Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2021 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему:

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ "ЕЛЕКТРОННИЙ ЖУРНАЛ ІЗ ЗАСТОСУВАННЯМ СТЕКУ ТЕХНОЛОГІЙ .NET"

Виконав: студент групи МІТ -41

Якименко А.С. _____
(посада, прізвище ім'я по-батькові) (підпис)

Керівник: доцент кафедри мережевих та інтернет технологій

к.т.н., асистент Махович О.І. _____
(посада, прізвище ім'я по-батькові) (підпис)

Київ 2021

Міністерство освіти і науки України
«Київський Національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко

«_____»_____21 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Якименку Антону Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи:
«Розробка інформаційної системи електронного додатку із застосуванням стеку технологій .NET»

затверджена на засіданні кафедри МІТ «11» грудня 2021 р. протокол № 6
2. Термін здачі закінченої роботи «30» травня 2021р
3. Вихідні дані до проекту (роботи)

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
1. Аналіз завдання розробки інформаційної системи «Електронний журнал»
2. Проектування програмного комплексу з використанням технологій .NET
3. Публікація проекту в мережі

5. Перелік графічного матеріалу 8-10 слайдів
Постановка завдання, Аналіз вимог до цифрової ІС, Визначення підходів та засобів розробки, Недоліки існуючих готових рішень, Запропоновані підходи, Етапи реалізації, Результати реалізації, Висновки

Дата видачі завдання
Керівник роботи Асистент кафедри мережевих та інтернет технологій к.т.н. О.І. Махович
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	20.03.21 – 4.04.21	
2	Розділ 1	5.04.21 – 15.04.21	
3	Розділ 2	16.04.21 – 1.05.21	
4	Розділ 3	2.05.21 – 25.05.21	
5	Доповідь та слайди	21.05.21 – 26.05.21	
6	Пояснювальна записка	5.04.21 – 27.05.21	

Здобувач вищої освіти _____Якименко Антон Сергійович
(підпис) (прізвище, ім'я, по батькові)

Керівник _____Махович Олександр Іванович
(підпис) (прізвище, ім'я, по батькові)

Міністерство освіти і науки України
«Київський Національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ БАКАЛАВРСЬКОЇ РОБОТИ

Направляється студент А.С. Якименко до захисту бакалаврської роботи
(прізвище та ініціали)
за спеціальністю 172 «Телекомунікації та радіотехніка»
(шифр і назва спеціальності)
на тему: «Розробка інформаційної системи електронного додатку із застосуванням стеку технологій .NET».

Бакалаврська робота і рецензія додаються.

Декан факультету _____ В.Є. Снитюк
(підпис)

Довідка про успішність

Якименка А. С. за період навчання на факультеті інформаційних технологій,
(прізвище та ініціали студента)
з 2017 року до 2021 року повністю виконав навчальний план за спеціальністю 172 «Телекомунікації та радіотехніка», з таким розподілом оцінок за:
національною шкалою: відмінно ____%, добре ____%, задовільно ____%;
шкалою ECTS: A ____%; B ____%; C ____%; D ____%; E ____%.

Методист факультету _____ _____
(підпис) (прізвище та ініціали)

Висновок керівника бакалаврської роботи

Студент (ка) _____

Перелік використаних джерел свідчить про вміння студента розбиратись в наукових питаннях та застосовувати їх при дослідженнях. Під час виконання бакалаврської роботи _____ показав відмінну теоретичну та практичну підготовку, знання матеріалу, вміння вирішувати самостійно питання і робити висновки. Роботу виконував уважно, сумлінно, акуратно.

Все це дозволяє оцінити виконану бакалаврську роботу студента _____ на оцінку «_____» та присвоїти йому кваліфікацію _____.

Керівник роботи _____ _____
(підпис)
“_____”_____2021 року

Висновок кафедри про бакалаврську роботу

Бакалаврську роботу розглянуто. Студент А. С. Якименко допускається до захисту
(прізвище та ініціали)
даної роботи в Державній екзаменаційній комісії.

Завідувач кафедри мережевих та інтернет технологій _____ Ю. В. Кравченко
(підпис) (прізвище та ініціали)
“____”_____2021 року

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Розробка інформаційної системи електронного додатку із застосуванням стеку технологій .NET» складається зі вступу, основної частини, що містить 3 розділи, списку літератури та джерел. Загальний обсяг роботи – 41 сторінка. Робота містить 21 рисунок. Список використаних джерел та літератури включає 18 джерел.

Об’єкт дослідження – використання інформаційної системи в межах підприємства задля покращення ефективності.

Мета роботи – продемонструвати потенціальні методи оновлення інформаційної системи підприємств задля покращення ефективності.

Предмет дослідження – розробка інформаційної системи веб-додатоком використовуючи засоби .NET

Метод дослідження – аналіз та порівняння підходів та технологій розробки веб-додатку.

Практичне значення роботи полягає у мінімізувані ресурсних витрат під час розробки та інтеграції оновленого рішення.

Результати здійснених у дипломній роботі досліджень можуть бути використані спеціалістами із розробки інформаційних систем для реалізації програмних додатків для підприємств.

Напрямки подальших досліджень: створення програмного продукту для корпоративних та державних підприємств на основі .NET технологій.

Ключові слова: інформаційна система, .NET, ASP.NET Core, IIS, Visual Studio, Entity Framework, шаблони проектування, Web Deploy.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

WWW – World Wide Web;

IIS – Internet Information Services;

ASP – Active Server Pages;

АС – автоматизована система;

БД – база даних;

ІзОД – інформація з обмеженим доступом;

ІР – інформаційні ресурси;

ІС – інформаційна система;

ІТ – інформаційні технології;

ІТС – інформаційно-телекомунікаційна система;

КС – комп’ютерна система;

НСД – несанкціонований доступ;

ОС – обчислювальна система;

ПБ – політика безпеки;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ПРД – правила розмежування доступу;

ТЗ – технічне завдання;

ТЗІ – технічний захист інформації.

ЗМІСТ

ВСТУП

Актуальність дослідження.

У нинішньому столітті комп'ютерних технологій всі дані і процеси намагаються оцифровувати. Це дозволяє в значній мірі збільшити швидкість і якість роботи будь-якої інфраструктури, де здебільшого буде здійснено вплив на інформаційну систему підприємств.

Гарним прикладом розвинених технологічних країн Європи може послужити Естонія, яка за досить короткий проміжок часу встигла стати одою з найбільш передових цифрових держав. Серед відомих технологічних досягнень Естонії можна відзначити можливість громадянам голосувати через інтернет і підписувати будь-які документи онлайн.

Дані тенденцій ростуть з кожним днем і позитивно позначаються на розвитку економіки і залучення іноземних інвесторів. Серед інших країн Україна так само не залишилася осторонь і робить все можливе, щоб впроваджувати сучасні рішення безпосередньо в саму структуру держави заради підвищення продуктивності робочих процесів і загальної зручності.

Однак серед позитивних моментів цифрової інтеграції варто враховувати основні проблеми, що не дозволяють провести цю процедуру в багатьох країнах. Основні з них: складність самої інтеграції за рахунок часом величезних обсягів даних і складних маніпуляцій над ними; дорожнеча найманої роботи значної кількості кваліфікованих фахівців; і як наслідок - відсутність фінансування з боку підприємства або держави.

Вищеописані проблеми можна вирішити наступним чином: використовувати стандартизований перелік універсальних інструментів розробки і шаблони проектування з наступною побудовою уніфікованих прототипів. Дані рішення комплексно посприяють скороченню часу на розробку структурної частини, варіацій, що розробляються, а також задіяних в них осіб. Відповідно до цього загальна вартість проектів істотно скоротиться.

У даних рішень було визначено такі переваги і недоліки:

Переваги:

• Можливість адаптувати готові рішення під потреби будь-якого підприємства;

• Швидкий процес розробки подальших проектів за рахунок повторного використання розроблених шаблонів;

• Одноразовий вибір загального стека технологій для всіх проектів;

• Спрощена підтримка всього програмного комплексу;

• Скорочення штатних співробітників;

• В основному - безшовна інтеграція (безпосередньо залежить від масштабів і вимог підприємства);

• Дешевизна розробки.

Недоліки:

• Тимчасове збільшення рівня безробіття на загальному тлі масштабів інтеграцій по країні (безпосередньо залежить від самого підприємства);

• Подальше утримування фахівців, що підтримують проект.

В якості цільового прикладу буде розглядатися розробка електронного журналу для державних освітніх установ, у яких відсутня можливість використовувати готові дорогі рішення від сумнівних і неперевірених постачальників.

Для досягнення цієї мети були визначені такі завдання:

• Дослідити існуючі рішення на ринку;

• Проаналізувати техніки моделювання та прототипування;

• Визначити необхідний набір засобів для мінімізації необхідних ресурсів розробки;

• Реалізувати програмний продукт як приклад потенційного рішення.

Результати даної роботи можуть бути корисні для фахівців, які націлені на швидку розробку універсальних рішень з мінімізованими витратами. Також ця робота стане в нагоді для підприємств, які досліджують способи збільшення продуктивності процесів.

1. АНАЛІЗ ЗАВДАННЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ЕЛЕКТРОННИЙ ЖУРНАЛ»

1.1 Аналіз вимог до функціональної частини інформаційної системи

Перед складанням вимог варто врахувати принципи і нюанси існуючої ІС, журналу паперового вигляду, а також її цифрових аналогів.

Це допоможе вивести основні вимоги до продукту, що буде розроблятися, ґрунтуючись на явні недоліки та переваги старої і нової ІС.

Визначення основних недоліків і переваг допоможе виявити спектр проблем. Серед проблем старої ІС можна виділити:

• Відсутність редагування внесених даних і зручного пошуку інформації;

• Скупчення макулатури, яку також необхідно зберігати;

• Витрата часу на фізичну присутність;

• Відсутність запасного зразка в разі втрати або механічних пошкоджень.

Проблеми цифрової ІС:

• Обов'язкова наявність інтернету з боку користувачів;

• Періодично тимчасові відмови в доступі через помилки в системі або технічних робіт.

Ознайомившись з проблемами кожної версії можна зробити висновки, що цифрова ІС має незначні проблеми, але водночас – компенсує всі недоліки ІС старого виду.

ІС може бути реалізована різними шляхами: сайтом, додатком, веб-сервісом або веб-додатком. З огляду на сучасні тенденції, підходи до розробки, вимог і специфіки самої ІС, для цільового прикладу буде реалізовано звичайний журнал освітнього закладу у вигляді веб-додатка.

В даний момент основними вимогами при розробці веб-додатка є наступні пункти:

• Кросбраузерність;

• Адаптивна верстка;

• Інтуїтивний та простий дизайн веб-сторінок;

• Пошукова система;

• Система авторизації.

Кросбраузерність – властивість, що описує підтримку будь-якого браузера, щоб ті в свою чергу однаково відображали всі елементи веб-сторінки. Адаптивна верстка – задає сценарій відображення, забезпечуючи правильне промальовування інтерфейсних елементів на екранах будь-якого пристрою (телевізори, персональні комп'ютери, планшети, смартфони). Дизайн веб-сторінок – відноситься до сфери UX/UI розробок, що відповідають за сценарій використання інтерфейсу, який слід робити практичним, зручним, а також – інтуїтивним. Пошукова система – інструмент комплексного пошуку інформації по структурі веб-сторінки або заданих елементів сайту. Система авторизації – відповідає за обмеження доступу користувачів до певних даних.

У прикладі даної роботи досить буде використовувати всі пункти, за винятком пошукової системи, через свою рудиментарність, і дизайну веб-сторінок, оскільки даний приклад розглядається виключно з точки зору технічної реалізації кінцевого продукту.

Надалі необхідно обрати тематичний напрямок додатку. Було обрано тематику електронного журналу. Оскільки ми живемо в умовах глобальної інформатизації світового простору. Інформатизація проникла в усі сфери людської діяльності і є одним з головних умов досягнення успіху. У зв'язку з цим, роль навчальних закладів, зазнає значних змін, сьогодні на перший план виходить не передача суми знань, накопичених людством, а технологія оперативного пошуку, осмислення, перетворення, зберігання і передачі інформації, а також технологія постановки проблем для дослідження і пошуку їх вирішення. Тому не випадково, головна проблема будь-якого нинішнього освітнього закладу – забезпечити підвищення якості освіти. Підвищення якості освіти можна домогтися за рахунок широкого використання інформаційних ресурсів і комп'ютерних технологій в навчанні.

Єдиний інформаційний простір освітнього закладу – це система, в якій на інформаційному рівні пов'язані всі учасники навчального процесу: адміністрація, викладачі, учні та їх батьки. У зв'язку з цим навчальним закладам посприяють системи електронних журналів.

Матеріально-технічна база загальноосвітніх закладів складається з комп'ютерів, що використовуються як в управлінні освітнім процесом, так і в навчальному процесі. Практично в кожному кабінеті є проектори, принтери, екрани.

Вищенаведені умови дозволяють активно впроваджувати і налагоджувати роботу електронного журналу і щоденника в освітній процес, який автоматизує процес контролю за успішністю, дублює записи журналу освітнього закладу, захищаючи його від спотворень, дає можливість контролювати накопичені оцінки з предметів.

Електронний журнал доступний в будь-якому місці, де є інтернет, за ним не треба «стояти в черзі», як це часто буває в різних установах. Адміністрація має можливість контролювати заповнюваність журналу а також повну картину успішності в будь-яких зрізах: по групі, з предметів, індивідуально по викладачеві

або студенту. Електронні журнали в цьому плані також є гарними помічниками для будь-якого учасника навчального закладу.

Електронний журнал - це обов'язковий стандарт інформатизації освітнього закладу в найближчому майбутньому. Виходячи з вище сказаного, можна прийти до наступних висновків про те, яким повинен бути електронний журнал:

• Електронний журнал схожий на шкільний паперовий журнал і має простий спосіб заповнення;

• Дані студента про оцінки (а також інші відомості, що вносяться до електронного журналу, такі як зауваження, пропуски і т.д.) доступні тільки тим, хто має право їх бачити: вчителі, адміністрація освітнього.

• Студенти мають можливість контролювати середню оцінку з предметів, тим самим працювати над підвищенням успішності. У цьому допоможе зведена відомість студентів з предметів, а також успішність з предметів.

• Електронний журнал здійснює контроль кількості і повноти виставлених оцінок, що забезпечує коректну атестацію студентів. Викладачі і адміністрація освітнього закладу мають інструменти контролю і діагностики успішності класу з кожного предмету, по окремих групах, по кожному викладачу і по кожному студенту;

• Електронний журнал забезпечує можливість контролю за пропусками уроків, за датою і предметом;

• Користувачі електронного журналу мають можливість на сторінці з оцінками класу візуально розрізняти оцінки за контрольні, самостійні та інші види робіт;

Вивчаючи існуючі рішення для наших регіонів - можна виявити багато недоліків різного характеру. Серед основних недоліків можна відзначити наступні пункти:

• Надто висока вартість готових рішень;

• Не інтуїтивність інтерфейсу користувача;

• Брак ресурсів виробничого та фінансового характерів, що будуть підтримувати роботу продукта;

• Неповнота користувальницьких інструкцій або їх повна відсутність;

• Відсутність адекватної технічної підтримки в разі збою системи;

• Відсутність випробувального терміну продукту перед покупкою.

1.2 Визначення підходу і засобів розробки

Раніше веб-сайти були набором статичних веб-сторінок. Однак з часом Інтернет трансформувався з набору інформаційних ресурсів в високорозвинені бізнес-інструмент технології. Принципи та підходи написання сайтів істотно змінилися. Нинішні сайти великих компаній складаються з набору додатків з інтерактивними можливостями, персоналізацією, взаємодією клієнтів з партнерами, а також і засоби інтеграції з корпоративними додатками компаній.

Мета даного розділу полягає в розгляді сучасних методів проектування і способів розробки цільової ІС. З огляду на сучасні тенденції і розвиток інструментів розробки, доцільно буде реалізовувати цільовий приклад як веб-додаток.

Способи розробки веб-додатків можуть бути розділені на три основні категорії:

• Використання чистого програмування;

• Реалізація через шаблонні веб-сторінки;

• Використання об'єктних середовищ або ж фреймворків.

Незважаючи на можливі перетини цих категорій, більшість відомих підходів пов'язані тільки з однією з них.

Програмний підхід веб-додатку – зовнішня програма, складена на деякій універсальній мові програмування високого рівня. Основною проблемою програмного підходу при розробці веб-додатків – орієнтація на написання коду. Розмітка мови HTML і інші конструкції форматування інтегруються в логіку роботи програми за допомогою операторів виведення.

Такий підхід явно обмежує можливості веб-дизайнерів вносити зміни в оформлення створюваної додатком сторінки. Веб-дизайнер може розробляти макет сторінки, після чого програміст повинен перетворювати це в код і пов'язати з основною програмою. Для зміни будь-якого елементу сторінки – необхідне втручання програміста.

Зовнішні програми. Найпростіший спосіб динамічно формувати веб-сторінки у відповідь на HTTP запит – обробка формування HTML сторінки зовнішньою програмою, яка буде брати до уваги передані в HTTP запиті параметри і формувати підсумкову сторінку.

Технологія Common Gateway Interface незалежною від типу веб-сервера, програмна технологія створення і виконання веб-додатків, визначала набір правил, яким повинна слідувати програма, щоб вона могла виконуватися на різних HTTP серверах і операційних системах.

Недоліки технології CGI, які роблять її не практичною в більшості випадків:

• Низька продуктивність. Для кожного HTTP запиту до CGI програмі веб-сервер запускає новий процес, який закінчує роботу тільки після завершення програми.

• Для складання і налагодження CGI програм розробник повинен володіти чималим досвідом програмування на одній з мов, на яких можна програмувати CGI програми.

• В CGI програмах програмний код і код розмітки повністю перемішані. Дизайнер повинен знати програмування, щоб міняти структуру веб-сторінок.

Прикладом технології розширення архітектури веб-сервера є прикладний інтерфейс Java Servlet API, який пов'язує веб-сервер з віртуальною машиною Java Virtual Machine (JVM). Віртуальна машина JVM підтримує виконання спеціальної Java програми (контейнер «сервлетів»), яка відповідає за управління даними сеансу роботи і виконання Java-сервлетів.

Сервлети це спеціальні класи на мові Java (програми), які мають доступ до інформації з HTTP запитів. Вони формують HTTP відповіді, які повертаються браузерам. На відміну від ISAPI розширень, технологія Servlet API переноситься між різними веб-серверами, операційними системами та комп'ютерними платформами. Сервлети виконуються однаково в будь-якому середовищі, яка надає сумісний з ними контейнер сервлетів. Технологія Servlet API використовується великою кількістю розробників і підтримується багатьма відомими веб-серверами.

Серед недоліків можна визначити лише складність вивчення такої технології.

Підходи, засновані на шаблонах, використовують в якості адресованих об'єктів «шаблонів», замість програми або скрипта. За своєю сутністю, шаблони є HTML файлами з додатковими «тегами», які задають методи включення динамічно сформованого контенту. Таким чином, файл шаблону містить HTML код, який описує загальну структуру сторінки, і додаткові серверні теги, розміщені таким чином, щоб формувати з їх допомогою зміст сторінки.

В даний час до найбільш поширених технологій розробки веб-додатків на основі шаблонів, відносяться наступні: Server-Side Includes (SSI), Cold Fusion, PHP, Active Server Pages (ASP) і Java Server Pages (JSP).

Іншою досить популярною технологією, заснованою на шаблонах, є технологія Cold Fusion, розроблена компанією Adobe. Перевага даного підходу полягає в тому, що даний шаблон може створюватися і підтримуватися дизайнером сторінки, який має базові знання мови HTML і веб-графіки, але не має досвіду програмування.

Технологія «PHP Hypertext Preprocessor» або просто PHP дозволяє розробникам вбудовувати програмний код в шаблони, за допомогою мови, схожого з мовою скриптів Perl.

Компанія Microsoft розробила технологію ASP (Active Server Pages), яка об'єднала можливості створення шаблонів, що включають скрипти, з доступом до набору OLE і COM об'єктів, що є в операційній системі Windows, в тому числі і до ODBC джерел даних. Дана технологія, об'єднана з безкоштовним веб-сервером Internet Information Server (IIS), швидко стала популярною серед програмістів, що використовують Visual Basic, які оцінили можливість використання в шаблонах мови VBScript. Як і PHP шаблони, ASP сторінки можуть включати блоки скриптів (використовують посилання на COM об'єкти), упереміш з HTML форматуванням.

На відміну від таких об'єктно-орієнтованих мов, як Java або C ++, мова, яка використовується в ASP сторінках, був лінійним і строго процедурним. На відміну від технології PHP, ASP не пов'язаний з однією конкретною скриптовою мовою. У ASP в якості стандартного мови використовується мову Visual Basic Scripting Edition (VBScript), але може використовуватися і мова JavaScript.

Звичайні скриптові технології на стороні сервера використовують різні об'єкти, але не дозволяють розробляти і використовувати власні класи і створювати на їх основі об'єкти. У зв'язку з цим подальший розвиток веб-технологій було пов'язано зі створенням спеціальних об'єктно-орієнтованих технологій розробки веб-додатків. Використання даних технологій дозволяє зробити розробку веб-додатків більш схожою на розробку звичайного об'єктно-орієнтованого програмного забезпечення.

Об'єктні середовища представляють собою наступний рівень вдосконалення розробки веб-додатків. Замість об'єднання розмітки і логіки в єдиний модуль, об'єктні середовища підтримують принцип відділення змісту від представлення.

В даний час є два підходи до створення об'єктно-орієнтованих веб-додатків:

• Підходи, засновані на наборі спеціальних веб-сторінок (веб-форм), пов'язаних з описами класів, об'єкти яких будуть створюватися, і використовуватися при їх виклику (наприклад, технологія ASP.Net Web Forms; технологія JavaServer Faces);

• Підходи, засновані на використанні наборів класів, відповідних шаблоном Model-View-Controller (MVC), наприклад, технології на основі мови Java – Tapestry, Struts, Spring і технологія компанії Microsoft – ASP.Net MVC.

З огляду на вищеописані підходи - був обраний підхід на основі об'єктних середовищ із застосуванням технологій ASP .Net MVC в складі технологій .Net Framework.

1.3 Аргументація переваг використання технологій ASP.NET

Головна перевага технології ASP.NET полягає в тому, що завдяки її використанню можна виконувати сценарні завдання на сервері. Завдяки сценарним завданням можна отримати швидкий доступ до баз даних, файлів і багатьом іншим ресурсам, що зберігаються на сервері. Також технологія ASP.NET дозволяє отримати доступ до електронної пошти. Ще однією перевагою розробки сайту на ASP.NET є можливість його створення за допомогою майстра сайту, який дозволяє робити розмітку тільки на головній сторінці, а решта просто прив'язувати до неї і заповнювати потрібний простір тим чи іншим контентом. Майстер сайту дозволяє зробити не тільки розмітку, а й повністю розробити дизайн сайту, його фон, елементи меню, шапку сайту і т.д. Можна виділити і ряд інших переваг технології ASP.NET перед іншими аналогічними технологіями:

• ASP.NET має перевагу в швидкості в порівнянні з іншими технологіями, заснованими на скриптах;

• Код компілюється, тому більшість помилок відловлюються ще на стадії розробки;

• Призначені для користувача елементи управління дозволяють виділяти часто використовувані шаблони;

• Можливість кешування всієї сторінки або її частини для збільшення продуктивності;

• Можливість кешування даних, що використовуються на сторінці;

• Наявність master-page для завдання шаблонів оформлення сторінок;

• Простота розгортання; розгортання ASP.NET-додатків виконується шляхом копіювання файлів програми в спеціальну папку на веб-сервері; перезапуск веб-сервера при цьому не потрібен;

• інтеграція з .NET Framework. ASP.NET є частиною платформи .NET Framework. Розробники можуть використовувати можливості, що надаються цією платформою, при створенні додатків.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ .NET

2.1 Проектування Моделей і Контексту даних

У попередніх розділах були визначені підхід, засоби розробки, шаблон прототипирования, а також спроектовані моделі даних і логіка самого додатка. Ґрунтуючись на перерахованих вище пунктах, можна приступати до реалізації.

При створенні проекту, фреймворк за нас конфігурує файлову структуру програми та згенерує базові файли: від деякий сторінок з розміткою, до файлів налаштувань проекту.

Весь функціонал буде забезпечуватися наступною структурою проекту (рис. 2.2.1.).

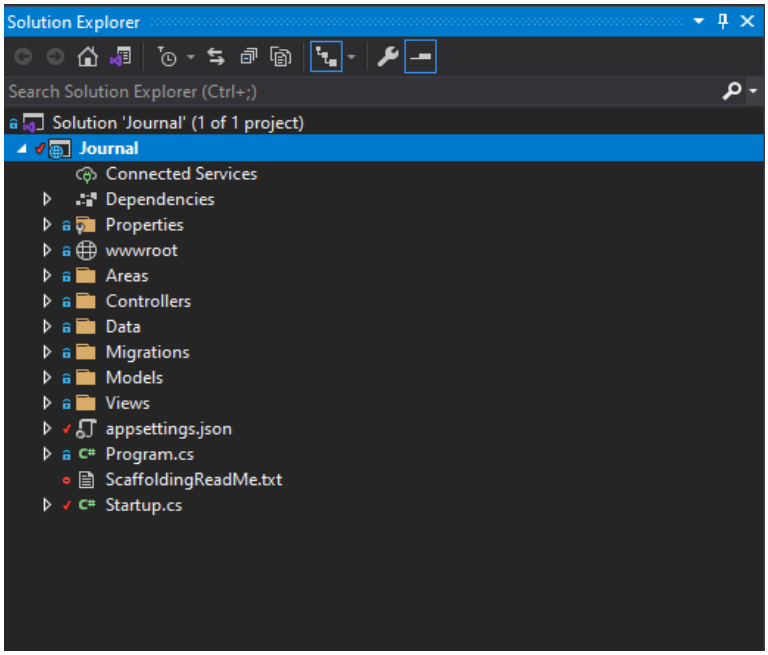

Рисунок 2.2.1. – Структура проекта в оглядачеві рішень.

Розглянемо, для чого потрібні всі ці папки і файли.

• wwwroot - містить статичний контент, такий як CSS-файли для зберігання стилів, зображення, JavaScript-сценарії і файл favicon.ico;

• Dependencies - містить графічне представлення NuGet, призначене для зручного управління пакетами;

• Controllers: містить файли класів контролерів. За замовчуванням в цю папку додається контролер HomeController;

• Models: містить файли моделей. За замовчуванням Visual Studio додає модель даних для відображення інформації про можливі помилки;

• Views: тут зберігаються відображення у вигляді CSHTML-файлів,

об'єднують код на мовах HTML і C #, і реалізують динамічне генерування

HTML-відповіді. Все відображення групуються по папках, кожна з яких відповідає одному контролеру. Після обробки запиту контролер відправляє одне з цих відображень клієнту. Також тут є каталог Shared, що містить загальні для всіх відображення;

• в файлі appsettings.json доступні настройки, ваш веб-додаток може

завантажити під час виконання, наприклад, рядок підключення бази даних для системи ASP.NET Identity.

• файл Program.cs є консольний додаток, що містить точку входу Main і виконує початкову конфігурацію, компіляцію і запуск веб-додатки. Він може викликати метод UseStartup <T> (), щоб вказати клас, здатний виконати додаткову конфігурацію.

• Startup.cs: містить конфігураційні дані для додаткової настройки сервісів, наприклад, ASP.NET Identity для аутентифікації SQLite для зберігання даних і т.п., а також для зв'язку між компонентами програми. Конкретна структура кожного окремого додатка, природно, буде відрізнятися, а гнучкість MVC дозволяє змінювати структуру, пристосовуючи її до своїх потреб. Але описані вище моменти будуть загальними для більшості проектів.

Програма, створення якої описується, буде емулювати роботу навчального журналу, де такі користувачі як, студент і викладач, що зайшли на сайт, зможуть моніторити записи журналу або створювати нові.

Перш за все визначимо моделі даних нашого застосування. Оскільки мова йде про навчальний журналі, то такими Моделями можуть бути: спеціальність студента, сам студент, його група і викладач, заняття, які він відвідує; предмети, які вивчає; ну і відповідно модель журналу, в якому буде вестися успішність студента.

У проекті вже за замовчуванням вказано стандартну папку Models. У ній будуть перебувати наші моделі. Натиснемо на цю папку правою кнопкою миші і в меню виберемо Add-> Class і створимо клас Student. Додамо в нього поля, що описують модель Студента (рис. 2.2.2.).

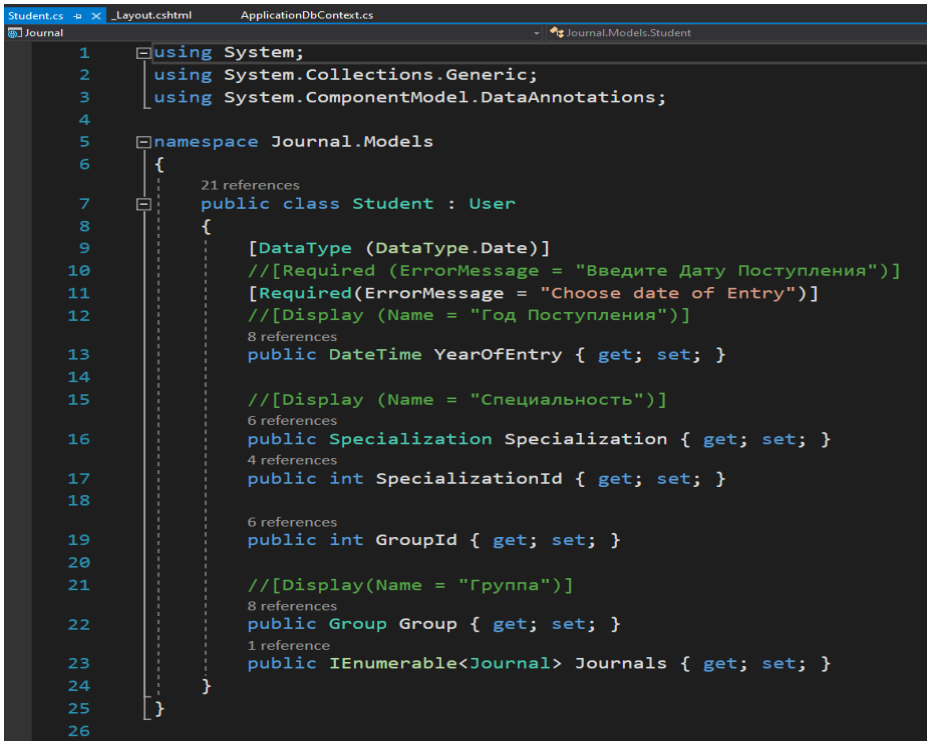

Рисунок 2.2.2. – Створення моделі студента з необхідними полями

Після того як повністю проведемо процедуру створення всіх необхідних Моделей - сформуємо контекст даних.

Контекст даних використовує Entity Framework для доступу до БД, ґрунтуючись на Моделях даного проекту. Додаємо в клас ApplicationDbContext набори даних для вищеописаних Моделей (рис. 2.2.3).

```
ApplicationDbContext.cs
Journal                                                    Journal.Data.ApplicationDbContext
using Journal.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Text;

namespace Journal.Data
{
    29 references
    public class ApplicationDbContext: DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options): base(options)
        {

        }
        7 references
        public DbSet<Group> Groups { get; set; }
        6 references
        public DbSet<Student> Students { get; set; }
        8 references
        public DbSet<Teacher> Teachers { get; set; }
        3 references
        public DbSet<Journal.Models.Journal> Journals { get; set; }
        4 references
        public DbSet<Class> Classes { get; set; }
        10 references
        public DbSet<Subject> Subjects { get; set; }
        6 references
        public DbSet<Specialization> Specializations { get; set; }
        7 references
        public DbSet<JournalSet> JournalSets { get; set; }

        1 reference
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);

            builder.Entity<Student>()
                .HasOne(s => s.Group)
                .WithMany(g => g.Students)
                .HasForeignKey(s => s.GroupId);

            builder.Entity<Student>()
                .HasMany(s => s.Journals)
                .WithOne(j => j.Student)
                .HasForeignKey(j => j.StudentId);

            builder.Entity<Subject>()
                .HasMany(s => s.Classes)
                .WithOne(c => c.Subject)
                .HasForeignKey(c => c.SubjectId);

            builder.Entity<Specialization>()
                .HasMany(spec => spec.Students)
                .WithOne(st => st.Specialization)
                .HasForeignKey(st => st.SpecializationId);

            builder.Entity<Teacher>()
                .HasMany(t => t.Groups)
                .WithOne(g => g.Teacher)
                .HasForeignKey(g => g.TeacherId);

            builder.Entity<JournalSet>()
                .HasMany(js => js.Journals)
                .WithOne(j => j.JournalSet)
                .HasForeignKey(j => j.JournalSetId);
        }
    }
}
```

Рисунок 2.2.3. – Контекст даних з властивостями відповідних до Моделей

Властивості public DbSet <Group> Groups {get; set;} допомагають отримувати з БД набір даних певного типу (наприклад, набір об'єктів Групи). Хоча ми будемо використовувати базу даних, але створювати явним чином ми її не будемо. За нас все зробить Entity Framework. Це так званий підхід Code First - у нас є модель, а грунтуючись на ній, фреймворк буде створювати таблиці в базі даних.

Як можна помітити, Модель являє собою звичайний клас на мові C #. Всі моделі в цьому випадку мають набір властивостей, що описують реальні властивості об'єкта. У той же час при створенні моделей слід дотримуватися деяких умовностей. Оскільки ми будемо використовувати для зберігання моделей базу даних Microsoft SQL Server, то для маніпуляції над об'єктами в базі даних нам потрібно визначити для них первинний ключ (Primary Key), який виконує роль універсального ідентифікатора об'єкта. Тому першою властивістю в кожної моделі йде властивість Id, призначеної для зберігання первинного ключа.

Безпосередньо, після додавання всіх властивостей, можна перейти до оформлення відносин між моделями. Для встановлення відносин між моделями в Entity Framework Core застосовують зовнішні ключі і навігаційні властивості класу моделі. Наступним кроком буде опис відносин: «один-до-багатьох», «багато-до-багатьох», «багато-до-одного». Відносини між сутностями можна описати за допомогою Entity Framework Fluent API із застосуванням методів HasOne, HasMany, WithOne, WithMany і т.д. відповідно. Методи HasOne, HasMany встановлюють навігаційну властивість для сутності, яка налаштовується. Після них можуть йти виклики методів WithOne, WithMany, які визначають навігаційну властивість пов'язаної сутності. Зовнішній ключ встановлюється за допомогою методу HasForeignKey.

2.2 Проектування Контролерів

Центральне місце ASP.NET Core MVC займають Контролери. При отриманні HTTP запиту система маршрутизації обирає відповідний контролер і передає йому вхідні параметри запиту. Контролер обробляє ці параметри і передає результат в зворотному напрямку.

У ASP.NET Core MVC Контролери є звичайні класи на мові C #, що успадковуються від абстрактного базового класу Microsoft.AspNetCore.Mvc.Controller. Якщо потрібно додати ще один Контролер, то

це можна зробити через контекстне меню папки Controllers, використовуючи готовий шаблон, причому в назві класу обов'язково повинен бути присутнім суфікс Controller.

Приклад оформлення одного з файлів Контролера (рис. 2.3.1).

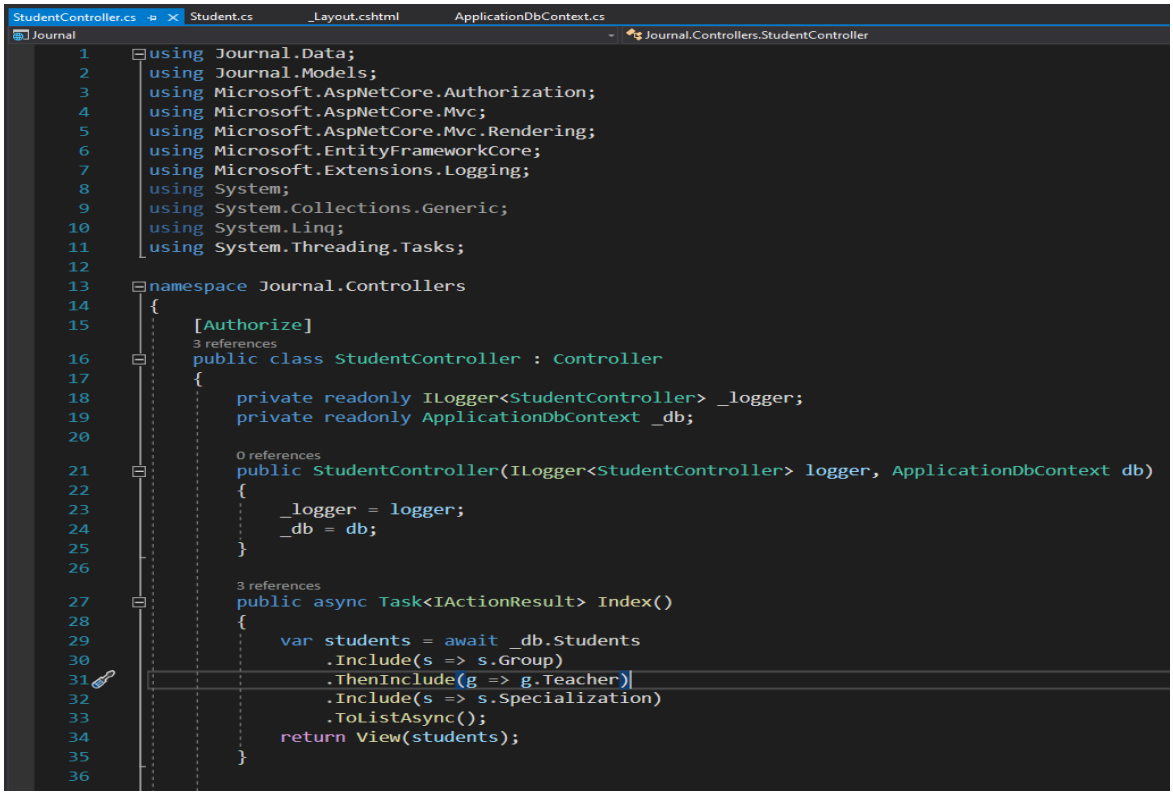

Рисунок 2.3.1. – Реалізація Контролера студента.

За допомогою навігаційних властивостей можна завантажувати пов'язані дані.

Для цього потрібно застосувати метод Include(), який дозволяє включити в набір даних також інформацію по зв'язаних таблицях.

Клас Контролера може мати різні члени класу: поля, властивості, методи, конструктори та ін. Методи, які мають модифікатор доступності public, можуть обробляти запити і розглядаються як дії (Actions). Наприклад, відкритий (public) клас StudentController є контролером, так як його ідентифікатор містить суфікс Controller і він, крім того, успадкований від класу Controller. Відкритий метод public IActionResult Index () є дією контролера StudentController і може обробляти запити типу GET (за замовчуванням). Методи в рамках однієї дії можуть

обслуговувати різні запити. Для того, щоб вказати тип запиту HTTP, потрібно застосувати до методу один з атрибутів [HttpGet], [HttpPost], [HttpPut], [HttpDelete] і [HttpHead]. Якщо атрибут не вказаний явно, то вважається, що застосований атрибут HttpGet. Наприклад, два методу Create, які призначені для обслуговування одного дії Create, обробляють запити двох типів: HttpGet і HttpPost (рис. 2.3.2.).

```
1 reference
public async Task<IActionResult> Create()
{

    ViewBag.SpecializationId = new SelectList(_db.Specializations, "Id", nameof(Specialization.Name));
    ViewBag.GroupId = new SelectList(_db.Groups, "Id", nameof(Group.Name));

    return View();
}

[HttpPost]
1 reference
public async Task<IActionResult> Create(Student student)
{
    if (!ModelState.IsValid)
    {
        return RedirectToAction(nameof(Create));
    }
    await _db.Students.AddAsync(student);
    await _db.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
1 reference
public async Task<IActionResult> Edit(int Id)
{
    var student = await _db.Students.FirstOrDefaultAsync(s => s.Id == Id);
    ViewBag.StudentId = new SelectList(_db.Students, "Id", nameof(Student.FirstName));
    ViewBag.SpecializationId = new SelectList(_db.Specializations, "Id", nameof(Specialization.Name));
    ViewBag.GroupId = new SelectList(_db.Groups, "Id", nameof(Group.Name));
    return View(student);
}
```

Рисунок 2.3.2. – Оформлення HttpPost запиту у методі Create

Використовуючи дані атрибути, для кожного Контролера були реалізовані методи відповідні підходу CRUD: Index, Create, Edit і Delete відповідно.

Підсумкова тека Контролерів буде виглядати наступним чином (рис. 2.3.3.).

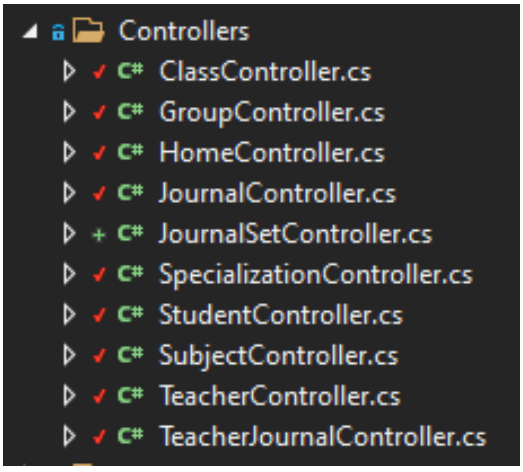

Рисунок 2.3.3 – Тека Контролерів

2.3 Проектування Відображень

За роботу з Відображеннями відповідає компонент ViewResult. Він виконує рендеринг відображень в вебсайт і повертає її в якості результату дії. Щоб повернути об'єкт ViewResult, потрібно викликати метод return View ().

Синтаксис Razor.

Синтаксис Razor дозволяє здійснити перехід від розмітки html до коду на C #. Конструкції, перед якими присутній символ @, вважаються кодом на мові C #. Всі конструкції Razor можна умовно розділити на два види:

1) конструкції одного рядка (рис.2.4.1.).

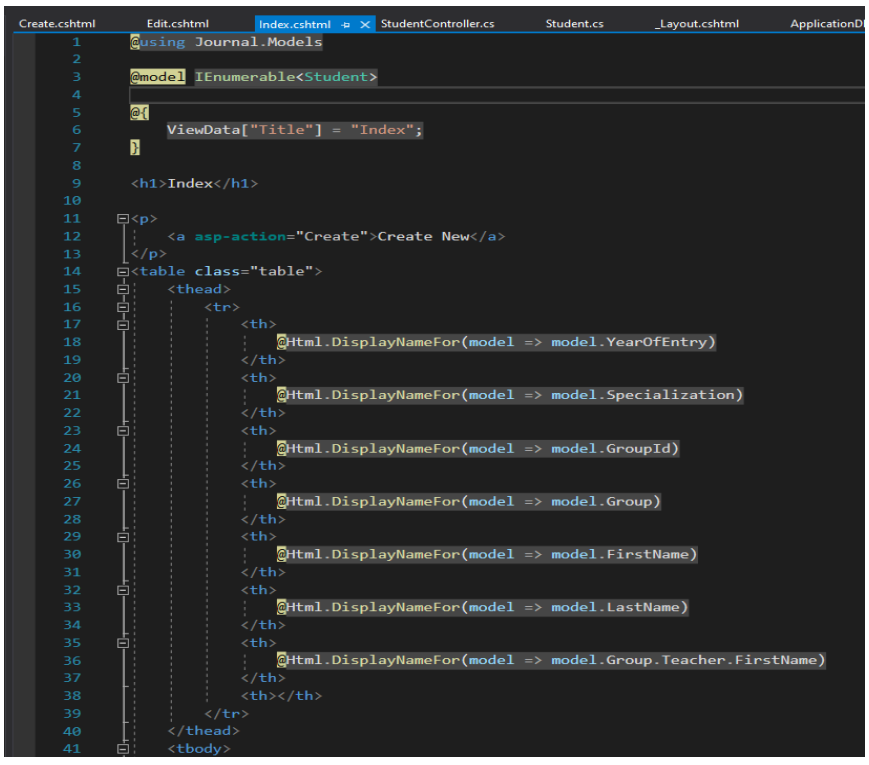

Рисунок 2.4.1 – Razor конструкція одного рядка (ViewData[“Title”] = “Index”;)

2) блоки коду (рис.2.4.2.).

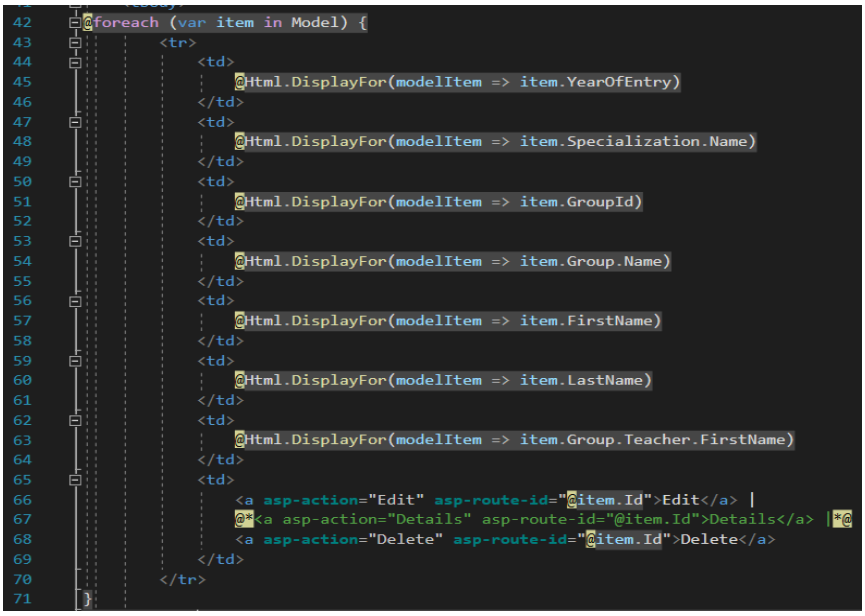

Рисунок 2.4.2 – Razor конструкція блоку (foreach (var item in Model) {…})

Також, якщо в конструкціях Razor потрібно вивести текст, то потрібно застосовувати конструкцію @: Some data, використовувати спеціальний тег <text> "Some else data." </ Text> або довільний html тег.

Щодо передачі даних, то існують різні способи реалізації передачі в поданні: ViewData, ViewBag і за допомогою Моделі Відображення. ViewData - це словник, що містить пари ключ - значення. Задати значення можна в контролер, а зчитати значення можна у Відображені. ViewBag дозволяє динамічно задавати через точкову нотацію різні властивості і з їх допомогою передавати дані, а також зчитувати дані в відображенні. Модель Уявлення - спосіб, яким слід віддавати перевагу для передачі основних результатів роботи контролера. Для передачі даних використовується метод View (Object).

Приклад виклику View (рис. 2.4.3).

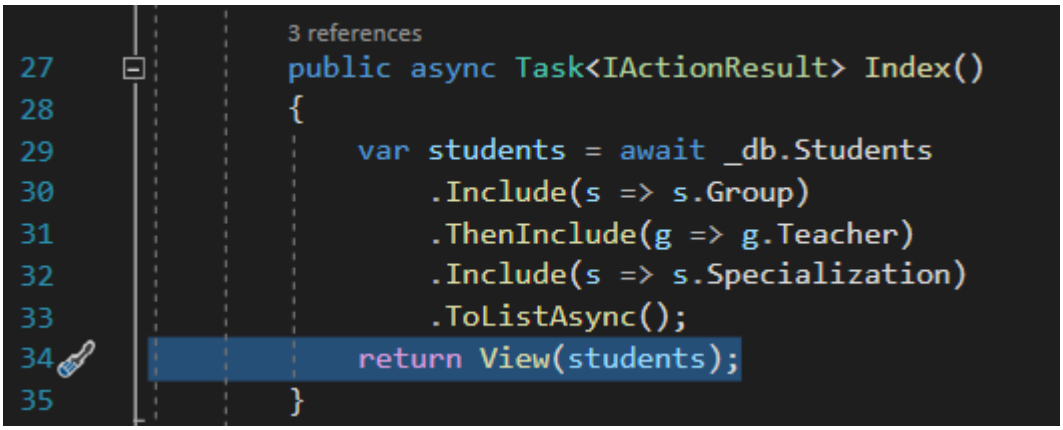

Рисунок 2.4.2 – Виклик View

В цьому випадку ми передаємо дані з таблиці Students через звернення до _db.Students.

Щоб мати можливість скористатися даними в Поданні, потрібно в ньому спочатку оголосити Модель Уявлення. Тип Моделі повинен відповідати типу об'єкта, який передається.

3. ПУБЛІКАЦІЯ ПРОЕКТУ В МЕРЕЖІ

3.1 Впровадження системи Авторизації

Розглянутий проект з точки зору функціоналу практично готовий, однак в нашому випадку перед публікацією проекту в мережу - необхідно впровадити систему авторизації для нових користувачів. Для багатьох проектів на інших фреймворків систему авторизації часто доводиться писати з нуля. Щастя серед вбудованих інструментів комплексу .NET присутній готова система, яку досить просто впровадити в проект.

Щоб це зробити - перейдемо в контекстне меню проекту, виберемо розділ Add і виберемо New Scaffolded item ... (рис. 3.1.1.).

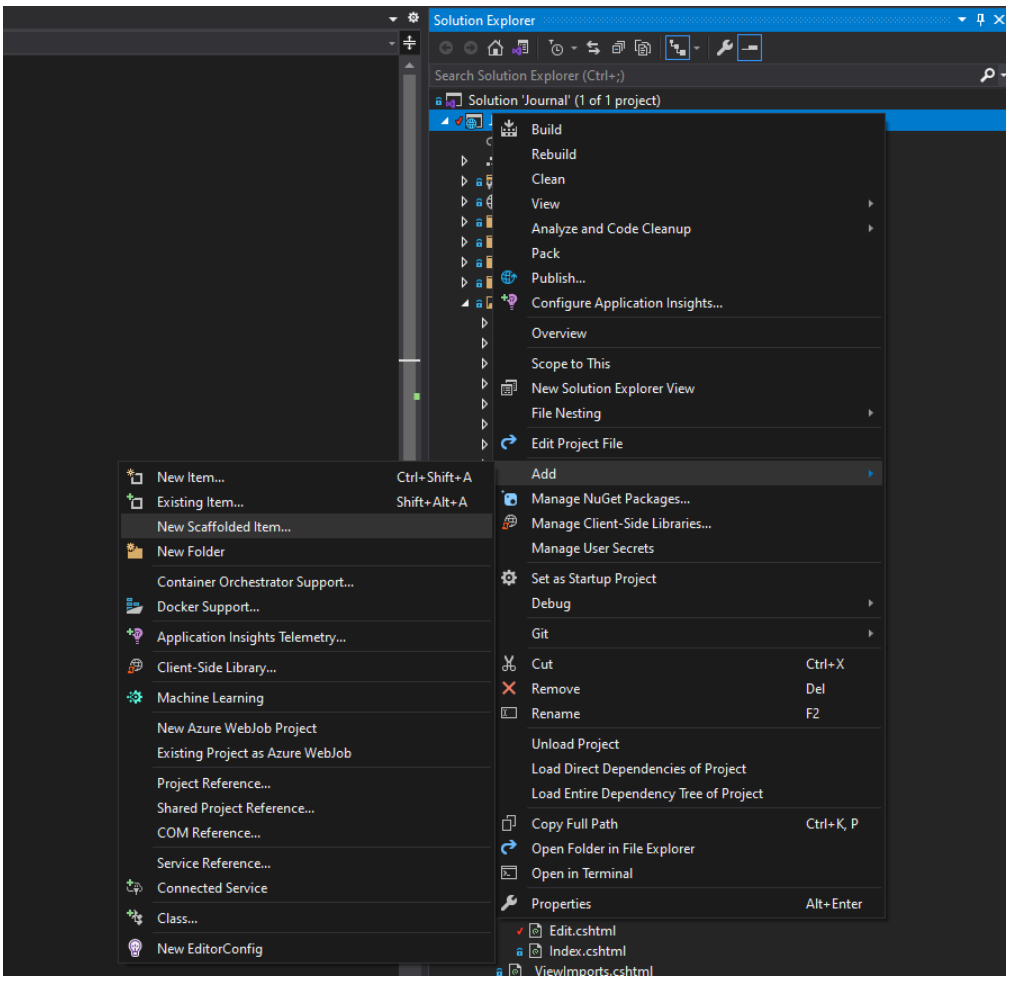

Рисунок 3.1.1 – Контекстне меню проекта

Таким чином перейдемо в меню і вже звідти виберемо розділ Identity, після в поле шляху вказуємо розташування файлу _Layout.cshtml, який розташовується в директорії Views / Shared. Також вибираємо опцію Override all files і дамо назву нового класу контексту, який вже буде ставитися до системи Авторизації.

В результаті отримаємо автоматично згенерувала директорію Areas в папці проекту (рис. 3.1.2.).

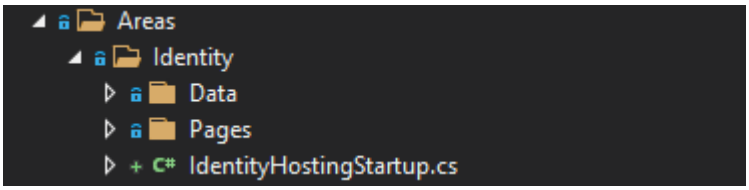

Рисунок 3.1.2 – Тека Areas

Наступним кроком підключимо систему авторизації в файлі Startup.cs безпосередньо після виклику методу UseRouting(), який як правило розташовується перед методом UseEndpoints(). Також підключимо маршрутизацію на базові сторінки RazorPages шляхом додавання виклику методу endpoints.MapRazorPages() (рис. 3.1.3.).

```
70
71          app.UseRouting();
72
73          app.UseAuthentication();
74          app.UseAuthorization();
75
76          app.UseEndpoints(endpoints =>
77          {
78              endpoints.MapControllerRoute(
79                  name: "default",
80                  pattern: "{controller=Home}/{action=Index}/{id?}");
81              endpoints.MapRazorPages();
82          });
83      }
```

Рисунок 3.1.2 – Підключення методів авторизації у Startup.cs

Після виконаних кроків достатньо додати кожному Контролеру атрибут авторизації щоб з ним можна було взаємодіяти тільки після того як користувач увійде в систему (рис. 3.1.4.).

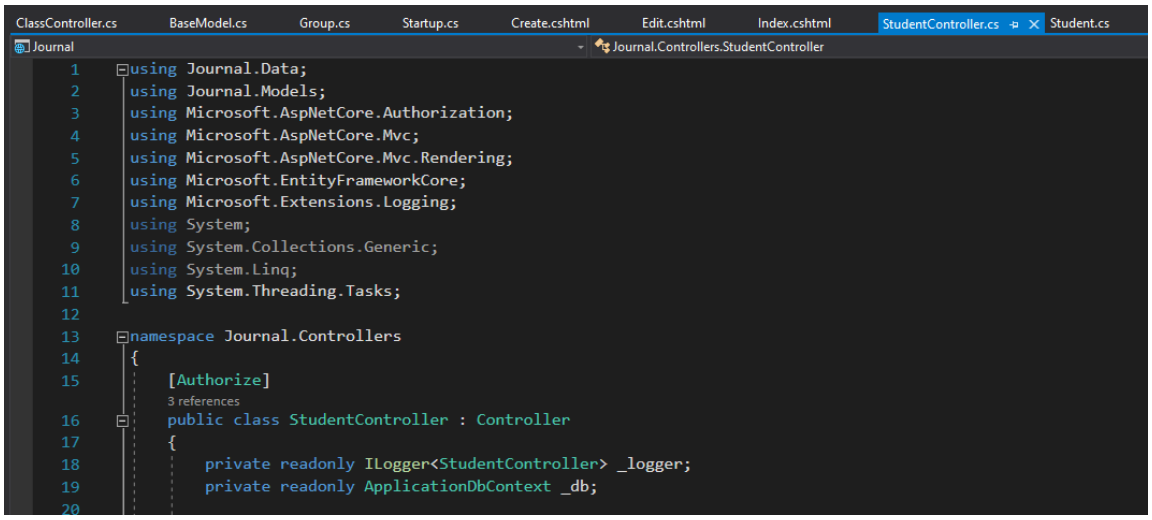

Рисунок 3.1.4 – Додавання атрибутів [Authorize]

3.2 Розгортання проекту

Публікація додатка означає створення скомпільованої додатки, яке можна розмістити на сервері. Розгортання додатки означає переміщення опублікованого додатка в систему розміщення. Публікація забезпечується пакетом SDK для .NET Core, а розгортання може проводитися різними способами. Ми ж розгорнемо проект на пристрої при за допомогою методу Web Deploy.

Попередні вимоги:

• Пакет SDK для .NET Core, встановлений на комп'ютері розробки.

• Сервер Windows Server з налагодженою роллю Веб-сервер (IIS). Якщо сервер не налаштований для розміщення веб-сайтів зі службами IIS, дотримуйтесь вказівок в розділі Налаштування служб IIS статті Розміщення ASP.NET Core в Windows зі службами IIS, а потім поверніться до цієї інструкції.

Виберемо в оглядачеві рішень проект та перейдемо в контекстне меню, там оберемо кнопку Publish. Наступним кроком – перейдемо в варіант публікації WebServer (IIS) (рис. 3.2.1.).

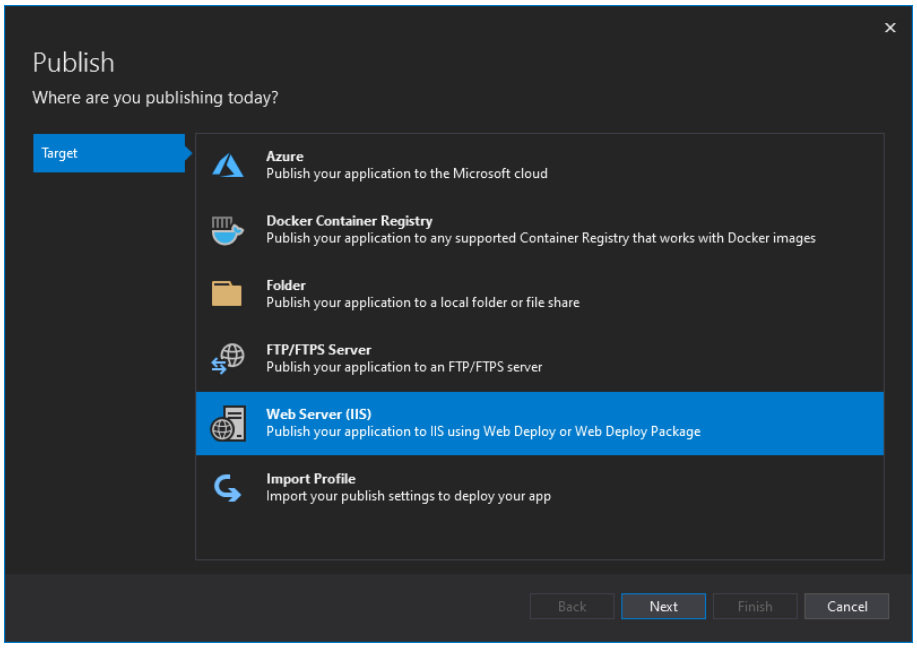

Рисунок 3.2.1 – Перехід до налаштувань публікації проекту у Web Deploy

Надалі пропишемо необхідні поля, які потребують метод публікації, а саме: розташування сервера, його ім'я і URL місце призначення. Також перевіримо підключення за допомогою кнопки Validate Connection і бачимо, що нам вдалося підключитися (рис. 3.2.2.).

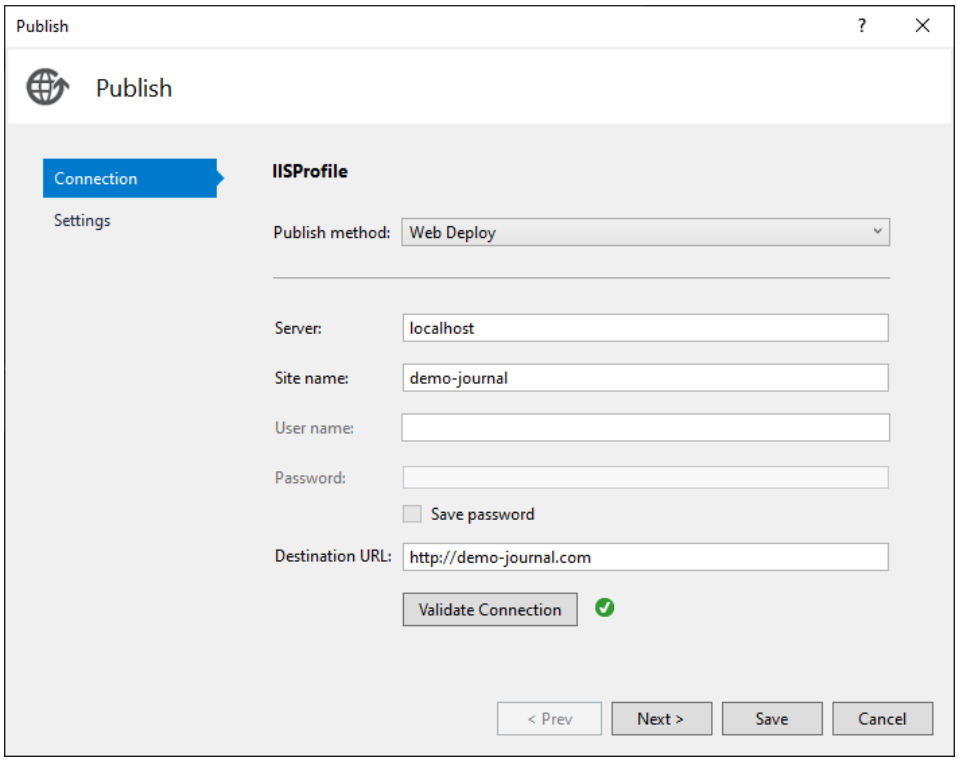

Рисунок 3.2.2 – Внесення данних серверу для публікації проекту у Web Deploy

Після всіх налаштувань – натиснемо на кнопку Publish і очікуємо поки проект збереться і запуститься. Як можна помітити, публікація пройшла успішно і в браузері моментально відкриється сайт (рис. 3.2.2.-3.2.3).

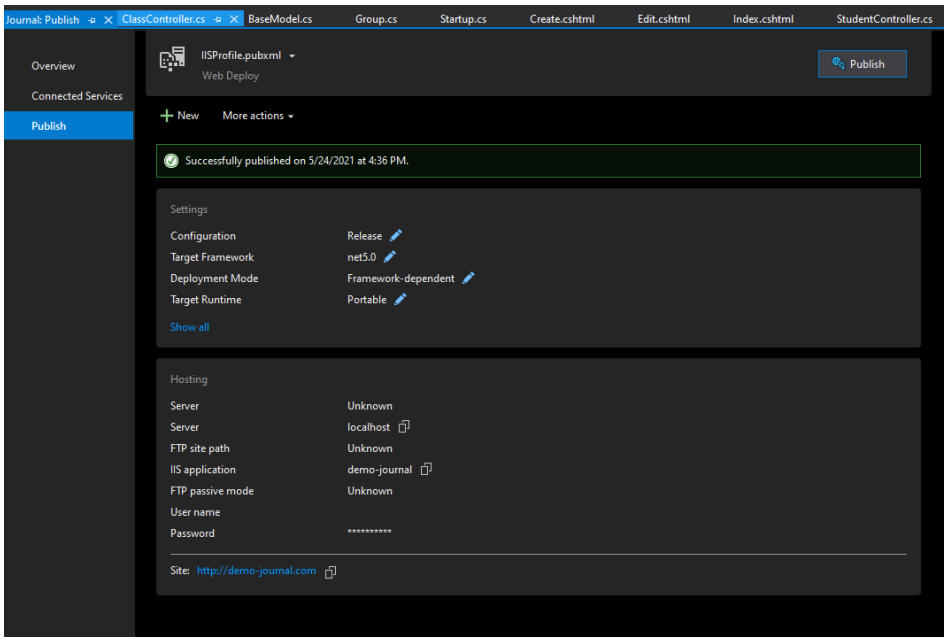

Рисунок 3.2.1 – Сторінка звіту успішної публікації

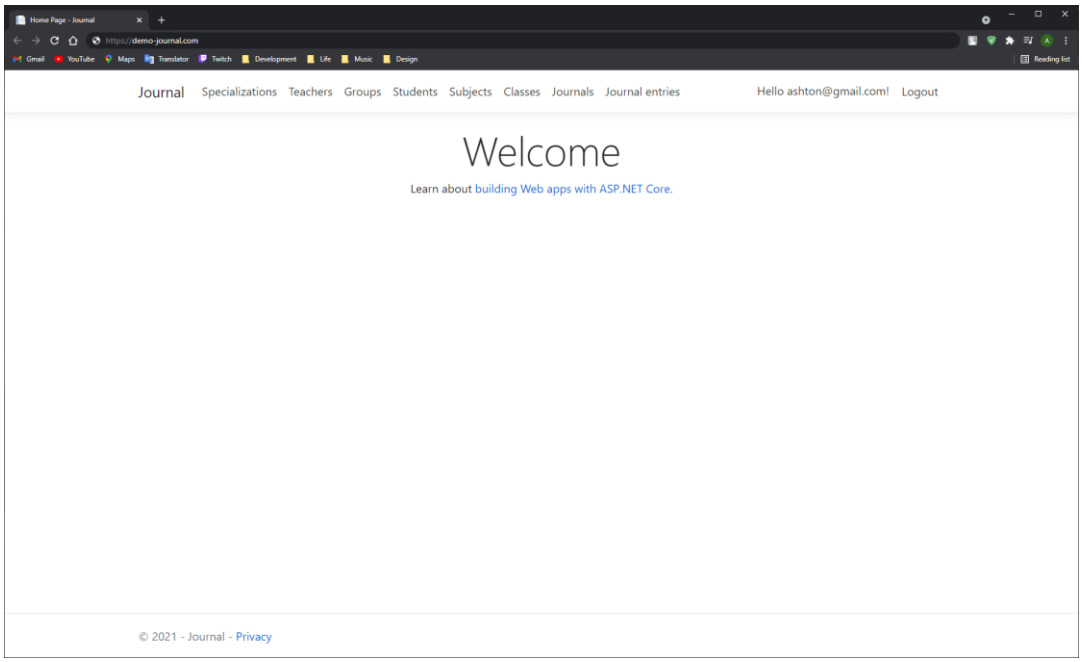

Рисунок 3.2.1 – Головна сторінка веб-додатку (з адресою demo-journal.com)

3.3 Огляд кінцевого виду реалізації проекту

Протягом трьох розділів було спроектовано, реалізовано та опубліковано багатосторінковий веб-додаток з автоматично згенерованим інтерфейсом та впровадженою системою авторизації.

Розглянемо усі сторінки з їх можливостями. Як згадувалось раніше у третьому розділі при підключенні системи авторизації – жоден користувач не зможе скористатися ІС без створеного облікового запису, під яким він заходитиме у систему (рис. 3.3.1).

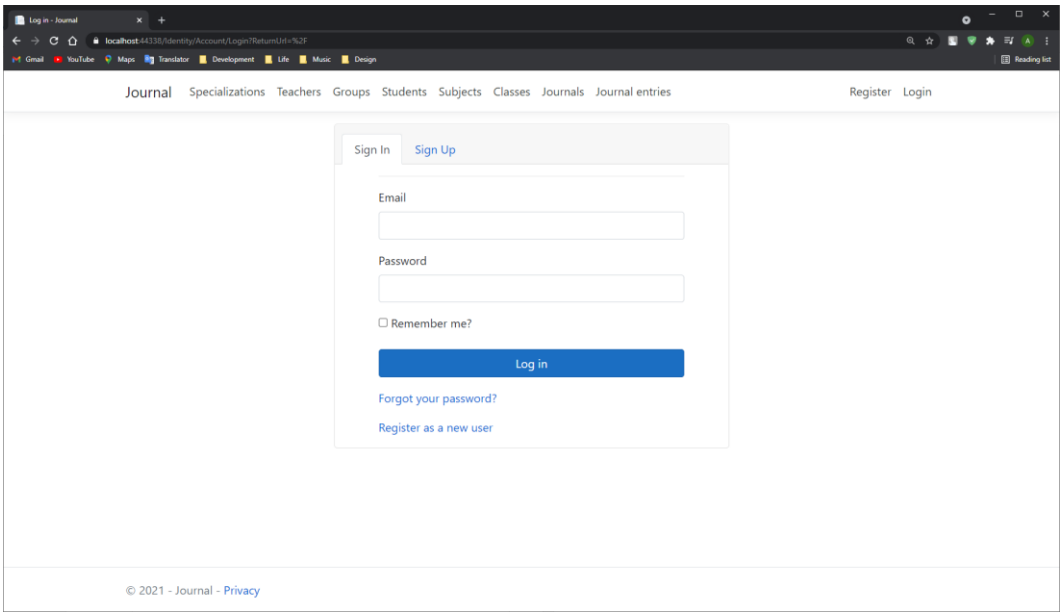

Рисунок 3.3.1 – Форма авторизації

Надалі у користувача буде можливість змінити свої дані у особистому кабінеті (рис. 3.3.2).

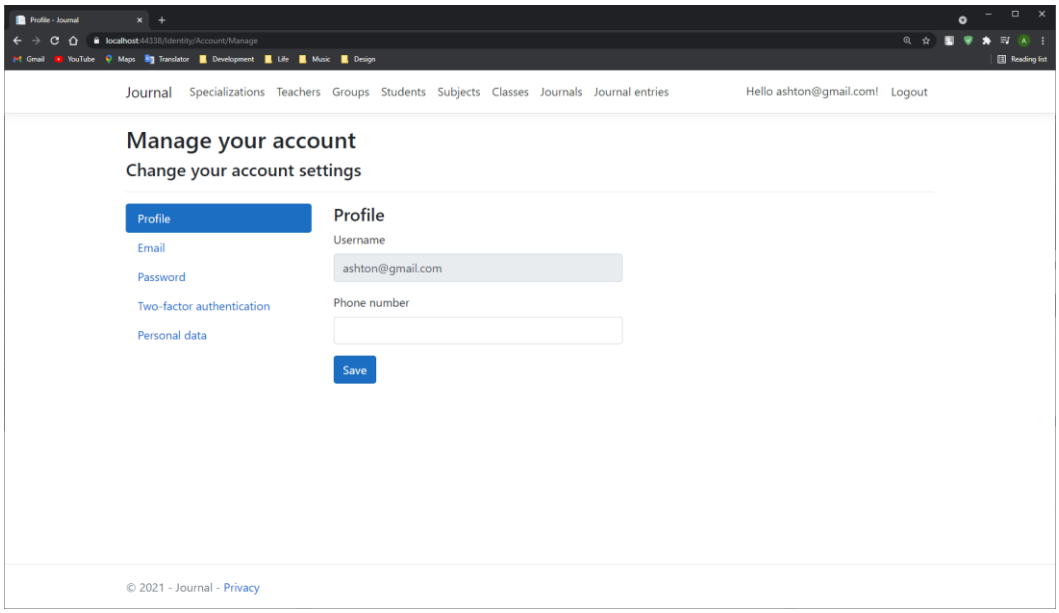

Рисунок 3.3.1 – Особистий кабінет користувача

Після авторизації у систему користувач може перейти до перегляду, редагування та видалення даних у журналі. Це буде відноситися до кожної сторінки у навігаційній панелі. В якості прикладів розглянемо сторінки перегляду навчального Предмету та Журналу. В них присутні елементи керування, що

надають можливості створення, редагування та видалення записів: Create new, Edit та Delete відповідно (рис 3.3.3-3.3.4).

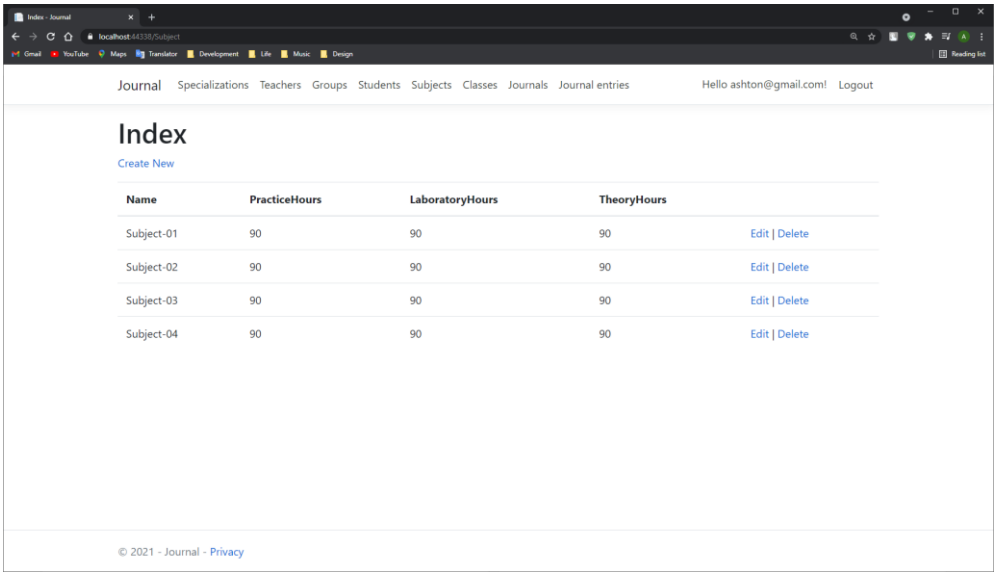

Рисунок 3.3.3 – Сторінка перегляду предметів

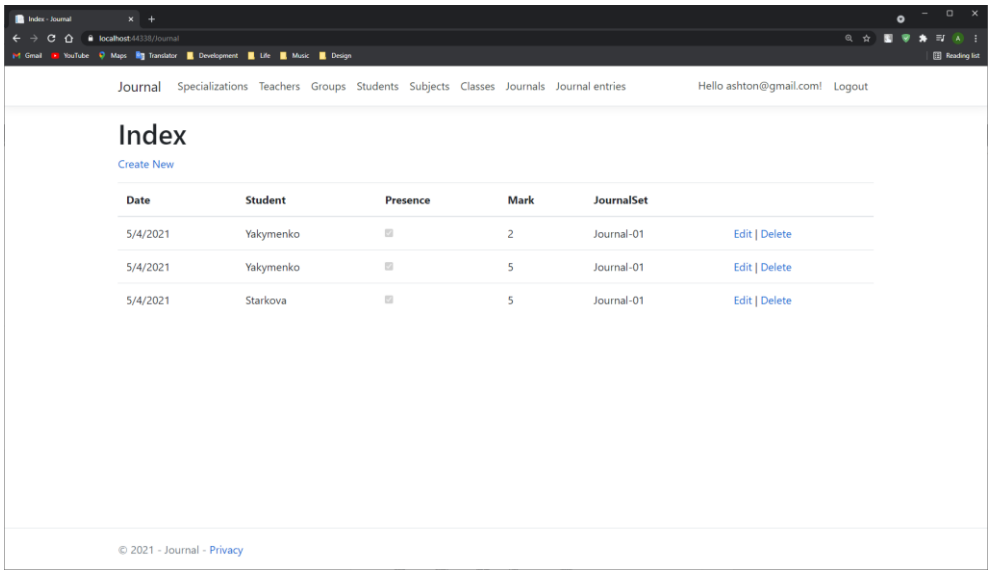

Рисунок 3.3.4 – Сторінка перегляду Журналів

ВИСНОВОК

В рамках написання дипломної роботи були розглянуті проблеми існуючих рішень для підприємств, які бажають підвищити свою ефективність через впровадження цифрового інформаційної системи. Для вирішення даних проблем було запропоновано рішення у вигляді реалізації власного продукту, що дозволяє істотно скоротити витрати. Рішення має на увазі за собою швидкий і ефективний метод розробки програмного комплексу, який буде актуальний в рамках підприємств, в тому числі державного типу, яким доступна невелика команда фахівців.

Щоб задовольнити всі перераховані вище умови було прийнято рішення звернутися до сучасних технологій розробки програмних комплексів - використовувати підхід на основі об'єктних середовищ, фреймворка ASP.NET Core, за шаблоном MVC.

У першому розділі було проаналізовано вимоги до функціонально частини інформаційної системи, вивчені всі нюанси існуючих рішень, а також подальший вибір підходу розробки з подальшою аргументацією на користь технологій .NET.

Під час роботи над другим розділом був обраний шаблон проектування, а також реалізовані основні частини програми: Моделі, Контролери і Уявлення. Кожному відповідний підрозділ містить мінімально необхідну інформацію з описом взаємодії перерахованих вище частин програми.

Створене веб-додаток імітує аналог паперового журналу має вигляд багатосторінкового сайту. На сайті у користувачів є можливість відслідковувати дані по журналу, створювати нові записи, вносити в них зміни і видаляти з бази даних. Попередньо сайт обов'язково буде просити користувача авторизуватися, перш ніж надати йому доступ до ресурсу.

Детальна інструкція публікації додатка в мережу і звітність по його кінцевого виду представлений в третьому розділі.

Перспективи розвитку і поліпшення розроблюваного проекту, як і перспективи розвитку сучасних технологій розробки - безмежні. Дані кошти розробки будуть обов'язково оновлюватися і спрощуватися, це значним чином сприятиме швидкому освоєнню даних технологій майбутніми фахівцями.

Завдяки даним технологіям з'являється можливість поліпшити ефективність в різних інфраструктур від малих до державних підприємств.

ПЕРЕЛІК ПОСИЛАНЬ ТА ДЖЕРЕЛ

1. Фримен А. ASP.NET MVC 3 Framework с примерами на C# для профессионалов: пер. з англ. / А. Фримен, С. Сандерсон. – М.: Вильямс, 2012.
2. Макки А. Введение в .Net 4.0 и Visual Studio 2010 для профессионалов: пер. з англ. / А. Макки. – М.: Вильямс, 2010. Controls Overview – http://msdn.microsoft.com/ru-ru/library/zsyt68f1.aspx
3. Офіційна документація від Microsoft – https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-5.0
4. Загальна збірка навчального матеріалу дисципліни Сучасні інтернет технології – https://moodle.fit.knu.ua/course/view.php?id=1210
5. Особливості ASP.NET технології – https://professorweb.ru/my/ASP_NET/mvc/level1/1_2.php
6. Матеріал для розробників METAHIT – https://metanit.com/sharp/aspnet5/1.1.php
7. .NET Core: можливості та перспективи – https://dou.ua/lenta/articles/net-core/
8. Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC 1st Edition: Jess Chadwick, Todd Snyder, Hrusikesh Panda – https://www.amazon.com/Programming-ASP-NET-MVC-Developing-Applications/dp/1449320317
9. ASP.NET Core MVC з прикладами на C# для професіоналів. 6-е видання: Адам Фрімен – https://balka-book.com/asp_net-63/asp_net_core_mvc_s_primerami_na_c_dlya_professionalov_6_e_izdanie-45999?utm_source=google&utm_medium=cpc&utm_campaign=smartproduct_it_1k&utm_content=autoplace&utm_term=it_1k&gclid=CjwKCAjw47eFBhA9EiwAy8kzNKJcqKpuiEU_eFATyktNCANC5Oof-YTpng-WS0spt0OPipOuc1EFhRoCTsMQAvD_BwE
10. ASP.NET Core 3 з прикладами на C # для професіоналів. Том 1: Адам Фрімен – https://balka-book.com/asp_net-63/asp_net_core_mvc_s_primerami_na_c_dlya_professionalov_6_e_izdanie-

45999?utm_source=google&utm_medium=cpc&utm_campaign=smartproduct_it_1k&utm_content=autoplace&utm_term=it_1k&gclid=CjwKCAjw47eFBhA9EiwAy8kzNKJcqKpuiEU_eFATyktNCANC5Oof-YTpng-WS0spt0OPipOuc1EFhRoCTsMQAvD_BwE

11. Розробка об'єктної моделі, алгоритмів і структури бази даних – https://studfile.net/preview/3021970/#2
12. Що таке C # : itProger – https://itproger.com/course/csharp
13. Введення в ASP.NET Core [Електронний ресурс]: Microsoft -Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-1.0
14. ASP.NET Core: Wikipedia – https://en.wikipedia.org/wiki/ASP.NET_Core
15. ASP.NET Core структура проекту: TutorialsTeacher – http://www.tutorialsteacher.com/core/aspnet-core-application-project-structure
16. ASP.NET Core Tag Helpers: TutorialsPoint – :https://www.tutorialspoint.com/asp.net_core_razor_tag_helpers.htm
17. ASP.NET Core Essential: CourseHunters – https://coursehunters.net/course/asp-net-core-essential
18. Початок роботи з ASP.NET Core і Visual Studio – https://ukr.small-business-tracker.com/getting-started-with-aspnet-core-892506