Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Югорский государственный университет

Институт цифровой экономики

КУРСОВАЯ РАБОТА

По дисциплине «Разработка веб-приложений с использованием баз данных»

«Сайт - спортивно-оздоровительный бассейн»

Студенты гр. 1191б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/А.Н.Свита /

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Е.И.Сафонов/

(подпись)

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Ханты-Мансийск

2021 г.

# Оглавление

[Введение 2](#_Toc75707935)

[1. Теоретическая часть 3](#_Toc75707936)

[2. Разработка веб-сайта 4](#_Toc75707937)

[2.1 Создание моделей 4](#_Toc75707940)

[2.2 Создание контроллеров 6](#_Toc75707941)

[2.3 Создание представлений 7](#_Toc75707942)

[2.4 Аутентификации и авторизации 9](#_Toc75707943)

[Заключение 11](#_Toc75707944)

[Список используемых источников 12](#_Toc75707945)

[Приложение А 13](#_Toc75707946)

[Приложение Б 14](#_Toc75707947)

# Введение

В данной работе было принято решение создать сайт про спортивно-оздоровительные бассейны.

Веб-сайт будет разработан на платформе ASP.NET Core, представляющую собой технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений.

Цель работы:

Задачи:

1. Проработать структуру базы данных
2. Создание моделей, описывающих структуру и логику используемых данных.
3. Создание контроллеров и представлений
4. Добавление аутентификации и авторизации.
5. Проработка ролевой политики

# Теоретическая часть

Веб-сайт разработан на платформе ASP.NET Core, что является продолжением развития платформы ASP.NET. Разработка была начата в 2014 году под условным названием ASP.NET vNext. В ноябре 2020 года вышла версия 5.0, которая и будет использована в данной работе.

ASP.NET Core может работать поверх кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC. [1]

Концепция паттерна MVC предполагает разделение приложения на три компонента:

1. Модель (model): описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными, например, логику валидации данных. Как правило, объекты моделей хранятся в базе данных.
2. Представление (view): отвечают за визуальную часть или пользовательский интерфейс, нередко html-страница, через который пользователь взаимодействует с приложением. Также представление может содержать логику, связанную с отображением данных. В то же время представление не должно содержать логику обработки запроса пользователя или управления данными.
3. Контроллер (controller): представляет центральный компонент MVC, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления, наполненного данными моделей. [2]

# Разработка веб-сайта



## Создание моделей

Для хранения базы данных используется Microsoft SQL Server Express, бесплатная система управления данными, обеспечивающая функциональное и надежное хранилище данных для веб-сайтов и настольных приложений.

Схема базы данных представлена в приложении А.1. Она состоит из 15 таблиц, связанных связью один ко многим.

Модели хранятся в папке Models, находящейся в корне проекта. Все модели оформляются как обычные классы на языке C#. В качестве примера будет рассмотрена модель KartaKlienta, описывающая данные об игре [Листинг кода Б. 1].

Листинг кода 1 – Модель KartaKlienta

public class KartaKlienta

{

public int Id {get; set;}

public virtual Karta Karta {get; set;}

public int KartaID {get; set;}

public virtual Klient Klient {get; set;}

public int KlientID {get; set;}

public DateTime DateNachala {get; set;}

public DateTime DateEnd {get; set;}

public int NomerKartKlienta {get; set;}

public virtual ICollection<Raspisanie> Raspisanies {get; set;}

public virtual ICollection<Ychet> Ychets {get; set;}

public virtual ICollection<ZanRti> ZanRtis {get; set;}

}

Модель KartaKlienta определяет ряд свойств: уникальный идентификатор Id, номер карты, дата начала и конца действия карты, и ссылку на связанные таблицы.

Для взаимодействия с базой создан контекст данных, в котором перечислены все модели [Листинг кода Б. 2].

Листинг кода 2 – Контекст данных

public class BasseinContext: DbContext

{

public BasseinContext(DbContextOptions<BasseinContext> options): base(options)

{ }

public DbSet<Group> Groups {get; set;}

public DbSet<Karta> Kartas {get; set;}

public DbSet<TypeGroup> TypeGroups {get; set;}

public DbSet<KartaKlienta> KartaKlientas {get; set;}

public DbSet<Klient> Klients {get; set;}

public DbSet<Kontakt> Kontakts {get; set;}

public DbSet<Raspisanie> Raspisanies {get; set;}

public DbSet<Spravka> Spravkas {get; set;}

public DbSet<Trener> Treners {get; set;}

public DbSet<TypeTrener> TypeTreners {get; set;}

public DbSet<TypeZanRti> TypeZanRtis {get; set;}

public DbSet<TypeVanna> TypeVannas {get; set;}

public DbSet<Vanna> Vannas {get; set;}

public DbSet<Ychet> Ychets {get; set;}

public DbSet<ZanRti> ZanRtis {get; set;}

}

Для подключения к базе данных задаются параметры подключения (способ подключения, в данном случае localhost, название базы данных и т.д.) в файле appsettings.json, для доступа к базе данных. [Листинг кода Б. 3].

Листинг кода 3 – Строка подключения

"ConnectionStrings": {

"DefaultConnection": "Server=localhost\\SQLEXPRESS;Database=DS3\_Data;Trusted\_Connection=True;MultipleActiveResultSets=true",

},

Для окончания настройки подключения файле Startup.cs нужно прописать созданное подключение и используемый контекст данных [Листинг кода Б. 4].

Листинг кода 4 – Редактирование Startup.cs

public void ConfigureServices(IServiceCollection services) {string connection = Configuration.GetConnectionString("DefaultConnection"); services.AddDbContext<BasseinContext>(options => options.UseSqlServer(connection));

## Создание контроллеров

При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки.

Для каждой модели создан свой контроллер, их функционал аналогичен друг другу. В начале работы получаем контекст данных [Листинг кода Б. 5].

Листинг кода 5 – Получение контекста данных

public KartaKlientasController(BasseinContext context)

{

\_context = context;

}

Каждый контроллер позволяет перейти на стартовую страницу, для этого вызывается функция Index, возвращающий представление с помощью метода View, который генерирует одноимённое представление. [Листинг кода Б. 5].

Листинг кода 5 – Функция Index

public async Task<IActionResult> Index()

{

var basseinContext = \_context.KartaKlientas.Include(k => k.Karta).Include(k => k.Klient);

return View(await basseinContext.ToListAsync());

}

Каждую запись в базе данных можно детально рассмотреть на отдельной странице, для этого в контроллерах определённа функция Details, принимающая id записи и генерирующее представление, в случае отсутствия записи будет возвращено NotFound [Листинг кода Б. 5].

Листинг кода 5 – Функция Details

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var kartaKlienta = await \_context.KartaKlientas

.Include(k => k.Karta)

.Include(k => k.Klient)

.FirstOrDefaultAsync(m => m.Id == id);

if (kartaKlienta == null)

{

return NotFound();

}

return View(kartaKlienta);

}

Помимо выше названных действий в контроллер добавлены функции Create, Edit, Delete, для создания, редактирования и удаления записей соответственно. Полное содержимое контроллера KartaKlientasController представлено в [Листинг кода Б. 5].

Доступ к контроллеру имеют только авторизованные пользователи, функции Create, Edit, доступны только пользователям, имеющим роль moderator или admin, также пользователь с ролью admin имеет доступ к функции Delete.

## Создание представлений

При попадании на ресурс пользователь получает веб-страницу, содержащую данные. В данном приложении для этого используются представления, формирующие внешний вид. В ASP.NET MVC Core представления - это файлы с расширением cshtml, содержащие код пользовательского интерфейса на языке html, а также конструкции Razor - специального движка представлений, который позволяет переходить от кода html к коду на языке C#.

После авторизации пользователь попадает на страницу Home/Index, содержащую ссылки на все остальные страницы (рисунок 1).

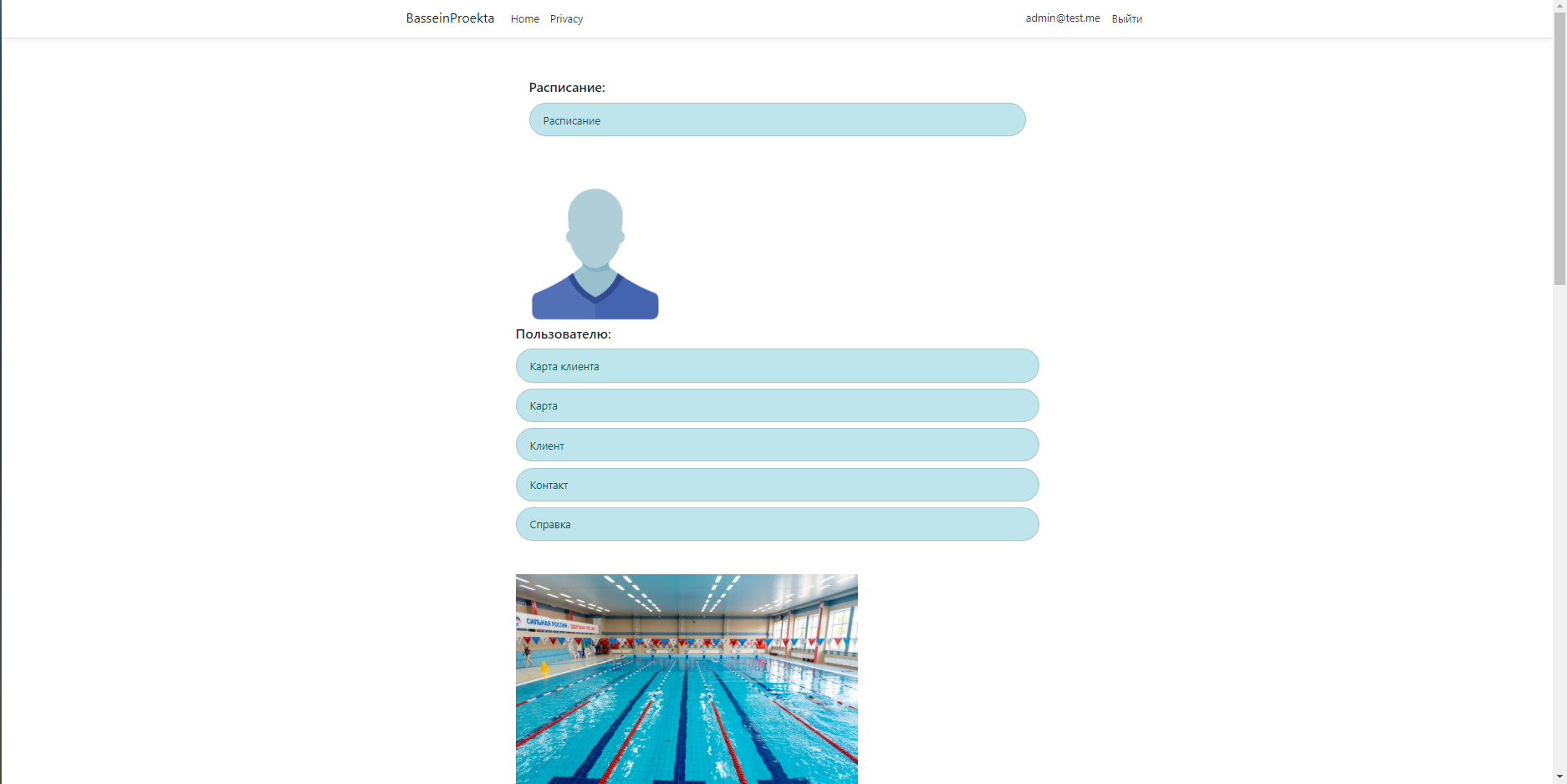


Рисунок 1 – Домашняя страница

Ссылки объединены по группам и вложены в отдельные блоки div. Полный код страницы представлен в приложении [Листинг кода Б. 6].

При нажатии на ссылку открывается страница, содержащая в себе список всех доступных записей, в случае если у пользователя роль moderator или admin, также отображается кнопка добавления новой записи (рисунок 2).

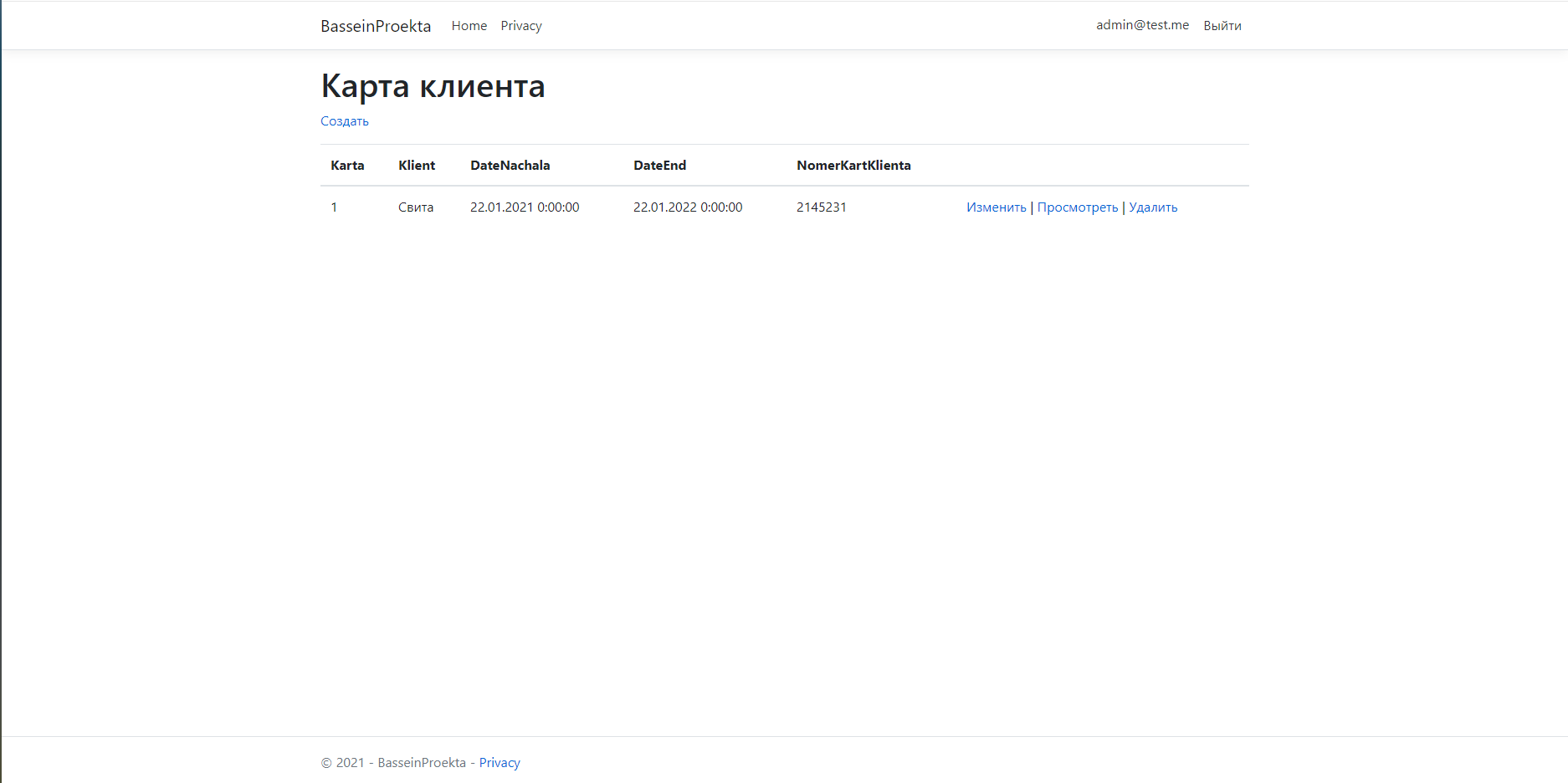


Рисунок 2 – Страница карта клиента

При на самой странице можно просматривать детально данные, а в случае ролей moderator и admin они могут их изменять, а роль admin может также удалять запись.

## Аутентификации и авторизации

Аутентификация — процедура проверки подлинности, в данном случае проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных. Авторизация — предоставление пользователю прав на выполнение определенных действий.

Для реализации данных функций будет использован ASP.NET Identity, представляющий собой встроенную в ASP.NET систему аутентификации и авторизации.

В начале создаём модель User и наследуем его от кдасса IdentityUser. Для работы с базой данных создаём контекст IdentityUser и наследуем его от IdentityDbContext<User> [Листинг кода Б. 7].

Листинг кода 7 – Контекст ApplicationContext

public class ApplicationContext : IdentityDbContext<User>

{Public ApplicationContext(DbContextOptions<ApplicationContext> options): base(options)

{

Database.EnsureCreated();

}

}

В appsettings.json добавляем новое соединение IdentityConnection, в класс Startup добавляем созданный контекст. Также добавляются сервисы для работы Identity, в котором также прописана отмена валидации пароля. Полный код класса Startup представлен в [Листинг кода Б.4].

Для работы с учётными данными пользователей создаём контроллер AccountController. Полный код контроллер представлен в [Листинг кода Б.8].

В нём определён метод регистрации. Для управления пользователями используется сервис Identity UserManager, для аутентификации пользователя используется сервис SignInManager. Метод CreateAsync добавляет данные пользователя в базу данных. Для отображения страницы регистрации (рисунок 4) в папке Views/ Account создаётся представление Register.

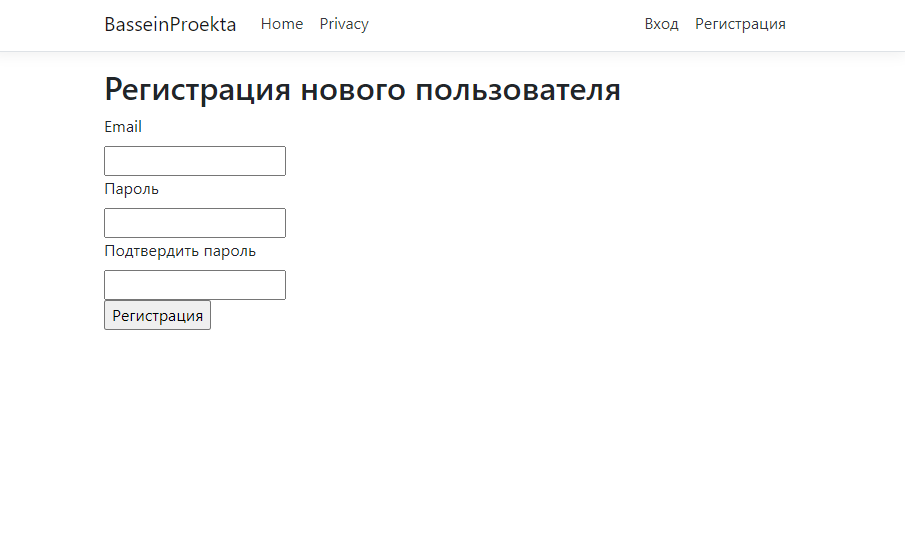


Рисунок 4 – Страница регистрации

Также к контроллер добавлены методы входа и выхода из аккаунта. Для входа создано представление, содержащее два поля ввода для логина и пароля (рисунок 5)

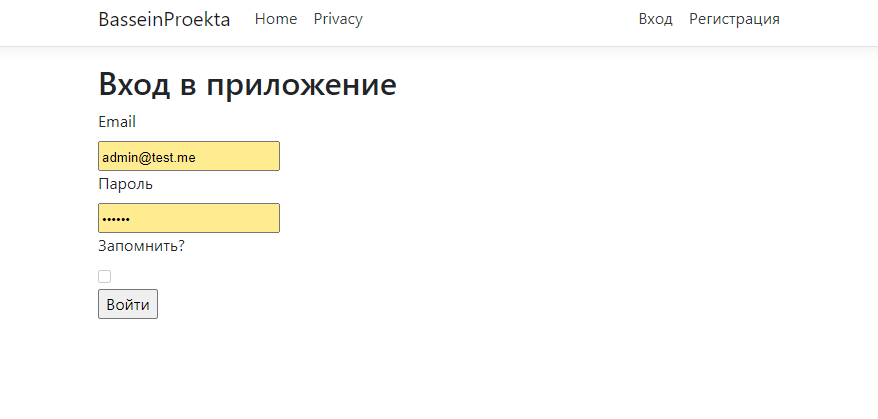


Рисунок 5 – Страница входа

Для работы с ролями добавляем контроллер RolesController, представленный в [Листинг кода Б.9]. Метод Index открывает представление Index.cshtml и выводит на экран список ролей, метод Create создаёт роль, Delete – удаляет. Метод Edit получает id пользователя и назначает тому роль.

# Заключение

Выполнены все поставленные задачи. В итоге получено работающее веб-приложение с использованием базы данных

Реализовано получение информации с базы данных, внесение новой информации, редактирование уже существующей и её удаление.

Реализована система аутентификации и авторизации.

# Список используемых источников

1. ASP.NET Core | Введение // METANIT.COM - Сайт о программировании [Электронный ресурс] Режим доступа: metanit.com/sharp/aspnet5/1.1.php, свободный (дата обращения: 10.06.21).
2. ASP.NET Core | Введение в MVC // METANIT.COM - Сайт о программировании [Электронный ресурс] Режим доступа: metanit.com/sharp/aspnet5/3.1.php, свободный (дата обращения: 10.06.21).
3. ASP.NET Core | Введение в ASP.NET Core Identity // METANIT.COM - Сайт о программировании [Электронный ресурс] Режим доступа: metanit.com/sharp/aspnet5/16.1.php, свободный (дата обращения: 10.06.21).

# Приложение А

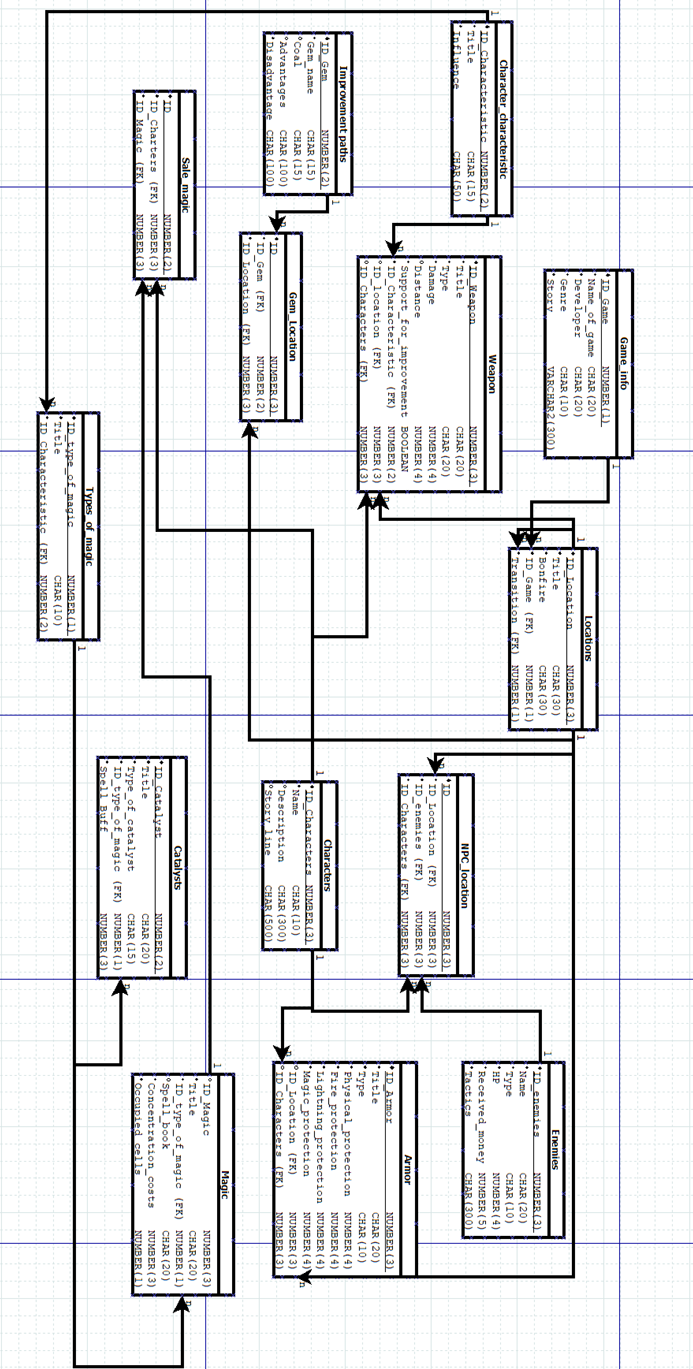


Рисунок А.1 – Схема базы данных

# Приложение Б

Листинг кода Б. 1 – Model KartaKlienta

public class KartaKlienta

{

public int Id {get; set;}

public virtual Karta Karta {get; set;}

public int KartaID {get; set;}

public virtual Klient Klient {get; set;}

public int KlientID {get; set;}

public DateTime DateNachala {get; set;}

public DateTime DateEnd {get; set;}

public int NomerKartKlienta {get; set;}

public virtual ICollection<Raspisanie> Raspisanies {get; set;}

public virtual ICollection<Ychet> Ychets {get; set;}

public virtual ICollection<ZanRti> ZanRtis {get; set;}

}

Листинг кода Б. 2 – Context BasseinContext

public class BasseinContext: DbContext

{

public BasseinContext(DbContextOptions<BasseinContext> options): base(options)

{ }

public DbSet<Group> Groups {get; set;}

public DbSet<Karta> Kartas {get; set;}

public DbSet<TypeGroup> TypeGroups {get; set;}

public DbSet<KartaKlienta> KartaKlientas {get; set;}

public DbSet<Klient> Klients {get; set;}

public DbSet<Kontakt> Kontakts {get; set;}

public DbSet<Raspisanie> Raspisanies {get; set;}

public DbSet<Spravka> Spravkas {get; set;}

public DbSet<Trener> Treners {get; set;}

public DbSet<TypeTrener> TypeTreners {get; set;}

public DbSet<TypeZanRti> TypeZanRtis {get; set;}

public DbSet<TypeVanna> TypeVannas {get; set;}

public DbSet<Vanna> Vannas {get; set;}

public DbSet<Ychet> Ychets {get; set;}

public DbSet<ZanRti> ZanRtis {get; set;}

}

Листинг кода Б. 3 – Строка подключения

"ConnectionStrings": {

"DefaultConnection": "Server=localhost\\SQLEXPRESS;Database=DS3\_Data;Trusted\_Connection=True;MultipleActiveResultSets=true",

"IdentityConnection": "Server=localhost\\SQLEXPRESS;Database=DS3\_Users;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

Листинг кода Б. 4 – Startup

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

string connection = Configuration.GetConnectionString("DefaultConnection");

services.AddDbContext<BasseinContext>(options => options.UseSqlServer(connection));

services.AddDbContext<ApplicationContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("IdentityConnection")));

services.AddIdentity<User, IdentityRole>(options => {

options.Password.RequiredLength = 3; // ìèíèìàëüíàÿ äëèíà

options.Password.RequireNonAlphanumeric = false; // òðåáóþòñÿ ëè íå àëôàâèòíî-öèôðîâûå ñèìâîëû

options.Password.RequireLowercase = false; // òðåáóþòñÿ ëè ñèìâîëû â íèæíåì ðåãèñòðå

options.Password.RequireUppercase = false; // òðåáóþòñÿ ëè ñèìâîëû â âåðõíåì ðåãèñòðå

options.Password.RequireDigit = false; // òðåáóþòñÿ ëè öèôðû

options.SignIn.RequireConfirmedAccount = false; //òðåáóþòñÿ ïîäòâåðæäåíèå email

}).AddEntityFrameworkStores<ApplicationContext>();

services.AddControllersWithViews();

services.AddRazorPages();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

env.EnvironmentName = "Production";

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseStatusCodePagesWithRedirects("/Home/Error");

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

Листинг кода Б. 5 – Controller KartaKlientasController

[Authorize]

public class KartaKlientasController : Controller

{

private readonly BasseinContext \_context;

public KartaKlientasController(BasseinContext context)

{

\_context = context;

}

// GET: KartaKlientas

public async Task<IActionResult> Index()

{

var basseinContext = \_context.KartaKlientas.Include(k => k.Karta).Include(k => k.Klient);

return View(await basseinContext.ToListAsync());

}

// GET: KartaKlientas/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var kartaKlienta = await \_context.KartaKlientas

.Include(k => k.Karta)

.Include(k => k.Klient)

.FirstOrDefaultAsync(m => m.Id == id);

if (kartaKlienta == null)

{

return NotFound();

}

return View(kartaKlienta);

}

[Authorize(Roles = "admin , moderator")]

// GET: KartaKlientas/Create

public IActionResult Create()

{

ViewData["KartaID"] = new SelectList(\_context.Kartas, "Id", "Id");

ViewData["KlientID"] = new SelectList(\_context.Klients, "Id", "FamiliR");

ViewData["SpravkaID"] = new SelectList(\_context.Spravkas, "Id", "Dopusk");

ViewData["GroupID"] = new SelectList(\_context.Groups, "Id", "Name");

return View();

}

// POST: KartaKlientas/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = "admin , moderator")]

public async Task<IActionResult> Create([Bind("Id,KartaID,KlientID,SpravkaID,GroupID,DateNachala,DateEnd,NomerKartKlienta")] KartaKlienta kartaKlienta)

{

if (ModelState.IsValid)

{

\_context.Add(kartaKlienta);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["KartaID"] = new SelectList(\_context.Kartas, "Id", "Id", kartaKlienta.KartaID);

ViewData["KlientID"] = new SelectList(\_context.Klients, "Id", "Id", kartaKlienta.KlientID);

return View(kartaKlienta);

}

[Authorize(Roles = "admin , moderator")]

/// GET: KartaKlientas/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var kartaKlienta = await \_context.KartaKlientas.FindAsync(id);

if (kartaKlienta == null)

{

return NotFound();

}

ViewData["KartaID"] = new SelectList(\_context.Kartas, "Id", "Id", kartaKlienta.KartaID);

ViewData["KlientID"] = new SelectList(\_context.Klients, "Id", "Id", kartaKlienta.KlientID);

}

// POST: KartaKlientas/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

[Authorize(Roles = "admin , moderator")]

public async Task<IActionResult> Edit(int id, [Bind("Id,KartaID,KlientID,SpravkaID,GroupID,DateNachala,DateEnd,NomerKartKlienta")] KartaKlienta kartaKlienta)

{

if (id != kartaKlienta.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(kartaKlienta);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!KartaKlientaExists(kartaKlienta.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["KartaID"] = new SelectList(\_context.Kartas, "Id", "Id", kartaKlienta.KartaID);

ViewData["KlientID"] = new SelectList(\_context.Klients, "Id", "Id", kartaKlienta.KlientID);

return View(kartaKlienta);

}

[Authorize(Roles = "admin , moderator")]

// GET: KartaKlientas/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var kartaKlienta = await \_context.KartaKlientas

.Include(k => k.Karta)

.Include(k => k.Klient)

.FirstOrDefaultAsync(m => m.Id == id);

if (kartaKlienta == null)

{

return NotFound();

}

return View(kartaKlienta);

}

// POST: KartaKlientas/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

[Authorize(Roles = "admin , moderator")]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var kartaKlienta = await \_context.KartaKlientas.FindAsync(id);

\_context.KartaKlientas.Remove(kartaKlienta);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool KartaKlientaExists(int id)

{

return \_context.KartaKlientas.Any(e => e.Id == id);

}

}

}

Листинг кода Б. 6– Index Home

@{

ViewData["Title"] = "Home Page";

}

@if (User.Identity.IsAuthenticated)

{

<div class="card-group" style="width: 75%; margin: 0 auto;">

<div class="card-body" style="margin: 25px;">

<h5 class="card-title">Расписание:</h5>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Raspisanies" asp-action="Index">

Расписание

</a>

</div>

</div>

<div class="card-group" style="width: 75%; margin: 0 auto;">

<div class="card-body" style="margin:5px;">

<img src="~/imag/user.png" />

<h5 class="card-title">Пользователю:</h5>

@if (User.IsInRole("admin") || User.IsInRole("Users"))

{

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="KartaKlientas" asp-action="Index">

Карта клиента

</a>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Kartas" asp-action="Index">

Карта

</a>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Klients" asp-action="Index">

Клиент

</a>

}

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Kontakts" asp-action="Index">

Контакт

</a>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Spravkas" asp-action="Index">

Справка

</a>

</div>

</div>

<div class="card-group" style="width: 75%; margin: 0 auto;">

<div class="card-body" style="margin: 5px;">

<img src="~/imag/bass.png" />

<h5 class="card-title">Бассейн:</h5>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Vannas" asp-action="Index">

Ванна

</a>

@if (User.IsInRole("admin") || User.IsInRole("moderator"))

{

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="TypeVannas" asp-action="Index">

Тип ванны

</a>

}

</div>

<div class="card-body" style="margin: 5px;">

<img src="~/imag/trener.png" />

<h5 class="card-title">Тренера:</h5>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Treners" asp-action="Index">

Тренер

</a>

@if (User.IsInRole("admin") || User.IsInRole("moderator")||User.IsInRole("trener"))

{

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="TypeTreners" asp-action="Index">

Тип тренера

</a>

}

</div>

<div class="card-body" style="margin: 5px;">

<img src="~/imag/group.png" />

<h5 class="card-title">Группа:</h5>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Groups" asp-action="Index">

Группа

</a>

@if (User.IsInRole("admin") || User.IsInRole("moderator") || User.IsInRole("trener"))

{

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="TypeGroups" asp-action="Index">

Тип группы

</a>

}

</div>

<div class="card-body" style="margin: 15px;">

<h5 class="card-title">Занятие:</h5>

<img src="~/imag/zan.png" />

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="ZanRtis" asp-action="Index">

Занятие

</a>

@if (User.IsInRole("admin") || User.IsInRole("moderator") || User.IsInRole("trener"))

{

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="TypeZanRtis" asp-action="Index">

Тип занятия

</a>

}

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Ychets" asp-action="Index">

Учет

</a>

</div>

</div>

@if (User.IsInRole("admin") || User.IsInRole("moderator"))

{

<div class="card-body" style="margin: 25px;">

<h5 class="card-title">Админу:</h5>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Roles" asp-action="Index">

Роли

</a>

<a class="list-group-item list-group-item-action list-group-item-info rounded-pill" style="margin-top: 10px;" asp-controller="Users" asp-action="Index">

Пользователи

</a>

</div>

}

}

else

{

<h4 class="text-danger">Зарегистрируйтесь для просмотра</h4>

}

Листинг кода Б. 7 – Context ApplicationContext

public class ApplicationContext : IdentityDbContext<User>

{Public ApplicationContext(DbContextOptions<ApplicationContext> options): base(options)

{

Database.EnsureCreated();

}

}

Листинг кода Б. 8 – Controller AccountController

public class AccountController : Controller

{

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

public AccountController(UserManager<User> userManager, SignInManager<User> signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

[HttpGet]

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Email };

// добавляем пользователя

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

// установка куки

await \_signInManager.SignInAsync(user, false);

return RedirectToAction("Index", "Home");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpGet]

public IActionResult Login(string returnUrl = null)

{

return View(new LoginViewModel { ReturnUrl = returnUrl });

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

// проверяем, принадлежит ли URL приложению

if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))

{

return Redirect(model.ReturnUrl);

}

else

{

return RedirectToAction("Index", "Home");

}

}

else

{

ModelState.AddModelError("", "Неправильный логин и (или) пароль");

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

// удаляем аутентификационные куки

await \_signInManager.SignOutAsync();

return RedirectToAction("Index", "Home");

}

}

Листинг кода Б. 9 – Controller RolesController

[Authorize(Roles = "admin , moderator")]

public class RolesController : Controller

{

RoleManager<IdentityRole> \_roleManager;

UserManager<User> \_userManager;

public RolesController(RoleManager<IdentityRole> roleManager, UserManager<User> userManager)

{

\_roleManager = roleManager;

\_userManager = userManager;

}

public IActionResult Index() => View(\_roleManager.Roles.ToList());

public IActionResult Create() => View();

[HttpPost]

public async Task<IActionResult> Create(string name)

{

if (!string.IsNullOrEmpty(name))

{

IdentityResult result = await \_roleManager.CreateAsync(new IdentityRole(name));

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(name);

}

[HttpPost]

public async Task<IActionResult> Delete(string id)

{

IdentityRole role = await \_roleManager.FindByIdAsync(id);

if (role != null)

{

IdentityResult result = await \_roleManager.DeleteAsync(role);

}

return RedirectToAction("Index");

}

public IActionResult UserList() => View(\_userManager.Users.ToList());

public async Task<IActionResult> Edit(string userId)

{

// получаем пользователя

User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

// получем список ролей пользователя

var userRoles = await \_userManager.GetRolesAsync(user);

var allRoles = \_roleManager.Roles.ToList();

ChangeRoleViewModel model = new ChangeRoleViewModel

{

UserId = user.Id,

UserEmail = user.Email,

UserRoles = userRoles,

AllRoles = allRoles

};

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(string userId, List<string> roles)

{

// получаем пользователя

User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

// получем список ролей пользователя

var userRoles = await \_userManager.GetRolesAsync(user);

// получаем все роли

var allRoles = \_roleManager.Roles.ToList();

// получаем список ролей, которые были добавлены

var addedRoles = roles.Except(userRoles);

// получаем роли, которые были удалены

var removedRoles = userRoles.Except(roles);

await \_userManager.AddToRolesAsync(user, addedRoles);

await \_userManager.RemoveFromRolesAsync(user, removedRoles);

return RedirectToAction("UserList");

}

return NotFound();

}

}