**ДИПЛОМЕН ПРОЕКТ ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ**

по професия код 481030 „Приложен програмист“

специалност код 4810301 Приложно програмиране“

Тема: „Сайт за новинарски публикации“

Автор:

Васил Тодоров Станчев, клас XII В

Ръководител:

Антони Димитров

Бургас, 2023

СЪДЪРЖАНИЕ

[1 Увод 3](#_Toc132965556)

[2 Цели и обхват на софтуерното приложение 3](#_Toc132965557)

[2.1 Цели 3](#_Toc132965558)

[2.2 Обхват 4](#_Toc132965559)

[3 Анализ на решението 4](#_Toc132965560)

[3.1 Потребителски изисквания и работен процес 4](#_Toc132965561)

[3.2 Примерен потребителски интерфейс 6](#_Toc132965562)

[3.3 Диаграми на анализа 8](#_Toc132965563)

[3.3.1 Диаграма на използване на казус 8](#_Toc132965564)

[3.3.2 E/R диаграма 9](#_Toc132965565)

[3.3.3 Диаграма на активност 10](#_Toc132965566)

[3.3.4 Диаграма на последователност 11](#_Toc132965567)

[3.3.5 Диаграма на класовете 14](#_Toc132965568)

[3.4 Модел на съдържанието / данните 16](#_Toc132965569)

[4 Дизайн 19](#_Toc132965570)

[4.1 Реализация на архитектурата на приложението 21](#_Toc132965571)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 22](#_Toc132965572)

[4.2.1 Модели 22](#_Toc132965573)

[4.2.2 Изгледи 24](#_Toc132965574)

[4.2.3 Контролери 24](#_Toc132965575)

[4.3 Организация и код на заявките към база от данни 25](#_Toc132965576)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 25](#_Toc132965577)

[5 Ефективност и бързодействие на решението 26](#_Toc132965578)

[6 Тестване 26](#_Toc132965579)

[6.1 Какво представлява NUnit тестване 26](#_Toc132965580)

[6.2 План на тестване 26](#_Toc132965581)

[6.3 Примерни тестови случаи 27](#_Toc132965582)

[7 Заключение и възможно бъдещо развитие 28](#_Toc132965583)

[7.1 Заключение 28](#_Toc132965584)

[7.2 Бъдещо развитие 29](#_Toc132965585)

[8 Използвани литературни източници и Уеб сайтове 29](#_Toc132965586)

[9 Приложения 29](#_Toc132965587)

[10 Критерии и показатели за оценяване 30](#_Toc132965588)

# Увод

Разработването на информационен сайт за новини е важна задача, която изисква детайлна и задълбочена софтуерна документация. Този вид уебсайт е предназначен да предоставя актуална информация за новините от различни категории като политика, спорт, здраве и други.

Софтуерната документация за този тип уебсайт трябва да бъде ясна, детайлна и да включва всички възможности, които предлага сайтът. В документацията трябва да се описват функционалностите на уебсайта като търсене на новини, филтриране на категории, абонаменти за електронни бюлетини, коментари на статии и други.

Важно е да се описват детайлите на архитектурата на сайта и взаимодействието между различните компоненти на софтуера. Освен това, трябва да се предоставят инструкции за инсталацията и конфигурацията на софтуерната среда, която е необходима за да бъде сайтът пуснат успешно.

С цел по-добро разбиране на уебсайта, документацията може да включва снимки и други визуални елементи.

# Цели и обхват на софтуерното приложение

## Цели

Целта на проекта е направата на сайт за новинарски публикации. Все по често се наблюдава липсата на стойностни сайтове за свободна журналистическа практика. С този сайт се цели промяната на това. Сайтът разполага с регистрация, която цели да разграничава по-заинтересованите потребители, като всички публикации биват разделени на одобрени и неодобрени. Тези публикации могат да бъдат създавани, променяни или изтривани от потребители с ролята редактор, предварително вкарани от администратора на сайта. След това тези публикации биват одобрени и те стават видими за нерегистрирани потребители. Ролята администратор има права над всичко свързано с сайта, включително и разпределянето на потребители предварително одобрени като редактори.

## Обхват

Обхватът на проекта е към абсолютно всички. Сайтът цели да направи медиите една по приятна среда за всички хора като това се цели чрез свободата на публикациите. Всеки е приветстван да се регистрира и да разглежда всички публикации. Сайтът също цели да подтиква млади или опитни журналисти да развиват своя опит и гледна точка, като им дава възможността да изразят важни за тях теми.

# Анализ на решението

## Потребителски изисквания и работен процес

Работният процес е последователност от действия, които се извършват, за да се постигне определен резултат. Този процес може да включва множество стъпки, взаимодействия между хора и системи, както и използване на различни технологии.

Когато създаваме продукт или услуга, трябва да разберем какви са потребителските изисквания и да ги включим в работния процес. Това ни позволява да създадем продукт или услуга, която е по-добра, по-ефективна и по-удобна за потребителите. В сегашния сайт беше наблегнато на няколко важни изисквания:

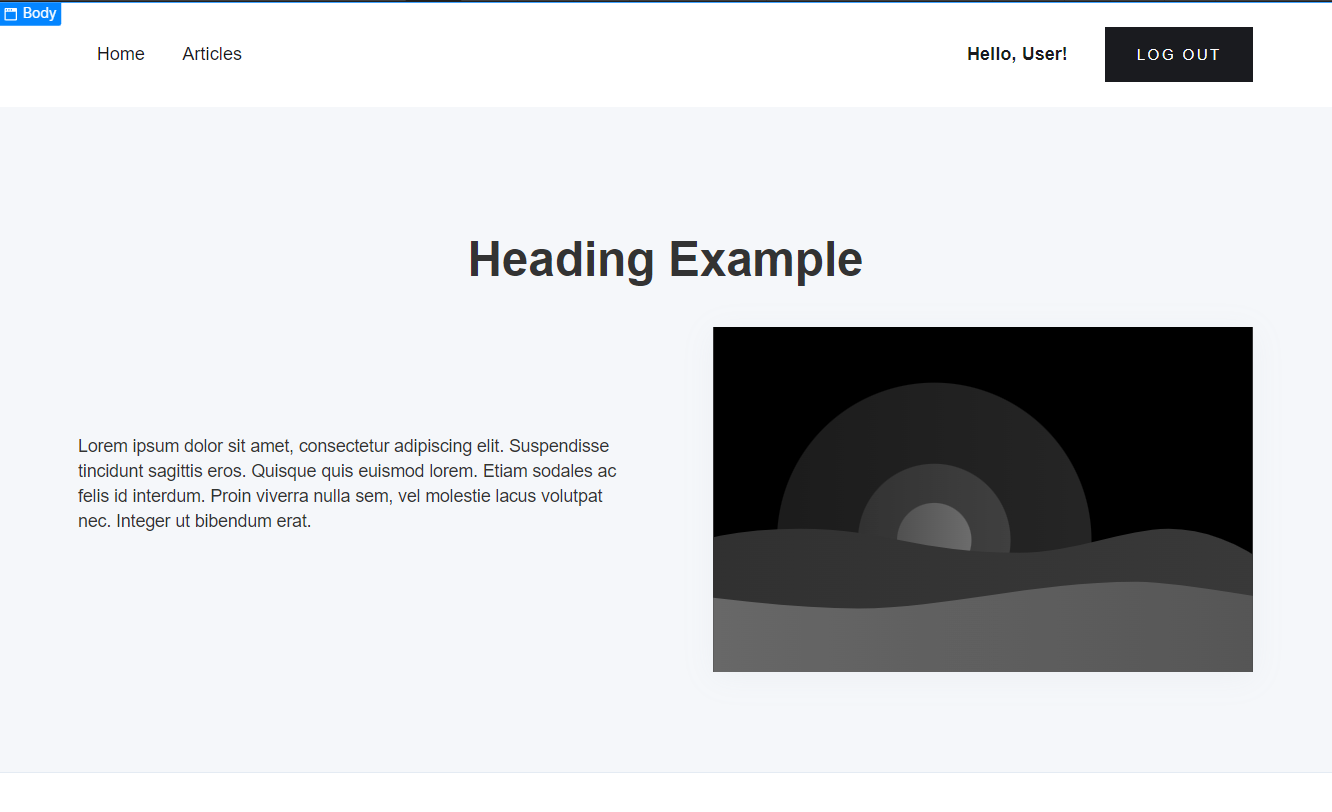
1. Актуалност на новините - потребителите очакват да видят актуална информация за новините от различни категории, като политика, спорт, здраве и други.
2. Лесно търсене и филтриране - потребителите искат да могат лесно да намерят новините, които търсят. Това може да се постигне чрез търсачка, филтриране на новините по категории или други методи.
3. Удобна навигация - потребителите очакват лесна и удобна навигация в сайта, която да им позволява бързо да намерят това, което търсят.
4. Атрактивен дизайн - потребителите искат сайтът да има атрактивен дизайн, който да ги привлича и да им предоставя удобство при ползването.
5. Възможност за коментари и споделяне - потребителите искат да могат да коментират новините и да ги споделят с други потребители.
6. Сигурност на сайта - потребителите очакват, че сайтът ще бъде сигурен и няма да има рискове за техните данни или устройства.

Работният процес за създаване на информационен сайт за новини трябва да включва следните стъпки:

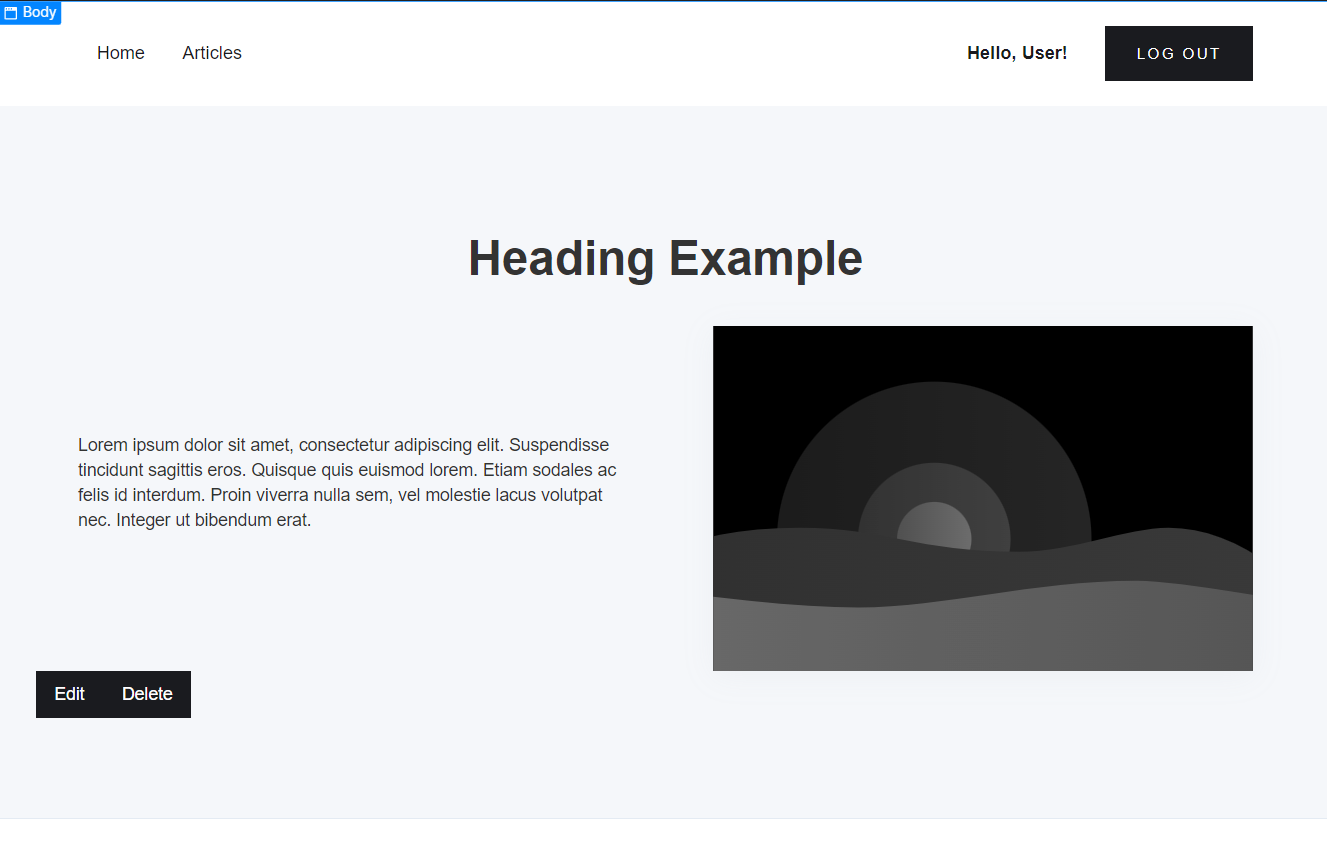
1. Определяне на целевата аудитория и изискванията - разработчиците трябва да определят кой ще бъде целевата аудитория на сайта и да изготвят списък с изискванията на потребителите.
2. Създаване на дизайн - след като са определени изискванията на потребителите, разработчиците трябва да създадат дизайн на сайта, който да отговаря на тези изисквания.

## Примерен потребителски интерфейс

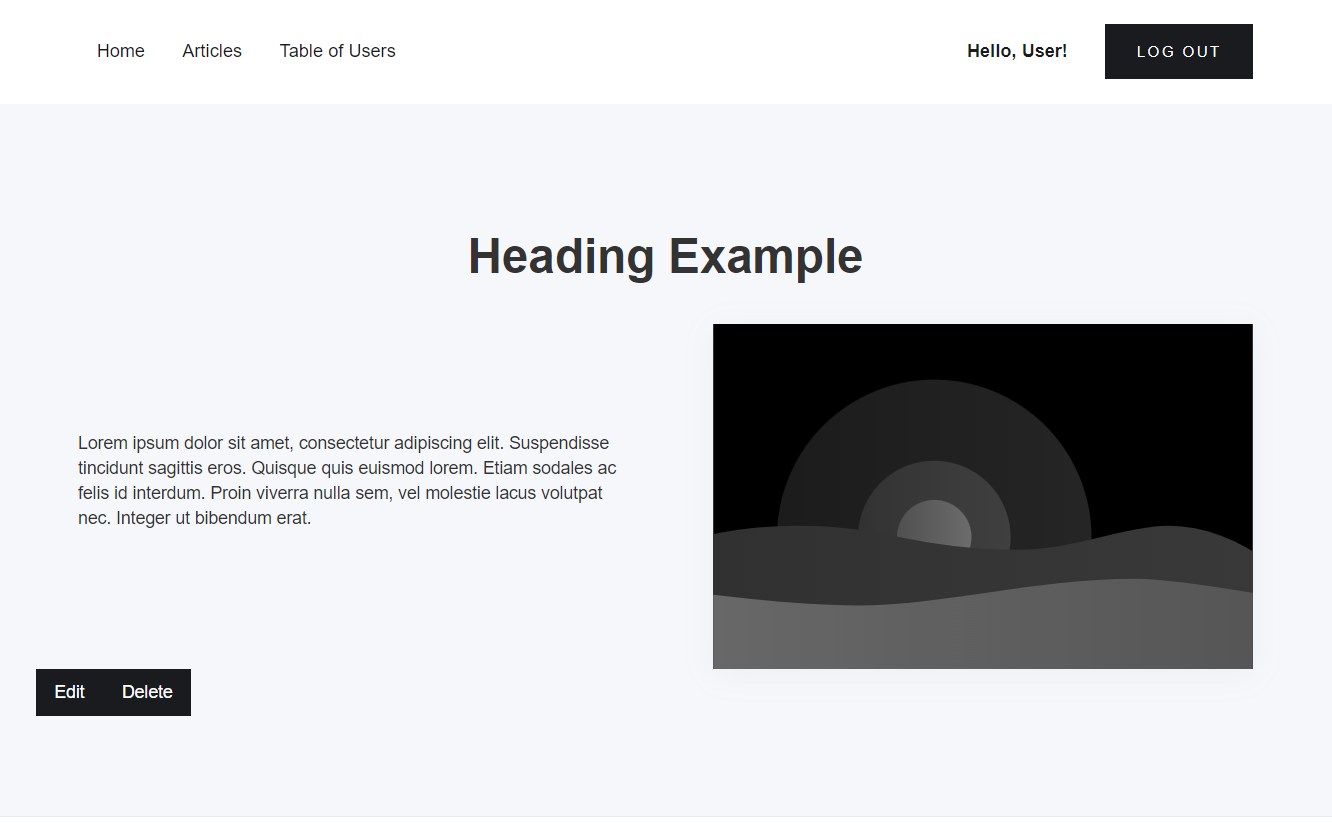
Фигурите 1, 2 и 3 демонстрират примерни потребителски интерфейси за различните групи от потребители. Те са представени в следната последователност: регистриран потребител, редактор и администратор.



Фигура 1



Фигура 2

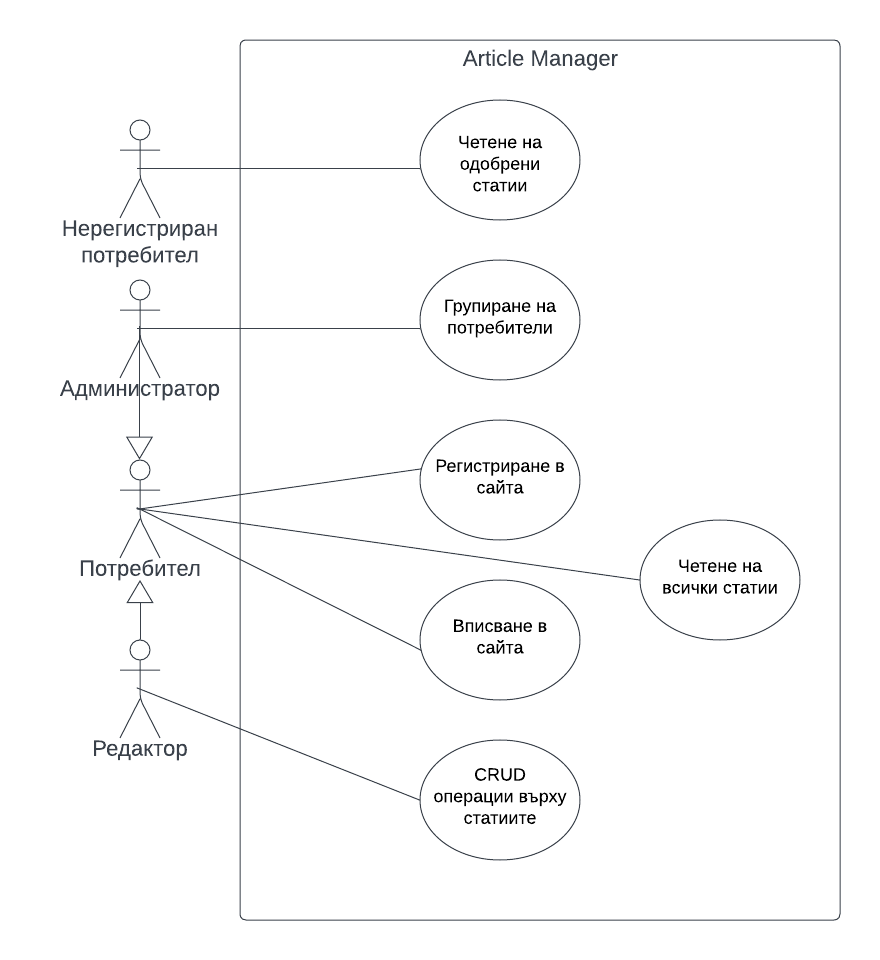


Фигура 3

В примерните потребителски интерфейс са използвани следните цветове: #000000, #FFFFFF и #808080

## Диаграми на анализа

### Диаграма на използване на казус

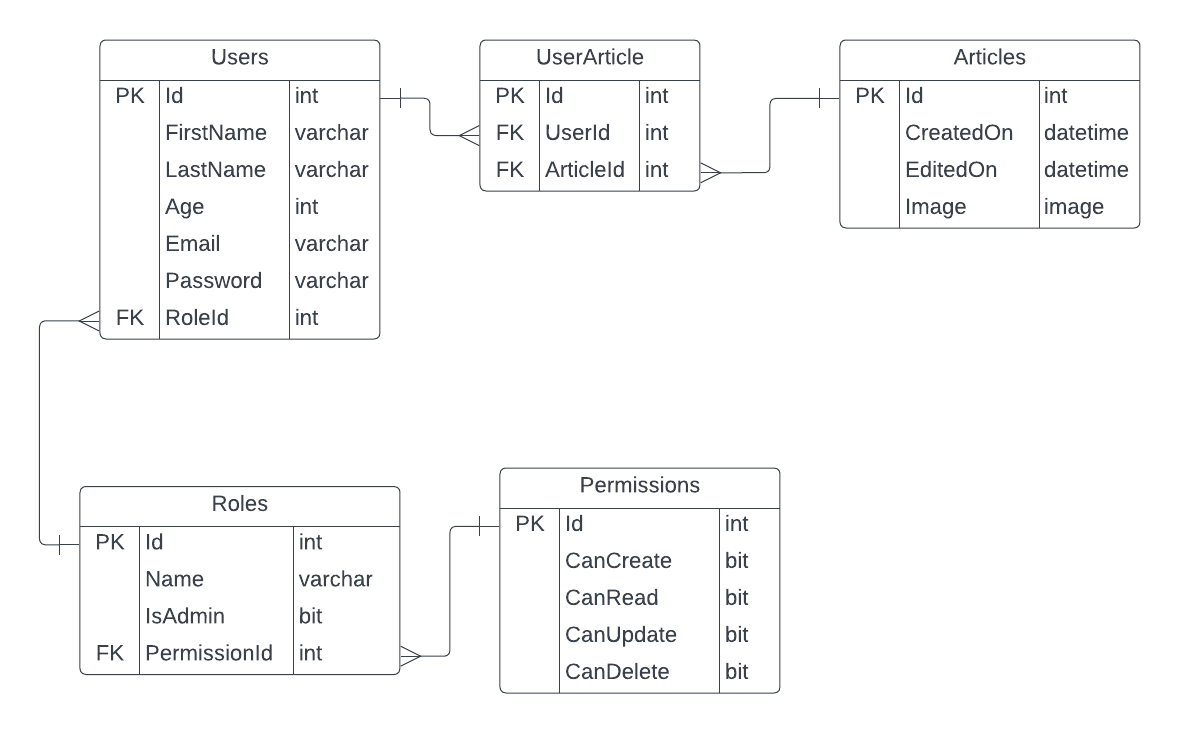


Фигура 4

Фигура 4 демонстрира основните действия на различните групи от потребители. Те са четири:

* Нерегистриран потребител – това е най-простият тип потребител, който без регистрация може да види вече одобрените публикации, като ако желае да види останалите, та той става обикновен потребител.
* Потребител – това са потребители, които желаят да видят по-голяма част от сайта и следователно могат да си направят регистрация или да се впишат ако вече имат. С това те вече имат достъп до абсолютно всички публикувани статии.
* Редактор – това са потребители, предварително одобрени от администратор, които имат права над създаването, редактирането и изтриването на публикации. Редактора разполага също със същите права като обикновен потребител.
* Администратор – това е потребител, който е с най-много правомощия над сайта. Той има права като редактор, като допълнение той е отговорен за групирането на потребителите.

### E/R диаграма

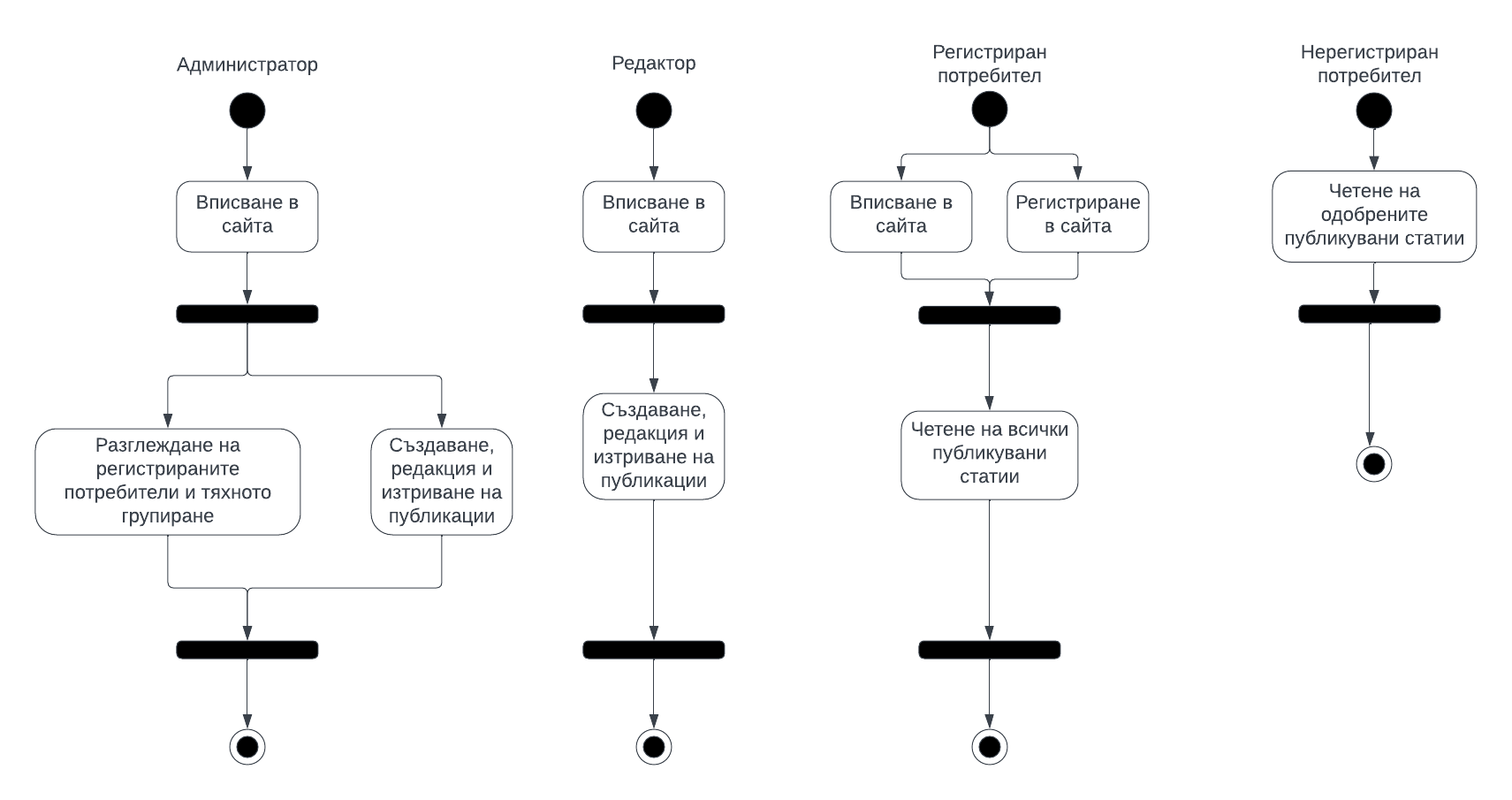


Фигура 5

Фигура 5 демонстрира структурата на базата данни използвана за сайта. Имаме 4 основни таблици и една междинна за връзката между редактор и статията, върху която е работил.

Таблицата свързана със статиите съдържа информация относно датите на промяна и създаване на самата публикация, както и поле за снимка на статията. Таблицата за потребителите съдържа основна информация за всеки от профилите на потребителите като име, години, имейли, пароли и роли. Таблицата за роли съдържа информация за 4-те роли, в които са групирани потребителите. Към тази таблица има помощна таблица, която определя различните права на самите роли.

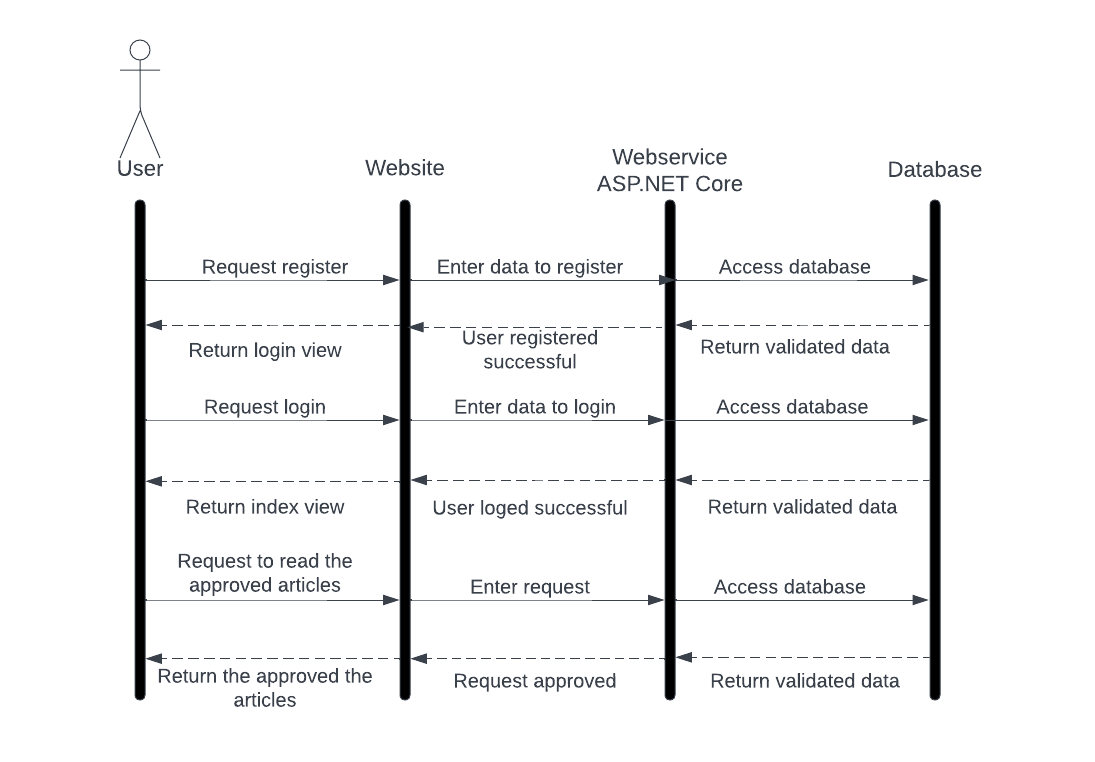
### Диаграма на активност



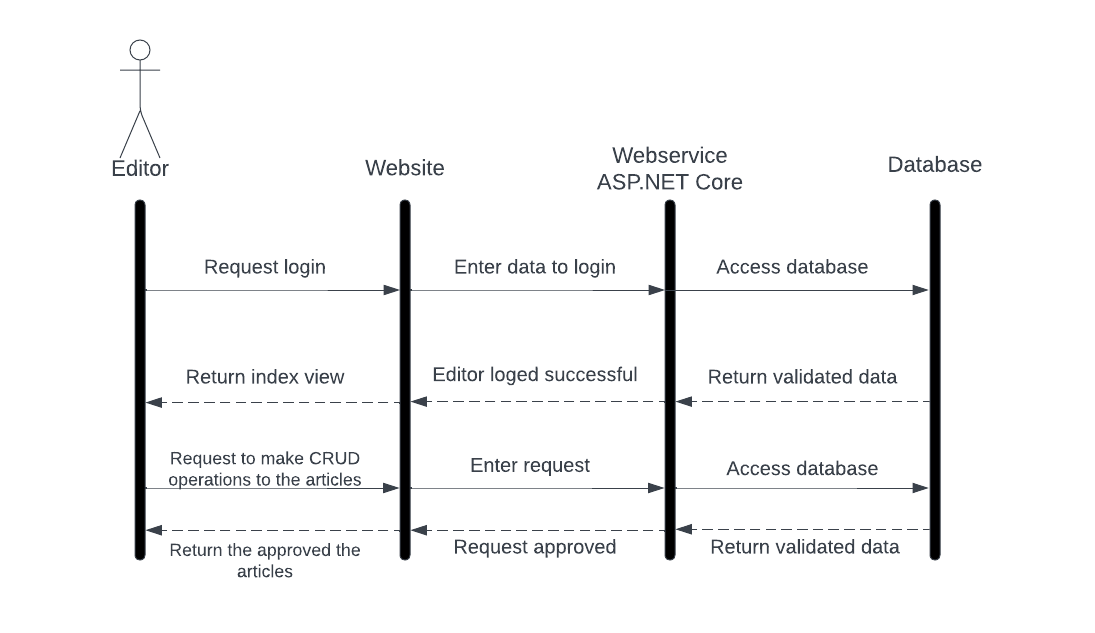
Фигура 6

Фигура 6 демонстрира основните възможни действия на всяка една от четирите групи. Администратора след вписване има възможността да наблюдава в табличен вид всички регистрирани потребители и може да ги разпределя в другите роли, да промени обикновения потребител в редактор. Той също има права над публикациите. Редактора след вписване има права над публикациите. Той може да създава, редактира и изтрива. Регистрирания потребител може да се впише в своя профил или да си направи такъв с регистрация. След което може да разгледа всички създадени публикации. Нерегистрираните потребител може да разгледа одобрените публикации.

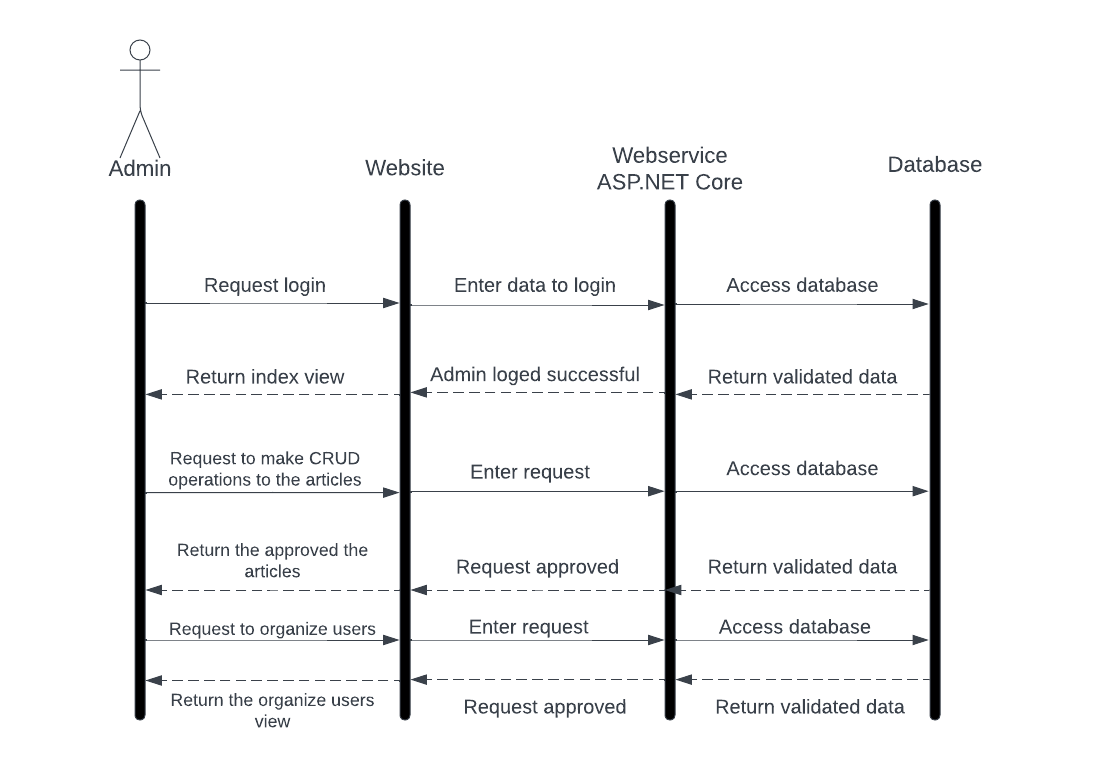
### Диаграма на последователност



Фигура 7



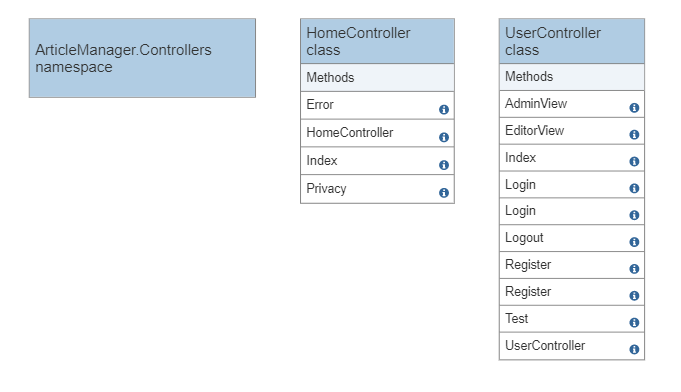
Фигура 8



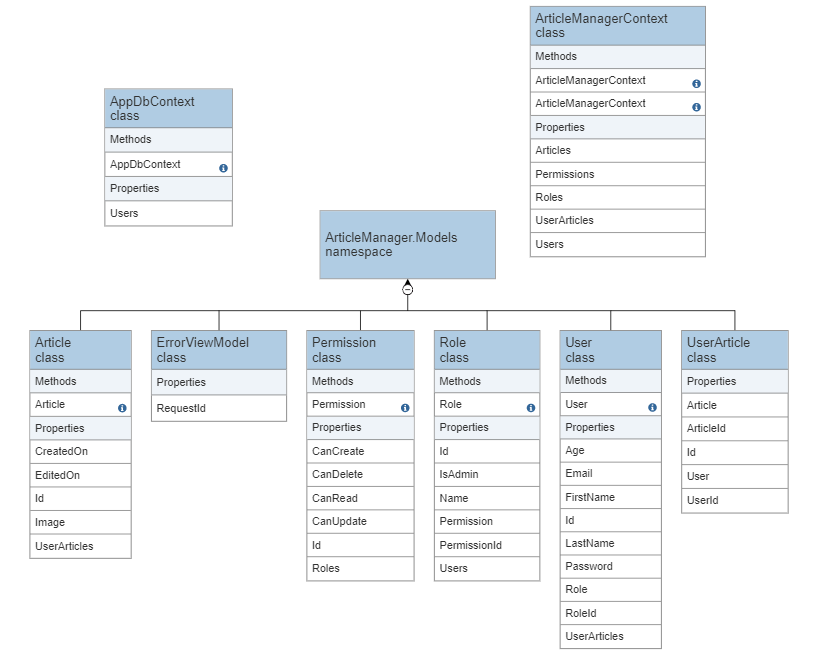
Фигура 9

Фигура 7, 8 и 9 демонстрират последователността на действия за всеки един от различните групи от потребители. Те са представени в следната последователност: регистриран потребител, редактор и администратор.

### Диаграма на класовете



Фигура 10



Фигура 11

## 

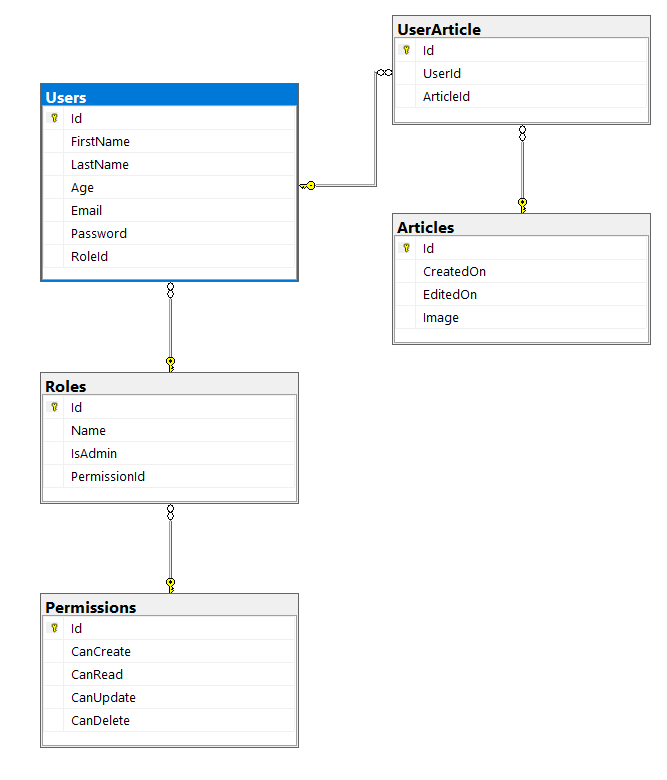
Фигура 10 и 11 демонстрират структурата на класовете в проекта. Фигура 8 демонстрира класовете в компонента „Контролер“, а Фигура 9 класовете в компонента „Модел“.

## Модел на съдържанието / данните

Данните в сайта биват съхранявани в база данни реализирана с технологията “MSSQL”. Microsoft SQL Server (MSSQL) е релационна база данни и беше избрана поради следните причини:

1. Съхранение на данни: “MSSQL” може да съхранява голям обем от данни и да осигури бърз достъп до тях. Това може да бъде особено полезно за проекти, които обработват големи количества информация.
2. Сигурност: “MSSQL” има различни механизми за сигурност, които могат да защитят данните на проекта от неоторизиран достъп.
3. Анализиране на данни: “MSSQL” има множество функции и инструменти за анализиране на данни, като например “OLAP” кубове и мощни средства за търсене. Това може да помогне за извличането на ценна информация от голямо количество данни.
4. Интеграция: “MSSQL” може да бъде лесно интегриран в много различни приложения и системи. Това може да помогне за по-добро сътрудничество и обмен на данни между различни системи.
5. Оптимизация на производителността: “MSSQL” има много опции за оптимизиране на производителността, като например индексиране на данни, кеширане на данни и оптимизиране на заявките. Това може да помогне за постигане на по-добра производителност на проекта.

Фигура 12 демонстрира схема на реализираната база данни в платформата “SSMS”(SQL Server Management Studio).



Фигура 12

Тук е демонстрирана заявка за създаване на таблица за потребители:

CREATE TABLE Users

(

Id int IDENTITY(1,1) NOT NULL,

FirstName varchar(50),

LastName varchar(50),

Age int,

Email varchar(50),

[Password] varchar(50),

[RoleId] int,

PRIMARY KEY(Id),

FOREIGN KEY (RoleId) REFERENCES Roles(Id)

)

Описание на таблиците:

1. Таблицата за потребители съдържа следните колони: “Id” от тип „int“, която играе ролята на първичен ключ. “FirstName” и “LastName” от тип “varchar” съдържат имената на потребителите. “Age” от тип “int” съдържа годините на потребителите. “Email” от тип “varchar” съдържа имейл адреса на потребителите. “Password” от тип “varchar” съдържа паролите на потребителите. “RoleId” от тип “int”, която играе ролята на композитен ключ към таблицата за роли.
2. Таблицата за ролите съдържа следните колони: “Id” от тип „int“, която играе ролята на първичен ключ. “Name” от тип “varchar” съдържа имената на ролите. “IsAdmin” от тип “bit” съдържа стойност, която бива определяна като истина или лъжа според която се подразбира дали дадената роля принадлежи на администратора. “PermissionId” от тип “int”, която играе ролята на композитен ключ към таблицата за позволенията.
3. Таблицата за позволенията съдържа следните колони: “Id” от тип „int“, която играе ролята на първичен ключ. “CanCreate”, “CanUpdate”, “CanRead” и “CanDelete” от тип “bit” съдържат стойности, които биват определяни като истина или лъжа според които се подразбира дали могат да бъдат извършвани операциите: създаване, обновяване, четене и изтриване.
4. Таблицата за публикациите съдържа следните колони: “Id” от тип „int“, която играе ролята на първичен ключ. “CreatedOn” от тип “datetime” съдържа датите на създаване на публикациите. “EditedOn” от тип “datetime” съдържа датите на направа на промени на публикациите. “Image” от тип “image” съдържа снимки, които са качени към публикациите.
5. Таблицата “UserArticle” се явява междинна за таблиците за потребители и публикации като съдържа следните колони: “Id” от тип „int“, която играе ролята на първичен ключ. “UserId” от тип “int”, която играе ролята на композитен ключ към таблицата за потребители. “ArticleId” от тип “int”, която играе ролята на композитен ключ към таблицата за роли.

Връзките между таблиците са едно към много. Таблицата “UserArticle” се явява като междинна с целта да избегне връзка много към много между таблиците за потребители и публикации.

Базирано на това може да бъде сведено, че базата данни е нормализирана в трета норма форма, защото няма транзитивна зависимост.

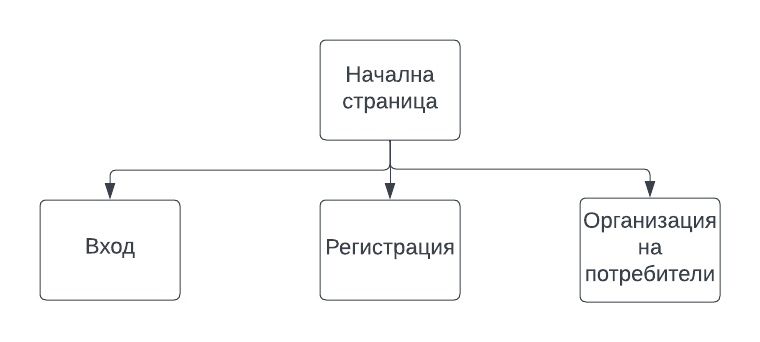
# Дизайн

Проектът е разработен на платформата “ASP.NET Core MVC” с езика “C#”. “ASP.NET Core” е платформа за уеб разработка, която може да бъде изключително полезен за даден проект и избран поради следните причини:

1. Крос-платформен: “ASP.NET Core” може да бъде използван на различни операционни системи, включително “Windows”, “macOS” и “Linux”. Това го прави подходящ за проекти, където е важно да има възможност за използване на различни платформи.
2. Бърз: “ASP.NET Core” е изключително бърз и е създаден да работи добре при големи натоварвания на сървъра. Това го прави подходящ за приложения с високо натоварване, където бързината е от съществено значение.
3. Модулен: “ASP.NET Core” се състои от множество модули, които могат да бъдат инсталирани и използвани според нуждите на проекта. Това позволява на разработчиците да използват само тези модули, които са им необходими, за да намалят размера на приложението и да го направят по-ефективно.
4. Сигурен: “ASP.NET Core” има вградени механизми за сигурност, които могат да защитят приложението от атаки като “SQL” инжекции и XSS атаки. Това може да улесни разработката на сигурни уеб приложения.
5. Лесен за употреба: “ASP.NET Core” има лесен и интуитивен начин за разработка на уеб приложения. Разработчиците могат да използват “VS” и други инструменти за разработка на уеб приложения, които правят процеса на разработка много по-бърз и по-лесен.
6. Разширяем: “ASP.NET Core” може да бъде разширен чрез използване на плъгини и допълнителни модули. Това позволява на разработчиците да добавят нови функционалности към приложението и да го направят по-мощно и гъвкаво.

Проекта е базиран на структурата “MVC”(Модел-Изглед-Контролер). Структурата “MVC” е популярен шаблон за проектиране, използван в разработката на софтуер, включително в “ASP.NET Core”. Използва се за разделяне на проблемите на дадено приложение на три основни компонента: модел, изглед и контролер.

## Реализация на архитектурата на приложението

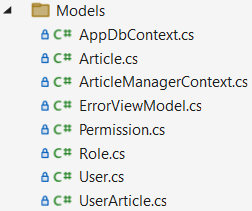


Фигура 13

Фигура 13 демонстрира каква е архитектурата на проекта. От началната страница следват общо 3 страници, “Организация на потребители” достъпна само до администратора. След регистрация потребителя бива препратен към страницата за вход, а след като си въведе данните потребителя бива препратен отново към началната страница.

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

### Модели



Фигура 14

Компонентът “Модел” представлява данните и бизнес логиката на приложението. Обикновено включва класове и структури от данни, които взаимодействат с източниците на данни на приложението, като бази данни. В проектът този компонент на модела е реализиран с помощта на класове и технологии за достъп до данни с помощта на “Entity Framework”. Фигура 14 демонстрира създадените модели в проекта.

Моделите “Article”, “Permission”, “Role”, “User” и “UserArticle” отговарят на таблиците създадени в базата данни. Моделите “AppDbContext” и “ArticleManagerContext” отговарят за връзката между проекта и базата данни. “ErrorViewModel” обработка на грешки, които могат да възникнат по време на работа на уеб приложението. Когато приложението срещне грешка, “ErrorViewModel моделът се използва за да събере информация за грешката и да я покаже на потребителя.

Тук е демонстриран моделът с класа за потребители:

public partial class User

{

public User()

{

UserArticles = new HashSet<UserArticle>();

}

[Key]

public int Id { get; set; }

public string? FirstName { get; set; }

public string? LastName { get; set; }

public int? Age { get; set; }

public string? Email { get; set; }

public string? Password { get; set; }

public int? RoleId { get; set; }

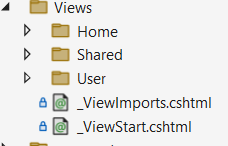
public virtual Role? Role { get; set; }

public virtual ICollection<UserArticle> UserArticles { get; set; }

}

Колоните от таблицата за потребители са представени като свойства към класа “User”. Атрибута “[Key]” служи за идентифициране на първичния ключ.

### Изгледи

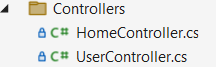


Фигура 15

Компонентът “Изглед” представлява потребителския интерфейс на приложението. Той определя как данните на приложението се представят на потребителя и как потребителят взаимодейства с тях. В проектът този компонент е реализиран чрез синтаксиса на “Razor” за генериране на “HTML”, “CSS” и “JavaScript” код. Фигура 15 демонстрира създадените изгледи в проекта. Те биват отделени в отделни папки спрямо тяхната принадлежност.

Папката “Home” съдържа изгледи, които отговарят към началната страница. Папката “Shared” съдържа изгледи, които са споделени за всички и се показват заедно с тях. Това в случая се явява във вида на навигационно меню. Папката “User” съдържа изгледи свързани с информацията на потребителите, като примери за това са страниците за вход и регистрация.

### Контролери



Фигура 16

Компонентът “Контролер” действа като посредник между компонентите “Модел” и “Изглед”. Той обработва потребителски заявки, извлича данни от модела и предава тези данни на изгледа за изобразяване. В проектът компонентът “Контролер” е реализиран с помощта на класове, които обработват входящи “HTTP” заявки и ги свързват с подходящи действия. Фигура 16 демонстрира създадените контролери в проекта.

Контролерът "HomeController" служи като основен контролер за началната страница. Той обработва заявките на потребителите, които искат да взаимодействат с изложените им действия и след това връща необходимата информация. Контролерът “UserController” служи като контролер, който отговаря за всички дейности свързани с потребителите. Примери за това са заявки с базата данни свързани с вписването или регистрирането на даден потребител.

## Организация и код на заявките към база от данни

Проектът е реализиран с методиката “Database First Approach”. По тази методика, базата данни е създадена първа и след това таблиците биват реализирани в компонента “Модел”.

За реализацията на връзката между базата и проекта са използвани методи от “Nuget Packets” на “Visual Studio” и по-точно “Microsoft.EntityFrameworkCore.SqlServer”, който позволява това. Самата заявка бива съхранявана в стрингов низ, който след това бива използван в метода за връзка.

## Наличие на потребителски интерфейс (конзолен, графичен, уеб)

Потребителският интерфейс представлява уеб страници. Те са създадени като компоненти в “Изгледи”. Те биват създадени с помощта на “Razor” методиката, която е специализирана за технологията “ASP.NET Core”. В точка 3.2 е представен примерен потребителски интерфейс с който може да бъде придобита представа за структурата на самата страница.

# Ефективност и бързодействие на решението

Настоящ проект не е първия по рода си. Налице са много новинарски сайтове, но настоящият проект се отличава с черти, които могат да бъдат счетени за коренно променящи. Една от тях е различното ниво на видимост на самите публикации. С разделянето им на одобрени и неодобрени, може да бъде предизвикан по-голям интерес у потребителите, които желаят да са по-запознати с всички подробности, колкото и малки, в днешната журналистика.

# Тестване

За тестването на ефективността на сайта беше използвана методът “NUnit”. В тази точка следват описание на тази методика и няколко примерни тестови случая.

## Какво представлява NUnit тестване

“NUnit” е един от най-популярните инструменти за тестване на софтуер в “.NET” екосистемата. Той предоставя библиотека за управление на тестовете, както и инструмент за изпълнение на тези тестове. Тестовете са написани в специален синтаксис, който позволява на “NUnit” да идентифицира тестовете и да ги изпълнява автоматично.

Тестовете в “NUnit” могат да бъдат организирани в няколко класа, като всеки клас представлява един тестов сценарий. Всяка тестова функция е отбелязана с атрибут, който указва, че това е тестова функция и какъв е очакваният резултат.

## План на тестване

Планирането на тестовете може да бъде разделено в няколко стъпки:

1. Преди да бъде започнато тестването, трябва да се определят целите на тестовете като например: тест за правилна регистрация, тест за правилен вход и т.н.
2. Трябва да бъдат избрани тестови сценарии, които да бъдат тествани, като се вземат предвид определените цели.
3. След избирането на тестовите сценарии, трябва да се напишат тестови случаи, които да проверят дали функционалността на сайта работи правилно.
4. Тестовите случаи трябва да бъдат изпълнени, за да се провери дали работят правилно.
5. Резултатите от тестването трябва да бъдат анализирани, за да се провери успеваемостта на тестовете и да се открият грешки, ако има такива.
6. Ако има грешки, трябва да бъдат оправени и да се повторят тестовете, за да се провери дали грешките са били коригирани.
7. Тестването трябва да се повтаря, докато всички тестови случаи да бъдат изпълнени успешно и да отговарят на целите, които са били поставени.

## Примерни тестови случаи

По-долу са описани примерни тестови случаи, които могат да бъдат реализирани към проекта:

1. Тест за зареждане на началната страница:
   1. Проверка за успешно зареждане на началната страница.
   2. Проверка за поне един новинарска публикация на началната страница.
2. Тест за регистрация на потребител:
   1. Проверка за успешно зареждане на страницата за регистрация на потребител.
   2. Проверка за успешна регистрация на потребител при въвеждане на валидни данни.
   3. Проверка за извеждане на подходящо съобщение за грешка при въвеждане на невалидни данни.
3. Тест за вход на потребител:
   1. Проверка за успешно зареждане на страницата за вход на потребител.
   2. Проверка за успешен вход на потребител при въвеждане на валидни данни.
   3. Проверка за извеждане на подходящо съобщение за грешка при въвеждане на невалидни данни.
4. Тест за показване на детайли за новинарска публикация:
   1. Проверка за успешно зареждане на страницата за детайли на новинарска публикация.
   2. Проверка за показване на правилната информация за избрания новинарска публикация.
5. Тест за добавяне на новинарска публикация:
   1. Проверка за успешно зареждане на страницата за добавяне на новинарска публикация.
   2. Проверка за успешно добавяне на новинарска публикация при въвеждане на валидни данни.
   3. Проверка за извеждане на подходящо съобщение за грешка при въвеждане на невалидни данни.

# Заключение и възможно бъдещо развитие

## Заключение

Проектът за новинарски публикации беше реализиран успешно. Проектирано и разработено беше функционално решение за такъв сайт, който позволява на потребителите да четат новинарски публикации, които напълно се вписват в идеята за свободна и по-достъпна журналистика. Сайтът се възползва от възможностите на технологията “ASP.NET Core” и езикът “C#”, за да създаде ефективен и лесен за използване уеб сайт, който отговаря на всички изискванията и очакванията. Проектът е завършен с цел, че ще служи като ценен ресурс за всички хора от обществото, като предоставя достъпна и лесна информация.

## Бъдещо развитие

Мислейки за бъдещо развитие, проектът може да бъде доразвит към няколко области, в които ще може допълнително да се подобри функционалността и потребителското изживяване. Една област на подобрение може да бъде прилагането на система за препоръки, която предлага статии на потребителите въз основа на тяхната история на четене или предпочитания. Това ще подпомогне с персонализирането на потребителското изживяване и ще ги задържи ангажирани със сайта за по-дълго време.

Друга област на бъдещо развитие може да бъде въвеждането на мултимедиен компонент в новинарския сайт, като видео или подкаст. Това ще разшири вида съдържание, достъпно за потребителите, и ще направи сайта по-ангажиращ и динамичен.

# Използвани литературни източници и Уеб сайтове

* (ASP.NET Core, n.d.)
* (Lock, 2021)
* (Flagman, n.d.)

# Приложения

# Критерии и показатели за оценяване

|  |  |  |
| --- | --- | --- |
| Критерии и показатели за оценяване | Максимален брой точки за показателите | Максимален брой точки за критерия |
| 1. Съответствие с изискванията за съдържание и структура на дипломния проект |  | 20 |
| 1. 1. логическа последователност и структура на изложението, балансиране на отделните части | 4 |  |
| 1.2. задълбоченост и пълнота при формулиране на обекта, предмета, целта и задачите в разработването на темата | 7 |  |
| 1.3. използване на подходящи изследователски методи | 4 |  |
| 1.4. стил и оформяне на дипломната работа (терминология, стил на писане, текстообработка и оформяне на фигури и таблици) | 5 |  |
| 2. Съответствие между поставените цели на дипломния проект и получените резултати |  | 20 |
| 2.1. изводите следват пряко от изложението, формулирани са ясно, решават поставените в началото на изследването цели и задачи и водят до убедителна защита на поставената теза | 10 |  |
| 2.2. оригиналност, значимост и актуалност на темата | 6 |  |
| 2.3. задълбоченост и обоснованост на предложенията и насоките | 4 |  |
| 3. Представяне на дипломния проект |  | 20 |
| 3.1. представянето на разработката по темата e ясно и точно | 5 |  |
| 3.2. онагледяване на експозето с:  а) презентация;  б) графични материали;  в) практически резултати;  г) компютърна мултимедийна симулация и анимация | 10 |  |
| 3.3. умения за презентиране | 5 |  |
| 4. Отговори на зададените въпроси от рецензента и/или членовете на комисията за защита на дипломен проект |  | 30 |
| 4.1. разбира същността на зададените въпроси и отговаря пълно, точно и убедително | 10 |  |
| 4.2. логически построени и точни отговори на зададените въпроси | 10 |  |
| 4.3. съдържателни и обосновани отговори на въпросите | 10 |  |
| 5. Използване на професионалната терминология, добър и ясен стил, обща езикова грамотност |  | 10 |
| 5.1. Правилно използване на професионалната терминология | 5 |  |
| 5.2. Ясен изказ и обща езикова грамотност | 5 |  |
| Общ брой точки: | Максимален бр. точки 100 | Максимален бр. точки 100 |