МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированное проектирование и программирование»

Тема «Программное средство «Новостной блог»»

Исполнитель

студент (ка) 3 курса группы 7 Воликов Дмитрий Анатольевич

(Ф.И.О.)

Руководитель работы преподаватель-стажер Якунович А. В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель

(подпись)

Минск 2023

**Содержание**

[Введение 6](#_Toc154061780)

[1. Постановка задачи и обзор литературы 7](#_Toc154061781)

[1.1. Аналитический обзор аналогов 7](#_Toc154061782)

[1.1.1. Аналог «SPUTNIK» 7](#_Toc154061783)

[1.1.2. Аналог «Onliner» 8](#_Toc154061784)

[1.1.3. Аналог «Lenta.ru» 8](#_Toc154061785)

[1.2. Разработка функциональных требований, определение вариантов использования 10](#_Toc154061786)

[1.2.1. Разработка требований к базе данных 10](#_Toc154061787)

[1.2.2. Определение вариантов использования 10](#_Toc154061788)

[1.3. Вывод 11](#_Toc154061789)

[2. Проектирование архитектуры проекта 12](#_Toc154061790)

[2.1. Модель базы данных 12](#_Toc154061791)

[2.1.1. Таблица Users 12](#_Toc154061792)

[2.1.2. Таблица Tags 12](#_Toc154061793)

[2.1.3. Таблица Categories 13](#_Toc154061794)

[2.1.4. Таблица Articles 13](#_Toc154061795)

[2.1.5. Таблица ArticleTags 13](#_Toc154061796)

[2.1.6. Таблица Comments 13](#_Toc154061797)

[2.1.7. Таблица Feedbacks 14](#_Toc154061798)

[2.1.8. Таблица Ratings 14](#_Toc154061799)

[2.2. Структура проекта 14](#_Toc154061800)

[2.3. Основные классы бизнес логики 15](#_Toc154061801)

[2.4. Работа с базой данных 15](#_Toc154061802)

[2.5. Вывод 16](#_Toc154061803)

[3. Разработка функциональной модели и модели данных ПС 17](#_Toc154061804)

[3.1. Модели данных 17](#_Toc154061805)

[3.2. Реализация паттерна MVVM 19](#_Toc154061806)

[3.3. Реализация паттерна Command 25](#_Toc154061807)

[3.4. Вывод 26](#_Toc154061808)

[4. Тестирование 27](#_Toc154061809)

[4.1. Вывод 30](#_Toc154061810)

[Заключение 31](#_Toc154061811)

[Список используемых источников 32](#_Toc154061812)

[Приложение А Диаграмма сущностей базы данных 33](#_Toc154061813)

[Приложение Б Диаграмма классов 34](#_Toc154061814)

[Приложение В Листинг LoginPage.xaml 35](#_Toc154061815)

[Приложение Г Листинг LoginPageVM 37](#_Toc154061816)

[Приложение Д Листинг объекта RelayCommand 38](#_Toc154061817)

[Приложение Е Диаграмма вариантов использования 39](#_Toc154061818)

[Приложение Ж Диаграмма последовательности 40](#_Toc154061819)

# Введение

Новости – это информация о текущих событиях, которая может быть интересна или важна для широкой общественности. Они могут касаться политики, экономики, спорта, культуры, науки и технологий, а также многих других сфер жизни.

Новости могут быть полезны для нас по многим причнам. Новости дают нам представление о том, что происходит в мире, в нашей стране и в нашем городе. Это помогает нам принимать более информированные решения о своей жизни. Они помогают нам быть образованными гражданами, могут научить нас о различных культурах, точках зрения и проблемах. Это помогает нам лучше понимать мир, в котором мы живем. Новости могут быть захватывающими и интересными. Они могут помочь нам отдохнуть и расслабиться после тяжелого дня.

Для курсового проекта было решено спректировать и реализовать приложение для новостного блога (далее – «Новостной портал», «Новостной блог», «Блог»). Цель такой работы – закрепить знания объектно-ориентированного программирвоания, путем реализации приложения с использованием Windows Presentation Foundation (WPF) и хранилищем информации, в роли которого выступает база данных.

# 1. Постановка задачи и обзор литературы

## 1.1. Аналитический обзор аналогов

На сегодняшний день пользователю предоставлено множество сайтов, которые посвящены теме новостей для различных тематик. Портал должен предоставлять пользователям доступ к новостям из различных источников, включая крупные новостные агентства, независимые СМИ и социальные сети. Это позволит пользователям получить объективную и разностороннюю картину происходящего.

Новостной портал должен быть удобным и простым в использовании. Пользователи должны иметь возможность быстро и легко найти интересующую их информацию, путем удобной фильтрации и сортировки событий по датам. В ином случае, никто не будет пользвоаться данным ресурсом.

В современном мире, где информация доступна в неограниченном количестве, новостные порталы конкурируют за внимание пользователей. Чтобы привлечь и удержать внимание аудитории, внешний вид портала должен иметь хорошо спроектированный дизайн.

1.1.1. Аналог «SPUTNIK»

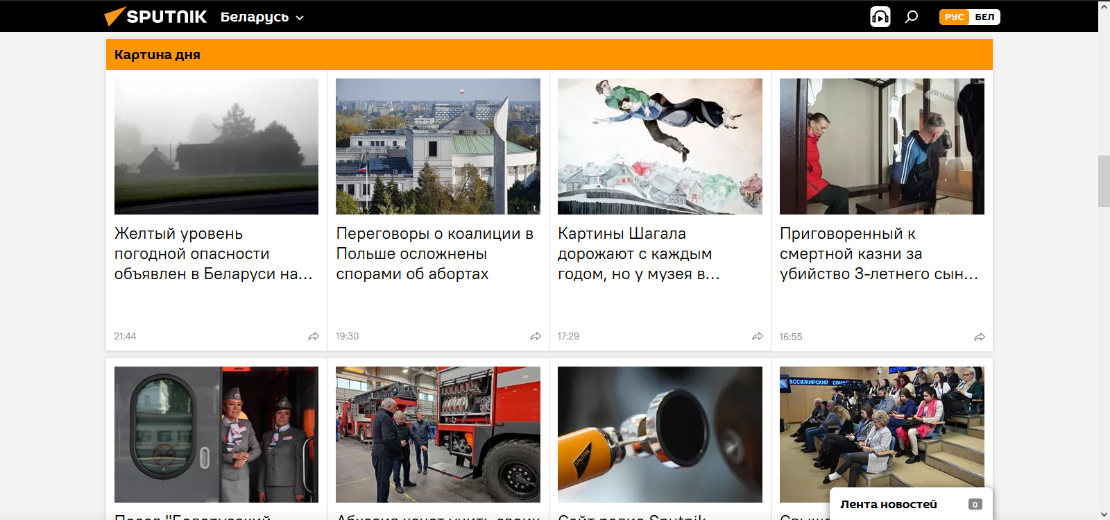
«Sputnik» [1] – довольно популярный среди белорусских пользователей, так как здесь информация довольно удобно структурирована, да и выглядит внешне привлекательно. Здесь всегда свежие новости, которые рекомендуют поисковые алгоритмы. Список новостей представлен ввиде сетки карточек, каждая из которых имеет свой заголовок, изображение и количество просмотров статьи.

Рисунок 1.1 – Главная страница новостного портала Sputnik.by

Также можно выделить приятный для глаза шрифт, который легко читается и гармонирует со стилевым решением портала. Можно заметить некий чёрно-оранжевый стиль проекта, который выглядит серьёзно и придаёт уверенность в том, что пользователь будет дальше читать.

1.1.2. Аналог «Onliner»

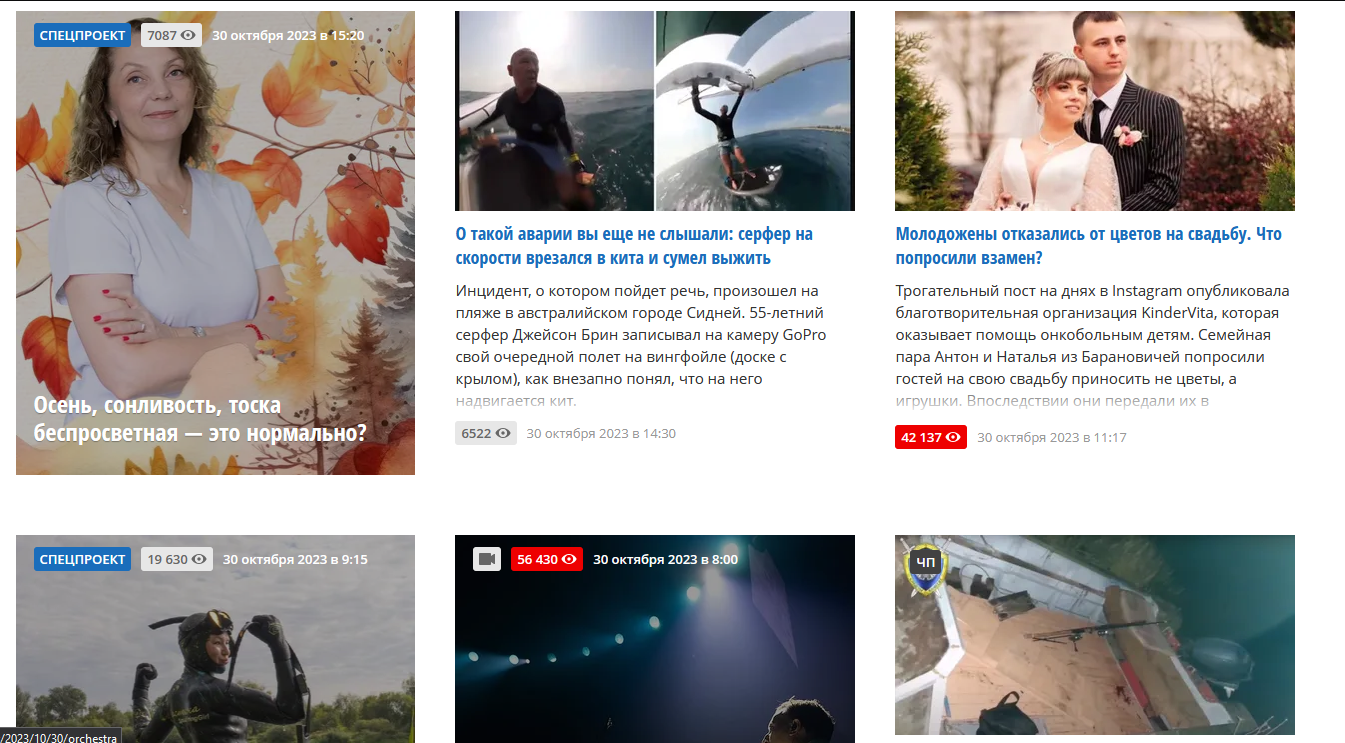
«Onliner» [2] – это известный сайт, который имеет большее количество различного функционала для конечного пользователя, в том числе и просмотр новостных статей. В отличии от предыдущего аналога, данный портал имеет довольно неплохое дизайнерской решение в построении сетки из новостных заголовков. Каждый блок имеет свой размер, а свободное пространство заполняется текстом. Отступы подобраны так, чтобы пользователь сумел легко различить границы статьи, но при этом чтобы не было много свободного пространства.

Рисунок 1.2 – Главная страница сайта Onliner

Легко заметить, что блоки сеток здесь больше и не разделены линиями. Это хорошее стилевое решение, ибо что-то большое сильнее привлекает внимание человека, а отсутствие линий-разделителей не сбивает это самое внимание.

1.1.3. Аналог «Lenta.ru»

«Lenta.ru» [3] – популярный русский новостной портал, который существует в сети с 1999 года. За такое количество лет сайт успел приобрести неплохую репутацию среди пользователей. Данный портал занимает пятое место среди других порталов в его тематике. Сайт содержит новости по различным темам, таким как Россия, мир, экономика, наука и техника, культура, спорт, интернет и СМИ, путешествия, забота о себе и многое другое. На сайте можно найти статьи, фотографии, видео и галереи. Сайт доступен на русском языке и имеет мобильную версию.

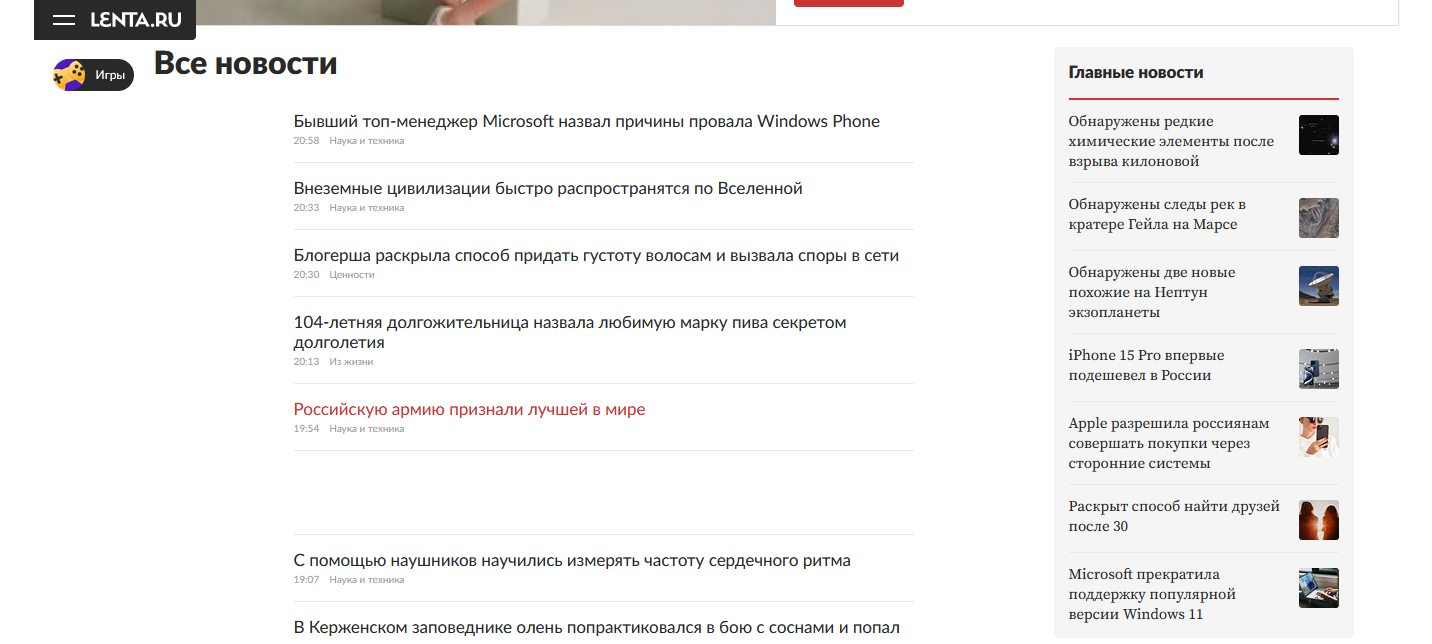
Рассмотрим данный аналог на некие признаки сходства с предыдущими, и выделим его особенности и недостатки, дабы на их основе составить план проектирования дизайн продуктв.

Рисунок 1.3 – Страница «Все новости» сайта Lenta.ru

Данный сервис имеет структуру списка для вывода новостных заголовков. Элементы списка разделяются нижним подчеркиванием блока, что выглядит довольно стильно и неброско. Справа можно заметить блок с самыми популярными новостями, рассчитанные по количеству прочтений.

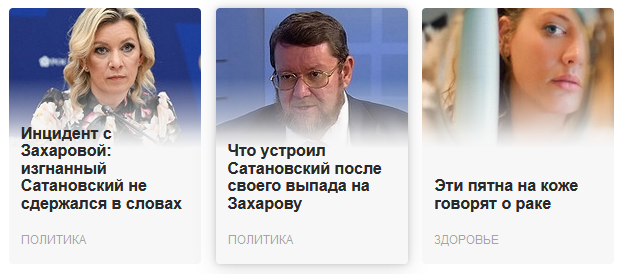
Каждая статья имеет в конце блок с карточками, которые указывают на другие статьи, что призывает пользователя прочитать ещё одну статью, а не сразу покинуть ресурс.

Рисунок 1.4 – Блок с дополнительными новостными заголовками

Наличие такого блока оправдывает себя, но только в том случае, если этот блок содержит информацию по такой же теме, на которую была написана основная статья. В ином случае, такие новостные заголовки могут быть пропущены пользователями по причине того, что в данный момент их они не инетересуют. Это стоит учесть при проектировании дизайна.

## 1.2. Разработка функциональных требований, определение вариантов использования

Важным этапом разработки проекта является изучение требований и определение возможных вариантов использования. Требования к проекту могут быть как функциональными, то есть связанными с основными функциями проекта, так и нефункциональными, то есть связанными с качественными характеристиками проекта.

1.2.1. Разработка требований к базе данных

Новостной портал – это довольно комплексный проект, который состоит из большого количества различных частей. Здесь можно выделить следующие этапы, которые будут присущи базе данных проекта:

* хранение информации о пользователях (данные авторизации, никнейм);
* хранение информации о статьях (автор, дата публикации, заголовок, основная информация, категория, теги и т.п.);
* хранение информации о категориях и тегах;
* создание основных ролей базы данных (пользователь, адинистратор);
* обеспечение защиты критически важных данных (пароли пользователей);
* база данных должна иметь высокую производительность и обеспечивать быстрый доступ к данным для обеспечения эффективной работы новостного портала;
* база данных должна быть удобной и простой в использовании для обеспечения эффективной работы пользователей.

1.2.2. Определение вариантов использования

Основными вариантами использования можно выделить следующие пункты:

* авторизация и регистрация пользователей;
* публикация статьи авторами;
* поиск статей с возможность фильтрации и сортировки;
* просмотр списка статей;
* просмотр определённой статьи;
* возможность оставлять комментарии;
* управление ресурсом со стороны администратора.

## 1.3. Вывод

Аналитический анализ дал некое понимание того, как следует реализовать базу данных и различные объекты новостного блога. Стилевое решение должно быть оформлено в виде сеточной структуры (либо в виде списка) с блоками большого размера, но не очень сильно. Страницы должны иметь некое свободное пространство для визуального разделения элементов управления. Блог должен иметь простой дизайн основных элементов страниц, дабы предоставить лучший опыт использования продуктом.

Каждая статья должна быть выполнена в качества объекта, со своими свойствами, которые будут харктеризировать данный объект. Это будет удобным способом реализации основных списков статей и страницы статьи.

# 2. Проектирование архитектуры проекта

## 2.1. Модель базы данных

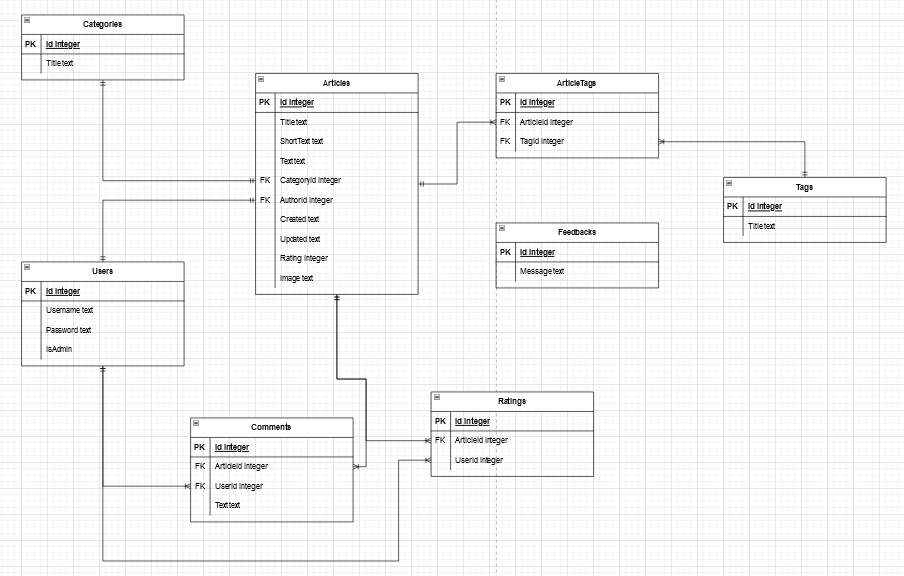
Для хранения и обработки данных была спроектирована база данных NEWS\_BLOG. База данных насчитывает 8 таблиц, каждая из которых играет свою ключевую роль в проекте. Такого количества таблиц достаточно для минимального функционала новостного блога. Диграмма сущнойстей базы данных представлена на рисунке 2.1 и в приложении А.

Рисунок 2.1 – Диаграмма сущностей базы данных

2.1.1. Таблица Users

Данная таблица используется для хранения данных авторизации пользователей.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждого пользователя;
* столбец Username. Хранит никнейм пользователя;
* столбец Password. Хранит пароль пользователя;
* столбец IsAdmin. Определитель того, администратор ли пользователь.

2.1.2. Таблица Tags

Данная таблица используется для хранения данных о тегах.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждого тега;
* столбец Title. Хранит заголовок тега.

2.1.3. Таблица Categories

Данная таблица используется для хранения данных о категориях.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждой категории;
* столбец Username. Хранит заголовок категории;

2.1.4. Таблица Articles

Данная таблица используется для хранения данных о статьях.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждой статьи;
* столбец Title. Хранит заголовок статьи;
* столбец ShortText. Хранит приветственный текст статьи;
* столбец Text. Хранит основной текст статьи;
* столбец AuthorId. Хранит уникальный идентификатор пользователя, который является автором статьи;
* столбец CategoryId. Хранит уникальный идентификатор категории;
* столбец Created. Хранит дату создания статьи;
* столбец Updated. Хранит дату обновления статьи;
* столбец Rating. Хранит значение рейтинга статьи;
* столбец Image. Хранит имя постера статьи.

2.1.5. Таблица ArticleTags

Данная таблица используется для хранения данных о связке статей и тегов.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждой связки;
* столбец ArticleId. Хранит уникальный идентификатор статьи;
* столбец TagId. Хранит уникальный идентификатор тега.

2.1.6. Таблица Comments

Данная таблица используется для хранения данных о комментариях, оставленных пользователями к статьям.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор комментария;
* столбец UserId. Хранит уникальный идентификатор пользователя;
* столбец ArticleId. Хранит уникальный идентификатор статьи;
* столбец Text. Хранит текст комментария.

2.1.7. Таблица Feedbacks

Данная таблица используется для хранения данных о предложениях и замечаниях от пользователей.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор записи;
* столбец Message. Хранит текст предложений и замечаний.

2.1.8. Таблица Ratings

Данная таблица используется для хранения данных о том, кто оценивал статью.

В её состав входят следующие столбцы:

* столбец Id. Хранит уникальный идентификатор каждой записи;
* столбец ArticleId. Хранит уникальный идентификатор статьи;
* столбец UserId. Хранит уникальный идентификатор пользователя.

## 2.2. Структура проекта

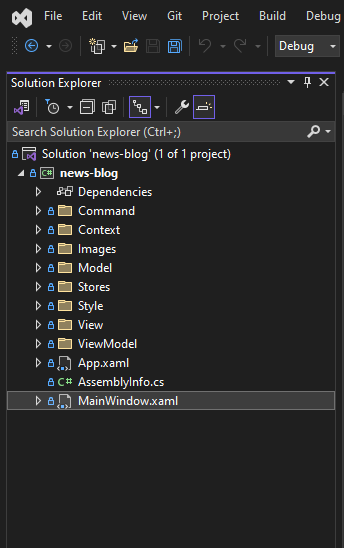
В ходе анализа и проектирования, была определена следующая структура новостного блога, которая характеризует минимальный набор инструментов для работы портала:

Рисунок 2.1 ­– Структура проекта новостного блога

Каждая хранит в себе следующие данные:

* папка Command. Хранит в себе классы, реализующие методы для Command паттерна;
* папка Context. Хранит в себе контекст данных БД, которое использует приложение;
* папка Images. Хранит в себе статические картинки и постеры статей;
* папка Model. Хранит классы, которые описывают сущности БД;
* папка Style. Хранит в себе словари ресурсов, стили для элементов;
* папка View. Хранит в себе представления и окна страниц;
* папка ViewModel. Хранит в себе ViewModel классы.

## 2.3. Основные классы бизнес логики

Рассмотрим конкретно классы моделей, контекста и обработчика данных, ибо основная бизнес логика блога описана в них.

Модели портала представлены следующими классами:

* класс User. Имеет свойства, которые хранят данные записи из таблицы Users;
* класс Tag. Имеет свойства, которые хранят данные записи из таблицы Tags;
* класс Category. Имеет свойства, которые хранят данные записи из таблицы Categories;
* класс Article. Имеет свойства, которые хранят данные записи из таблицы Articles;
* класс ArticleTag. Имеет свойства, которые хранят данные данные записи из таблицы ArticleTags;
* класс Comment. Имеет свойства, которые хранят данные данные записи из таблицы Comments;
* класс Rating. Имеет свойства, которые хранят данные данные записи из таблицы Ratings;
* класс Feedback. Имеет свойства, которые хранят данные данные записи из таблицы Feedbacks;

Класс контекста хранит в себе списки всех данных таблиц базы данных.

Класс DataWorker реализует функционал создания, удаления, редактирования и получения данных из БД.

Схема диаграммы классов приведена в приложении Б.

## 2.4. Работа с базой данных

При разработке приложения использовался Entity Framework 7. Клиент взаимодействует с сервисом посредством вызова методов сервиса. Если база данных отсутствует, то приложение автоматически её создаст и заполнит некоторыми данными, дабы начать полноценную работу с проектом.

## 2.5. Вывод

Первым делом спроектирована база данных, ибо она определяет то, какие данные будут храниться и обрабатываться. На основе выбора базы данных и её структуры был установлен пакет Entity Framework 7 для связи проекта с самой базой данных.

Каждая запись в таблицах должна иметь возможность просматриваться, редактироваться и удаляться. Следовательно, был создан общий класс для всех этих задач. В нём определены основные методы для работы с базой данных и её сущностями.

# 3. Разработка функциональной модели и модели данных ПС

## 3.1. Модели данных

Кодовая реализиция модели данных User:

namespace news\_blog.Model

{

public class User

{

public int Id { get; set; }

public string? Username { get; set; }

public string? Password { get; set; }

public int IsAdmin { get; set; }

}

}

Листинг 3.1 – Класс модели данных User

Кодовая реализация модели данных Tag:

namespace news\_blog.Model

{

public class Tag

{

public int Id { get; set; }

public string? Title { get; set; }

}

}

Листинг 3.2 – Класс модели данных Tag

Кодовая реализация модели данных Category:

namespace news\_blog.Model

{

public class Category

{

public int Id { get; set; }

public string? Title { get; set; }

}

}

Листинг 3.3 – Класс модели данных Category

Кодовая реализация модели данных Article:

using System;

namespace news\_blog.Model

{

public class Article

{

public int Id { get; set; }

public string? Title { get; set; }

public string? ShortText { get; set; }

public string? Text { get; set; }

public int AuthorId { get; set; }

public int CategoryId { get; set; }

public DateTime Created { get; set; }

public DateTime Updated { get; set; }

public int Rating { get; set; }

public string? Image { get; set; }

}

}

Листинг 3.4 – Класс модели данных Article

Кодовая реализация модели данных ArticleTag:

namespace news\_blog.Model

{

public class ArticleTag

{

public int Id { get; set; }

public int ArticleId { get; set; }

public int TagId { get; set; }

}

}

Листинг 3.5 – Класс модели данных ArticleTag

Кодовая реализация модели данных Comment:

namespace news\_blog.Model

{

public class Comment {

public int Id { get; set; }

public int UserId { get; set; }

public int ArticleId { get; set; }

public string? Text { get; set; }}

}

Листинг 3.6 – Класс модели данных Comment

Кодовая реализация модели данных Rating:

namespace news\_blog.Model

{

public class Rating

{

public int Id { get; set; }

public int UserId { get; set; }

public int ArticleId { get; set; }

}

}

Листинг 3.7 – Класс модели данных Rating

Кодовая реализация модели данных Feedback:

namespace news\_blog.Model

{

public class Feedback

{

public int Id { get; set; }

public string? Message { get; set; }

}

}

Листинг 3.8 – Класс модели данных Feedback

## 3.2. Реализация паттерна MVVM

Model-View-ViewModel (MVVM) - это шаблон проектирования, который помогает разделить бизнес-логику приложения и логику представления от пользовательского интерфейса.

В MVVM есть три основных компонента: модель (Model), представление (View) и представитель (ViewModel). Модель представляет собой данные и бизнес-логику приложения. Представление отвечает за отображение данных на экране. Представитель связывает модель и представление, обеспечивая обмен данными между ними.

Преимущества использования MVVM:

* чистое разделение бизнес-логики и логики представления от пользовательского интерфейса;
* улучшенная тестируемость кода;
* улучшенная тестируемость кода.

Для реализации паттерна MVVM первым делом определяется шаблон класса представителя, от которого буду наследоваться другие классы представители. Кодовая реализация класса представителя ViewModelBase:

using System;

using System.ComponentModel;

namespace news\_blog.ViewModel

{

public class ViewModelBase : INotifyPropertyChanged

{

public event PropertyChangedEventHandler? PropertyChanged;

protected void NotifyPropertyChanged(String propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}

Листинг 3.9 – Шаблон класса представителя ViewModelBase

Теперь проект имеет возможность создавать свои собственные классы ViewModel.

Дабы понять как устроен паттерн MVVM, возьмём для примера механизм авторизации и разберём его на три основных компонента шаблона: представление, класс представитель и модель (описана в прошлом пункте главы).

Чтобы приложение не было ресурсоёмким, сделаем его одностраничным. Для этого в главном окне определим обалсть вывода содержимого представлений в зависимости от того, какой класс представитель используется в данный момент. Код представлен в листинге 3.10.

<Grid Grid.Row="1">

<Grid.Resources>

<DataTemplate DataType="{x:Type frontendViewModels:MainPageVM}">

<frontendViews:MainPage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:LoginPageVM}">

<frontendViews:LoginPage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:RegisterPageVM}">

<frontendViews:RegisterPage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:ArticlePageVM}">

<frontendViews:ArticlePage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:FeedbackPageVM}">

<frontendViews:FeedbackPage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:ProfilePageVM}">

<frontendViews:ProfilePage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:CreateArticlePageVM}">

<frontendViews:CreateArticlePage />

</DataTemplate>

<DataTemplate DataType="{x:Type frontendViewModels:EditArticlePageVM}">

<frontendViews:EditArticlePage />

</DataTemplate>

</Grid.Resources>

<ContentControl Content="{Binding CurrentViewModel}"/>

</Grid>

Листинг 3.10 – Вывод содрежимого в зависимоти от класса представителя

Чтобы главное окно приложения «понимало» какой класс представитель используется в данный момент, создадим класс NavigationStore, который будет хранить в себе значение текущего класса представителя. Код представлен в листинге 3.11.

using news\_blog.ViewModel;

using System;

namespace news\_blog.Stores

{

public class NavigationStore

{

public event Action? CurrentViewModelChanged;

private ViewModelBase? \_currentViewModel;

public ViewModelBase CurrentViewModel

{

get => \_currentViewModel!;

set

{

\_currentViewModel = value;

OnCurrentViewModelChanged();

}

}

private void OnCurrentViewModelChanged()

{

CurrentViewModelChanged?.Invoke();

}

}

}

Листинг 3.11 – Класс NavigationStore

Создадим похожий класс UserStore, чтобы хранить данные об авторизованном пользователе. Код представлен в листинге 3.12.

using news\_blog.Model;

using System;

namespace news\_blog.Stores

{

public class UserStore

{

public event Action? CurrentUserChanged;

private User? \_currentUser;

public User? CurrentUser

{

get => \_currentUser!;

set

{

\_currentUser = value;

OnCurrentUserChanged();

}

}

private void OnCurrentUserChanged()

{

CurrentUserChanged?.Invoke();

}

}

}

Листинг 3.12 – Класс UserStore

Теперь создадис класс представитель для главного окна. Класс будет хранить в себе данные о текущем авторизованном пользователе (объект UserStore класса), а также значение текущего класса представителя (объект NavigationStore). Код представлен в листинге 3.13.

using news\_blog.Model;

using news\_blog.Stores;

namespace news\_blog.ViewModel.Frontend

{

public class MainWindowVM : ViewModelBase

{

private readonly NavigationStore \_navigationStore;

private readonly UserStore \_userStore;

public ViewModelBase CurrentViewModel => \_navigationStore.CurrentViewModel;

public User CurrentUser => \_userStore.CurrentUser!;

public MainWindowVM(NavigationStore navigationStore, UserStore userStore)

{

\_navigationStore = navigationStore;

\_userStore = userStore;

DataWorker.ClearUnusedImages();

\_navigationStore.CurrentViewModelChanged += OnCurrentViewModelChanged;

\_userStore.CurrentUserChanged += OnCurrentUserChanged;

}

private void OnCurrentViewModelChanged() => NotifyPropertyChanged(nameof(CurrentViewModel));

private void OnCurrentUserChanged() => NotifyPropertyChanged(nameof(CurrentUser));

}

}

Листинг 3.13 – Класс MainWindowVM

Теперь определим представление страницы авторизации. Для этого создадим новый xaml файл, в котором определим основные элементы управления, с которыми будет взаимодействовать наш пользователь. Код представлен в приложении В.

Страница авторизации также должна иметь свой класс представитель, чтобы получать данные пользовательского ввода. Он также будет хранить в себе данные о текущем классе представителе. Код класса представлен в приложении Г.

Теперь для того, чтобы приложение отображало страницу авторизации и могло получать данные пользовательского ввода, определим начальный класс представитель при загрузке приложения.

using Microsoft.EntityFrameworkCore;

using news\_blog.Context;

using news\_blog.Stores;

using news\_blog.ViewModel.Frontend;

using System.Windows;

namespace news\_blog

{

public partial class App : Application

{

private readonly NavigationStore \_navigationStore;

private readonly UserStore \_userStore;

public App()

{

\_navigationStore = new();

\_userStore = new();

using (var db = new ApplicationContext())

{

db.Database.Migrate();

}

}

protected override void OnStartup(StartupEventArgs e)

{

\_navigationStore.CurrentViewModel = new LoginPageVM(\_navigationStore, \_userStore);

\_userStore.CurrentUser = null;

MainWindow = new MainWindow()

{

DataContext = new MainWindowVM(\_navigationStore, \_userStore)

};

MainWindow.Show();

base.OnStartup(e);

}

}

}

Листинг 3.16 – Определение вывода страницы авторизации

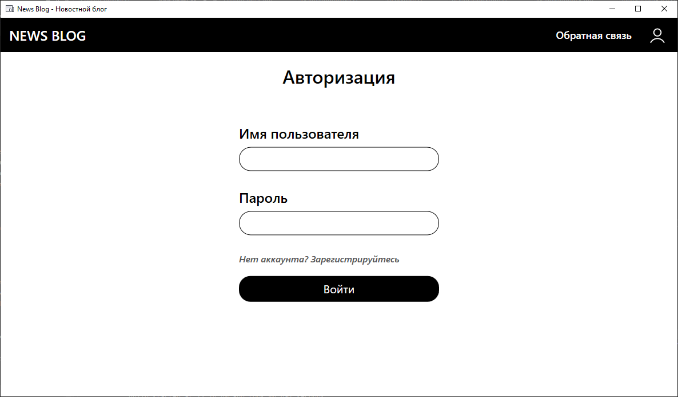
Теперь при загрузке приложения пользователю будет отображена страница авторизации, в которой уже есть связь с классом представителем для того, чтобы получать данные пользовательского ввода.

Рисунок 3.1 – Страница авторизации пользователя

Страница авторизации выполнена по всем правилам паттерна MVVM, отображается конкретно, а данные связаны со свойствами, определёнными в классе LoginPageVM. В результате можно увидеть, что страница отображается корректно. Но механизм авторизации ещё не закончен. Для этого следует определить ещё один паттерн в приложении.

## 3.3. Реализация паттерна Command

Теперь, чтобы читать данные пользовательского ввода и реализовать авторизацию пользователя, следует определить шаблон класс RelayCommand. Данный класс будет отвечать за то, чтобы выполнялись какие-либо действия при некотором событии.

using System;

using System.Windows.Input;

namespace news\_blog.Command

{

public class RelayCommand : ICommand

{

private Action<object> \_execute;

private Func<object, bool> \_canExecute;

public event EventHandler? CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null!)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object? parameter) => \_canExecute == null || \_canExecute(parameter!);

public void Execute(object? parameter) => \_execute(parameter!);

}

}

Листинг 3.17 – Класс RelayCommand

Далее определим объект класса RelayCommand в классе LoginPageVM, чтобы реализовать механизм авторизации. Данный объект будет читать данные пользовательского ввода, проверять их на корректность и реализовывать авторизацию самого пользователя. Код объекта RelayCommand для авторизации представлен в приложении Д.

Теперь механизм авторизации полностью готов. При вводе пустых данных приложение уведомит пользователя о том, какие поля не были заполнены. Если по значению из поля «Имя пользователя» находится такой и пароли совпадают, приложение заносит в объект UserStore объект пользователя, которого оно получит из базы данных.

## 3.4. Вывод

Приложение имеет все инструменты для того, чтобы реализовывать различные функциональные возможности.

Построение приложения реализвоано с помощью паттерна MVVM, чтобы упростить расширение приложения другими функциями, которые потребуются по ходу анализа темы.

Связка с паттерном Command реализует нам полную отстранённость представления от модели и класса представителя. Выполнение функций срабатывает за счёт того, что к элементу управления привязано названия функции, которая должна сработать при событии, а не сам объект RelayCommand.

# 4. Тестирование

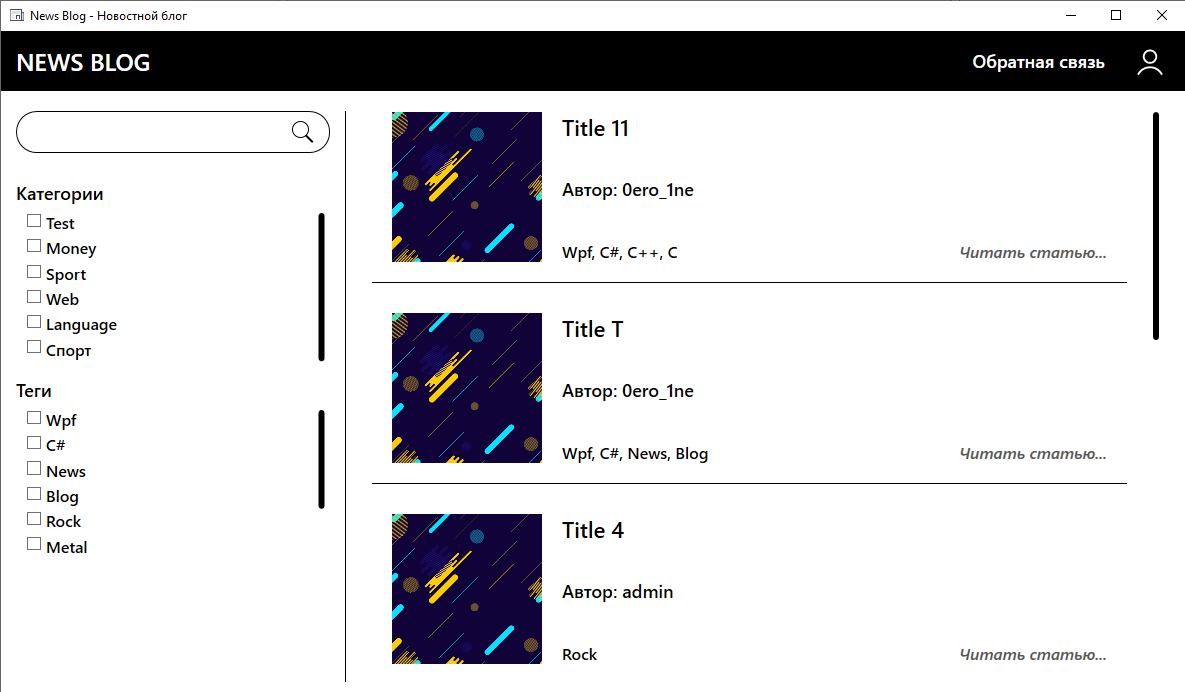
При запуске приложения запускается главная страница приложения, на которой выводится список статей. На это странице пользователь имеет возможность искать определённые статьи по их названию, а также фильтровать контент по значениям категорий и тегов.

Рисунок 4.1 – Главная страница приложения

Проверим возможность фильтрации статей. Выберем теги C# и WPF, а также категорию Test.

Рисунок 4.2 – Фильтрация статей по тегам и категории

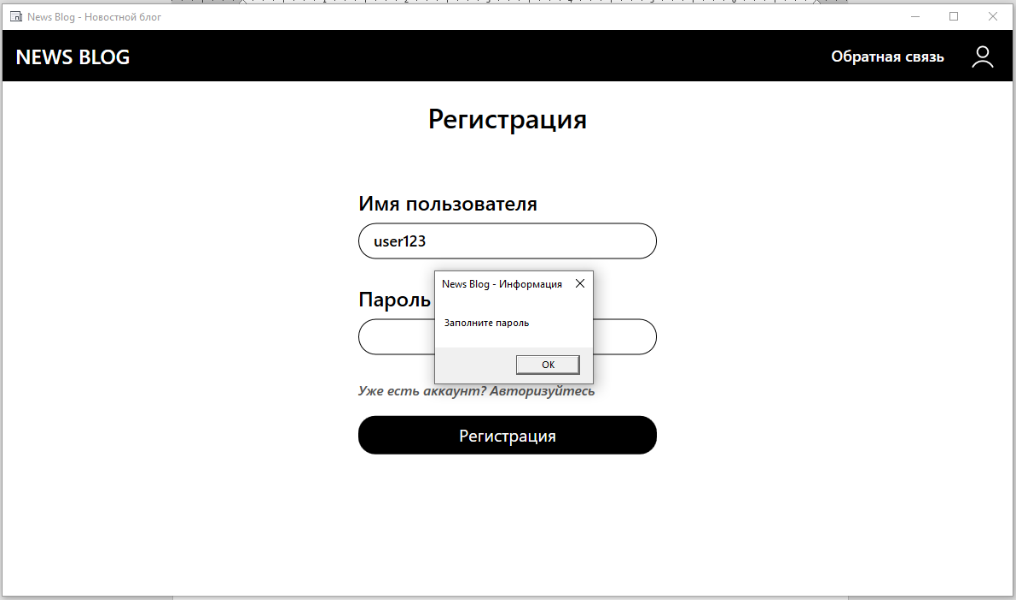
Теперь попробуем авторизоваться в приложении. Для этого создадим новый аккаунт на странице регистрации, на которую можно попасть по ссылке на странице авторизации.

Рисунок 4.3 – Страница регистрации

Как можно заметить, продукт уведомляет пользователя о том, что произошла ошибка и какая именно. Таким образом пользователь понимает, что ему требуется сделать для того, чтобы успешно зарегистрироваться. Введём корректные данные и проверим автоматическую авторизацию и страницу пользователя.

Рисунок 4.4 – Страница профиля пользователя

Регистрация и автоматическая авторизация работают корректно. Пользователь имеет возможность создавать свои собственные статьи, которые будут видны другим таким же пользователям. Чтобы создать статью, была создана отдельная страница, в которой пользователь вводит данные о статье и приложение сохраняет ей.

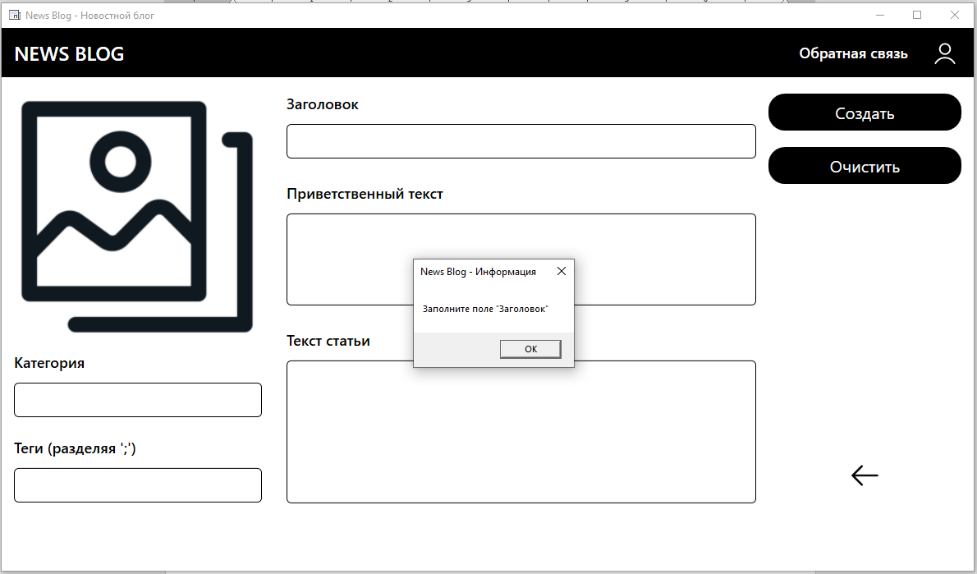
Проверим правильность создания статьи. Заполним требуемые поля, а проверим вывод ошибок в случае неправильно введённых данных.

Рисунок 4.5 – Уведомление об ошибке на странице создания статьи

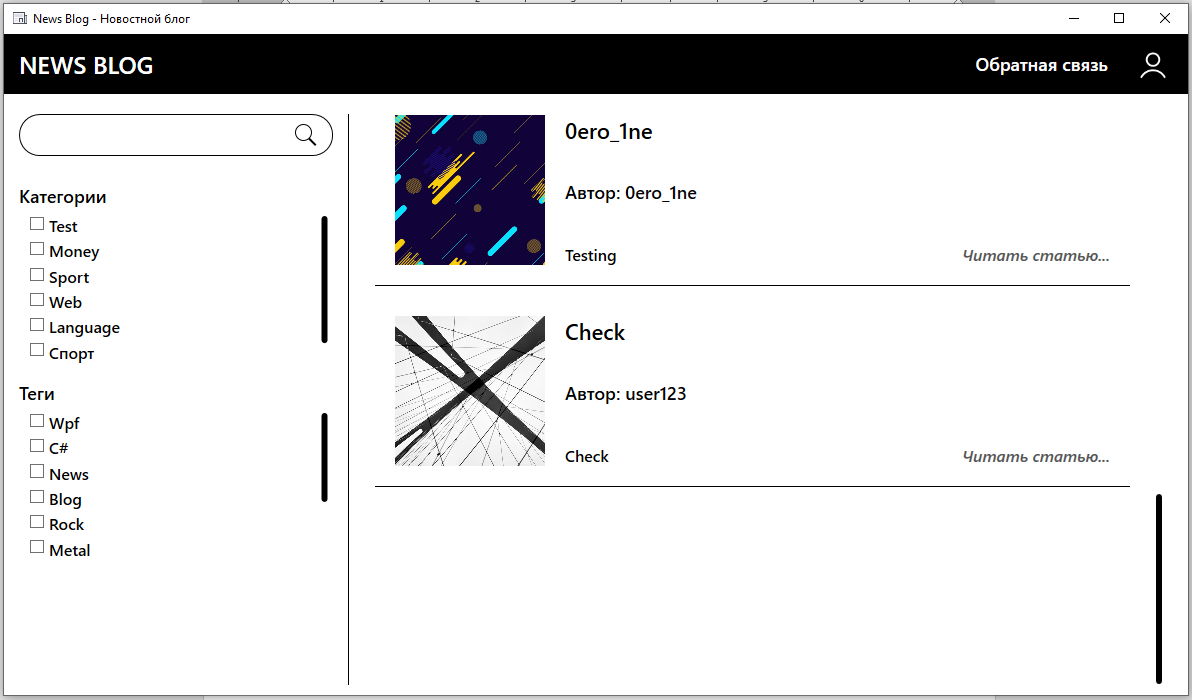
Ошибка обрабатываются корректно, что и требуется от хорошо продуманного приложения. Заполним форму правильными данными и проверим вывод статьи на главной странице приложения.

Рисунок 4.6 – Отображение новой статьи на главной странице

Как можно увидеть, статья успешно сохранилась и отобразилась на главной странице. Теперь проверим следующий функционал:

* оставление комментария под статьёй;
* оценивание статьи другим пользователем (автор статьи не может сам оценивать свою статью).

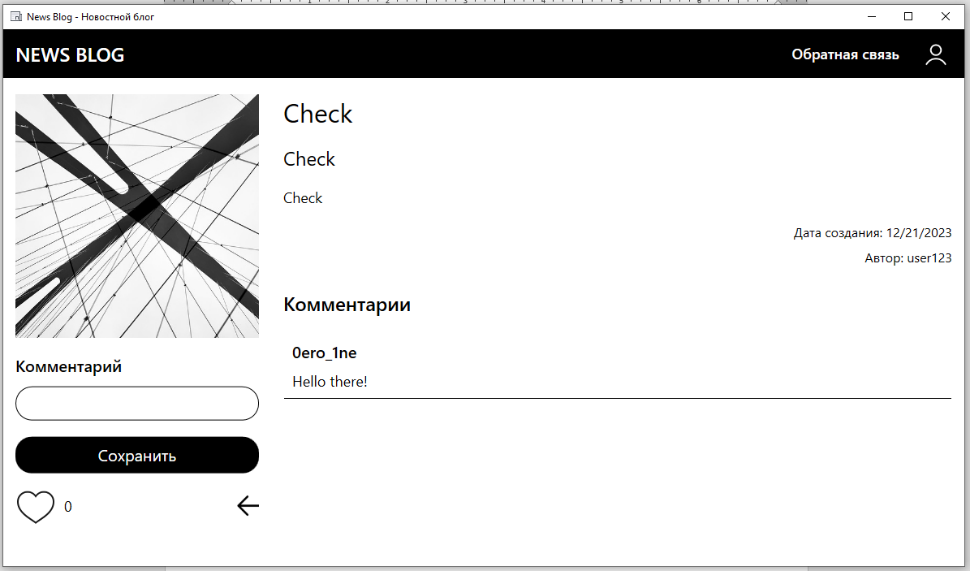
Аторизуемся под другим пользователем, чтобы объективно оценить корректность выше указанных функциональных возможностей.

Рисунок 4.7 – Страница отображения статьи

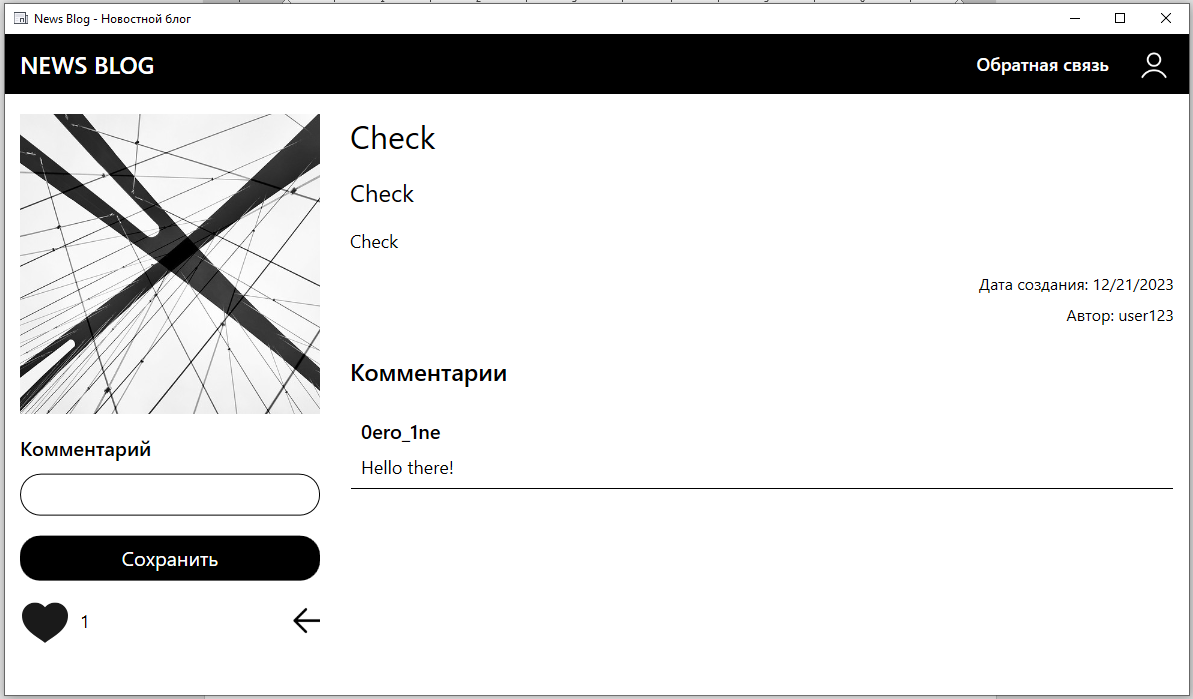
Статья отображается корректно, а также и комментарии, которые были оставлены другим пользователем. Проверим способность оценивать статью.

Рисунок 4.8 – Оценка пользователя на странице статьи

Всё работает корректно, без всяческих ошибок. Комментарии отображаются корректно, а другие пользователи имеют возможность оценивать статьи.

## 4.1. Вывод

Весь реализованный функционал работает корректно, обрабатывает ошибки и уведомляет пользователя о том, что произошла исключительная ситуация. Пользователь видит сообщение ошибки, что даёт ему возможность понять причину ошибки, а затем исправить её для дальнейшего пользования приложением.

# Заключение

Тематика новостей довольно популярная сегодня. Каждый день люди получают порцию новостей, и даже неважно каким способом (газета, телевизор или интернет). Так человек знает актуальные данные по определённым событиям, темам и т.п. Однако следует учесть, что не вся информация является гарантировано правдивой.

Создание новостного блога нельзя назвать тяжёлой задачей, если начать делать её правильно. Для этого стоит изучить пару аналогов и понять, что все они имеют много сходств. А если имеется много сходств, значит уже есть некое готовое решение открытое решения для его изменения и расширения.

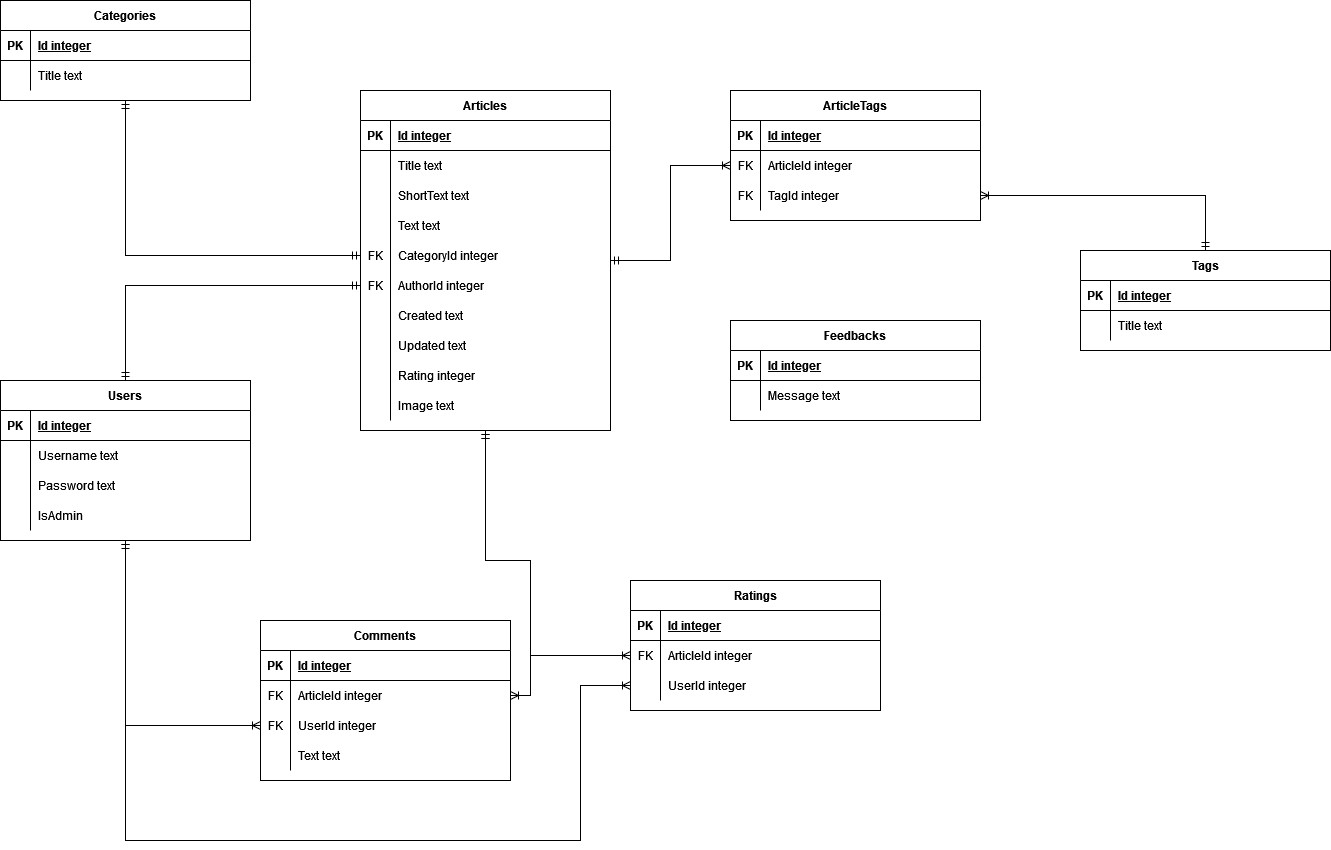
Чтобы написать хорошее и расширяемое приложение, используя технологию WPF, следует реализовать паттерн MVVM, который даёт нам строгие ограничения при построении продукта. Представление не должно иметь прямого доступа к данным. Для этого есть компонент, который является посредником между данными и представлением. Паттерн MVVM лучше реализовывать в связке с шаблоном Command, который даёт нам ещё более строгие ограничения для представлений.

Дизайн также играет важную роль. Пользователь не захочет использовать приложение, если оно построено так, что сам пользователь не сможет понять как и что ему делать. Дабы избежать такой ситуации, разработчик должен изучить дизайнерские решения аналогов, подчеркнуть сильны и слабые стороны, и уже с учётом этого проектировать приложение.

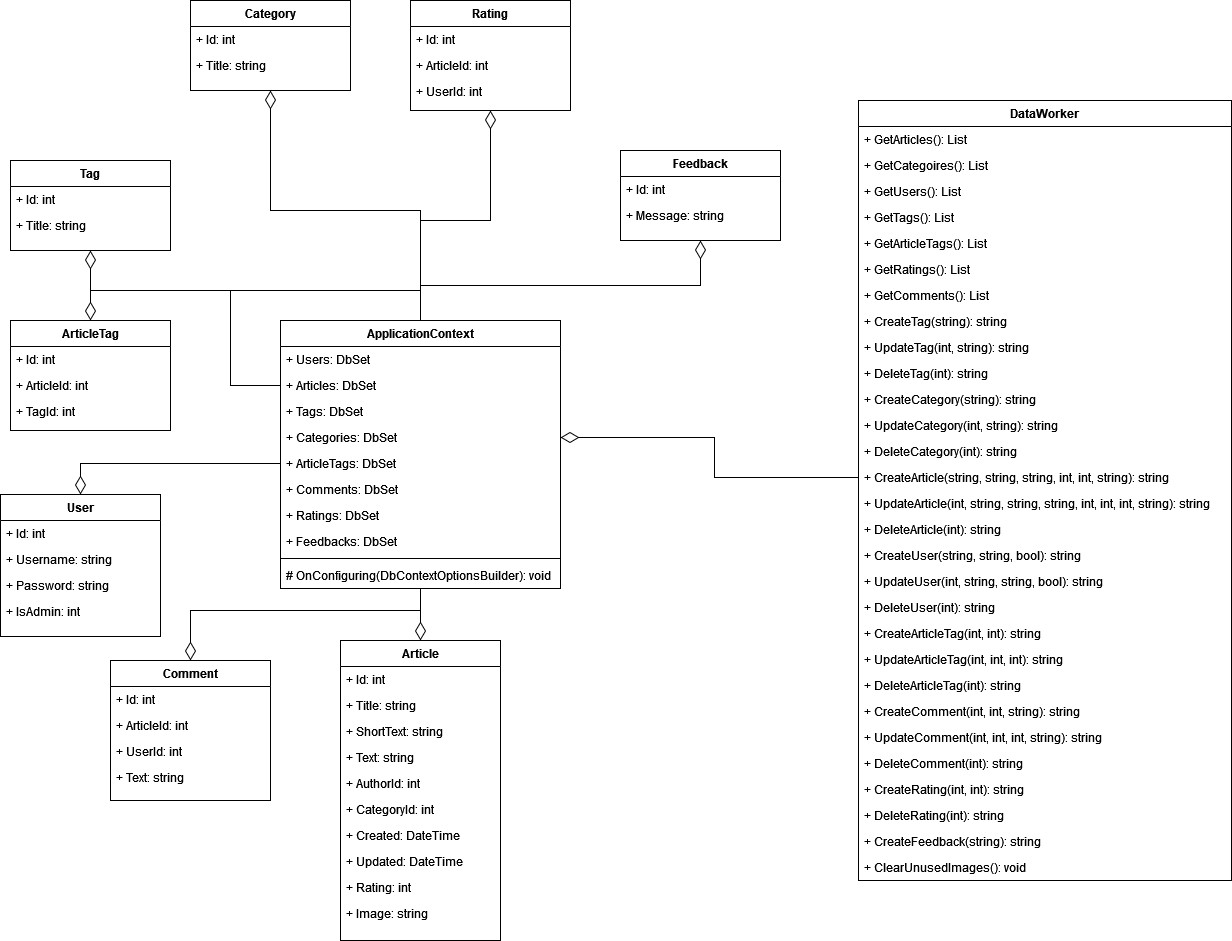
# Список используемых источников

1. Sputnik.by – [Электронный адрес] / Режим доступа: https://sputnik.by/ – Дата доступа: 10.10.2023
2. Onliner – [Электронный адрес] / Режим доступа: https://www.onliner.by/ – Дата доступа: 10.10.2023
3. Lenta.ru – [Электронный адрес] / Режим доступа: https://lenta.ru/ – Дата доступа: 10.10.2023
4. METANIT.COM – [Электронный адрес] / Режим доступа: https://metanit.com/ – Дата доступа: 11.11.2023
5. Professor Web – [Электронный адрес] / Режим доступа: https://professorweb.ru/my/WPF/base\_WPF/level1/base\_WPF\_index.php – Дата доступа: 12.11.2023
6. Q&A Habr – [Электронный адрес] / Режим доступа: https://qna.habr.com/ – Дата доступа: 13.11.2023
7. Форум StackOverflow – [Электронный ресурс] / Режим доступа: https://ru.stackoverflow.com/ – Дата доступа 02.10.2023.

# Приложение А Диаграмма сущностей базы данных



# Приложение Б Диаграмма классов



# Приложение В Листинг LoginPage.xaml

<UserControl x:Class="news\_blog.View.Frontend.LoginPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:news\_blog.View.Frontend"

mc:Ignorable="d"

d:DesignHeight="640" d:DesignWidth="1200"

Background="White">

<Grid

Margin="15,20,15,20">

<StackPanel

Orientation="Vertical">

<Label

Content="Авторизация"

Style="{DynamicResource FormLabel}"

Margin="0,0,0,60"

FontSize="32"

HorizontalContentAlignment="Center"/>

<StackPanel

Orientation="Vertical"

Width="Auto"

HorizontalAlignment="Center">

<Label

Content="Имя пользователя"

Style="{DynamicResource FormLabel}"

Margin="0,0,0,5"

FontSize="24"/>

<Border

Style="{DynamicResource SearchBorder}"

Width="350"

HorizontalAlignment="Left"

Margin="0,0,0,30">

<TextBox

Text="{Binding Username}"

Style="{DynamicResource SearchTextBox}"/>

</Border>

<Label

Content="Пароль"

Style="{DynamicResource FormLabel}"

Margin="0,0,0,5"

FontSize="24"/>

<Border

Style="{DynamicResource SearchBorder}"

Width="350"

HorizontalAlignment="Left"

Margin="0,0,0,30">

<PasswordBox

Style="{DynamicResource PasswordBox}"

PasswordChanged="PasswordChangedEvent"/>

</Border>

<TextBlock

Text="Нет аккаунта? Зарегистрируйтесь"

Style="{DynamicResource LoginRegisterLink}">

<TextBlock.InputBindings>

<MouseBinding Command="{Binding OpenRegisterPage}" MouseAction="LeftClick" />

</TextBlock.InputBindings>

</TextBlock>

<Border

Margin="0,20,0,0"

Width="350"

HorizontalAlignment="Left"

Style="{DynamicResource SimpleButtonBorder}">

<Button

Content="Войти"

Width="330"

Style="{DynamicResource SimpleButton}"

Template="{DynamicResource NoMouseOverButtonTemplate}"

Command="{Binding LoginUser}" />

</Border>

</StackPanel>

</StackPanel>

</Grid>

</UserControl>

# Приложение Г Листинг LoginPageVM

using news\_blog.Model;

using news\_blog.Stores;

namespace news\_blog.ViewModel.Frontend

{

public class LoginPageVM : ViewModelBase

{

private readonly NavigationStore \_navigationStore;

private readonly UserStore \_userStore;

public ViewModelBase CurrentViewModel => \_navigationStore.CurrentViewModel;

public User CurrentUser => \_userStore.CurrentUser;

private string? \_username;

public string? Username

{

get => \_username;

set

{

\_username = value!;

NotifyPropertyChanged(nameof(Username));

}

}

private string? \_password;

public string? Password

{

get => \_password;

set

{

\_password = value!;

NotifyPropertyChanged(nameof(Password));

}

}

public LoginPageVM(NavigationStore navigationStore, UserStore userStore)

{

\_navigationStore = navigationStore;

\_userStore = userStore;

}

}

}

# Приложение Д Листинг объекта RelayCommand

private RelayCommand? loginUser;

public RelayCommand? LoginUser

{

get

{

return loginUser ?? new RelayCommand(obj =>

{

if (Username == null || Username == "")

{

MessageBox.Show("Заполните имя пользователя", "News Blog - Информация", MessageBoxButton.OK);

return;

}

if (Password == null || Username == ""){

MessageBox.Show("Заполните пароль", "News Blog - Информация", MessageBoxButton.OK);

return;

}

User user = DataWorker.GetUsers().SingleOrDefault(u => u.Username == Username)!;

if (user == null)

{

MessageBox.Show("Неверные имя пользователя или пароль", "News Blog - Информация", MessageBoxButton.OK);

return;

}

var passwordBytes = Encoding.UTF8.GetBytes(Password!);

var passwordHash = SHA256.HashData(passwordBytes);

var passwordHashString = Convert.ToHexString(passwordHash);

if (user.Password != passwordHashString)

{

MessageBox.Show("Неверные имя пользователя или пароль", "News Blog - Информация", MessageBoxButton.OK);

return;

}

\_userStore.CurrentUser = user;

\_navigationStore.CurrentViewModel = new MainPageVM(\_navigationStore, \_userStore);

});

}

}

# Приложение Е Диаграмма вариантов использования

# Приложение Ж Диаграмма последовательности

