# Аннотация

В данном документе пойдет речь о сайте для пейнтбольного клуба «DeltaBall», предназначенном для облегчения взаимодействия между клиентами и администраторами клуба.

Пояснительная записка включает в себя следующие разделы:

– Назначение и область применения;
– Постановка задачи;
– Описание программы;
– Программа и методика испытаний;
– Руководство оператора;
– Мероприятия по информационной безопасности.

# Содержание

# Введение

Сайт разрабатывается для ведения учета игр, которые проводятся в клубе, обеспечения работы специальной скидочной программы и своевременного обновления информации о клубе «DeltaBall».

Разработка ведется на основании технического и дипломного задания.

# 1. Назначение и область применения

Сайт предназначен для просмотра информации о клубе, обеспечения всего необходимого функционала для бронирования полигонов, а также для обновления базы данных клуба. Программным продуктом могут пользоваться сотрудники клуба и его клиенты.

Аналог программного продукта представлен на рисунке 1 – сайт крупнейшего клуба активного отдыха в Санкт-Петербурге «Pikabum» (https://pikabum.ru/).


Рисунок 1 – Главная страница сайта клуба Pikabum.

Его преимуществами являются:

– Простой интуитивно понятный интерфейс;

– Наличие большого объема структурированной информации на главной странице;

– Возможность забронировать площадку несколькими способами (на сайте, по телефону, с помощью электронной почты и в социальных сетях).

Его недостатками являются:

– Некоторые ссылки ведут на несуществующие страницы;

– Некорректная верстка на некоторых страницах (например, на странице оплаты и контактов);

– Наличие неработающих форм.

# 2. Постановка задачи

Необходимо разработать сайт для управления играми пользователей и для хранения основной информации о клубе, удовлетворяющий следующим требованиям:

- Администратор должен иметь доступ к панели администрирования сайта, которая позволяет просматривать и изменять информацию в базе данных, а также управлять запланированными играми пользователей;
- Клиент должен иметь доступ к личному кабинету, в котором он может просматривать и изменять контактную информацию о себе, узнать подробную информацию о своем ранге;
- В профиле клиента должна быть реализована возможность бронирования полигонов для игры. После заполнения всей информации об игре, пользователю становится доступна уникальная ссылка-приглашение;
- Клиенты должны иметь возможность присоединятся к существующим играм используя ссылки приглашения. После этого информация об этих играх должна отображаться в профиле пользователя;
- После завершенной игры пользователь должен получать опыт, который перерастает в новый ранг, при достижении определенного значения;
- Все пользователи имеют доступ к главной странице, которая должна содержать следующие блоки информации:
    1) Основная информация о деятельности клуба;
    2) Информация о полигонах клуба;
    3) Общая информация об услугах, которые предоставляет клуб.
- На всех формах должна быть предусмотрена валидация, для защиты от ввода неверных данных.

# 3. Описание программы

## 3.1 Общие сведения

Наименование программы: Сайт клуба «DeltaBall»

Для создания сайта клуба «DeltaBall» использовались следующие средства:

– Языки программирования JavaScript, HTML, CSS, C#;
– ASP Net Core Identity – фреймворк для написания структуры сайта;
– Entity Framework – фреймворк для связи сайта с базой данных;
– Bootstrap 5.2 – фреймворк для адаптивной верстки;
– JQuery – известная библиотека для JavaScript.

База данных реализована в Microsoft SQL Server Management Studio 2019.

Для работы с сайтом подойдет любое устройство с выходом в Интернет и установленным на нем браузером.

## 3.2 Функциональное назначение

В зависимости от роли авторизованного пользователя он должен иметь возможность:

– Добавлять и редактировать информацию в базе;
– Управлять запланированными играми;
– Просматривать и изменять личную информацию в профиле.

## 3.3 Описание логической структуры

Логическая структура проекта является древовидной и представлена на рисунке 2. ER диаграмма данных представлена на рисунке 3.

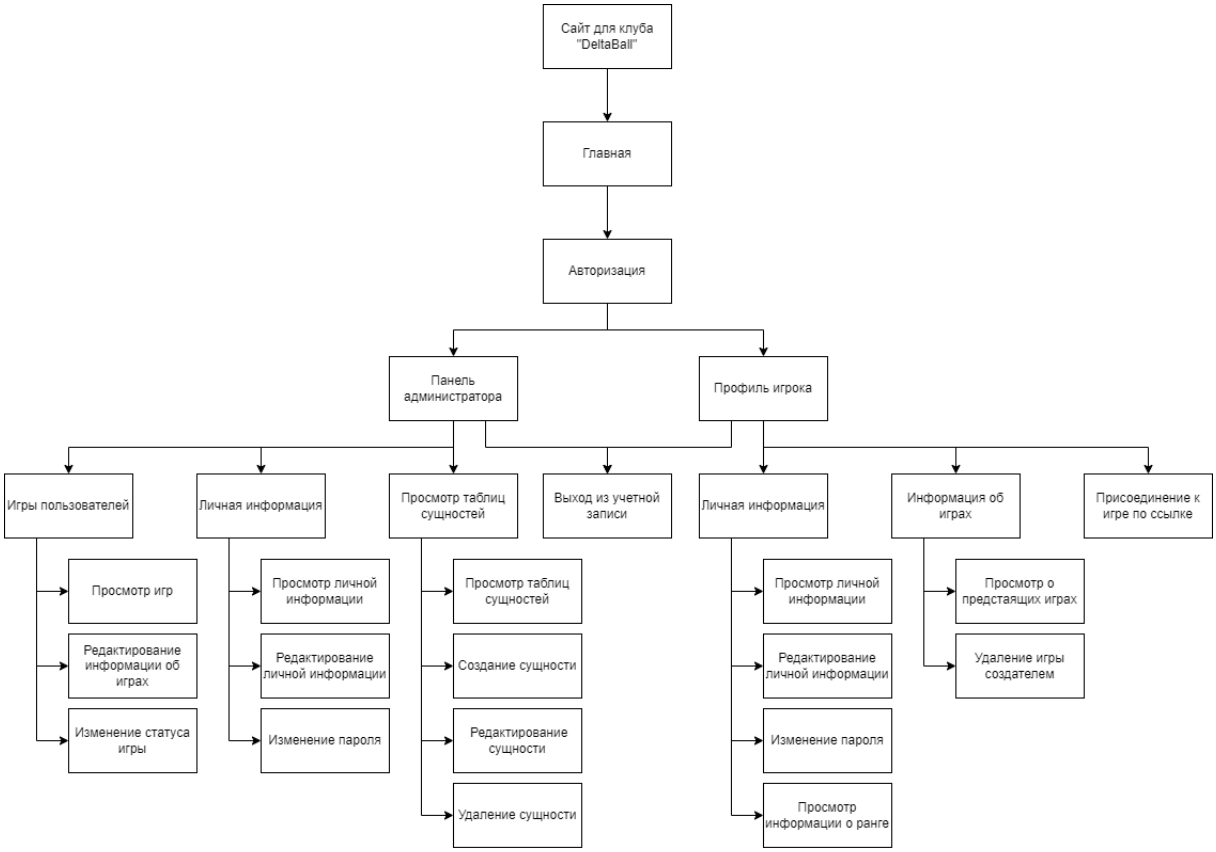


Рисунок 2 – Логическая структура проекта

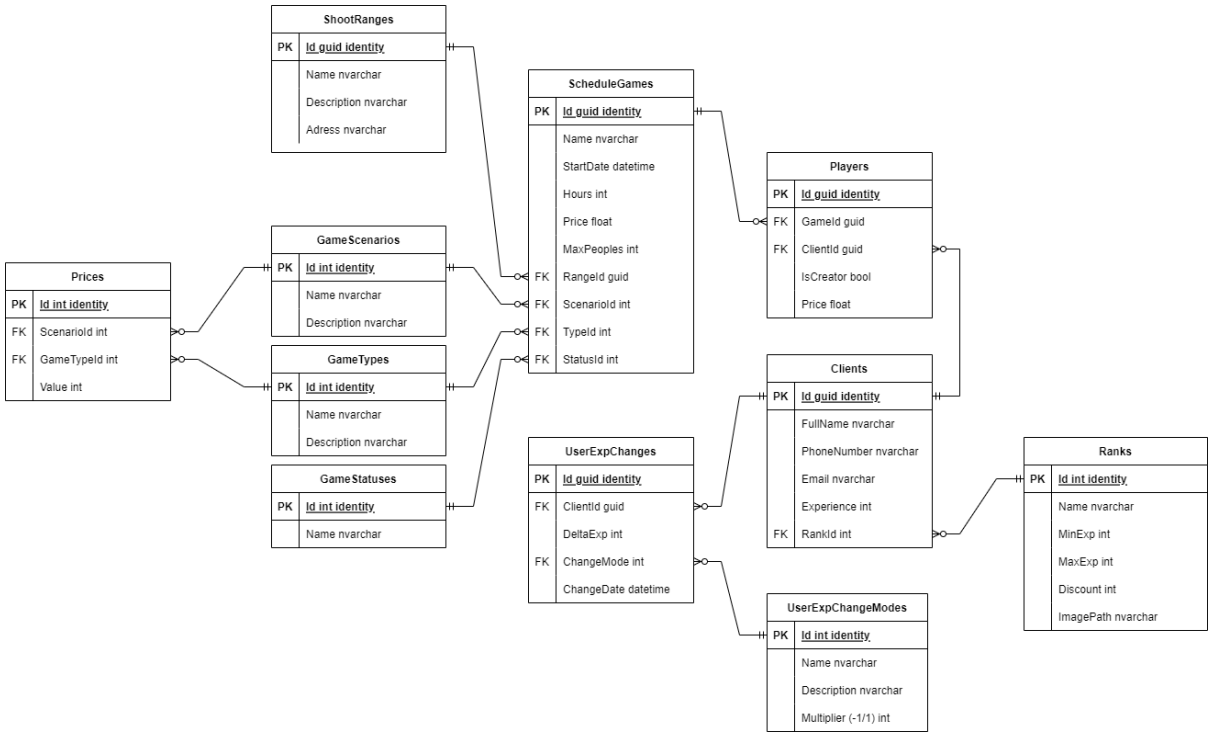


Рисунок 3 – ER диаграмма

### 3.4 Используемые технические средства

Для корректной работы сайта необходимо любое устройство, имеющее доступ в Интернет и с установленным браузером.

### 3.5 Вызов и загрузка

Для входа на сайт необходимо запустить проект DeltaBall.csproj в Visual Studio, затем запустить программу для выполнения. После этого откроется браузер с сайтом клуба на главной странице (см. рисунок 4).

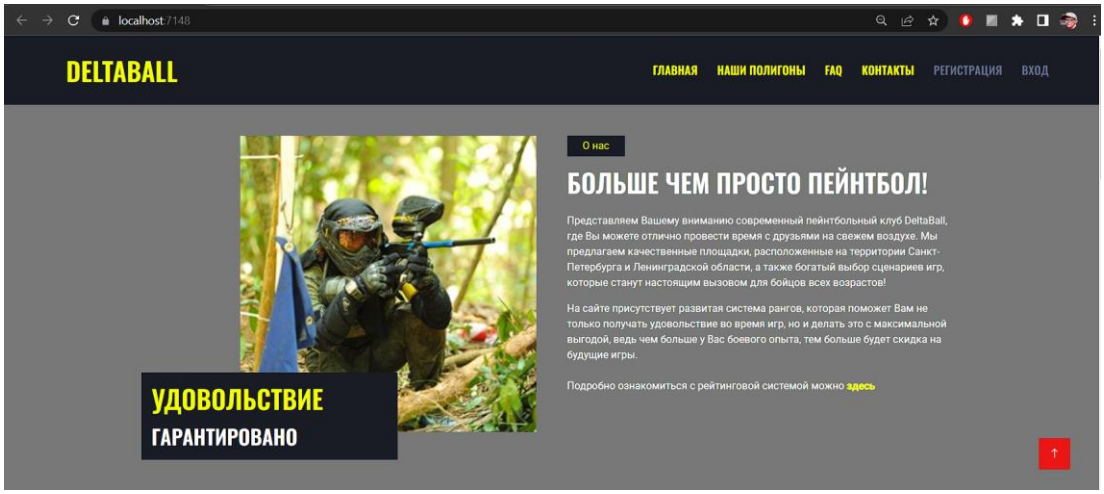


Рисунок 4 – Главная страница сайта.

### 3.6 Входные и выходные данные

Входные данные:

– На странице авторизации Логин и пароль пользователя
– На странице регистрации ФИО, номер телефона, адрес электронной почты и пароль
– На странице добавления новой игры Название игры, дата ее начала, продолжительность, максимальное количество участников, полигон, сценарий игры и тип игры
– На странице добавления новой сущности панели администратора Информация зависит от конкретной сущности

Выходные данные:

– Личная информация клиента
  ФИО, номер телефона, адрес электронной почты, ранг игрока и текущий опыт
– Список игр указанного игрока
  Подробная информация о всех играх, в которых клиент принимал участие
– История начисления опыта указанного игрока
  Подробная информация о начислении опыта клиента
– Список выбранного типа сущностей на панели администратора
  Информация о конкретной сущности зависит от ее типа

# 4. Программа и методика испытаний

## 4.1 Объекты испытаний

Объектом испытаний является сайт для пейнтбольного клуба «DeltaBall»

## 4.2 Цель испытаний

Испытания проводятся с целью проверки соответствия проекта требованиям, указанным в техническом задании.

## 4.3 Требования к программе

Соответствие программного продукта требованиям, указанным в техническом задании.

## 4.4 Методы испытаний

Испытания проводятся в следующем порядке:

1) Тестирование функционала главной страницы
   Все кнопки должны ссылаться на существующие страницы, «карусель» полигонов и «аккордеон» FAQ должны работать исправно.
2) Валидация форм авторизации и регистрации
   На формах должна присутствовать примитивная валидация, которая не дает пользователю отправить пустую форму.
3) CRUD операции для сущностей на панели администратора
   На панели администратора должны отображаться все таблицы сущностей или заглушки, если сущностей в таблице нет. На формах добавления и редактирования должна присутствовать простейшая валидация.
4) Просмотр профиля игрока
   На странице профиля игрока вся информация должна отображаться корректно. Все расчеты должны выполняться правильно.

5) Создание тестовой игры
   Клиенту необходимо создать тестовую игру, чтобы проверить корректность сохранения данных в базе.
6) Присоединение к игре по пригласительной ссылке
   После создания тестовой игры, клиент должен поделиться ссылкой с другим клиентом, чтобы тот смог присоединиться.
7) Изменение информации об игре на панели администратора
   Имитация начала игры: на панели администратора меняется статус игры на «Активна».
8) Тестирование начисления опыта игрока и изменения его ранга
   Имитация завершения игры: на панели администратора меняется статус игры на «Завершена». При этом всем игрокам должен начислиться опыт за завершенную игру.

### 4.5 Тестовый пример

В тестовом примере будет продемонстрировано выполнение пунктов 5-8 методики испытаний.

– Создание тестовой игры.
  Для этого необходимо зарегистрироваться или авторизоваться на сайте, перейти в профиль и нажать на «Создание игры». После заполнения предлагаемой формы, появится подробная информация об этой игре (см. рисунок 5).

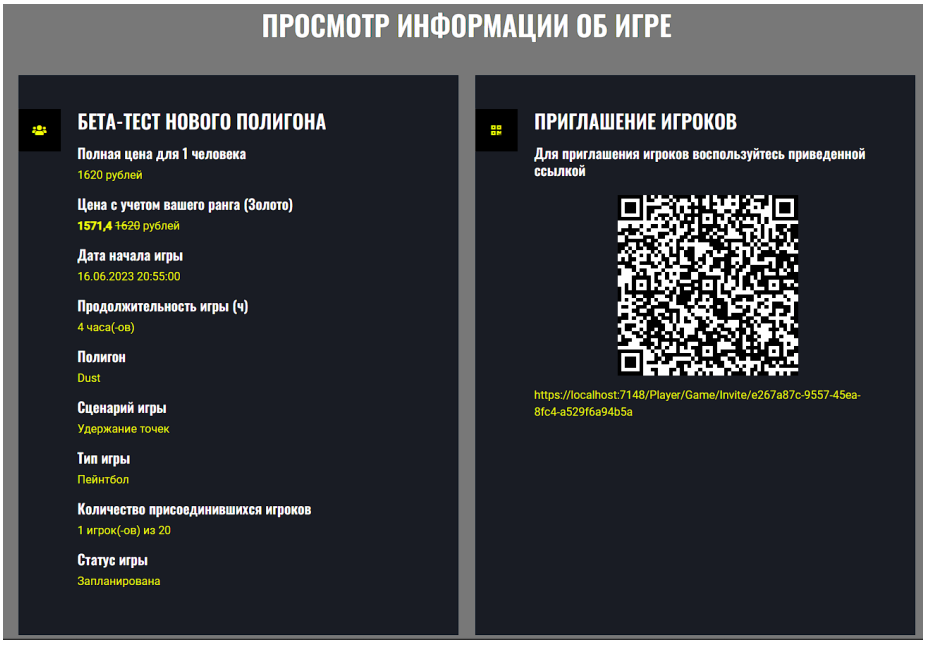


Рисунок 5 – Создание тестовой игры

– Присоединение к игре по пригласительной ссылке. После создания игры, необходимо передать пригласительную ссылку другому игроку. Когда второй игрок зайдет на сайт по ссылке и авторизуется, то перед ним предстанет подробная информация об игре (см. рисунок 6). После подтверждения записи, информация об этом сохранится в базе, а игрок может отслеживать игру в профиле (см. рисунок 7).

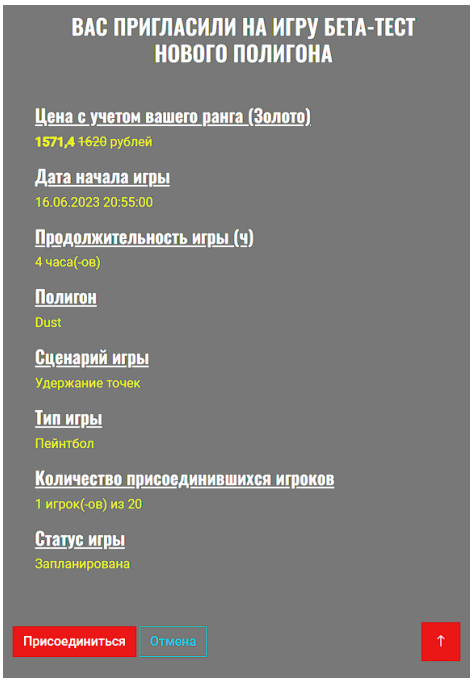


Рисунок 6 – Приглашение на игру

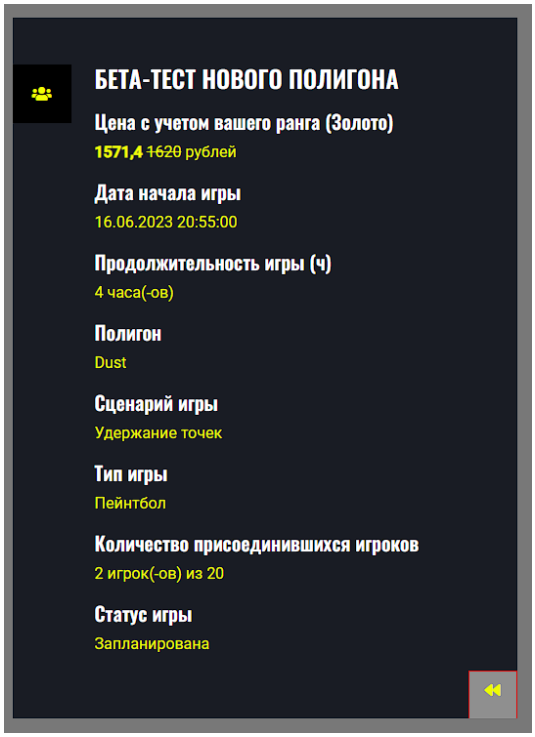


Рисунок 7 – Игра после ее принятия

– Изменение информации об игре на панели администратора. Имитация начала игры на полигоне. Администратору необходимо изменить статус игры на панели администрирования (см. рисунок 8).

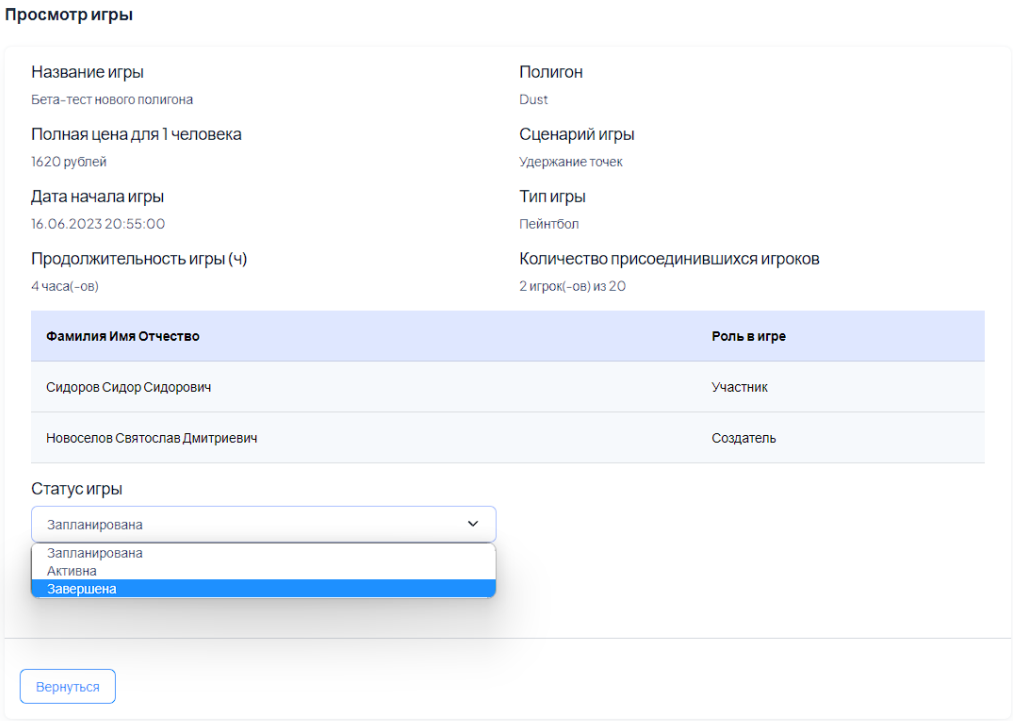


Рисунок 8 – Процесс изменения статуса игры администратором

– Тестирование начисления опыта игрока и изменения его ранга

Имитация завершения игры на полигоне. После завершения игры администратор меняет статус игры на «Завершена». После этого всем игрокам-участникам начисляется опыт и ранг, если потребуется. На рисунке 9 и 10 продемонстрирован результат этих действий.

**Таблица истории начисления опыта**

| Фамилия Имя Отчество | Количество присваиваемого опыта | Название причины | Дата изменения опытаы |
|---|---|---|---|
| Новоселов Святослав Дмитриевич | 50 | Игра завершена | 01.06.2023 17:03:56 |
| Сидоров Сидор Сидорович | 50 | Игра завершена | 01.06.2023 17:03:56 |

Рисунок 9 – Отображение начисления опыта после завершения игры на панели администратора

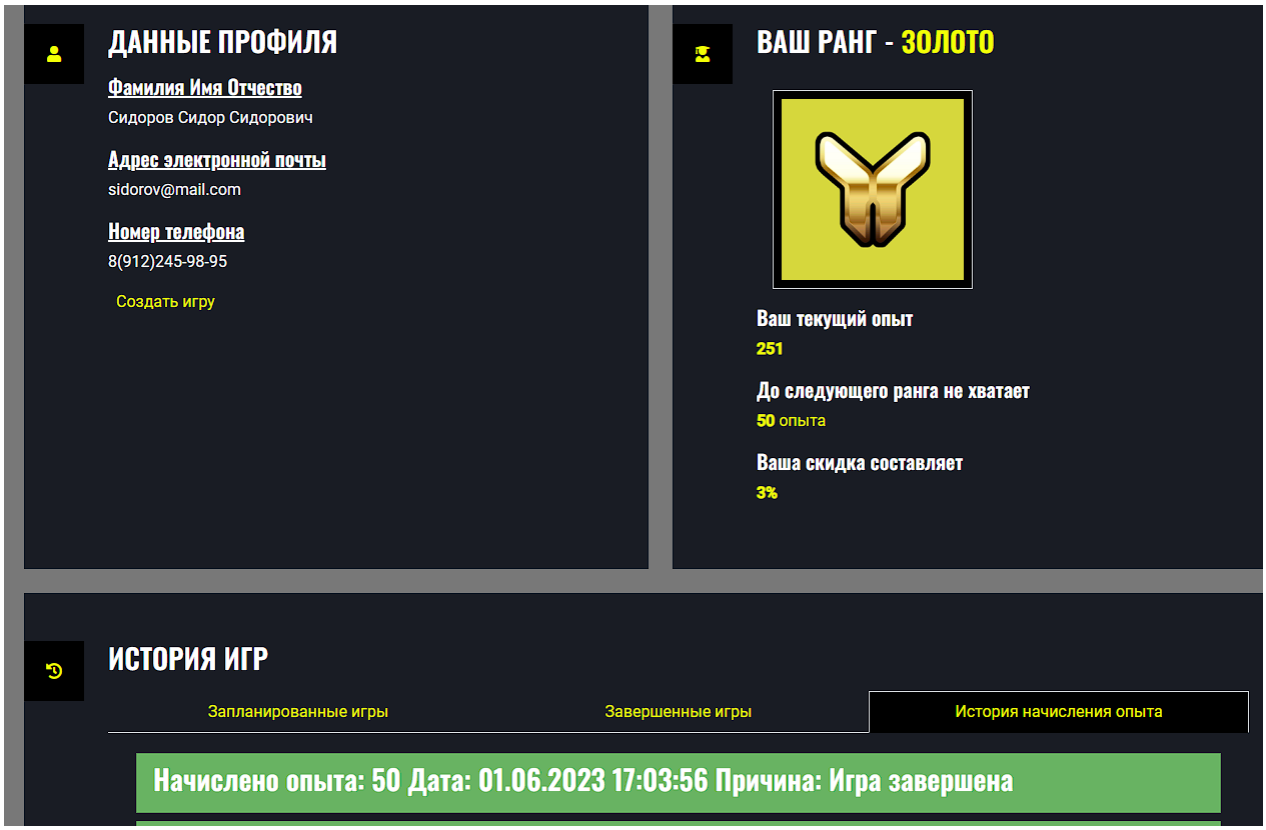


Рисунок 10 – Отображение начисленного опыта в окне ранга и истории

# 5. Руководство пользователя

## 5.1 Введение

Сайт для пейнтбольного клуба «DeltaBall» предназначен для ведения учета проводимых в клубе игр, а также своевременного обновление базы данных организации. Данное руководство предназначено для администраторов клуба.

## 5.2 Обзор интерфейса пользователя

При попадании на сайт, пользователь оказывается на главной информационной страницы, чтобы попасть в панель администратора, необходимо авторизоваться. Для этого в правом верхнем углу есть кнопка «Войти» (см. рисунок 4). После авторизации, администратор попадает в панель администрирования (см. рисунок 11).

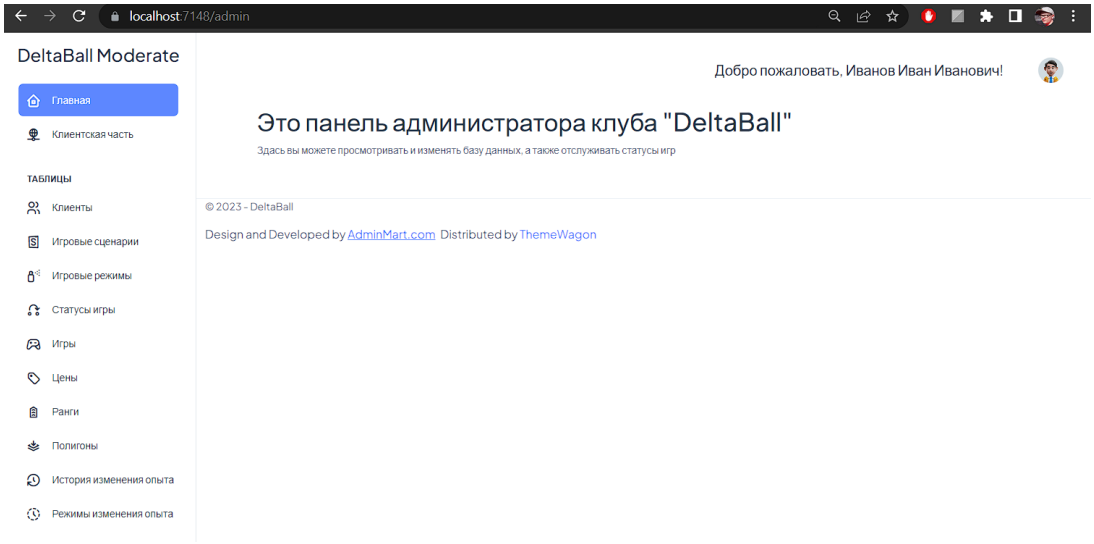


Рисунок 11 – Внешний вид панели администратора

В данном сегменте сайта, пользователь имеет возможность просмотреть сущности базы данных с помощью навигационной панели слева. При этом все таблицы будут появляться в основном поле в центре. В правом углу шапки есть возможность перейти к форме редактирования личных данных администратора, а также здесь расположена кнопка выхода из учетной записи. На некоторых таблицах в правой части могут

располагаться дополнительные кнопки (см. рисунок 12), которые предназначены для изменения данных.

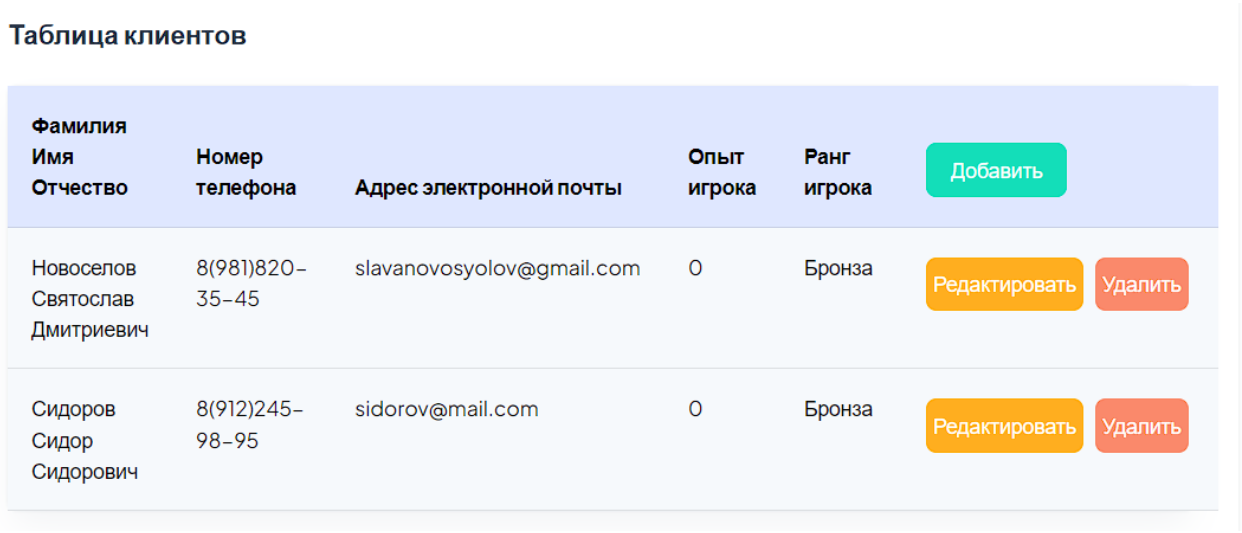


Рисунок 12 – Кнопки редактирования базы данных

### 5.3 Обзор функций пользователя

К основным функциям администратора относятся:

– Добавление новых сущностей

  Для добавления новой сущности необходимо нажать на кнопку «Создать» в правом верхнем углу таблицы. После этого откроется форма для создания выбранной сущности (см. рисунок 13). После ввода всей необходимой информации, нажмите на кнопку «Создать».

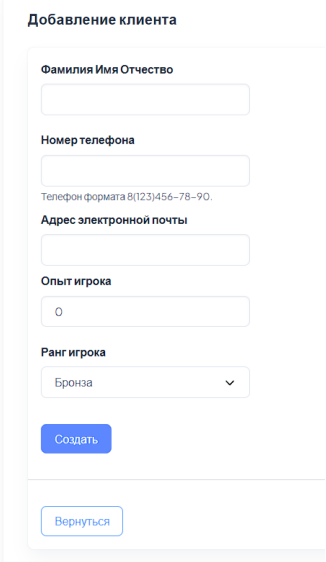


Рисунок 13 – Пример создания нового пользователя

– Редактирование существующих сущностей

Для редактирования выбранной сущности необходимо нажать на кнопку «Редактировать» напротив этой сущности. После этого откроется форма для редактирования выбранной сущности (см. рисунок 14). После изменения всей необходимой информации, нажмите на кнопку «Сохранить».

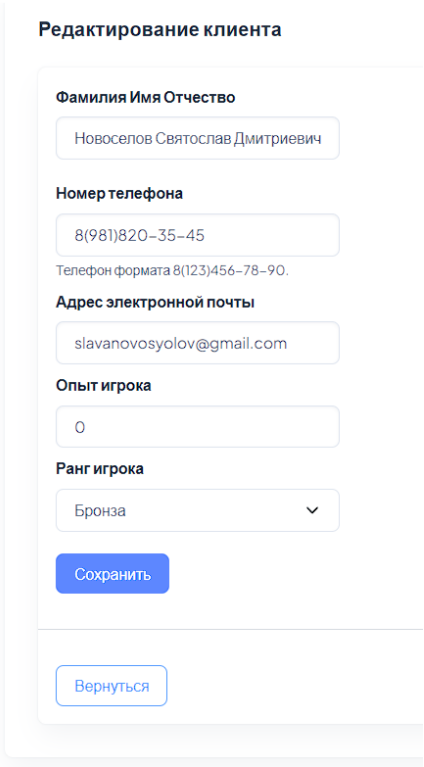


Рисунок 14 – Пример редактирования пользователя

– Удаление сущностей

Для удаления выбранной сущности необходимо нажать на кнопку «Удалить» напротив этой сущности. После этого откроется окно для подтверждения удаления выбранной сущности (см. рисунок 15). После подтверждения, сущность будет удалена, а пользователя вернет к просмотру таблицы.

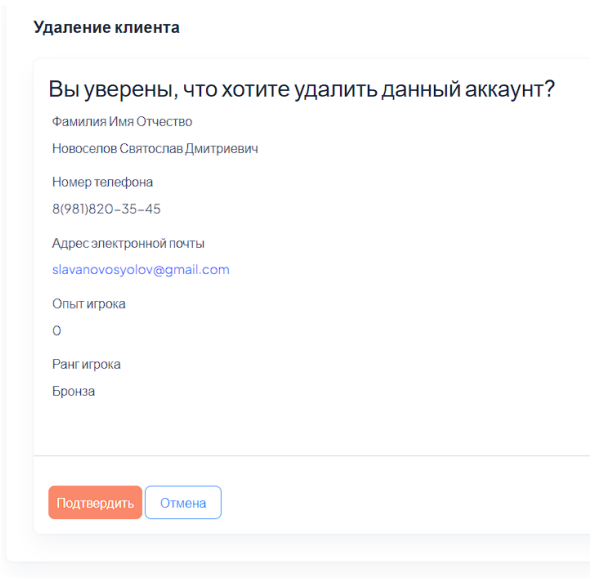


Рисунок 15 – Попытка удаления пользователя

– Управление запланированными играми

В разделе «Игры» панели администрирования пользователь может посмотреть список всех игр. При просмотре игры, администратор может менять ее статус (см. рисунок 16).

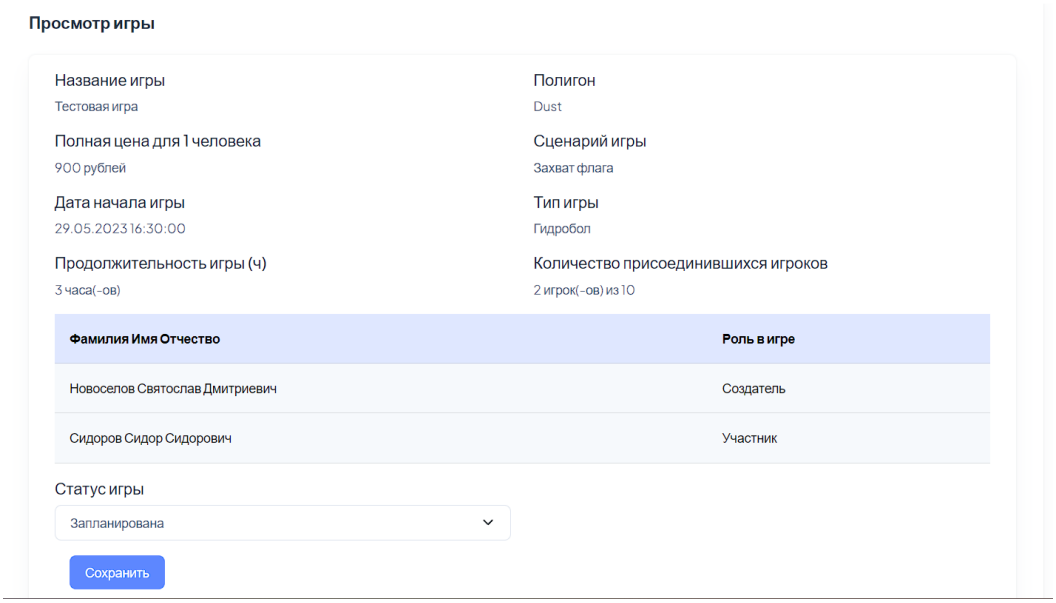


Рисунок 16 – Просмотр информации об игре

# 6. Мероприятия по информационной безопасности

Во время проектирования структуры базы данных и сайта были выявлены следующие возможные уязвимости:

– Отсутствие механизмов авторизации и аутентификации;

– Хранение незашифрованных паролей в базе данных;

– Отсутствие разграничения доступа.

В следствие этих уязвимостей могут возникнуть следующие угрозы:

– Несанкционированный доступ к персональным данным пользователей;

– Возможность проведения межсайтовой подделки запроса.

Для борьбы с уязвимостями и угрозами в области пользовательских данных было применено расширение для фреймворка ASP Net Core – Identity. Данное расширение позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями и выполнять другие действия с аккаунтом. Для этого на этапе создания базы данных происходит автоматическое добавление подсегмента ASP Net Core Identity, структура которого представлена на рисунке 17.

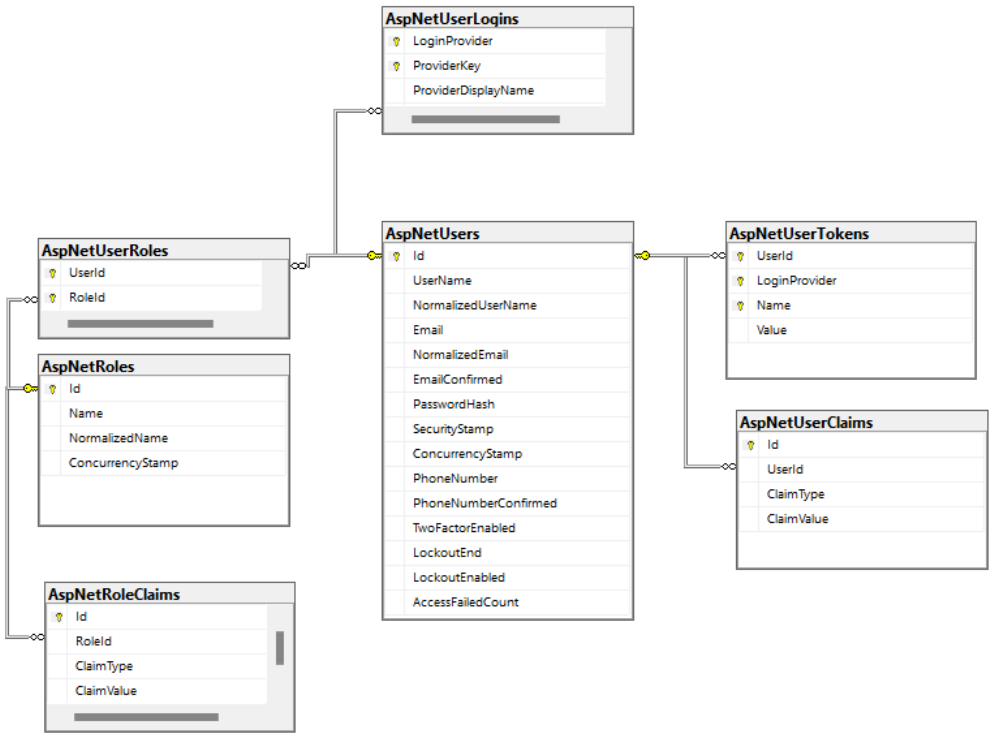

Рисунок 17 – База данных ASP Net Core Identity

В данном проекте активно использовались следующие таблицы:

– Таблица AspNetUsers содержит основную информацию об учетных записях пользователей (см. рисунок 18);
– Таблица AspNetRoles содержит роли пользователей (см. рисунок 19);
– Таблица AspNetUserRoles является связующей таблицей (см. рисунок 20).

SWYATOW2.DeltaBa...dbo.AspNetUsers | SWYATOW2.DeltaBa....AspNetUserRoles

| Id | UserName | Normalize... | Email | Normalize... | EmailConfi... | PasswordH... | SecuritySta... | Co |
|---|---|---|---|---|---|---|---|---|
| dead95a3ee7a | Иванов Ив... | ИВАНОВ И... | ivanov_delt... | IVANOV_DE... | True | AQAAAAIA... | 74BRVZQILY... | db |
| 6801ad76-e... | Сидоров С... | СИДОРОВ ... | sidorov@m... | SIDOROV@... | True | AQAAAAIA... | TVY3ASO6S... | 13 |
| e4d0f819-fe... | Новоселов ... | НОВОСЕЛО... | slavanovosy... | SLAVANOV... | True | AQAAAAIA... | 6ZDQDKUC... | 16 |

Рисунок 18 – Таблица AspNetUsers

SWYATOW2.DeltaBa...dbo.AspNetRoles

| Id | Name | NormalizedName | Concurrenc... |
|---|---|---|---|
| 0531adfe-8fad-4e95-8d59-0b2294b982fc | Admin | ADMIN | *NULL* |
| 514087fb-c72f-4712-b1e1-a3128e44e478 | Client | CLIENT | *NULL* |

Рисунок 19 – Таблица AspNetRoles

SWYATOW2.DeltaBa....AspNetUserRoles

| UserId | RoleId |
|---|---|
| 41dd2d65-6650-4cce-b3fd-dead95a3ee7a | 0531adfe-8fad-4e95-8d59-0b2294b982fc |
| 6801ad76-eb13-4e24-b133-47b8312c1d0c | 514087fb-c72f-4712-b1e1-a3128e44e478 |
| e4d0f819-fe46-4190-80c5-cd34e8e2daac | 514087fb-c72f-4712-b1e1-a3128e44e478 |

Рисунок 20 – Таблица AspNetUserRoles

Благодаря Identity-системе, обеспечивается автоматическое хеширование паролей в базе по алгоритму SHA-256 с 128-и битной солью и разграничение пользователей по ролям (см. рисунок 21).

| PasswordHash |
|---|
| AQAAAAIAAYagAAAAEAij0i4enGDgmK3fzeCziY/Vtnig/CY+68GOo6REE8rZ1uoRX6nFQ7zhb/R32tmMHw== |
| AQAAAAIAAYagAAAAEDNgc00pwqECz35N+tvxGLmo29kPhtV7Ceu/cfU2albxpkJw03DHM00nOknBcQ23FQ== |
| AQAAAAIAAYagAAAAEC134CQ2th45eIBIeb1A6kzTwEwPIGo3UcrTqAgyOJG2UQrHGqxGdE4MsZ2Q1asaZw== |
| *NULL* |

Рисунок 21 – Захешированные пароли, хранящиеся в базе данных

В данном проекте у авторизованных пользователей есть две роли – администратор и клиент, которые имеют доступ к двум областям сайта – панели администратора и профилю игрока соответственно. Если авторизованный пользователь попытается войти в защищенную область, к которой он не имеет доступа, то будет выводится сообщение, представленное на рисунке 22. Если неавторизованный пользователь попытается войти в защищенную область, ему предложат авторизоваться или зарегистрироваться.

Помимо разграничения доступа к областям сайта, применяется валидация форм авторизации, регистрации, создания игр и сущностей на панели администратора для защиты от неправильно введенных данных и защита от подделки межсайтовых запросов.

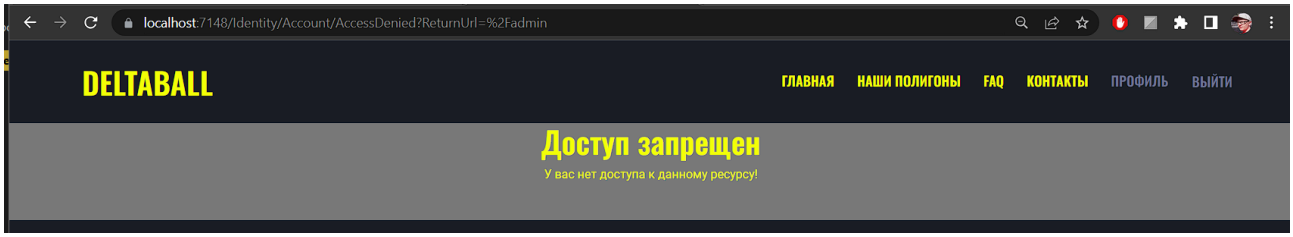


Рисунок 22 – Страница «Доступ запрещен»

# Заключение

В результате работы над дипломным проектом, был создан сайт для пейнтбольного клуба «DeltaBall», который полностью удовлетворяет техническому заданию. Сайт был создан на языке программирования C# с применением фреймворков ASP Net Core Identity и Bootstrap. Разработка проводилась в среде разработки Microsoft Visual Studio 2022 и СУБД Microsoft SQL Server Management Studio 2019.

Сайт имеет привлекательный дизайн, простую структуру и богатый функционал. Это поможет привлечь новых клиентов в клуб, а также поддерживать их «жизненный цикл», благодаря продвинутой системе скидок.

Для развития можно добавить новых привилегий для высоких рангов, добавить новые уникальные сценарии и реализовать новый функционал, например:

– Форум;

  На форуме новые игроки могут задавать вопросы более опытным о проведении игр. Форум будет также полезен для поиска новых игроков для игр, путем публикации пригласительной ссылки.

– Развитие системы штрафов и поощрений;

  Для этого можно выдавать дополнительный опыт за выслугу игр, во время дней рождений.

– Новости клуба.

  Ведение колонки новостей поможет в привлечении новых клиентов, а также уведомления клиентов об изменениях в правилах клуба.

# Приложение А

## Текст программы

Ниже приведены основные методы для работы с данными, полученные от пользователя:

```
        // Переадресация на форму для добавления нового клиента
    public IActionResult Create()
    {
        ViewData["Title"] = "Добавление клиента";
        ViewData["RankId"] = new SelectList(_dataManager.Ranks.GetRanks(), nameof(Client.Rank.Id),
nameof(Client.Rank.Name), _dataManager.Ranks.GetRankById(1));
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    // Обработка полученных данных и сохранение их в БД
    public async Task<IActionResult> Create([Bind("Id,FullName,PhoneNumber,Email,Experience,RankId")] Client client)
    {
        ModelState.ClearValidationState("Rank");
        ModelState.MarkFieldValid("Rank");
        if (ModelState.IsValid)
        {
            await _dataManager.Clients.SaveClientAsync(client, client.Email);
            return RedirectToAction("Info", "Table", new { name = "Clients" }, null);
        }
        ViewData["RankId"] = new SelectList(_dataManager.Ranks.GetRanks(), nameof(Client.Rank.Id),
nameof(Client.Rank.Name));
        ViewData["Title"] = "Добавление клиента";
        return View(client);
            }

        //Переадресация на форму редактирования клиента и заполнение ее полей данными
    public async Task<IActionResult> Edit(Guid id)
    {
        var client = _dataManager.Clients.GetClientById(id);
        if (client == default)
        {
            return NotFound();
        }
        ViewData["RankId"] = new SelectList(_dataManager.Ranks.GetRanks(), nameof(Client.Rank.Id),
nameof(Client.Rank.Name));
        ViewData["Title"] = "Редактирование клиента";
        return View(client);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
        // Обработка полученных данных и сохранение их в БД
        public async Task<IActionResult> Edit(Guid id, [Bind("Id,FullName,PhoneNumber,Email,Experience,RankId")]
Client client)
    {
        if (id != client.Id)
        {
            return NotFound();
        }
        ModelState.ClearValidationState("Rank");
        ModelState.MarkFieldValid("Rank");
        if (ModelState.IsValid)
        {
            await _dataManager.Clients.SaveClientAsync(client, null);
            return RedirectToAction("Info", "Table", new {name = "Clients"}, null);
        }
        ViewData["Title"] = "Редактирование клиента";
```

```
        ViewData["RankId"] = new SelectList(_dataManager.Ranks.GetRanks(), nameof(Client.Rank.Id),
nameof(Client.Rank.Name));
        return View(client);
            }

            // Переадресация на форму удаления
        public async Task<IActionResult> Delete(Guid id)
        {
            var client = _dataManager.Clients.GetClientById(id);
            if (client == null)
            {
                return NotFound();
            }
            ViewData["Title"] = "Удаление клиента";
            return View(client);
        }

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        // Подтверждение удаления пользователя
        public async Task<IActionResult> DeleteConfirmed(Guid id)
        {
            var client = _dataManager.Clients.GetClientById(id);
            if (client != default)
            {
                _dataManager.Clients.DeleteClient(client);
            }
            return RedirectToAction("Info", "Table", new { name = "Clients" }, null);
            }
```

Здесь приведены пример стандартного репозитория для связи сайта с базой данных с помощью EntityFramework:

```
public class ClientRepo : IClientRepo
    {
        private readonly AppDbContext _context;
        private readonly UserManager<IdentityUser> _userManager;
        public ClientRepo(AppDbContext context, UserManager<IdentityUser> manager)
        {
            _context = context;
            _userManager = manager;
        }
        /// <summary>
        /// Получить клиента по его ID
        /// </summary>
        /// <param name="id">Guid клиента</param>
        /// <returns></returns>
        public Client GetClientById(Guid id)
        {
            return GetClients().FirstOrDefault(x=>x.Id == id);
        }
        /// <summary>
        /// Получить список всех клиентов
        /// </summary>
        /// <returns></returns>
        public IEnumerable<Client> GetClients()
        {
            return _context.Clients.Include(x => x.Rank).OrderBy(x=>x.FullName).ToList();
        }
        /// <summary>
        /// Создать нового или изменить существующего клиента
        /// </summary>
        /// <param name="obj">Объект клиента</param>
        public async Task<bool> SaveClientAsync(Client obj, string password)
        {
```

```
        IdentityUser user;
                        try
        {
                                if (_context.Clients.Any(x => x.Id == obj.Id))
                                {
                                        user = _context.Users.First(x => x.Id == obj.Id.ToString());
                                        user.UserName = obj.FullName;
                                        user.NormalizedUserName = obj.FullName.ToUpper();
                                        user.Email = obj.Email;
                                        user.NormalizedEmail = obj.Email.ToUpper();
                                        user.PhoneNumber = obj.PhoneNumber;
            user.PhoneNumberConfirmed = true;
            user.EmailConfirmed = true;
                                        _context.Entry(user).State = EntityState.Modified;
                                        _context.Entry(obj).State = EntityState.Modified;
                                }
                                else
                                {
                                        user = new IdentityUser()
                                        {
                                                Id = obj.Id.ToString(),
                                                UserName = obj.FullName,
                                                NormalizedUserName = obj.FullName.ToUpper(),
                                                Email = obj.Email,
                                                NormalizedEmail = obj.Email.ToUpper(),
                                                PhoneNumber = obj.PhoneNumber,
                                                EmailConfirmed = true,
                                                PhoneNumberConfirmed = true,
                                                LockoutEnabled = false,
                                                TwoFactorEnabled = false,
                                           PasswordHash = new
PasswordHasher<IdentityUser>().HashPassword(null, password)
                                        };
                                        _context.Entry(obj).State = EntityState.Added;
                                }
                                _context.SaveChanges();
                                await _userManager.AddToRoleAsync(user, "Client");
                        }
                        catch (Exception e)
        {
           return false;
        }
                        return true;
                }
                /// <summary>
                /// Каскадное удаление клиента из базы
                /// </summary>
                /// <param name="obj">Объект клиента</param>
                /// <returns></returns>
                public bool DeleteClient(Client obj)
    {
        if(_context.Clients.Any(x => x.Id == obj.Id))
        {
           _context.Entry(_context.Users.FirstOrDefault(x => x.Id == obj.Id.ToString())).State = EntityState.Deleted;
           _context.Entry(obj).State = EntityState.Deleted;
           _context.SaveChanges();
           return true;
        }
        return false;
    }
}
```

## Источники, использованные при разработке

1. ГОСТ Р ИСО_МЭК 25051-2017 Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию.
2. ЕСПД Единая система программной документации.

## Internet – ресурсы

1. htmlbook.ru | Для тех кто делает сайты [Электронный ресурс] - http://htmlbook.ru/
2. MDN Web Docs | HTML [Электронный ресурс] – https://developer.mozilla.org/ru/docs/Web/HTML
3. Metanit.com | Руководство по ASP Net Core 7 [Электронный ресурс] - https://metanit.com/sharp/aspnet6/
4. Microsoft | Introduction to Identity on ASP Net Core [Электронный ресурс] – https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity
5. Bootstrap | Документация [Электронный ресурс] - https://bootstrap-4.ru/docs/5.3/getting-started/introduction/
6. Professor Web | Работа с Entity Framework [Электронный ресурс] - https://professorweb.ru/my/entity-framework/6/level1/