областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

Специальность:09.02.07«Информационные системы и программирование»

**ДИПЛОМНЫЙ ПРОЕКТ**

на тему:

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

ДП-02069846-09.02.07-21.04.81

|  |  |
| --- | --- |
| Руководитель: | \_\_\_\_\_\_\_ И.В. Лапшина |
| Выполнил: | \_\_\_\_\_\_\_ Д.В. Волнухин |
|  |  |
| «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 | |

2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc71031854)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc71031855)

[1.1. **Техническое задание на разработку программного продукта** 6](#_Toc71031856)

[1.2. **Организация ввода-вывода пользовательского интерфейса** 6](#_Toc71031857)

[**1.2.1** **Входные данные** 6](#_Toc71031858)

[**1.2.2** **Выходные данные** 6](#_Toc71031859)

[**1.2.3** **Интерфейс пользователя** 6](#_Toc71031860)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 7](#_Toc71031861)

[2.1. **Выбор и обоснование языка программирования** 8](#_Toc71031862)

[2.2. **Описание программы** 8](#_Toc71031863)

[**2.2.1.** **Общие сведения** 8](#_Toc71031864)

[**2.2.3.** **Используемые технические средства** 8](#_Toc71031865)

[**2.2.4.** **Вызов и загрузка** 8](#_Toc71031866)

[3. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ 9](#_Toc71031867)

[**3.1.** **Введение** 10](#_Toc71031868)

[**3.2.** **Информационная безопасность проекта** 11](#_Toc71031869)

[ПРИЛОЖЕНИЯ 12](#_Toc71031870)

[**ПРИЛОЖЕНИЕ 1** 14](#_Toc71031871)

[**ПРИЛОЖЕНИЕ1 (*Продолжение)*** 15](#_Toc71031872)

[**ПРИЛОЖЕНИЕ1 (*Продолжение)*** 16](#_Toc71031873)

[**ПРИЛОЖЕНИЕ1** 18](#_Toc71031874)

[**ПРИЛОЖЕНИЕ1(*Продолжение)*** 19](#_Toc71031875)

[**ПРИЛОЖЕНИЕ2** 21](#_Toc71031876)

[**ПРИЛОЖЕНИЕ3** 23](#_Toc71031877)

[**ПРИЛОЖЕНИЕ4** 25](#_Toc71031878)

[**ПРИЛОЖЕНИЕ4(*Продолжение)*** 26](#_Toc71031879)

[**ПРИЛОЖЕНИЕ4(*Продолжение)*** 27](#_Toc71031880)

[**ПРИЛОЖЕНИЕ5** 29](#_Toc71031881)

# ВВЕДЕНИЕ

Для выполнения дипломного проекта потребовалось знание HTML, CSS, C#. Основная цель работы заключается в создании Web-сайта. В результате реализации данного сайта я закрепил знания в области web-разработки.

HTML — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML).

**С#** — это современный язык программирования, созданный компанией Microsoft. На нём можно писать программы любой сложности для любых платформ и операционных систем. C# устроен так, чтобы программист мог писать **меньше кода.** Для этого в языке много «магических конструкций» и «синтаксического сахара» — способов выразить идею лаконично. Это**кроссплатформенный язык,** то есть ваша программа будет работать не только в той операционной системе, в которой она написана, но и в других.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

|  |  |
| --- | --- |
| Руководитель: | \_\_\_\_\_\_\_ И.В. Лапшина |
| Выполнил: | \_\_\_\_\_\_\_ Д.В. Волнухин |
|  |  |
| «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 | |

* 1. **Техническое задание на разработку программного продукта**

Основанием для разработки послужило задание на дипломный проект от ОГПОУ «Костромской политехнический колледж».

* 1. **Организация ввода-вывода пользовательского интерфейса**
     1. **Входные данные**

Входными данными являются загрузка и редактирование учебного материала.

* + 1. **Выходные данные**

Выходными данными является беспрепятственное отображение учебного материала.

* + 1. **Интерфейс пользователя**

Интерфейс построен таким образом, что любой пользователь, не имеющий необходимых знаний в программировании может отыскать нужный материал и изучить его самостоятельно.

На странице сайта находится вкладка для администратора в которой он редактирует и добавляет учебный материал для пользователей. На общей странице есть вкладки (Главная, Темы, Обратная связь).

# СПЕЦИАЛЬНАЯ ЧАСТЬ

|  |  |
| --- | --- |
| Руководитель: | \_\_\_\_\_\_\_ И.В. Лапшина |
| Выполнил: | \_\_\_\_\_\_\_ Д.В. Волнухин |
|  |  |
| «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 | |

* 1. **Выбор и обоснование языка программирования**

Для решения поставленной задачи был выбран язык C# так как данный язык не слишком сложен изучении и очень гибок для решения задач по написанию веб-сайта. Один из плюсов **ASP.NET** помогает просто и быстро связать серверный код с клиентским в серверных приложениях и веб-разработке. Благодаря использованию любых шаблонов и доступности документации, большинство возникающих проблем решаются довольно быстро.

* 1. **Описание программы**
     1. **Общие сведения**

Web-сайт “Электронное пособие” нужен для изучения учебного материала самостоятельно или дистанционно.

* + 1. **Функциональное назначение**

Web-сайт создаётся для получения знаний самостоятельно или дистанционно, а также для упрощения работы преподавателю.

* + 1. **Используемые технические средства**

Приложение было разработано на A**SP.NET**  Core.

* + 1. **Вызов и загрузка**

Приложение работает на любой ОС где есть браузер.

# ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

|  |  |
| --- | --- |
| Руководитель: | \_\_\_\_\_\_\_ И.В. Лапшина |
| Выполнил: | \_\_\_\_\_\_\_ Д.В. Волнухин |
|  |  |
| «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 | |

## **Введение**

Информационная безопасность (англ. Information Security, а также — англ. InfoSec) — практика предотвращения несанкционированного доступа, использования, раскрытия, искажения, изменения, исследования, записи или уничтожения информации. Это универсальное понятие применяется вне зависимости от формы, которую могут принимать данные (электронная или, например, физическая). Основная задача информационной безопасности — сбалансированная защита конфиденциальности, целостности и доступности данных, с учётом целесообразности применения и без какого-либо ущерба производительности организации. Это достигается, в основном, посредством многоэтапного процесса управления рисками, который позволяет идентифицировать основные средства и нематериальные активы, источники угроз, уязвимости, потенциальную степень воздействия и возможности управления рисками. Этот процесс сопровождается оценкой эффективности плана по управлению рисками.

Для защиты интернет-ресурсов от хакеров используются различные варианты Web Application Firewall – многосетевых экранов, позволяющих на прикладном уровне эффективно отражать атаки злоумышленников.

* Надежные пароли. Для генерации и хранения паролей можно применять специальные менеджеры.
* Ограничение доступа. Круг лиц, которые имеют доступ к учетной записи администратора сайта, должен быть строго ограничен.
* Безопасные протоколы. При подключении к сайту через файлообменные и прочие подобные программы необходимо использовать только безопасные протоколы с шифрованием SFTP или SCP.
* Программное обеспечение сервера. Необходимо регулярно обновлять программное обеспечение, поскольку большая часть атак производится через уязвимости устаревшего ПО.
* Антивирусы. Компьютер администратора сайта должен быть максимально защищен. Периодически проверяйте сайт на вирусы.
* Бэкапы. Регулярно создавать резервные копии сайта.

## **Информационная безопасность проекта**

* Так же установлено новейшее программное обеспечение сервера, которое предотвращает DOS и другие виды атак на сайт
* Созданы резервные копии сайта, на случай не предвиденных проблем.

# ПРИЛОЖЕНИЯ

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

**Техническое задание**

ДП-02069846-09.02.07-21.05.37

ПРИЛОЖЕНИЕ 1

Web-сайт

3 листа

|  |  |
| --- | --- |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин  «\_\_»\_\_\_\_\_\_2021 |

## **ПРИЛОЖЕНИЕ 1**

1. **ВВЕДЕНИЕ**

Приложение разработано с целью изучения учебного материала дистанционно илил самостоятельно.

1. **ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

Основанием для разработки послужило задание на дипломный проект от ОГПОУ «Костромской политехнический колледж».

1. **НАЗНАЧЕНИЕ РАЗРАБОТКИ**

Назначением разработки данного приложения направленно на изучение учебного материала самостоятельно или дистанционно, а также для упрощения работы преподавателю.

Впроцессе работы с приложением получить:

* полностью функционирующии сайт;
* интуитивно понятный интерфейс

1. **ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

* Программа должна полноценно функционировать
* В процессе разработки не должно быть скрытых ошибок
* Все возникшие неполадки в программе должны быть устранены как можно быстрее

## **ПРИЛОЖЕНИЕ1 (*Продолжение)***

1. **ТРЕБОВАНИЯ К ПРОГРАММЕ**
   1. **Требования к функциональным характиристикам**

На сайте должна быть функция просмотра учебного материала, а для администратора редактирование и добавление.

* 1. **Требования к надёжности**

Надежность системы в целом зависит от надежности используемого устройства.

* 1. **Условия эксплотации**

Стандартные условия эксплуатации программных продуктов

* 1. **Требования к составу и параметрам технических средств**

Для нормальной работы сайта необходимо:

* Стабильное интернет соединение и функционирующий браузер.
  1. **Требования к маркировке и упаковке**

Не предъявляются.

## **ПРИЛОЖЕНИЕ1 (*Продолжение)***

* 1. **Требования к транспортированию и хранению**

Не предъявляются.

* 1. **Специальные требования**

Не предъявляются.

1. **ТРЕБОВАНИЯ К ПРОГРАМНОЙ ДОКУМЕНТАЦИИ**

ГОСТ 19.105-78 – требования к оформлению программных документов, комплексов и систем независимо от их назначения и области применения. ГОСТ 19.105-78 содержит полный перечень документации, которая должна сопровождать законченный программный продукт.

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

»

**Пользовательские экраны**

ДП-02069846-09.02.07-21.05.90

ПРИЛОЖЕНИЕ 2

Web-сайт.

2 листа

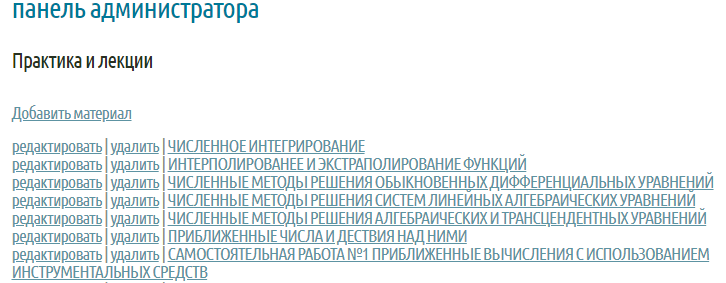
|  |  |
| --- | --- |
|  |  |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин |

«\_\_»\_\_\_\_\_\_2021

## **ПРИЛОЖЕНИЕ 2**

Рис. П2.1 Панель администратора

Рис.П2.2 Панель администратора после авторизации



## **ПРИЛОЖЕНИЕ 2(*Продолжение)***

Рис.П2.3 Загруженный учебный материал на сайте

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

**Руководство программиста**

ДП-02069846-09.02.07-21.05.33

ПРИЛОЖЕНИЕ 2

Web-сайт

1 лист

|  |  |
| --- | --- |
|  |  |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин |

«\_\_»\_\_\_\_\_\_2021

## **ПРИЛОЖЕНИЕ 2**

1. **ВВЕДЕНИЕ**

Программный продукт запускается с помощью файла Ebook.sln

1. **ОСНОВАНИЕ РАЗРАБОТКИ**

Основанием для разработки послужило задание на дипломный проект от ОГПОУ «Костромской политехнический колледж».

1. **НАЗНАЧЕНИЕ РАЗРАБОТКИ**

Программа называется «Электронное пособие по предмету “Численные методы”», и предназначена для изучения предмета самостоятельно иди дистанционно, а также для упрощения работы преподавателю

1. **ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ**

Приложение будет работать на любом устройстве с браузером.

1. **ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

Программная документация включает в себя:

* Описание программы
* Руководство программиста
* Руковдство оператора

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

»

**Руководство оператора**

ДП-02069846-09.02.07-21.05.34

ПРИЛОЖЕНИЕ 3

Web-сайт

1 лист

|  |  |
| --- | --- |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин  «\_\_»\_\_\_\_\_\_2021 |

## **ПРИЛОЖЕНИЕ 3**

1. **ИНФОРМАЦИОННАЯ ЧАСТЬ**

Сайт создан для изучения предмета самостоятельно иди дистанционно, а также для упрощения работы преподавателю.

1. **НАЗНАЧЕНИЕ ПРОГРАММЫ**

Программый продукт может быть использован студентами для изучения учебного материала дистанционно или самостоятельно.

1. **УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

Приложение будет работать на любых устройствах со стабильным интернетом и с функционирующем браузером.

1. **СООБЩЕНИЯ**

Сообщений нет.

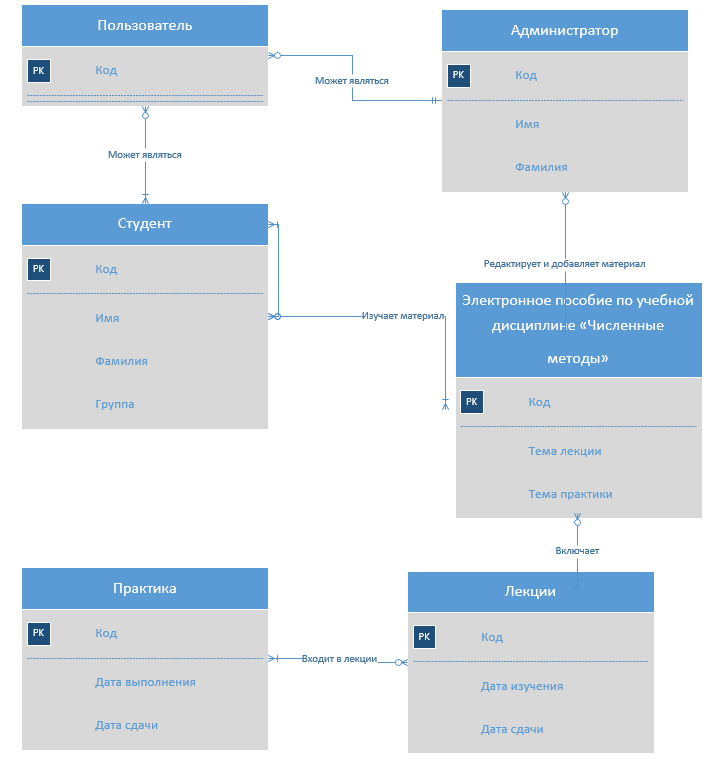
**5. СТРУКТУРАНЯ СХЕМА**

Изучение

Выбор нужного раздела

Запуск сайта

**6. ER-ДИАГРАММА**

****

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

**Паспорт программы**

ДП-02069846-09.02.07-21.05.05

ПРИЛОЖЕНИЕ 5

Web-сайт

1 лист

|  |  |
| --- | --- |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин  «\_\_»\_\_\_\_\_\_2021 |

## **ПРИЛОЖЕНИЕ 5**

1. **НАЗНАЧЕНИЕ**

Программый продукт может использоватся студентом или преподавателем любым. Приложение создано с целью изучения учебного материала самостоятельно или дистанционно,а также для упрощения работы преподаателю.

1. **ЗАПУСК ПРОГРАММЫ**

Программный продукт запускается с помощью файла EBook.sln

1. **ОБЪЁМ**

Программа занимает 50 МБ.

1. **СПЕЦИАЛЬНЫЕ УКАЗАНИЯ ОПЕРАТОРУ**

Специальных указаний нет.

1. **АВТОРЫ**

Студент группы 4-2П9 Волнухин Д.В.

1. **ДАТА СОЗДАНИЯ**

26.04.2021

областное государственное бюджетное профессиональное образовательное учреждение

**«КОСТРОМСКОЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

|  |
| --- |
| **УТВЕРЖДАЮ** Руководитель \_\_\_\_\_\_И.В. Лапшина «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г. |

Специальность:09.02.07«Информационные системы и программирование»

**ЭЛЕКТРОННОЕ ПОСОБИЕ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»**

**Текст программы**

ДП-02069846-09.02.07-21.04.12

ПРИЛОЖЕНИЕ 6

Web-сайт

10листов

|  |  |
| --- | --- |
| **Выполнил**: | \_\_\_\_\_\_\_\_\_Д.В. Волнухин  «\_\_»\_\_\_\_\_\_2021 |

Файл Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using EBook.Domain;

using EBook.Domain.Repositories.Abstract;

using EBook.Domain.Repositories.EntityFramework;

using EBook.Service;

namespace EBook

{

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration) => Configuration = configuration;

public void ConfigureServices(IServiceCollection services)

{

//подключаем конфиг из appsetting.json

Configuration.Bind("Project", new Config());

//подключаем нужный функционал приложения в качестве сервисов

services.AddTransient<ITextFieldsRepository, EFTextFieldsRepository>();

services.AddTransient<IServiceItemsRepository, EFServiceItemsRepository>();

services.AddTransient<DataManager>();

//подключаем контекст БД

services.AddDbContext<AppDbContext>(x => x.UseSqlServer(Config.ConnectionString));

//настраиваем identity систему

services.AddIdentity<IdentityUser, IdentityRole>(opts =>

{

opts.User.RequireUniqueEmail = true;

opts.Password.RequiredLength = 6;

opts.Password.RequireNonAlphanumeric = false;

opts.Password.RequireLowercase = false;

opts.Password.RequireUppercase = false;

opts.Password.RequireDigit = false;

}).AddEntityFrameworkStores<AppDbContext>().AddDefaultTokenProviders();

//настраиваем authentication cookie

services.ConfigureApplicationCookie(options =>

{

options.Cookie.Name = "myCompanyAuth";

options.Cookie.HttpOnly = true;

options.LoginPath = "/account/login";

options.AccessDeniedPath = "/account/accessdenied";

options.SlidingExpiration = true;

});

//настраиваем политику авторизации для Admin area

services.AddAuthorization(x =>

{

x.AddPolicy("AdminArea", policy => { policy.RequireRole("admin"); });

});

//добавляем сервисы для контроллеров и представлений (MVC)

services.AddControllersWithViews(x =>

{

x.Conventions.Add(new AdminAreaAuthorization("Admin", "AdminArea"));

})

//выставляем совместимость с asp.net core 3.0

.SetCompatibilityVersion(CompatibilityVersion.Version\_3\_0).AddSessionStateTempDataProvider();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

//!!! порядок регистрации middleware очень важен

//в процессе разработки нам важно видеть какие именно ошибки

if (env.IsDevelopment())

app.UseDeveloperExceptionPage();

//подключаем поддержку статичных файлов в приложении (css, js и т.д.)

app.UseStaticFiles();

//подключаем систему маршрутизации

app.UseRouting();

//подключаем аутентификацию и авторизацию

app.UseCookiePolicy();

app.UseAuthentication();

app.UseAuthorization();

//регистриуруем нужные нам маршруты (ендпоинты)

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute("admin", "{area:exists}/{controller=Home}/{action=Index}/{id?}");

endpoints.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

Файл Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace EBook

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

Файл compilerconfig.json

[

{

"outputFile": "wwwroot/css/main.css",

"inputFile": "wwwroot/sass/main.scss"

}

]

Файл bundleconfig.json

[

{

"outputFileName": "wwwroot/js/scripts.js",

"inputFiles": [

"wwwroot/js/jquery.min.js",

"wwwroot/js/browser.min.js",

"wwwroot/js/breakpoints.min.js",

"wwwroot/js/util.js",

"wwwroot/js/main.js"

]

}

]

Файл appsettings.json

{

"Project": {

"ConnectionString": "Data Source=(local)\\SQLEXPRESS; Database=MyCompany; Persist Security Info=false; User ID='sa'; Password='sa'; MultipleActiveResultSets=True; Trusted\_Connection=False;",

"EbookName": "Численные методы",

"EbookPhone": "+7 (111) 111-11-11",

"EbookEmail": "contact@mycompany.com"

}

}

Файл \_ViewStart.cshtml

@{

Layout = "\_Layout";

}

Файл \_ViewImports.cshtml

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@using EBook.Models

@using EBook.Service

@using EBook.Domain.Entities

@using EBook.Models.ViewComponents

Файл SidebarPartial.cshtml

<div class="col-4 col-12-medium">

<**cache**>

@(await Component.InvokeAsync<SidebarViewComponent>())

</**cache**>

</div>

Файл ScriptsPartial.cshtml

<script **src**="~/js/scripts.min.js" **asp-append-version**="true"></script>

Файл MetatagsPartial.cshtml

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />

@if (ViewBag.Title != null)

{

<title>@ViewBag.Title</title>

}

else

{

<title>@Config.EbookName</title>

}

@if (ViewBag.Description != null)

{

<meta name="description" content="@ViewBag.Description" />

}

@if (ViewBag.Keywords != null)

{

<meta name="keywords" content="@ViewBag.Keywords" />

}

Файл HeaderPartial.cshtml

<div id="header-wrapper">

<div class="container">

<div class="row">

<div class="col-12">

<header id="header">

<h1><a href="/" id="logo">@Config.CompanyName</a></h1>

<nav id="nav">

<a **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Главная</a>

<a **asp-area**="" **asp-controller**="Services" **asp-action**="Index" **asp-route-id**="">Практика и лекции</a>

<a **asp-area**="" **asp-controller**="Home" **asp-action**="Contacts">История возникновения</a>

</nav>

</header>

</div>

</div>

</div>

</div>

Файл FooterPartial.cshtml

!-- Footer -->

<div id="footer-wrapper">

<div class="container">

<div class="row">

<div class="col-12">

<div id="copyright">

&copy; @Config.EbookName.

</div>

</div>

</div>

</div>

</div>

Файл CssPartial.cshtml

<link rel="stylesheet" **href**="~/css/main.min.css" **asp-append-version**="true" />

Файл \_Layout.cshtml

<!DOCTYPE HTML>

<html>

<head>

@await Html.PartialAsync("MetatagsPartial")

@await Html.PartialAsync("CssPartial")

</head>

<body>

<div id="page-wrapper">

@await Html.PartialAsync("HeaderPartial")

<div id="main">

<div class="container">

<div class="row main-row">

<div class="col-8 col-12-medium">

@RenderBody()

</div>

@await Html.PartialAsync("SidebarPartial")

</div>

</div>

</div>

@await Html.PartialAsync("FooterPartial")

</div>

@await Html.PartialAsync("ScriptsPartial")

</body>

</html>

Файл Default.cshtml

@model IQueryable<ServiceItem>

@if (Model?.Any() == true)

{

<section>

<h2>Содержание</h2>

<ul class="small-image-list">

@foreach (ServiceItem entity in Model)

{

<li>

<a **asp-area**="" **asp-controller**="Services" **asp-action**="Index" **asp-route-id**="@entity.Id">

<img src="~/images/@entity.TitleImagePath" alt="" class="left" />

</a>

<h4>

@entity.Title

</h4>

<p>

@entity.Subtitle

</p>

</li>

}

</ul>

</section>

}

Файл Show.cshtml

@model ServiceItem

@{

ViewBag.Title = Model.MetaTitle;

ViewBag.Description = Model.MetaDescription;

ViewBag.Keywords = Model.MetaKeywords;

}

<div>

<h2>

@Model.Title

</h2>

<h2>

@Model.Subtitle

</h2>

<div>

<img class="img-block" src="~/images/@Model.TitleImagePath" />

</div>

</div>

<div>

@Html.Raw(Model.Text)

</div>

Файл Index.cshtml

@model IQueryable<ServiceItem>

@{

TextField textField = ViewBag.TextField;

ViewBag.Title = textField.MetaTitle;

ViewBag.Description = textField.MetaDescription;

ViewBag.Keywords = textField.MetaKeywords;

}

<div>

@if (Model.Any())

{

<ul class="big-image-list">

@foreach (ServiceItem entity in Model)

{

<li>

<a **asp-area**="" **asp-controller**="Services" **asp-action**="Index" **asp-route-id**="@entity.Id">

<img src="~/images/@entity.TitleImagePath" alt="" class="left" />

</a>

<h4>@entity.Title</h4>

<p>@entity.Subtitle</p>

</li>

}

</ul>

}

</div>

Файл Index.cshtml

@model TextField

@{

ViewBag.Title = Model.MetaTitle;

ViewBag.Description = Model.MetaDescription;

ViewBag.Keywords = Model.MetaKeywords;

}

<div>

@Html.Raw(Model.Text)

</div>

Файл Contacts.cshtml

@model TextField

@{

ViewBag.Title = Model.MetaTitle;

ViewBag.Description = Model.MetaDescription;

ViewBag.Keywords = Model.MetaKeywords;

}

<div>

@Html.Raw(Model.Text)

</div>

Файл Login.cshtml

@model LoginViewModel

<div>

<h1>Вход в личный кабинет</h1>

<div class="div-box">

<form **asp-area**="" **asp-controller**="Account" **asp-action**="Login" method="post" **asp-route-returnUrl**="@ViewBag.returnUrl">

<div **asp-validation-summary**="All"></div>

<div>

<label **asp-for**="UserName"></label>

<input **asp-for**="UserName" />

<span **asp-validation-for**="UserName"></span>

</div>

<div>

<label **asp-for**="Password"></label>

<input **asp-for**="Password" />

<span **asp-validation-for**="Password"></span>

</div>

<div>

<label **asp-for**="RememberMe"></label>

<input **asp-for**="RememberMe" />

<span **asp-validation-for**="RememberMe"></span>

</div>

<div>

<input type="submit" value="Войти" />

</div>

</form>

</div>

</div>

Файл Extensions.cs

namespace EBook.Service

{

public static class Extensions

{

public static string CutController(this string str)

{

return str.Replace("Controller", "");

}

}

}

Файл Config.cs

namespace EBook.Service

{

public class Config

{

public static string ConnectionString { get; set; }

public static string EbookName { get; set; }

public static string EbookPhone { get; set; }

public static string EbookEmail { get; set; }

}

}

Файл AdminAreaAuthorization.cs

using System;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.ApplicationModels;

using Microsoft.AspNetCore.Mvc.Authorization;

namespace EBook.Service

{

public class AdminAreaAuthorization : IControllerModelConvention

{

private readonly string area;

private readonly string policy;

public AdminAreaAuthorization(string area, string policy)

{

this.area = area;

this.policy = policy;

}

public void Apply(ControllerModel controller)

{

if (controller.Attributes.Any(a =>

a is AreaAttribute && (a as AreaAttribute).RouteValue.Equals(area, StringComparison.OrdinalIgnoreCase))

|| controller.RouteValues.Any(r =>

r.Key.Equals("area", StringComparison.OrdinalIgnoreCase) && r.Value.Equals(area, StringComparison.OrdinalIgnoreCase)))

{

controller.Filters.Add(new AuthorizeFilter(policy));

}

}

}

}

Файл LoginViewModel.cs

using System.ComponentModel.DataAnnotations;

namespace EBook.Models

{

public class LoginViewModel

{

[Required]

[Display(Name = "Логин")]

public string UserName { get; set; }

[Required]

[UIHint("password")]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Запомнить меня?")]

public bool RememberMe { get; set; }

}

}

Файл SidebarViewComponent.cs

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using EBook.Domain;

namespace EBook.Models.ViewComponents

{

public class SidebarViewComponent : ViewComponent

{

private readonly DataManager dataManager;

public SidebarViewComponent(DataManager dataManager)

{

this.dataManager = dataManager;

}

public Task<IViewComponentResult> InvokeAsync()

{

return Task.FromResult((IViewComponentResult) View("Default", dataManager.ServiceItems.GetServiceItems()));

}

}

}

Файл AppDbContextModelSnapshot.cs

// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using EBook.Domain;

namespace EBook.Migrations

{

[DbContext(typeof(AppDbContext))]

partial class AppDbContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.1")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

b.HasData(

new

{

Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",

ConcurrencyStamp = "0b0f0920-b522-4ade-a8df-070e56a082af",

Name = "admin",

NormalizedName = "ADMIN"

});

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

b.HasData(

new

{

Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",

AccessFailedCount = 0,

ConcurrencyStamp = "bcdaf249-77f5-4638-887b-115f917d4622",

Email = "my@email.com",

EmailConfirmed = true,

LockoutEnabled = false,

NormalizedEmail = "MY@EMAIL.COM",

NormalizedUserName = "ADMIN",

PasswordHash = "AQAAAAEAACcQAAAAEL6OhqfHLlRaAiB4egTuziIsGqDbZj9G33HZHdbPkhp3ZC6YXF1iypXhLFraEyD/lQ==",

PhoneNumberConfirmed = false,

SecurityStamp = "",

TwoFactorEnabled = false,

UserName = "admin"

});

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(450)");

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(450)");

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

b.HasData(

new

{

UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8",

RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab"

});

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(450)");

b.Property<string>("Name")

.HasColumnType("nvarchar(450)");

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("MyCompany.Domain.Entities.ServiceItem", b =>

{

b.Property<Guid>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("uniqueidentifier");

b.Property<DateTime>("DateAdded")

.HasColumnType("datetime2");

b.Property<string>("MetaDescription")

.HasColumnType("nvarchar(max)");

b.Property<string>("MetaKeywords")

.HasColumnType("nvarchar(max)");

b.Property<string>("MetaTitle")

.HasColumnType("nvarchar(max)");

b.Property<string>("Subtitle")

.HasColumnType("nvarchar(max)");

b.Property<string>("Text")

.HasColumnType("nvarchar(max)");

b.Property<string>("Title")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("TitleImagePath")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("ServiceItems");

});

modelBuilder.Entity("MyCompany.Domain.Entities.TextField", b =>

{

b.Property<Guid>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("uniqueidentifier");

b.Property<string>("CodeWord")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<DateTime>("DateAdded")

.HasColumnType("datetime2");

b.Property<string>("MetaDescription")

.HasColumnType("nvarchar(max)");

b.Property<string>("MetaKeywords")

.HasColumnType("nvarchar(max)");

b.Property<string>("MetaTitle")

.HasColumnType("nvarchar(max)");

b.Property<string>("Subtitle")

.HasColumnType("nvarchar(max)");

b.Property<string>("Text")

.HasColumnType("nvarchar(max)");

b.Property<string>("Title")

.HasColumnType("nvarchar(max)");

b.Property<string>("TitleImagePath")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("TextFields");

b.HasData(

new

{

Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),

CodeWord = "PageIndex",

DateAdded = new DateTime(2021, 3, 23, 15, 27, 14, 832, DateTimeKind.Utc).AddTicks(6622),

Text = "Содержание заполняется администратором",

Title = "Главная"

},

new

{

Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),

CodeWord = "PageServices",

DateAdded = new DateTime(2021, 3, 23, 15, 27, 14, 833, DateTimeKind.Utc).AddTicks(201),

Text = "Содержание заполняется администратором",

Title = "Наши услуги"

},

new

{

Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),

CodeWord = "PageContacts",

DateAdded = new DateTime(2021, 3, 23, 15, 27, 14, 833, DateTimeKind.Utc).AddTicks(264),

Text = "Содержание заполняется администратором",

Title = "Контакты"

});

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

Файл 20210323152715\_\_initial.cs

using System;

using Microsoft.EntityFrameworkCore.Migrations;

namespace EBook.Migrations

{

public partial class \_initial : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "AspNetRoles",

columns: table => new

{

Id = table.Column<string>(nullable: false),

Name = table.Column<string>(maxLength: 256, nullable: true),

NormalizedName = table.Column<string>(maxLength: 256, nullable: true),

ConcurrencyStamp = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetRoles", x => x.Id);

});

migrationBuilder.CreateTable(

name: "AspNetUsers",

columns: table => new

{

Id = table.Column<string>(nullable: false),

UserName = table.Column<string>(maxLength: 256, nullable: true),

NormalizedUserName = table.Column<string>(maxLength: 256, nullable: true),

Email = table.Column<string>(maxLength: 256, nullable: true),

NormalizedEmail = table.Column<string>(maxLength: 256, nullable: true),

EmailConfirmed = table.Column<bool>(nullable: false),

PasswordHash = table.Column<string>(nullable: true),

SecurityStamp = table.Column<string>(nullable: true),

ConcurrencyStamp = table.Column<string>(nullable: true),

PhoneNumber = table.Column<string>(nullable: true),

PhoneNumberConfirmed = table.Column<bool>(nullable: false),

TwoFactorEnabled = table.Column<bool>(nullable: false),

LockoutEnd = table.Column<DateTimeOffset>(nullable: true),

LockoutEnabled = table.Column<bool>(nullable: false),

AccessFailedCount = table.Column<int>(nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetUsers", x => x.Id);

});

migrationBuilder.CreateTable(

name: "ServiceItems",

columns: table => new

{

Id = table.Column<Guid>(nullable: false),

TitleImagePath = table.Column<string>(nullable: true),

MetaTitle = table.Column<string>(nullable: true),

MetaDescription = table.Column<string>(nullable: true),

MetaKeywords = table.Column<string>(nullable: true),

DateAdded = table.Column<DateTime>(nullable: false),

Title = table.Column<string>(nullable: false),

Subtitle = table.Column<string>(nullable: true),

Text = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_ServiceItems", x => x.Id);

});

migrationBuilder.CreateTable(

name: "TextFields",

columns: table => new

{

Id = table.Column<Guid>(nullable: false),

Subtitle = table.Column<string>(nullable: true),

TitleImagePath = table.Column<string>(nullable: true),

MetaTitle = table.Column<string>(nullable: true),

MetaDescription = table.Column<string>(nullable: true),

MetaKeywords = table.Column<string>(nullable: true),

DateAdded = table.Column<DateTime>(nullable: false),

CodeWord = table.Column<string>(nullable: false),

Title = table.Column<string>(nullable: true),

Text = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_TextFields", x => x.Id);

});

migrationBuilder.CreateTable(

name: "AspNetRoleClaims",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

RoleId = table.Column<string>(nullable: false),

ClaimType = table.Column<string>(nullable: true),

ClaimValue = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetRoleClaims", x => x.Id);

table.ForeignKey(

name: "FK\_AspNetRoleClaims\_AspNetRoles\_RoleId",

column: x => x.RoleId,

principalTable: "AspNetRoles",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "AspNetUserClaims",

columns: table => new

{

Id = table.Column<int>(nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

UserId = table.Column<string>(nullable: false),

ClaimType = table.Column<string>(nullable: true),

ClaimValue = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetUserClaims", x => x.Id);

table.ForeignKey(

name: "FK\_AspNetUserClaims\_AspNetUsers\_UserId",

column: x => x.UserId,

principalTable: "AspNetUsers",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "AspNetUserLogins",

columns: table => new

{

LoginProvider = table.Column<string>(nullable: false),

ProviderKey = table.Column<string>(nullable: false),

ProviderDisplayName = table.Column<string>(nullable: true),

UserId = table.Column<string>(nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetUserLogins", x => new { x.LoginProvider, x.ProviderKey });

table.ForeignKey(

name: "FK\_AspNetUserLogins\_AspNetUsers\_UserId",

column: x => x.UserId,

principalTable: "AspNetUsers",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "AspNetUserRoles",

columns: table => new

{

UserId = table.Column<string>(nullable: false),

RoleId = table.Column<string>(nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetUserRoles", x => new { x.UserId, x.RoleId });

table.ForeignKey(

name: "FK\_AspNetUserRoles\_AspNetRoles\_RoleId",

column: x => x.RoleId,

principalTable: "AspNetRoles",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_AspNetUserRoles\_AspNetUsers\_UserId",

column: x => x.UserId,

principalTable: "AspNetUsers",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "AspNetUserTokens",

columns: table => new

{

UserId = table.Column<string>(nullable: false),

LoginProvider = table.Column<string>(nullable: false),

Name = table.Column<string>(nullable: false),

Value = table.Column<string>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_AspNetUserTokens", x => new { x.UserId, x.LoginProvider, x.Name });

table.ForeignKey(

name: "FK\_AspNetUserTokens\_AspNetUsers\_UserId",

column: x => x.UserId,

principalTable: "AspNetUsers",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.InsertData(

table: "AspNetRoles",

columns: new[] { "Id", "ConcurrencyStamp", "Name", "NormalizedName" },

values: new object[] { "44546e06-8719-4ad8-b88a-f271ae9d6eab", "0b0f0920-b522-4ade-a8df-070e56a082af", "admin", "ADMIN" });

migrationBuilder.InsertData(

table: "AspNetUsers",

columns: new[] { "Id", "AccessFailedCount", "ConcurrencyStamp", "Email", "EmailConfirmed", "LockoutEnabled", "LockoutEnd", "NormalizedEmail", "NormalizedUserName", "PasswordHash", "PhoneNumber", "PhoneNumberConfirmed", "SecurityStamp", "TwoFactorEnabled", "UserName" },

values: new object[] { "3b62472e-4f66-49fa-a20f-e7685b9565d8", 0, "bcdaf249-77f5-4638-887b-115f917d4622", "my@email.com", true, false, null, "MY@EMAIL.COM", "ADMIN", "AQAAAAEAACcQAAAAEL6OhqfHLlRaAiB4egTuziIsGqDbZj9G33HZHdbPkhp3ZC6YXF1iypXhLFraEyD/lQ==", null, false, "", false, "admin" });

migrationBuilder.InsertData(

table: "TextFields",

columns: new[] { "Id", "CodeWord", "DateAdded", "MetaDescription", "MetaKeywords", "MetaTitle", "Subtitle", "Text", "Title", "TitleImagePath" },

values: new object[,]

{

{ new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"), "PageIndex", new DateTime(2021, 3, 23, 15, 27, 14, 832, DateTimeKind.Utc).AddTicks(6622), null, null, null, null, "Содержание заполняется администратором", "Главная", null },

{ new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"), "PageServices", new DateTime(2021, 3, 23, 15, 27, 14, 833, DateTimeKind.Utc).AddTicks(201), null, null, null, null, "Содержание заполняется администратором", "Наши услуги", null },

{ new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"), "PageContacts", new DateTime(2021, 3, 23, 15, 27, 14, 833, DateTimeKind.Utc).AddTicks(264), null, null, null, null, "Содержание заполняется администратором", "Контакты", null }

});

migrationBuilder.InsertData(

table: "AspNetUserRoles",

columns: new[] { "UserId", "RoleId" },

values: new object[] { "3b62472e-4f66-49fa-a20f-e7685b9565d8", "44546e06-8719-4ad8-b88a-f271ae9d6eab" });

migrationBuilder.CreateIndex(

name: "IX\_AspNetRoleClaims\_RoleId",

table: "AspNetRoleClaims",

column: "RoleId");

migrationBuilder.CreateIndex(

name: "RoleNameIndex",

table: "AspNetRoles",

column: "NormalizedName",

unique: true,

filter: "[NormalizedName] IS NOT NULL");

migrationBuilder.CreateIndex(

name: "IX\_AspNetUserClaims\_UserId",

table: "AspNetUserClaims",

column: "UserId");

migrationBuilder.CreateIndex(

name: "IX\_AspNetUserLogins\_UserId",

table: "AspNetUserLogins",

column: "UserId");

migrationBuilder.CreateIndex(

name: "IX\_AspNetUserRoles\_RoleId",

table: "AspNetUserRoles",

column: "RoleId");

migrationBuilder.CreateIndex(

name: "EmailIndex",

table: "AspNetUsers",

column: "NormalizedEmail");

migrationBuilder.CreateIndex(

name: "UserNameIndex",

table: "AspNetUsers",

column: "NormalizedUserName",

unique: true,

filter: "[NormalizedUserName] IS NOT NULL");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "AspNetRoleClaims");

migrationBuilder.DropTable(

name: "AspNetUserClaims");

migrationBuilder.DropTable(

name: "AspNetUserLogins");

migrationBuilder.DropTable(

name: "AspNetUserRoles");

migrationBuilder.DropTable(

name: "AspNetUserTokens");

migrationBuilder.DropTable(

name: "ServiceItems");

migrationBuilder.DropTable(

name: "TextFields");

migrationBuilder.DropTable(

name: "AspNetRoles");

migrationBuilder.DropTable(

name: "AspNetUsers");

}

}

}

Файл DataManager.cs

using EBook.Domain.Repositories.Abstract;

namespace EBook.Domain

{

public class DataManager

{

public ITextFieldsRepository TextFields { get; set; }

public IServiceItemsRepository ServiceItems { get; set; }

public DataManager(ITextFieldsRepository textFieldsRepository, IServiceItemsRepository serviceItemsRepository)

{

TextFields = textFieldsRepository;

ServiceItems = serviceItemsRepository;

}

}

}

Файл AppDbContext.cs

using System;

using EBook.Domain.Entities;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace EBook.Domain

{

public class AppDbContext : IdentityDbContext<IdentityUser>

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<TextField> TextFields { get; set; }

public DbSet<ServiceItem> ServiceItems { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<IdentityRole>().HasData(new IdentityRole

{

Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",

Name = "admin",

NormalizedName = "ADMIN"

});

modelBuilder.Entity<IdentityUser>().HasData(new IdentityUser

{

Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",

UserName = "admin",

NormalizedUserName = "ADMIN",

Email = "my@email.com",

NormalizedEmail = "MY@EMAIL.COM",

EmailConfirmed = true,

PasswordHash = new PasswordHasher<IdentityUser>().HashPassword(null, "superpassword"),

SecurityStamp = string.Empty

});

modelBuilder.Entity<IdentityUserRole<string>>().HasData(new IdentityUserRole<string>

{

RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab",

UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8"

});

modelBuilder.Entity<TextField>().HasData(new TextField {

Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),

CodeWord = "PageIndex",

Title = "Главная"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),

CodeWord = "PageServices",

Title = "Практика и лекции"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),

CodeWord = "PageContacts",

Title = "История возникновения"

});

}

}

}

Файл EFTextFieldsRepository.cs

using System;

using System.Linq;

using EBook.Domain.Entities;

using EBook.Domain.Repositories.Abstract;

using Microsoft.EntityFrameworkCore;

namespace EBook.Domain.Repositories.EntityFramework

{

public class EFTextFieldsRepository : ITextFieldsRepository

{

private readonly AppDbContext context;

public EFTextFieldsRepository(AppDbContext context)

{

this.context = context;

}

public IQueryable<TextField> GetTextFields()

{

return context.TextFields;

}

public TextField GetTextFieldById(Guid id)

{

return context.TextFields.FirstOrDefault(x => x.Id == id);

}

public void SaveTextField(TextField entity)

{

if (entity.Id == default)

context.Entry(entity).State = EntityState.Added;

else

context.Entry(entity).State = EntityState.Modified;

context.SaveChanges();

}

public void DeleteTextField(Guid id)

{

context.TextFields.Remove(new TextField() { Id = id });

context.SaveChanges();

}

public TextField GetTextFieldByCodeWord(string codeWord)

{

return context.TextFields.FirstOrDefault(x => x.CodeWord == codeWord);

}

}

}

Файл EFServiceItemsRepository.cs

using System;

using System.Linq;

using EBook.Domain.Entities;

using EBook.Domain.Repositories.Abstract;

using Microsoft.EntityFrameworkCore;

namespace EBook.Domain.Repositories.EntityFramework

{

public class EFServiceItemsRepository : IServiceItemsRepository

{

private readonly AppDbContext context;

public EFServiceItemsRepository(AppDbContext context)

{

this.context = context;

}

public IQueryable<ServiceItem> GetServiceItems()

{

return context.ServiceItems;

}

public ServiceItem GetServiceItemById(Guid id)

{

return context.ServiceItems.FirstOrDefault(x => x.Id == id);

}

public void SaveServiceItem(ServiceItem entity)

{

if (entity.Id == default)

context.Entry(entity).State = EntityState.Added;

else

context.Entry(entity).State = EntityState.Modified;

context.SaveChanges();

}

public void DeleteServiceItem(Guid id)

{

context.ServiceItems.Remove(new ServiceItem() { Id = id });

context.SaveChanges();

}

}

}

Файл ITextFieldsRepository.cs

using EBook.Domain.Entities;

using System;

using System.Linq;

namespace EBook.Domain.Repositories.Abstract

{

public interface ITextFieldsRepository

{

IQueryable<TextField> GetTextFields();

TextField GetTextFieldById(Guid id);

TextField GetTextFieldByCodeWord(string codeWord);

void SaveTextField(TextField entity);

void DeleteTextField(Guid id);

}

}

Файл IServiceItemsRepository.cs

using EBook.Domain.Entities;

using System;

using System.Linq;

namespace EBook.Domain.Repositories.Abstract

{

public interface IServiceItemsRepository

{

IQueryable<ServiceItem> GetServiceItems();

ServiceItem GetServiceItemById(Guid id);

void SaveServiceItem(ServiceItem entity);

void DeleteServiceItem(Guid id);

}

}

Файл TextField.cs

using System.ComponentModel.DataAnnotations;

namespace EBook.Domain.Entities

{

public class TextField : EntityBase

{

[Required]

public string CodeWord { get; set; }

[Display(Name = "Название страницы (заголовок)")]

public override string Title { get; set; } = "Информационная страница";

[Display(Name = "Cодержание страницы")]

public override string Text { get; set; } = "Содержание заполняется администратором";

}

}

Файл ServiceItem.cs

using System.ComponentModel.DataAnnotations;

namespace EBook.Domain.Entities

{

public class ServiceItem : EntityBase

{

[Required(ErrorMessage = "Заполните название темы")]

[Display(Name = "Название темы")]

public override string Title { get; set; }

[Display(Name = "Краткое описание темы")]

public override string Subtitle { get; set; }

[Display(Name = "Полное описание темы")]

public override string Text { get; set; }

}

}

Файл EntityBase.cs

using System;

using System.ComponentModel.DataAnnotations;

namespace EBook.Domain.Entities

{

public abstract class EntityBase

{

protected EntityBase() => DateAdded = DateTime.UtcNow;

[Required]

public Guid Id { get;set; }

[Display(Name = "Название (заголовок)")]

public virtual string Title { get; set; }

[Display(Name = "Краткое описание")]

public virtual string Subtitle { get; set; }

[Display(Name = "Полное описание")]

public virtual string Text { get; set; }

[Display(Name = "Титульная картинка")]

public virtual string TitleImagePath { get; set; }

[Display(Name = "SEO метатег Title")]

public string MetaTitle { get; set; }

[Display(Name = "SEO метатег Description")]

public string MetaDescription { get; set; }

[Display(Name = "SEO метатег Keywords")]

public string MetaKeywords { get; set; }

[DataType(DataType.Time)]

public DateTime DateAdded { get; set; }

}

}

Файл ServicesController.cs

using EBook.Domain;

using Microsoft.AspNetCore.Mvc;

using System;

namespace EBook.Controllers

{

public class ServicesController : Controller

{

private readonly DataManager dataManager;

public ServicesController(DataManager dataManager)

{

this.dataManager = dataManager;

}

public IActionResult Index(Guid id)

{

if (id != default)

{

return View("Show", dataManager.ServiceItems.GetServiceItemById(id));

}

ViewBag.TextField = dataManager.TextFields.GetTextFieldByCodeWord("PageServices");

return View(dataManager.ServiceItems.GetServiceItems());

}

}

}

Файл HomeController.cs

using EBook.Domain;

using Microsoft.AspNetCore.Mvc;

namespace EBook.Controllers

{

public class HomeController : Controller

{

private readonly DataManager dataManager;

public HomeController(DataManager dataManager)

{

this.dataManager = dataManager;

}

public IActionResult Index()

{

return View(dataManager.TextFields.GetTextFieldByCodeWord("PageIndex"));

}

public IActionResult Contacts()

{

return View(dataManager.TextFields.GetTextFieldByCodeWord("PageContacts"));

}

}

}

Файл AccountController.cs

using EBook.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

namespace EBook.Controllers

{

[Authorize]

public class AccountController : Controller

{

private readonly UserManager<IdentityUser> userManager;

private readonly SignInManager<IdentityUser> signInManager;

public AccountController(UserManager<IdentityUser> userMgr, SignInManager<IdentityUser> signinMgr)

{

userManager = userMgr;

signInManager = signinMgr;

}

[AllowAnonymous]

public IActionResult Login(string returnUrl)

{

ViewBag.returnUrl = returnUrl;

return View(new LoginViewModel());

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl)

{

if (ModelState.IsValid)

{

IdentityUser user = await userManager.FindByNameAsync(model.UserName);

if (user != null)

{

await signInManager.SignOutAsync();

Microsoft.AspNetCore.Identity.SignInResult result = await signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

return Redirect(returnUrl ?? "/");

}

}

ModelState.AddModelError(nameof(LoginViewModel.UserName), "Неверный логин или пароль");

}

return View(model);

}

[Authorize]

public async Task<IActionResult> Logout()

{

await signInManager.SignOutAsync();

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

}

}

}

Файл Edit.cshtml (TextFields)

@model TextField

@{

string strTitle = "Редактировать запись";

ViewBag.Title = strTitle;

}

<script src="~/js/ckeditor/ckeditor.js"></script>

<div>

<h2>@strTitle</h2>

<div>

<form **asp-area**="Admin" **asp-controller**="TextFields" **asp-action**="Edit" method="post" enctype="multipart/form-data">

<input **type**="hidden" **asp-for**="Id" />

<input **type**="hidden" **asp-for**="DateAdded" />

<input **type**="hidden" **asp-for**="CodeWord" />

<div **asp-validation-summary**="All"></div>

<div class="div-box">

<label **asp-for**="Title"></label>

<input **asp-for**="Title" />

<span **asp-validation-for**="Title"></span>

</div>

<div class="div-box">

<label **asp-for**="Text"></label>

<textarea **asp-for**="Text"></textarea>

<span **asp-validation-for**="Text"></span>

</div>

<div class="div-box">

<label **asp-for**="MetaTitle"></label>

<input **asp-for**="MetaTitle" />

<span **asp-validation-for**="MetaTitle"></span>

</div>

<div class="div-box">

<label **asp-for**="MetaDescription"></label>

<input **asp-for**="MetaDescription" />

<span **asp-validation-for**="MetaDescription"></span>

</div>

<div class="div-box">

<label **asp-for**="MetaKeywords"></label>

<input **asp-for**="MetaKeywords" />

<span **asp-validation-for**="MetaKeywords"></span>

</div>

<input type="submit" value="Сохранить" />

</form>

</div>

</div>

<script>

window.onload = function() {

var newCKEdit = CKEDITOR.replace('@Html.IdFor(x=>x.Text)');

newCKEdit.updateElement();

}

</script>

Файл Index.cshtml

@model IQueryable<ServiceItem>

@{

string strTitle = "Панель администратора";

ViewBag.Title = strTitle;

}

<div>

<h2>@strTitle</h2>

<div>

<h3>Практика и лекции</h3>

<div class="div-box">

<a **asp-area**="Admin" **asp-controller**="ServiceItems" **asp-action**="Edit" **asp-route-id**="">Добавить материал</a>

</div>

@if (Model.Any())

{

<div>

@foreach (ServiceItem entity in Model)

{

<div>

<a **asp-area**="Admin" **asp-controller**="ServiceItems" **asp-action**="Edit" **asp-route-id**="@entity.Id">редактировать</a>

|

<form style="display: inline-block;" id="form-@entity.Id" **asp-area**="Admin" **asp-controller**="ServiceItems" **asp-action**="Delete" method="post">

<input type="hidden" name="id" value="@entity.Id">

<a href="#" onclick="document.getElementById('form-@entity.Id').submit();">удалить</a>

</form>

|

<a **asp-area**="" **asp-controller**="Services" **asp-action**="Index" **asp-route-id**="@entity.Id">

@($"{entity.Title}")

</a>

</div>

}

</div>

}

</div>

<div class="div-box">

<h3>Редактировать страницы сайта</h3>

<a **asp-area**="Admin" **asp-controller**="TextFields" **asp-action**="Edit" **asp-route-codeWord**="PageIndex">Главная</a>

<a **asp-area**="Admin" **asp-controller**="TextFields" **asp-action**="Edit" **asp-route-codeWord**="PageServices">Практика и лекции</a>

<a **asp-area**="Admin" **asp-controller**="TextFields" **asp-action**="Edit" **asp-route-codeWord**="PageContacts">История возникновения</a>

</div>

<div class="div-box">

<form **asp-area**="" **asp-controller**="Account" **asp-action**="Logout" method="post">

<input type="submit" value="Выйти" />

</form>

</div>

</div>

Файл TextFieldsController.cs

using Microsoft.AspNetCore.Mvc;

using EBook.Domain;

using EBook.Domain.Entities;

using EBook.Service;

namespace EBook.Areas.Admin.Controllers

{

[Area("Admin")]

public class TextFieldsController : Controller

{

private readonly DataManager dataManager;

public TextFieldsController(DataManager dataManager)

{

this.dataManager = dataManager;

}

public IActionResult Edit(string codeWord)

{

var entity = dataManager.TextFields.GetTextFieldByCodeWord(codeWord);

return View(entity);

}

[HttpPost]

public IActionResult Edit(TextField model)

{

if (ModelState.IsValid)

{

dataManager.TextFields.SaveTextField(model);

return RedirectToAction(nameof(HomeController.Index), nameof(HomeController).CutController());

}

return View(model);

}

}

}

Файл ServiceItemsController.cs

using System;

using System.IO;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using EBook.Domain;

using EBook.Domain.Entities;

using EBook.Service;

namespace EBook.Areas.Admin.Controllers

{

[Area("Admin")]

public class ServiceItemsController : Controller

{

private readonly DataManager dataManager;

private readonly IWebHostEnvironment hostingEnvironment;

public ServiceItemsController(DataManager dataManager, IWebHostEnvironment hostingEnvironment)

{

this.dataManager = dataManager;

this.hostingEnvironment = hostingEnvironment;

}

public IActionResult Edit(Guid id)

{

var entity = id == default ? new ServiceItem() : dataManager.ServiceItems.GetServiceItemById(id);

return View(entity);

}

[HttpPost]

public IActionResult Edit(ServiceItem model, IFormFile titleImageFile)

{

if (ModelState.IsValid)

{

if (titleImageFile != null)

{

model.TitleImagePath = titleImageFile.FileName;

using (var stream = new FileStream(Path.Combine(hostingEnvironment.WebRootPath, "images/", titleImageFile.FileName), FileMode.Create))

{

titleImageFile.CopyTo(stream);

}

}

dataManager.ServiceItems.SaveServiceItem(model);

return RedirectToAction(nameof(HomeController.Index), nameof(HomeController).CutController());

}

return View(model);

}

[HttpPost]

public IActionResult Delete(Guid id)

{

dataManager.ServiceItems.DeleteServiceItem(id);

return RedirectToAction(nameof(HomeController.Index), nameof(HomeController).CutController());

}

}

}

Файл HomeController.cs

using EBook.Domain;

using Microsoft.AspNetCore.Mvc;

namespace EBook.Areas.Admin.Controllers

{

[Area("Admin")]

public class HomeController : Controller

{

private readonly DataManager dataManager;

public HomeController(DataManager dataManager)

{

this.dataManager = dataManager;

}

public IActionResult Index()

{

return View(dataManager.ServiceItems.GetServiceItems());

}

}

}