КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

ВЕБ-ЗАСТОСУНОК РОБОТИ КОМІСІЇ З ПОСЕЛЕННЯ

Виконав студент 4-го курсу
Ярослав ПРИХОДЬКО _____
(підпис)

Науковий керівник:
доцент, кандидат педагогічних наук
Наталія РУСІНА _____
(підпис)

Засвідчую, що в цій роботі немає запозичень праць інших авторів без відповідних посилань.
Студент _____

Роботу розглянуто й допущено до захисту
на засіданні кафедри теорії та технології програмування
«____» _____ 2023 р., протокол № ___

Завідувач кафедри
Микола НІКІТЧЕНКО _____

Київ – 2023

РЕФЕРАТ

Обсяг роботи: загальний - 41 сторінка, з них основний текст – 40 сторінок, 19 ілюстрацій, 3 таблиці, 16 джерел посилань, 1 додаток.

ASP.NET MVC, BOOTSTRAP, ВЕБ-ЗАСТОСУНОК, VIEW, ПРЕДСТАВЛЕННЯ, РОЗРОБКА, БАЗА ДАНИХ, POSTGRES, RAZOR PAGES.

Об’єктом роботи є автоматизація процесів, які пов’язані з роботою комісії з поселення студентів.

Предметом роботи є веб-застосунок для автоматизації роботи комісії з поселення студентів.

Метою роботи є розробка веб-застосунку для автоматизації роботи комісії з поселення студентів, що полегшить процеси комунікації між мешканцями студентських гуртожитків та адміністрацією.

Методи розроблення: створення UML діаграм для аналізу функціональних вимог та проектування веб-застосунку, підхід code-first до проектування бази даних.

Інструменти розроблення: інтегроване середовище розробки Visual Studio; графічний інтерфейс для адміністрування бази даних pgAdmin 4; контейнеризаційний рушій Docker, мови програмування: C#, JavaScript; мови розміток, стилів: HTML; база даних: Postgres; бібліотеки: Bootstrap, jQuery.

Результати роботи: розроблено веб-застосунок для автоматизації роботи комісії з поселення, що готовий до введення в експлуатацію та подальшого розширення функціоналу за потреби користувачів.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 4
ВСТУП 5
РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ АНАЛОГІВ НА РИНКУ 7
1.1 Електронний кабінет мешканця студмістечка КНУ 7
1.2 Кабінет мешканця «SMART Університет» 9
РОЗДІЛ 2.ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ 10
2.1 Інтегроване середовище Visual Studio 10
2.2 Мова програмування C# 10
2.3 Мова розмітки HTML 11
2.3 Платформа .NET 7 12
2.4 Фреймворк ASP.NET Core MVC 13
2.5 Фреймворк Bootstrap 14
2.6 Контейнерезаційний рушій Docker 15
2.7 Набір технологій AJAX 16
РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ 18
3.1 Вимоги до веб-застосунку 18
3.2 Графічне представлення моделі даних 21
3.3 Реалізація веб-застосунку 25
РОЗДІЛ 4. ІНСТРУКЦІЇ З КОРИСТУВАННЯ 31
ВИСНОВКИ 38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39
ДОДАТКИ 41
ДОДАТОК А. Діаграма сутностей 41

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

HTML – Hyper Text Markup Language, мова гіпертекстної розмітки.

C# – Мова програмування

XML – Extensible Markup Language, мова розмітки

HTTP – HyperText Transfer Protocol, протокол передачі даних

КНУ – Київський національний університет імені Тараса Шевченка

AJAX – Asynchronous JavaScript and XML

IDE – інтегроване середовище розробки

ASP.NET – Active Server Pages for .NET

MVC – Model View Controller

UML – Unified Modeling Language

CSS – Cascading Style Sheets

.xlsx – розширення файлів-таблиць

ВСТУП

Оцінка сучасного стану об’єкта розробки. У сучасних реаліях з приходом інформаційних та мережевих технологій з'явилася можливість автоматизувати багато процесів що раніше виконувались виключно вручну, у більшості цих процесів є одна спільна риса – одноманітність виконуваних завдань. Робота комісії з поселення у студентських гуртожитках також складається з великої кількості повторюваної роботи, такої як поселення та виселення студентів, зворотній зв'язок зі студентами, тощо. На цей момент, існує велика кількість сайтів студмістечок та і самих гуртожитків, проте майже всі вони створюються з довідковою метою, де зібрана інформація про гуртожитки яка до того ж нерідко є неповною.

Актуальність роботи та підстави для її виконання. Будь яка університетська організація має досить велику кількість ручної роботи пов'язаної з отримання шаблонних документів та комунікацією зі студентами. Автоматизація виконання подібних завдань значно полегшить ці процеси, надасть можливість централізовано керувати всіма етапами процесу поселення, дозволить електронну обробку документів та забезпечить швидкий обмін інформацією між членами комісії й студентами включаючи збір та обробку документів, розгляд звернень, заповнення форм та ін.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення веб-застосунку для автоматизації роботи комісії з поселення.

Для досягнення мети поставлено наступні завдання:

- оглянути існуючі аналоги на ринку в Україні;
- проаналізувати роботу комісії з поселення з точки зору автоматизації комунікації з мешканцями гуртожитку;
- сформулювати функціональні вимоги до веб-застосунку та розробити модель даних;
- реалізувати веб-застосунок.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є автоматизація процесів, які пов’язані з роботою комісії з поселення студентів, що зможе полегшити ці процеси та отримати централізований портал для комунікації зі студентами.

В якості засобів розробки було використано інтегроване середовище розробки Visual Studio 2022 [1], що є відкритим та безкоштовним для особистого користування. Для дослідження стану бази даних під час розробки використано pgAdmin [9] – безкоштовне програмне забезпечення для підключення до бази даних та отримання графічного представлення даних в ній. Git [10] – відкрите програмне забезпечення для контролю версій. Контейнеризаційний рушій Docker [8] для запуску бази даних та системи адміністрування баз даних.

Можливі сфери застосування. Веб-застосунок такого типу є актуальним не тільки на рівні роботи комісії з поселення студентів, а й на рівні автоматизації процесів комунікації між студентами та адміністрацією загалом на рівні університету, що дозволяє створити централізований ресурс.

РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ АНАЛОГІВ НА РИНКУ

1.1 Електронний кабінет мешканця студмістечка КНУ

Розглянемо існуючий аналог на ринку, це електронний кабінет мешканця студмістечка КНУ.

Електронний кабінет мешканця студмістечка КНУ – це веб-застосунок що дозволяє замовляти в дирекції довідки й документи, послуги слюсаря чи електрика [6].

В електронному кабінеті доступний функціонал (рисунок 1.1):

- можливість реєстрації (зауважимо що для початку повноцінного користування кабінетом мешканця, необхідно його активувати, до процесу активації належить заповнення особистої інформації, в тому числі номер кімнати та номер гуртожитку) та входу користувачів;
- можливість замовити документи різних типів (тип користувач вказує самостійно);
- можливість зняття з місця реєстрації (підпис паспортиста), де користувач завантажує копію свого паспорта та прикріпляє до звернення;
- можливість замовити послуги столяра (що суперечить опису на головній сторінці кабінету) чи електрика, що здійснюється шляхом заповнення спільної форми де вказується необов’язковий коментар, дата та час, коли буде зручно мешканцю, а також обирається послуга якого саме зі спеціалістів необхідна та яка саме.

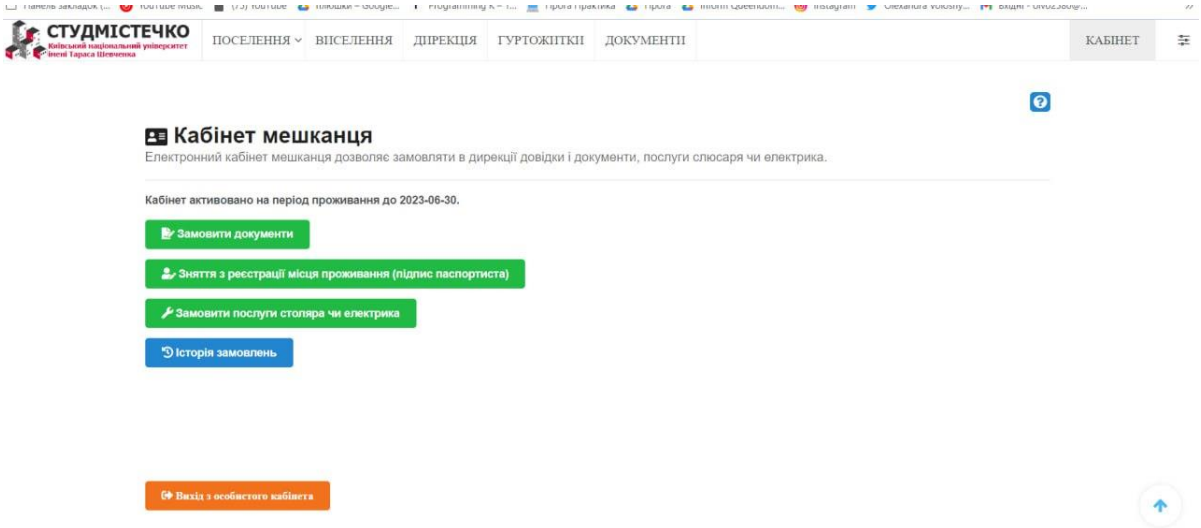

Рисунок 1.1 – Фрагмент особистої сторінки

Проаналізувавши функціонал цього веб-застосунку можна виділити наступні його переваги:

- автоматизує частину комунікаційної робити між дирекцією студмістечка та мешканцями гуртожитків, що пов’язана з поданням та розглядом заявок на виконання ремонтних робіт у кімнатах мешканців;
- автоматизує частину паперової роботи з отримання деяких документів від мешканців (копії паспорту).

Але окрім переваг він також має й недоліки:

- процес реєстрації має непотрібне продовження у вигляді активації кабінету мешканця, що робить гіршим користувацький досвід, до того ж без процесу активації кабінет лишається практично марним для користувача;
- кабінет орієнтований виключно на роботу між мешканцями гуртожитків студмістечка та дирекцією студмістечка;
- кабінет має застарілий користувацький інтерфейс, що погіршує досвід користування веб-застосунком.

1.2 Кабінет мешканця «SMART Університет»

Відділ цифрової трансформації західно-українського національного університету пропонує своїм студентам інформаційну систему «SMART Університет» [7], що дозволяє цифровізувати велику кількість процесів комунікації. У цій системі наявний окремий кабінет мешканця, який має призначення централізувати процеси пов’язані з проживанням студентів у гуртожитках. Як зазначено на головні сторінці системи:

У кабінеті мешканця гуртожитку мешканець може знайти інформацію про своє місце проживання, подати заяву на поселення та переселення, подати заявку на ремонт, переглядати історію заяв, спілкуватися з адміністрацією гуртожитку, а також здійснювати онлайн-оплату за проживання (рисунок 1.2) [7].

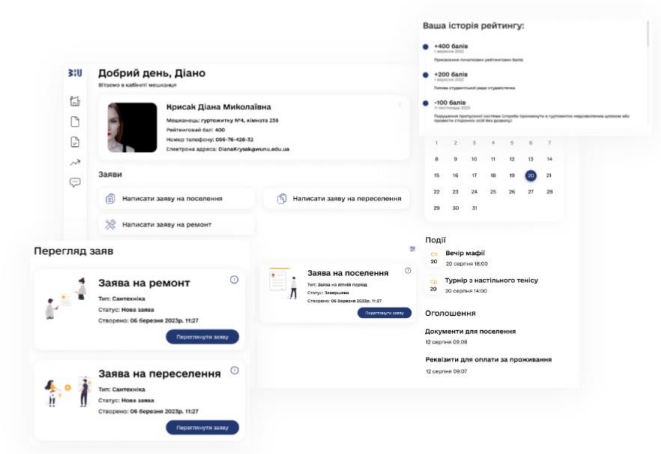

Рисунок 1.2 – Фрагмент головної сторінки системи

Інформаційна система "SMART Університет" цифровізує процеси проживання студентів у гуртожитках, забезпечує зручний доступ до інформації, заявок та спілкування з адміністрацією.

РОЗДІЛ 2.ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Інтегроване середовище Visual Studio

Visual Studio — це комплексне інтегроване середовище розробки [11], яке можна використовувати для написання, редагування, налагодження та створення коду, а потім розгортання програми. Окрім редагування та налагодження коду, Visual Studio містить компілятори, інструменти завершення коду, керування джерелами, розширення та багато інших функцій для покращення кожного етапу процесу розробки програмного забезпечення [2].

Visual Studio надає можливість створювати, налагоджувати та тестувати програми на платформі .NET і мовою C++, редагувати сторінки ASP.NET у веб-дизайнері, розробляти крос-платформні програми для мобільних і настільних комп'ютерів з використанням .NET або створювати адаптивний веб-інтерфейс користувача з використанням C# [12]. Серед версій було обрано Community Edition так як, вона є безкоштовною для студенів та при цьому повнофункціональною IDE.

2.2 Мова програмування C#

Мова програмування C# [12], розроблена компанією Microsoft. Вона є частиною платформи .NET і широко використовується для розробки різноманітних типів програм, включаючи веб-застосунки, настільні програми, мобільні застосунки та багато іншого.

До особливостей мови C# можна віднести:

- C# є сильно типізованою мовою, що означає, що типи даних повинні бути вказані і перевірені в процесі компіляції, але окрім цього вона також підтримує і елементи «не безпечного» програмування такі як вказівники, димічний тип даних тощо;
- C# підтримує основні принципи об'єктно-орієнтованого програмування, такі як інкапсуляція, наслідування та поліморфізм;

- окрім об'єктно-орієнтованого програмування, C# також підтримує елементи функціонального програмування, асинхронного програмування, LINQ (Language Integrated Query) та багато іншого;
- C# має багатий набір бібліотек та фреймворків, що дозволяють виконувати різноманітні завдання, такі як робота з базами даних, мережеве програмування, робота з файлами, написання графічного інтерфейсу тощо [12].

2.3 Мова розмітки HTML

HTML (Hyper Text Markup Language) є мовою розмітки, яка використовується для створення структури і вигляду веб-сторінок [3]. Вона визначає, як вміст сторінки повинен бути структурований та відображений у веб-браузері [13].

До основних концепцій HTML варто віднести:

- теги: HTML використовує теги для опису елементів веб-сторінки. Теги зазвичай мають відкриваючий <tag> і закриваючий </tag> теги, наприклад <p> для параграфу або <h1> для заголовка першого рівня;
- атрибути: теги можуть мати атрибути, які визначають додаткові властивості елементів. Наприклад, атрибут src в тезі <img src="image.jpg"> вказує шлях до зображення;
- структура: HTML визначає структуру сторінки за допомогою блочних елементів, таких як і <section>, а також лінійних елементів, таких як <p> і , що дозволяє групувати елементи та встановлювати їх взаємне розташування;
- посилання: HTML дозволяє створювати посилання на інші сторінки, документи або ресурси за допомогою тегу <a>, адресу посилання необхідно вказати за допомогою атрибута href;

- зображення: HTML дозволяє вставляти зображення на сторінку за допомогою тегу <img>, наявна можливість вказати шлях до зображення за допомогою атрибута src і задати альтернативний текст за допомогою атрибута alt;
- таблиці: HTML дозволяє створювати таблиці з даними за допомогою тегів <table>, <tr> і <td>, можна встановлювати заголовки, злиті комірки, стиль рядків і багато іншого;
- форми: HTML дозволяє створювати веб-форми для введення даних використовуючи тег <form> та елементи форми, такі як <input>, <select>, <textarea> і багато інших, введені дані можна надіслати на сервер для обробки.

2.3 Платформа .NET 7

Платформа .NET є високорівневим середовищем для розробки програмного забезпечення, розробленим компанією Microsoft [12]. Основні риси та можливості платформи .NET:

- платформа .NET підтримує кілька мов програмування, зокрема C#, Visual Basic.NET, F# та інші, тобто можна обрати ту мову яка найкраще підходить потреби розробки;
- за допомогою платформи .NET можна розробляти застосунки, які працюють на різних операційних системах, таких як Windows, macOS і Linux. Це реалізується за допомогою .NET Core, крос-платформного фреймворку, який є складовою частиною платформи .NET;
- .NET використовує спеціальну віртуальну машину під назвою Common Language Runtime (CLR), яка виконує керований код, це дозволяє забезпечити безпеку, контроль пам'яті та інші переваги, а також спрощує розробку та підтримку програмного забезпечення;

- платформа .NET має велику кількість готових бібліотек та компонентів, які значно спрощують розробку різних типів застосунків, включаючи веб-застосунки, мобільні застосунки, бази даних і багато іншого[4];
- для розробки на платформі .NET можна використовувати різні IDE, але найпопулярнішим є Visual Studio, воно надає потужні інструменти для розробки, налагодження і тестування застосунків на платформі .NET.

2.4 Фреймворк ASP.NET Core MVC

ASP.NET MVC (Model-View-Controller) є одним із фреймворків розробки веб-застосунків на платформі .NET. Він надає шаблон архітектури MVC, що дозволяє ефективно розділити логіку додатку на модель (Model), представлення (View) та контролер (Controller) [12].

Основні особливості та переваги ASP.NET MVC на базі платформи .NET:

- архітектура MVC дозволяє розділити логіку додатку на компоненти. Модель відповідає за обробку даних та бізнес-логіку, представлення відображає дані та інтерфейс користувача, а контролер керує потоком даних між моделлю та представленням;
- ASP.NET MVC дозволяє використовувати різні технології для реалізації представлення, включаючи HTML, CSS, JavaScript, а також різноманітні фреймворки для розробки фронтенду, такі як Angular або React;
- ASP.NET MVC надає можливість створювати як традиційні веб-сторінки, так і веб-сервіси API для забезпечення зв'язку з іншими застосунками;

- ASP.NET MVC дозволяє розширювати функціональність за допомогою власних компонентів, фільтрів, маршрутів та інших розширень;
- фреймворк сприяє тестуванню застосунків, оскільки логіка розподілена на окремі компоненти, які можна тестувати окремо;
- ASP.NET MVC надає різноманітні засоби для забезпечення безпеки застосунків, включаючи автентифікацію, авторизацію та захист від атак.

2.5 Фреймворк Bootstrap

Bootstrap – це фреймворк для розробки веб-інтерфейсів. Він забезпечує швидкий та простий спосіб створення стильних та адаптивних веб-сайтів [14]. Основні особливості Bootstrap:

- Bootstrap надає широкий набір готових компонентів, таких як кнопки, форми, меню, картки, модальні вікна та багато інших. Це дозволяє ефективно будувати інтерфейси без необхідності писати весь код з нуля;
- Одна з ключових особливостей Bootstrap – це підтримка адаптивного дизайну – сайти, побудовані на Bootstrap, автоматично пристосовуються до різних розмірів екранів, включаючи мобільні пристрої, планшети та настільні комп'ютери, це забезпечує зручне та коректне відображення контенту на різних пристроях;
- Bootstrap має розширену систему налаштувань та можливість використовувати готові теми, є можливість налаштувати вигляд компонентів, кольори, шрифти та інші параметри за допомогою CSS-класів або змінювати значення змінних;
- Bootstrap також надає набір JavaScript компонентів, таких як слайдери, випадаючі меню, модальні вікна, підказки та багато інших,

вони працюють на основі бібліотеки jQuery і допомагають забезпечити інтерактивність та функціональність на веб-сторінках;

- Bootstrap підтримує всі сучасні браузери і забезпечує сумісність з останніми версіями HTML, CSS та JavaScript, а також має активну спільноту розробників, яка постійно вносить зміни до фреймворку, публікує нові релізи та надає допомогу користувачам [15].

2.6 Контейнерезаційний рушій Docker

Docker – це відкрите програмне забезпечення, яке дозволяє упаковувати, розгортати та управляти програмними додатками у віртуалізованому середовищі, відомому як контейнер [5]. Контейнери Docker використовують контейнеризацію, що дає змогу запускати застосунки та їх залежності у віртуальному ізольованому середовищі безпосередньо на операційній системі.

Основні поняття та можливості Docker [8]:

- Docker упаковує застосунок та всі його залежності (бібліотеки, середовище виконання і т.д.) в контейнер, який є виконавчим екземпляром цього додатку, що може бути запущений та зупинений незалежно від інших контейнерів, кожен контейнер використовує спільне ядро операційної системи, але має власні ізольовані ресурси;
- образ Docker є статичним пакетом, який містить всі необхідні файли та налаштування для запуску контейнера, образи використовуються для створення контейнерів і можуть бути побудовані з використанням Dockerfile або завантажені з реєстру образів Docker;
- Dockerfile – це текстовий файл, в якому описується конфігурація образу Docker. Він містить команди для встановлення залежностей, копіювання файлів, виконання налаштувань тощо. За допомогою Dockerfile можна автоматизувати процес побудови образу;
- реєстр образів Docker (Docker Registry) – це централізоване сховище для образів Docker, що надає можливість завантажувати готові

образи з публічного реєстру, такого як Docker Hub, або створювати власний приватний реєстр для зберігання та розповсюдження власних образів;

- Docker Compose – це інструмент, який дозволяє визначати та управляти багатоконтейнерними додатками, за допомогою файлу конфігурації Docker Compose можна визначити сервіси, мережі, змінні середовища та інші налаштування для запуску і взаємодії декількох контейнерів;

2.7 Набір технологій AJAX

AJAX (Asynchronous JavaScript and XML) – це набір технологій, що дозволяють взаємодіяти з сервером без перезавантаження сторінки. AJAX використовує комбінацію JavaScript, XML (але також може використовувати JSON або інші формати даних) і HTTP-запитів для асинхронного обміну даними між клієнтом (браузером) і сервером [16].

Основні принципи та компоненти AJAX:

- JavaScript є основною мовою програмування, яка використовується для взаємодії з DOM (Document Object Model) сторінки, обробки подій і виконання асинхронних запитів до сервера;
- XMLHttpRequest: Об'єкт XMLHttpRequest – це основний механізм, який дозволяє виконувати асинхронні HTTP-запити до сервера, за допомогою цього об'єкта можна надсилати запити на сервер, отримувати відповіді та маніпулювати даними;
- після виконання запиту до сервера і отримання відповіді, AJAX дозволяє обробляти ці дані без перезавантаження сторінки. Це може включати оновлення певних елементів сторінки, динамічне додавання або видалення контенту, зміну стилів тощо;
- одна з ключових особливостей AJAX - це асинхронність, що означає, що запити до сервера виконуються незалежно від основного потоку

веб-сторінки. Це дозволяє користувачам продовжувати взаємодіяти зі сторінкою, не очікуючи завершення запитів.

Вказані технології були обрані на основі переваг, які вони надають в процесі розробки. Акцент при виборі технологій був на полегшенні та шаблонізації процесу розробки, що дозволяє створити веб-застосунок, який буде достатньо легко підтримувати, так як він міститиме мінімальну кількість штучно ускладнених конструкцій.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Вимоги до веб-застосунку

До часто виконуваних завдань комісії з поселення студентів можна віднести:

- повідомлення студентів про необхідність надсилання деякої інформації про себе (переважно через google forms);
- збір документів від певних груп студентів;
- обробка звернень студентів та абітурієнтів пов’язаних з певними питаннями щодо проживання, пролонгації договору на проживання, поселення, виселення тощо.

На основі вищенаведених завдань можна сформувати наступні вимоги:

- веб-застосунок повинен надавати можливість користувачам комунікувати з адміністрацією через механізм звернень до комісії з поселення, а членам комісії з поселення в свою чергу відповідати на такі звернення;
- члени комісії з поселення повинні мати можливість збирати однотипну інформацію, документи певної групи користувачів, а також сповістити їх про щось за допомогою текстового повідомлення.

Конкретизуємо вимоги на рівні функцій веб-застосунку.

Можливості для членів комісії з поселення:

- створити роль для подальшого надання певних послуг певним групам користувачів, що належать до цієї ролі;
- створити сповіщення для всіх користувачів певної групи (ролі) та відповідно побачити хто з користувачів вже прочитав сповіщення, а також необов’язкова опція прикріпити посилання з назвою під час створення;

- створити форму для заповнення користувачами певної групи (ролі), для кожного створюваного поля вказати тип (рядок, число, можливо інші), а також позначити деякі з полів обов’язковими, вказати час відкриття та час закриття форми, відкрити або закрити самостійно, завантажити результати в форматі .xlsx;
- створити документний тег, що працюватиме подібно до форми, але буде призначений для збору документів від студентів певної групи (ролі), відповідно вказавши час відкриття, час закриття, відкрити або закрити самостійно, завантажити всі прикріплені документи;
- обробити звернення користувачів що дозволить за необхідності створити чат з користувачем та комісією для обговорення та закрити звернення після вирішення питання, що обговорюється у зверненні.

Можливості для користувачів:

- реєстрації та редагування власної особистої інформації.
- переглянути нові та старі сповіщення, позначити як прочитане.
- переглянути доступні та обрати форму для заповнення, заповнити форму, а також у разі якщо форма передбачає редагування відповідей то відредагувати відповідь.
- переглянути власні документи, що прикріплені до документних тегів, додати документ до доступного тегу в якому ще немає документа користувача.
- створити звернення до комісії з поселення студентів та отримати відповідно відповідь у чаті, що може бути створений комісією з поселення.

Варто зауважити що член комісії з поселення є лише учасником ролі з додатковими можливостями, тобто абсолютно всі вимоги, що наведені з боку користувача мають бути реалізовані й з боку члена комісії з поселення.

На основі цих пунктів було створено дві Use-Case діаграми вимог щодо функціональності веб-застосунку з боку звичайного користувача (рисунок 3.1) та члена комісії з поселення (рисунок 3.2).

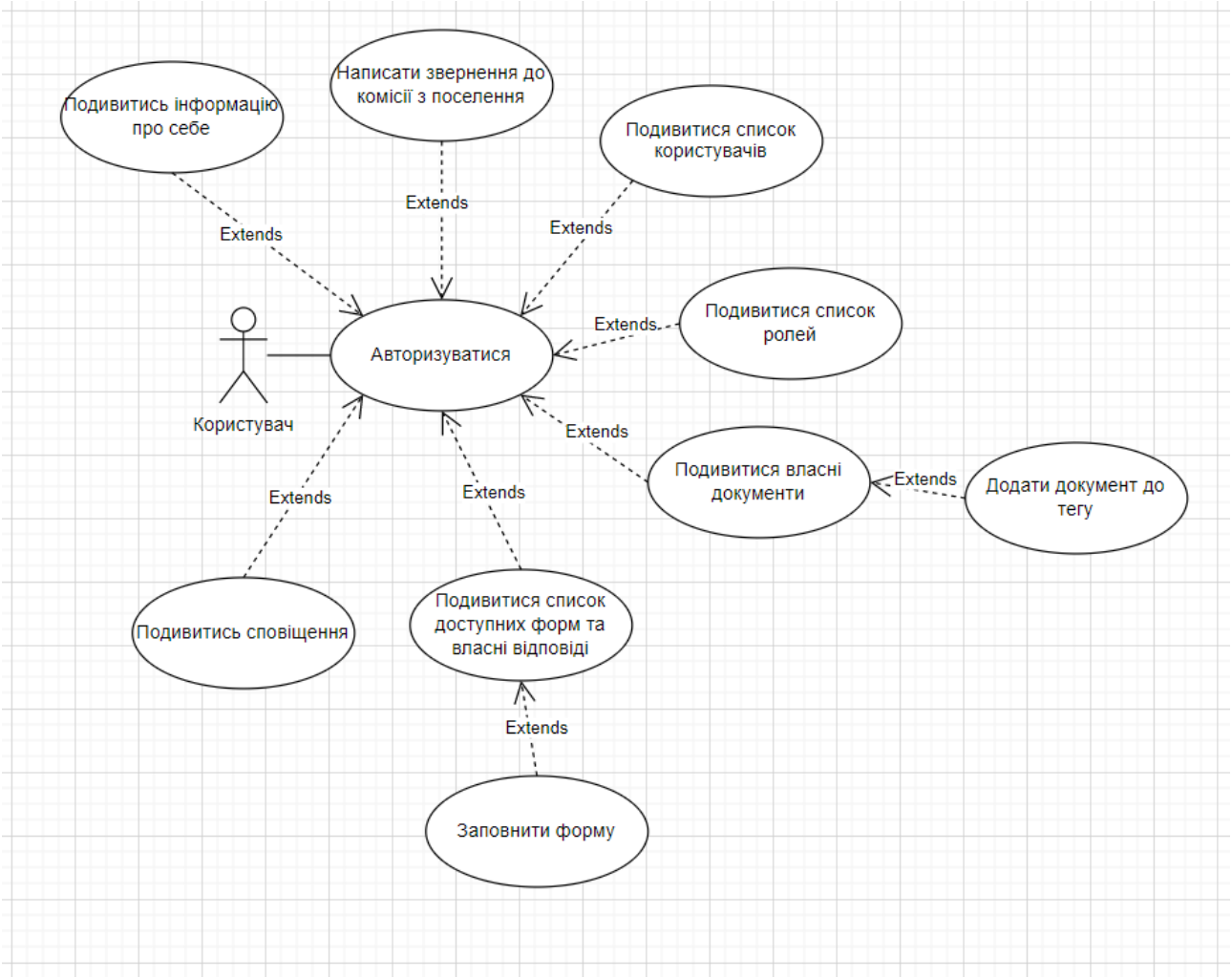

Рисунок 3.1 – Use-Case діаграма користувача

Зазначимо що у такій моделі відсутній прямий поділ на вже поселених студентів, непоселених студентів та абітурієнтів, що претендують на поселення, в цьому випадку всі користувачі можуть користуватися функціоналом система, так як нерідко трапляється ситуація коли ще не поселеній особі необхідно обмінюватися інформацією з комісією з поселення (передати документи необхідні для поселення або реєстрації в черзі на поселення), у такій ситуації члену комісії поселення буде необхідно лише створити необхідну роль та додавати користувачів які повинні надати необхідну інформацію.

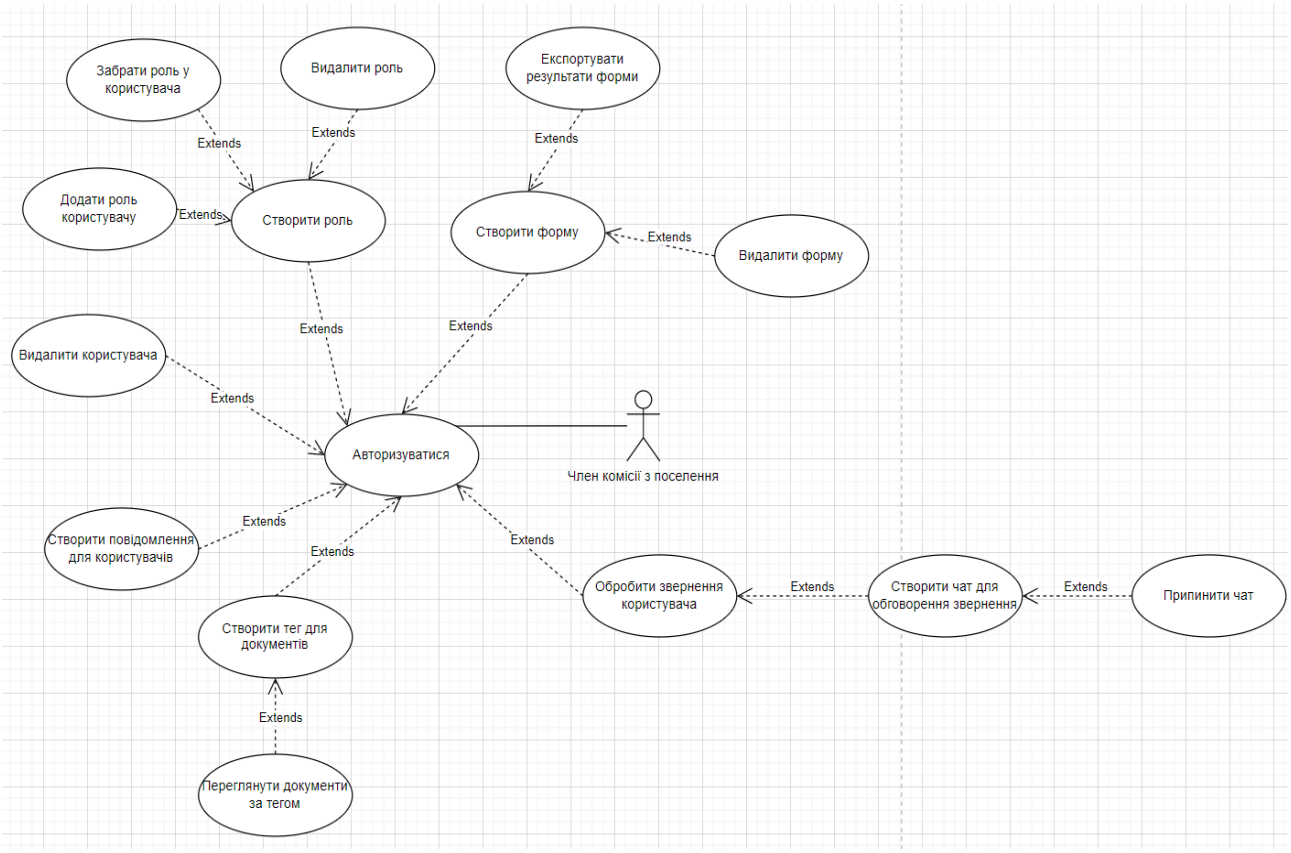

Рисунок 3.2 – Use-Case діаграма члена комісії з поселення

3.2 Графічне представлення моделі даних

Перед початком розробки на основі функціональних вимог необхідно скласти діаграму сутностей (додаток А), що дозволить правильно спроектувати логіку та систему класів, що відносяться до моделі. Від правильно спроектованої моделі даних залежить легкість внесення змін до існуючого та розширення веб-застосунку новим функціоналом, тому цей етам є дуже важливим.

За основу був взятий підхід, де присутня деяка центричність навколо сутності користувача та його ролей, що дозволяє керувати доступом до різних частин функціоналу на основі цих ролей (форм, документних тегів тощо). Також наведено детальний опис всіх сутностей що були утворені (таблиця 1).

Таблиця 1 – Сутності моделі даних

Сутність	Властивості	Опис

User	Id* Username FirstName Patronymic LastName Email PhoneNumber TelegramNickname? PasswordHash Roles	Репрезентує користувача веб-застосунку, зберігає базову особисту інформацію, всю необхідну контактну інформацію та авторизаційну інформацію.
Role	* Id + Name + IsSystem + Description?	Репрезентує роль користувача, має ім’я, опціональний опис та позначення чи є системною (системними ролями є «мешканець», «член комісії з поселення», «голова комісії з поселення»)
FormDefinition	Id * Title Description FieldDefinitions CreatedTimeStamp StartTimeStamp EndTimeStamp IsClosed CanEditAnswers CreatorId	Репрезентує форму, що створюється членами комісії з послення, яка описує такі властивості форми як назва, опис, дата та час створення, початок та кінець дії, чи є закритою, чи можна редагувати відповіді, інформацію про користувача що створив форму та роль для кого призначена форма, а також безпосередньо визначення полів

	RoleId	та їхніх типів у визначеному форматі.
FormAnswer	Id * FilledTimeStamp UpdateTimeStamp Data FormId UserId	Репрезентує відповідь користувача на форму, містить безпосередню інформацію про дані що були записані в поля форми у визначеному форматі, а також дату та час заповнення та редагування, а також посилання на користувача що заповнив форму та на саму форму.
GlobalNotification	Id * Title Message TimeStamp Link LinkName IsuuerId RoleId	Репрезентує сповіщення від комісії з поселення користувачам певної ролі. Містить інформацію про вміст сповіщення, автора та призначення (посилання на роль користувачів для кого сповіщення).
UserNotification	Id * IsRead ReadTimeStamp? UserId GlobalNotificationId	Репрезентує сповіщення що надійшло користувачу, повинно формуватися для всіх користувачів відповідної ролі коли створюється GlobalNotification
File	Id * Title Data	Репрезентує файл. Так як файли можуть зберігатися у різному форматі, окреме виділення

		подібної абстракції надасть можливість розширення функціоналу до використання файлів у різних форматах (посилання на файл, сам вміст файлу або щось інше).
DocumentTag	Id * Title Description? CreatedTimeStamp StartTimeStamp EndTimeStamp IsClosed FileId RoleId CreatorId	Репрезентує документний тег, містить інформацію схожу до форми, щодо дати та часу створення, початку та кінця форми і тд, має посилання на файл в якому міститься шаблон для документу якщо він необхідний, а також роль та користувача що створив даний тег.
Document	Id * FileId UserId DocumentTagId	Репрезентує документ що користувач прикріпив до документного тегу. Набір посилань на інші сутності (файл, користувач, документний тег)
ApplicationToCommision	Id * Title Message CreatedTimeStamp IsClosed UserId	Репрезентує звернення до комісії з поселення, яке створює користувач, містисть інформацію про безпосередньо звернення та має посилання на користувача що створив звернення.

ChatWithCommission	Id * ApplicationToCommissionId	Репрезентує чат з комісією, що створюється на основі звернення.
ChatMessage	Id * Text UserId	Репрезентує повідомлення у чаті щодо звернення до комісії з поселення користувача.

Розроблена початкова модель також може бути вдосконалена в майбутньому за допомогою механізмів міграцій в базах даних.

3.3 Реалізація веб-застосунку

Для формування бази даних та реалізації доступу до даних було використано підхід code-first що дозволяє, написавши класи моделі (рисунок 3.3), сформувати базу даних на основі них за допомогою Entity Framework (рисунок 3.4). Це в свою чергу, дозволяє походу розробки у разі необхідності досить швидко змінити модель даних, так як замість реального джерела даних може використовуватись In-memory база даних, що насправді є лише набором C# колекцій, що зберігаються в оперативній пам’яті яку займає запущений веб-застосунок.

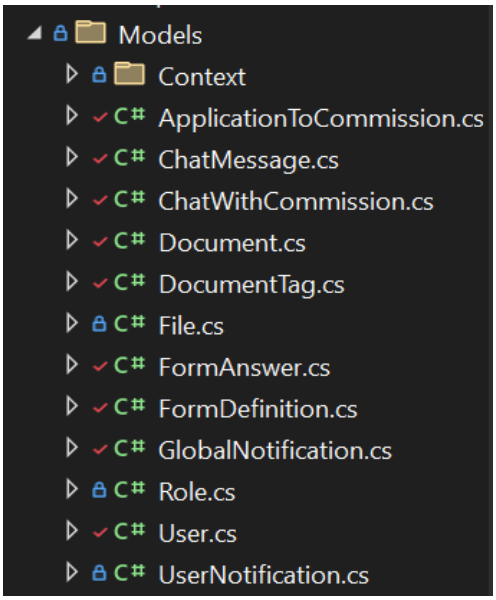

Рисунок 3.3 – Моделі, написані до формування бази даних

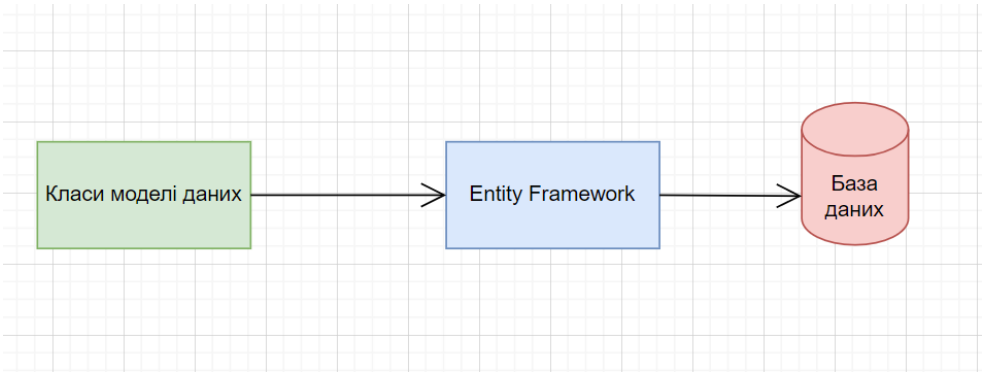

Рисунок 3.4 – Діаграма code-first підходу

Entity Framework підтримує досить широкий набір реляційних баз даних. В якості бази даних було обрано Postgres [9], так як ця база даних надає широкий набір типів та інших корисних інструментів що можуть знадобитися в процесі розробки або подальшого розвитку веб-застосунку. Також з переваг є можливість запуску як самої бази так і користувацького графічного інтерфейсу для адміністрування всередині Docker контейнерів, що дозволяє не встановлювати зайве програмне забезпечення, а також мати відтворюване середовище для розробки та тестування.

Для самого веб-застосунку на рівні інтерфейсу та програмної логіки використано фреймворк ASP.NET Core MVC на базі платформи .NET, яка

дозволяє реалізовувати складні застосунки мовою C# як основною. Цей фреймворк дозволяє будувати пайплайн обробки запиту, де кожен компонент обробляє запит по черзі, після чого передає запит наступному компоненту. Останній обробник надсилає відповідь клієнту. У випадку MVC останнім обробником є контролери, що обробляють запит згідно шляху посилання та вмісту.

MVC – це шаблон проєктування, що дозволяє відокремлювати дані (Model), логіку що оперує цими даними (Controller), представлення цих даних (View) (рисунок 3.5). Це дозволяє досить просто додавати нові точки обробки запитів (endpoint), просто створивши відповідний метод в контролері та представлення, саме тому була обрана саме така модель.

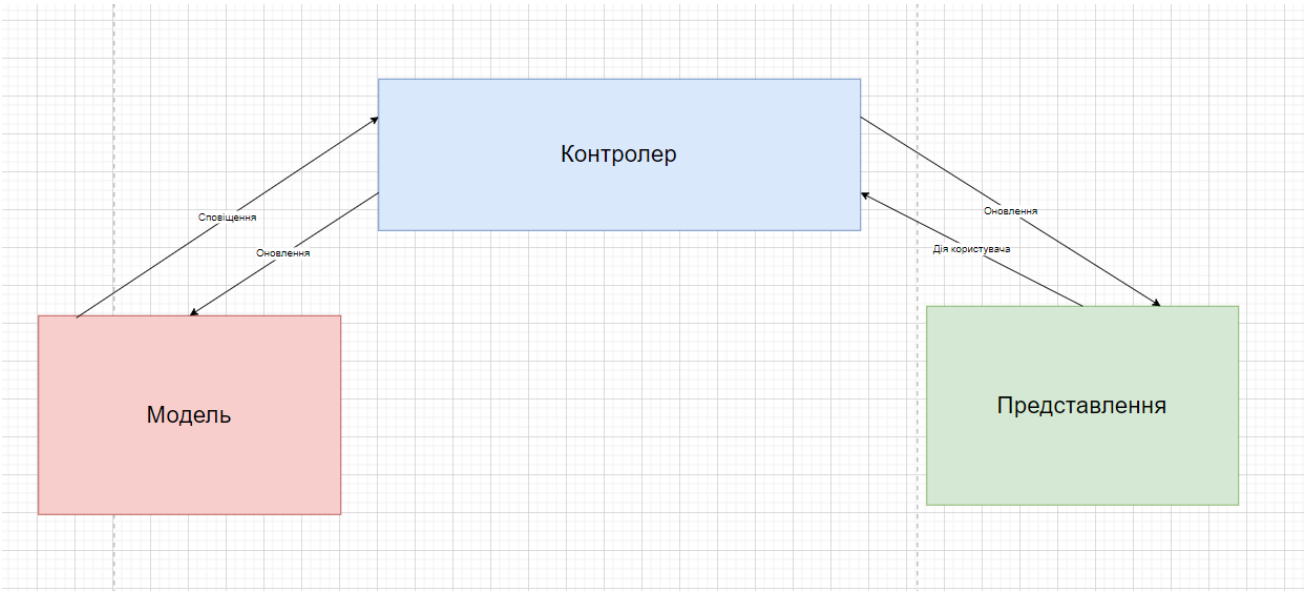

Рисунок 3.5 – Схема роботи шаблону проєктування MVC

В реалізації веб-застосунку наявні такі контролери:

- AccountController – відповідає за авторизацію та реєстрацію користувачів;
- ProfileController – відповідає за надання особистої сторінки користувача, яка налічує такі вкладки як «Інформація», «Сповіщення», «Документи»;

- UsersController – відповідає за надання списку користувачів, відповідних ролей, фільтрацію списку за ролями, а також за керування цими сутностями (створення нової ролі, видалення існуючої ролі, редагування існуючої ролі, додавання ролі користувачу, видалення ролі у користувача);
- HomeController – відповідає за надання базових сторінок, такі як «Головна», «Не знайдено» і тд;
- DocumentsController – відповідає за роботу з документами (створення нового документного тегу, прикріплення документу до документного тегу, список документів для документного тегу, список документних тегів, завантаження документів зібраних у документному тегу, закриття документного тегу);
- FormsController – відповідає за роботу з формами (створення нової форми, заповнення форми, редагування відповіді на форму, перегляд списку форм, закриття форми, видалення форми, завантаження результатів форми, перегляд результатів форми);
- NotificationsController – відповідає за роботу зі сповіщеннями (створення нового сповіщення, перегляд статусу сповіщень у користувачів, позначити сповіщення як прочитане);
- ApplicationsController – відповідає за роботу зі зверненнями користувачів до комісії з поселення (створити нове звернення, закрити звернення, створити чат за зверненням, переглянути чат за зверненням, переглянути список звернень, надіслати повідомлення у чат за зверненням).

Додатково за всіма сутностями були створені API контролери для управління цими сутностями у разі розробки додаткових клієнтів.

Для представлень у фреймворку ASP.NET Core MVC використовується технологія Razor Pages, що дозволяє формувати файли html шаблонів в яких присутні кодові вставки мовою C# що дозволяє використовувати такі можливості як часткове представлення (для повторного використання шаблонних представлень однієї моделі), умовні конструкції (для можливості

вибору на основі деякої логіки яким саме має бути представлення даних), циклічні конструкції (для створення повторюваних представлень, наприклад таблиць чи списків).

Також у представленнях було використано JavaScript-бібліотеку JQuerry та технологію AJAX для:

- створення спливаючих вікон що дозволяє не переходити на нову сторінку при виконанні простих дій такі як додавання нової ролі користувачу, створення нового повідомлення користувачам, створення звернення до комісії з поселення тощо;
- Створення динамічних фільтрів та меню що дозволяють підвантажувати новий вміст у елемент сторінки не перезавантажуючи, що як зменшує навантаження на сервер так і покращує досвід користування веб-застосунком (рисунок 3.6).

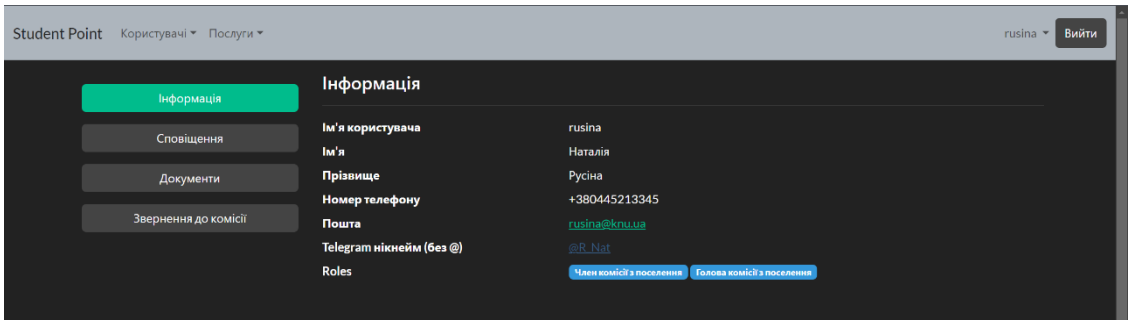

Рисунок 3.6 – Фрагмент особистої сторінки користувача

Для зручного зберігання даних про поля форм та відповідей на них використано серіалізацію в JSON, так як ці дані можуть мати різну форму. Враховуючи перспективи розширення можливостей форм використано наступний формат для опису полів форми (таблиця 2).

Таблиця 2 – Опис властивостей поля форми

Назва	Тип	Можливі значення
Name	string	*
Type	enum	Int, String
Order	int	1, 2, ...

А для опису відповіді користувача використано формат з відповідними полями та значеннями (таблиця 3).

Таблиця 3 – Опис властивостей відповіді на форму

Назва	Тип
FieldName	string
Value	string

Такий формат дозволить записувати відповіді користувачів у таблицю бази даних.

РОЗДІЛ 4. ІНСТРУКЦІЇ З КОРИСТУВАННЯ

Інструкція користувача

При завантаженні веб-застосунку на початковій сторінці представлено загальну інформацію про веб-застосунок (рисунок 4.1).

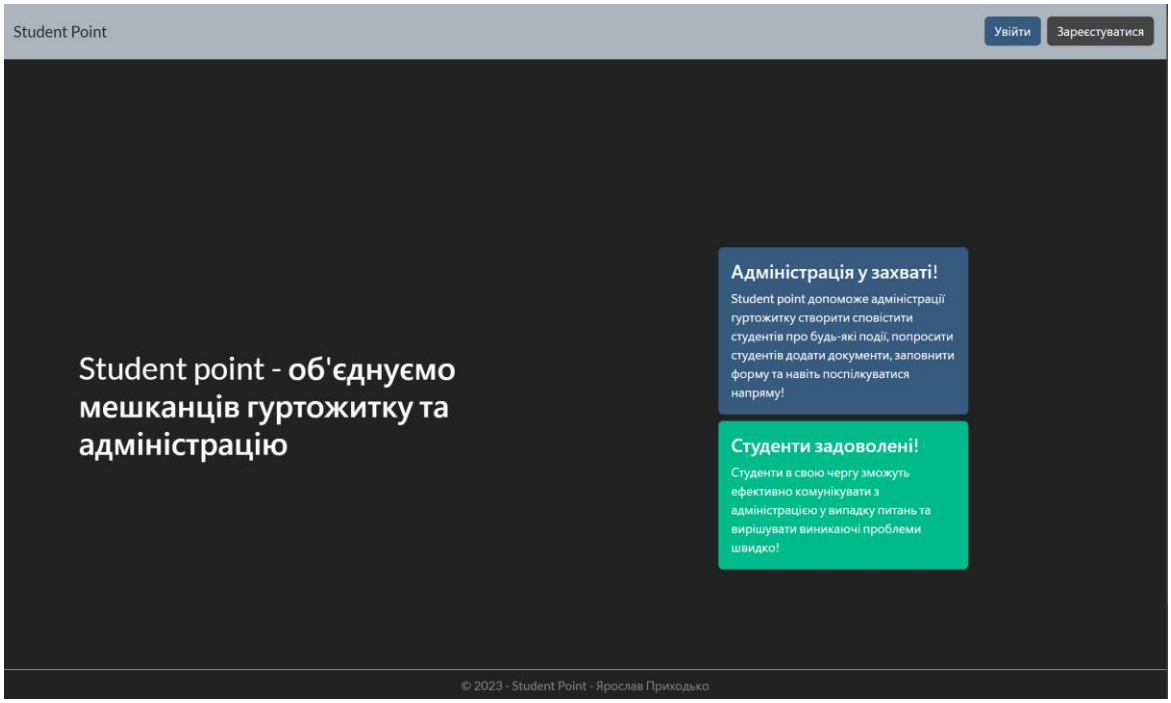

Рисунок 4.1 – Фрагмент початкової сторінки

У верхній частині веб-застосунку незалежно від поточної сторінки відображається панель що у разі якщо користувач не авторизований містить логотип веб-застосунку (зліва), а також посилання на сторінки реєстрації («Зареєструватися») та авторизації («Увійти») (справа). Якщо користувач вже авторизувався то на панелі також зліва присутні розділи («Користувачі», «Послуги») з випадаючими списками посилань на інші сторінки, а також зліва замість посилань, що відображаються неавторизованому користувачу наявні посилання на особисту сторінку користувача та кнопка виходу (рисунок 4.2).

Рисунок 4.2 – Фрагмент вигляду верхньої панелі авторизованого користувача

Користувач має можливість скористатися інформацією про членів комісії з поселення для додаткового зв’язку з ними (рисунок 4.3).

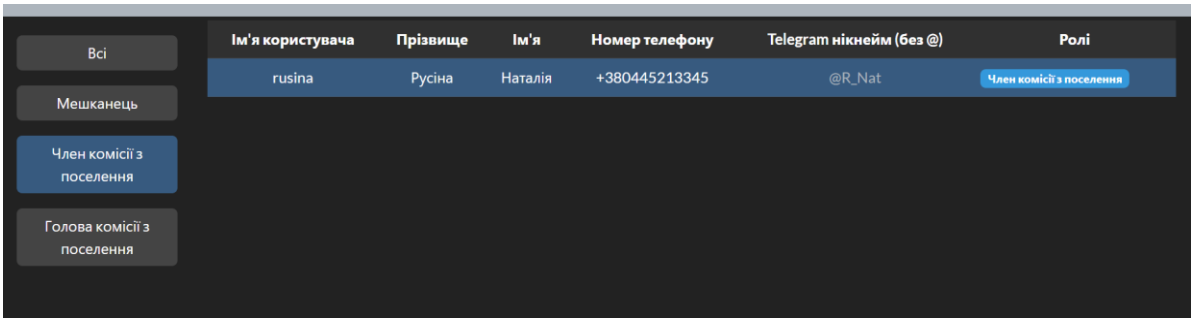

Рисунок 4.3 – Фрагмент інформації про членів комісії з поселення

На особистій сторінці користувача доступна інформація про себе, сповіщення, що надійшли користувачеві, документи користувача, а також інформація щодо актуального статусу та чати з приводу звернень до комісії з поселення (рисунок 4.4).

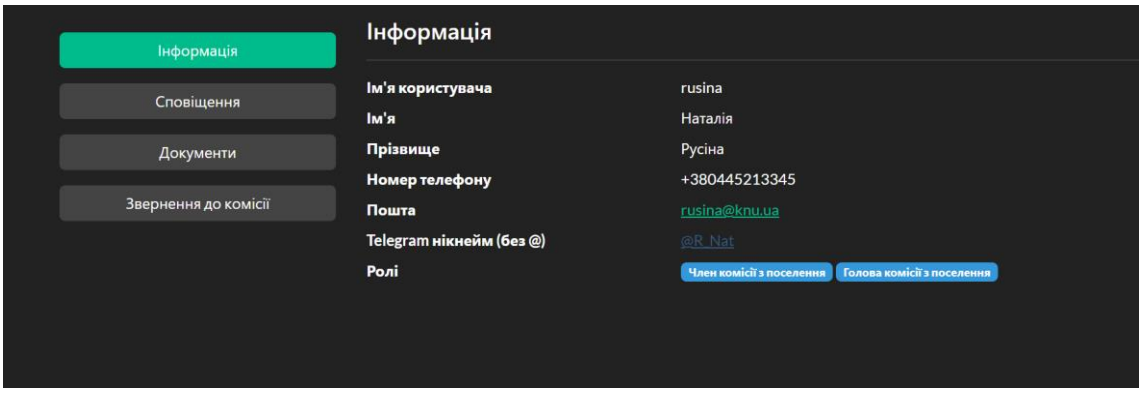

Рисунок 4.4 – Особиста сторінка користувача, інформація

У вкладці сповіщень розміщені всі сповіщення, що надійшли користувачу та не були ним прочитані (рисунок 4.5). Для сповіщень процес переведення з стану «непрочитане» у стан «прочитане» відбувається користувачем самостійно за допомогою кнопки, що дозволяє йому самому керувати цими сповіщеннями, а також сповіщення може містити кнопку з посиланням яке заливив член комісії з поселення під час створення сповіщення. Варто зауважити що стан чи прочитане сповіщення можуть також переглядати члени комісії з поселення.

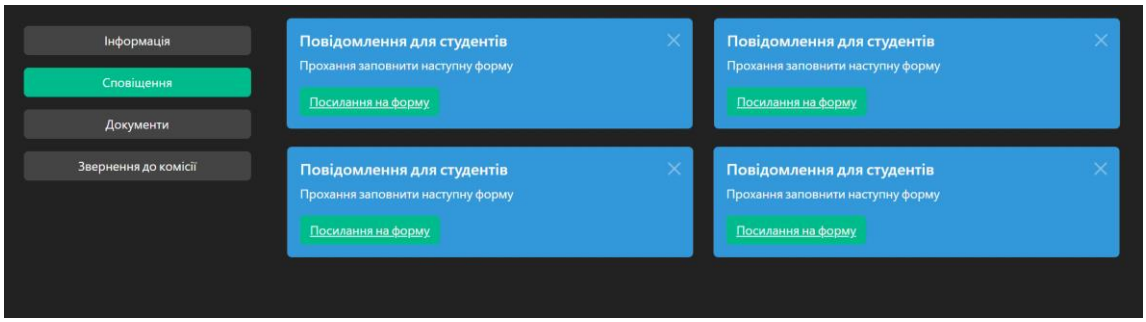

Рисунок 4.5 – Вкладка сповіщень користувачу

У вкладці з документами, користувач може додати документи до призначених тегів, а також переглянути документи, що вже були ним прикріплені (рисунок 4.6).

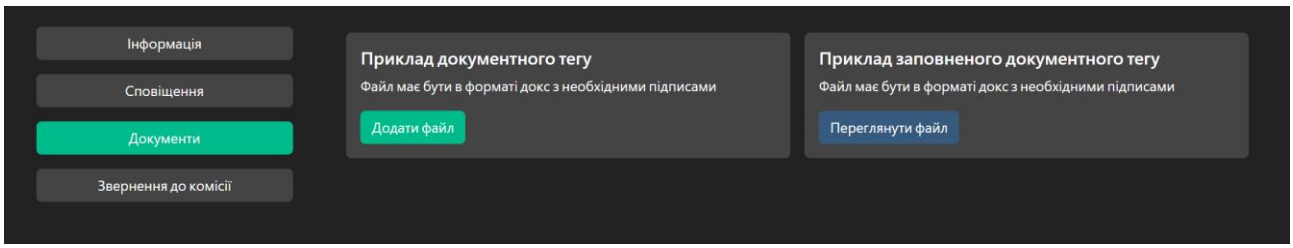

Рисунок 4.6 – Вкладка документів користувача

У вкладці звернень до комісії з поселення користувач може переглянути відкриті та закриті звернення, а також переглянути чати з приводу кожного з звернень (рисунок 4.7). Окрім перегляду користувач тут також може створити нове звернення.

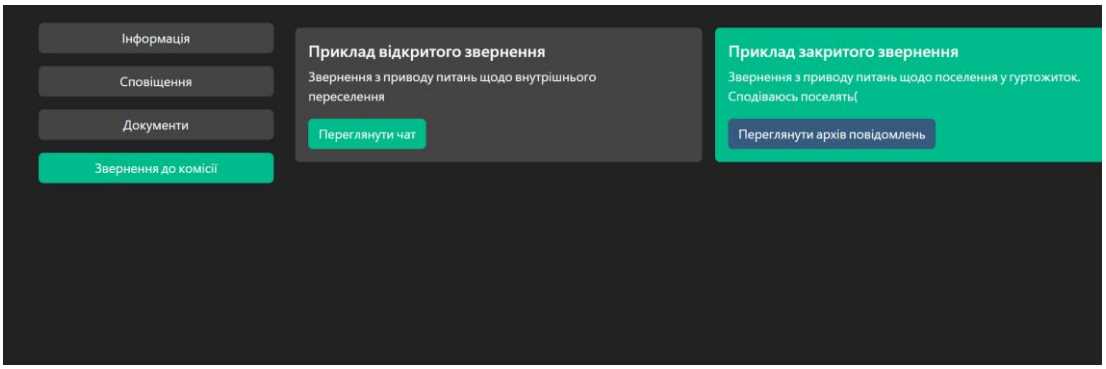

Рисунок 4.7 – Вкладка зі зверненнями до комісії

У розділі послуги присутній список сторінок послуг, що доступні користувачу: «документи» та «форми». На сторінці «документи» користувачу відображається таблиця з усіма доступними йому документними тегами. Якщо тег вже має документ, то документ можна переглянути, інакше можна

прикріпити новий документ. Сторінка «форми» аналогічно дозволяє отримати актуальний список всіх доступних форм у вигляді таблиці та у разі якщо форма ще не заповнена – заповнити її, а якщо вже заповнена, то переглянути та редагувати відповідь якщо це налаштовано у самій формі. При заповненні форми користувач отримує до заповнення поля відповідних типів які валідуються, що запобігає ситуації під час якої відбувається некоректне введення інформації в поле.

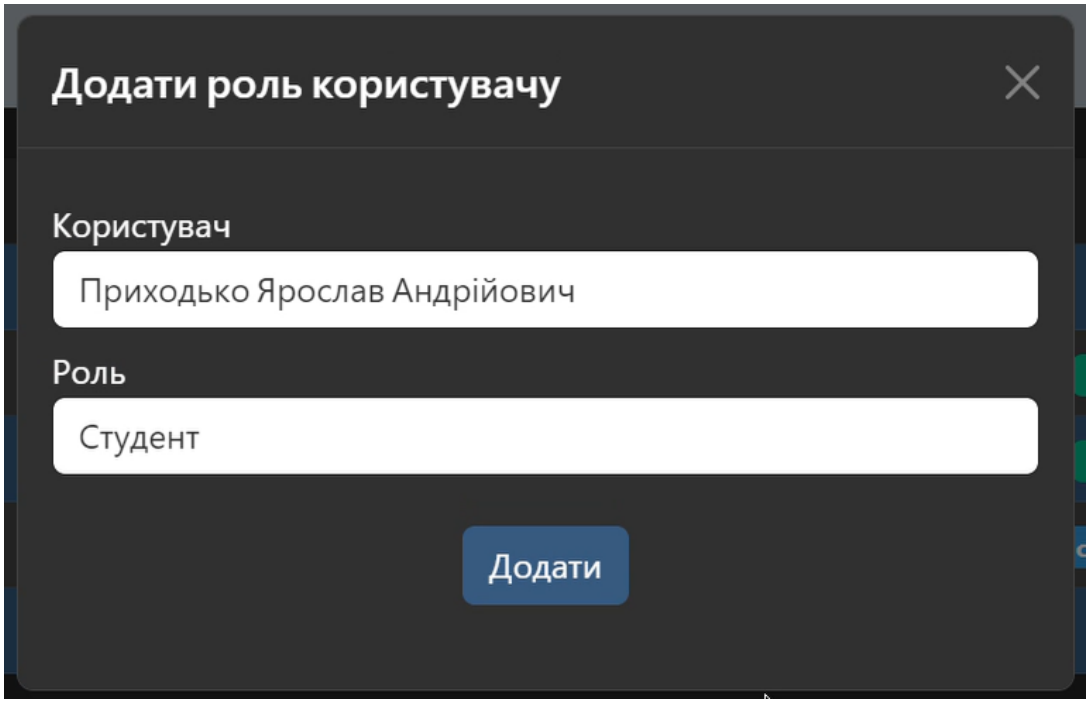

Рисунок 4.8 – Приклад спливаючого вікна, додавання ролі користувачу

Загалом весь користувацький інтерфейс побудовано за шаблоном, у якому зліва знаходиться колонка вибору, а справа головне вмістове вікно. Окрім цього у веб-застосунку скрізь де це можливо для введення інформації використовуються спливаючі вікна (modal window) (рисунок 4.8), що дозволяє покращити користувацький досвід, так як надає можливість не переходити на нову сторінку щоразу коли є потреба відредагувати якусь інформацію або відкрити чат з повідомленнями.

Інструкція члена комісії з поселення

Окрім функціоналу, що доступний всім користувачам членам комісії з поселення також доступні наступні функції з управління користувачами та

ролями, що надає гнучкість у створенні форм, документних тегів та для конкретних користувачі. Для цього на сторінці зі списком користувачів є додаткові можливості представлені у вигляді кнопок, кожна з яких створює спливаюче вікно у якому можна редагувати та додавати необхідну інформацію. Тут можна переглянути всіх користувачів за ролями, додати роль користувачу, забрати роль у користувача, видалити користувача, створити нову роль, редагувати або видалити існуючу роль (рисунок 4.9).

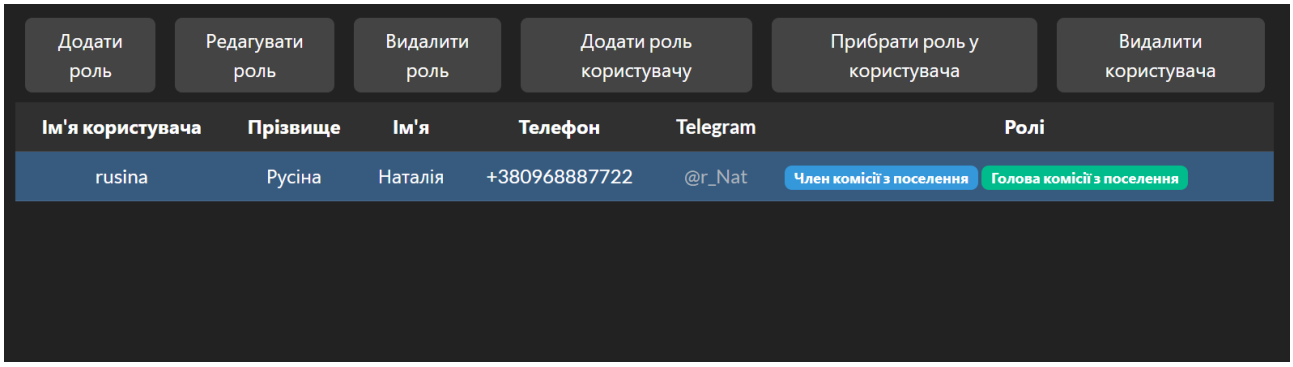

Рисунок 4.9 – Набір кнопок для контролю користувачів та ролей

На сторінці зі зверненнями користувачів члени комісії з поселення можуть відповідати на ці звернення шляхом створення чату та спілкування через нього з користувачем, що звернувся. В будь-який момент часу коли член комісії вирішить що проблема, що викладена у зверненні вирішена може закрити звернення.

На сторінці сповіщень член комісії з поселення може створити повідомлення, що у вигляді сповіщення буде надіслано всім користувачам певної ролі, після цього стає доступною інформація про користувачів, що позначили це сповіщення як прочитане. Опціонально до сповіщення може бути додане посилання та назва посилання що створить кнопку під час перегляду користувачем, за допомогою якої він може виконати додаткову дію, необхідну комісії з поселення (рисунок 4.10).

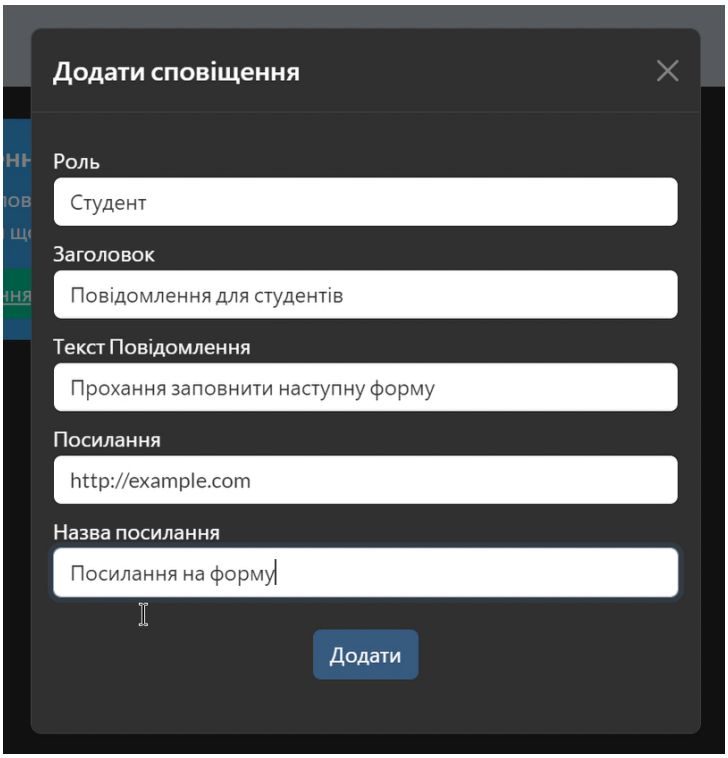

Рисунок 4.10 – Вікно створення сповіщення для певної ролі

На сторінках форм та документних тегів член комісії з поселення має можливість створювати відповідно форми та документні теги для певних ролей. У випадку форм необхідно вказати загальну інформацію про форму, таку як назва, опис тощо, а також можна додати необхідну кількість полів, кожне поле повинне мати назву та тип. Для документних тегів відбувається схожий процес. Після заповнення обох видів запитів на збір інформації від користувачів є можливість подивитися відповідні прикріплені документи та результати заповненя форм для кожного з користувачів (рисунок 4.11), а також експортувати їх у форматі .xlsx або .csv.

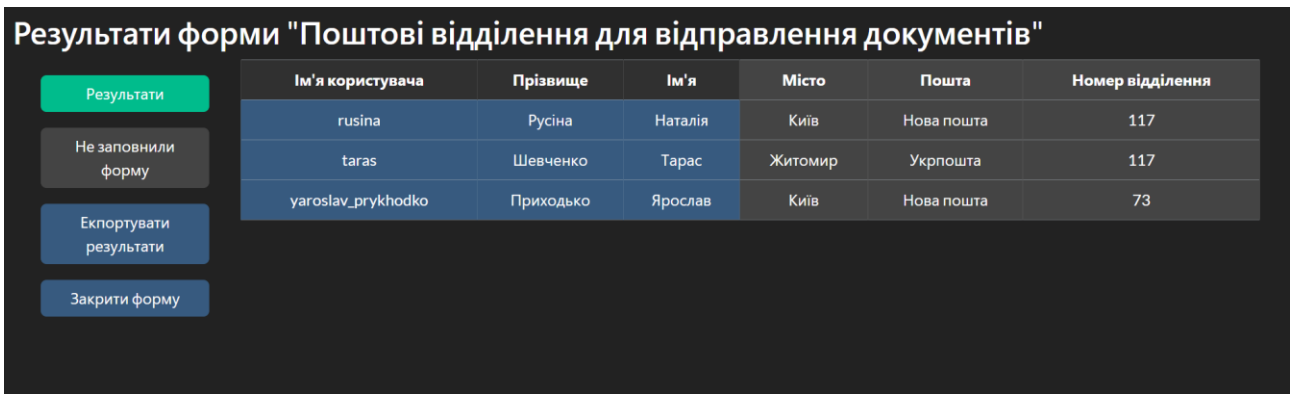

Рисунок 4.11 – Приклад результатів заповнення форми

У вкладці «не заповнили форму» можна переглянути користувачів, що ще не заповнили форму, а також надіслати їм нагадування про це.

Інструкція голови комісії з поселення

Роль голови комісії з поселення розширяє функціонал члена комісії з поселення можливостями додати нового члена комісії з поселення, або забрати цю роль у користувача що її має. Крім цього голова комісії з поселення може передати цю роль іншому користувачу (при цьому користувач, що мав цю роль раніше, позбувається її).

ВИСНОВКИ

В ході виконання кваліфікаційної роботи, відповідно до поставленої мети, були виконані завдання:

- оглянуто існуючі аналоги на ринку в Україні;
- проаналізовано роботу комісії з поселення з точки зору автоматизації комунікації з мешканцями гуртожитку;
- сформулювало функціональні вимоги до веб-застосунку та розроблено модель даних;
- реалізовано веб-застосунок.

Для виконання роботи було обрано стек технологій, що складається з

- мов програмування C# та JavaScript, що разом зі своїми фреймворками та бібліотеками ASP.NET Core MVC, jQuery, AJAX надали широкий спектр інструментів для ефективної веб-розробки;
- контенеризаційного рушія Docker, за допомогою якого була можливість запускати базу даних Postgres та систему з її адміністрування pgAdmin у відтворюваному середовищі без необхідності встановлення цього програмного забезпечення в локальну систему.

Створений під час виконання роботи веб-застосунок може бути введений в експлуатацію комісіями з поселення студентів у КНУ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Visual Studio 2022 [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://visualstudio.microsoft.com/.
2. What is Visual Studio? | Microsoft Learn [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022.
3. HTML: Hyper Text Markup Language [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTML.
4. .NET | Build. Test. Deploy. [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/.
5. Docker Docs: How to build, share, and run applications | Docker Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/.
6. Кабінет мешканця – Режим доступу до ресурсу: https://studmisto.knu.ua/vkhid
7. «SMART Університет» - ЗУНУ [Електронний ресурс] – Режим доступу до ресурсу: https://smart.wunu.edu.ua/
8. Mouat A. Using Docker: Developing and Deploying Software with Containers / Adrian Mouat., 2016. – 354 с.
9. Riggs S. PostgreSQL Administration Cookbook, 9.5/9.6 Edition / S. Riggs, G. Ciolli, G. Bartolini., 2017. – 556 с.
10. Laster B. Professional Git / Brent Laster., 2016. – 480 с.
11. Strauss D. Getting Started with Visual Studio 2022: Learning and Implementing New Features 2nd ed. Edition / Dirk Strauss., 2022. – 331 с.
12. Price M. C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7, 7th Edition 7th ed. Edition / Mark J. Price., 2022. – 818 с.

13. Robson E. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages 2nd Edition / E. Robson, E. Freeman., 2012. – 762 с.
14. Lett J. Bootstrap Reference Guide: Bootstrap 4 and 3 Cheat Sheets Collection (Bootstrap 4 Tutorial) / Jacob Lett., 2018. – 102 с.
15. Bootstrap · The most popular HTML, CSS, and JS library in the world. [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/.
16. Riordan R. Head First Ajax / Rebecca Riordan., 2008. – 497 с.

ДОДАТКИ

ДОДАТОК А. Діаграма сутностей