МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра інформаційних технологій

Пояснювальна записка

до кваліфікаційної роботи
другого (магістерського) рівня

на тему **РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ КОНТЕТОМ WEB-САЙТУ**

Виконав: студент 2 курсу, групи ІКК-2.1
спеціальності
121 Інженерія програмного забезпечення

Таран А. О.

Керівник Григор'єва Т.І.

Рецензент Л.Т. Йона

Одеса – 2023 р.

ДОВІДКА

кафедри ІТ про виконану магістерську роботу

студента 2 курсу ФКПІ та КН групи ІПЗ

Тарана Андрія Олеговича

на тему Розробка системи управління контетом web-сайту

Висновок нормоконтролера кваліфікаційна робота відповідає вимогам. Нормоконтролер канд. техн. наук, доц. (науковий ступінь, вчене звання, посада) (підпис, дата) 15.12.2023 Вікторія ІВ. (і. б. прізвище)

Висновок відповідального за наявність плагіату згідно з сертифікатом ID унікальність роботи підтверджена

Відповідальна особа канд. техн. наук, доц. (науковий ступінь, вчене звання, посада) (підпис, дата) 15.12.2023 Вікторія ІВ (і. б. прізвище)

Попередня експертиза (захист) магістерської роботи (бакалаврської роботи чи магістерської роботи)

студ. Таран А.О. (прізвище і.б.) проведена "18" 12 2023р.

Висновки кваліфікаційна робота виконана з деякими недоліками. В роботі реалізована програмна система, що надає базовий функціонал для створення, редагування та організації веб-сторінок. Магістерська робота рекомендується до захисту

Члени комісії (підпис) д.т.н., проф. Стеценковська І.В. (науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н., доц. Григор'єва Т.І. (науковий ступінь, вчене звання, посада, прізвище і.б.)

(підпис) к.т.н., доц. Гордалов В.Е. (науковий ступінь, вчене звання, посада, прізвище і.б.)

МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра інформаційних технологій
Освітній ступінь магістр
Галузь знань 12 Інформаційні технології Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доц.
Т.І. Григор'єва
"27" 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ

Тарану Андрію Олеговичу

1. Тема роботи Розробка системи управління контентом web-сайту
керівник роботи Григор'єва Тетяна Ігорівна, к.т.н., доцент.
затверджені наказом закладу вищої освіти від 25.09.2023 р. № 1957 зі змінами від 04.12.2023 р. № 3102
2. Строк подання студентом роботи 11.12.2023 р.
3. Вихідні дані до роботи
Стандарти написання коду з використанням мови програмування C#, документація по програмним модулям ASP.NET.
Навчальні посібники по предметам "Проектування інформаційних систем", "Якість программного забезпечення та тестування".

4. Зміст розрахунково-пояснювальної записки
1 «Мета та завдання»
2 «Пошук і порівняння аналогів»
3 «Формування вимог до пз»
4 «Надання сайту відповідного вигляду»
5 «Проектування архітектури»
6 «Опис створення сайту»
7 «Створення функціоналу»
8 «Реалізація і керівництво користувача»
9 «Опублікування проекту»

5. Перелік графічного матеріалу (з зазначенням обов'язкових креслень)
Слайд 1 – Мета та завдання
Слайд 2 – Формування вимог до ПЗ

Слайд 3 – UML діаграма

Слайд 4 – Проектування архітектури

Слайд 5 – Створення функціоналу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.09.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Мета та завдання	25.09.2023-02.10.2023	вик
2	Пошук і порівняння аналогів	02.10.2023-09.10.2023	вик
3	Формування вимог до пз	09.10.2023-16.10.2023	вик
4	Надання сайту відповідного вигляду	16.10.2023-30.10.2023	вик
5	Проектування архітектури	30.10.2023-13.11.2023	вик
6	Опис створення сайту	13.11.2023-20.11.2023	вик
7	Створення функціоналу	20.11.2023-27.11.2023	вик
8	Реалізація і керівництво користувача	27.11.2023-04.12.2023	вик
9	Опублікування проекту	04.12.2023-11.12.2023	вик

Студент _____ А.О. Таран
(підпис)

Керівник роботи _____ Т.І. Григор'єва
(підпис)

ВІДГУК
наукового керівника на кваліфікаційну
магістерську роботу
на тему «**Розробка системи управління контетом web-сайту**»
студента Тарана А. О.
Спеціальність: 121 Інженерія програмного забезпечення

Актуальність розробки системи управління веб-контентом визначається сучасними вимогами та тенденціями в сфері веб-розробки та управління контентом.

В роботі успішно реалізувана програмна система, що надає базовий функціонал для створення, редагування та організації веб-сторінок. Організація базової структури та вибір html-шаблону для макету сайту також вказують на практичні навички студента Таран А. О. у сфері розробки.

Текстова частина магістерської роботи викладена послідовно і чітко.

Результати дослідження представлені у тезах доповідей IX Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих учених «Гуманітарний і інноваційний ракурс професійної майстерності: пошуки молодих вчених».

До недоліків варто віднести:

1. В аналізі існуючих програмних продуктів можна було б надати більше критичних висновків і оцінок. Також, бажано було б ретельніше розглянути результативність розробленої системи та її ефективність у порівнянні з конкурентами.

2. Необхідно детальніше дослідити аспекти, пов'язані з вибором та реалізацією конкретних технологій, щоб забезпечити максимальну ефективність та сумісність системи.

Вказані недоліки не знижують цінності виконаної роботи.

В цілому магістерська робота студента Таран А. О. відповідає вимогам до кваліфікаційних робіт магістрів і заслуговує на оцінку "задовільно". Студент Таран А. О. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 «Інженерія програмного забезпечення».

Науковий керівник:
завідувачка кафедри Інформаційних технологій,
кандидат технічних наук, доцент Т.І.Григор'єва

РЕЦЕНЗІЯ

на магістерську роботу студента Таран А. О.

Спеціальність: 121 Інженерія програмного забезпечення

на тему «**Розробка системи управління контетом web-сайту**»

Магістерська робота студента Тарана А. О. присвячена розробці системи управління контетом web-сайту.

Актуальність теми безсумнівна, адже в сучасному світі віртуальна присутність є важливим аспектом для бізнесу, освіти, громадських організацій та інших сфер. Система управління веб-контентом дозволяє легко та ефективно створювати та редагувати веб-сторінки, що важливо для забезпечення актуальної та привабливої інформації.

Студентом розроблено систему управління web-контентом, що надає функції створення, редагування, контролю та організації веб-сторінок. Створено і організувано його базову структуру, набір файлів і папок (просторів імен), обрано html шаблон макету для сайту.

В своїй магістерській роботі студент представив аналіз сучасного стану предметної області, що свідчить про його здатність здійснювати самостійний пошук та обробку інформації. Однак, аналіз може бути більш глибоким та об'єктивним, охоплюючи більш широкий спектр джерел. Слід, також, відзначити, що висвітлення методології могло б бути більш детальним, щоб забезпечити зрозумілість та повноту проведеного дослідження.

Проте вказані недоліки не впливають на позитивне враження від роботи. Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог до кваліфікаційних робіт магістрів і заслуговує на оцінку "задовільно". Студент Таран А.О. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 «Інженерія програмного забезпечення».

Рецензент,
завідувачка кафедри
комп'ютерної інженерії та
інноваційних технологій,
кандидат технічних наук, доцент  Л.Г. Йона

UNICHECK
by Turnitin

Ім'я користувача:
Анна Серединко

ID перевірки:
1016010074

Дата перевірки:
15.12.2023 17:46:31 MSK

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.12.2023 19:03:40 MSK

ID користувача:
100001433

Назва документа: Таран 2

Кількість сторінок: 43 Кількість слів: 6521 Кількість символів: 49221 Розмір файлу: 1.81 MB ID файлу: 1015695765

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

25.2%
Схожість

Найбільша схожість: 4.89% з Інтернет-джерелом (http://inmad.vntu.edu.ua/portal/static/63D5DA87-B50E-466C-999E-05B...

24.8% Джерела з Інтернету | 455 ..... Сторінка 45

0.31% Джерела з Бібліотеки | 3 ..... Сторінка 48

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 1

Підозріле форматування | 14 сторінок

РЕФЕРАТ

Текстова частина : 62 с., 37 рис., 1 табл., 1 додаток, 10 джерел.

Мета роботи є:

- вивчення базової структури сайту;
- тестування його працездатності на різних ПК користувачів;
- дослідження застосування різних мов програмування які використовуються під час створення веб-сторінки;
- основних правил та вимог до коректної роботи веб- сторінки та плагінів на ній.

У результаті роботи було проведено:

- дослідження середи розробки ASP.NET Core MVC;
- проаналізовані методи розробки;
- наведено приклади їх використання;
- роблено порівняння аналогів.

HTML, CSS, ВЕБ-ДОДАТОК, БАЗИ ДАНИХ, HTML-ЕЛЕМЕНТИ, ASP.NET. У данній роботі розглянуто розробку веб-сторінки, створену за допомогою мов Java Script та ASP.NET Core

ABSTRACT

Text part: 62 p., 37 figures, 1 table, 1 appendix, 10 sources.

he purpose of the work is:

studying the basic structure of the site;

testing its performance on different PCs of users;

research on the application of various programming languages used when creating a eb page;

basic rules and requirements for the correct operation of the web page and plugins on

s a result of the work, the following was carried out:

study of the ASP.NET Core MVC development environment;

analyzed development methods;

examples of their use are given;

a comparison of analogs is made.

TML, CSS, WEB APPLICATION, DATABASES, HTML ELEMENTS, ASP.NET.
his work deals with the development of a web page created using the Java Script and SP.NET Core languages

ЗМІСТ

ВСТУП

Метою роботи є розробка системи управління web-контентом, що надає функції створення, редагування, контролю та організації веб-сторінок.

Завдання - створити новий проект, організувати його базову структуру, створити набір файлів і папок (просторів імен) і також це вибрати html шаблон макет для сайту.

Для досягнення поставленої мети були сформульовані наступні завдання:

- аналіз предметної області. Щоб розробити програмну систему, яка принесе реальну користь певним користувачам, необхідно з'ясувати, які завдання вона повинна вирішувати для цих людей і якими властивостями володіти.

- аналіз існуючих аналогічних програмних продуктів. Будуть вивчені існуючі аналоги з метою виявлення переваг і недоліків - візуальних і функціональних.

- формування вимог до майбутньої системи і вибір платформи розробки. В результаті будуть детально з'ясовано вимоги до розроблюваної системі і вибрані відповідні кошти розробки.

- проектування загальної архітектури додатку. В процесі проектування архітектури буде встановлений набір компонентів, з яких буде побудована система і які вирішуються ними підзадачі в рамках загальних завдань системи, визначені загальні інтерфейси, через які вони будуть взаємодіяти і способи взаємодії компонентів один з одним.

- реалізація спроектованих підсистем. В ході реалізації буде розроблена система, яка задовольняє сформульованим раніше вимогам на основі спроектованої раніше архітектури.

- проведення тестування системи на прикладі реальної задачі. В процесі тестування буде проведена перевірка відповідності реалізованої системи поставленим раніше вимогам і задачам.

Всі ці завдання будуть розглянуті в наступних розділах.

При створенні CMS необхідно враховувати той момент, що можливо їй будуть користуватися люди, які не мають досвіду в роботі з Web-сайтами. Крім того, важливо забезпечити безпеку системи. Для цього треба визначити, в яких місцях система може стати вразливою, і створити способи захисту, які будуть доступні дуже вузькому колу користувачів.

1 Опис основних понять

Система управління контентом - це програма, призначена для роботи в Інтернеті або веб-додаток. Фраза «робота в Інтернеті» означає роботу CMS на хостингу, що надається провайдером послуг.

Хостинг - це віддалений веб-сервер з встановленою версією операційної системи і іншим супутнім програмним забезпеченням . Програмне забезпечення сервера призначене допомогти адміністратору сервера при організації клієнтських площадок і простому користувачеві при організації візуальних засобів для роботи з сайтом.

Веб-сервером називають комп'ютер, призначений для виконання на ньому сервісного програмного забезпечення, а також саме програмне забезпечення, встановлене на ньому і здійснює взаємодію з HTTP-протоколу з браузерами: приймає http-запити від браузерів і видає їм http-відповіді з html-сторінкою , зображенням, файлом, медіа-потоком або іншими даними .

В основному провайдером використовуються такі операційні системи, як FreeBSD, Debian, Windows Server та інші.

Як веб-сервера може використовуватися Apache або IIS і необхідні для роботи розширення, такі як бази даних, компілятори та інші. У нашому випадку, буде використаний сервер з операційною системою Windows Server і в якості веб-сервера - Microsoft IIS . Microsoft Internet Information Services є другим по полярності сервером в глобальній мережі інтернет.

В основі випуску IIS 7.5 лежить повністю модульний web-сервер, що включає більше 40 компонентів, які можна об'єднувати в компактні web-сервери, оптимізовані для необхідної ролі в топології програми. Ці компоненти створюються на основі нового шару розширюваності, що дозволяє розробникам розширювати або заміщати практично будь-яку функцію сервера в машинному коді .

Версія 7.0 пропонує розширюваність компонентів виконання, управління і робочих компонентів, полегшуючи створення комплексних рішень у відповідності з конкретними потребами.

У платформі IIS 7.5 в порівнянні з попередньою версією вирішуються багато проблем, пов'язаних з керованістю і експлуатацією сервера. Вона володіє принципово новою системою настройки, що забезпечує повністю делеговане управління вузлами. Нові інтерфейси API для управління і діагностичні компоненти роблять процедури розгортання, адміністрування і усунення неполадок сервера значно простіше і зручніше, ніж будь-коли раніше.

Web-сервер IIS підтримує кілька різних технологій створення web-додатків:

- ASP.NET - для систем Windows це основне, на сьогоднішній день, засіб створення web-додатків і web-служб. Підтримка ASP.NET вбудована в IIS починаючи з версії 6.0.
- ASP - застаріла технологія створення динамічних web-сторінок на основі сценаріїв. Входить в поставку IIS, починаючи з версії 3.0.
- CGI - стандартна технологія створення динамічних веб-сторінок.
- ISAPI - для Windows систем це найбільш потужна технологія, що надає повний доступ до всіх можливостей IIS.

За допомогою CG I і ISAPI до web-серверу IIS можуть підключатися сторонні кошти підтримки web-додатків, наприклад, PHP і Perl.

Одним з цікавих і перспективних нововведень в IIS, починаючи з версії 7.0, є пакет IIS Media Pack. Два додаткових безкоштовних модуля дозволять перетворити web-сервер в сучасний інструмент медіа-мовлення. Нові технології Microsoft дозволяють оптимізувати і грамотно управляти цифровим потоком медіа-даних. Сервер дозволяє виробляти мовлення даних в різних форматах: ASF, AVI, FLV, M4V, MOV, MP3, MP4, RM, RMVB, WMA, WMV.

Ще однією особливістю є вбудована підтримка нової технології Silverlight . Це технологія представлення даних в Інтернеті. Призначена для запуску на різних платформах. Вона дозволяє створювати насичені, візуально привабливі web-

сторінки, що працюють в різних браузерах, пристроях і настільних операційних системах (наприклад Apple Macintosh).

Поверх усього перерахованого раніше програмного забезпечення встановлюється так звана панель управління хостингом, що дозволяє працювати з веб-хостингом зі зручної графічної візуальної середовища. Так виглядає готовий до роботи сервер провайдера послуг.

1.1 Класифікація CMS систем

CMS системи діляться за місцем зберігання даних на:

- використовують бази даних
- використовують flat-файли.

База даних дозволяє зберігати величезні масиви різних даних, якими наповнюватиметься сайт за допомогою CMS, а також дані самої CMS. Переваги систем, що використовують бази даних полягають у великій кількості різних функцій, доповнень, розширень. Недоліком, хоч і незначним є складність в освоєнні таких систем, а також під час налаштування.

CMS, які не використовують бази даних для зберігання контенту сайту, використовують для цього різні типи файлів, такі як txt, xml та інші. Перевагами систем на файлах є легкість настройки системи, відсутність можливих проблем з сервером бази даних на хостингу, а так само можливість заощадити при виборі тарифного плану при покупці хостингу.

1.2 Програмне забезпечення рекомендоване при створенні веб-додатка

ASP.NET Core MVC - це легкий фреймворк з відкритим вихідним кодом, оптимізований для роботи з ASP.NET Core. ASP.NET Core MVC пропонує заснований на паттерне спосіб створення динамічних веб сайтів, де включено поділ відповідальності.

MySQL - це система управління базами даних (СКБД), яка поширюється як

вільне програмне забезпечення (користувачі мають право на необмежену установку, запуск, вільне використання).

База даних (БД) - набір деяких даних, які зберігаються в упорядкованій формі.

Програми які використовувалися : Microsoft Visual Studio 2019, Microsoft SQL Server Management Studio 18, SQL Server 2019 Configuration Manager.

2 ПОШУК І ПОРІВНЯННЯ АНАЛОГІВ

Щоб сформувати уявлення про ринок систем управління контентом, необхідні подібні зразки програмних продуктів. Вони будуть розглянуті далі в цьому розділі і в кінці опису аналогів буде зведена порівняльна таблиця.

2.1 WordPress

WordPress - вільно поширювана система управління вмістом сайту з відкритим вихідним кодом; написана на PHP; сервер бази даних - MySQL; випущена під ліцензією GNU GPL версії 2. Сфера застосування - від блогів до досить складних новинних ресурсів. Вбудована система «тем» і «плагінів» разом з вдалою архітектурою дозволяє конструювати проекти широкої функціональної складності.

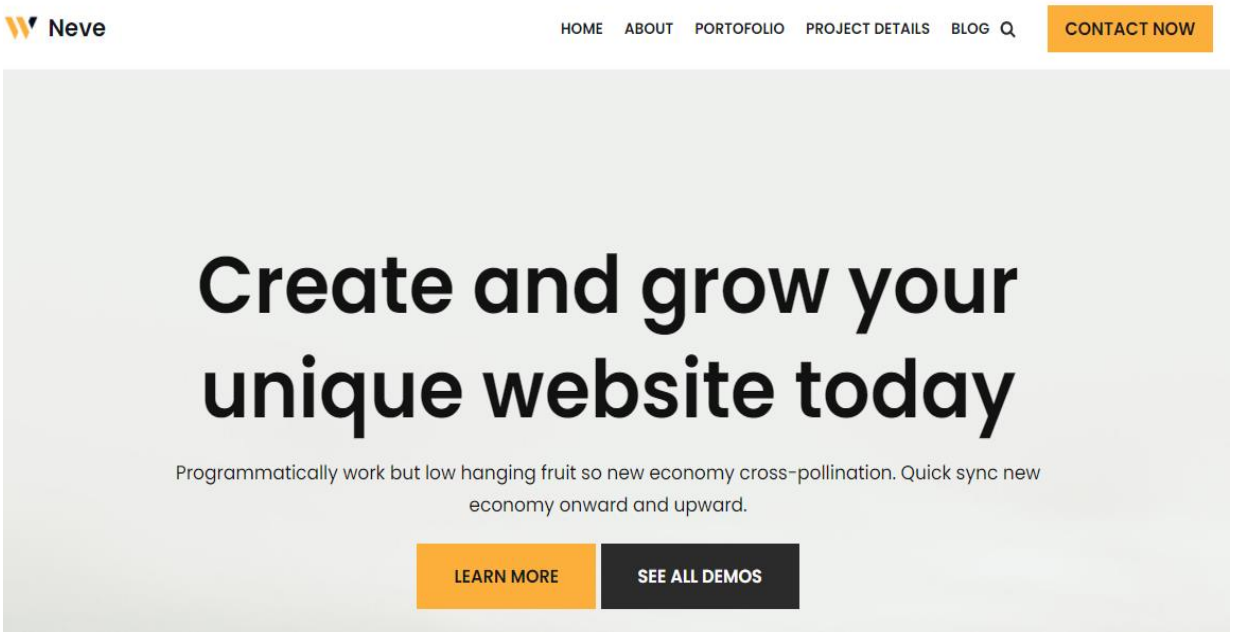


Рисунок 2.1 – Приклад сайту створеного за допомогою WordPress

Можливості WordPress

Інструменти публікації

WordPress дозволяє легко керувати своїм вмістом. Створювати чернетки, планувати публікацію та переглядати свої редакції публікацій. Можливість зробити свій вміст загальнодоступним або приватним та захистити повідомлення та сторінки паролем.

Управління користувачами

Не всі потребують однакового доступу до веб-сайту. Адміністратори керують сайтом, редактори працюють із вмістом, автори та співавтори пишуть цей вміст, а передплатники мають профіль, яким вони можуть керувати. Це дозволяє мати різноманітних співавторів на веб-сайті.

Управління медіафайлами

Можливість швидко та легко завантажувати зображення та медіа на WordPress. Легкість в завантажуванні медіафайлів на веб-сайт, та можливість додавання альтернативного тексту та підписів. Також є кілька інструментів для редагування зображень.

Повна сумісність із стандартами

Кожен фрагмент коду, створеного WordPress, повністю відповідає стандартам, встановленим W3C. Це означає, що веб-сайт буде працювати в сучасному браузері, зберігаючи при цьому сумісність із браузером наступного покоління.

2.2 Joomla!

Joomla! - система управління вмістом, написана на мовах PHP і JavaScript, що використовує в якості сховища бази даних СУБД MySQL або інші стандартні промислові реляційні СУБД. Є вільним програмним забезпеченням, поширюваним під ліцензією GNU GPL.

Рисунок 2.2 – Приклад сайту створеного за допомогою Joomla!

Можливості Joomla! -

Використання баз даних для зберігання вмісту.

Можливість налаштування структури сайту під визначені види вмісту: Новини, Обзори, Описання Продукції та прочее.

Можливість додавати нові функції та модулі на веб-сайті.

Возможность смены тем визуального оформления сайта.

Можливість виробничого розміщення на сторінках позицій для виводу модулів, що відображають визначену інформацію.

Можливість управління користувачами, призначення ім рівня доступу та прав на перегляд техніки або матеріалів.

Можливість змісту мови, на якій відображаються елементи управління сайтом.

• Можливість роботи на серверах під управлінням різних операційних систем: Linux, FreeBSD, сервер MacOSX, Solaris та AIX.

2.3 Drupal

Drupal - система управління вмістом, яка використовується також як каркас для веб-додатків, написана на мові PHP і використовує як сховище даних

реляційну базу даних. Drupal є вільним програмним забезпеченням, захищеним ліцензією GPL.

Рисунок 2.3 – Приклад сайту створеного за допомогою Drupal

Можливості Drupal

Фільтр

Зручний інструмент, що дозволяє сортувати матеріали за заданими критеріями. Наприклад, фільтрація відео по тривалості або сортування статей за кількістю переглядів. Ця функція часто зустрічається в магазинах, дозволяючи відображати товари певного кольору, розміру, вартості і так далі;

Особистий кабінет

Після реєстрації, у відвідувача повинен з'явитися аккаунт і вся інформація про нього. Щоб підвищити інтерес відвідувачів, можна присвоювати їм нагороди за активність, відображати дати відвідування, надати можливість змінювати аватар, публікувати статуси і відомості про себе;

Каталог

На всіх великих ресурсах існує каталог. Він може складатися з статей, новин, товарів, оглядів, фотографій, відео роликів та інших матеріалів. CMS для сайту легко справляється із завданням за допомогою розширення Taxonomy;

Блоки

Це відмінний спосіб розмістити рекламу, важливі новини, схожі матеріали або форму зворотного зв'язку. Адміністратор має блоки в будь-якому регіоні на свій розсуд. Друпал приділив цим елементам окремий пункт в адмінці. Через нього можна додавати, відключати, вибирати локацію або коригувати наявні блоки;

Уявлення

Унікальна система уявлень дозволяє ефективно групувати контент на сторінці. Адміністратор може розмістити списки в кілька колонок, вставляти фотографії серед тексту і поєднувати кілька елементів в одному матеріалі;

Синоніми

Щоб URL адреси складалися не з цифр, а з читаних слів, передбачений спеціальний інструмент. Він автоматично формує адреси, грунтуючись на заголовок статті. Це не тільки естетично, але і ефективно для СЕО просування;

Форма реєстрації

Вона може мати будь-яку структуру і змінюватися в залежності від побажань адміністратора. У число обов'язкових полів часто включається адреса електронної пошти, логін і пароль. При необхідності можна розширити список номером телефону, повним ім'ям і іншими відомостями;

Статистика

Власник ресурсу в режимі реального часу відстежує нові матеріали, коментарі, доступні оновлення, помилки і інші показники через адміністративну панель.

Таблиця 2.1 - Порівняння характеристик систем управління контентом

Характеристики	WordPress	Joomla!	Drupal
Легкість використання	+	+	-
Візуальний редактор	+	+	+
Плагіни	+	+	+
Шаблони	MySQL,MariaDB	MySQL, PostgreSQL, SLServer	MySQL,MariaDB, Percona Server, PostgreSQL,SQLite
База даних		+	
Безопасність	-	-	+

3 ФОРМУВАННЯ ВИМОГ ДО ПЗ

3.1 Діаграма варіантів використання UML

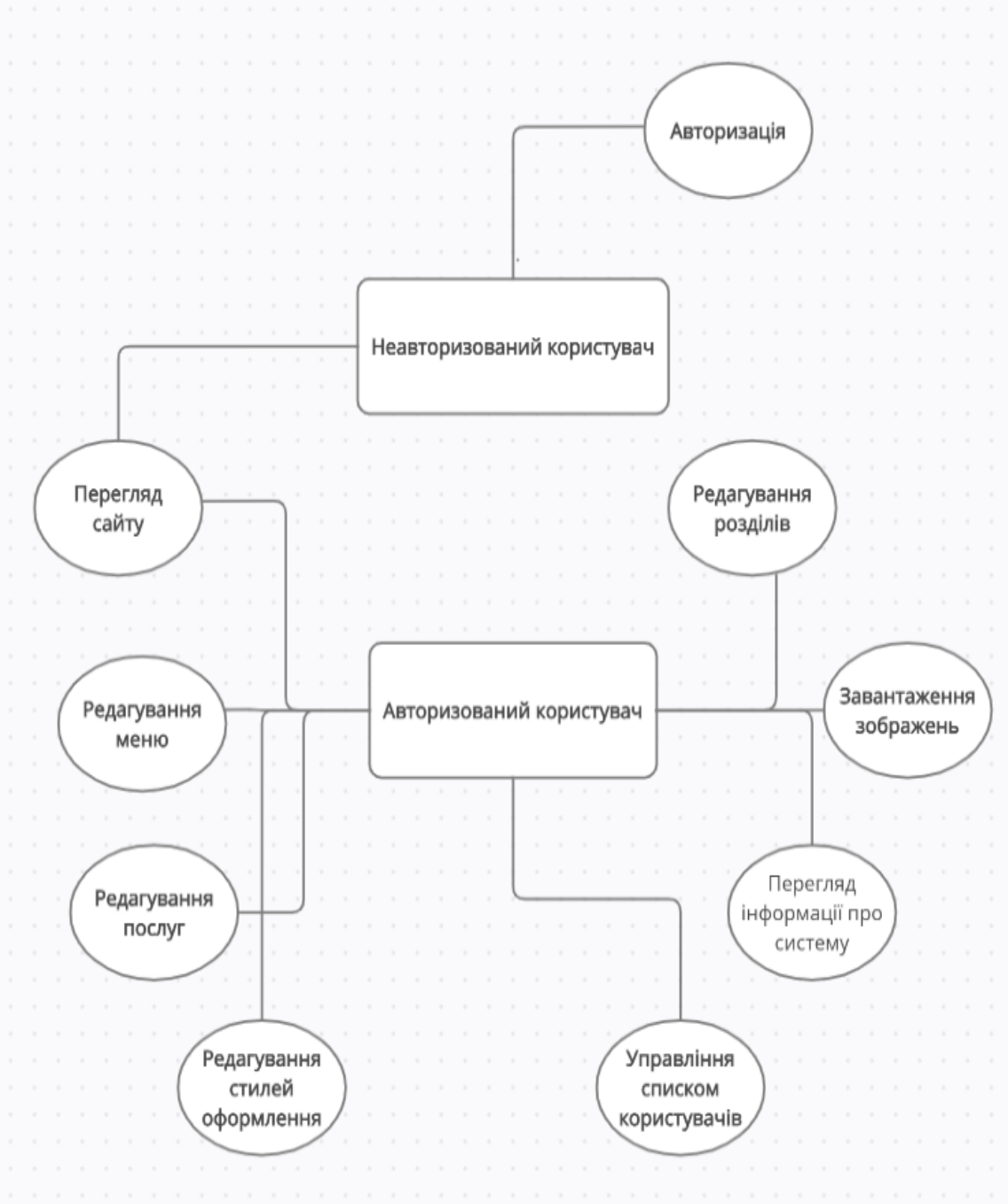


Рисунок 3.1 - Діаграма варіантів використання UML

3.2 Вимоги до проектування системи

Щоб визначити вимоги до проектування системи, описуються вимоги у вигляді UML діаграми варіантів використання.

Формуються формальні вимоги до системи.

У системі є 2 діючих особи:

Авторизований користувач

Неавторизований користувач

Кожен варіант використання відображає послідовність дій, які може здійснювати система у відповідь на дії користувачів.

Дії, які доступні неавторизованого користувачеві системи:

«Перегляд сторінок»

- Варіант використання починається після того, як користувач переходить на сторінку свого сайту.

- Додаток виводить користувачеві запитувані сторінки сайту, оформлені або з використанням стандартного шаблона оформлення, або з використанням шаблону оформлення, застосованого авторизованим користувачем системи.

- Варіант використання завершується.

«Авторизація»

Основний потік подій:

- Варіант використання починається після того, як неавторизований користувач переходить на сторінку адміністративної панелі системи.

- Додаток виводить форму авторизації і пропонує користувачеві ввести логін і пароль.

- Користувач вводить логін і пароль.

- Додаток перевіряє введені дані і підтверджує їх. Користувач стає авторизованим користувачем.

- Варіант використання завершується.

Альтернативний потік подій:

- Додаток буде сповіщати про те, що введені дані не вірні.

- Додаток пропонує ввести логін і пароль ще раз.

«Редагування меню»

- Варіант використання починається, коли користувач додає, змінює, видаляє пункти меню сайту.

- Додаток реєструє внесені зміни і зберігає в SQL-файли для подальшого

використання сайтом.

- Варіант використання завершується.

«Редагування стилів оформлення»

- Варіант використання починається, коли користувач застосовує, змінює або видаляє шаблони оформлення.

- Варіант використання завершується.

«Редагування розділів»

- Варіант використання починається, коли користувач змінює, додає або видаляє розділ.

- Додаток реєструє внесені зміни і зберігає в SQL-файли для подальшого використання сайтом.

- Варіант використання завершується.

3.3 Надання сайту відповідного вигляду

3.3.1 Основи CSS

CSS (Cascading Style Sheets) - мова таблиць стилів, який дозволяє прикріплювати стиль (наприклад, шрифти і колір) до структурованих документів (наприклад, документами HTML і додатків XML). Зазвичай CSS-стилі використовуються для створення і зміни стилю елементів веб-сторінок і призначених для користувача інтерфейсів, написаних на мовах HTML і XHTML, але також можуть бути застосовані до будь-якого виду XML-документа, в тому числі XML, SVG і XUL. Відокремлюючи стиль подання документів від вмісту документів, CSS спрощує створення веб-сторінок і обслуговування сайтів.

CSS підтримує таблиці стилів для конкретних носіїв, тому автори можуть адаптувати подання своїх документів до візуальних браузерам, слуховим пристроїв, принтерів, брайльовським пристроїв, кишеньковим пристроям і т.д.

Каскадні таблиці стилів описують правила форматування елементів за допомогою властивостей і допустимих значень цих властивостей. Для кожного елемента можна використовувати обмежений набір властивостей, інші

властивості не будуть чинити на нього ніякого впливу.

Оголошення стилю складається з двох частин: селектора і оголошення. В HTML імена елементів нечутливі до регістру, тому «h1» працює так само, як і «H1». Оголошення складається з двох частин: ім'я властивості (наприклад, color) і значення властивості (grey). Селектор повідомляє браузеру, який саме елемент форматувати, а в блоці оголошення (код в фігурних дужках) перераховуються форматує команди - властивості і їх значення.

Способи підключення CSS до документа [ред | правити код] Правила CSS пишуться на формальній мові CSS. Правила можуть розташовуватися як в самому веб-документі, зовнішній вигляд якого вони описують, так і в зовнішніх файлах, що мають формат CSS. Формат CSS - це текстовий файл, в якому міститься перелік правил CSS і коментарів до них. Стиль CSS можуть бути підключені або впроваджені в описуваний ними веб-документ чотирма способами:

Коли опис стилів знаходиться в окремому файлі, воно може бути підключено до документа за допомогою елемента <link>, включеного в елемент <head>:

```
<!DOCTYPE html>
<html>
   <head>
      .....
      <link rel="stylesheet" href="style.css">
   </head>
   <body>
      .....
   </body>
</html>
```

Рисунок 4.1 - Спосіб підключення CSS №1

Коли файл стилів розміщується окремо від батьківського документа, він може бути підключений до документа інструкцією @import в елементі <style>:

```
<!DOCTYPE html>
<html>
   <head>
      .....
      <style media="all">
         @import url(style.css);
      </style>
   </head>
</html>
```

Рисунок 4.2 - Спосіб підключення CSS №2

Коли стилі описані всередині документа, вони можуть бути включені в елемент <style>, який, включається в елеменет <head>:

```
<!DOCTYPE html>
<html>
   <head>
      .....
      <style>
         body {
            color: red;
         }
      </style>
   </head>
   <body>
      .....
   </body>
</html>
```

Рисунок 4.3 - Спосіб підключення CSS №3

Коли стилі описані в тілі документа, вони можуть розташовуватися в атрибутах окремого елемента :

```
<!DOCTYPE>
<html>
   <head>
      .....
   </head>
   <body>
      <p style="font-size: 20px; color: green; font-family: arial, helvetica, sans-serif">
         .....
      </p>
   </body>
</html>
```

Рисунок 4.4 - Спосіб підключення CSS №4

3.3.2 Основи MySQL

MySQL це система керування базами даних з відкритим вихідним кодом (СУРБД) з моделлю клієнт-сервер. СУРБД - це програмне забезпечення або служба, яка використовується для створення та управління базами даних на основі реляційної моделі.База даних - це просто набір структурованих даних. Наприклад, коли ви робите Селфі: ви натискаєте кнопку і фотографуєте себе. Ваша фотографія - це дані, а галерея вашого телефону - це база даних. База даних - це місце, в якому зберігаються дані. Слово «реляційний» означає, що дані, що зберігаються в наборі даних, організовані у вигляді таблиць. Кожна таблиця пов'язана в деякому роді. Якщо програмне забезпечення не підтримує реляційну модель даних, просто назвіть її СУБД.Компьютери, які встановлюють і запускають програмне забезпечення СУРБД, називаються клієнтами. Коли їм потрібно отримати доступ до даних, вони підключаються до сервера СУРБД. Це система «клієнт-сервер».

MySQL є одним з багатьох варіантів програмного забезпечення СУРБД. Вважається, що СУРБД і MySQL однакові через популярність MySQL. Назвіть кілька великих веб-додатків, таких як Facebook, Twitter, YouTube, Google і Yahoo! всі використовують MySQL для зберігання даних. Хоча спочатку він створювався для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu.

Одне або декілька пристроїв (клієнтів) підключаються до сервера через певну мережу. Кожен клієнт може зробити запит з графічного інтерфейсу користувача (GUI) на своїх екранах, і сервер видасть бажаний результат, якщо обидва кінці розуміють інструкцію. Не вдаючись в технічні аспекти, основні процеси, що відбуваються в середовищі MySQL, однакові:

MySQL створює базу даних для зберігання і управління даними, що визначають відносини кожної таблиці.

Клієнти можуть робити запити, вводячи певні команди SQL на MySQL.

Додаток сервера відповість запитаної інформацією і з'явиться на стороні клієнта.

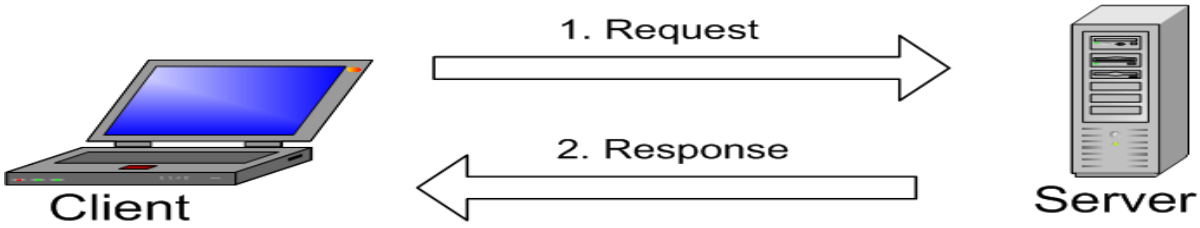


Рисунок 4.6 – Структура клієнт – сервер

3.3.3 Основи ASP.NET Core MVC

ASP.NET - це частина веб-платформи для розробки серверних веб-додатків. Платформою для ASP.NET є .NET Framework, що дозволяє використовувати функції .NET.Framework в додатках ASP.NET.

ASP.NET MVC 3 - це програмне забезпечення для створення веб-додатків, на основі шаблону MVC. Переваги такого підходу будуть розглянуті пізніше.

Переваги ASP.NET MVC:

Можливість розширення і доповнює платформа. Компоненти платформи ASP.NET MVC можна легко замінити або налаштувати. Розробник може підключати власний механізм уявлень, політику маршрутизації URL-адрес і інші компоненти.

Розширена підтримка маршрутизації ASP.NET. Цей потужний компонент зіставлення URL-адрес дозволяє створювати додатки з зрозумілими URL-адресами, які можна використовувати в пошуку. URL-адреси не повинні містити розширення імен файлів і призначені для підтримки шаблонів іменування URL-адрес, які забезпечують адресацію, оптимізовану для пошукових систем (SEO).

Підтримка існуючих функцій ASP.NET. ASP.NET MVC дозволяє використовувати такі функції, як перевірка справжності за допомогою форм і Windows, перевірка достовірності за URL-адресою, членство і ролі, кешування виведення і даних, управління станом сеансу і профілю, спостереження за працездатністю, система конфігурації і архітектура постачальника [25].

3.3.4 Razor - движок уявлень

У ASP.NET 3 використовується движок уявлень Razor. До його ключових можливостей можна віднести:

Зрозумілий і стислий синтаксис, що значно знижує обсяг коду View

Простота у вивченні, це пов'язано з тим, що він базується на існуючих мовах C # і Visual Basic

Підтримка підсвічування синтаксису в Visual Studio [27].

3.3.5 MVC (Model-View-Controller)

Зараз популярний шаблон проектування MVC [28]. Він служить для відділення логіки додатка від призначеного для користувача інтерфейсу. Прояснимо що таке шаблон проектування.

Це набір типових рішень проектування, каркас архітектури або її фрагмента. Якщо бібліотека - це пакет повторно використовуваного коду, то шаблон проектування - це пакет повторно використовуваних рішень.

Шаблон MVC дозволяє розділити дані, подання та обробку дій користувача на три окремих компоненти:

Модель (Model). Модель надає дані (зазвичай для View), а також реагує на запити (зазвичай від контролера), змінюючи свій стан.

Представлення (View). Відповідає за відображення інформації (призначений для користувача інтерфейс).

Поведінка (Controller). Інтерпретує дані, введені користувачем, і інформує модель і уявлення про необхідність відповідної реакції.

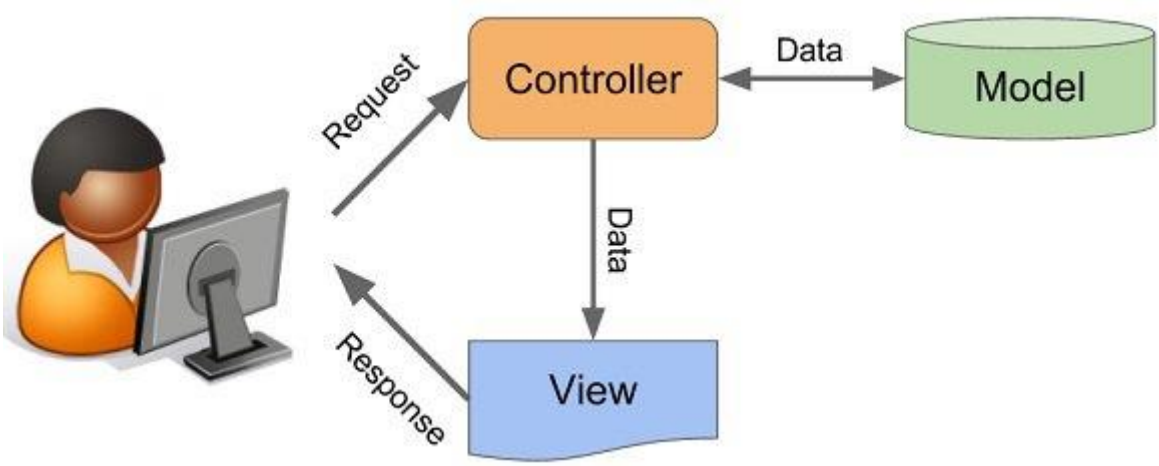


Рисунок 4.7 – Відносини між компонентами .

Переваги розробки на платформі MVC

Чіткий поділ логічних шарів. Відділення подання від контролера надає можливість простої заміни движка уявлення без модифікації коду контролера, незалежність від реалізації моделі дозволяє описати тільки інтерфейси об'єктів і підміняти реалізацію при необхідності.

Повний контроль над кодом розмітки. Повний контроль над розміткою може бути особливо важливий, якщо в веб-додатку використовується код, що працює на стороні клієнта в браузері користувача

Логічне поділ функціональності. У веб-додатках MVC Framework є чіткий поділ на дії контролерів, і кожна дія має власних URI.

Красиві URL-адреси. Це досягається зарахунок використання гнучкої системи маршрутизації.

Прозорий процес обробки запиту. Процес обробки запиту легко простежується, оскільки обробка запиту розділена на невелику кількість дуже простих кроків.

Можливість розширення. У MVC Framework можливе використання різних бібліотек для обробки подань, власних алгоритмів створення об'єктів контролерів і розширення механізмів функціонування компонентів бібліотеки

Простота автоматичного тестування. Наприклад, при тестуванні коду логіки додатка (контролера), можна підмінити модель версією, створеної спеціально для тестування і надає мінімальний набір даних.

Недоліки:

Високий "поріг входу" в технологію

Необхідні глибокі знання HTML, CSS, JavaScript

4 ПРОЕКТУВАННЯ АРХІТЕКТУРИ

4.1 Прототипування графічного інтерфейсу користувача

Прототипування допомагає в процесі створення якісних користувальницьких інтерфейсів. Створення прототипів дозволяє вносити швидкий огляд багатьох ідей і виявляти недоліки в проектованому додатку на самому ранньому етапі розробки. Прототипи можуть бути представлені в різноманітних формах: від паперових начерків, до дизайну, створеного в професійному пакеті, який виглядає дуже близько до бажаного результату. В даному проекті для створення прототипу був використаний сайт для побудови діаграм та блок схем.

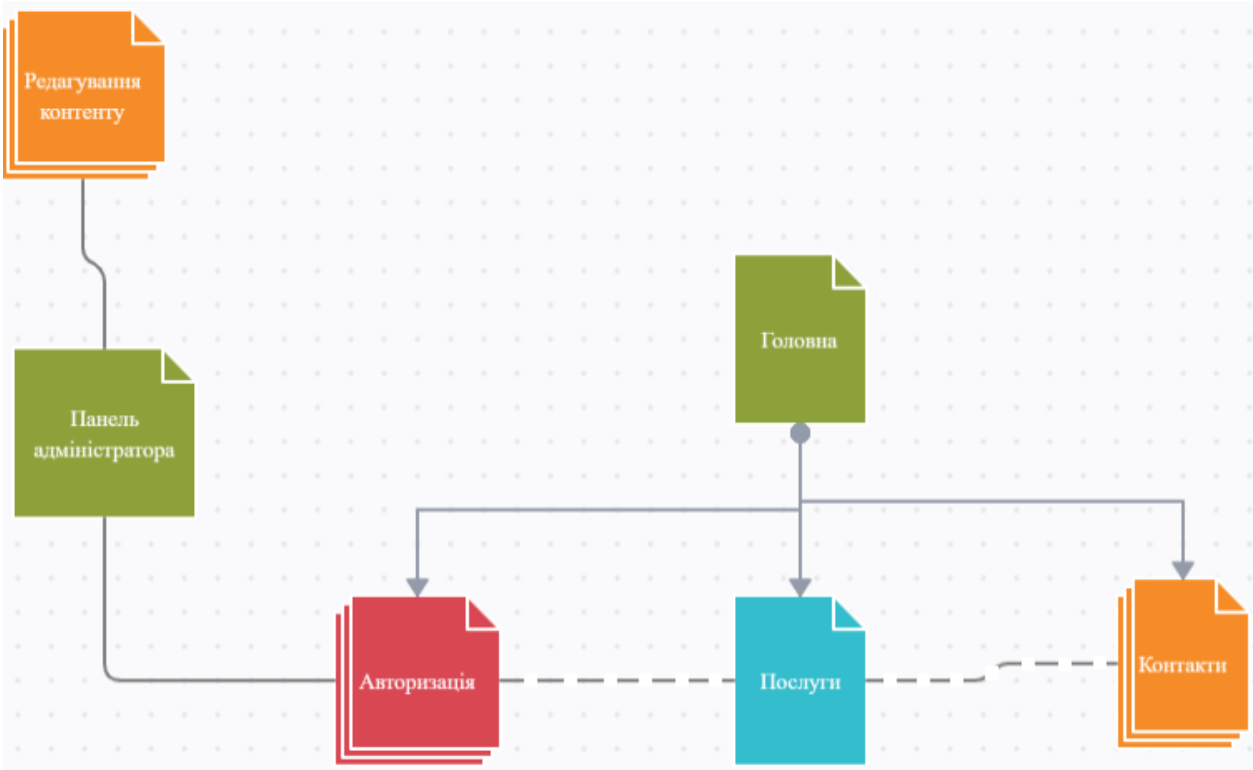


Рисунок 5.1 – Прототип інтерфейсу

4.2 Загальна архітектура програмного засобу

ASP.NET MVC 3] використовує шаблон Model-View-Controller, який широко застосовується в веб-програмуванні.

Шаблон MVC відокремлює логіку моделі даних додатки від логіки уявлення і бізнес-логіки. Крім того, в платформі ASP.NET MVC логічне поділ реалізується фізично в структурі проекту, в якій контролери та подання зберігаються в папках, для визначення відносин між якими використовуються певні правила іменування.

Для того щоб зрозуміти принципи роботи компонентів MVC-додатки, розглянемо схему обробки запитів MVC-додатків.

Життєвий цикл запиту для MVC-додатки

HTTP-запит що відправляється до веб-сервера передається середовищі виконання ASP.NET, яка ініціалізує інфраструктуру MVC Framework і передає запит для обробки компоненту маршрутизації. На підставі таблиці маршрутизації, що завантажується при запуску веб-додатки, модуль маршрутизації визначає імена контролера і методу контролера, який повинен обробити запит, а також параметри запиту, які повинні бути передані контролера. Після цього генерується контекст запиту, що містить параметри запиту та середовища виконання програми (такі як URL-запиту, IP-адреса клієнта і сервера і т.п.), створюється екземпляр класу контролера і йому передається управління шляхом виклику відповідного методу класу контролера - дії контролера в термінах MVC .

Метод контролера на підставі параметрів запиту виконує деяку логіку, вибирає уявлення, яка має бути відображена користувачеві, і передає управління механізмом генерації розмітки (движком уявлення в термінах MVC), який вже відображає уявлення.

Для обміну даних між поданням і контролером використовується спеціальна колекція ViewData - є основною сполучною ланкою між контролером і поданням.

Після того як розмітка була згенерована движком уявлення, веб-сервер повертає її в якості відповіді користувачу по протоколу HTTP. На цьому життєвий цикл обробки запиту MVC-додатком закінчується.

4.3 Опис створення сайту

Основні деталі веб-додатка

В даному веб-додатку можна буде розширювати функціонал, наприклад додавати новини, послуги, ціни і т.д.

Функціонал на головній сторінці

Для того щоб була можливість редагування буде реалізована панель адміністратора.

У даній панелі користувач зможе змінювати або наповнювати сайт інформацією, редагувати всі пункти меню, основні розділи і також редагувати SEO інформацію для успішного просування в інтернеті тобто заголовок опис і ключові слова.

Функціонал для інших розділів: послуги, назви послуги, короткий опис послуги, повний опис послуги, завантажити нову титульну картинку і SEO теги сайтую

Список технологій які використовувалися для того щоб створити цю програму -

Середовище розробки -Visual Studio

Сервера баз даних - Microsoft SQL Server.

Міграції для автоматизації операцій з базою даних (оновлювати таблиці, створювати).

Технологія Identify - для можливості виконувати операції з користувачами в додатку, такі як аутентифікація і авторизація, додавати ролі користувачів, адміністраторів.

Тип додатка ASP.NET Core MVC 3.1

Для клієнтських технологій використовувався макет який відповідає стандарту html5.

Для мови розмітки CSS використовувався препроцесор SASS, який привносить в нього можливості об'єктноорієнтованої мови програмування такі як- створення змінних, функцій.

4.4 Реалізація проекту

Макет треба розділити на файли в певних папках в проекті.

Також реалізується візуальний текстовий редактор для того щоб в панелі адміністратора можна було редагувати контент для сайту. Цей редактор буде автоматично генерувати html розмітку.

У порожньому проекті потрібно створити базовий набір файлів і папок.

Створюється директорія в root (це стандартна для ASP.NET Core директорія) де будуть зберігатися статичні файли стилів, зображень, скриптів і т.д. У цій директорії потрібно створити папки CSS для стилів, images для зображень, scripts і папка для SASS стилів.

Також створюється директорія Areas (області) це частина в рамках всього програми і в відгалуженні цій галузі можуть бути представлені окремі набори контролерів, уявлень.

Область адмін в якій будуть виконуватися всі операції по адмініструванню сайту.Так само в області Admin створюються папки Controllers і папка Views (уявлення і контролери для основної частини програми)

Створюється директорія для доменної моделі (Domain), в цій папці буде зберігається вся доменна модель (послуги, текстові поля, контекст підключення до бази даних).

Створюються моделі рівня додатки, view компоненти і директорія де буде вся логіка для послуг у програмі. В папці Views розміщується html-макет.

В якості редактора тексту виступає Ckeditor. Функціонал цього редактора досить простий

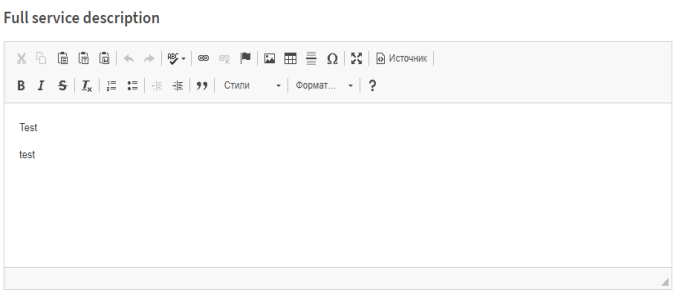


Рисунок 6.2 – Інтерфейс Ckeditor

4.5 Початок виконання проекту

Перейшовши в робочий простір створюється файл «_Layout» з розміткою html.

Це майстер сторінка, її потрібно визначити, для цього створюється файл «_ViewStart» і для визначення змінних і tag-helpers створюється файл «_ViewImports».

Даний етап потрібен для того щоб не дублювався код в уявленнях (Views), в проекті використовувався стандарт tag-helper з пакету Microsoft AspNetCore Mvc TahHelper. Це збірка для вбудованих допоміжних функцій тегів ASP.NET Core.

Директива @addTagHelper робить допоміжні функції тегів доступними в уявленні. У цьому випадку файл уявлення має вигляд «pages / _ViewImports. Cshtml», який за замовчуванням успадковується всіма файлами в папці pages і вкладеними папками. забезпечення доступності допоміжних функцій тегів.), який вказує, що всі допоміжні функції тегів у зазначеній збірці (Microsoft. AspNetCore. MVC. TagHelpers) будуть доступні для кожного файлу уявлення в каталозі views або підкаталозі уявлень. Перший параметр після директиви @addTagHelper вказує завантажуються допоміжні функції тегів (ми використовуємо "*" для всіх допоміжних функцій тегів), а другий параметр "Microsoft.AspNetCore.Mvc.TagHelpers" вказує збірку, що містить допоміжні функції тегів.

На даному етапі для зручності подальшої розробки можна розбити код на частини (Footer, Header, Sidebar і т.д) робиться це для того щоб виключити дублювання коду і щоб було зручніше працювати з кодом програми.

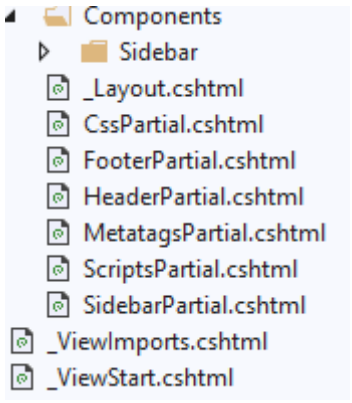


Рисунок 6.3 –Основні файли розмітки веб-додатка

Щоб уникнути зайвих запитів та завантажень файлу стилів який не змінювався используеться tag-helper asp-append-version = "true" /

У файлі для метатегов для того що б заголовок був динамічним потрібно довай таку конструкцію

```
@if (ViewBag.Title != null)
{
    <title>@ViewBag.Title</title>
}
else
{
    <title>@Config.CompanyName</title>
}
@if (ViewBag.Description != null)
{
    <meta name="description" content="@ViewBag.Description" />
}
@if (ViewBag.Keywords != null)
{
    <meta name="keywords" content="@ViewBag.Keywords" />
}
```

Рисунок 6.3 – Файл Metatags

Всі скрипти які додавалися до шаблону для бистрішої загрузки стискаються спеціальним плагіном в Visual Studio або в будь-який призначеної для цього програмі

І так само додається tag-helper "asp-append-version =" true "" Таким чином знижується кількість запитів до сервера.

На даному етапі до майстер-сторінці підключаються часткові файли які були створені

```
@await Html.PartialAsync("MetatagsPartial.cshtml")
@await Html.PartialAsync("CssPartial.cshtml")
ead>
dy class="is-preload">
<div id="page-wrapper">

    <!-- Header -->
    @await Html.PartialAsync("HeaderPartial.cshtml")

    <!-- Main -->
    <section class="wrapper style1">
        <div class="container">
            <div class="row gtr-200">
                <div class="col-8 col-12-narrower">
                    <div id="content">
                        <article>
                            <header>

                            </header>


                            @RenderBody()
                            </article>
                    </div>
                </div>
                @await Html.PartialAsync("SidebarPartial.cshtml")
            </div>
        </div>
    </section>

</div>
<!-- Footer -->
@await Html.PartialAsync("FooterPartial.cshtml")

<!-- Scripts -->
@await Html.PartialAsync("ScriptsPartial.cshtml")
```

Рисунок 6.5 – Підключення часткових файлів

На майстер сторінці вибирається місце де буде відображатися динамічний контент, для цього використовується метод @ Render.Body.

```
</header>


@RenderBody()
</article>
```

Рисунок 6.6 – Реалізація динамічного контенту

Тут буде динамічно підставлятися контент в залежності від того на якій сторінці ми знаходимося.

Компіляція стилів за допомогою SASS:

SASS розшифровується як Syntactically Awesome Style Sheets - якщо

перекладати дослівно, то це звучить як: «Синтаксично приголомшливі таблиці стилів».

Ця технологія була придумана і втілена Гемптоном Катлін (Hampton Catlin). SASS маніпулює CSS-правил, використовуючи змінні, так звані міксини (mixins), успадкування і вкладеність.

Вихідні і скомпільовані файли мають розширення .sass і .scss, відповідно. Вихідні тексти переводяться в добре відформатований CSS-код за допомогою командного рядка або веб-плагіна.

SASS спрощує написання CSS-коду і дозволяє динамічно їм маніпулювати. Це відмінний спосіб створення більш функціональних CSS-кодів, який дозволяє прискорити виконання щоденної роботи веб-розробників і дизайнерів.

Необхідний набір пакетів

CodeGenerationDesign - цей пакет потрібен для того що у нас була можливість створювати здійснював студії мав можливість генерувати для нас заготовки класів контролерів і уявлень.

EntityFramework - система для роботи з базою даних і з усіма сутностями
Для сервера баз даних буде використовуватися Microsoft SQL Server
також цей пакет буде нам потрібен в процесі роботи з доменною моделлю контексту бази даних.

Панель адміністратора - це захищена область (не всі користувачі зможуть туди зайти) і для того щоб реалізувати подібний функціонал потрібно використовувати технологію Identity.

5 СТВОРЕННЯ ФУНКЦІОНАЛУ

5.1 Сервіси функціоналу

Створення функціоналу починається з файлу Startup.cs, в даному класі потрібно прописати сервіси.

Технологія ASP.NET Core має модульну архітектуру і весь необхідний функціонал буде підключатися за допомогою сервісів, це означає те що при створенні нової програми увесь потрібний функціонал доведеться підключати самостійно через сервіс. Це один з плюсів бо тут підключається тільки той функціонал який нам потрібен через сервіс

В першу чергу поключается сервіс для того щоб була підтримка контролерів і уявлень

```
public void ConfigureServices(IServiceCollection services)
{

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


    services.AddTransient<ITextFieldsRepository, EFTextFieldsRepository>();
    services.AddTransient<IServiceItemsRepository, EFServiceItemsRepository>();
    services.AddTransient<DataManager>();


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


    services.AddIdentity<IdentityUser, IdentityRole>(opts =>
    {
        opts.User.RequireUniqueEmail = true;
        opts.Password.RequiredLength = 6;
        opts.Password.RequireNonAlphanumeric = false;
        opts.Password.RequireLowercase = false;
        opts.Password.RequireUppercase = false;
        opts.Password.RequireDigit = false;
    }).AddEntityFrameworkStores<AppDbContext>().AddDefaultTokenProviders();


    services.ConfigureApplicationCookie(options =>
    {
        options.Cookie.Name = "myCompanyAuth";
        options.Cookie.HttpOnly = true;
        options.LoginPath = "/account/login";
        options.AccessDeniedPath = "/account/accessdenied";
        options.SlidingExpiration = true;
    });


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

Рисунок 7.1 – Startup.cs

Використовуючи колекцію сервісів додається підтримка контролерів і уявлень (MVC)

Якщо користувач знаходиться в оточенні Development (в процесі створення нашого сайту) то йому потрібно знати які помилки виникають в процесі роботи нашого веб-додатка.

Додається система маршрутизації, підключаються Endpoint (маршрути) і підтримка статичних файлів (зображення, файли скрипти різні документи).

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

Рисунок 7.2 – Маршрути

На даному етапі необхідно створити файл (appsettings.json) з необхідними налаштуваннями веб-додатка, тут буде записаний рядок підключення до бази даних, а також інформація про проект (назва проекту контакти і email)

```
'Project": {
  "ConnectionString": "Data Source=(local)\\DIPLOM; Database=Diploma;
  "CompanyName": "My Diploma",
  "CompanyPhone": "+380000000000",
  "CompanyEmail": "contact@mycompany.com"
```

Рисунок 7.3 – Підключення до бази даних

```
public abstract class EntityBase
{
    Ссылок: 0
    protected EntityBase() => DateAdded = DateTime.UtcNow;

    [Required]
    Ссылок: 18
    public Guid Id { get; set; }

    [Display(Name = "Name (heading)")]
    Ссылок: 15
    public virtual string Title { get; set; }

    [Display(Name = "Short description")]
    Ссылок: 7
    public virtual string Subtitle { get; set; }

    [Display(Name = "Full description)")]
    Ссылок: 13
    public virtual string Text { get; set; }

    [Display(Name = "Title picture")]
    Ссылок: 6
    public virtual string TitleImagePath { get; set; }
```

Рисунок 7.4 – Перелік властивостей

В папці domain створюється простір entities (сутності) і базовий клас для всіх сутностей.

Визначається ряд властивостей: айді (первинний ключ) типу guId і також відзначається що значення для цієї властивості обов'язково, тому що запис без ідентифікатора існувати не може.

Також тут прописується набір властивостей: назва послуги короткий опис, повний опис, титульна картинка.

Щоб зв'язати доменні об'єкти потрібно створити контекст бази даних для веб-додатка в якому буде зв'язок логіки роботи додатка з базою даних.

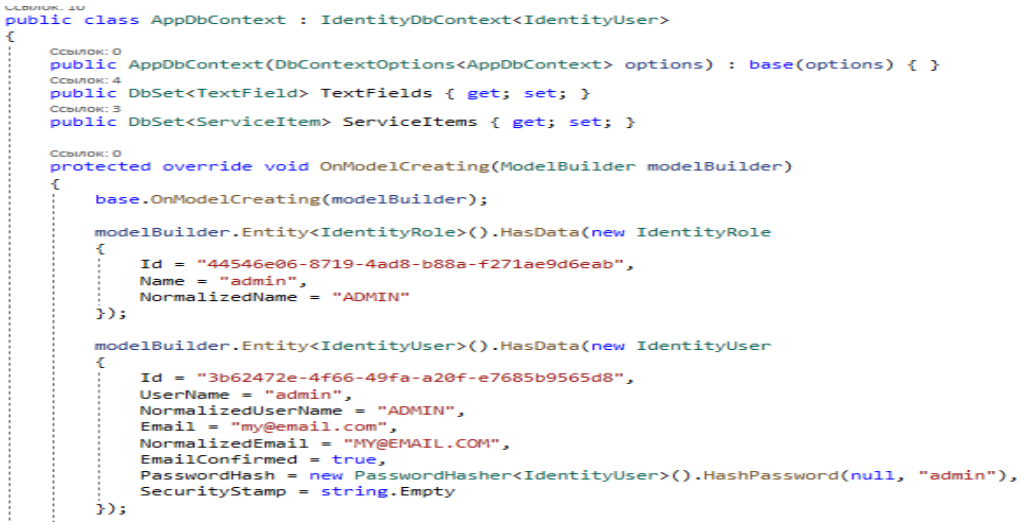


Рисунок 7.5 – Контекст бази данних

Контекст бази даних був створений для того щоб всі операції в коді сайту виконувалися через контекст бази даних (технологію entity framework) і транслювалися в SQL запити на сервер з базою даних.

Для того щоб виробляти операції над даними класами наприклад робити вибірку даних об'єктів з бази даних, записувати дані в базу даних оновлювати і видаляти використовується наступний функціонал.

Клас repository для маніпуляції над об'єктами. Тут будуть знаходитися інтерфейси і відповідні їм класи.

Тут визначаються інтерфейси для доменних об'єктів які реалізують потрібне нам поведінку наприклад інтерфейс і ITextFieldsRepository містить такі методи як: зробити вибірку всіх текстових полів, вибрати текст поля за ідентифікатором, зберегти зміни в базу даних і видалити текстове поле.

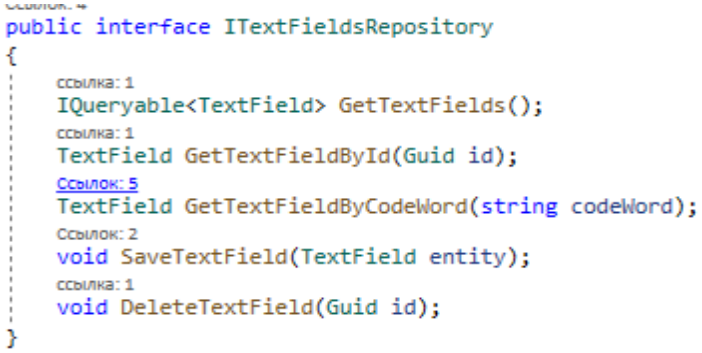


Рисунок 7.6 – Інтерфейси доменних об'єктів

5.2 Створення міграції

Для того щоб зв'язати модель доменну і базу даних необхідно створити міграцію і оновити базу даних.

Міграція це інструмент який дозволяє відстежувати зміни в базі даних відповідно до зміни коду базі, наприклад якщо в послугах з'явитися нова властивість використовуючи міграцію не потрібно вручну звертатися до бази даних і вручну створювати для таблиці відповідну колонку. Ці зміни будуть автоматично додані в базу даних.

Створюється міграція через інструмент powershell (також можна використовувати командний рядок).

У вікні powershell треба перейти в папку де знаходиться проект і виконується наступна команда.

```
dotnet ef migrations add _initial
Build started...
Build succeeded.
info: Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 3.1.1 initialized 'AppDbContext' using provider 'Microsoft.
th options: None
Done. To undo this action, use 'ef migrations remove'
```

Рисунок 7.7 – Створення міграції

Міграція створена і у менеджері баз даних можна побачити що ось з'явилася база даних проекту і в ній все таблиці які були прописані в конфігураційних файлах.

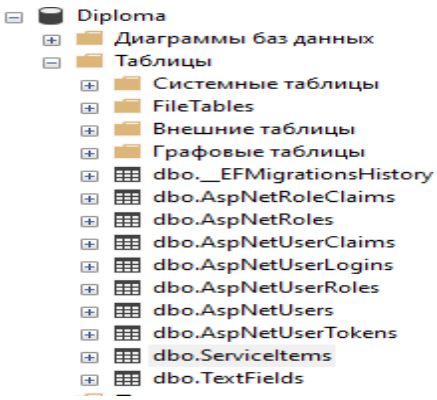


Рисунок 7.8 – Таблиці у базі данних

5.3 Реалізація та керівництво користувача

Опис сторінок додатку

При відкритті веб-додатка у вікні браузера, користувач повинен побачити головну веб-сторінку свого сайту зі стандартним шаблоном оформлення.

Залежно від статусу користувача - авторизований він чи ні, йому доступний ряд сторінок. Для користувача, який не пройшов авторизацію, доступні сторінки:

- «Авторизація»
- Перегляд сторінок сайту

Щоб скористатися функціоналом настройки сайту, необхідно пройти процедуру авторизації. Для авторизованого користувача доступні сторінки:

Перегляд сторінок сайту

«Головна»

«Основні налаштування»

«Редактор меню»

«Розділи»

«Головна»

На даній сторінці можна побачити привітання поточного користувача. Вона є початковою сторінкою додатка. На сторінці відсутній будь-який функціонал.

«Авторизація»

Для переходу в рядок адміністратора необхідно ввести в адресному рядку браузера "адмін"

Логін. Поле обов'язково для заповнення, служить для авторизації в системі.

Пароль. Поле обов'язково для заповнення, забезпечує конфіденційність користувальницької облікового запису.

Якщо процедура авторизації пройшла успішно (задані логін і пароль існують), користувач стає авторизованим і перенаправляється на вітальну сторінку XCMS «Головна». Йому стають доступні сторінки редагування основних

параметрів, додавання, видалення і редагування пунктів меню, категорій і текстів статей, користувачів, стилів оформлення і вихід.

При невдалій спробі авторизації видається повідомлення про помилку з зазначенням причини помилки.

"Основні налаштування"

Сторінка з головними налаштуваннями сайту. Форма сторінки містить:

Редагування сторінок: головна,послуги,контакти. Редагує текст, який буде виводитися сторінках сайту.

Додати послугу. Додає назву послуги, опис, повний опис, та зображення.

5.4 Публікування проєкту

Після завершення роботи над додатком додаток його можна опублікувати, щоб воно стало доступне широкому колу користувачів. Як правило, для хостингу додаток буде застосовуватися один із зовнішніх хостингів, список яких можна подивитися тут. В даному ж випадку розглянемо основні моменти публікації і розгортання програми на локальному комп'ютері.

Традиційно веб-сервер IIS (Internet Information Services) застосовувався для розгортання веб-додатків. Для хостірованія веб-додатків ASP.NET Core також може застосовуватися IIS, тільки на відміну від попередніх версій ASP.NET тепер його роль зводитиметься до проксі-сервера. Хостіроване додатків ASP.NET Core на IIS відбувається за допомогою нативного модуля AspNetCoreModule, який налаштований таким чином, щоб перенаправляти запити на веб-сервер Kestrel. Цей модуль управляє запуском зовнішнього процесу dotnet.exe, в рамках якого хостіруется додаток, і перенаправляє всі запити від IIS до цього хостірующему процесу.

При розробці в Visual Studio публікувати додатки дуже легко - середовище розробки має для цього весь необхідний інструментарій. Так, візьмемо який-небудь проект і в Visual Studio натиснемо на нього правою кнопкою миші і в контекстному меню виберемо пункт Publish - і відкриється вікно публікації програми.

Є кілька варіантів публікації:

Microsoft Azure App Service: публікація в хмарі Azure

IIS, FTP, etc: публікація через FTP

Folder: публікація у вигляді окремого пакета в файлової системі поточної робочої машини

Import Profile: імпорт профілю, який містить налаштування публікації

Microsoft Azure Virtual Machines: публікація в хмарі Azure, в порівнянні з першою опцією володіє великими можливостями по управлінню інфраструктурою розгортання

В даному випадку вибирається опція Folder для створення пакета для публікації в файлової системі. І також вкажемо шлях, по якому буде знаходитися пакет. У моєму випадку це каталог "C: \ CoreApp". І в кінці натиснемо на кнопку Publish.

Далі відкриється вікно, де будуть оображаться вибрані настройки конфігурації, і для продовження публікації натиснемо в ньому кнопку Publish:

І після закінчення публікації по зазначеному з'являться опубліковані файли.

Налаштування IIS

Перш за все нам треба включити функціональність Web Server (IIS) і налаштувати ролі сервера. Для цього перейдемо по шляху Панель управління -> Програми та засоби -> Включення або відключення компонентів Windows. У списку компонентів знайдемо Служби IIS (Internet Information Services) і відзначимо її:

Потім необхідно встановити спеціальний пакет .NET Core Windows Server Hosting. Його можна знайти, перейшовши на сторінку https://www.microsoft.com/net/download/all. Далі на цій сторінці треба вибрати потрібну версію .NET Core Runtime (.NET Core Runtime> .NET Core Runtime xyz Далі на сторінці обраної версії .NET Core Runtime перейти до підрозділу Windows і вибрати Server Hosting Installer. Після цього завантажиться потрібний пакет. Цей пакет встановлює .NET Core Runtime, .NET Core Library і модуль ASP.NET Core

Module. Даний модуль, як говорилося вище, якраз і створює проксінг між IIS і сервером Kestrel.

Після установки цього пакета виконаємо в командному рядку команду iisreset або вручну перезапустити IIS, щоб сервер застосував зміни.

Конфігурація сервера

Для конфігурації IIS перейдемо до консолі управління веб-сервером. Для цього перейдемо по шляху Панель управління -> Адміністрування -> Диспетчер служб IIS:

В поле "Фізичний шлях" вказується каталог, в якому опубліковано додаток. А в якості імені вузла визначимо "localhost". Натиснемо на OK, і програма буде запущена.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

1. В межах даної роботи проведено дослідження щодо реалізції веб-сторінки на мові HTML та фреймворці ASP.NET Core MVC.

2. Разроблена та реалізована базова платформа для роботи у середовищі WEB.

3. Був проведений ряд імітаційних експериментів з різними інструментами для створення веб-додатка. Проведено порівняльний аналіз усіх програм та джерел які застосовувались в роботі.

4. Проведено аналіз базових принципів роботи з мовами HTML, CSS та C#. Також наведені приклади роботи з базами даних на основі MySql.

5. Практичне значення роботи складається в тому, що завдяки розробленій платформі кожен зможе створити базовий веб-додаток на фреймворці ASP.NET Core MVC.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ

1. 2020 Developer Survey [Електронний ресурс] // Stack Overflow. – 2020. – Режим доступу до ресурсу: https://insights.stackoverflow.com/survey/2020.

2. Node.js [Електронний ресурс] // OpenJS Foundation – Режим доступу до ресурсу: https://nodejs.org/en/.

3. .NET [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://dotnet.microsoft.com/.

4.Resilient file system (refs) overview. Електронний ресурс https://docs.microsoft.com/en-us/windows-server/storage/refs/refs-overvie

5. Чорноус Г. О. Оптимізація ціноутворення на основі моделей інтелектуального аналізу даних/ Г. О. Чорноус, С. А. Рибальченко // Вісник Київського національного університету. Економіка. – 2015. – Вип. 172. – С. 52-58.

6. Ярмоленко Ю. А. Співвідношення між різновидами ціноутворення в інформаційній економіці / Г. О. Чорноус, Ю. А. Ярмоленко // В кн.: Кузнеца, 2019. – С. 120-135.

7. Pleskach V. Information Security Management System in Distributed Information Systems / V. Pleskach, M. Pleskach, O. Zelikovska // Proceedings of the 2019 IEEE International Conference on Advanced Trends in Information Theory. –Kyiv, 2019. – P. 300-303

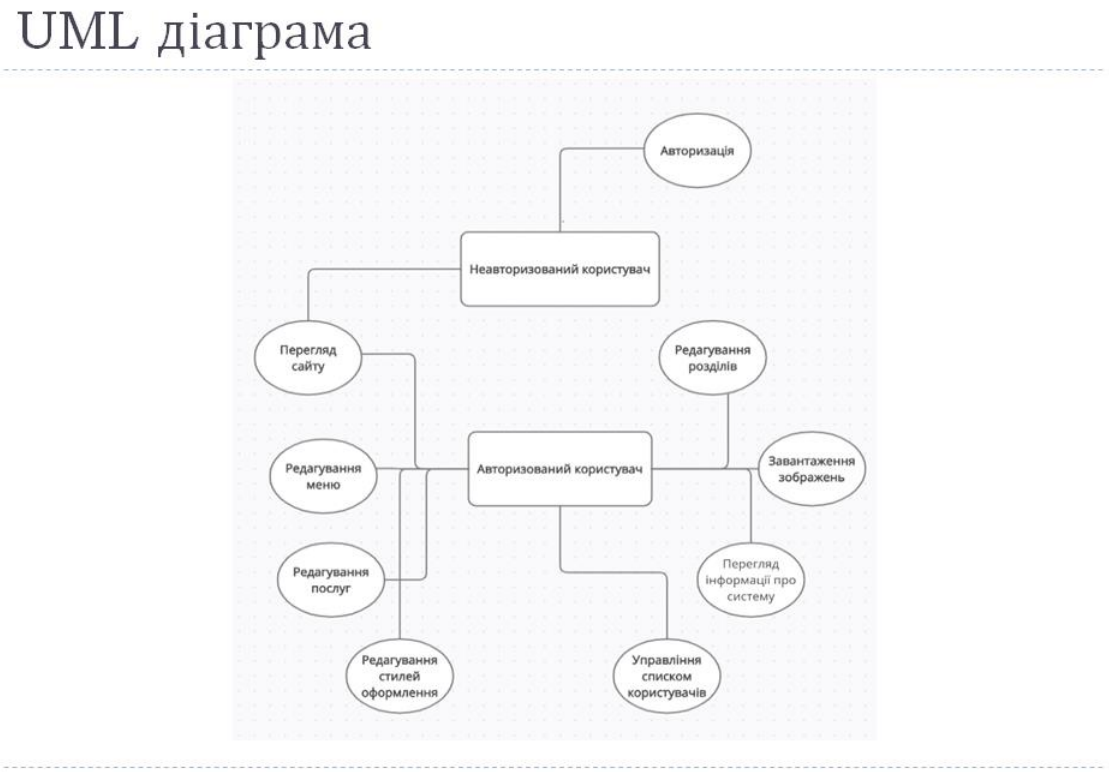
8. TechEmpower Web Framework Benchmarks [Електронний ресурс] – Режим доступу до ресурсу: https://www.techempower.com/benchmarks/.

9. Життєвий цикл додатка MVChttps://metanit.com/sharp/articles/mvc/13.php

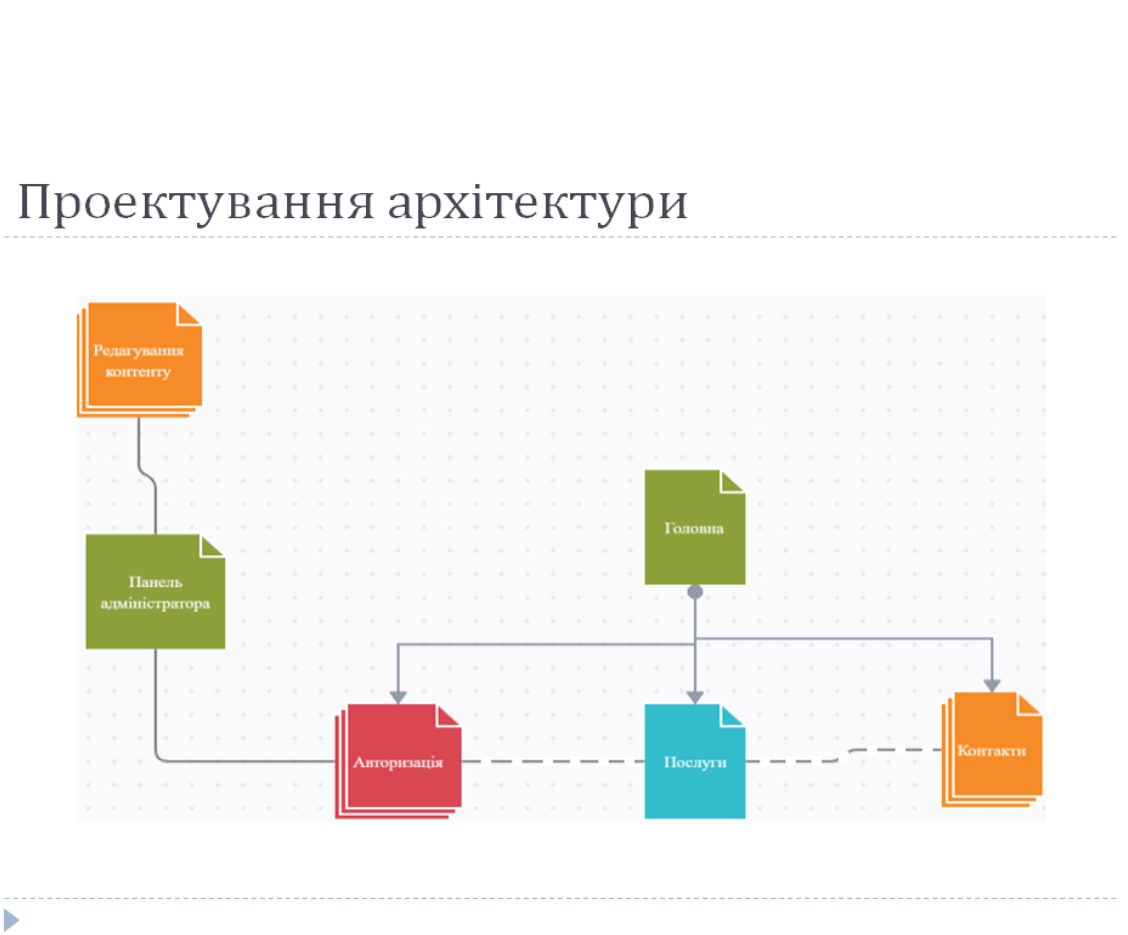
10.Основні візуальні редактори - https://joomla-abc.ru/joomla-3-x/rasshireniya-joomla-3-x/vizual-nye-redaktory-joomla-3.html.

Додаток А

Перелік копій демонстраційного матеріалу



Слайд 5 – UML діаграма



Слайд 6 – Проектування архітектури

Слайд 7 – Створення функціоналу

Додаток Б
Лістинг програмних модулей системи

Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Diploma.Domain;
using Diploma.Domain.Repositories.Abstract;
using Diploma.Domain.Repositories.EntityFramework;
using Diploma.Service;

namespace MyCompany
{
    public class Startup
    {
        public IConfiguration Configuration { get; }
        public Startup(IConfiguration configuration) => Configuration = configuration;

        public void ConfigureServices(IServiceCollection services)
        {

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


            services.AddTransient<ITextFieldsRepository, EFTextFieldsRepository>();
            services.AddTransient<IServiceItemsRepository, EFServiceItemsRepository>();
            services.AddTransient<DataManager>();


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


            services.AddIdentity<IdentityUser, IdentityRole>(opts =>
            {
                opts.User.RequireUniqueEmail = true;
                opts.Password.RequiredLength = 6;
                opts.Password.RequireNonAlphanumeric = false;
                opts.Password.RequireLowercase = false;
                opts.Password.RequireUppercase = false;
                opts.Password.RequireDigit = false;
            }).AddEntityFrameworkStores<AppDbContext>().AddDefaultTokenProviders();


            services.ConfigureApplicationCookie(options =>
            {
                options.Cookie.Name = "myCompanyAuth";
                options.Cookie.HttpOnly = true;
                options.LoginPath = "/account/login";
                options.AccessDeniedPath = "/account/accessdenied";
                options.SlidingExpiration = true;
            });
```

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


            services.AddControllersWithViews(x =>
            {
                x.Conventions.Add(new AdminAreaAuthorization("Admin", "AdminArea"));
            })

                .SetCompatibilityVersion(CompatibilityVersion.Version_3_0).AddSessionState
TempDataProvider();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {



            if (env.IsDevelopment())
                app.UseDeveloperExceptionPage();


            app.UseStaticFiles();


            app.UseRouting();


            app.UseCookiePolicy();
            app.UseAuthentication();
            app.UseAuthorization();


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

AppDbContext.cs

```
using System;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Diploma.Domain.Entities;

namespace Diploma.Domain
{
    public class AppDbContext : IdentityDbContext<IdentityUser>
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
        public DbSet<TextField> TextFields { get; set; }
        public DbSet<ServiceItem> ServiceItems { get; set; }
```

```
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<IdentityRole>().HasData(new IdentityRole
            {
                Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",
                Name = "admin",
                NormalizedName = "ADMIN"
            });

            modelBuilder.Entity<IdentityUser>().HasData(new IdentityUser
            {
                Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",
                UserName = "admin",
                NormalizedUserName = "ADMIN",
                Email = "my@email.com",
                NormalizedEmail = "MY@EMAIL.COM",
                EmailConfirmed = true,
                PasswordHash = new PasswordHasher<IdentityUser>().HashPassword(null,
"admin"),
                SecurityStamp = string.Empty
            });

            modelBuilder.Entity<IdentityUserRole<string>>().HasData(new
IdentityUserRole<string>
            {
                RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab",
                UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),
                CodeWord = "PageIndex",
                Title = "Главная"
            });
            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),
                CodeWord = "PageServices",
                Title = "Наши услуги"
            });
            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),
                CodeWord = "PageContacts",
                Title = "Контакты"
            });
        }
    }
}
```