ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО
ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА "АПОСТОЛ АРНАУДОВ"
Р У С Е
==========
ул. "Потсдам" № 3; п.код 7005 тел. 082/84-60-96; факс 84-14-38
e-mail: electroschool@abv.bg

ЗАДАНИЕ ЗА ДИПЛОМЕН ПРОЕКТ
ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ – ЧАСТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА

по професия код 481030 „Приложен програмист"
специалност код 4810301 „Приложно програмиране"

на Алекс Георгиев Иванов ученик от XII „В" клас

Тема: СОФТУЕРНА РАЗРАБОТКА НА ИНТЕРНЕТ ФОРУМ

Изисквания за разработката на дипломния проект :

Разработване на софтуерно решение подходящо за употреба като система за управление на Интернет Форум. Разработката да има поне един от следниете потребителски интерфейси: Конзолна Апликация, Десктоп Приложение или Уеб Апликация.

График за изпълнение:

а) дата на възлагане на дипломния проект - 18.01.2022г.

б) контролни проверки и консултации 1. до 04.02.2022г.
2. до 04.03.2022г.
3. до 31.03.2022г.
4. до 29.04.2022г.

в) краен срок за предаване на дипломния проект - 05.05.2022г.

Ученик: Алекс Георгиев Иванов .....

Ръководител-консултант: Берк Минхаз Алямач .....

Директор: инж. Надежда Русева .....

Съдържание

Увод **4**
Основни Use Case-ове **5**
Гости (Guests) 5
Регистрирани Потребители (Users) 6
Администратори (Admins) 7
Архитектура на базата от данни **9**
Класове и предназначение **11**
Category.cs 11
Tag.cs 11
Post.cs 12
Comment.cs 13
ApplicationUser.cs 14
ApplicationDbContext.cs 15
IPostService.cs 16
ICommentService.cs 17
ITagService.cs 18
ICategoryService.cs 19
IApplicationUserService.cs 20
IRoleService.cs 21
ISharedService.cs 21
IPostAdminService.cs 22
IApplicationUserAdminService.cs 22
Сигурност **23**
SQL Injection 23
Cross Site Scripting (XSS) 23
Parameter Tampering 24
Cross-Site Request Forgery (CSRF) 24
Deployment **25**

Използвани технологии 27
.NET Core/ C# (версия 10.0) - версия 6.0 27
ASP.NET Core – версия 6.0 27
Entity Framework Core - версия 6.0.....28
MS SQL Server.....28
SignalR.....29
Други използвани технологии и библиотеки:.....30
Използвани IDE-та.....30
Проблеми по време на разработка 31
Бъдеще на приложението 32
Заключение.....33
Използвана литература.....34

Увод

Целта на проекта ми е изграждането на уеб форум за дискусии с добра архитектура, модерни технологии и приятен UI. Започвам да разработвам проекта с ASP.NET Core MVC.

Очакванията ми са да постигна много добър user experience и да подпомогна повишаването на нивото на образованост и обща култура на хората в световен мащаб, тъй като форумите са много добър източник на информация във всякакви сфери и направления.

ПРЕДИЗВИКАТЕЛСТВОТО ПРИЕТО!

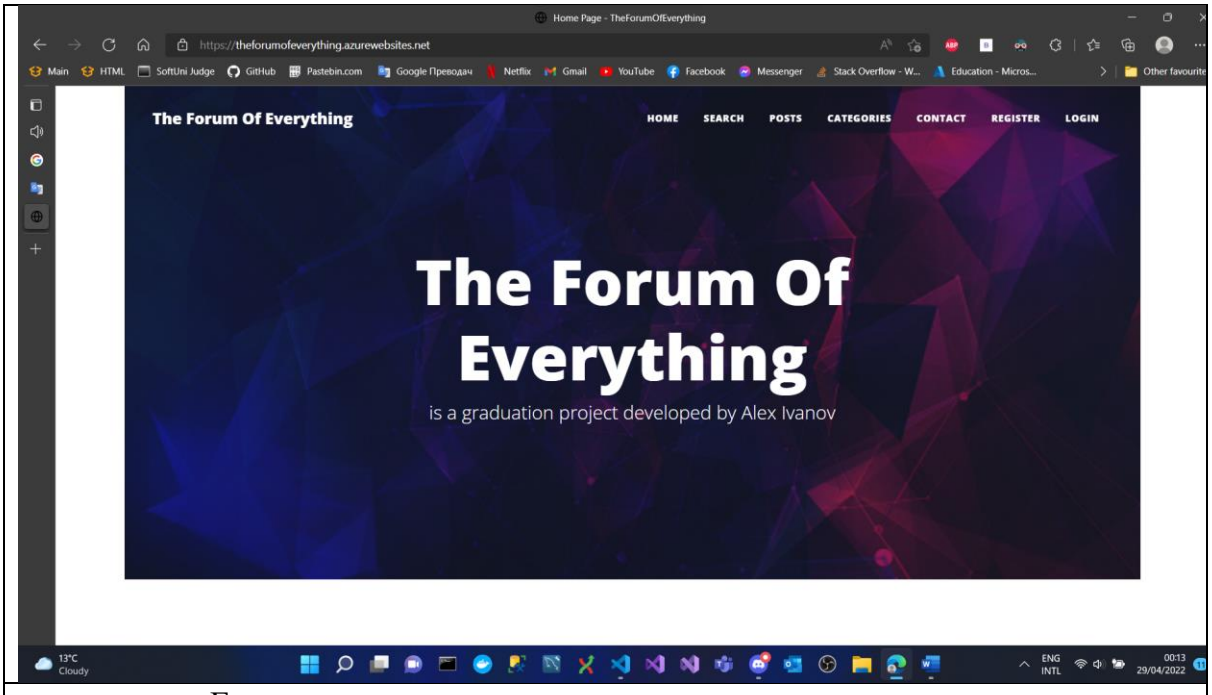
Основни Use Case-ове

В своето приложение имам три основни таргет (целеви) групи: гости, потребители и админи. В тази глава от дипломната си работа ще ви обясня основните неща, които могат да правят тези три групи.

Гости (Guests)

Това са нерегистрираните потребители т.е. „гости“ на сайта. На първо място гостите имат възможноста да се регистрират, което автоматично ще ги направи официални потребители на уеб приложението, което от своя страна ще им даде повече възможности. Освен това те имат възможноста да:

- Разглеждат категориите и таговете и съответно постовете в тях, но без да отварят съдържанието на поста
- Търсят постове по име, но отново без да могат да отворят поста, идеята на това е да могат да проверят дали във форума има пост, който да отговаря частично или цялостно на това, което търсят и ако намерят нещо, което им допада да се регистрират и по този начин хем ще бъдат официални потребители на форума и ще имат повече възможности, хем ще получат отговор на въпроса, който са имали.
- Да пращат своята обратна връзка чрез бутона “Contact”, да потърсят помощ или да докладват проблем ако има
- Да достъпят privacy полицата на сайта
- Да се регистрират, с което малко да се изкачат в йерархията

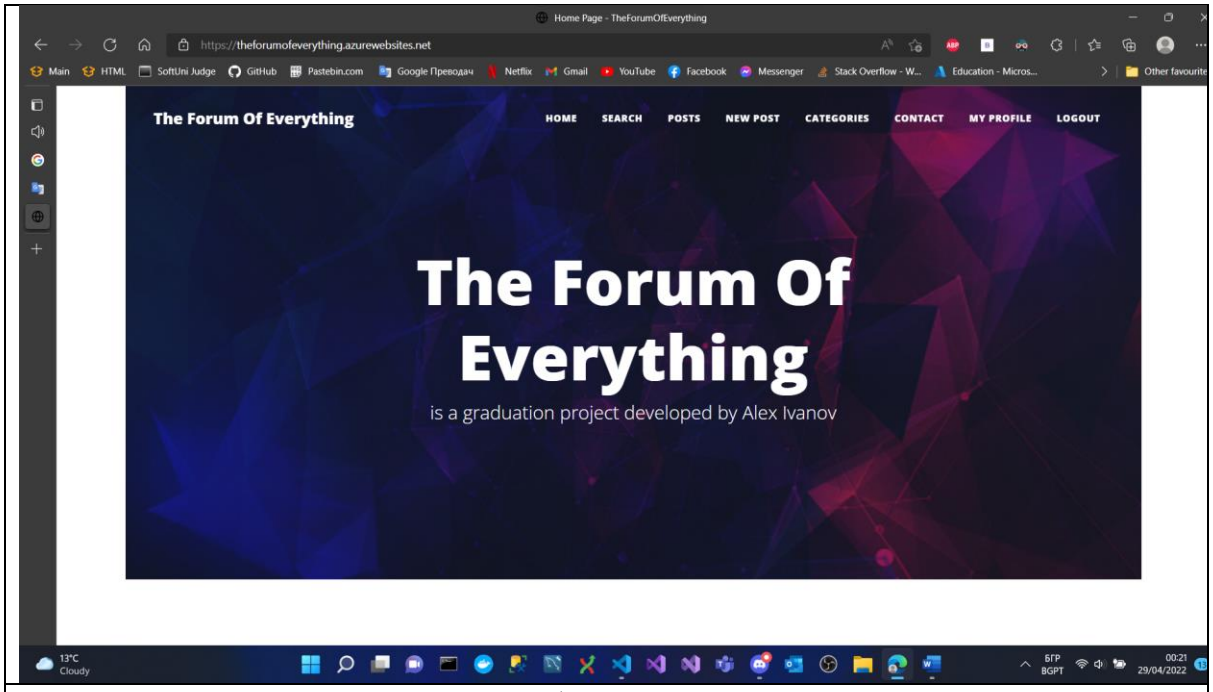


Главният прозорец на гостите и навигационната им лента

Регистрирани Потребители (Users)

Това е основната таргет група в моя проект. Това са потребителите, които са вече регистрирани и са получили достъп до почти пълната функционалност на сайта. Тези потребители имат доста повече възможности спрямо гостите. Те имат всички възможности, които имат гостите, но имат и доста повече, а именно:

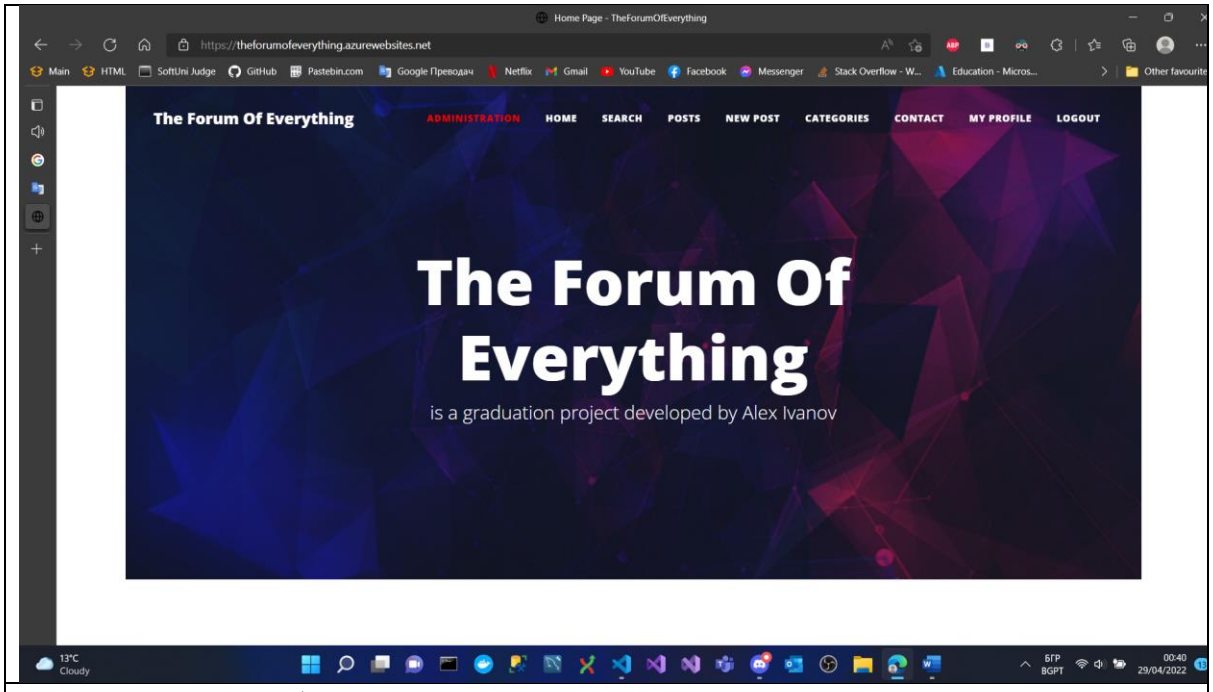
- Да се логват в уеб сайта, с което да докажат, че са регистрирани и да получат пълната функционалност, която им е обещана
- Да отварят конкретни постове и да преглеждат съдържанието и коментарите им
- Да създават собствени постове
- Да променят своите постове
- Да трият своите постове
- Да преглеждат чужди профили, които са им направили впечатление, както и всички постове от съответния профил
- Да достъпват и да правят промени по информацията в собствения си профил
- Да пишат коментари
- Да излизат от профила си като по този начин автоматично стават гости, докато отново не се логнат (влязат) в профила си



Главният прозорец на потребителите и навигационната им лента

Администратори (Admins)

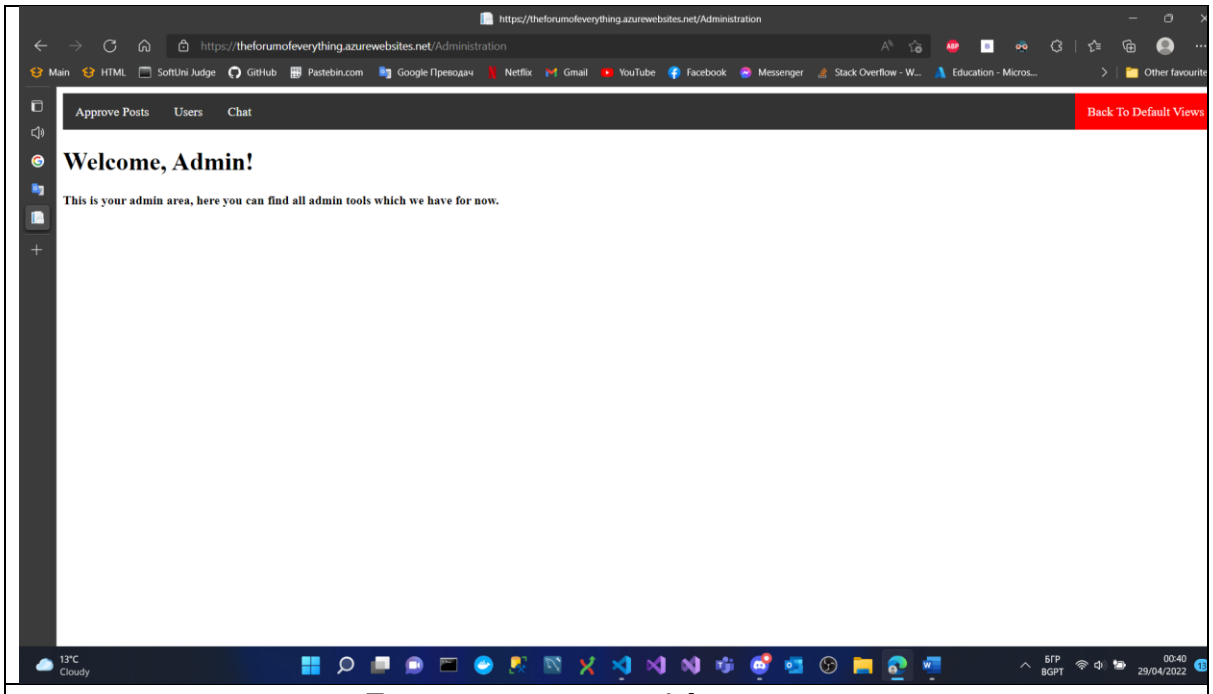
Администраторите в приложението са хората с най-много възможности и към момента са най-високо в йерархията. Казвам към момента, защото предвиждам създаване на още един вид потребители, които ще бъдат Developers и ще имат още повече възможности, но това ще стане през следващия релийз на приложението. Така за администраторите, те имат всички възможности на предходните потребители, но имат и още допълнителни неща, а именно да трият и променят, както своите, така и чуждите постове. Те за разлика от останалите потребители имат своя собствена Area (зона), която могат да достъпят през червения бутон в навигационната си лента.



Червеният бутон, чрез който администраторите могат да влязат в своята Area

В своята Area администраторите имат следните функционалности:

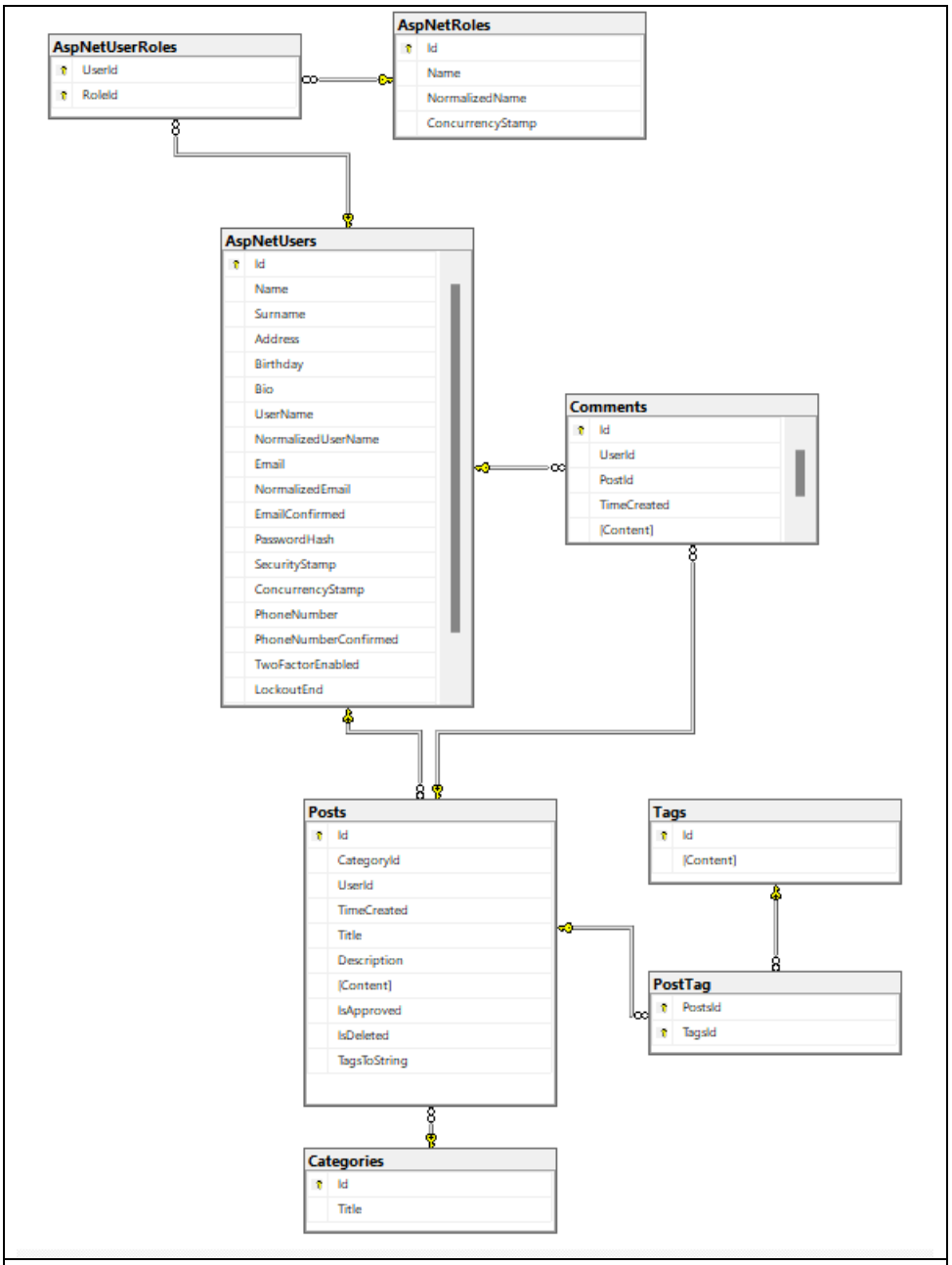
- Да одобряват постовете след като бъдат пратени за одобрение от обикновените потребители и преди да бъдат качени
- Да трият пост преди или след като бъде одобрен
- Да променят пост преди или след като бъде одобрен
- Да търсят пост по име, сред постовете чакащи одобрение
- Да правят някой обикновен потребител админ
- Да премахват админ като така го правят обикновен потребител
- Да си чатят с останалите админи като могат да бъдат спокойни, че чата им ще остане таен, тъй като не се запазва в базата от данни и след презареждане на страницата се изтриват всички съобщения



Главният прозорец в Admin area-та

Архитектура на базата от данни

За база от данни за своя проект се спрях на MS SQL Server. За да създам и после да променям своята база използвах EF Core (Entity Framework Core) и CF (Code First) подход. В базата си от данни имам следните таблици и колони:



Database диаграма на TheForumOfEverythingDB

Както забелязвате на диаграмата, имам 6 основни таблици и 2 мапинг (свързващи) таблици.

Основните са:

- AspNetUsers
- AspNetRoles
- Posts
- Tags
- Comments
- Categories

А мапинг таблиците са:

- AspNetUserRoles
- PostTag

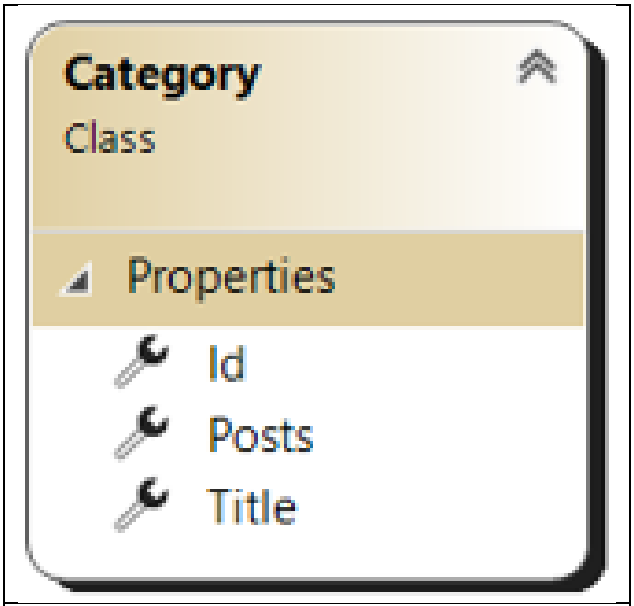
На база диаграмата бих могъл да кажа, че в своята база от данни имам следните релации:

- Постове към тагове – много към много
- Потребители към роли – много към много
- Категории към постове– едно към много
- Постове към коментари– едно към много
- Потребители към постове– едно към много
- Потребители към коментари- едно към много

Класове и предназначение

Category.cs

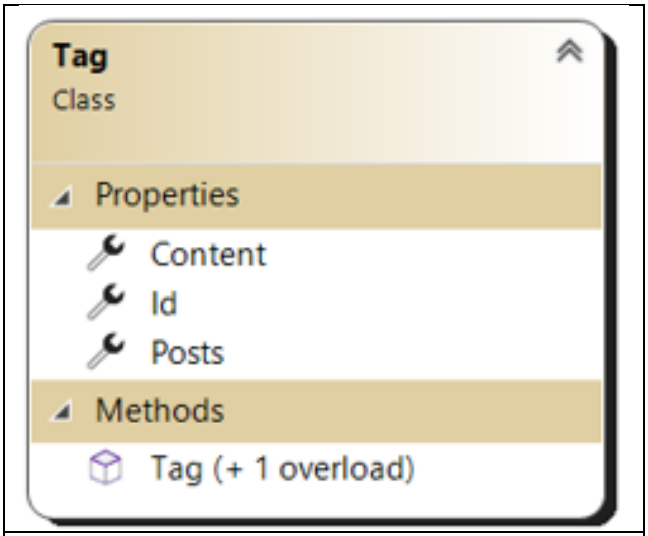
Моделен клас за съхранение на категория. Вътре са включени всички постове от тази категория, име и идентификатор на категорията.



Клас диаграма за Category.cs

Tag.cs

Моделен клас за съхранение на таг. Вътре са включени всички постове с този таг, контент(име) и идентификатор на тага. Този клас има и допълнителен конструктор, приемащ контента на тага с цел по-бързо сетване.

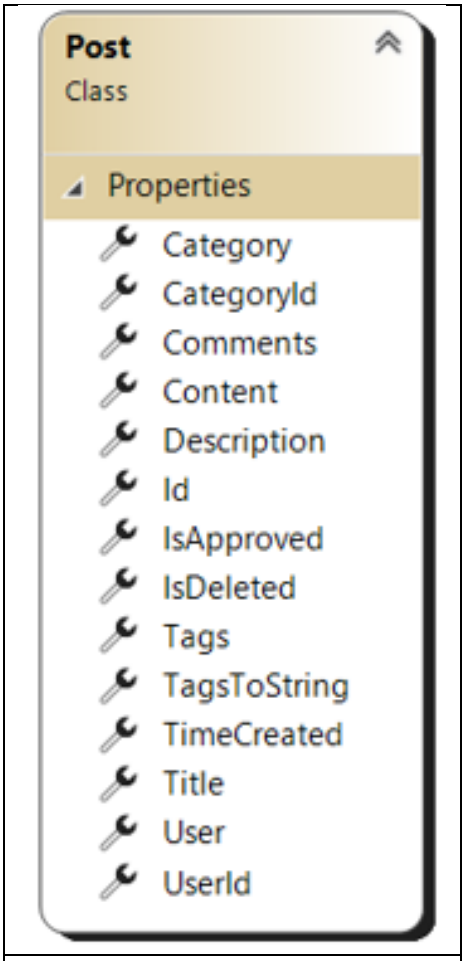


Клас диаграма за Tag.cs

Post.cs

Моделен клас за съхранение на пост, който включва:

- Категорията към която принадлежи поста (обект от клас Category)
- Идентификатора на съответната категория (Guid форматиран към string)
- Всички коментари към поста (HashSet от обекти от клас Comment)
- Съдържание (string)
- Описание (string)
- Идентификатор (Guid форматиран към string)
- Boolean променлива показваща дали поста е одобрен
- Boolean променлива показваща дали поста е изтрит
- Всички тагове на поста (HashSet от обекти от клас Tag)
- Всички тагове на поста форматирани към string и разделени с интервал
- Време на създаване на поста (DateTime форматирано към string)
- Заглавие (string с максимална дължина от 150 символа)
- Потребителя, който е създал поста (обект от клас ApplicationUser)
- Идентификатора на съответният потребител (Guid форматиран към string)



Клас диаграма за Post.cs

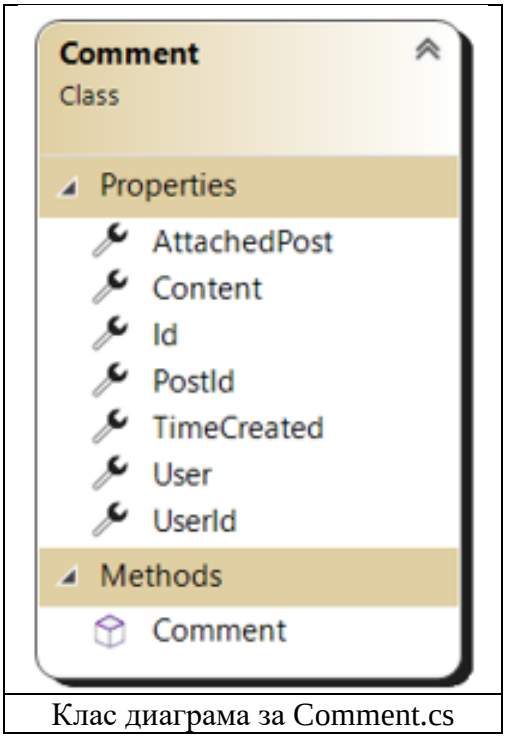
Всички пропъртита (освен HashSet-овете) съдържат атрибут „[Required]“, което означава че са задължителни за „сетване“, за да може обекта да бъде вкаран в таблицата (NOT NULL).

Comment.cs

Моделен клас за съхранение на коментар, който включва:

- Поста към който е прикачен коментара (обект от класа Post)
- Идентификатора на съответния пост (Guid форматиран към string)
- Съдържание на коментара (string)
- Идентификатор на коментара (Guid форматиран към string)
- Време на създаване на коментара (DateTime форматирано към string)
- Потребителя, който е създал коментара (обект от клас ApplicationUser)
- Идентификатора на съответният потребител (Guid форматиран към string)

Всички пропъртита съдържат атрибут „[Required]“, което означава че са задължителни за „сетване“, за да може обекта да бъде вкаран в таблицата (NOT NULL).



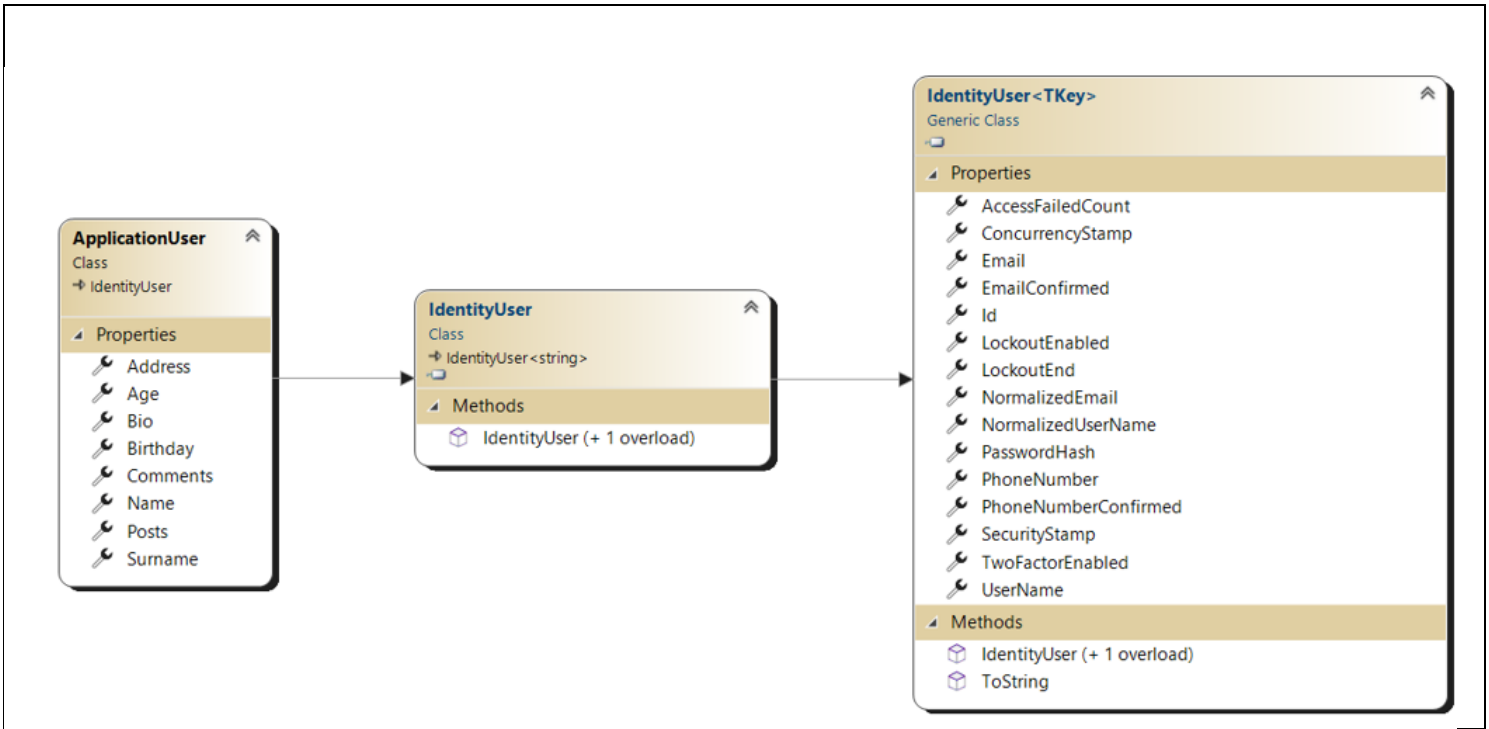
Клас диаграма за Comment.cs

ApplicationUser.cs

Моделен клас за съхранение на потребител, който включва:

- Адреса на потребителя (string- nullable)
- Годините на потребителя (int (сами се сетват на база рождения ден))
- Биография (string- nullable)
- Рожден ден (DateTime)
- Всички коментари на потребителя (HashSet от обекти от клас Comment)
- Всички постове на потребителя (HashSet от обекти от клас Post)
- Име (string- nullable)
- Презиме (string- nullable)

Този клас наследява класа IdentityUser с цел да бъде разширен (да бъдат добавени нови колони в таблицата) класа по подразбиране, по тази причина класа ApplicationUser съдържа в себе си всички пропъртита и на класа IdentityUser като адрес, идентификатор, никнейм, имейл и тн. Останалите пропъртита на класа IdentityUser се виждат на диаграмата по-долу.



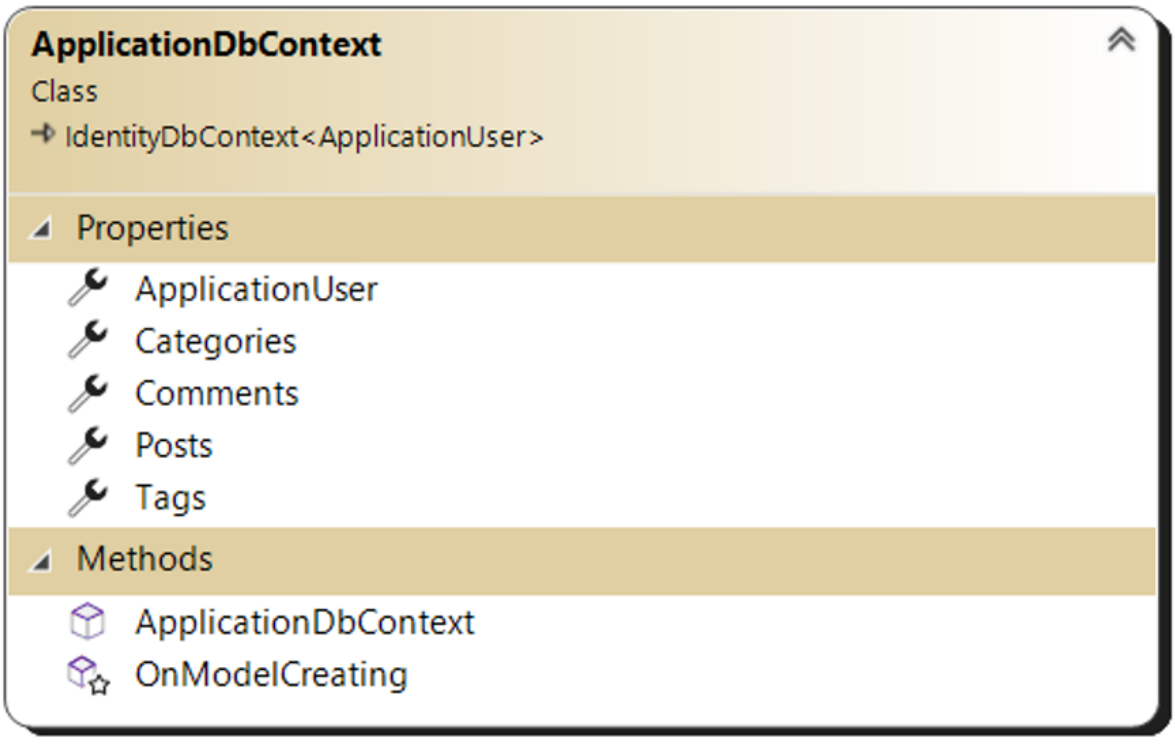
Клас диаграма за ApplicationUser.cs

ApplicationDbContext.cs

Клас за setup-ване на базата от данни, който включва:

- DbSet от ApplicationUser
- DbSet от Category
- DbSet от Comment
- DbSet от Post
- DbSet от Tag

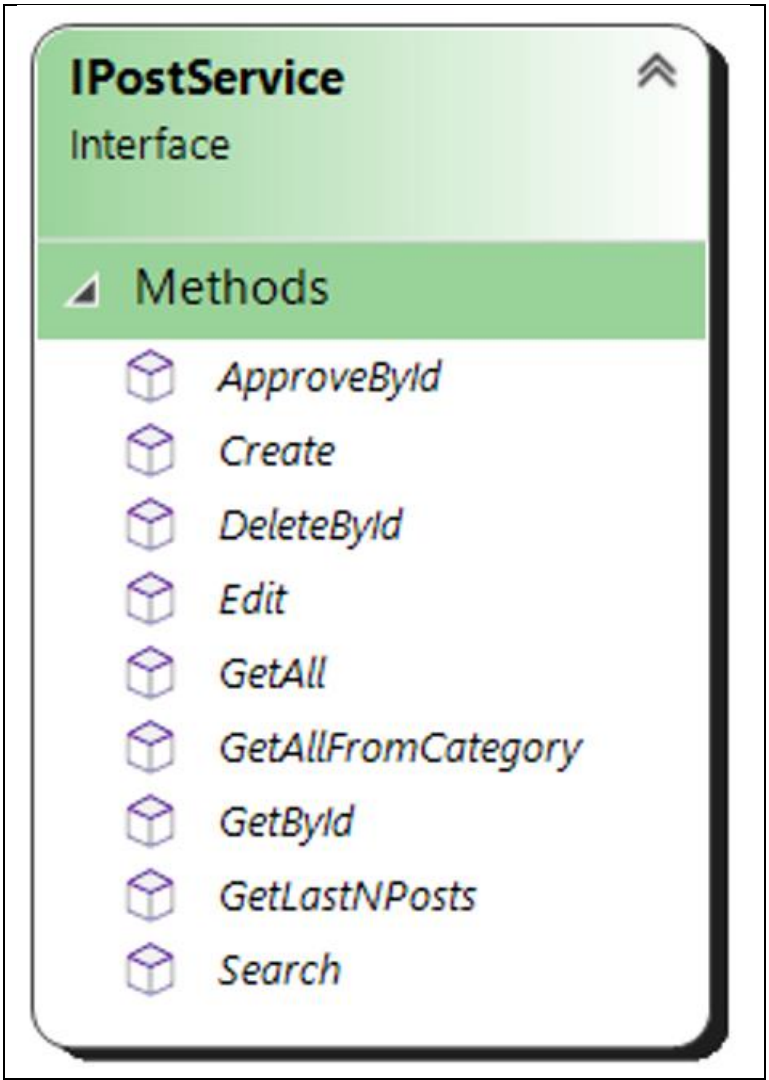
Също така класа съдържа метод onModelCreating, който служи за допълнително описване на таблиците и релациите между тях в базата от данни.



Клас диаграма за ApplicationDbContext.cs

IPostService.cs

Интерфейсът съдържащ в себе си описание на методи за одобряване на постове, създаване, триене, промяна, извличане на всички постове, взимане на всички постове от дадена категория, вземане на даден пост, вземане на последните N на брой постове и търсене на постове по име.



Клас диаграма за IPostService.cs

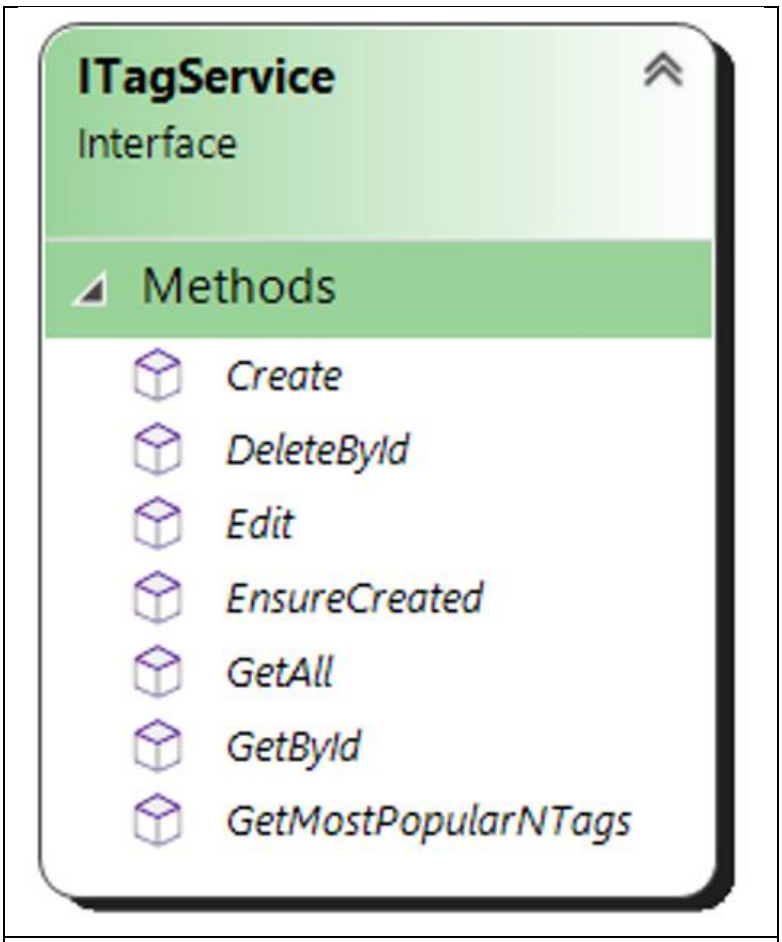
ICommentService.cs

Интерфейсът съдържащ в себе си описание на методи за създаване, триене, промяна, извличане на всички коментари, вземане на даден коментар и вземане на последните N на брой коментари.

Клас диаграма за ICommentService.cs

ITagService.cs

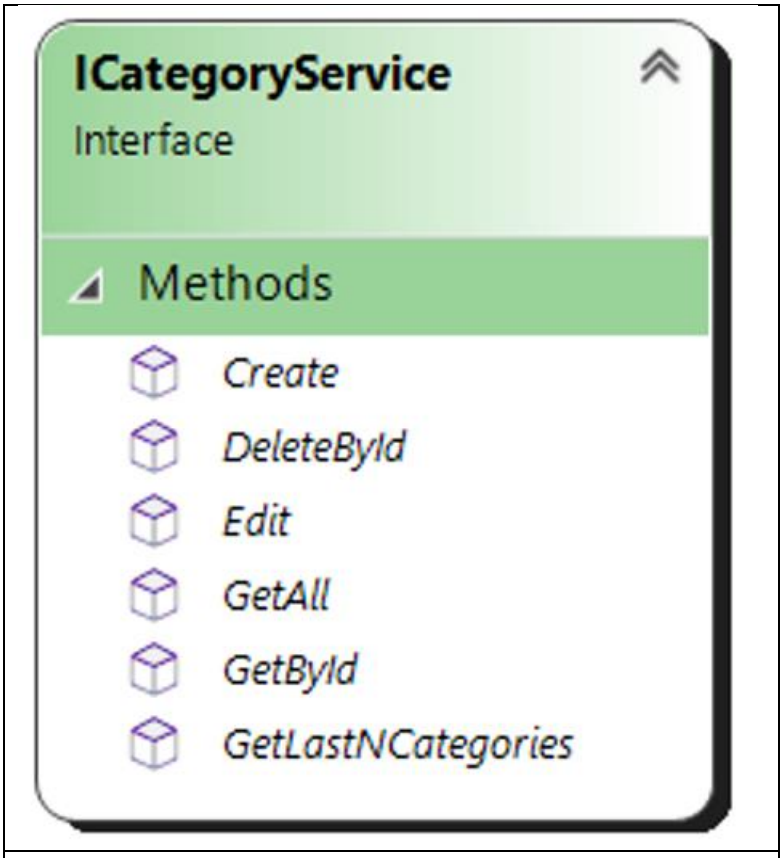
Интерфейсът съдържащ в себе си описание на методи за създаване, триене, промяна, извличане на всички тагове, проверка дали всички подадени тагове са създадени, вземане на даден таг и вземане на най-използваните N на брой тагове.



Клас диаграма за ITagService.cs

ICategoryService.cs

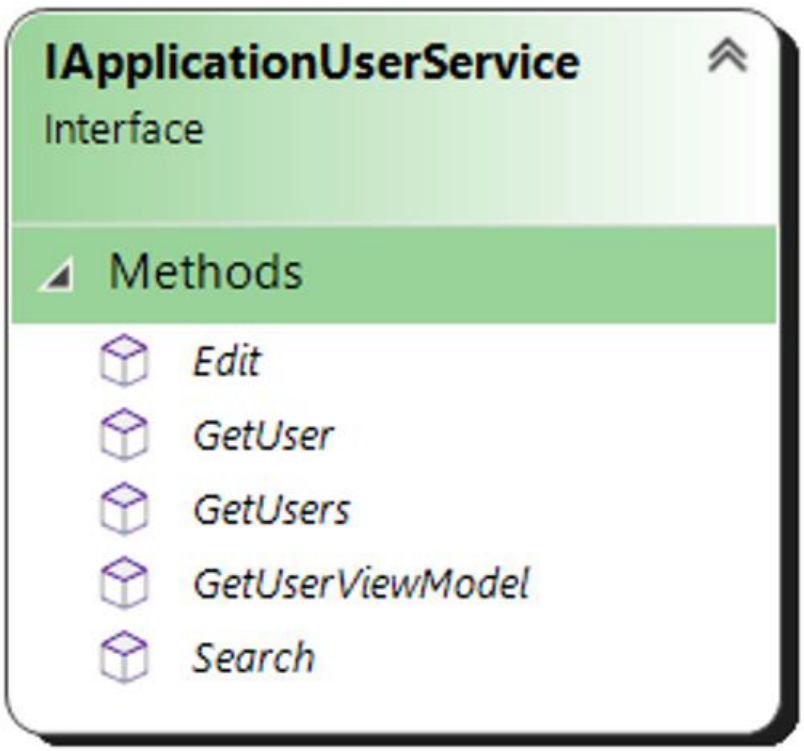
Интерфейсът съдържащ в себе си описание на методи за създаване, триене, промяна, извличане на всички категории, вземане на дадена категория и вземане на последните N на брой категории.



Клас диаграма за ICategoryService.cs

IApplicationUserService.cs

Интерфейсът съдържащ в себе си описание на методи за промяна, извличане на даден потребител, вземане на всички потребители и търсене по имейл.



Клас диаграма за IApplicationUserService.cs

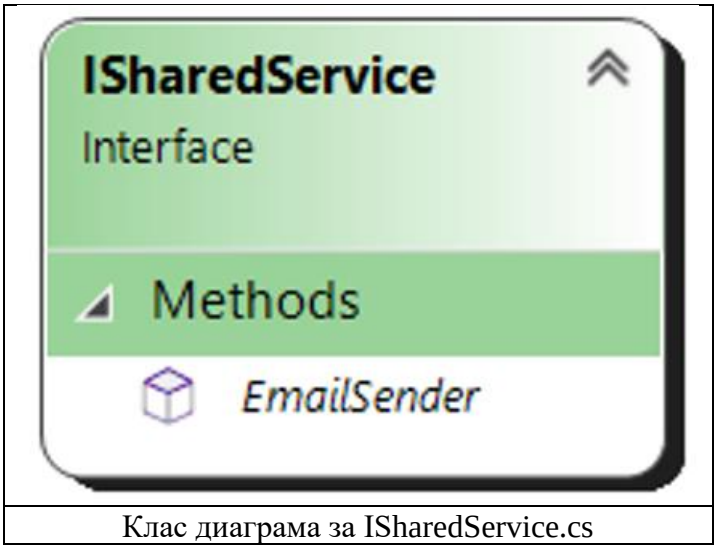
IRoleService.cs

Интерфейсът съдържащ в себе си описание на методи за добавяне на потребител към роля и за създаване на нова роля.

Клас диаграма за IRoleService.cs

ISharedService.cs

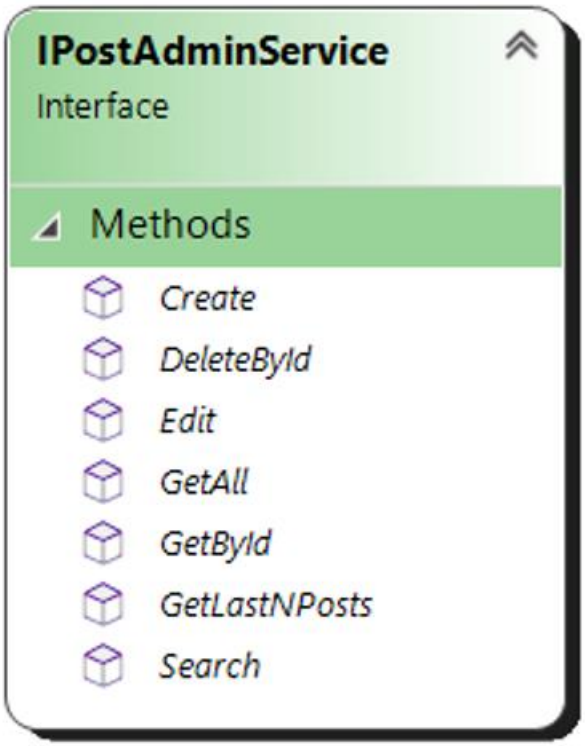
Интерфейсът съдържащ в себе си описание на метод за изпращане на имейлите за обратна връзка към служебния имейл и за връщане на отговор на изпращача.



Клас диаграма за ISharedService.cs

IPostAdminService.cs

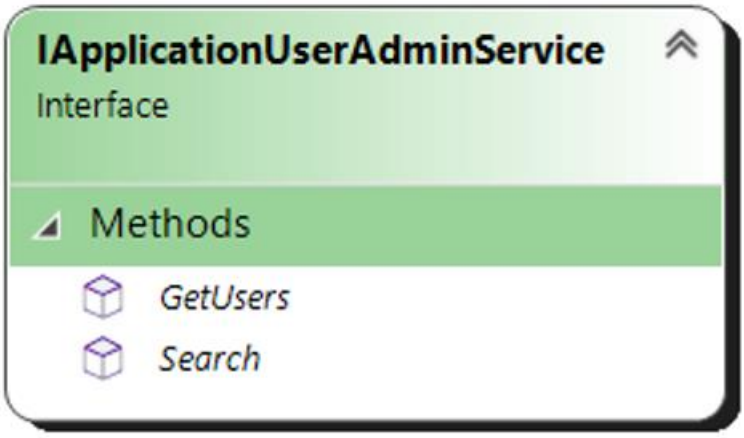
Интерфейсът съдържащ в себе си описание на методи за създаване, триене, промяна, извличане на всички постове, вземане на даден пост, вземане на последните N на брой постове и търсене на постове по име. Този интерфейс е създаден за Administrator Area-та.



Клас диаграма за IPostAdminService.cs

IApplicationUserAdminService.cs

Интерфейсът съдържащ в себе си описание на методи за извличане на всички потребители и за търсене на потребители по имейл. Този интерфейс е създаден за Administrator Area-та.



Клас диаграма за IApplicationUserAdminService.cs

Сигурност

По време на разработката на приложението обърнах сериозно внимание на сигурноста. Част от защитите идват от ASP.NET Core, но те не ми бяха достатъчни в някои моменти и използвах някои външни библиотеки за защита като HtmlSanitizer например.

SQL Injection

SQL Injection е техника за инжектиране на код, която може да унищожи вашата база данни. Това е една от най-често срещаните техники за уеб хакване. Някои често срещани примери за SQL Injection включват:

- Извличане на скрити данни, където можете да модифицирате SQL заявка, за да върнете допълнителни резултати.
- Подхвърляне на логиката на приложението, където можете да промените заявка, за да се намесва в логиката на приложението.
- UNION атаки, при които можете да извличате данни от различни таблици на база данни.

Реших този проблем като използвах Entity Framework Core, който сам по себе си защитава от този тип атаки, защото никъде и никога не се използват заявки в чист вид.

Cross Site Scripting (XSS)

Това е уязвимост в сигурността, която може да се намери в някои уеб приложения. XSS атаките позволяват на нападателите да инжектират клиентски скриптове в уеб страници, преглеждани от други потребители. Липсата на защита от този тип атаки е много често срещан проблем при съвременните уеб сайтове, за който трябва много да се внимава, тъй като може да доведе до всякакви поражения върху приложението.

За да подсигуря сигурноста на приложението си от този тип атаки използвам HtmlSanitizer. Това е open-source библиотека, която проверява всички входни данни към приложението за скриптове и при откриването на такива ги „неутрализира“.

Към настояща дата това е най-добрата защита от XSS атаки.

Parameter Tampering

Атаката за подправяне на уеб параметри се основава на манипулиране на параметри, обменени между клиент и сървър, за да се модифицират данните на приложението, като потребителски идентификационни данни и разрешения, цена и количество продукти и т.н. Обикновено тази информация се съхранява в бисквитки, скрита форма полета или низове на URL заявка и се използва за увеличаване на функционалността и контрола на приложението.

За да се предпазя от този тип атаки използвам силна енкапсулация в приложението си и по този начин елеминирам тази опасност.

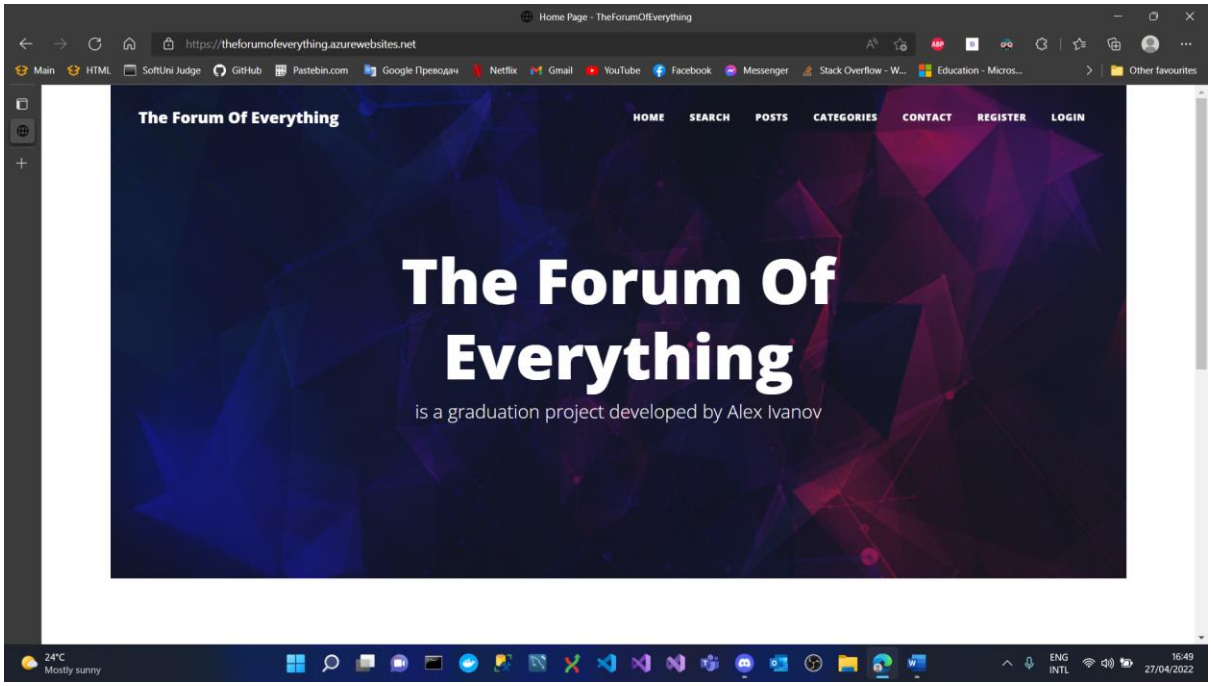
Cross-Site Request Forgery (CSRF)

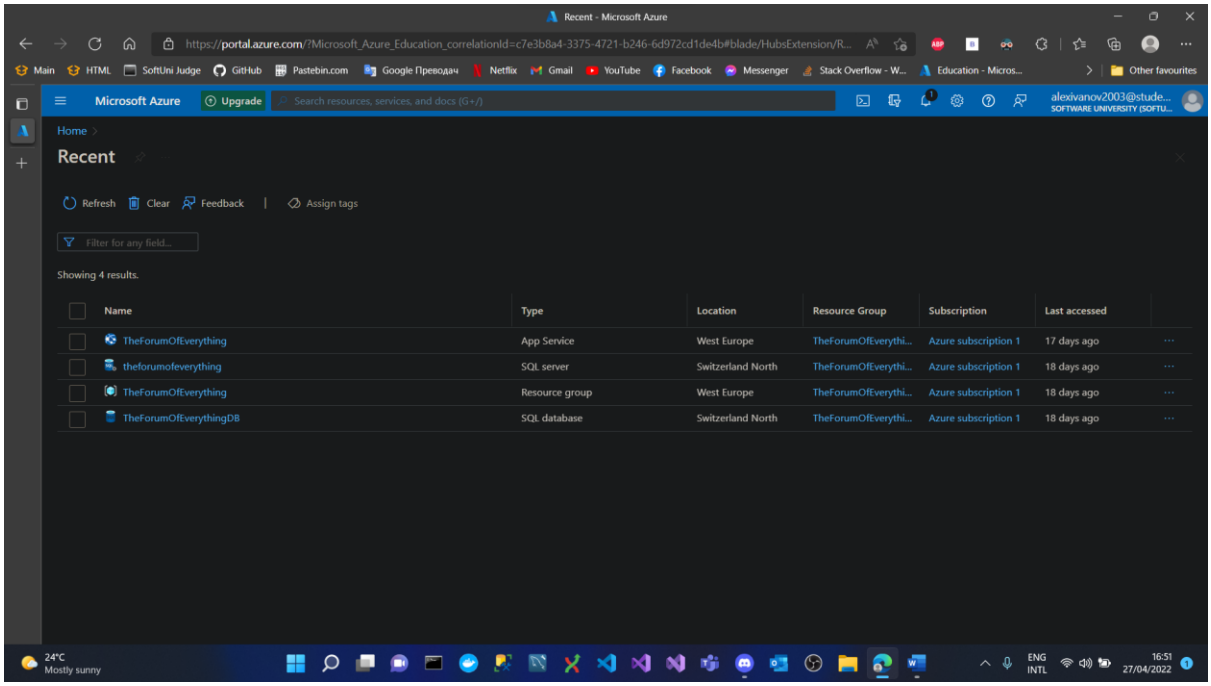
Фалшифицирането на междусайтови заявки, известно още като атака с едно щракване или управление на сесия и съкратено като CSRF или XSRF, е вид злонамерена експлоатация на уебсайт, при която се изпращат неоторизирани команди от потребител, на когото уеб приложението има доверие.

За да защитя софтуера си от това използвам GET заявки само за действия за преглед или само за четене(view or read-only actions). Тези типове заявки не трябва да трансформират данни и трябва да показват само записани данни. Това ограничава броя на заявките, които са уязвими към CSRF атаки.

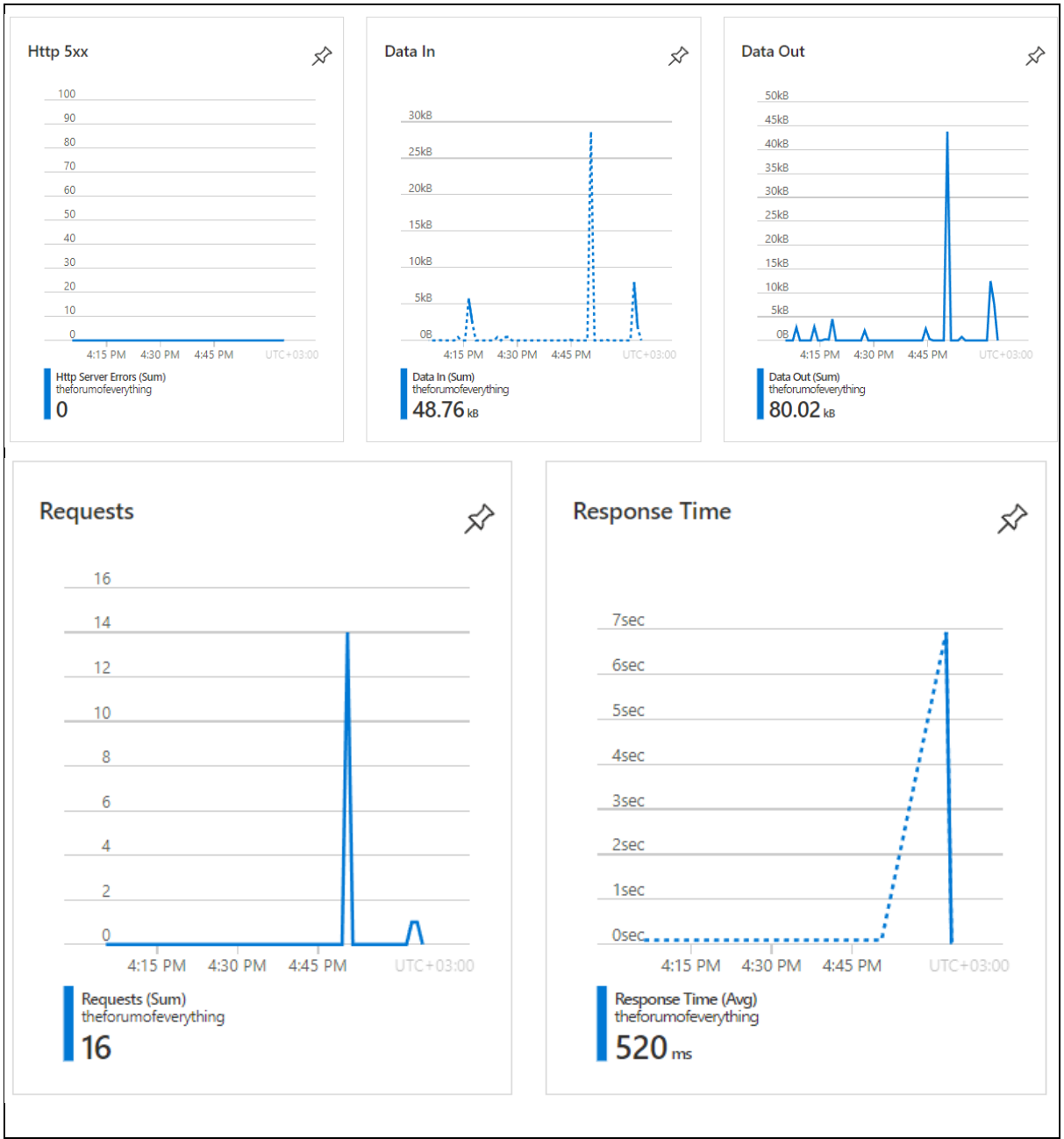
Deployment

За среда за Deployment се спрях на Azure. Microsoft предоставят безплатни услуги на стойност 200 евро за първия месец. Създадох си базата от данни в Azure като сървъра се намира в Швейцария. Уеб сървъра на самото приложение отново е качен в Azure, но физически е в Германия. Срещнах известни трудности по време на конфигурацията, но в рамките на няколко дни всичко беше нагласено и качено.





Azure предоставят много удобен tool за мониторинг, което беше много голям плюс спрямо останалите платформи.



Мониторинг диаграми в Azure

Използвани технологии

.NET Core/ C# (версия 10.0) - версия 6.0

.NET Core е нова версия на .NET Framework, която е безплатна платформа за разработка с отворен код с общо предназначение, поддържана от Microsoft. Това е cross-platform framework, който работи на операционни системи Windows, macOS и Linux.

.NET Core Framework може да се използва за изграждане на различни типове приложения като мобилни, настолни, уеб, облачни, IoT, machine learning, микросървиси, игри и т.н.

.NET Core е написан от нулата, за да бъде модулен (modular), лек, бърз и cross-platform Framework. Той включва основните функции, които са необходими за стартиране на основно приложение .NET Core. Други функции се предоставят като пакети NuGet, които можете да добавите в приложението си, ако е необходимо. По този начин приложението .NET Core ускорява производителността, намалява обема на паметта и става лесно за поддръжка.

ASP.NET Core – версия 6.0

ASP.NET Core е безплатен framework за изграждане на уебсайтове и уеб приложения на .NET Core с помощта на HTML, CSS и JavaScript.

ASP.NET Core MVC 5 е framework, базирана на архитектура Model-View-Controller (MVC). Разработчиците могат да създават динамични уеб приложения, използвайки ASP.NET MVC framework, която позволява ясно разделяне на проблемите, бърза разработка и TDD (Test-driven development

Entity Framework Core - версия 6.0

Entity Framework (EF) Core е лека, разширяема, с отворен код и cross-platform версия на популярната технология за достъп до данни на Entity Framework.

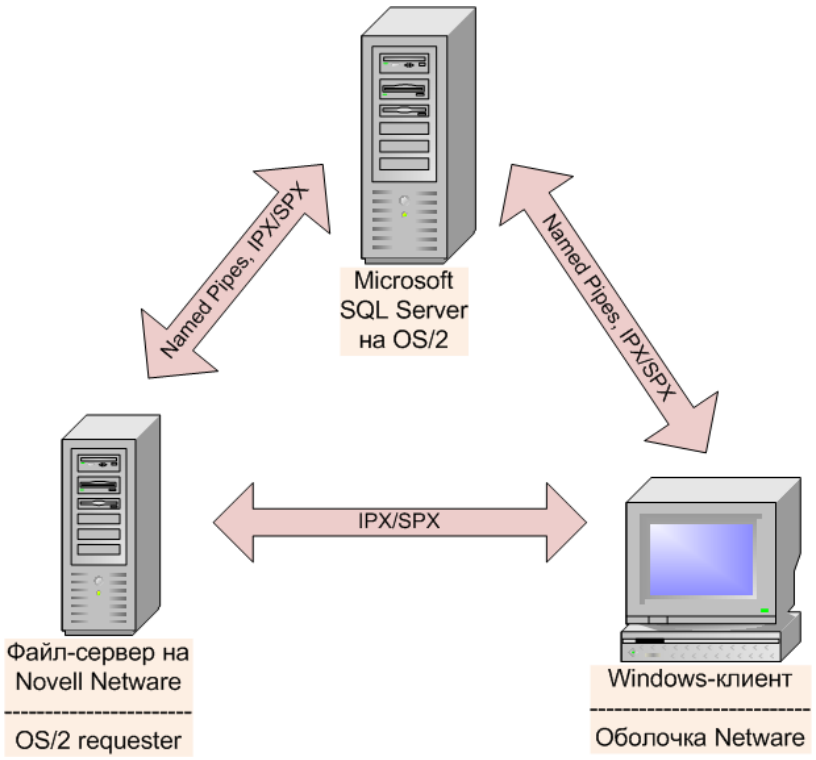
EF Core може да служи като обектно-релационен mapper (O/RM), който:

- Позволява на .NET разработчиците да работят с база данни, използвайки .NET обекти.
- Елиминира необходимостта от по-голямата част от кода за достъп до данни, който обикновено трябва да бъде написан.

EF Core поддържа много engines за бази от данни.

MS SQL Server

Microsoft SQL Server е сървърна система за управление на бази от данни (и по-точно на релационни бази от данни) на компанията Microsoft. Microsoft SQL Server е предназначена за управление на големи сървърно базирани БД, за разлика от MS Access, която е desktop базирана и не е предназначена за управление на големи корпоративни БД.

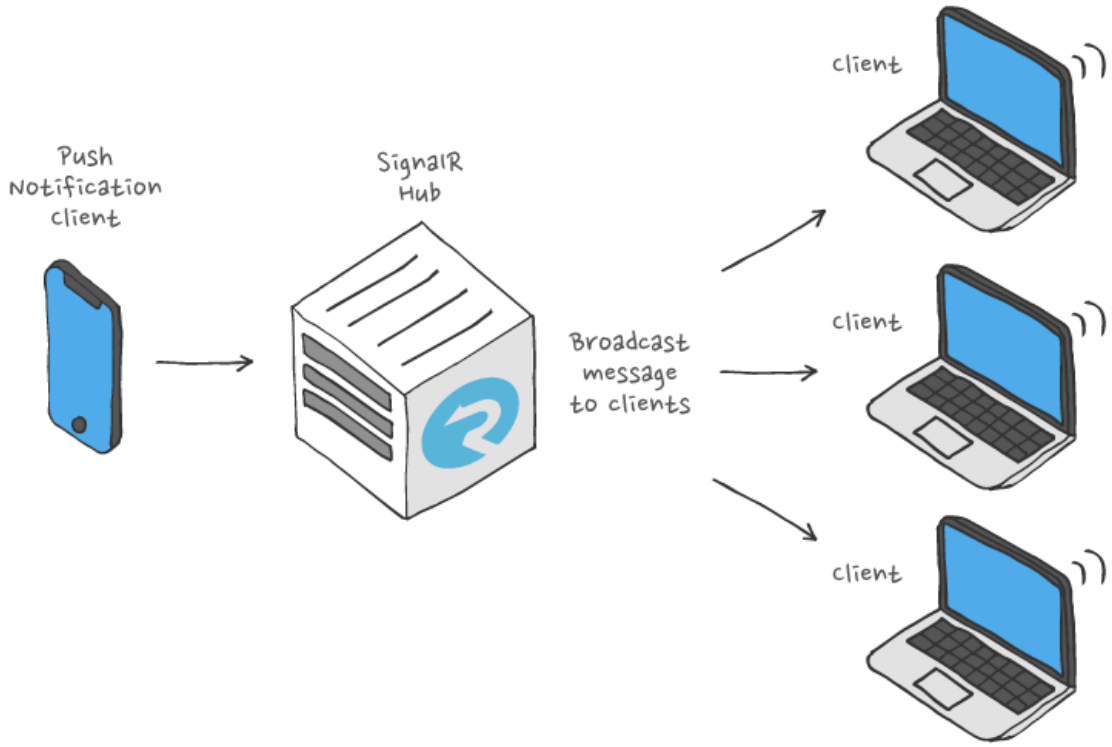


SignalR

ASP.NET SignalR е библиотека за ASP.NET разработчиците, за да добавят уеб функционалност в реално време към своите приложения. Уеб функционалността в реално време е способността да има код от страна на сървъра да изпраща съдържание към свързаните клиенти, както се случва, в реално време.

SignalR се възползва от няколко транспорта, като автоматично избира най-добрия наличен транспорт предвид възможностите на клиента и сървъра. SignalR се възползва от WebSocket, HTML5 API, който позволява двупосочна комуникация между браузъра и сървъра. SignalR ще използва WebSockets под кориците (“under the covers”), когато е наличен, и грациозно ще се върне към други техники и технологии, когато не е, докато кодът на приложението остава същият.

SignalR също така предоставя прост API на високо ниво за извършване на RPC от сървър до клиент (извикване на JavaScript функции в браузъра на клиента от .NET код от страна на сървъра) в ASP.NET приложение, както и добавяне на полезни hooks за управление на връзката , като събития за свързване/изключване, групиране на връзки, оторизация.



Други използвани технологии и библиотеки:

- HtmlSanitizer
- Razor Pages
- Bootstrap 5
- NUnit

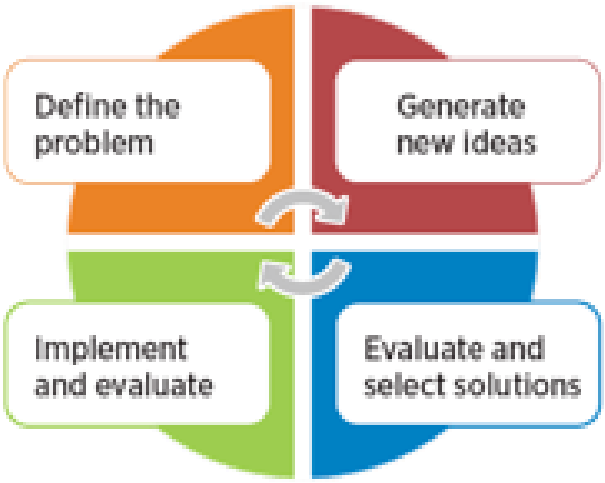
Използвани IDE-та

- Visual Studio 2022 Enterprise
- Visual Studio Code
- Microsoft SQL Server Management Studio 18
- Notepad++
- Git Extensions

Проблеми по време на разработка

По време на разработката се сблъсках лице в лице с редица проблеми от всякакво естество. Прооменял съм базата от данни десетки пъти било то за да добавя нова функционалност или за да разреша някакъв проблем. В своето приложение имах проблем с XSS атаките, понеже приложението ми беше доста податливо на такива, но бих могъл да кажа, че вече е напълно защитено.

Като цяло, в процеса на разработка на приложението се сблъсках с наистина много и различни проблеми, било то бъгове или просто неща, които не знаех и трябваше да науча, за да завърша проекта. Научих много нови неща и успях да постигна желаните резултати, което от своя страна ми костваше много нерви и безсънни нощи.



Бъдеще на приложението

Като цяло форумът е нещо много хубаво, защото има толкова много нови и нови неща, които могат да бъдат направени. За в бъдеще планувам да закупя домейн и хостинг, където проекта да може да влезе в реална употреба, с което да носи знания и да помага на хората, именно което е и основната цел на форума ми.

Също така планувам да създавам още един вид потребители на приложението, а именно потребители с роля „Developers“, които ще имат още много възможности и ще изместят администраторите от върха на веригата.

Заключение

TheForumOfEverything вече е напълно функционален и постигна всички поставени цели. След предуставянето му на над 50 потребители и анкетирането им след това, можем да заключим, че проекта е успешен като до момента той не разполага с негативни отзиви. От анекдотични мнения на потребители и чрез емпирично тестване се вижда, че крайния резултат е ефективен и проекта перфектно върши желаните от него задачи.

Проектът успя да постигне зададените цели.

МИСИЯТА ИЗПЪЛНЕНА!

Използвана литература

1. Въведение в Програмиране със C#
   Колектива на Телерик (2011)
   ISBN: 978-954-400-527-6

2. Fundamentals of Computer Programming with C#
   Светлин Наков и колектив (2013)
   ISBN: 978-954-400-773-7

3. Въведение в .Net
   Денис Колисинченко (2016)
   ISBN: 978-954-8898-86-7

4. Основи на програмирането със C#
   Светлин Наков и колектив (2017)
   ISBN: 978-619-00-0635-0

5. https://docs.microsoft.com/en-us/dotnet/
   Official .Net documentation by Microsoft.

6. https://docs.microsoft.com/en-us/dotnet/csharp/
   Official C# documentation by Microsoft.

7. https://docs.microsoft.com/en-us/visualstudio/
   Official Visual Studio documentation by Microsoft.

8. Overview of Entity Framework Core - EF Core | Microsoft Docs
   Official Entity Framework Documentation

9. SQL Server technical documentation - SQL Server | Microsoft Docs
   SQL Server Documentation

10. HTML Sanitizer API - Web APIs | MDN (mozilla.org)