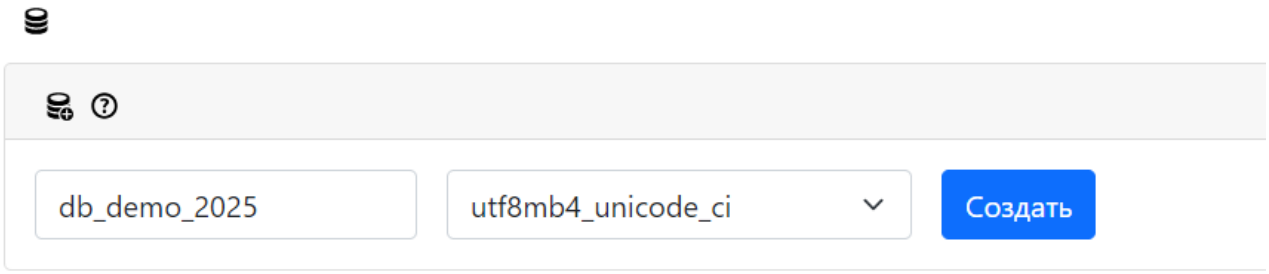


Этап 1. Подготовка. Создание базы данных

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



2. Переносим файлы **bootstrap** в папку **assets/bootstrap**.
3. Запускаем PhpMyAdmin и создаем базу данных **db_demo_2025**.

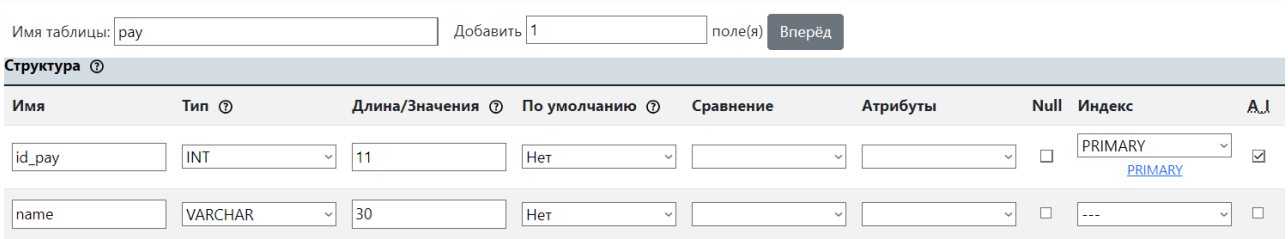


4. База данных будет содержать **6 таблиц**.
   - **application** - заявки;
   - **pay** - методы оплаты;
   - **role** - роли пользователя;
   - **service** - услуги;
   - **status** - статусы заявок;
   - **user** - пользователи.

   Таблицы имеют соответствующие первичные и внешние ключи для создания связей между таблицами:
   - **application**: имеет внешние ключи к таблицам **user**, **pay**, **service**, и **status**.
   - **user**: имеет внешний ключ к таблице **role**.

Далее приведем два варианта: описание полей таблиц для ручного создания в конструкторе PhpMyAdmin и код для создания через SQL-запрос(ы) к базе данных. Выберите удобный для себя вариант и заполните таблицы.

5. Таблица **pay**.



- **id_pay**: int(11) NOT NULL AUTO_INCREMENT — первичный ключ, идентификатор метода оплаты.
- **name**: varchar(30) NOT NULL — название метода оплаты, не пустое.
  Пример данных:
  - Наличные
  - Банковская карта

Листинг 1.1 Структура таблицы pay

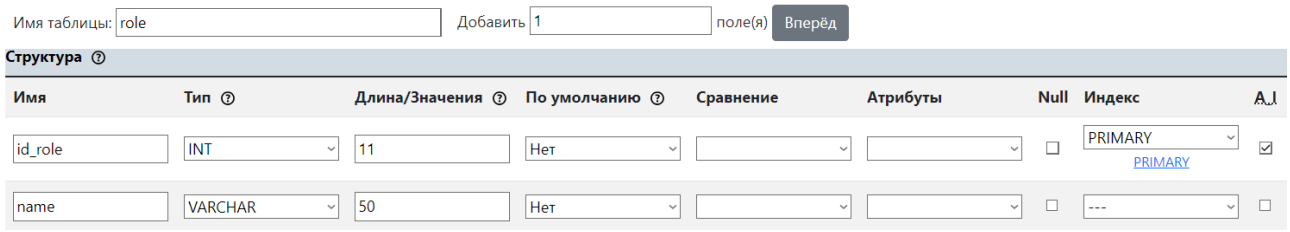
```
CREATE TABLE pay (
    id_pay INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
```

```
    PRIMARY KEY (id_pay)
);
```

Листинг 1.2 Заполнение таблицы pay

```
INSERT INTO pay (name) VALUES
('Наличные'),
('Банковская карта');
```

6. Таблица **role.**



- **id_role**: int(11) NOT NULL AUTO_INCREMENT — первичный ключ, идентификатор роли.
- **name**: varchar(50) NOT NULL — название роли, не пустое.
  Пример данных:
  - user
  - admin

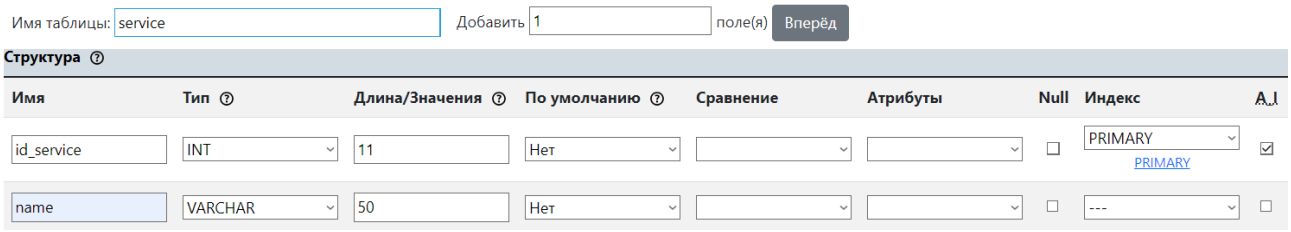
Листинг 1.3 Структура таблицы role

```
CREATE TABLE role (
    id_role INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    PRIMARY KEY (id_role)
);
```

Листинг 1.4 Заполнение таблицы role

```
INSERT INTO role (name) VALUES
('user'),
('admin');
```

7. Таблица **service**.



- **id_service**: int(11) NOT NULL AUTO_INCREMENT — первичный ключ, идентификатор услуги.
- **name**: varchar(50) NOT NULL — название услуги, не пустое.

Пример данных:

- Иная услуга
- Общий клининг
- Генеральная уборка
- Послестроительная уборка
- Химчистка ковров и мебели

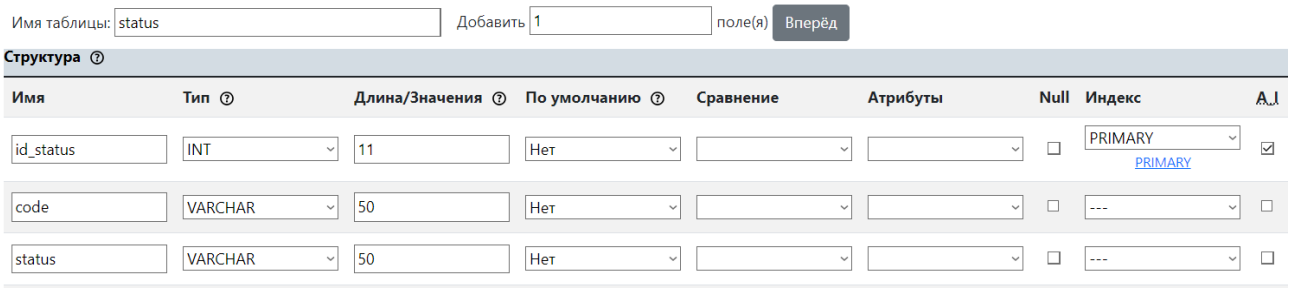
Листинг 1.5 Структура таблицы service

```
CREATE TABLE service (
    id_service INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    PRIMARY KEY (id_service)
);
```

Листинг 1.6 Заполнение таблицы service

```
INSERT INTO service (name) VALUES
('Иная услуга'),
('Общий клининг'),
('Генеральная уборка'),
('Послестроительная уборка'),
('Химчистка ковров и мебели');
```

8. Таблица **status**.



- **id_status**: int(11) NOT NULL AUTO_INCREMENT — первичный ключ, идентификатор статуса.
- **code**: varchar(50) NOT NULL — код статуса, не пустое.
- **status**: varchar(50) NOT NULL — название статуса, не пустое.

  Пример данных:
  - ('new', 'Новая')
  - ('canceled', 'Отменена')
  - ('confirmed', 'Выполнена')
  - ('process', 'В работе')

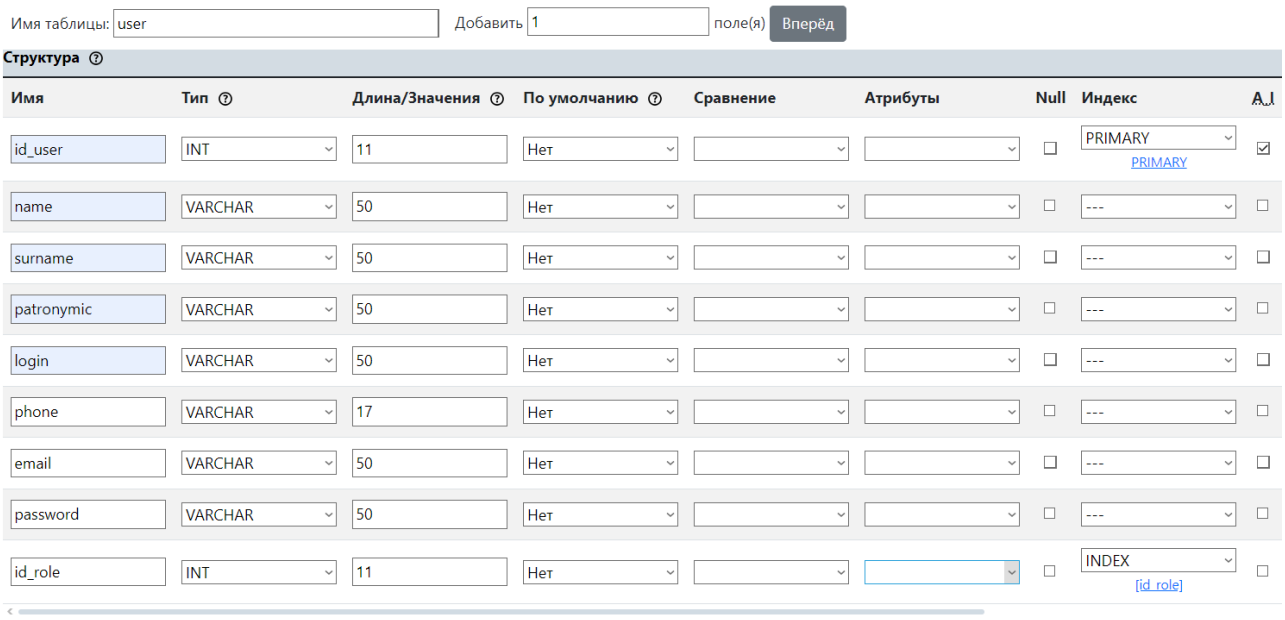
Листинг 1.7 Структура таблицы status

```
CREATE TABLE status (
  id_status int(11) NOT NULL AUTO_INCREMENT,
  code varchar(50) NOT NULL,
  status varchar(50) NOT NULL,
  PRIMARY KEY (id_status)
)
```

Листинг 1.8 Заполнение таблицы status

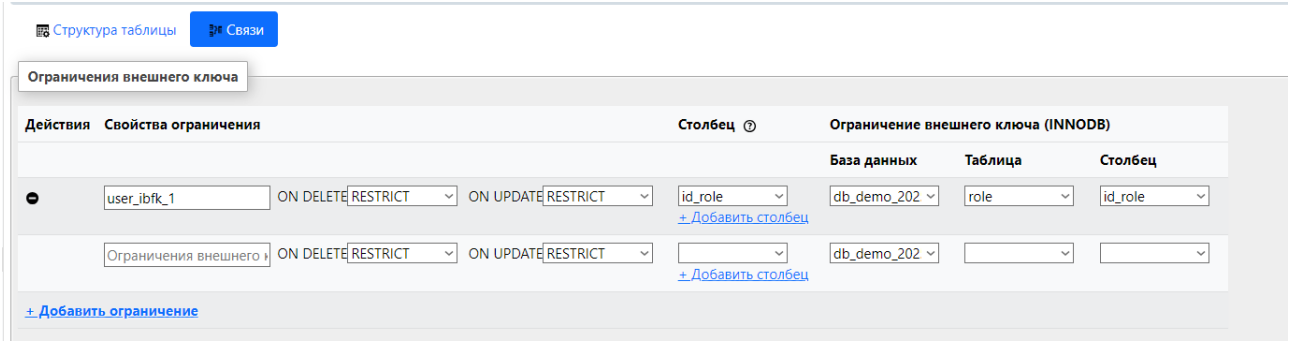
```
INSERT INTO status (`code`, `status`) VALUES
('new', 'Новая'),
('canceled', 'Отменена'),
('confirmed', 'Выполнена'),
('process', 'В работе');
```

9. Таблица **user**.



- **id_user**: int(11) NOT NULL AUTO_INCREMENT — первичный ключ, идентификатор пользователя.
- **name**: varchar(50) NOT NULL — имя пользователя.
- **surname**: varchar(50) NOT NULL — фамилия пользователя.
- **patronymic**: varchar(50) NOT NULL — отчество пользователя.
- **login**: varchar(50) NOT NULL — логин для входа.
- **phone**: varchar(17) NOT NULL — телефонный номер.
- **email**: varchar(50) NOT NULL — адрес электронной почты.
- **password**: varchar(50) NOT NULL — пароль пользователя.
- **id_role**: int(11) NOT NULL — идентификатор роли, связанный с таблицей **role**.

На вкладке **Связи** устанавливаем связь с таблице **role**.



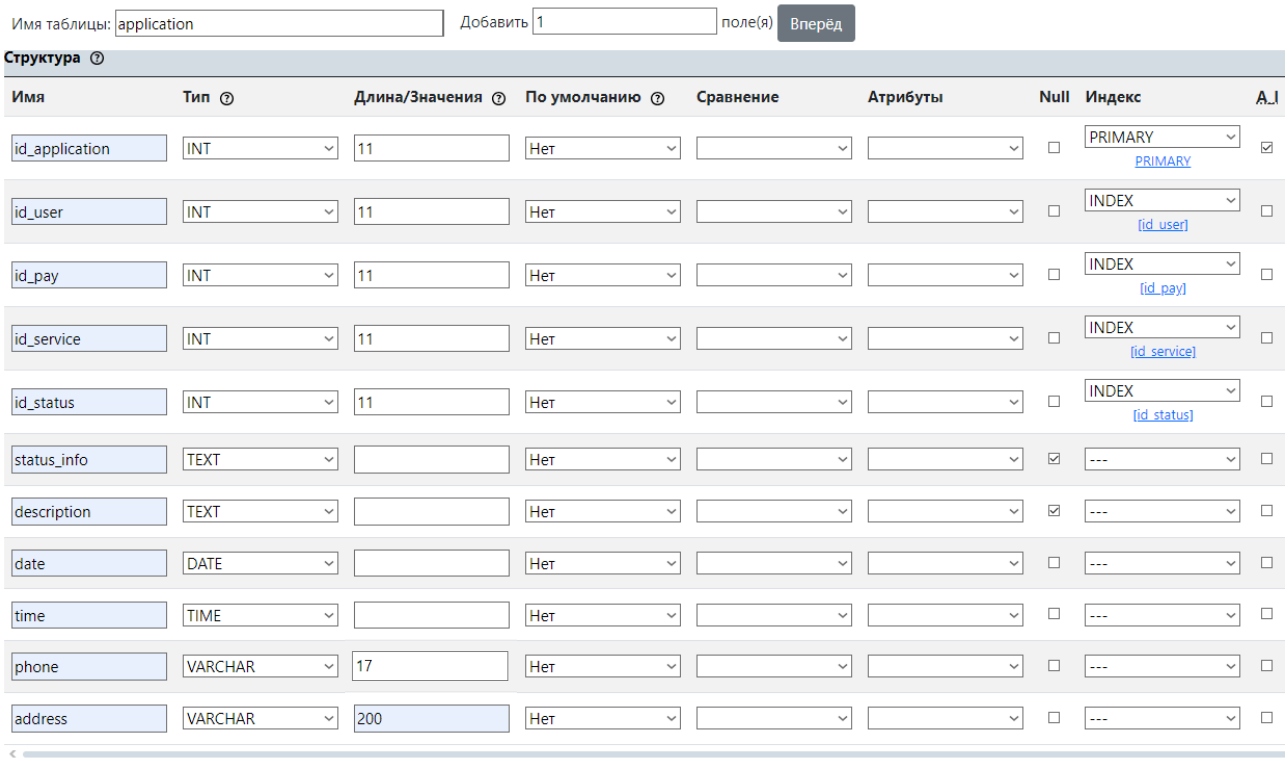
Листинг 1.9 Структура таблицы user

```
CREATE TABLE user (
  id_user int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) NOT NULL,
  surname varchar(50) NOT NULL,
  patronymic varchar(50) NOT NULL,
  login varchar(50) NOT NULL,
  phone varchar(17) NOT NULL,
  email varchar(50) NOT NULL,
  password varchar(50) NOT NULL,
  id_role int(11) NOT NULL,
  PRIMARY KEY (id_user),
  FOREIGN KEY (id_role) REFERENCES role(id_role)
)
```

Важно! Заполнить таблицу данными можно позже через форму регистрации. Внимательно посмотрите в задании наличие каких пользователей и с какими данными обязательно.
В примерном задании это логин – **adminka** и пароль **password**, остальные регистрационные данные на ваше усмотрение. Например,

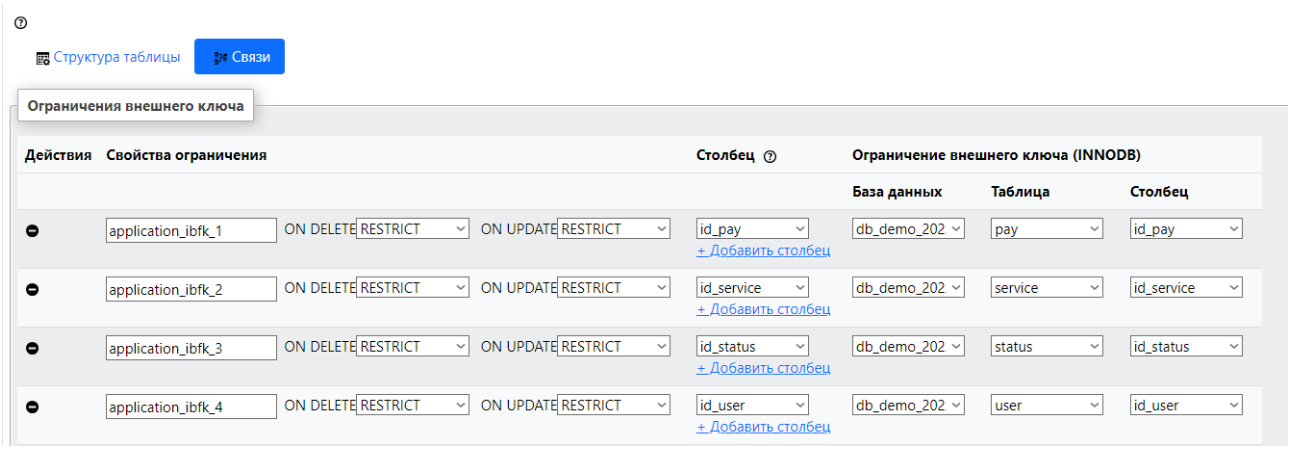
id_user	name	surname	patronymic	login	phone	email	password	id_role
1	admin	admin	admin	adminka	89999999999	admin@admin.ru	password	2
2	user	user	user	user	89999999999	user@user.ru	123456	1

10. Таблица **application.**



- **id_application**: int(11) — первичный ключ, идентификатор заявки, автозаполнение, не пустое.
- **id_user**: int(11) — идентификатор пользователя, связанный с таблицей user, не пустое.
- **id_pay**: int(11) NOT NULL — идентификатор способа оплаты, связанный с таблицей pay, не пустое.
- **id_service**: int(11) NOT NULL — идентификатор услуги, связанный с таблицей service, не пустое.
- **id_status**: int(11) NOT NULL — идентификатор статуса заявки, связанный с таблицей status, не пустое.
- **status_info**: text — дополнительная информация о статусе.
- **description**: text — описание заявки.
- **date**: date NOT NULL — дата создания или выполнения заявки, не пустое.
- **time**: time NOT NULL — время заявки, не пустое.
- **phone**: varchar(17) NOT NULL — телефонный номер.
- **address**: varchar(200) NOT NULL — адрес исполнения заявки.

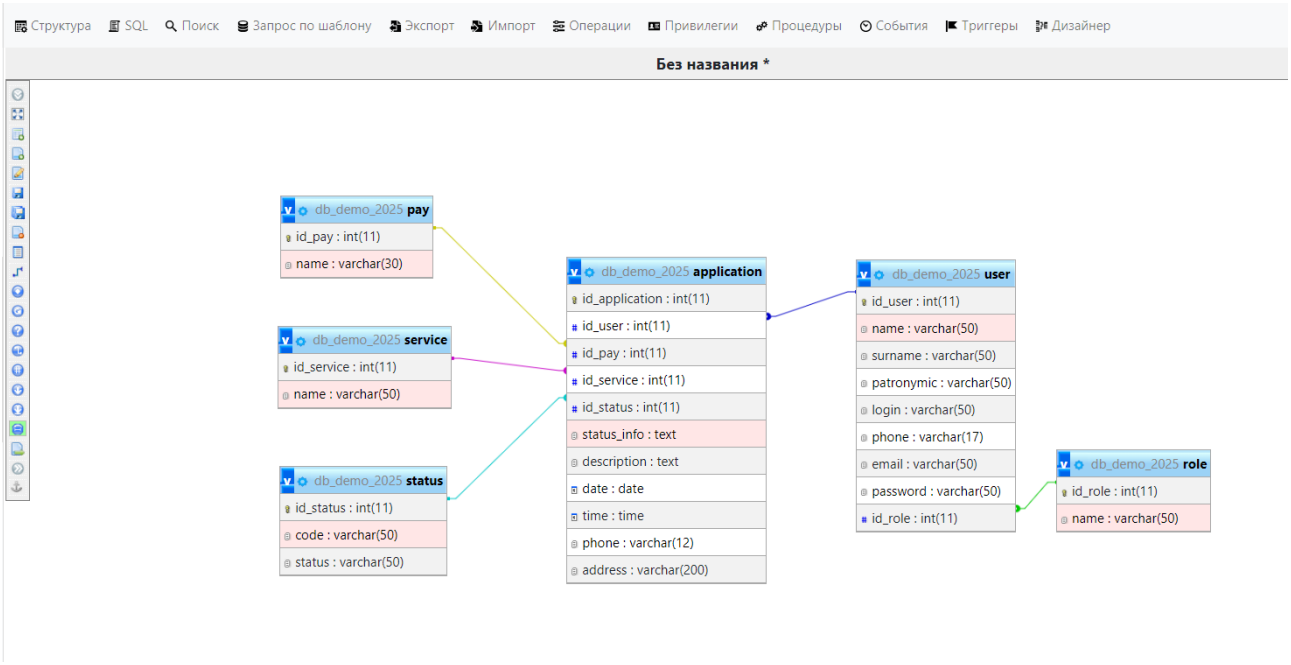
На вкладке **Связи** устанавливаем связь с таблицами.



Листинг 1.10 Структура таблицы application

```
CREATE TABLE application (
  id_application int(11) NOT NULL AUTO_INCREMENT,
  id_user int(11) NOT NULL,
  id_pay int(11) NOT NULL,
  id_service int(11) NOT NULL,
  id_status int(11) NOT NULL,
  status_info text,
  description text,
  date date NOT NULL,
  time time NOT NULL,
  phone varchar(17) NOT NULL,
  address varchar(200) NOT NULL,
  PRIMARY KEY (id_application),
  FOREIGN KEY (id_user) REFERENCES user(id_user),
  FOREIGN KEY (id_pay) REFERENCES pay(id_pay),
  FOREIGN KEY (id_service) REFERENCES service(id_service),
  FOREIGN KEY (id_status) REFERENCES status(id_status)
)
```

Проверьте схему БД на вкладке Дизайнер.



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_2025");

    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 **assets/style/**. Наполняем файл базовыми стилями.

Листинг 2.3 Собственные стили style.css

```css
/* Минимальная высота секции */
section{ min-height: 82vh; }
```

```css
/* Анимация заголовков */
@keyframes titles {
    0%{ font-size: 0.7em; }
    100%{ font-size: 64px; }
}

/* Использование анимации заголовков */
h3{
    height: 80px;
    animation: titles 0.5s linear 1;
}

/* Появление поля для описание другого типа услуги на странице подачи заявки */
div.service{ display: none; }
input#service:checked+label ~ div.service{ display: block; }
```

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

Файл header.php

1. Создаем файл **header.php** в папке **include/**.
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

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Мой Не Сам</title>
    <link rel="stylesheet" href="/assets/bootstrap-5.3.3-dist/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

```html
<body>
    <main>
        <header>
                //здесь будет вставка элемента nav из Zeal
        </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 `bg-success` и добавляем `navbar-dark`.

Меняем содержимое ссылки по умолчанию на собственное, правим размеры и расположение блоков в зависимости от собственного . Получаем:

```
<a class="navbar-brand" href="/">
    <img src="/assets/media/logo.png" alt="Logo" width="50" height="50"
        class="d-inline-block">
    Мой Не Сам
</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>
        <header>
                <!-- начало вставки из Zeal -->

                <nav class="navbar navbar-expand-lg bg-success navbar-dark p-3 fs-4">
                    <div class="container-fluid">
                        <a class="navbar-brand fs-3" href="/">
                          <img src="/assets/media/logo.png" alt="Logo" width="30"
height="30" class="d-inline-block align-text-top">
                          Мой Не Сам
                        </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>
        </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
    session_start();
    $menu = "";
    if(isset($_SESSION['login'])){
        if($_SESSION['role'] == "admin"){
            $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="/">Вход</a>
                 </li>
                 <li class="nav-item">
                   <a class="nav-link" href="/register/">Регистрация</a>
                 </li>';
    }
?>
<!DOCTYPE html>
   ...
```

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

Файл footer.php

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

Листинг 2.9 Подвал сайта

```
        <footer class="p-3 bg-dark text-light">
          <p class="mb-0">Мой Не Сам</p>
          <p>Сделано в качестве демонстрационного экзамена</p>
        </footer>
    </main>
    <script src="/assets/bootstrap-5.3.3-dist/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 **корневой директории**.

2. Подключаем «шапку» и «подвал» через **include**.

Листинг 3.1 Страница index.php

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

    <?php
        include ("include/footer.php");
    ?>
```

3. Устанавливаем блоки для размещения контента.

Листинг 3.2 Страница index.php. Контент

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

4. Добавляем заголовок страницы.

Листинг 3.3 Страница index.php. Контент

```
...
<h3 class="text-center display-3">Вход</h3>
...
```

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 index.php. Форма авторизации

```
<form action="/admin/controllers/login.php" method="post" class="w-50 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 px-3 rounded-pill fs-3"
              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 px-3 rounded-pill fs-3"
              id="password" name="password" required>

   </div>

   <input type="submit" class="btn btn-success my-3 w-100 shadow-sm p-3
          fs-3 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 index.php. Проверка логина

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

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

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

Листинг 3.6 Страница index.php Вывод сообщения об ошибке

```
<section>
    <div class="container py-3">

        <?php
            if(isset($_GET['error'])){
                echo "<div class='border border-danger text-danger
                                   text-center p-3 my-3 mx-auto w-50'>
                {$_GET['error']}
                </div>";
            }
        ?>

        <h1 ...>Вход</h1>
```

Страница Регистрация

1. Копируем страницу **Вход** *(index.php)* в папку **register/**.
2. Исправляем адрес подключения header и footer.

```
include ("../include/header.php");
include ("../include/footer.php");
```

3. Меняем заголовок страницы на **Регистрация**.
4. 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 символы кириллицы и пробелы pattern="[а-яА-ЯЁё\s]{2,40}"
Имя	text	name	обязательное символы кириллицы и пробелы pattern="[а-яА-ЯЁё\s]{2,40}"
Отчество	text	patronymic	обязательное символы кириллицы и пробелы pattern="[а-яА-ЯЁё\s]{2,40}"
Логин	text	login	обязательное, уникальное
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

Листинг 3.7 Страница register/index.php. Форма регистрации

```
<section>
    <div class="container py-3">
        <h3 class="display-3 text-center">Регистрация</h3>
        <?php
            if(isset($_GET['error'])){
                echo "<div class='w-50 mx-auto text-danger
                                border border-danger p-3 my-3'>
                    {$_GET['error']}
                </div>";
            }
        ?>
        <form action="/admin/controllers/registration.php" method="post" class="w-50
mx-auto">
            <div class="mb-3">
                <label for="name" class="form-label fs-3">Имя</label>
                <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="name" name="name" pattern="[а-яА-ЯЁё\s]{2,40}" required>
            </div>
            <div class="mb-3">
                <label for="surname" class="form-label fs-3">Фамилия</label>
                <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="surname" name="surname" required pattern="[А-Яа-яЁё\s]{2,40}">
            </div>
            <div class="mb-3">
                <label for="patronymic" class="form-label fs-3">Отчество</label>
                <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="patronymic" name="patronymic" required pattern="[А-Яа-яЁё\s]{2,40}">
            </div>
            <div class="mb-3">
                <label for="login" class="form-label fs-3">Логин</label>
                <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="login" name="login" required pattern="[A-Za-z0-9]{2,40}">
            </div>
            <div class="mb-3">
                <label for="phone" class="form-label fs-3">Телефон</label>
                <input type="tel" class="form-control fs-3 rounded-pill shadow-sm px-3"
id="phone" name="phone" required pattern="\+?[0-9(\)\-\]+" placeholder="+7(XXX)-XXX-XX-
XX" minlength="17" maxlength="17">
            </div>
            <div class="mb-3">
                <label for="email" class="form-label fs-3">Email</label>
                <input type="email" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="email" name="email" required>
```

```
            </div>
            <div class="mb-3">
                <label for="password" class="form-label fs-3">Пароль</label>
                <input type="password" class="form-control fs-3 rounded-pill shadow-sm
px-3" id="password" name="password" required minlength="6">
            </div>
            <button type="submit" class="btn btn-success p-3 rounded-pill shadow-sm fw-
bold w-100 fs-3 my-3">Зарегистрироваться</button>
        </form>
    </div>
</section>
```

5. Для запуска скрипта валидации (который прикрепили в подвале сайта **include/footer.php**) и вывода сообщений об ошибках валидации добавим:
    - для формы: класс – **.needs-validation** и атрибут **novalidate**.
    - **для валидируемых полей**:
        - блок с классом **.form-text** и подсказкой для пользователя (не забудьте редактировать id);
        - 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=" w-50 mx-auto needs-validation" novalidate>

    <div class="mb-3">
      <label for="name" class="form-label fs-3">Имя</label>
      <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-3"
id="name" name="name" pattern="[а-яА-ЯЁё\s]{2,40}" required>
        <div id="nameHelp" class="form-text fs-6">
            Кириллица и пробелы, от 2 до 40 символов
        </div>
        <div class="invalid-feedback fs-4">
            Проверьте введенные данные
        </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 Заявка. Создание заявки

1. Копируем страницу **Вход** *(index.php)* в папку **application/**.
2. Исправляем адрес подключения header и footer.

```
include ("../include/header.php");
include ("../include/footer.php");
```

3. Подключаем файл **function.php**.
4. Редактируем код проверки логина.

Листинг 3.9 Страница application/index.php. Проверка логина

```
include ("../function/function.php");

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

5. Меняем заголовок страницы на **Подача заявки**.
6. Удаляем вывод ошибки авторизации.
7. 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.

Поле / подпись	Тип	Ключевое слово	Параметры
Адрес	text	address	обязательное
Телефон	tel	phone	обязательное pattern="\+?[0-9(\)\-\]+" placeholder="+7(XXX)-XXX-XX-XX" длина – 17 символов
Желаемая дата	date	date	обязательное
Желаемое время	time	time	обязательное
Вид услуги	select	category	содержимое определяет функция fnOutService()
Иная услуга	checkbox	service	при установленном чекбоксе появляется дополнительное текстовое поле Описание
Описание	textarea	description	обязательное
Способ оплаты	select	pay	содержимое определяет функция fnOutPay()

Заметьте, что на данной странице присутствуют изображения. Так же на этой странице используется несколько функций для вывода справочной информации: fnOutService, fnOutPay.

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

```
<section>
    <div class="container py-3">
        <h3 class="display-3 text-center">Подача заявки</h3>
        <div class="image w-50 mx-auto my-4">
            <img src="/assets/media/application.jpg" alt="application" class="w-100
rounded-pill">
        </div>
        <form action="/admin/controllers/add_application.php" method="post" novalidate
class="needs-validation w-50 mx-auto">
            <div class="mb-3">
                <label for="address" class="form-label fs-3">Адрес</label>
                <input type="text" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="address" name="address" required >
                <div class="invalid-feedback fs-4">
                   Поле обязательно для заполнения
                </div>
            </div>
            <div class="mb-3">
                <label for="phone" class="form-label fs-3">Телефон</label>
                <input type="tel" class="form-control fs-3 rounded-pill shadow-sm px-3"
id="phone" name="phone" required pattern="\+?[0-9(\)\-\]+" placeholder="+7(XXX)-XXX-XX-
XX" minlength="17" maxlength="17">
                <div class="invalid-feedback fs-4">
                   Поле обязательно для заполнения
                </div>
            </div>
            <div class="mb-3">
                <label for="date" class="form-label fs-3">Желаемая дата</label>
                <input type="date" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="date" name="date" required>
                <div class="invalid-feedback fs-4">
                   Поле обязательно для заполнения
                </div>
            </div>
            <div class="mb-3">
                <label for="time" class="form-label fs-3">Желаемое время</label>
                <input type="time" class="form-control fs-3 rounded-pill shadow-sm px-
3" id="time" name="time" required>
                <div class="invalid-feedback fs-4">
                   Поле обязательно для заполнения
                </div>
            </div>

            <div class="mb-3">
                <label for="category" class="form-label fs-3">Вид услуги</label>
                <select class="form-select form-select-lg fs-3 rounded-pill shadow-sm
px-3" id="category" name="category">
                    <?=fnOutService()?>
                </select>
            </div>
```

```
            <div class="mb-3 form-check p-0">
                <input class="form-check-input p-3 ms-2" type="checkbox" value=""
id="service">
                <label class="form-check-label fs-3 ms-3" for="service">
                    Иная услуга
                </label>
                <div class="service">
                    <label for="description" class="form-label fs-3">Описание</label>
                    <textarea class="form-control fs-3 rounded-5 shadow-sm" rows="6"
id="description" name="description"></textarea>
                    <div class="invalid-feedback fs-4">
                    Поле обязательно для заполнения
                    </div>
                </div>
            </div>

            <div class="mb-3">
                <label for="pay" class="form-label fs-3">Способ оплаты</label>
                <select class="form-select form-select-lg fs-3 rounded-pill shadow-sm
px-3" id="pay" name="pay">
                    <?=fnOutPay()?>
                </select>
            </div>

            <button type="submit" class="btn btn-success p-3 rounded-pill shadow-sm fw-
bold w-100 fs-3 my-3">Сформировать заявку</button>
        </form>
    </div>
</section>
```

Этап 4. Функционал входа, выхода, регистрации

Функционал Вход

1. Создаем страницу **login.php** в папке **admin/controllers/**.
2. Стартуем 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 *error* **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");
```

```
    $sql = "SELECT `login`, `role`.`name` AS `role` FROM `user` INNER JOIN `role`
ON `role`.`id_role` = `user`.`id_role` WHERE `login` = '{$_POST['login']}' AND
`password` = '{$_POST['password']}'";

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

    if(!$result->num_rows){
        header('location: /?error=Не верный логин или пароль');
        exit;
    }else{
        $user = $result->fetch_assoc();

        $_SESSION['login'] = $user['login'];
        $_SESSION['role'] = $user['role'];

        if($_SESSION['role'] != 'admin'){
            header('location: /profile/');
            exit;
        }else{
            header('location: /admin/');
            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();
    session_unset();
    session_destroy();
    header('location: /');
    exit;
?>
```

Важно! Очистить содержимое сессии можно и другим способом, например функцией **unset()** очистить каждый элемент массива SESSION.

Функционал Регистрация

1. Создаем страницу **regisration.php** в папке **admin/controllers/**.
2. Стартуем сессию. Подключаемся к базе.
3. Проверяем логин на уникальность.

a. Если логин повторяется, переходим на страницу **register/index.php** с отображением ошибки.

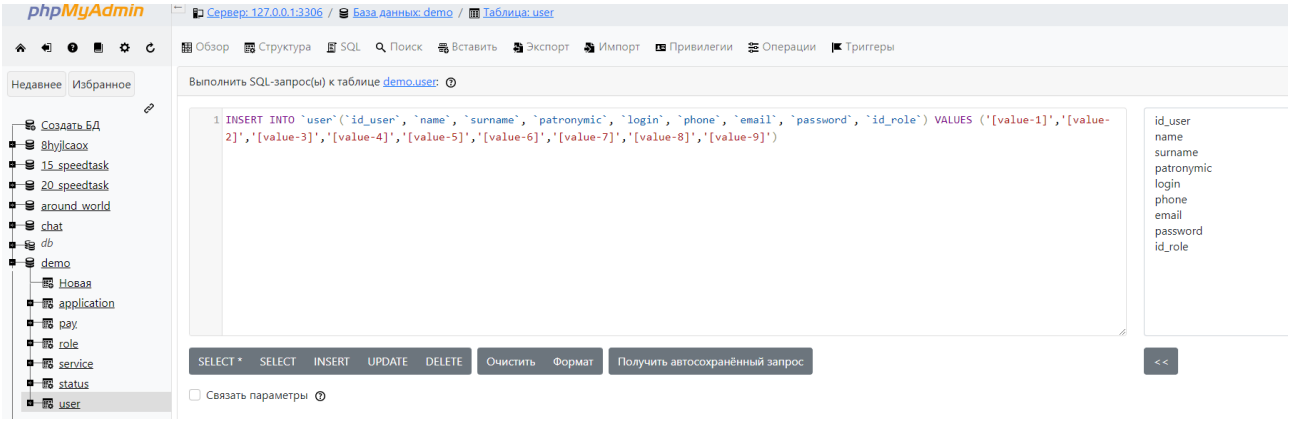
b. Если совпадений не найдено, записываем в базу данные, получаемые из формы.

   - В сессию записываем логин и роль пользователя.
   - Переходим на страницу **Личный кабинет**.

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

```
<?php
    session_start();
    include("../../function/connect.php");

    $sql = "SELECT * FROM `user` WHERE `login` = '{$_POST['login']}'";

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

    if($result->num_rows){
        header('location: /register/?error=Пользователь с таким login уже
зарегистрирован');
        exit;
    }else{
        $sql_role = "SELECT `id_role` FROM `role` WHERE `name` = 'user'";
        $role = $connect->query($sql_role)->fetch_assoc()['id_role'];

        $sql = sprintf("INSERT INTO `user`(`id_user`, `name`, `surname`,
`patronymic`,`login`, `phone`, `email`, `password`, `id_role`) VALUES
(NULL,'%s','%s','%s','%s','%s','%s','%s','%s')",
        $_POST['name'],
        $_POST['surname'],
        $_POST['patronymic'],
        $_POST['login'],
        $_POST['phone'],
        $_POST['email'],
        $_POST['password'],
        $role);
        if($connect->query($sql)){
            $_SESSION['login'] = $_POST['login'];
            $_SESSION['role'] = 'user';
            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 заявки** *(application/index.php)* в папку **profile/**.
2. Меняем заголовок страницы на **Личный кабинет**.
3. От формы оставляем только кнопку **Сформировать заявку**. Внесем в неё изменения.

Листинг 5.2 Ссылка Подать заявление

```
...
// было
<input type="submit"
   class=" btn btn-success my-3 w-100 shadow-sm p-3 fs-3 rounded-pill shadow-sm fw-bold
w-100 fs-3 my-3"
   value="Сформировать заявку">

//стало
<a href="/application/"
   class="btn btn-success my-3 w-100 shadow-sm p-3 fs-3 rounded-pill shadow-sm fw-bold
w-100 fs-3 my-3

">
    Сформировать заявку
</a>
...
```

4. В контенте страницы вместо формы выводим результаты выполнения функции `fnOutCardProfile`.
   Сами функции описаны в файле **function.php** (см. далее).

Листинг 5.3 Вывод данных из функций

```
<section>
    <div class="container py-3">
        <h3 ...> Личный кабинет</h3>
        <?php
            echo fnOutCardProfile($_SESSION['login']);
        ?>
        <a ...>Оставить заявку</a>
    </div>
</section>
```

Файл function.php

1. Создаем файл **function.php** в папке **function/**.
2. Файл содержит подключение к базе данных и пять функции:
   - **fnOutService** – возвращает список возможных услуг.
   - **fnOutPay** – возвращает список возможных способов оплаты.
   - **fnOutCardProfile** – используя логин, получает id пользователя и выводит все его обращения в виде карточек.
   - **fnOutCardAdmin** – для администратора выводит карточки обращений всех пользователей с дополнительными кнопками смены статуса (выполнено, отменено, в работе).
   - **fnOutCardAdminStatus** – используя код статуса выводит карточки обращений всех пользователей в зависимости от статуса.
     Рассмотрим ее подробнее на следующем этапе проектирования.

Листинг 5.4 Структура function.php

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

    function fnOutService(){...}
    function fnOutPay(){...}
    function fnOutCardProfile($login) {...}
    function fnOutCardAdmin(){...}
    function fnOutCardAdminStatus($code) {...}
?>
```

fnOutService

Листинг 5.5 Функция fnOutService

```php
function fnOutService(){
    global $connect;
    $sql = "SELECT * FROM `service` WHERE `id_service` > 0";
    $result = $connect->query($sql);
    $data = "";
    foreach($result as $item){
        $data .= sprintf('<option value="%s">%s</option>', $item['id_service'],
                                                           $item["name"]);
    }
    return $data;
}
```

fnOutPay

Листинг 5.6 Функция fnOutPay

```php
function fnOutPay(){
    global $connect;
    $sql = "SELECT * FROM `pay`";
    $result = $connect->query($sql);
    $data = "";
    foreach($result as $item){
        $data .= sprintf('<option value="%s">%s</option>', $item['id_pay'],
$item["name"]);
    }
```

```
    return $data;
}
```

fnOutCardProfile

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 5.7 Функция fnOutCardProfile

```
function fnOutCardProfile($login){
    global $connect;
    $sql = "SELECT * FROM `user` WHERE `login` = '{$login}'";
    $result = $connect->query($sql);
    $user = $result->fetch_assoc();
    $sql = "SELECT `id_application`, `status`,`status_info`, `date`, `time`,
           `pay`.`name` AS `pname`, `service`.`name` AS `sname`, `code`,
            `description`, `service`.`id_service` FROM `application` INNER JOIN
            `service` ON `application`.`id_service` = `service`.`id_service`
            INNER JOIN `pay` ON `application`.`id_pay` = `pay`.`id_pay`
            INNER JOIN `status` ON `application`.`id_status` =
            `status`.`id_status` WHERE `application`.`id_user` =
            {$user['id_user']} ORDER BY `id_application` DESC";
    $result = $connect->query($sql);
    if(!$result->num_rows){
        $data = "<h4 class='display-6 text-center'>Заявок не найдено<h4>";
    }else{
        $data = "<div class='cards my-4 row row-cols-1 row-cols-md-3 g-4'>";

        // место для вывода карточек Листинг 5.8

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

    return $data;
}
```

Выводим информацию об обращениях. Код карточки можно скопировать из Zeal, удалив ненужные элементы**:** атрибут style, изображение и ссылку: *Bootstrap5 – Guides – Cards – раздел* ***Grid cards.***

5. Простой вариант вывода всех обращений пользователя реализуем через цикл **foreach**.

Листинг 5.8 Функция fnOutCardProfile. Простая реализация

```
foreach($result as $item){
    $data .= sprintf('
        <div class="col">
            <div class="card mb-3">
                <div class="card-body">
                <h5 class="card-title fs-3">Заявка №%s</h5>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Статус заявки</span>: %s
```

```
                    </p>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Категория</span>: %s
                    </p>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Дата</span>: %s
                    </p>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Время</span>: %s
                    </p>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Способ оплаты</span>: %s
                    </p>
                    <p class="card-text mb-2">
                        <span class="fw-semibold">Описание</span>: %s
                    </p>
                </div>
            </div>
        </div>',
        $item['id_application'],
        $item['status'],
        $item['sname'],
        $item['date'],
        $item['time'],
        $item['pname'],
        ($item['id_service'] == "0") ? $item['description'] : $item['sname']
    );
}
```

Есть 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-info-emphasis** для карточек со статусом «В работе», **text-body-tertiary** для отмененных и **success** для выполненных.

Обратите внимание что на карточке со статусом «Отменено» добавляется пункт «Причина отмены».

```
            <p class="card-text mb-2">
                <span class="fw-semibold">Причина отмены</span>:%s
            </p>
```

Содержимое карточек неизменно (см. Листинг 5.8), редактируем только классы контейнера **.card.**

Листинг 5.9 Функция fnOutCardProfile. Изменение стилей карточек

```
foreach($result as $item){
   if($item['code'] == "new"){
        $data .= sprintf('
        <div class="col">
```

```
            <div class="card mb-3">
               // см. код Листинг 5.8
            </div>
        </div>',
        $item['id_application'],
        $item['status'],
        $item['sname'],
        $item['date'],
        $item['time'],
        $item['pname'],
        ($item['id_service'] == "0") ? $item['description'] : $item['sname']);
    }elseif($item['code'] == "canceled"){
        $data .= sprintf('
        <div class="col">
            <div class="card mb-3 text-body-tertiary">
                // см. код Листинг 5.8 + блок Причина отмены
            </div>
        </div>',
        $item['id_application'],
        $item['status'],
        $item['status_info'],
        $item['sname'],
        $item['date'],
        $item['time'],
        $item['pname'],
        ($item['id_service'] == "0") ? $item['description'] : $item['sname']);
    }elseif($item['code'] == "confirmed"){
        $data .= sprintf('
        <div class="col">
            <div class="card mb-3 text-success">
                // см. код Листинг 5.8
            </div>
        </div>',
        $item['id_application'],
        $item['status'],
        $item['sname'],
        $item['date'],
        $item['time'],
        $item['pname'],
        ($item['id_service'] == "0") ? $item['description'] : $item['sname']);
    }elseif($item['code'] == "process"){
        $data .= sprintf('
        <div class="col">
            <div class="card mb-3 text-info-emphasis">
                // см. код Листинг 5.8
            </div>
        </div>',
        $item['id_application'],
        $item['status'],
        $item['sname'],
        $item['date'],
```

```
        $item['time'],
        $item['pname'],
        ($item['id_service'] == "0") ? $item['description'] : $item['sname']);
    }
}
```

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
    // Подключение шапки сайта
    include("../include/header.php");
    // Подключение функций
    include("../function/function.php");
    // Проверка авторизован ли пользователь
    if(!isset($_SESSION['login'])){
        header('location: /');
        exit;
    }
    // Проверка является ли пользователь администратором
    if($_SESSION['role'] != 'admin'){
        header('location: /profile/');
        exit;
    }
?>
```

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

Листинг 6.2 Страница admin/index.php

```
<section>
    <div class="container py-3">
        <h3 class="text-center display-3">
            Панель администратора
        </h3>
        <?php echo fnOutCardAdmin();?>
    </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 `fnOutCardProfile($login)` на `fnOutCardAdmin()`.

3. Редактируем функцию. Удаляем блок определения пользователя.


```
function fnGetCardAdmin(){
    global $connect;
    $sql = "SELECT * FROM `user` WHERE `login` = '{$login}'";
    $result = $connect->query($sql);
    $user = $result->fetch_assoc();
    $sql = "SELECT `id_application`, `status`,`status_info`, `date`, `time`, `pay`.`name`
            AS `pname`, `service`.`name` AS `sname`, `code`, `description`, `service`.`id_service` FROM `application`
            INNER JOIN `service` ON `application`.`id_service` = `service`.`id_service`
            INNER JOIN `pay` ON `application`.`id_pay` = `pay`.`id_pay`
            INNER JOIN `status` ON `application`.`id_status` = `status`.`id_status` WHERE `application`.`id_user` = {$user['id_user']} ORDER BY `id_application` DESC";
    $result = $connect->query($sql);
    if(!$result->num_rows){
        $data = "<h4 class='display-6 text-center'>Заявок не найдено<h4>";
    }else{
        $data = "<div class='cards my-4 row row-cols-1 row-cols-md-3 g-4'>";
```

4. Дополняем перечень выбираемых данных информацией о Фамилии, Имени, Отчестве заявителя, номер телефона, указанный в заявке, а также адрес.


```
function fnGetCardAdmin(){
    global $connect;
    $sql = "SELECT `id_application`, `status`,`status_info`, `date`, `time`, `pay`.`name`
            AS `pname`, `service`.`name` AS `sname`, `code`, `description`, `service`.`id_service` FROM `application`
            INNER JOIN `service` ON `application`.`id_service` = `service`.`id_service`
            INNER JOIN `pay` ON `application`.`id_pay` = `pay`.`id_pay`
            INNER JOIN `status` ON `application`.`id_status` = `status`.`id_status` WHERE `application`.`id_user` = {$user['id_user']} ORDER BY `id_application` DESC";
    $sql = "SELECT `id_application`, `user`.`name` AS `u_name`, `surname`, `patronymic`, `status`,`status_info`, `date`, `time`, `address`, `pay`.`name`
            AS `pname`, `service`.`name` AS `sname`, `code`, `description`, `service`.`id_service`, `application`.`phone`AS `a_phone` FROM `application`
            INNER JOIN `service` ON `application`.`id_service` = `service`.`id_service`
            INNER JOIN `pay` ON `application`.`id_pay` = `pay`.`id_pay`
            INNER JOIN `status` ON `application`.`id_status` = `status`.`id_status`
            INNER JOIN `user` ON `application`.`id_user` = `user`.`id_user` ORDER BY `id_application` DESC";
```

Листинг 6.3 Выборка данных

```
$sql = "SELECT `id_application`, `user`.`name` AS `u_name`, `surname`, `patronymic`,
`status`,`status_info`, `date`, `time`,`address`, `pay`.`name` AS `pname`,
`service`.`name` AS `sname`, `code`, `description`, `service`.`id_service`,
`application`.`phone`AS `a_phone` FROM `application`
INNER JOIN `service` ON `application`.`id_service` = `service`.`id_service`
INNER JOIN `pay` ON `application`.`id_pay` = `pay`.`id_pay`
INNER JOIN `status` ON `application`.`id_status` = `status`.`id_status`
INNER JOIN `user` ON `application`.`id_user` = `user`.`id_user`
ORDER BY `id_application` DESC";
```

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

Листинг 6.4 Вывод карточки с отмененным заказом

```
$data .= sprintf('
    <div class="col">
        <div class="card mb-3 text-body-tertiary">
            <div class="card-body">
                <h5 class="card-title fs-3">Заявка №%s</h5>
                <p class="card-text mb-2">
                    <span class="fw-semibold">Фамилия</span>: %s
                </p>
                <p class="card-text mb-2">
```

```
                <span class="fw-semibold">Имя</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Отчество</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Адрес</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Телефон</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Статус заявки</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Причина отказа</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Категория</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Дата</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Время</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Способ оплаты</span>: %s
            </p>
            <p class="card-text mb-2">
                <span class="fw-semibold">Описание</span>: %s
            </p>
        </div>
    </div>
</div>',
$item['id_application'],
$item['surname'],
$item['u_name'],
$item['patronymic'],
$item['address'],
$item['a_phone'],
$item['status'],
$item['status_info'],
$item['sname'],
$item['date'],
$item['time'],
$item['pname'],
($item['id_service'] == "0") ? $item['description'] : $item['sname']);
```

6. Для карточек с новыми обращениями (**new**):

- добавляем в вывод кнопки: **Выполнено, В работе** и **Отменить,** с текстовым полем для указания причины. 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="buttons d-flex justify-content-between flex-wrap">
    <a href="../admin/controllers/update_applicate.php?id=%s&type=process"
       class="btn btn-outline-warning p-2 w-100 rounded-3 shadow-sm fw-bold fs-6 my-2"
       title="Изменение статуса заявки на \'В работе\'">В работе</a>
    <a href="../admin/controllers/update_applicate.php?id=%s&type=confirmed"
       class="btn btn-outline-success p-2 w-100 rounded-3 shadow-sm fw-bold fs-6 my-2"
       title="Изменение статуса заявки на \'Выполнено\'">Выполнено</a>

    <form action="../admin/controllers/update_applicate.php" method="GET">
        <input type="hidden" name="id" value="%s">
        <input type="hidden" name="type" value="canceled">
        <textarea class="form-control fs-3 rounded-5 shadow-sm" rows="6" name="info"
                  required placeholder="Причина отмены"></textarea>
        <button class="btn btn-outline-danger p-2 w-100 rounded-3 shadow-sm fw-bold fs-
6 my-2"  title="Изменение статуса заявки на \'Отменено\'">Отменить</button>
    </form>
</div>
```

- добавляем в список переменных **application_id**.

Листинг 6.6 Данные для подстановки

```
                ...
                   $item['id_application'],
                   $item['surname'],
                   $item['u_name'],
                   $item['patronymic'],
                   $item['address'],
                   $item['a_phone'],
                   $item['status'],
                   $item['status'],
                   $item['sname'],
                   $item['date'],
                   $item['time'],
                   $item['pname'],
                   ($item['id_service'] == "0") ? $item['description'] :
$item['sname'],
                   $item['id_application'],
                   $item['id_application'],
                   $item['id_application']);
            }
```

Так для улучшения пользовательского опыта рекомендуется добавление сортировки в админ панель . К этой части относятся пункты 7, 8 и 9. Вы можете их опустить и перейти сразу к этапу 7.

7. Добавим вкладки из *Bootstrap - Guides - Navs and Tabs - JavaScript behavior - vertical.*

Листинг 6.7 Код Админ панели с сортировкой обращений

```
<section>
    <div class="container py-3">
        <h3 class="display-3 text-center">Панель администратора</h3>

        <div class="d-flex align-items-start my-5">
            <div class="nav flex-column nav-pills me-3 w-25" id="v-pills-tab"
role="tablist" aria-orientation="vertical">
                <button class="nav-link active" id="v-pills-all-tab" data-bs-
toggle="pill" data-bs-target="#v-pills-all" type="button" role="tab" aria-controls="v-
pills-all" aria-selected="true">Все заявки</button>
                <button class="nav-link" id="v-pills-new-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-new" type="button" role="tab" aria-controls="v-pills-new"
aria-selected="false">Новые заявки</button>
                <button class="nav-link" id="v-pills-canceled-tab" data-bs-
toggle="pill" data-bs-target="#v-pills-canceled" type="button" role="tab" aria-
controls="v-pills-canceled" aria-selected="false">Отмененные заявки</button>
                <button class="nav-link" id="v-pills-confirmed-tab" data-bs-
toggle="pill" data-bs-target="#v-pills-confirmed" type="button" role="tab" aria-
controls="v-pills-confirmed" aria-selected="false">Подтвержденные заявки</button>
                <button class="nav-link" id="v-pills-process-tab" data-bs-toggle="pill"
data-bs-target="#v-pills-process" type="button" role="tab" aria-controls="v-pills-
process" aria-selected="false">Заявки в процессе</button>
            </div>

            <div class="tab-content w-75" id="v-pills-tabContent">
                <div class="tab-pane fade active show" id="v-pills-all" role="tabpanel"
aria-labelledby="v-pills-all-tab" tabindex="0"><?=fnOutCardAdmin()?></div>
                <div class="tab-pane fade" id="v-pills-new" role="tabpanel" aria-
labelledby="v-pills-new-tab" tabindex="0"><?=fnOutCardAdminStatus('new')?></div>
                <div class="tab-pane fade" id="v-pills-canceled" role="tabpanel" aria-
labelledby="v-pills-canceled-tab"
tabindex="0"><?=fnOutCardAdminStatus('canceled')?></div>
                <div class="tab-pane fade" id="v-pills-confirmed" role="tabpanel" aria-
labelledby="v-pills-confirmed-tab"
tabindex="0"><?=fnOutCardAdminStatus('confirmed')?></div>
                <div class="tab-pane fade" id="v-pills-process" role="tabpanel" aria-
labelledby="v-pills-process-tab"
tabindex="0"><?=fnOutCardAdminStatus('process')?></div>
            </div>
        </div>
    </div>
</section>
```

8. Также допишем дополнительные стили и функцию для корректной работы

Листинг 6.8 Дополнительные стили style.css

```
/* Изменение стилизации Bootstrap*/
.nav-pills .nav-link{
    color: #198754 !important;
}

.nav-pills .nav-link.active{
    background-color: #198754 !important;
    color: #FFFFFF !important;
}
```

9. Код функции **fnOutCardAdminStatus** частично совпадает с кодом **fnOutCardAdmin**.

Листинг 6.9 Код функции fnOutCardAdminStatus

```
function fnOutCardAdminStatus($code){
    global $connect;
    $sql = "SELECT `id_status` FROM `status` WHERE `code` = '{$code}'";
    $result = $connect->query($sql);
    $status = $result->fetch_assoc();
    $sql = "SELECT `id_application`, `user`.`name` AS `u_name`, `surname`,
`patronymic`, `status`,`status_info`, `date`, `time`,`address`, `pay`.`name` AS
`pname`, `service`.`name` AS `sname`, `code`, `description`,
`service`.`id_service`, `application`.`phone`AS `a_phone` FROM `application` INNER
JOIN `service` ON `application`.`id_service` = `service`.`id_service` INNER JOIN
`pay` ON `application`.`id_pay` = `pay`.`id_pay` INNER JOIN `status` ON
`application`.`id_status` = `status`.`id_status` INNER JOIN `user` ON
`application`.`id_user` = `user`.`id_user` WHERE `application`.`id_status` =
'{$status['id_status']}' ORDER BY `id_application` DESC";
    $result = $connect->query($sql);
    if(!$result->num_rows){
        if($code == "new"){
            $data = "<h4 class='display-6 text-center'>Новых заявок не
найдено<h4>";
        }elseif($code == "process"){
            $data = "<h4 class='display-6 text-center'>Заявок в процессе не
найдено<h4>";
        }elseif($code == "canceled"){
            $data = "<h4 class='display-6 text-center'>Отменных заявок не
найдено<h4>";
        }elseif($code == "confirmed"){
            $data = "<h4 class='display-6 text-center'>Выполненных заявок не
найдено<h4>";
        }
    }else{
        $data = "<div class='cards row row-cols-1 row-cols-md-3 g-4'>";
        foreach($result as $item){
            if($item['code'] == "new"){
                // код вывода из функции fnOutCardAdmin()
            }elseif($item['code'] == "canceled"){
                // код вывода из функции fnOutCardAdmin()
            }elseif($item['code'] == "confirmed"){
```

```
                // код вывода из функции fnOutCardAdmin()
            }elseif($item['code'] == "process"){
                // код вывода из функции fnOutCardAdmin()
            }
        }
        $data .= "</div>";
    }
    return $data;
}
```

Этап 7. Обновление статуса и подача заявки

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

В зависимости от переданного параметра **type** статус обращения будет сменяться **confirmed – Выполнено**, **canceled – Отменен, process – В работе**.

Листинг 7.1 Смена статуса обращения

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

    $sql = "SELECT `id_status` FROM `status` WHERE `code` = '{$_GET['type']}'";
    $result = $connect->query($sql);
    $status = $result->fetch_assoc();
    if($_GET['type'] == "canceled"){
        $sql = sprintf("UPDATE `application` SET `id_status`='%s', `status_info`='%s'
WHERE `id_application`='%s'", $status['id_status'], $_GET['info'], $_GET['id']);
    }else{
        $sql = sprintf("UPDATE `application` SET `id_status`='%s' WHERE
`id_application`='%s'", $status['id_status'], $_GET['id']);
    }

    if($connect->query($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");

    $sql = "SELECT * FROM `user` WHERE `login` = '{$_SESSION['login']}'";
    $result = $connect->query($sql);
    $user = $result->fetch_assoc();
```

```
    $sql = "SELECT `id_status` FROM `status` WHERE `code` = 'new'";
    $result = $connect->query($sql);
    $status = $result->fetch_assoc();

    if($_POST["description"] != ""){
        $sql = sprintf("INSERT INTO `application`(`id_application`, `id_user`,
`id_pay`, `id_service`, `id_status`, `description`, `date`, `time`, `phone`, `address`)
VALUES (NULL,'%s','%s','%s','%s','%s','%s','%s','%s','%s')",
$user['id_user'], $_POST['pay'], 1,  $status['id_status'], $_POST['description'],
$_POST['date'], $_POST['time'], $_POST['phone'], $_POST['address']);
    }else{
        $sql = sprintf("INSERT INTO `application`(`id_application`, `id_user`,
`id_pay`, `id_service`, `id_status`, `description`, `date`, `time`, `phone`, `address`)
VALUES (NULL,'%s','%s','%s','%s','%s','%s','%s','%s','%s')",
$user['id_user'], $_POST['pay'], $_POST['category'],$status['id_status'],
$_POST['description'], $_POST['date'], $_POST['time'], $_POST['phone'],
$_POST['address']);
    }
    if($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. Завершение работы

Завершающим этапом выполнения работы должны стать:

1. Тестирование работы проекта!
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.