**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1–40 04 01 «Информатика и технологии программирования»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Избранные главы информатики»

на тему: «Новостной портал»

Исполнитель: студент гр. ИП–31

В.А. Дранев

Руководитель: старший преподаватель

Т.Л. Романькова

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2021

Содержание

[Введение 5](#_Toc72364001)

[1 Теоретические сведения 6](#_Toc72364002)

[1.1 Обзор существующих программных продуктов 6](#_Toc72364003)

[1.2 Постановка задачи 6](#_Toc72364004)

[1.3 Язык и среда программирования 7](#_Toc72364005)

[2 Алгоритмический анализ 12](#_Toc72364006)

[2.1 База данных 12](#_Toc72364007)

[2.2 Анализ задания на проектирование 14](#_Toc72364008)

[3 Разработка программного кода 25](#_Toc72364009)

[3.1 Структурная схема программы 25](#_Toc72364010)

[3.2 Создание приложения 25](#_Toc72364011)

[3.3 Интерфейс пользователя 31](#_Toc72364012)

[Заключение 38](#_Toc72364013)

[Список использованных источников 39](#_Toc72364014)

[Приложение A 40](#_Toc72364015)

[Проект EntityModels 40](#_Toc72364016)

[Проект BLL 46](#_Toc72364017)

[Проект CourseWork 49](#_Toc72364018)

# ВВЕДЕНИЕ

В настоящее время мы можем наблюдать, что жизнь современного человека все больше переходит в формат онлайн. Неотъемлемой частью такой жизни являются различные информационные ресурсы, которые значительно экономят время человека и одновременно оперативно предоставляют необходимую информацию, руководствуясь его интересами. К таким источникам относятся новостные порталы, порталы с новинками кино, музыки, игр. Также мы можем видеть, что пользователи данных порталов с каждым годом становятся более избирательны и требовательны не только к эффективности их работы, но и внешнему виду и удобству использования.

В связи с этим создания такого рода порталов стало достаточно востребовано в настоящее время.

Интернет–портал – это многофункциональная площадка с разнообразным интерактивным сервисом, включающая в себя обширные возможности и услуги, в том числе путём предоставления пользователям ссылок на другие сайты.

Можно привести следующую классификацию интернет–порталов:

* горизонтальный или универсальный интернет–портал, представляет собой сайт, охватывающий различные темы и предлагающий пользователю широкий набор сервисов, ориентированный на максимально широкую аудиторию. Такие порталы, как правило, совмещают в себе разнообразные функции, предлагают разноплановый контент и разнообразные сервисы: новости, финансовые сводки, информацию о погоде, интерактивные, развлекательные и игровые сервисы;
* вертикальный интернет–портал – это веб–ресурс узкой тематической направленности, предоставляющий пользователям различные сервисы по определенным интересам.

С целью собственного продвижения, многие наполняют страницы своих сайтов разнообразной и максимально подробной справочной информацией, относящейся к предлагаемым товарам, статьями по тематике магазина, превращаясь, таким образом, в смешанные порталы, сочетающие в себе достаточный объем информационных ресурсов и серьезную бизнес–составляющую.

Цель – разработать новостной портал, который несет в себе информацию, максимально доступно и удобно подает ее пользователю, обеспечивает актуальность и полезность этой информации.

Таким образом в современном мире, в эпоху большого потока информации хороший новостной сайт играет важную роль в жизни каждого человека.

# **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

## **Обзор существующих программных продуктов**

Рассмотрим подробнее новостной портал.

Новостной сайт – веб–сайт, который специализируется на публикации новостей в интернете, является частным проявлением интернет–издания.

Новостные сайты могут быть не привязаны к какой–то, определённой категории новостей. Это значит, что на его основе можно сделать новостной сайт любой тематики, например: новости образования, новости медицины, новости экономики, и так далее. Однако, существуют также и сайты, которые специализируются на публикации новостей определенной тематики.

Главный плюс новостных сайтов – это актуальность и разнообразие информации. Помимо ежедневно и даже ежечасного обновления новостей, можно узнать прогноз погоды в своем регионе, развлекательные статьи и интересные факты.

Большинство новостных ресурсов имеют удобный интерфейс и навигацию. Сначала размещаются самые свежие и актуальные новости, а в колонках с разделами можно найти интересные авторские статьи, рейтинги и другие полезные материалы.

Многие сайты предоставляют удобный и информативный поиск по разным категориям, будь то заголовок статьи, часть текста, автор, количество просмотров, количество комментариев, время добавления, популярность.

Тематические новостные порталы, к примеру, портал новостей спорта, портал новостей погоды, портал игровых новостей имеют тематичкое оформление и используют в статьях слова характерные только данной тематической области.

Чтобы выбрать качественный новостной портал, можно использовать следующие критерии:

* актуальность и скорость подачи информации;
* правдивость;
* доступность;
* экономия времени пользователей;
* привлекательный дизайн и удобный интерфейс.

## **Постановка задачи**

Жизнь современного человека все больше переходит в формат онлайн. Неотъемлемой частью такой жизни являются различные информационные ресурсы, которые должны экономить время человека и одновременно оперативно предоставлять необходимую информацию, руководствуясь его интересами и интересами других пользователей, например, просмотр наиболее популярной статьи среди пользователей сайта.

Целью курсового проекта является разработка и создание новостного портала, который позволял бы просматривать статьи, оставлять отзывы о них, сортировать статьи для поиска интересующей записи, также сайт должен позволять пользователям создавать свои статьи и управлять ими из личного кабинета.

Помимо вышеуказанных функций, новостной сайт должен позволять незарегистрированным пользователям просмотривать статьи и комментарии к ним и позволять им зарегистрироваться для создания своих статей. Также сайт должен позволять просматривать информацию о пользователе и позволять редактировать свои данные на сайте.

Управление пользователями, как и статьями, должно осуществляться с помощью пользователей с большим уровнем доступа, что накладывает на проект требование о существовании уровней доступа. Из задачи можно выделить три уровня доступа:

* зарегистрированный пользователь;
* менеджер;
* администратор.

## **Язык и среда программирования**

Согласно [3] язык C# и связанную с ним среду [.NET Framework](https://professorweb.ru/my/csharp/charp_theory/level1/1_3.php) можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков.

Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для использования в .NET Framework.

Например, с применением C# и .NET Framework можно создавать динамические веб–страницы, приложения Windows Presentation Foundation, веб–службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах.

C# является объектно–ориентированным языком. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности.

Главная особенность таких компонентов в том, что они представляют собой модель программирования со свойствами, методами и событиями. У них есть атрибуты, предоставляющие декларативные сведения о компоненте. Они включают в себя собственную документацию.

C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# подходит для создания и применения программных компонентов.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

1. поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) – своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках;
2. кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ – Windows, MacOS, Linux, Android, iOS, Tizen;
3. мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов.

Согласно [1] платформа **ASP.NET Core** представляет технологию от компании Microsoft, предназначенную для создания различного рода веб–приложений: от небольших веб–сайтов до крупных веб–порталов и веб–сервисов.

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

Разработка над платформой началась еще в 2014 году. Тогда платформа условно называлась ASP.NET vNext. В июне 2016 года вышел первый релиз платформы. А в декабре 2019 года вышла версия ASP.NET Core 3.1, которая собственно и будет охвачена в текущем руководстве.

ASP.NET Core теперь полностью является opensource–фреймворком. Все исходные файлы фреймворка доступны на [GitHub](https://github.com/aspnet/).

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

И таким образом, с помощью ASP.NET Core мы можем создавать кросс–платформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой. То есть мы можем запускать веб–приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания веб–приложения можно использовать традиционный IIS, либо кросс–платформенный веб–сервер Kestrel.

Благодаря модульности фреймворка все необходимые компоненты веб–приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

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. А Web Forms полностью ушли в прошлое.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций.

Одной из таких функций являются тэг–хелперы (tag helper), которые позволяют более органично соединять синтаксис html с кодом С#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, мы можем создавать приложения в среде Visual Studio Code, которая является кросс–платформенной и может работать как на Windows, так и на Mac OS X и Linux.

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты.

Если суммировать, то можно выделить следующие ключевые отличия ASP.NET Core от предыдущих версий ASP.NET:

* новый легковесный и модульный конвейер HTTP–запросов;
* возможность развертывать приложение как на IIS, так и в рамках своего
* собственного процесса;
* использование платформы .NET Core и ее функциональности;
* распространение пакетов платформы через NuGet;
* интегрированная поддержка для создания и использования пакетов NuGet;
* единый стек веб–разработки, сочетающий Web UI и Web API;
* конфигурация для упрощенного использования в облаке;
* встроенная поддержка для внедрения зависимостей;
* расширяемость;
* кроссплатформенность: возможность разработки и развертывания приложений ASP.NET на Windows, Mac и Linux;
* развитие как open source, открытость к изменениям.

Эти и другие особенности и возможности стали основой для новой модели программирования.

Хоть платформа ASP.NET Core и обеспечивает кроссплатформенность, данная технология не подходит для реализации проекта в силу требований задания. Так как программа должна быть установленна на компьютере и собирать сведения о нем. Поэтому создавать браузерное решение поставленной задачи не целесообразно.

В результате, язык C# и принципы объектно – ориентированного программирования подходят для реализации программы, а технология WPF не только проста в использовании, но и обеспечит кроссплатформенность, что позволит запускать программу на практически любом компьютере.

Для хранения данных можно использовать SQL Express. Microsoft SQL Server Express - это версия системы управления реляционными базами данных Microsoft [SQL](https://ru.bmstu.wiki/SQL_(Structured_Query_Language)) Server, которая является бесплатной для загрузок, распространения и использования. Он содержит базу данных, специально предназначенную для встраиваемых и маломасштабных приложений.

Продукт ведет свои корни к продукту Microsoft Database Engine (MSDE), поставляемому с SQL Server 2000. Внешний доступ предоставляется через слой протоколов SQL Server. Все операции в SQL Server могут быть выполнены путем обращения через определенный формат, называемый Tabular Data Stream (TDS). TDS протокол уровня приложения, используемый для передачи данных между сервером и клиентом. TDS может быть обернут другими физическими и транспортными протоколами, включая TCP/IP. Следовательно, доступ к SQL Server доступен и вне него. Архитектура можно увидеть на рисунке 1.

Кроме того, есть нативная поддержка Common Language Runtime (CLR), что позволяет эффективно использовать инструменты разработки Microsoft.

Так как разрабатывается интернет портал, следовало бы использовать паттерн MVC. MVC — это паттерн проектирования веб-приложений, который включает в себя несколько более мелких шаблонов. При использовании MVC на три отдельных компонента разделены модель данных приложения, пользовательский интерфейс и логика взаимодействия пользователя с системой, благодаря чему модификация одного из этих компонентов оказывает минимальное воздействие на остальные или не оказывает его вовсе.

Основная цель применения MVC состоит в разделении данных и бизнес-логики от визуализации. За счет такого разделения повышается возможность повторного использования программного кода: например, добавить представление данных какого-либо существующего маршрута не только в виде HTML, но и в форматах JSON, XML, PDF, XLSX становится очень просто и не требует исменений слоя бизнес-логики исходного маршрута. Также упрощается и сопровождение программного кода: внесение изменений во внешний вид, например, не отражаются на бизнес-логике, а изменения бизнес-логики не затрагивают визуализацию.

Модель / Model — предоставляет собой объектную модель некой предметной области, включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные и/или изменяя своё состояние. При этом модель не содержит в себе информации о способах визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую;

Представление / View — отвечает за отображение информации (визуализацию). Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком; на уровне API можно экспортировать данные как в JSON, так в XML или XSLX;

Контроллер / Controller — обеспечивает связь между пользователем и системой, использует модель и представление для реализации необходимой реакции на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация — проверяются права пользователя на выполнение действий или получение информации.

# **АЛГОРИТМИЧЕСКИЙ АНАЛИЗ**

## **База данных**

В качестве базы данных будем использовать реляционную базу данных. Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных.

В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей.

К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.

Таблица Authors, согласно таблице 2.1, хранит в своих полях данные связанные с сущностью класса Author.cs, который описывает аккаунт пользователя.

Таблица 2.1 – **Поля таблицы Authors**

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Id | Int | Уникальное поле, для всей таблицы |
| Name | nvarchar(max) | Имя пользователя |
| Email | nvarchar(max) | Адрес почты пользователя |
| Password | nvarchar(max) | Пароль пользователя |
| Description | nvarchar(max) | Описание аккаунта |
| Phone | nvarchar(max) | Номер пользователя |
| Role | Int | Роль аккаунта пользователя |

Таблица Topics, согласно таблице 2.2, хранит в своих полях данные связанные с сущностью класса Topic.cs, который описывает темы статьи.

Таблица 2.2 – **Поля таблицы Topics**

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Id | Int | Уникальное поле, для всей таблицы |
| Name | nvarchar(max) | Название темы статьи |

Таблица ArticleComments, согласно таблице 2.3, хранит в своих полях данные связанные с сущностью класса ArticleComment.cs, который описывает комментарий к статье.

Таблица 2.3 – **Поля таблицы ArticleComments**

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Id | Int | Уникальное поле, для всей таблицы |
| Text | nvarchar(max) | Текст комментария |
| DateTime | Datetime2(7) | Дата и время написания комментария |
| AuthorId | Int | Ссылка на написавшего автора |
| ArticleId | Int | Ссылка на связаную запись статьи |

Таблица Articles, согласно таблице 2.4, хранит в своих полях данные связанные с суностью класса Article.cs, который описывает сущность статьи.

Таблица 2.4 – **Поля таблицы Articles**

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Id | int | Уникальное поле, для всей таблицы |
| Title | nvarchar(max) | Заголовок статьи |
| Text | nvarchar(max) | Текст статьи |
| ImageBytes | varbinary(max) | Картинка статьи хранимая в байтах |
| NumberOfViews | int | Количество просмотров статьи |
| IsApproved | bit | Корректность статьи |
| Date | datetime2(7) | Дата написания статьи |

Продолжение таблицы 2.4

|  |  |  |
| --- | --- | --- |
| AuthorId | int | Ссылка на связаную сущность автора, который написал статью |
| TopicId | int | Ссылка на связанную сущность заголовка статьи |

На рисунке 2.1 приведена схема того, как база данных будет выглядеть после создания.

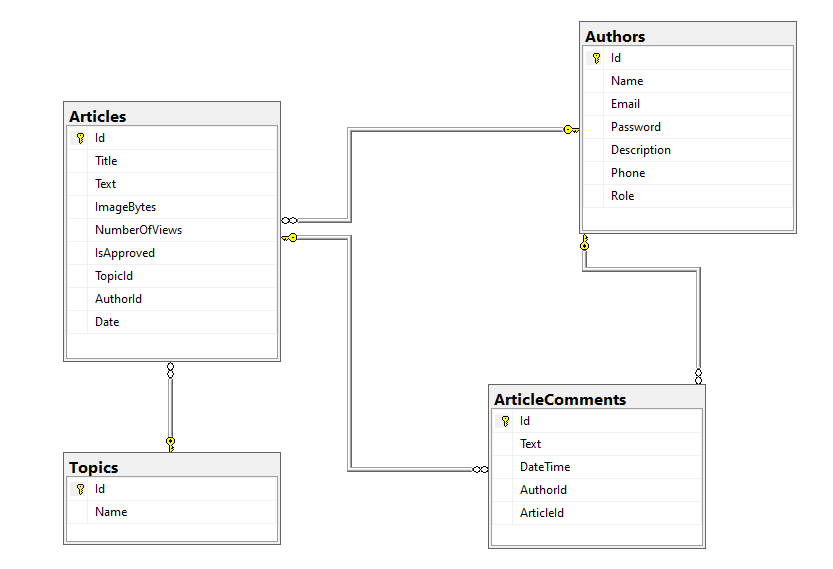


Рисунок 2.1 – Схема базы данных

## **Анализ задания на проектирование**

На рисунке 2.2 графически показаны возможности пользователе, с уровнем доступа администратора, в созданном приложении по просмотру и добавлению статей, среди них:

* повышение уровня доступа пользователей;
* опровержение статей;
* подтверждение корректности татьи;
* просмотр статей;
* написание комментариев к статьям;
* добавление статей;
* удаление статей;
* уменьшение уровня доступа пользователей.

Данные функции доступны пользователям только с уровнем доступа «Администратор»

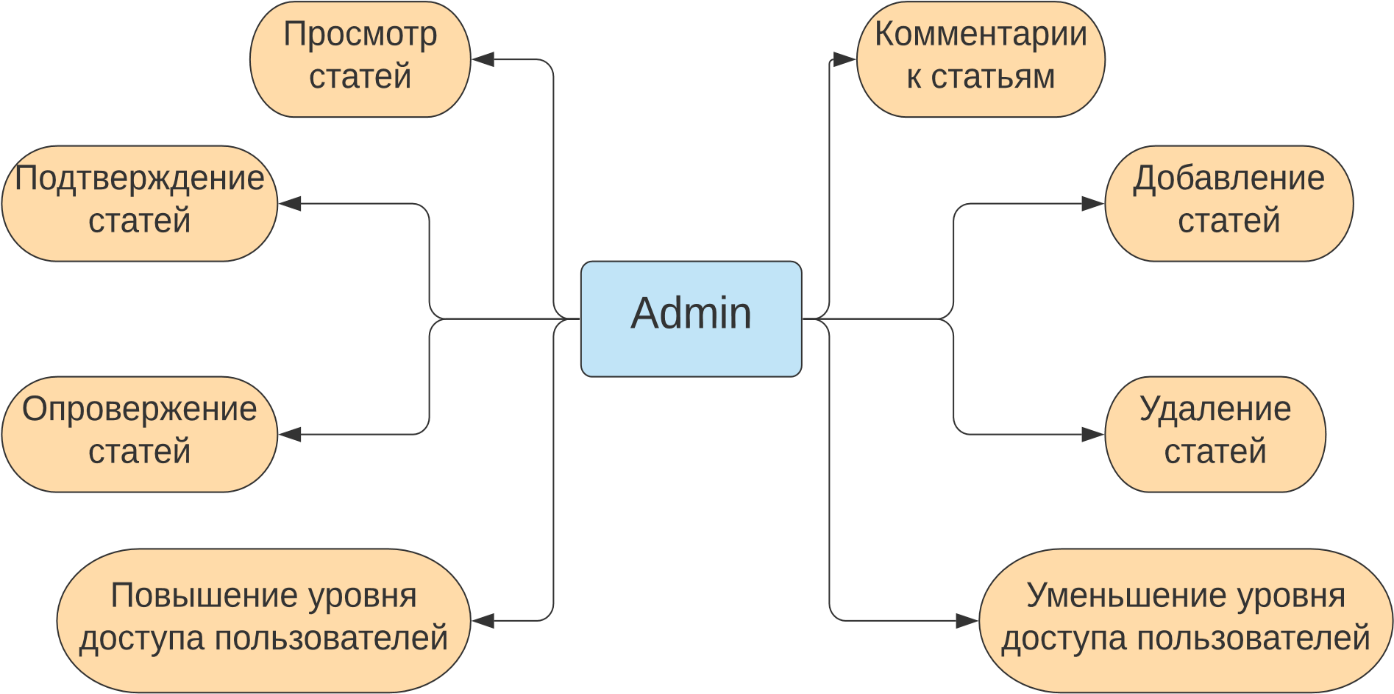


Рисунок 2.2 – Возможности пользователя с уровнем доступа «Админ»

Так как администратор так же является пользователем, ему доступны функции связаные с добавлением статей. Поэтому, кроме администирования, он также может создавать контент для сайта.

На рисунке 2.3 графически показаны возможности пользователя, с уровнем доступа менеджер, в созданном приложении по просмотру и добавлению статей, среди них:

* опровержение статей;
* подтверждение корректности татьи;
* просмотр статей;
* написание комментариев к статьям;
* добавление статей;
* удаление статей.

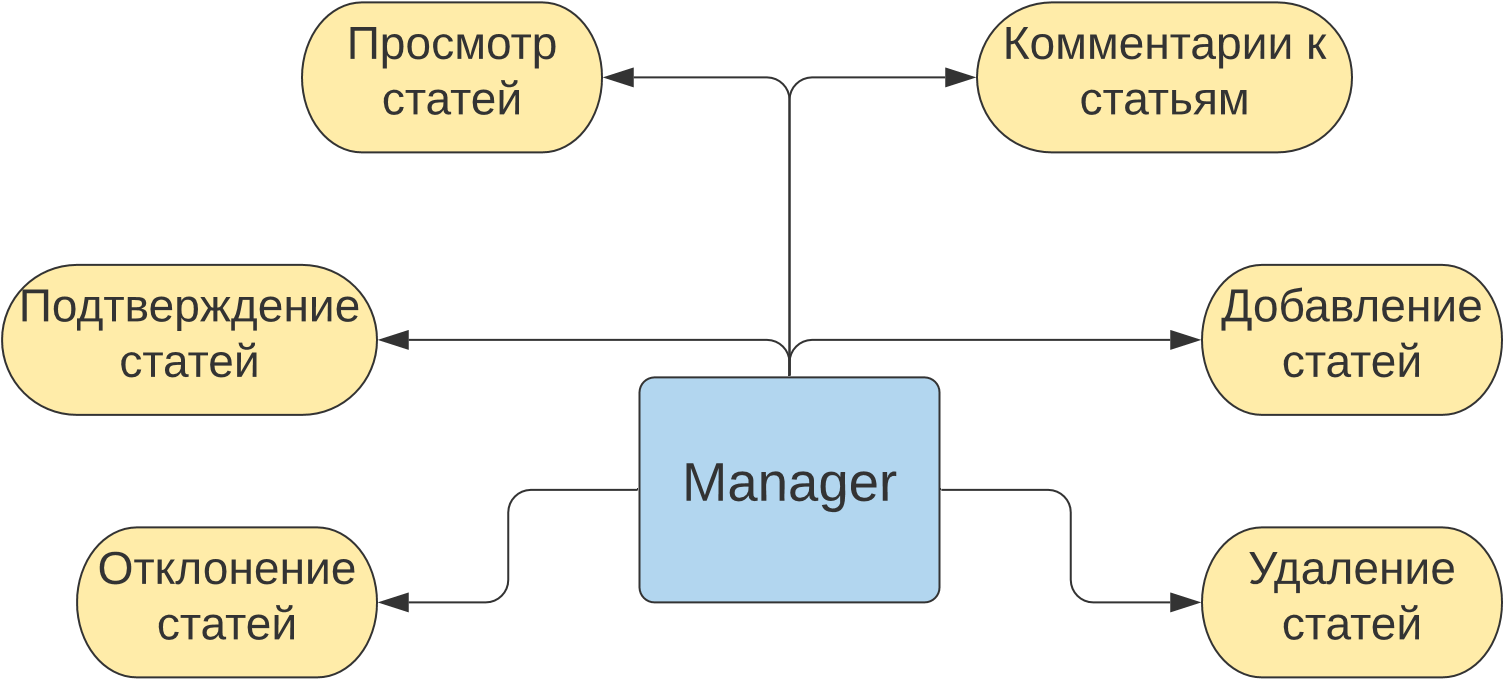


Рисунок 2.3 – Возможности пользователя с уровнем доступа «Менеджер»

Так как менеджер так же является пользователем, ему доступны функции связаные с добавлением статей. Поэтому он также может создавать контент для сайта, но ему не достуна часть возможностей пользователя с уровнем доступа «Администратор».

На рисунке 2.4 графически показаны возможности пользователе, с уровнем доступа автора, в созданном приложении по просмотру и добавлению статей, среди них:

* просмотр статей;
* написание комментариев к статьям;
* добавление статей;
* удаление статей.

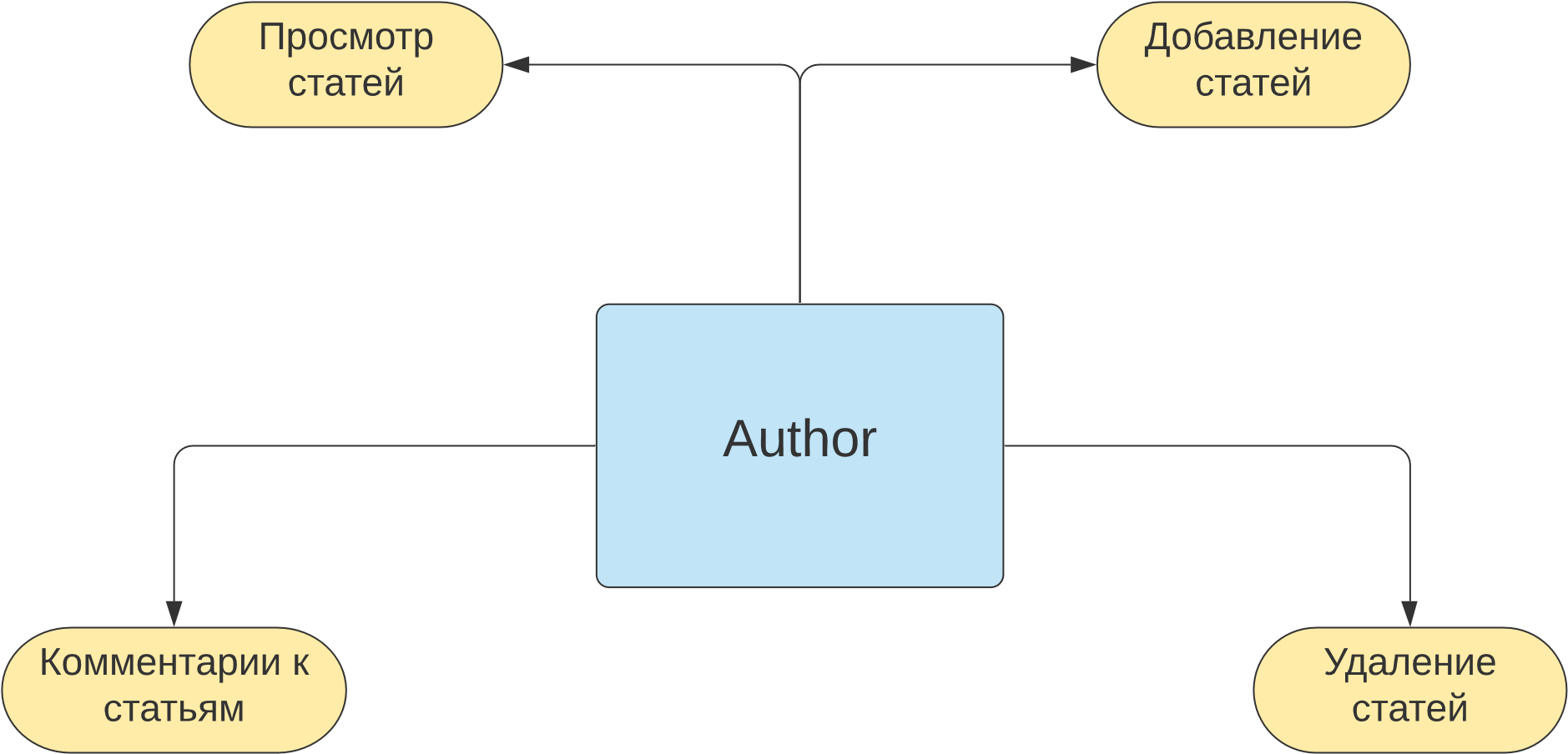


Рисунок 2.4 – Возможности пользователя с уровнем доступа «Автор»

Уровень доступа «Автор» – самый низкий уровень доступа в системе. Данная группа пользователей имеет возможность создавать свои статьи, управлять ими, просматривать и комментировать статьи других пользователей.

Рассмотрим функции и возможности, которые должны присутствовать в проекте.

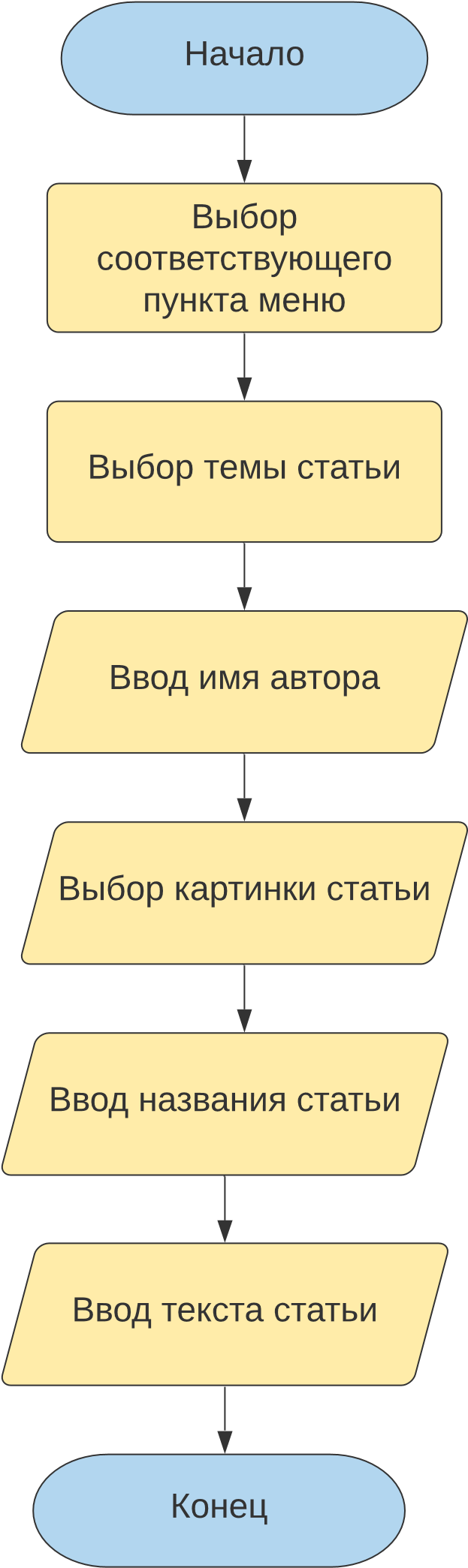


Рисунок 2.5 – Добавление автором новой статьи

Добавление статей доступно всем авторизованным пользователям, то есть автору, менежеру и администратору. Согласно рисунку 2.5 для добавления нужно выбрать соответствуюий пункт меню, из предложенного списка выбрать тему статьи, ввести имя автора, если пользователь хочет остаться анонимным, либо оставить свое имя, выбрать картинку для статьи, которая будет отображаться в процессе создания статьи и просмотра этой статьи другими пользователями. Далее следует ввести название статьи и ввести сам текст статьи, который будет просматриваться пользователями.

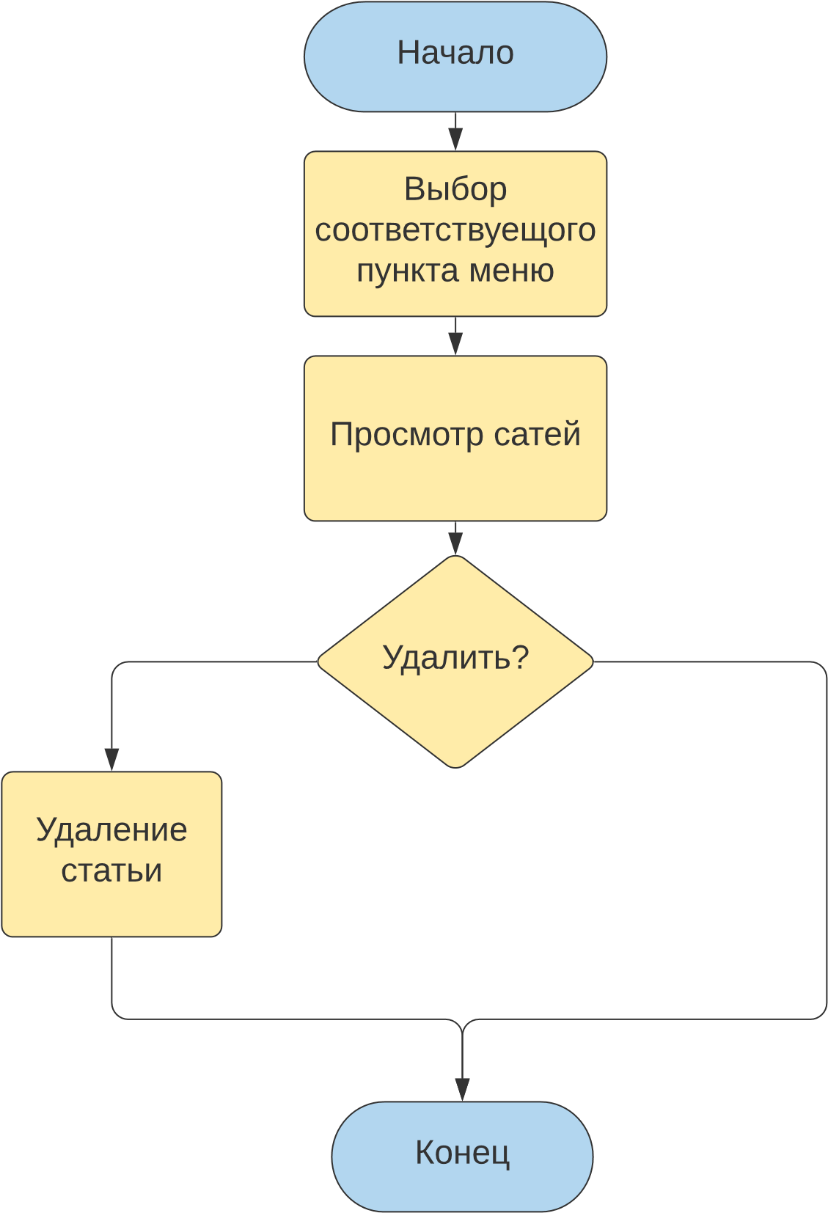


Рисунок 2.6 – Просмотр написаных автором статей

Каждый пользователь должен обладать возможностью просмотра статей, которые он написал, и возможностью удалить свою статью. Согласно рисунку 2.6 для просмотра и удаления статей нужно выбрать соответствующий пункт меню, после чего выведется список всех написаных статей. Все выведеные статьи можно просмотреть подробнее, либо удалить, путем нажатия на соответствующую кнопку.

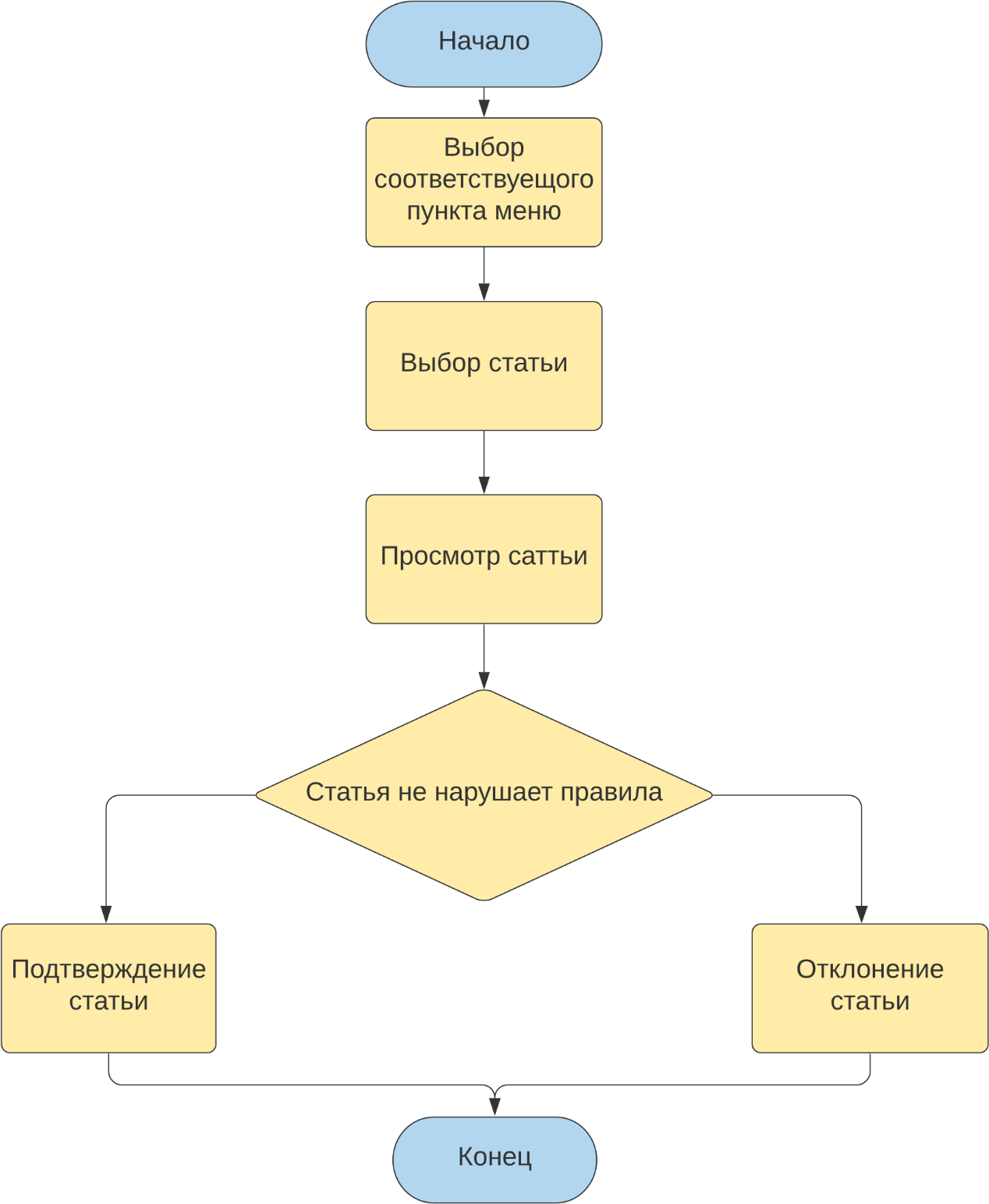


Рисунок 2.7 – Подтверждение статьи

После написания автором статьи, статья должна быть одобрена администрацией. Для одобрения статьи, пользователь с уровнем доступа менеджер или администратор, согласно рисунку 2.7, должен выбрать соответствующий пункт меню. После нажатия, выводится списк статей, которые нужно одобрить. Для этого статью можно просмотреть, в полном виде, в таком какой ее будут видеть пользователи. После просмотра статью можно одобрить или отклонить. В случае одобрения, статья будет показываться всем пользователям, в случае отклонения, статья будет доступна только автору.

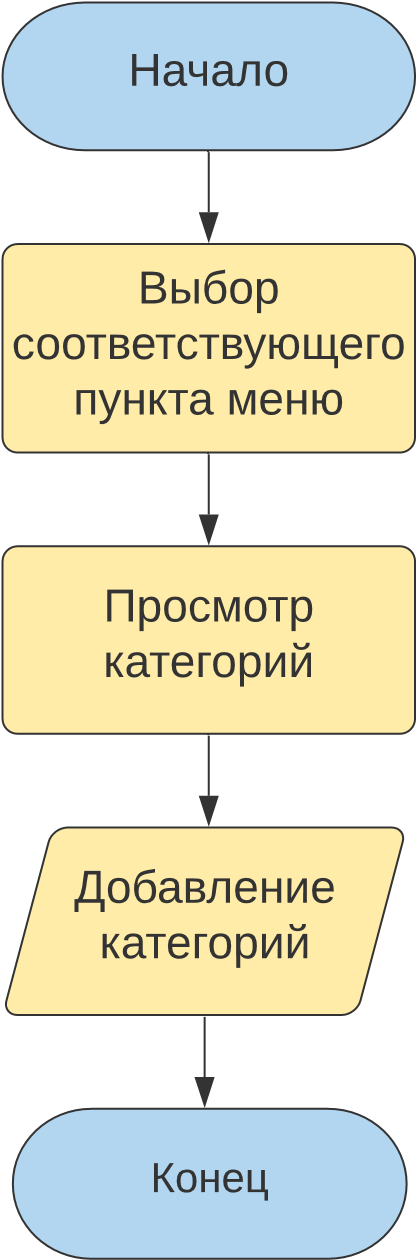


Рисунок 2.8 – Добавление категории для статей

Для добавления категории для статей пользователь, с уровнем доступа менеджера или админа, согласно рисунку 2.8, должен выбрать соответствуюий пункт меню. После чего, выведется список всех категорий, которые на данный момент имеются в системе. По каждой категории имеется информация о статьях. В поле для ввода, вводится название новой категории для добавления и нажимается соответствующая кнопка для добавления.

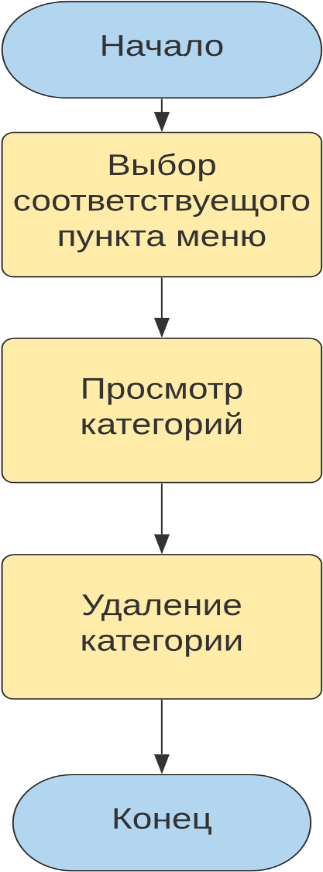


Рисунок 2.9 – Удаление категории для статей

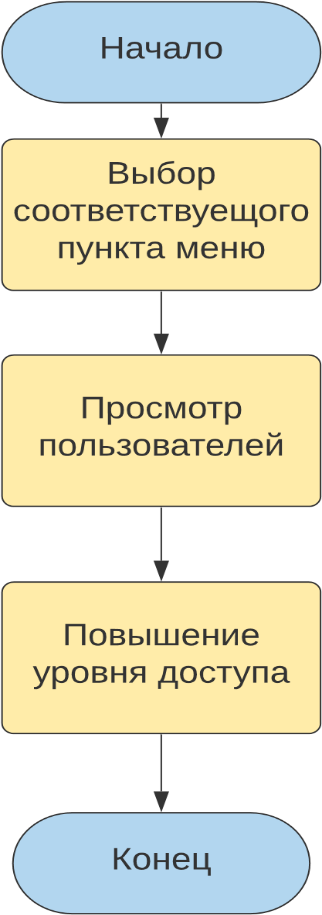
Для удаления, согласно рисунку 2.9, пользователь должен выбрать категорию и нажать на соответствующую кнопку.

Рисунок 2.10 – Повышение уровня доступа пользователей

Для повышения уровня доступа пользователей, пользователь с уровнем доступа администратора, согласно рисунку 2.10, должен выбрать соответствующий пункт меню. После чего, выведется список пользователей с некоторыми данными о них. После выбора определенного пользователя, с помощью соответствующих кнопок, приоритет пользователя повышается.

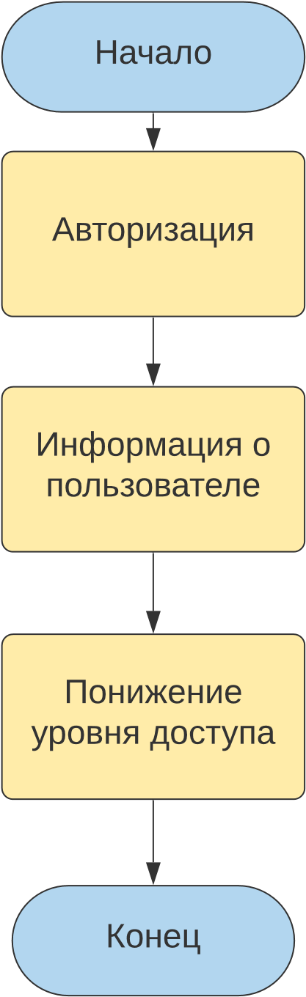


Рисунок 2.11 – Понижение уровня доступа пользователей

Согласно рисунку 2.11, для понижения уровня доступа, следует выбрать соответствуюий пункт меню, выбрать пользователя и нажать соответствующую кнопку.



Рисунок 2.12 – Редактирование данных пользователя

Для редактирования данных пользователя, согласно рисунку 2.12, пользователю следует авторизоваться, после этого, перейти в просмотр информации о себе. Далее выбрать пункт редактирования, где пользователю предоставится возможность изменения информации о себе.

# **РАЗРАБОТКА ПРОГРАММНОГО КОДА**

## **Структурная схема программы**

Программой является сервер, который реагирует на действия пользователя и возвращает результат, в ввиде HTML страниц в ответ на запрос пользователя, согласно рисунку 3.1.



Рисунок 3.1 – Структурная схема программы

## **Создание приложения**

Весь проект разделен на 3 слоя, а именно, слой доступа к данным, слой бизнес логики и слой пользовательского интерфейса. Данное разделение соответствует N-Tier architecture. Схема данной архитектуры представлена на рисунке 3.2.

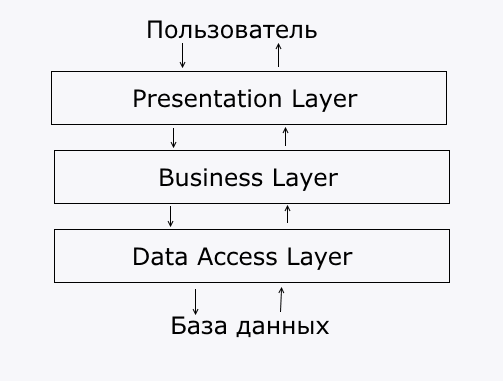


Рисунок 3.2 – Схема N-Tier architecture

Классы, уровня доступа к данным, представляют собой описание доменных сущностей, согласно таблицам 3.1, 3.2, 3.3, 3.4. Подробнее классы описаны в приложении А. Далее опишим их.

Таблица 3.1 – **Класс Author**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип данных** | **Описание** |
| Свойства | Id | Int | Уникальный идентификатор |
| Name | String | Имя автора |
| Email | String | Почта автора |
| Password | String | Пароль |
| Description | String | Описание аккаунта |
| Phone | String | Номер телефона |
| Role | Struct | Роль пользователя |

Таблица 3.2 – **Класс Topic**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип данных** | **Описание** |
| Свойства | Id | Int | Уникальный идентификатор |
| Name | String | Название |

Таблица 3.3 – **Класс ArticleComment**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип данных** | **Описание** |
| Свойства | Id | Int | Уникальный идентификатор |
| Text | String | Текст комментария |
| DateTime | DateTime | Дата и время |
| ArticleId | Int | Идентификатор статьи |
| Article | Article | Для получения связанной статьи |
| AuthorId | Int | Идентификатор автора |
| Author | Author | Для получения связанного автора |

Таблица 3.4 – **Класс Article**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип данных** | **Описание** |
| Свойства | Id | Int | Уникальный идентификатор |
| Title | String | Название статьи |
| Text | String | Текст статьи |
| ImageBytes | Byte[] | Картинка статьи |
| NumberOfViews | Int | Количество просмотров |
| IsApproved | Bool | Корректность статьи |
| Date | DateTime | Дата написания |
| Topic | Topic | Для получения связанной темы |
| Author | Author | Для получения связанного автора |

Также уровень доступа к данным содержит в себе классы-репозитории, которые используются для доступа к базе данных, согласно таблице 3.5. Подробнее классы описаны в приложении А. Далее опишем один из них, так как они одинаковы для всех доменных сущностей.

Таблица 3.5 – **Класс ArticleRepository**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | Add | Int | Добавляет статью в базу |
| GetAll() | List<Article> | Возвращает все статьи из базы |
| Remove | Void | Удаляет статью из базы |
| Update | Void | Обновляет статью в базе |

Классы, уровня бизнес логики содержат в себе классы-сервисы, которые позволяют слою презентации получить данные из базы данных, через классы-репозитории,согласно таблице 3.6. Подробнее классы описаны в приложении А. Далее опишем один из них, так как они одинаковы для всех доменных сущностей.

Таблица 3.6 – **Класс ArticleService**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | Add | Int | Добавляет статью в базу |
| GetAll() | List<Article> | Возвращает все статьи из базы |
| Remove | Void | Удаляет статью из базы |
| Update | Void | Обновляет статью в базе. |

Уровень представления содержит в себе классы-контроллеры, которые реагируют на действия пользователя, согласно таблицам 3.7, 3.8, 3.9, 3.10. Подробнее классы описаны в приложении А. Далее опишим их.

Таблица 3.7 – **Класс HomeController**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | Index | IActionResult | Возвращает пользователю html страничку со всеми статьями |

Таблица 3.8 – **Класс AccountController**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | Register | IActionResult | Возвращает html страничку для ввода данных для регистрации. |
| Register | Принимает данные от пользователя и регистрирует аккаунт |
| Login | Возвращает html страничку для авторизации |

Продолжение таблицы 3.8

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logout |  | Позволяет пользователю выйти из аккаунта |
| Login | Принимает данные от пользователя и авторизирует его |
| ShowProfile | Возвращает html страничку с информацией об аккаунте и пользователе |
| EditProfile | Возвращает html страничку для редактирования информации об аккаунте |
| EditProfile | Принимает отредактированные данные и обновляет базу данных |

Таблица 3.9 – **Класс ArticleController**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | Add | IActionResult | Возвращает html страничку для добавления статьи |
| AddArticle | Принимает данные и добавляет новую статью |
| AddComment | Принимает данные и добавляет комментарий к статье |

Продолжение таблицы 3.9

|  |  |  |  |
| --- | --- | --- | --- |
|  | RemoveArticle |  | Удаляет выбранную статью |
| ShowArticle |  | Возвращает html страничку для просмотра выбранной статьи |
| ArticlesByAuthor |  | Возвращает html страничку для просмотра статей автора |

Таблица 3.10 – **Класс SiteManagementController**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Возвращаемый тип** | **Описание** |
| Методы | ManagementBody | IActionResult | Возвращает html страницу для управления сайтом |
| GetArticlePartialView | Возвращает html страницу для проверки статей |
| ApproveArticle | Подтверждение корректности статьи |
| DisapproveArticle | Опровержение корректности статьи |
| GetTopicPartialView | Возвращает html страницу для управления темами |
| AddTopic | Добавляет тему стетей |
| RemoveTopic | Удаляет тему статей |
| GetAuthorPartialView | Возвращает html страницу для управления пользователями |

Продолжение таблицы 3.10

|  |  |  |  |
| --- | --- | --- | --- |
|  | IncreaceRoleToAuthor |  | Увеличивает уровень доступа пользователя |
| DecreaceRoleToAuthor | Уменьшает уровень доступа пользователя |

## **Интерфейс пользователя**

Пользовательский интерфейс – это совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги.

Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода / вывода, которые физически обеспечивают связь пользователя и компьютера.

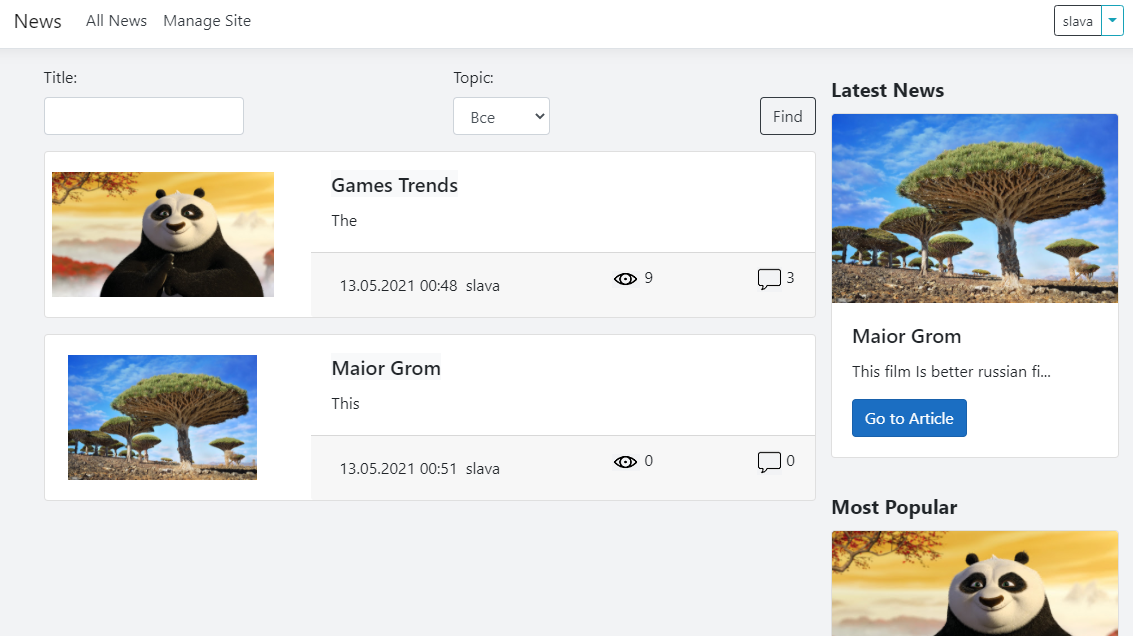


Рисунок 3.3 – Главная страница сайта

Согласно рисунку 3.3, после перехода по адрусу сайта, пользователь видит главную страницу сайта. Здесть можно просмотреть все статьи, просмотреть каждую статью в отдельности, прочесть комментарии к статьям, или воспользоваться предоженными статьями.

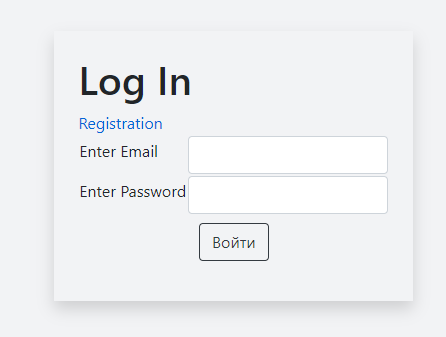


Рисунок 3.4 – Форма авторизации пользователей

Согласно рисунку 3.4, зарегистрированный пользователь с помощью логина и пароля, может авторизироваться на сайте, что даст ему ряд новых возможностей, либо нажать на кнопку регистрации и ввести свои данные для создания нового аккаунта, с помощью которого он сможет авторизироваться позже.

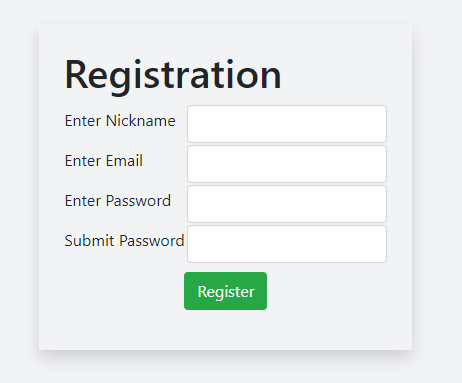


Рисунок 3.5 – Форма регистрации пользователей

Согласно рисунку 3.5, для авторизации пользователь должен ввести псевдоним, почту, пароль и подтвердить свой пароль. После этого, зарегистрированный пользователь может авторизироваться на сайте с помощью своей почты и парля.

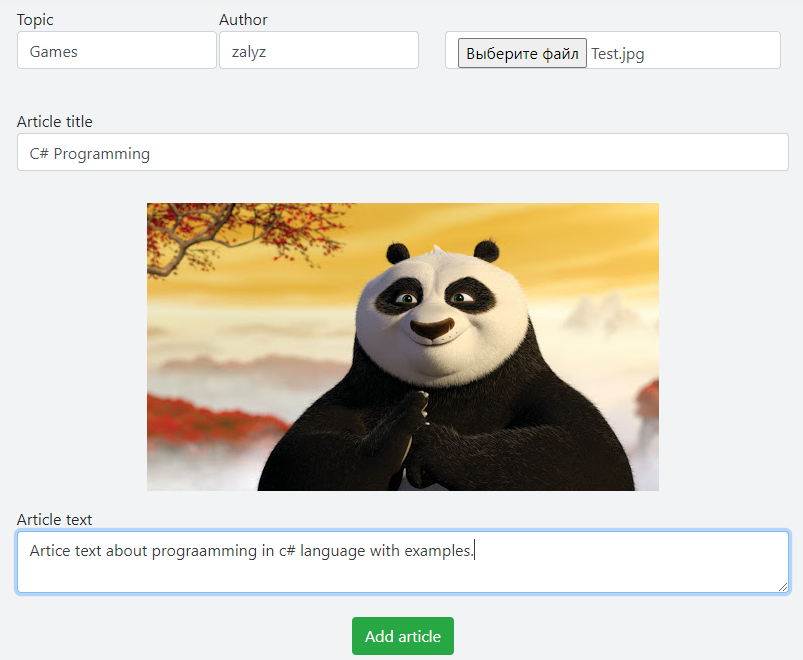


Рисунок 3.6 – Форма создания автором новой статьи

Согласно рисунку 3.6, для создания новой статьи, нужно ввести имя автора, выбрать тему, ввести название статьи и сам текст статьи. Обязательно нужно выбрать картинку, которая будет представлять статью. После добавления, статья отправляется на одобрение модерацией сайта.

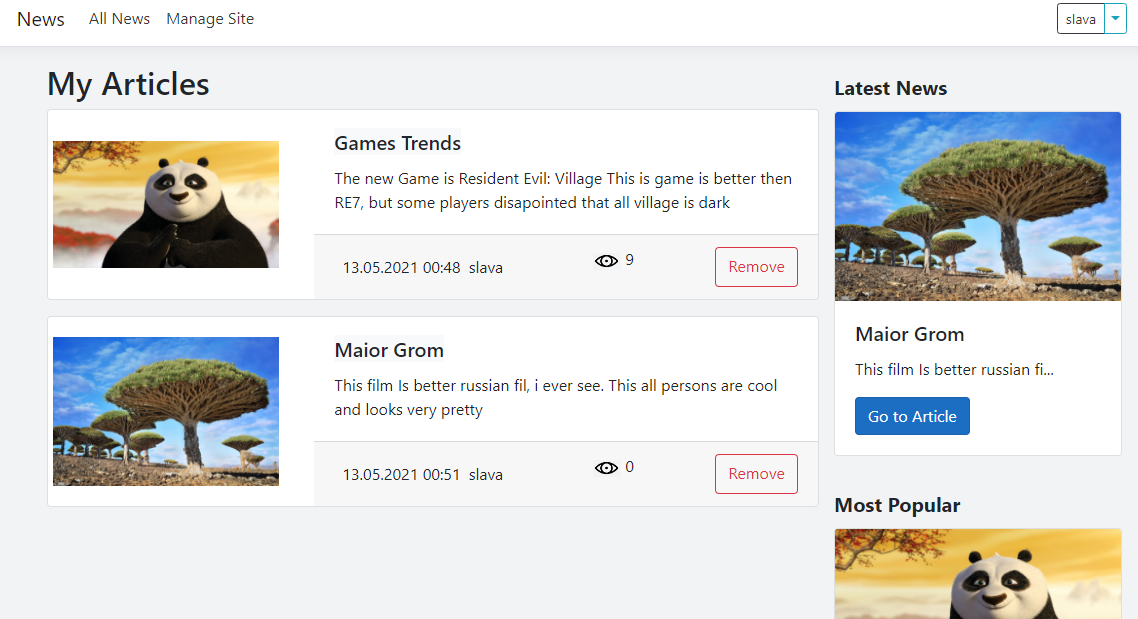


Рисунок 3.7 – Просмотр пользователем, написанных им статей

Согласно рисунку 3.7, авторизированный пользователь может просмотреть сисок своих статей и, в случае необходимости, удалить ее.

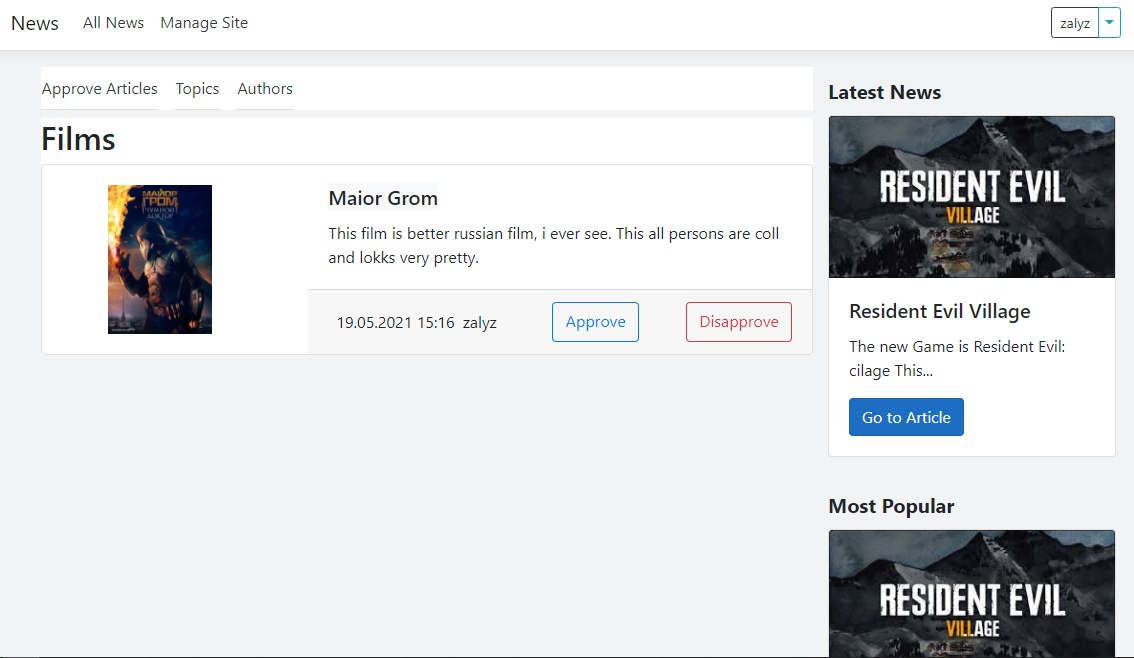


Рисунок 3.8 – Окно проверки статей модерацией сайта

Согласно рисунку 3.8, в данном окне можно подтвердить корректность статьи, либо опровергнуть, что позволяет контролировать содержимое и смысл статей. Для проверки, нажав на название статьи, можно перейти на страницу статьи и прочитать ее содержимое.

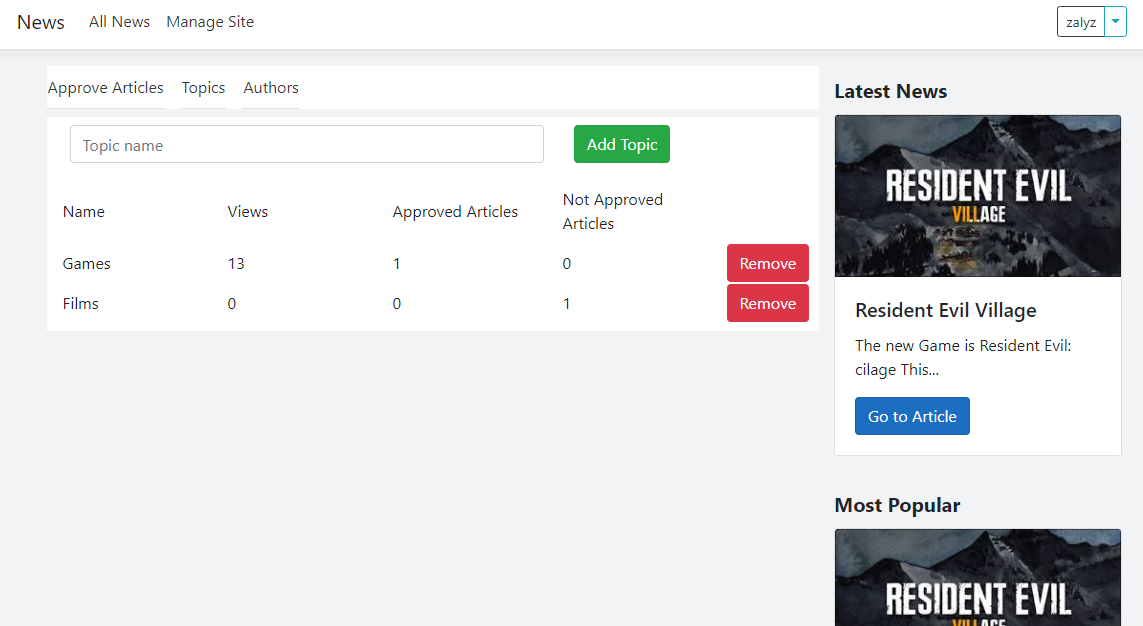


Рисунок 3.9 – Окно для управления темами статей

Согласно рисунку 3.9, в данном окне можно просмотреть некоторую статистику по темам, добавить либо удалить тему статей. Добавление осуществляется с помощью ввода названия темы в соответствующее поле и нажатия на соответствующую кнопку добавления.

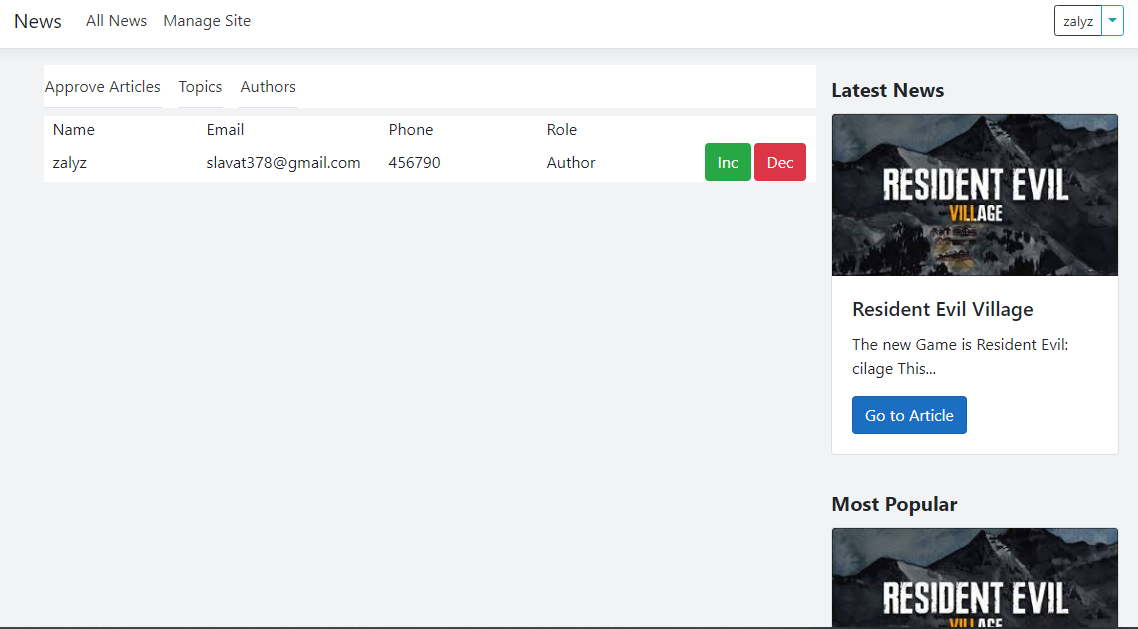


Рисунок 3.10 – Окно управления уровнем доступа пользователей

Согласно рисунку 3.10, в данном окне можно просмотреть некоторую информацию каждого пользователя и, при желании повысить или понизить уровень доступа каждого их них.

# **ЗАКЛЮЧЕНИЕ**

В ходе данного курсового проекта было смоделировано веб–приложение, которое позволяет неавторизированным пользователям просматривать написанные статьи и комментарии, авторизированным пользователям просматривать статьи как и неавторизированным, и создавать свои статьи, редактировать информацию о себе.

Авторизированные пользователи с уровнем доступа менеджер или админ, имеют право одобрять статьи или отклонять их, добавлять новые темы для статей и просматривать информаию о статьях по каждой теме.

Все статьи имеют свои показатели, такие как количество просмотров и количество комментариев, благодаря чему пользователь сам ожет выбирать статью по популярности.

На главной странице для всех пользователей доступна фильтрация статей, для быстрого и более удобного поиска интересуюих пользователястатей. В критерии поиска входит название статьи и название темы статьи.

Всем авторизированные пользователи погут оставлять комментарии к статьям и просматривать все комментарии под этой статьей.

Были реализованы поставленные задачи.

В планах развития проекта – создание распределенной базы данных, для повышения надежности, адаптация данного сайта под разрешение мобильных телефонов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фриман, Адам Pro ASP.NET Core MVC 2 / Адам Фриман. – Лондон: Apress. 2017. – 1013 с
2. Натан, Адам WPF 4 Подробное руководство / Адам Натан. – Санкт–Петербург: Символ–Плюс. 2019. – 878 с
3. Рихтер, Джефри CLR via C#: Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Джефри Рихтер. – Санкт–Петербург: Питер, 2013. – 893 с
4. Обзор Windows Forms: Официальная документация. – Электронные данные. – Режим доступа: https://docs.microsoft.com/ru–. – Дата доступа: 05.04.2021.
5. Общие сведения о WPF: Официальная документация. – Электронные данные. – Режим доступа: [https://docs.microsoft.com/ru–](https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/introduction-to-wpf) .– Дата доступа: 04.05.2021.

ПРИЛОЖЕНИЕ A

(обязательное)

## Проект EntityModels

Файл Article.cs

using System;

using System.ComponentModel.DataAnnotations;

using EntityModels.Users;

namespace EntityModels.DamainEntities

{

public class Article

{

[Key]

public int Id { get; set; }

public string Title { get; set; }

public string Text { get; set; }

public byte[] ImageBytes { get; set; }

public int NumberOfViews { get; set; }

public bool IsApproved { get; set; } = false;

public DateTime Date { get; set; }

public Topic Topic { get; set; }

public Author Author { get; set; }

}

}

Файл ArticleComment.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using EntityModels.Users;

namespace EntityModels.DamainEntities

{

public class ArticleComment

{

[Key]

public int Id { get; set; }

public string Text { get; set; }

public DateTime DateTime { get; set; }

[ForeignKey("Article")]

public int ArticleId { get; set; }

public Article Article { get; set; }

public int AuthorId { get; set; }

[ForeignKey("AuthorId")]

public Author Author { get; set; }

}

}

Файл Topic.cs

namespace EntityModels.DamainEntities

{

public class Topic

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Файл NewsContext.cs

using Microsoft.EntityFrameworkCore;

using EntityModels.DamainEntities;

using EntityModels.Users;

namespace EntityModels.Context

{

public class NewsContext : DbContext

{

public DbSet<Article> Articles { get; set; }

public DbSet<Topic> Topics { get; set; }

public DbSet<ArticleComment> ArticleComments { get; set; }

public DbSet<Author> Authors { get; set; }

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

{

}

}

}

Файл Author.cs

using System.ComponentModel.DataAnnotations;

namespace EntityModels.Users

{

public class Author

{

[Key]

public int Id { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Password { get; set; }

public string Description { get; set; }

public string Phone { get; set; }

public Role Role { get; set; }

}

}

Файл Role.cs

namespace EntityModels.Users

{

public enum Role

{

Author = 1,

Manager = 2,

Admin = 3,

}

}

Файл IRepository.cs

using System.Collections.Generic;

namespace EntityModels.Interfaces

{

public interface IRepository<T>

where T : class

{

public int Add(T entity);

public List<T> GetAll();

public void Update(T entity);

public void Remove(int id);

}

}

Файл ArticleCommentRepository.cs

using System.Linq;

using System.Collections.Generic;

using EntityModels.Context;

using EntityModels.Interfaces;

using EntityModels.DamainEntities;

using Microsoft.EntityFrameworkCore;

namespace EntityModels.Repositories

{

internal class ArticleCommentRepository : IRepository<ArticleComment>

{

private readonly NewsContext \_newsContext;

public ArticleCommentRepository(NewsContext newsContext)

{

\_newsContext = newsContext;

}

public int Add(ArticleComment entity)

{

var state = \_newsContext.ArticleComments.Add(entity);

\_newsContext.SaveChanges();

return state.Entity.Id;

}

public List<ArticleComment> GetAll()

{

return \_newsContext.ArticleComments.Include(e => e.Author).ToList();

}

public void Remove(int id)

{

var entity = \_newsContext.ArticleComments.First(e => e.Id == id);

\_newsContext.ArticleComments.Remove(entity);

\_newsContext.SaveChanges();

}

public void Update(ArticleComment entity)

{

\_newsContext.ArticleComments.Update(entity);

\_newsContext.SaveChanges();

}

}

}

Файл ArticleRepository.cs

using System.Linq;

using System.Collections.Generic;

using EntityModels.Context;

using EntityModels.Interfaces;

using EntityModels.DamainEntities;

using Microsoft.EntityFrameworkCore;

namespace EntityModels.Repositories

{

internal class ArticleRepository : IRepository<Article>

{

private readonly NewsContext \_newsContext;

public ArticleRepository(NewsContext newsContext)

{

\_newsContext = newsContext;

}

public int Add(Article entity)

{

var state = \_newsContext.Articles.Add(entity);

\_newsContext.SaveChanges();

return state.Entity.Id;

}

public List<Article> GetAll()

{

return \_newsContext.Articles.Include(e => e.Author).Include(e => e.Topic).ToList();

}

public void Remove(int id)

{

var entity = \_newsContext.Articles.First(e => e.Id == id);

\_newsContext.Articles.Remove(entity);

\_newsContext.SaveChanges();

}

public void Update(Article entity)

{

var state = \_newsContext.Articles.Update(entity);

\_newsContext.SaveChanges();

}

}

}

Файл AuthorRepository.cs

using System.Linq;

using System.Collections.Generic;

using EntityModels.Context;

using EntityModels.Interfaces;

using EntityModels.Users;

namespace EntityModels.Repositories

{

internal class AuthorRepository : IRepository<Author>

{

private readonly NewsContext \_newsContext;

public AuthorRepository(NewsContext newsContext)

{

\_newsContext = newsContext;

}

public int Add(Author entity)

{

var state = \_newsContext.Authors.Add(entity);

\_newsContext.SaveChanges();

return state.Entity.Id;

}

public List<Author> GetAll()

{

return \_newsContext.Authors.ToList();

}

public void Remove(int id)

{

var entity = \_newsContext.Authors.First(e => e.Id == id);

\_newsContext.Authors.Remove(entity);

\_newsContext.SaveChanges();

}

public void Update(Author entity)

{

\_newsContext.Authors.Update(entity);

\_newsContext.SaveChanges();

}

}

}

Файл TopicRepository.cs

using System.Linq;

using System.Collections.Generic;

using EntityModels.Context;

using EntityModels.Interfaces;

using EntityModels.DamainEntities;

namespace EntityModels.Repositories

{

internal class TopicRepository : IRepository<Topic>

{

private readonly NewsContext \_newsContext;

public TopicRepository(NewsContext newsContext)

{

\_newsContext = newsContext;

}

public int Add(Topic entity)

{

var state = \_newsContext.Topics.Add(entity);

\_newsContext.SaveChanges();

return state.Entity.Id;

}

public List<Topic> GetAll()

{

return \_newsContext.Topics.ToList();

}

public void Remove(int id)

{

var entity = \_newsContext.Topics.First(e => e.Id == id);

\_newsContext.Topics.Remove(entity);

\_newsContext.SaveChanges();

}

public void Update(Topic entity)

{

\_newsContext.Topics.Update(entity);

\_newsContext.SaveChanges();

}

}

}

Файл AddDal.cs

using EntityModels.Context;

using EntityModels.DamainEntities;

using EntityModels.Interfaces;

using EntityModels.Repositories;

using EntityModels.Users;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using System;

namespace EntityModels

{

public static class AddDal

{

public static void AddDalClasses(this IServiceCollection services)

{

services.AddDbContext<NewsContext>(options =>

{

options.UseSqlServer(@"Data Source=.\sqlexpress;Initial Catalog=NewsDb;Integrated Security=True");

options.LogTo(Console.WriteLine);

});

services.AddTransient<IRepository<Article>, ArticleRepository>();

services.AddTransient<IRepository<Topic>, TopicRepository>();

services.AddTransient<IRepository<ArticleComment>, ArticleCommentRepository>();

services.AddTransient<IRepository<Author>, AuthorRepository>();

}

}

}

## Проект BLL

Файл IService.cs

using System.Collections.Generic;

namespace BLL.BusinessInterfaces

{

public interface IService<T>

where T : class

{

public int Add(T entity);

public List<T> GetAll();

public void Update(T entity);

public void Remove(int id);

}

}

Файл ArticleCommentService.cs

using System.Collections.Generic;

using BLL.BusinessInterfaces;

using EntityModels.DamainEntities;

using EntityModels.Interfaces;

namespace BLL.Services

{

internal class ArticleCommentService : IService<ArticleComment>

{

private readonly IRepository<ArticleComment> \_articleCommentRepository;

public ArticleCommentService(IRepository<ArticleComment> articleCommentRepository)

{

\_articleCommentRepository = articleCommentRepository;

}

public int Add(ArticleComment entity)

{

return \_articleCommentRepository.Add(entity);

}

public List<ArticleComment> GetAll()

{

return \_articleCommentRepository.GetAll();

}

public void Remove(int id)

{

\_articleCommentRepository.Remove(id);

}

public void Update(ArticleComment entity)

{

\_articleCommentRepository.Update(entity);

}

}

}

Файл ArticleService.cs

using System.Collections.Generic;

using BLL.BusinessInterfaces;

using EntityModels.DamainEntities;

using EntityModels.Interfaces;

namespace BLL.Services

{

public class ArticleService : IService<Article>

{

private readonly IRepository<Article> \_articelRepository;

public ArticleService(IRepository<Article> articelRepository)

{

\_articelRepository = articelRepository;

}

public int Add(Article entity)

{

return \_articelRepository.Add(entity);

}

public List<Article> GetAll()

{

return \_articelRepository.GetAll();

}

public void Remove(int id)

{

\_articelRepository.Remove(id);

}

public void Update(Article entity)

{

\_articelRepository.Update(entity);

}

}

}

Файл AuthorService.cs

using System.Collections.Generic;

using BLL.BusinessInterfaces;

using EntityModels.Interfaces;

using EntityModels.Users;

namespace BLL.Services

{

internal class AuthorService : IService<Author>

{

private readonly IRepository<Author> \_authorRepository;

public AuthorService(IRepository<Author> authorRepository)

{

\_authorRepository = authorRepository;

}

public int Add(Author entity)

{

return \_authorRepository.Add(entity);

}

public List<Author> GetAll()

{

return \_authorRepository.GetAll();

}

public void Remove(int id)

{

\_authorRepository.Remove(id);

}

public void Update(Author entity)

{

\_authorRepository.Update(entity);

}

}

}

Файл TopicService.cs

using System.Collections.Generic;

using BLL.BusinessInterfaces;

using EntityModels.DamainEntities;

using EntityModels.Interfaces;

namespace BLL.Services

{

internal class TopicService : IService<Topic>

{

private readonly IRepository<Topic> \_topicRepository;

public TopicService(IRepository<Topic> topicRepository)

{

\_topicRepository = topicRepository;

}

public int Add(Topic entity)

{

return \_topicRepository.Add(entity);

}

public List<Topic> GetAll()

{

return \_topicRepository.GetAll();

}

public void Remove(int id)

{

\_topicRepository.Remove(id);

}

public void Update(Topic entity)

{

\_topicRepository.Update(entity);

}

}

}

Файл AddBll.cs

using BLL.BusinessInterfaces;

using BLL.Services;

using EntityModels;

using EntityModels.DamainEntities;

using EntityModels.Users;

using Microsoft.Extensions.DependencyInjection;

namespace BLL

{

public static class AddBll

{

public static void AddBllClassess(this IServiceCollection service)

{

service.AddDalClasses();

service.AddTransient<IService<Article>, ArticleService>();

service.AddTransient<IService<ArticleComment>, ArticleCommentService>();

service.AddTransient<IService<Topic>, TopicService>();

service.AddTransient<IService<Author>, AuthorService>();

}

}

}

## Проект CourseWork

Файл AccountController.cs

using System.Collections.Generic;

using System.Security.Claims;

using System.Threading.Tasks;

using CourseWork.Models.AccountModels;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Mvc;

using BLL.BusinessInterfaces;

using EntityModels.Users;

using System.Linq;

namespace CourseWork.Controllers

{

public class AccountController : Controller

{

private readonly IService<Author> \_authorService;

public AccountController(IService<Author> authorService)

{

\_authorService = authorService;

}

[HttpGet]

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(AccountRegistrationValidation accountValidation)

{

if (ModelState.IsValid)

{

var author = new Author

{

Name = accountValidation.Nickname,

Email = accountValidation.Email,

Password = accountValidation.Password,

Role = Role.Author,

};

author.Id = \_authorService.Add(author);

await Authenticate(author);

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

}

return View();

}

[HttpGet]

public IActionResult Login()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Login(AccountAuthenticateValidation accountAuthenticate)

{

if (ModelState.IsValid)

{

var author = \_authorService.GetAll()

.FirstOrDefault(e => e.Email == accountAuthenticate.Email &&

e.Password == accountAuthenticate.Password);

if (author != null)

{

await Authenticate(author);

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

}

else

{

ModelState.AddModelError("", "No such account");

}

}

return View();

}

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync();

return Redirect("/");

}

private async Task Authenticate(Author author)

{

var claims = new List<Claim>

{

new Claim("Id", author.Id.ToString()),

new Claim(ClaimsIdentity.DefaultNameClaimType, author.Name),

new Claim("Email", author.Email),

new Claim(ClaimsIdentity.DefaultRoleClaimType, author.Role.ToString())

};

ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));

}

public IActionResult ShowProfile()

{

var id = int.Parse(User.Claims.First(e => e.Type == "Id").Value);

var author = \_authorService.GetAll().First(e => e.Id == id);

return View(author);

}

[HttpGet]

public IActionResult EditProfile()

{

var id = int.Parse(User.Claims.First(e => e.Type == "Id").Value);

var author = \_authorService.GetAll().First(e => e.Id == id);

return View(author);

}

[HttpPost]

public IActionResult EditProfile(EditProfileViewModel editModel)

{

var author = \_authorService.GetAll().First(e => e.Id == int.Parse(User.Claims.First(e => e.Type == "Id").Value));

author.Name = editModel.Name;

author.Email = editModel.Email;

author.Description = editModel.Description;

author.Phone = editModel.Phone;

\_authorService.Update(author);

return RedirectToAction("ShowProfile");

}

}

}

Файл ArticleController.cs

using BLL.BusinessInterfaces;

using CourseWork.Models.ArticleModels;

using EntityModels.DamainEntities;

using EntityModels.Users;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System;

using System.IO;

using System.Linq;

namespace CourseWork.Controllers

{

public class ArticleController : Controller

{

private readonly IService<Article> \_articleService;

private readonly IService<Topic> \_topicService;

private readonly IService<Author> \_authorService;

private readonly IService<ArticleComment> \_articleCommentService;

public ArticleController(

IService<Article> articleService,

IService<Topic> topicService,

IService<Author> authorService,

IService<ArticleComment> articleCommentService)

{

\_articleService = articleService;

\_topicService = topicService;

\_authorService = authorService;

\_articleCommentService = articleCommentService;

}

public IActionResult Add()

{

var articleViewModel = new ArticleAddViewModel

{

Topics = \_topicService.GetAll(),

};

return View(articleViewModel);

}

[HttpPost]

public IActionResult AddArticle(ArticleAddValidation articleAddValidation)

{

if (ModelState.IsValid)

{

var article = new Article

{

Title = articleAddValidation.Title,

Text = articleAddValidation.Text,

ImageBytes = GetByteArrayFromImage(articleAddValidation.Image),

NumberOfViews = 0,

Topic = \_topicService.GetAll().First(e => e.Name == articleAddValidation.Topic),

Author = \_authorService.GetAll().First(e => e.Name == articleAddValidation.UserName),

Date = DateTime.Now

};

\_articleService.Add(article);

}

return RedirectToAction("Add");

}

public IActionResult AddComment(int articleId, string text)

{

var comment = new ArticleComment

{

ArticleId = articleId,

AuthorId = int.Parse(User.Claims.First(e => e.Type == "Id").Value),

DateTime = DateTime.Now,

Text = text,

};

\_articleCommentService.Add(comment);

return RedirectToAction("ShowArticle", new { articleId });

}

public IActionResult RemoveArticle(int articleId)

{

\_articleService.Remove(articleId);

return RedirectToAction("ArticlesByAuthor", "Article");

}

public IActionResult Edit()

{

return View();

}

[HttpGet]

public IActionResult ShowArticle(int articleId)

{

var showArticleViewModels = new ShowArticleViewModel

{

Article = \_articleService.GetAll().First(e => e.Id == articleId),

ArticleComments = \_articleCommentService.GetAll().Where(e => e.ArticleId == articleId).ToList(),

};

showArticleViewModels.Article.NumberOfViews += 1;

\_articleService.Update(showArticleViewModels.Article);

return View(showArticleViewModels);

}

public IActionResult ArticlesByAuthor()

{

var authorId = int.Parse(User.Claims.First(e => e.Type == "Id").Value);

var artices = \_articleService.GetAll().Where(e => e.Author.Id == authorId);

return View(artices);

}

private byte[] GetByteArrayFromImage(IFormFile file)

{

using (var target = new MemoryStream())

{

file.CopyTo(target);

return target.ToArray();

}

}

}

}

Файл HomeController.cs

using BLL.BusinessInterfaces;

using CourseWork.Models;

using CourseWork.Models.ArticleModels;

using EntityModels.DamainEntities;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

namespace CourseWork.Controllers

{

[Authorize]

public class HomeController : Controller

{

private readonly IService<Article> \_articleServices;

private readonly IService<ArticleComment> \_articleCommentService;

private readonly IService<Topic> \_topicService;

public HomeController(IService<Article> articleService, IService<ArticleComment> articleCommentService, IService<Topic> topicService)

{

\_articleServices = articleService;

\_articleCommentService = articleCommentService;

\_topicService = topicService;

}

[Authorize(Roles = "Author")]

public IActionResult Index(int? topicId, string title, int page = 1)

{

int pageSize = 5;

//фильтрация

IEnumerable<Article> articles = null;

if (!string.IsNullOrEmpty(title))

{

articles = \_articleServices.GetAll().Where(e => e.Title.Contains(title) && e.IsApproved == true);

}

else

{

articles = \_articleServices.GetAll().Where(e => e.IsApproved == true);

}

if (topicId != null && topicId != 0)

{

articles = articles.Where(e => e.Topic.Id == topicId);

}

// пагинация

var count = articles.Count();

var items = articles.Skip((page - 1) \* pageSize).Take(pageSize).ToList();

var commentsNumber = new List<int>();

foreach (var item in items)

{

commentsNumber.Add(

\_articleCommentService.GetAll().Count(e => e.ArticleId == item.Id));

}

// формируем модель представления

var viewModel = new ArticlesPaginationViewModel

{

ArticlePageViewModel = new ArticlePageViewModel(count, page, pageSize),

ArticleFilterViewModel = new ArticleFilterViewModel(\_topicService.GetAll(), topicId, title),

Articles = items.ToList(),

CommentsCount = commentsNumber,

};

return View(viewModel);

}

[Authorize(Roles = "Manager")]

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

Файл SiteManagementController.cs

using BLL.BusinessInterfaces;

using CourseWork.Models.SiteManagement;

using EntityModels.DamainEntities;

using EntityModels.Users;

using Microsoft.AspNetCore.Mvc;

using System.Linq;

namespace CourseWork.Controllers

{

public class SiteManagementController : Controller

{

private readonly IService<Topic> \_topicService;

private readonly IService<Article> \_articleService;

private readonly IService<Author> \_authorService;

public SiteManagementController(

IService<Topic> topicService,

IService<Article> articleService,

IService<Author> authorService)

{

\_topicService = topicService;

\_articleService = articleService;

\_authorService = authorService;

}

public IActionResult ManagementBody()

{

return View(new SiteManagementViewModel());

}

public IActionResult GetArticlePartialView()

{

var articlePartialView = "ApproveArticle";

return View("ManagementBody", new SiteManagementViewModel

{

PartialViewName = articlePartialView,

});

}

public IActionResult ApproveArticle(int id)

{

var article = \_articleService.GetAll().First(e => e.Id == id);

article.IsApproved = true;

\_articleService.Update(article);

return RedirectToAction("GetArticlePartialView");

}

public IActionResult DisapproveArticle(int id)

{

\_articleService.Remove(id);

return RedirectToAction("GetArticlePartialView");

}

public IActionResult GetTopicPartialView()

{

var topicPartialView = "TopicManagement";

return View("ManagementBody", new SiteManagementViewModel

{

PartialViewName = topicPartialView,

});

}

public IActionResult AddTopic(string topicName)

{

\_topicService.Add(new Topic

{

Name = topicName,

});

return RedirectToAction("GetTopicPartialView");

}

public IActionResult RemoveTopic(int id)

{

\_topicService.Remove(id);

return RedirectToAction("GetTopicPartialView");

}

public IActionResult GetAuthorPartialView()

{

var authorPartialView = "AuthorManagement";

return View("ManagementBody", new SiteManagementViewModel

{

PartialViewName = authorPartialView,

});

}

public IActionResult IncreaceRoleToAuthor(int authorId)

{

var author = \_authorService.GetAll().First(e => e.Id == authorId);

switch (author.Role)

{

case Role.Author:

author.Role = Role.Manager;

break;

case Role.Manager:

author.Role = Role.Admin;

break;

case Role.Admin:

default:

break;

}

\_authorService.Update(author);

return RedirectToAction("GetAuthorPartialView");

}

public IActionResult DecreaceRoleToAuthor(int authorId)

{

var author = \_authorService.GetAll().First(e => e.Id == authorId);

switch (author.Role)

{

case Role.Manager:

author.Role = Role.Author;

break;

case Role.Admin:

author.Role = Role.Manager;

break;

case Role.Author:

default:

break;

}

\_authorService.Update(author);

return RedirectToAction("GetAuthorPartialView");

}

}

}

Файл AccountAuthenticateValidation.cs

using System.ComponentModel.DataAnnotations;

namespace CourseWork.Models.AccountModels

{

public class AccountAuthenticateValidation

{

[Required(ErrorMessage = "Enter email address")]

[DataType(DataType.EmailAddress)]

public string Email { get; set; }

[Required(ErrorMessage = "Enter password")]

[DataType(DataType.Password)]

public string Password { get; set; }

}

}

Файл AccountRegistrationValidation.cs

using System.ComponentModel.DataAnnotations;

namespace CourseWork.Models.AccountModels

{

public class AccountRegistrationValidation

{

[Required(ErrorMessage = "Enter a nickname")]

[DataType(DataType.Text)]

public string Nickname { get; set; }

[Required(ErrorMessage = "Enter email address")]

[DataType(DataType.EmailAddress)]

public string Email { get; set; }

[Required(ErrorMessage = "Enter password")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required(ErrorMessage = "Confirm your password")]

[Compare("Password")]

public string ConfirmPassword { get; set; }

}

}

Файл EditProfileViewModel.cs

using System.ComponentModel.DataAnnotations;

namespace CourseWork.Models.AccountModels

{

public class EditProfileViewModel

{

[Required]

public string Name { get; set; }

[Required]

[DataType(DataType.EmailAddress)]

public string Email { get; set; }

public string Description { get; set; }

[DataType(DataType.PhoneNumber)]

public string Phone { get; set; }

}

}

Файл ArticleAddValidation.cs

using Microsoft.AspNetCore.Http;

using System.ComponentModel.DataAnnotations;

namespace CourseWork.Models.ArticleModels

{

public class ArticleAddValidation

{

[Required(ErrorMessage = "Enter topic")]

public string Topic { get; set; }

[Required(ErrorMessage = "Invalid user name")]

public string UserName { get; set; }

[Required(ErrorMessage = "Enter Title")]

public string Title { get; set; }

[Required(ErrorMessage = "Choose an image for article")]

public IFormFile Image { get; set; }

[Required(ErrorMessage = "Enter an article text.")]

public string Text { get; set; }

}

}

Файл ArticleAddViewModel.cs

using EntityModels.DamainEntities;

using System.Collections.Generic;

namespace CourseWork.Models.ArticleModels

{

public class ArticleAddViewModel

{

public IEnumerable<Topic> Topics { get; set; }

public ArticleAddValidation ArticleAddValidation { get; set; }

}

}

Файл ArticleFilterViewModel.cs

using EntityModels.DamainEntities;

using Microsoft.AspNetCore.Mvc.Rendering;

using System.Collections.Generic;

namespace CourseWork.Models.ArticleModels

{

public class ArticleFilterViewModel

{

public ArticleFilterViewModel(List<Topic> topics, int? topic, string title)

{

topics.Insert(0, new Topic { Name = "Все", Id = 0 });

Topics = new SelectList(topics, "Id", "Name", topic);

SelectedTopic = topic;

SelectedTitle = title;

}

public SelectList Topics { get; private set; }

public int? SelectedTopic { get; private set; }

public string SelectedTitle { get; private set; }

}

}

Файл ArticlePageViewModel.cs

using System;

namespace CourseWork.Models.ArticleModels

{

public class ArticlePageViewModel

{

public int PageNumber { get; private set; }

public int TotalPages { get; private set; }

public ArticlePageViewModel(int count, int pageNumber, int pageSize)

{

PageNumber = pageNumber;

TotalPages = (int)Math.Ceiling(count / (double)pageSize);

}

public bool HasPreviousPage

{

get

{

return (PageNumber > 1);

}

}

public bool HasNextPage

{

get

{

return (PageNumber < TotalPages);

}

}

}

}

Файл ArticlesPaginationViewModel.cs

using EntityModels.DamainEntities;

using System.Collections.Generic;

namespace CourseWork.Models.ArticleModels

{

public class ArticlesPaginationViewModel

{

public List<Article> Articles { get; set; }

public List<int> CommentsCount { get; set; }

public ArticlePageViewModel ArticlePageViewModel { get; set; }

public ArticleFilterViewModel ArticleFilterViewModel { get; set; }

}

}

Файл ShowArticleViewModel.cs

using EntityModels.DamainEntities;

using System.Collections.Generic;

namespace CourseWork.Models.ArticleModels

{

public class ShowArticleViewModel

{

public Article Article { get; set; }

public List<ArticleComment> ArticleComments { get; set; }

}

}

Файл ArticleManagementViewModel.cs

using EntityModels.DamainEntities;

using System.Collections.Generic;

namespace CourseWork.Models.SiteManagement

{

public class ArticleManagementViewModel

{

public List<Article> Articles { get; set; }

}

}

Файл SiteManagementViewModel.cs

namespace CourseWork.Models.SiteManagement

{

public class SiteManagementViewModel

{

public string PartialViewName { get; set; }

}

}

Файл TopicManagementViewModel.cs

using EntityModels.DamainEntities;

using System.Collections.Generic;

namespace CourseWork.Models.SiteManagement

{

public class TopicManagementViewModel

{

public List<Topic> Topics { get; set; }

public List<int> Approved { get; set; }

public List<int> NotApproved { get; set; }

public List<int> AmountOfViews { get; set; }

}

}

Файл ApproveArticleViewComponent.cs

using BLL.BusinessInterfaces;

using EntityModels.DamainEntities;

using Microsoft.AspNetCore.Mvc;

using System.Linq;

using CourseWork.Models.SiteManagement;

namespace CourseWork.ViewComponents.ManagementBody

{

public class ApproveArticleViewComponent : ViewComponent

{

private readonly IService<Article> \_articleService;

public ApproveArticleViewComponent(IService<Article> articleService)

{

\_articleService = articleService;

}

public IViewComponentResult Invoke()

{

var model = new ArticleManagementViewModel

{

Articles = \_articleService.GetAll().Where(e => e.IsApproved == false).ToList(),

};

return View("ApproveArticle", model);

}

}

}

Файл AuthorManagementViewComponent.cs

using BLL.BusinessInterfaces;

using EntityModels.Users;

using Microsoft.AspNetCore.Mvc;

namespace CourseWork.ViewComponents.ManagementBody

{

public class AuthorManagementViewComponent : ViewComponent

{

private readonly IService<Author> \_authorService;

public AuthorManagementViewComponent(IService<Author> authorService)

{

\_authorService = authorService;

}

public IViewComponentResult Invoke()

{

var authors = \_authorService.GetAll();

return View("AuthorManagement", authors);

}

}

}

Файл TopicManagementViewComponent.cs

using BLL.BusinessInterfaces;

using CourseWork.Models.SiteManagement;

using EntityModels.DamainEntities;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

namespace CourseWork.ViewComponents.ManagementBody

{

public class TopicManagementViewComponent : ViewComponent

{

private readonly IService<Topic> \_topicService;

private readonly IService<Article> \_articleService;

public TopicManagementViewComponent(IService<Topic> topicService, IService<Article> articleService)

{

\_topicService = topicService;

\_articleService = articleService;

}

public IViewComponentResult Invoke()

{

var approvedArticlesNumber = new List<int>();

var notApprovedArticlesNumber = new List<int>();

var amounOfViews = new List<int>();

foreach (Topic topic in \_topicService.GetAll())

{

approvedArticlesNumber.Add(

\_articleService.GetAll()

.Where(e => e.Topic.Name == topic.Name && e.IsApproved == true)

.Count()

);

notApprovedArticlesNumber.Add(

\_articleService.GetAll()

.Where(e => e.Topic.Name == topic.Name && e.IsApproved == false)

.Count()

);

amounOfViews.Add(\_articleService.GetAll()

.Where(e => e.Topic.Name == topic.Name && e.IsApproved == true)

.Sum(e => e.NumberOfViews)

);

}

var model = new TopicManagementViewModel

{

Topics = \_topicService.GetAll(),

Approved = approvedArticlesNumber,

NotApproved = notApprovedArticlesNumber,

AmountOfViews = amounOfViews,

};

return View("TopicManagement", model);

}

}

}

Файл EditProfile.cshtml

@model EntityModels.Users.Author

<title>Edit Profile</title>

<div>

<form method="post" **asp-antiforgery**="true">

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

<div class="form-group">

<label **asp-for**="Name">Change Name</label>

<input **asp-for**="Name" class="form-control" **value**="@Model.Name"/>

</div>

<div class="form-group">

<label **asp-for**="Email">Change Email</label>

<input **asp-for**="Email" class="form-control" **value**="@Model.Email"/>

</div>

<div class="form-group">

<label **asp-for**="Description">Change Description</label>

<textarea **asp-for**="Description" class="form-control" wrap="hard">@Model.Description</textarea>

</div>

<div class="form-group">

<label **asp-for**="Phone">Change Phone</label>

<input **asp-for**="Phone" class="form-control" **value**="@Model.Phone"/>

</div>

<button type="submit" class="btn btn-success">Save</button>

<a class="btn btn-outline-info" **asp-action**="ShowProfile">Back</a>

</form>

</div>

Файл Login.cshtml

@model CourseWork.Models.AccountModels.AccountAuthenticateValidation

@{

ViewData["Title"] = "Login";

}

<div class="d-flex justify-content-center align-content-center">

<form class="shadow mt-5 p-4" method="post" **asp-action**="Login" **asp-controller**="Account" **asp-antiforgery**="true">

<h1>Log In</h1>

<a **asp-action**="Register" **asp-controller**="Account">Registration</a>

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

<table>

<tr class="form-group">

<td>

<label **asp-for**="Email">Enter Email</label>

</td>

<td>

<input class="form-control" **type**="text" **asp-for**="Email" />

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

</td>

</tr>

<tr class="form-group">

<td>

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

</td>

<td>

<input class="form-control" **type**="password" **asp-for**="Password" />

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

</td>

</tr>

</table>

<div class="form-group d-flex justify-content-center mt-2">

<input type="submit" value="Войти" class="btn btn-outline-dark" />

</div>

</form>

</div>

Файл Register.cshtml

@model CourseWork.Models.AccountModels.AccountRegistrationValidation

@{

ViewData["Title"] = "Register";

}

<div class="d-flex justify-content-center align-content-center">

<form class="shadow mt-4 p-4" method="post" **asp-action**="Register" **asp-controller**="Account" **asp-antiforgery**="true">

<h1>Registration</h1>

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

<table>

<tr>

<td class="form-group">

<label **asp-for**="Nickname">Enter Nickname</label>

</td>

<td>

<input class="form-control" **type**="text" **asp-for**="Nickname" />

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

</td>

</tr>

<tr>

<td class="form-group">

<label **asp-for**="Email">Enter Email</label>

</td>

<td>

<input class="form-control" **type**="text" **asp-for**="Email" />

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

</td>

</tr>

<tr>

<td class="form-group">

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

</td>

<td>

<input class="form-control" **asp-for**="Password" />

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

</td>

</tr>

<tr>

<td class="form-group">

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

</td>

<td>

<input class="form-control" **asp-for**="ConfirmPassword" />

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

</td>

</tr>

</table>

<div class="form-group d-flex justify-content-center mt-2">

<input type="submit" class="btn btn-success" value="Register" **asp-action**="Register" />

</div>

</form>

</div>

Файл ShowProfile.cshtml

@model EntityModels.Users.Author

<div>

<div>

<h1>@Model.Name</h1>

</div>

<div>

<label class="font-weight-bold">Email: </label>

<p style="font-size:larger">@Model.Email</p>

</div>

<div>

<label class="font-weight-bold">Description: </label>

<p style="font-size:larger">@Model.Description</p>

</div>

<div>

<label class="font-weight-bold">Phone: </label>

<p style="font-size:larger">@Model.Phone</p>

</div>

<div>

<label class="font-weight-bold">Role: </label>

<p style="font-size:larger">@Model.Role</p>

</div>

<div>

<a **asp-controller**="Account" **asp-action**="EditProfile" class="btn btn-outline-info">Edit Profile</a>

</div>

</div>

Файл Add.cshtml

@model CourseWork.Models.ArticleModels.ArticleAddViewModel

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

<div>

<form class="" action="AddArticle" method="post" role="form" **asp-antiforgery**="true" enctype="multipart/form-data">

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

<div class="d-sm-flex justify-content-between">

<div class="form-group">

<label class="form-check-label" for="chuseTopic">Topic</label>

<input class="form-control" id="chuseTopic" **type**="text" **asp-for**="ArticleAddValidation.Topic" list="topicsList" placeholder="Topic" />

<datalist id="topicsList">

@foreach (var topic in Model.Topics)

{

<option **value**="@topic.Name" />

}

</datalist>

<span **asp-validation-for**="@Model.ArticleAddValidation.Topic"></span>

</div>

<div class="form-group">

<label class="form-check-label" for="userName">Author</label>

<input class="form-control" id="userName" **type**="text" **asp-for**="ArticleAddValidation.UserName" placeholder="Author" **value**="@User.Identity.Name" />

<span **asp-validation-for**="@Model.ArticleAddValidation.UserName"></span>

</div>

<div class="p-3 form-group">

<input class="m-2 form-control" id="AddImg" **type**="file" **asp-for**="ArticleAddValidation.Image" **name**="articleAddValidation.Image" onchange="showImg(event);" />

<span **asp-validation-for**="@Model.ArticleAddValidation.Image"></span>

</div>

</div>

<div class="form-group">

<label class="form-check-label" for="articleTitle">Article title</label>

<input id="articleTitle" class="form-control" **type**="text" **asp-for**="ArticleAddValidation.Title" placeholder="Title" />

<span **asp-validation-for**="@Model.ArticleAddValidation.Title"></span>

</div>

<div class="form-group">

<div id="divForImg" class="p-3 d-flex justify-content-center align-content-center">

</div>

<div class="">

<label class="form-check-label" for="articleText">Article text</label>

<textarea id="articleText" class="form-control w-100 text-dark" wrap="hard" **asp-for**="ArticleAddValidation.Text" style="max-height: 100%"></textarea>

<span **asp-validation-for**="@Model.ArticleAddValidation.Text"></span>

</div>

</div>

<div class="d-flex justify-content-center mt-4">

<button class="btn btn-success" type="submit">Add article</button>

</div>

</form>

</div>

@section scripts{

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

}

Файл ArticlesByAuthor.cshtml

@using EntityModels.DamainEntities;

@model IEnumerable<Article>;

<div>

<h2>My Articles</h2>

@if (Model.Any())

{

@foreach (var article in Model)

{

<div class="card mb-3" style="max-width: 100%;">

<div class="row g-0">

<div class="col-md-4">

<img class="card-img-overlay" style="height: 100%; width: 100%; object-fit:contain;" src="data:image/jpeg;base64,@Convert.ToBase64String(article.ImageBytes)" alt="ArticleImage">

</div>

<div class="col-md-8">

<div class="card-body">

<h5 class="card-title"><a class="card-link badge-light" **asp-action**="ShowArticle" **asp-controller**="Article" **asp-route-articleId**="@article.Id">@article.Title</a></h5>

<p class="card-text">@article.Text</p>

</div>

<div class="card-footer d-flex justify-content-between">

<ul type="None" class="m-0 p-2 d-sm-flex">

<li>@article.Date.ToString("dd.MM.yyyy HH:mm")</li>

<li class="ml-2">@article.Author.Name</li>

</ul>

<div>

<span><img src="~/eyeIcon.png" alt="Views" /></span>

<span>@article.NumberOfViews</span>

</div>

<a class="btn btn-outline-danger" **asp-controller**="Article" **asp-action**="RemoveArticle" **asp-route-articleId**="@article.Id">Remove</a>

</div>

</div>

</div>

</div>

}

}

</div>

Файл ShowArticle.cshtml

@using System.Security.Claims

@model CourseWork.Models.ArticleModels.ShowArticleViewModel

<div>

<div>

<h2>@Model.Article.Title</h2>

<div class="p-2">

<span class="font-weight-bold text-capitalize">@Model.Article.Author.Name</span>

<div>

@Model.Article.Date.ToString("dd.MM.yyyy HH:mm")

<span><img src="~/eyeIcon.png"/></span>

<span>@Model.Article.NumberOfViews</span>

<span><img src="~/comments.png"/></span>

<span>@Model.ArticleComments.Count</span>

</div>

</div>

<div>

<img class="card-img-top" src="data:image/jpeg;base64,@Convert.ToBase64String(Model.Article.ImageBytes)" alt="Card image cap">

</div>

<div class="mt-3">

<p class="text-md-left text-wrap">@Model.Article.Text</p>

</div>

</div>

@if (User.Identity.IsAuthenticated)

{

<div>

<form **asp-antiforgery**="true">

<h3>Leav comment</h3>

<p>From: <b>@User.Identity.Name</b></p>

<div>

<span>Comment</span>

<textarea class="form-control w-100 text-dark" name="text" wrap="hard"></textarea>

</div>

<div class="d-flex justify-content-end">

<button class="btn btn-outline-success mt-3" type="submit"

**asp-controller**="Article"

**asp-action**="AddComment"

**asp-route-articleId**="@Model.Article.Id">

Save Comment

</button>

</div>

</form>

</div>

}

@if (Model.ArticleComments.Any())

{

<div class="mt-4">

<h3>Coments:</h3>

@foreach (var item in Model.ArticleComments)

{

<div class="bg-white p-1 mb-2">

<span class="text-info">@item.DateTime.ToString("dd.MM.yyyy HH.mm")</span>

<span class="text-primary"> From: <b>@item.Author.Name</b> </span>

<p class="text fs-4">Text: @item.Text</p>

</div>

}

</div>

}

</div>

Файл Index.cshtml

@model CourseWork.Models.ArticleModels.ArticlesPaginationViewModel

@{

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

}

<div>

<form class="d-flex justify-content-between">

<div class="form-group">

<label>Title: </label>

<input name="title" value="@Model.ArticleFilterViewModel.SelectedTitle" class="form-control" />

</div>

<div class="form-group">

<label>Topic: </label>

<select **name**="topicId" **asp-items**="Model.ArticleFilterViewModel.Topics" class="form-control"></select>

</div>

<div class="form-group">

<br />

<button type="submit" class="btn mt-2 btn-outline-dark">Find</button>

</div>

</form>

</div>

<div>

@for (int i = 0; i < Model.Articles.Count; i++)

{

<div class="card mb-3" style="max-width: 100%;">

<div class="row g-0">

<div class="col-md-4">

<img class="card-img-overlay" style="height: 100%; width: 100%; object-fit:contain;" src="data:image/jpeg;base64,@Convert.ToBase64String(Model.Articles[i].ImageBytes)" alt="ArticleImage">

</div>

<div class="col-md-8">

<div class="card-body">

<h5 class="card-title"><a class="card-link badge-light" **asp-action**="ShowArticle" **asp-controller**="Article" **asp-route-articleId**="@Model.Articles[i].Id">@Model.Articles[i].Title</a></h5>

<p class="card-text">@Model.Articles[i].Text.Substring(0,4)</p>

</div>

<div class="card-footer d-flex justify-content-between">

<ul type="None" class="m-0 p-2 d-sm-flex">

<li>@Model.Articles[i].Date.ToString("dd.MM.yyyy HH:mm")</li>

<li class="ml-2">@Model.Articles[i].Author.Name</li>

</ul>

<div>

<span><img src="~/eyeIcon.png" alt="Views" /></span>

<span>@Model.Articles[i].NumberOfViews</span>

</div>

<div>

<span><img src="~/comments.png" alt="Comments" /></span>

<span>@Model.CommentsCount[i]</span>

</div>

</div>

</div>

</div>

</div>

}

</div>

@if (Model.ArticlePageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.ArticlePageViewModel.PageNumber - 1)"

**asp-route-title**="@(Model.ArticleFilterViewModel.SelectedTitle)"

**asp-route-topicId**="@(Model.ArticleFilterViewModel.SelectedTopic)"

class="btn btn-outline-dark">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.ArticlePageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.ArticlePageViewModel.PageNumber + 1)"

**asp-route-title**="@(Model.ArticleFilterViewModel.SelectedTitle)"

**asp-route-topicId**="@(Model.ArticleFilterViewModel.SelectedTopic)"

class="btn btn-outline-dark">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

Файл ApproveArticle.cshtml

@using CourseWork.Models.SiteManagement;

@model ArticleManagementViewModel

<div>

@if (Model.Articles.Any())

{

foreach (var key in Model.Articles.GroupBy(e => e.Topic.Name).OrderBy(e => e.Key))

{

<h2>@key.Key</h2>

foreach (var article in key)

{

<div class="card mb-3" style="max-width: 100%;">

<div class="row g-0">

<div class="col-md-4">

<img class="card-img-overlay" style="height: 100%; width: 100%; object-fit:contain;" src="data:image/jpeg;base64,@Convert.ToBase64String(article.ImageBytes)" alt="ArticleImage">

</div>

<div class="col-md-8">

<div class="card-body">

<h5 class="card-title"><a class="card-link badge-light" **asp-action**="ShowArticle" **asp-controller**="Article" **asp-route-articleId**="@article.Id">@article.Title</a></h5>

<p class="card-text">@article.Text</p>

</div>

<div class="card-footer d-flex justify-content-between">

<ul type="None" class="m-0 p-2 d-sm-flex">

<li>@article.Date.ToString("dd.MM.yyyy HH:mm")</li>

<li class="ml-2">@article.Author.Name</li>

</ul>

<a class="btn btn-outline-primary" **asp-controller**="SiteManagement" **asp-action**="ApproveArticle" **asp-route-id**="@article.Id">Approve</a>

<a class="btn btn-outline-danger" **asp-controller**="SiteManagement" **asp-action**="DisapproveArticle" **asp-route-id**="@article.Id">Disapprove</a>

</div>

</div>

</div>

</div>

}

}

}

else

{

<h2>There are no article to approve</h2>

}

</div>

Файл AuthorManagement.cshtml

@using EntityModels.Users

@model IEnumerable<Author>

<div>

<form>

<table class="m-2">

<tr>

<td>Name</td>

<td>Email</td>

<td>Phone</td>

<td>Role</td>

</tr>

@foreach (var author in Model)

{

<tr>

<td class="w-25">@author.Name</td>

<td class="w-25">@author.Email</td>

<td class="w-25">@author.Phone</td>

<td class="w-25">@author.Role.ToString()</td>

<td>

<a class="btn btn-success" **asp-action**="IncreaceRoleToAuthor" **asp-route-authorId**="@author.Id">Inc</a>

</td>

<td>

<a class="btn btn-danger" **asp-action**="DecreaceRoleToAuthor" **asp-route-authorId**="@author.Id">Dec</a>

</td>

</tr>

}

</table>

</form>

</div>

Файл TopicManagement.cshtml

@using CourseWork.Models.SiteManagement;

@model TopicManagementViewModel

<div>

<div class="p-2">

<form class="d-flex" **asp-action**="AddTopic" method="get" **asp-antiforgery**="true">

<div class="col-8">

<input class="form-control" type="text" name="topicName" placeholder="Topic name"/>

</div>

<div class="col-3">

<button class="btn btn-success" type="submit" **asp-action**="AddTopic">Add Topic</button>

</div>

</form>

</div>

<div class="p-2">

<form **asp-antiforgery**="true">

<table>

<tr>

<td class="p-2 w-25">Name</td>

<td class="p-2 w-25">Views</td>

<td class="p-2 w-25">Approved Articles</td>

<td class="p-2 w-25">Not Approved Articles</td>

</tr>

@if (Model.Topics.Any())

{

for (int i = 0; i < Model.Topics.Count; i++)

{

<tr>

<td class="p-2 w-25">@Model.Topics[i].Name</td>

<td class="p-2 w-25">@Model.AmountOfViews[i]</td>

<td class="p-2 w-25">@Model.Approved[i]</td>

<td class="p-2 w-25">@Model.NotApproved[i]</td>

<td>

<button type="submit" class="btn btn-danger" **asp-controller**="SiteManagement" **asp-action**="RemoveTopic" **asp-route-id**="@Model.Topics[i].Id">Remove</button>

</td>

</tr>

}

}

</table>

</form>

</div>

</div>

Файл \_Layout.cshtml

@using EntityModels.DamainEntities;

@inject BLL.BusinessInterfaces.IService<Article> \_articles

@{

var letest = \_articles.GetAll().Where(e => e.IsApproved == true).LastOrDefault();

var mostPopular = \_articles.GetAll().Where(e => e.IsApproved == true).OrderByDescending(e => e.NumberOfViews).FirstOrDefault();

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - CourseWork</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body style="background: #f2f3f5;">

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">News</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">All News</a>

</li>

@if (User.Identity.IsAuthenticated)

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="SiteManagement" **asp-action**="ManagementBody">Manage Site</a>

</li>

}

</ul>

</div>

<div class="my-2 my-lg-0">

@if (User.Identity.IsAuthenticated)

{

<div class="btn-group">

<a class="btn btn-outline-dark btn-sm" **asp-controller**="Account" **asp-action**="ShowProfile">

@User.Identity.Name

</a>

<button type="button" class="btn btn-sm btn-outline-info dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

<span class="sr-only">Toggle Dropdown</span>

</button>

<div class="dropdown-menu">

<a class="dropdown-item" **asp-controller**="Article" **asp-action**="ArticlesByAuthor">My Articles</a>

<a class="dropdown-item" **asp-controller**="Article" **asp-action**="Add">Write Article</a>

<a class="dropdown-item" **asp-controller**="Account" **asp-action**="Logout">Logout</a>

</div>

</div>

}

else

{

<div class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="Login">Log in</a>

</div>

}

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3 col-9 d-flex">

<div class="col-12">

@RenderBody()

</div>

<div>

<div>

<div>

<label class="col-form-label-lg modal-title font-weight-bold">Latest News</label>

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="data:image/jpeg;base64,@Convert.ToBase64String(letest?.ImageBytes)" alt="Card image cap">

<div class="card-body">

<h5 class="card-title">@letest?.Title</h5>

<p class="card-text">@letest?.Text.Substring(0, letest.Text.Length / 3)...</p>

<a **asp-action**="ShowArticle" **asp-controller**="Article" **asp-route-articleId**="@letest.Id" class="btn btn-primary">Go to Article</a>

</div>

</div>

</div>

<div class="mt-4">

<label class="col-form-label-lg modal-title font-weight-bold">Most Popular</label>

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="data:image/jpeg;base64,@Convert.ToBase64String(mostPopular?.ImageBytes)" alt="Card image cap">

<div class="card-body">

<h5 class="card-title">@mostPopular?.Title</h5>

<p class="card-text">@mostPopular?.Text.Substring(0, mostPopular.Text.Length / 3) ...</p>

<a class="btn btn-primary" **asp-action**="ShowArticle" **asp-controller**="Article" **asp-route-articleId**="@mostPopular.Id">Go to Article</a>

</div>

</div>

</div>

</div>

</div>

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2021 - CourseWork - <a **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</a>

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

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

@RenderSection("Scripts", required: false)

</body>

</html>

Файл ManagementBody.cshtml

@using CourseWork.Models.SiteManagement;

@model SiteManagementViewModel;

<body>

<div class="navbar-collapse bg-white">

<ul class="navbar-nav d-flex flex-row">

<li class="nav-tabs mr-3">

<a class="nav-link text-dark" **asp-action**="GetArticlePartialView">Approve Articles</a>

</li>

<li class="nav-tabs mr-3">

<a class="nav-link text-dark" **asp-action**="GetTopicPartialView">Topics</a>

</li>

<li class="nav-tabs mr-3">

<a class="nav-link text-dark" **asp-action**="GetAuthorPartialView">Authors</a>

</li>

</ul>

</div>

<div class="bg-white mt-2">

<div>

@if (!string.IsNullOrEmpty(Model.PartialViewName))

{

@await Component.InvokeAsync(Model.PartialViewName);

}

</div>

</div>

</body>

Файл Startup.cs

using BLL;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace CourseWork

{

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)

{

services.AddBllClassess();

services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");

options.AccessDeniedPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");

});

services.AddControllersWithViews();

}

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

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

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.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

Файл Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace CourseWork

{

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>();

});

}

}