Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ВИБОРУ ТА БРОНЮВАННЯ НОМЕРІВ У ГОТЕЛЯХ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1
Проценко П.Р.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Проценку Павлу Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для вибору та бронювання номерів у готелях

затверджена наказом університету від 20.05.2024 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 01 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, SQL Server Management Studio, програмне забезпечення Visual Studio.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд систем для онлайн бронювання готелей.
2. Проєктування архітектури вебзастосунка.
3. Реалізація функціонала вебзастосунка.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми бронювання житла, постановка задачі, діаграма прецедентів, діаграма бази даних, схема ASP.NET MVC, класи моделей, тестові зображення роботи вебзастосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-13.04.24	
3	Аналіз літератури з досліджуваної проблеми	14.04.24-19.04.24	
4	Аналіз технічних засобів	19.04.24-24.04.24	
5	Розробка вебзастосунку	26.04.24-18.05.24	
6	Програмна реалізація	19.05.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	31.05.24	
9	Рецензування	01.06.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	10.06.24	
12	Попередній захист кваліфікаційної роботи	10.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _______
(підпис)

Керівник роботи _____ доц. Руденко Д.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 58 с., 15 табл., 31 рис., 22 джерела.

БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, MS SQL, ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРФЕЙС ДОСТУПУ, ASP.NET, ТУРИЗМ

Об’єктом роботи є вебзастосунок бронювання готелей.

Метою роботи є розробка вебзастосунка для бронювання готелей з врахуванням особливостей поточних потреб.

Виконано аналіз ринку аналогічних продуктів. Покращено навички роботи з UML-діаграмами. Проведено дослідження технологій мови програмування C#, які доцільно використати для реалізації проєкту. Створено базу даних.

У результаті роботи створено базу даних та реалізовано вебзастосунок для бронювання житла.

DATABASE, DATABASE MANAGEMENT SYSTEM, MSSQL, INFORMATION SYSTEM, ACCESS INTERFACE, ASP.NET, TOURISM

The object of the work is a hotel booking web application.

The aim of the work is to develop a web application for hotel booking taking into account the features of current needs.

Market analysis of similar products has been conducted. Skills in working with UML diagrams have been improved. Research on C# programming language technologies, which are appropriate for project implementation, has been carried out. A database has been created.

As a result of the work, a database has been created and a web application for accommodation booking has been implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

.NET – бренд набору технологій і фреймворків, розроблених Microsoft, які базуються на концепції спільної мовної інфраструктури

ASP.NET – Active Server Pages .NET. Це фреймворк для розробки вебзастосунків, розроблений компанією Microsoft

CSS – Cascading Style Sheets. Це мова стилів, яка використовується для оформлення та візуального оформлення вебсторінок й вебзастосунків

HTML – HyperText Markup Language. Це основна мова розмітки, яка використовується для створення та структурування вебсторінок

LINQ – Language Integrated Query. Це функціональна частина платформи .NET, яка дозволяє виконувати структуровані та зручні запити до даних у мовах програмування, що підтримуються платформою .NET, таких як C# або VB.NET

MVC – Model-View-Controller. Це патерн архітектури програмного забезпечення, який використовується для розробки вебзастосунків

SQL – Structured Query Language. Це стандартна мова запитів, яка використовується для взаємодії з реляційними базами даних

XML – «eXtensible Markup Language». Це мова розмітки, яка використовується для представлення та обміну даними у структурованому форматі

UML – Unified Modeling Language. Це стандартизована мова моделювання, яка використовується для візуального представлення дизайну програмного забезпечення

СКБД – система керування базами даних

ВСТУП

В сучасному світі інформаційні технології відіграють важливу роль у багатьох сферах діяльності, включаючи туризм. Використання онлайн-систем бронювання стає все більш популярним, адже вони пропонують зручний та швидкий спосіб пошуку та бронювання житла. Проте, наявні системи не завжди враховують специфічні потреби мандрівників, наприклад, наявність бомбосховища чи генератора, а також не надають можливості одразу замовляти додаткові послуги.

Метою даної роботи є розробка вебзастосунка для онлайн бронювання житла, який буде враховувати додаткові властивості, такі як бомбосховище, генератор тощо, а також надавати можливість одразу замовляти додаткові послуги.

Об’єктом дослідження є вебзастосунки для онлайн бронювання житла.

Предметом дослідження є функціональні можливості вебзастосунків для онлайн бронювання житла з урахуванням додаткових властивостей та можливості замовлення додаткових послуг.

Для досягнення мети роботи використовувались наступні методи дослідження:

- аналіз літературних джерел;
- вивчення наявних вебзастосунків для онлайн бронювання житла;
- проєктування та розробка вебзастосунка;
- тестування вебзастосунка.

Актуальність роботи полягає у висхідній ролі інформаційних технологій у повсякденному житті та у різних сферах діяльності, включаючи туризм. Сучасні онлайн-системи бронювання житла стають все більш популярними, оскільки пропонують зручний та швидкий спосіб пошуку та бронювання житла. Однак, наявні системи не завжди задовольняють специфічні потреби мандрівників, такі як наявність бомбосховища або генератора, і не дозволяють

одразу замовляти додаткові послуги. Це створює необхідність у розробці нових, більш функціональних вебзастосунків, які враховували б ці вимоги. Розробка такого вебзастосунка сприятиме підвищенню зручності та безпеки мандрівників, що робить цю роботу надзвичайно актуальною в сучасних умовах.

1 ОГЛЯД НАЯВНИХ СИСТЕМ

1.1 Опис предметної області

У сучасному світі використання інформаційних систем стало невід’ємною частиною життя у багатьох сферах діяльності. Вони використовуються в бізнесі для автоматизації процесів, в освіті для покращення навчання та доступу до знань, в медицині для управління медичною інформацією та діагностики, у сфері розваг для створення ігор та розважальних застосунків, а також в повсякденному житті для спрощення комунікації та зручного доступу до різноманітної інформації. Популярність цих систем пояснюється їхньою здатністю полегшувати рутинні завдання, підвищувати ефективність та забезпечувати швидкий доступ до потрібної інформації, що робить їх необхідними в сучасному цифровому світі.

Галузь туризму визнається однією з найбільш розвинутих у сучасній економіці. Це пов’язано з величезним обсягом різноманітної інформації, яка вимагає систематичного збору, відбору, зберігання, обробки та передавання. Однак, ключовим фактором для просування будь-якого туристичного підприємства є використання сучасних інформаційних систем і технологій. Вони є важливим інструментом у створенні, пропозиції та рекламі туристичного продукту. У сучасних умовах, ці технології стають вирішальним фактором у підвищенні конкурентоспроможності туристичних підприємств, як в Україні, так і за кордоном [1].

Грамотне використання інформаційних технологій допомагає підприємствам у сфері туризму випередити конкурентів і залучити клієнтів. Розглянемо етапи, з якими стикаються туристичні агенти при підборі туру. Першим етапом є вибір країни та місця відпочинку, що базується на рекомендаціях друзів, родичів, а також аналізі інформації з різних джерел, включаючи фільми та рекламу. Інформаційні технології допомагають агентам пропонувати кілька варіантів. Другий етап – вибір місця розміщення, де також

інформаційні технології надають детальну інформацію про готелі, їх розташування та послуги. На третьому етапі – авіаперевезення і трансфер, інформаційні технології забезпечують готовність до прибуття в нове місце, включаючи договори з авіакомпаніями та надання гідів для орієнтації та надання інформації на місці [1].

Отже, використання інформаційних технологій у сфері туризму сприяє випередженню конкурентів та привертанню клієнтів, що своєю чергою виконує основну задачу будь-якого бізнесу – збільшення доходу. Технології допомагають туристичним агентствам у кожному етапі, особливо важливою є їхня роль у підготовці до прибуття в нове місце, забезпечуючи готовність клієнтів та забезпечуючи їх зручним перебуванням.

Крім онлайн-бронювання, можна також використовувати та інші варіанти (таблиця 1.1.), але кризи останніх років (карантин COVID-19, повномасштабне вторгнення Росії в Україну) продемонстрували ненадійність такої комунікації.

Таблиця 1.1 – Деякі пособи бронювання без використання онлайн-платформ.

Спосіб	**Переваги**	**Недоліки**
1	**2**	**3**
Телефонний дзвінок	+Персоналізований підхід: можливість обговорити усі деталі прямо з представником готелю. +Гнучкість: можливість узгодити додаткові запити або уточнити умови проживання.	-Часовий затяг: потребує часу на здійснення дзвінка та очікування оператора. -Обмежена доступність: може бути складно отримати доступ до бажаного готелю у випадку високої зайнятості ліній. -Додаткові витрати: якщо готель знаходиться в іншій країні, доведеться значно переплатити за дзвінок.

Продовження таблиці 1.1

1	2	3
Агентства подорожей	+Експертна допомога: агенти можуть надати поради та рекомендації щодо вибору готелю. +Узгодження пакетів: можливість узгодити готель разом з іншими послугами (наприклад, авіаквитки).	-Додаткові витрати: агенти часто беруть комісійну плату за свої послуги. -Обмежений вибір: агентства можуть пропонувати обмежений вибір готелів порівняно з онлайн-платформами.
Спілкування в готелі	+Гнучкість: можливість отримати інформацію безпосередньо від представників готелю. +Можливість переговорів: нагода узгодити умови проживання безпосередньо на місці. +Наочність: перед бронюванням є можливість оглянути наживо, за що саме сплачуватимете.	-Ризик недоступності: можливість, що конкретний готель буде повністю заброньований. -Додаткові витрати: потребує фізичної присутності та витрат на мандрівку до готелю. -Додаткові ризики: залишитись «на вулиці» через перевантаженість готелей в сезон або ж проживати в незручному місці (наприклад, далеко від моря).

Також варто зауважити, що кожен рік частка людей, що використовують Інтернет для замовлення різноманітних послуг збільшується [2], що потребує відповідної реакції будь-якого бізнесу. А із початком повномасштабного вторгнення кількість переміщень населення України через різні причини (тимчасово переміщенні особи, волонтери, військові тощо) зросла, натомість в готелях та інших бізнесів, які надають свої послуги, змінились умови, на що впливає безліч чинників.

1.2 Огляд систем для онлайн бронювання готелей

1.2.1 Огляд системи Booking.com

У 1996 році Білл Гейтс створив сервіс для бронювання готелів під назвою Expedia.com. Він давав змогу попередньо зарезервувати номер в онлайн-режимі, без додаткових підтверджень телефоном. Хоча сайт за короткий час став популярним серед мандрівників, він усе ж мав один значний недолік: користувач не міг подивитися фотографії готелю, прочитати відгуки та опис номерів. Тобто, бронюючи житло, людина ніколи не знала, у яких умовах перебуватиме. Та оскільки альтернатив цьому сервісу в онлайні не було – Expedia.com продовжував успішно функціонувати. Цього ж року випускник коледжу Герт-Ян Бруїнсма вирішує поїхати на відпочинок до Будапешта, та стикається з проблемою вибору готелю. Він не розуміє, як має забронювати номер, побачити який немає можливості. Тоді чоловіку спала на думку ідея створити такий самий сервіс, але з фотографіями, відгуками та описом деталей, які зазвичай цікавлять туристів. Результатом реалізації цієї ідеї став проєкт Bookings.nl [3].

У 1999 році двоюрідні брати Енді Філіппс та Адріан Крітчлоу запускають сайт для бронювання готелів під назвою Active Hotels. Він працює за тією ж системою, що й проєкт Бруїнсма. Засновники почали співпрацювати з готелями, просили в них фото та опис номерів, завантажували на ресурс і пропонували користувачам зручно бронювати житло без передоплати.

Хоча Bookings.nl та Active Hotels конкурували, ЗМІ не спостерігали відкритого конфлікту між ними. Власники компаній були повністю заглиблені в розвиток своїх проєктів, адже попри популярність ресурсів, їм не вдавалося вийти на високий дохід. Тож у 2004 році американська компанія Priceline придбала Active Hotels, у 2005-му – Bookings.nl, а в 2006-му об’єднала їх в один усім відомий нині проєкт – Booking.com (рисунок 1.1).

Рисунок 1.1 – Історія іменування Booking.com

Головний та найдієвіший інструмент, який використовує маркетингова команда Booking, – це поширення емоцій від подорожей через відеоролики. Цей формат дає можливість продемонструвати красиві локації та передати максимум почуттів глядачам, що викликає бажання почати планувати подорож. Загалом, акцент саме на стан та емоції людини, які вона отримає за умови придбання певної послуги є сучасним трендом, адже це реальна можливість отримати новий досвід або просто задоволення від життя.

Одна з таких рекламних кампаній має назву «Життя, повне відкриттів» і містить шість відео тривалістю від 15 до 35 секунд. Ролики зробили невеликими навмисно, щоб продемонструвати глядачам якомога більше яскравих емоцій за короткий час. У такий спосіб компанія проводить паралель з подорожами: декілька днів і нових вражень змінять ваше світовідчуття. Герої роликів катаються красивими місцями, наважуються стрибнути в холодну воду, демонструють наявність пристосувань для людей з обмеженими можливостями.

Booking є лідером серед сайтів з оренди житла у подорожах завдяки своїй адаптивності до змін, наявності вибору для різних сегментів аудиторії та вмінню віртуально продемонструвати емоції, які отримає мандрівник. Команда маркетологів не прагне до регулярної активності в соціальних мережах, але знає, що їхня аудиторія лояльна, адже всю потрібну інформацію легко може знайти на офіційному сайті.

На початку своєї діяльності власники об'єднаних компаній орієнтувалися на власні потреби та згідно з ними поліпшували роботу сайту (рисунок 1.2). Зараз працюють за схожим принципом, оскільки всі члени команди є активними амбасадорами компанії. Завдяки цьому сайт часто публікує цікаві та зручні маршрути, щоб бути максимально корисними для свого споживача.

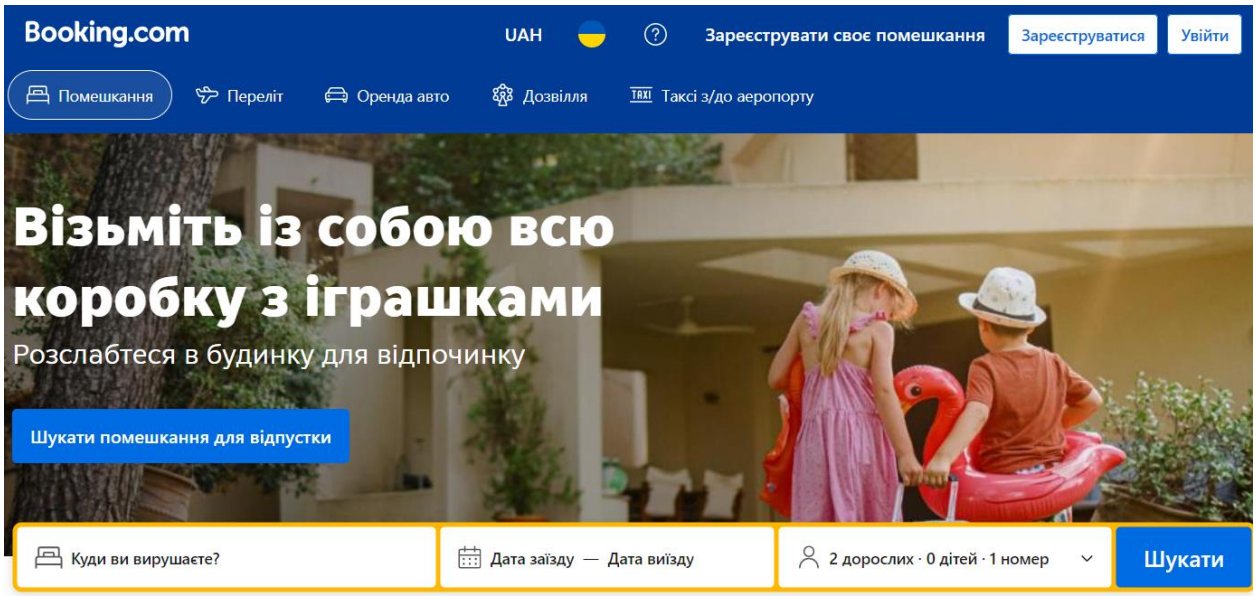

Рисунок 1.2 – Головна сторінка Booking.com

Проте варто зауважити, що після початку повномасштабного вторгнення з'явилось багато викликів, які є неактуальними для більшості інших країн – чи є бомбосховище, чи можливо військовим/ВПО/волонтерам переночувати ніч безоплатно, чи є генератор тощо. Наразі відповідні фільтри у Booking.com

відсутні, що створює певні перешкоди й потребує додаткової комунікації, що не завжди є зручним.

1.2.2 Огляд системи Airbnb

Ідея виникла після того, як двоє із засновників почали орендувати надувні матраци у своєму будинку в Сан-Франциско для відвідувачів конференції. Звідси й оригінальна назва Airbed & Breakfast. [4]

У 2009 році було введено назву Airbnb, і її пропозиції виросли за рамки надувних матраців, включивши додаткові кімнати, квартири, цілі будинки тощо. Місця, в яких він діяв, також зростали. У 2011 році Airbnb відкрила офіс у Німеччині, а в 2013 році – європейську штаб-квартиру в Дубліні, Ірландія. Основним місцем розташування компанії залишається Сан-Франциско.

Окрім США та Європи, компанія має представництва в Австралії, Азії, на Кубі, а також в інших країнах (загалом понад 220 країн і регіонів). Він також розширив свої туристичні пропозиції, включивши програми місцевих заходів під назвою Experiences.

У 2020 році Airbnb запустив програму Frontline Stays для розміщення служб першої допомоги під час пандемії COVID. Також у 2020 році компанія стала публічною. Її звичайні акції торгуються на біржі Nasdaq під символом ABNB.

Airbnb – це онлайн-ринок, який об’єднує людей, які хочуть здати свою нерухомість в оренду, з людьми, які шукають житло, як правило, для короткострокового проживання.

Airbnb пропонує господарям відносно простий спосіб отримати певний дохід від своєї власності.

Гості часто виявляють, що оренда житла через Airbnb дешевша та приємніша за готелі.

Airbnb отримує більшу частину свого доходу, стягуючи комісію як з гостей, так і з господарів.

Таким чином Airbnb змінив індустрію подорожей. Останні статистичні дані показують, що зараз він працює в 191 різній країні, має понад 4 мільйони хостів Airbnb по всьому світу та понад 6 мільйонів активних оголошень на платформі (рисунок 1.3). Його розмір і поширення гарантують, що, незалежно від того, чи віддають перевагу гості приватність будинку для відпочинку, чи хочуть заощадити на витратах, орендуючи кімнату в чиємусь домі, Airbnb зможе задовольнити їхні потреби.

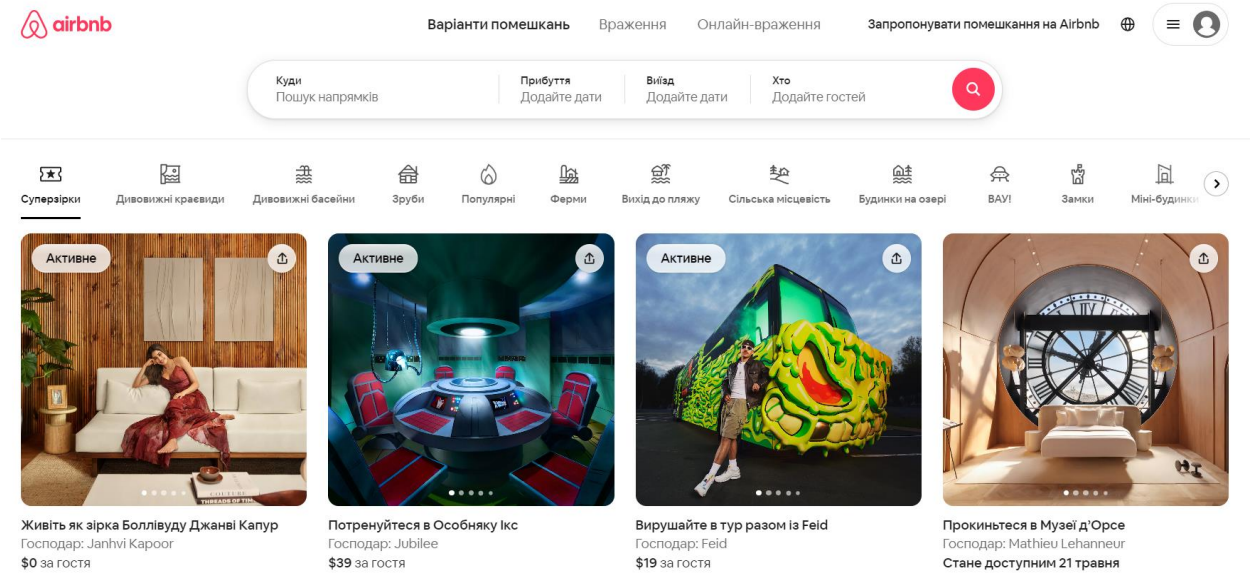

Рисунок 1.3 – Головна сторінка Airbnb

Переваги використання Airbnb:

– можливість знайти унікальне та недороге житло в бажаному місці. Airbnb пропонує широкий вибір варіантів, від квартир до будинків, які часто дешевші за готелі;

– можливість відчути місцеву автентичність та взаємодіяти з господарями. Це дає змогу краще пізнати культуру та традиції місця, де ви подорожуєте;

– гнучкість у виборі розміщення та тривалості перебування. На Airbnb можна знайти житло на будь-який термін, від однієї ночі до кількох місяців;

– можливість заробити для власників нерухомості. Здача житла через Airbnb дає змогу отримувати додатковий дохід.

Недоліки використання Airbnb:

– складнощі з пошуком житла, особливо для сімей з дітьми. Деякі власники можуть відмовляти в розміщенні сімей з дітьми;

– невизначеність щодо якості житла та послуг. Оскільки житло надається приватними особами, його стан та рівень сервісу можуть відрізнятися від очікувань;

– необхідність сплачувати додаткові збори, крім вартості оренди. Airbnb стягує комісію як з орендарів, так і з власників, до того ж чим довший період проживання, тим більша комісія, що іноді може нівелювати перевагу над бронюванням готелю;

– відсутність гарантій та юридичного захисту, як у готелях. У разі виникнення проблем вирішувати їх доводиться самостійно.

В цілому, всі ці системи дають можливість тільки для бронювання житла, без можливості комунікації протягом проживання та не враховують варіант замовлення додаткових послуг, наприклад, екскурсія може бути як додаткова опція в описі, але замовити її можна лише за додаткової комунікації.

1.3 Постановка завдання

Основне завдання: Створення системи для онлайн бронювання житла з врахуванням різних додаткових властивостей (бомбосховище, генератор тощо) та з можливістю одразу замовляти додаткові послуги.

Структура проєкту:

1. Створення та наповнення бази даних:

– розробка бази даних, яка буде містити різноманітні дані з предметної області, такі як характеристики житла, доступність додаткових властивостей, ціни та інші параметри;

– встановлення необхідних обмежень та валідація даних для забезпечення цілісності бази даних.

2. Розробка функціонала системи:

– створення інтерфейсу для комфортного використання системи кінцевим користувачем;

– розподіл функціонала системи за ролями, такими як адміністратор проєкту, адміністрація готелю та клієнти;

– реалізація можливості швидкого та зручного бронювання житла, враховуючи різні параметри і додаткові послуги;

– впровадження системи контролю доступу та захисту даних для забезпечення безпеки користувачів та конфіденційності інформації.

2 РОЗРОБКА ПРОЄКТУ

2.1 Проєктування архітектури вебзастосунка

2.1.1 Діаграма прецедентів

При розробці вебзастосунка бронювання готелів використання UML-діаграм може суттєво покращити процес розробки та експлуатації системи, забезпечуючи численні переваги як для розробників, так і для замовників. Ці діаграми слугують потужним інструментом для візуалізації архітектури системи, сприяючи чіткому розумінню її компонентів, функціоналу та взаємозв’язків.

UML-діаграми слугують універсальною мовою спілкування між членами команди розробників, мінімізуючи ризик непорозумінь та сприяючи ефективній співпраці (типи діаграм – рисунок 2.1). Завдяки стандартизованій нотації UML, різні учасники проєкту можуть легко обмінюватися інформацією про дизайн системи, що робить процес розробки більш організованим та результативним.

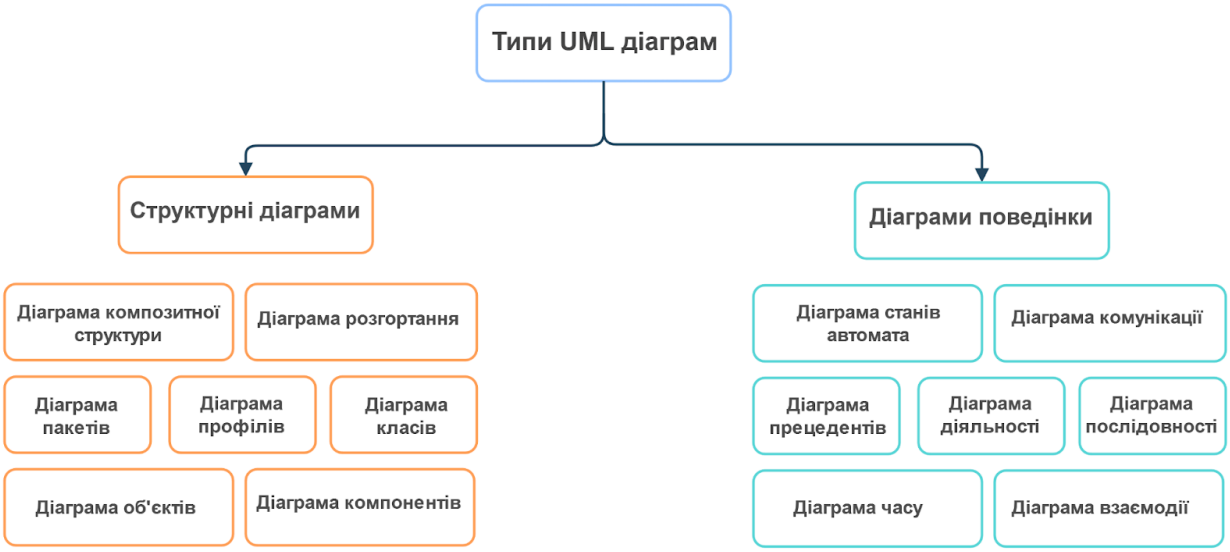

Рисунок 2.1 – Типи UML діаграм [11]

Використання UML-діаграм на ранніх стадіях розробки дозволяє проводити ретельний аналіз та планування архітектури системи. Це дає можливість виявити потенційні проблеми та недоліки на ранніх етапах, значно скорочуючи витрати часу та ресурсів на їх виправлення на пізніших стадіях.

Після завершення розробки UML-діаграми стають цінним ресурсом для підтримки та вдосконалення системи. Вони надають чітке уявлення про структуру та компоненти системи, що полегшує її модифікацію та адаптацію до нових вимог.

У контексті вебзастосунка бронювання готелів UML-діаграми стають особливо корисними завдяки складності та багатогранності таких систем. Вони допомагають розробникам керувати цією складністю, чітко розмежовуючи компоненти та їх взаємодії, що робить процес розробки більш організованим та ефективним.

Загальні переваги UML-діаграм:

– візуалізація: UML-діаграми надають чітке та лаконічне візуальне представлення архітектури системи, що робить її більш зрозумілою для всіх зацікавлених сторін, включаючи розробників, тестувальників, менеджерів проєктів та замовників [6];

– комунікація: UML-діаграми слугують спільною мовою для спілкування між членами команди, що допомагає уникнути непорозумінь та сприяє кращому співробітництву. Завдяки чіткій та стандартизованій нотації UML, різні учасники проєкту можуть легко обмінюватися інформацією про дизайн системи, що мінімізує ризик помилок та непорозумінь [7];

– аналіз та планування: UML-діаграми допомагають розробникам аналізувати та планувати архітектуру системи, виявляти потенційні проблеми та приймати обґрунтовані рішення щодо дизайну. Створюючи UML-діаграми на ранніх стадіях розробки, можна виявити та усунути потенційні проблеми, що економить час та ресурси на пізніших етапах [7];

– підтримка: UML-діаграми можуть слугувати цінним джерелом інформації під час підтримки та вдосконалення системи, надаючи чітке

уявлення про її структуру та компоненти. Добре документовані UML-діаграми полегшують розуміння та модифікацію наявної системи, що робить її більш гнучкою та адаптивною до нових вимог [6].

Переваги UML-діаграм для вебзастосунка бронювання готелів:

– складність системи: вебзастосунки бронювання готелів можуть бути складними системами з багатьма компонентами та взаємозв’язками. UML-діаграми допомагають розробникам керувати цією складністю та організовувати інформацію про систему [8]. Завдяки UML-діаграмам можна чітко розмежувати різні компоненти системи та їх взаємодії, що робить процес розробки більш організованим та ефективним;

– різноманітність функціонала: вебзастосунки бронювання готелів пропонують різноманітні функції, такі як пошук готелів, бронювання номерів, управління профілями користувачів, обробка платежів тощо. UML-діаграми допомагають розробникам моделювати ці функції та їх взаємозв’язки [9]. Завдяки UML-діаграмам можна чітко описати та візуалізувати поведінку кожної функції, що полегшує її реалізацію та тестування;

– інтеграція з сторонніми системами: вебзастосунки бронювання готелів часто інтегруються зі сторонніми системами, такими як платіжні шлюзи, системи управління готелями та системи відгуків. UML-діаграми допомагають розробникам моделювати ці інтеграції та забезпечити їхню безперебійну роботу [10]. Завдяки UML-діаграмам можна чітко визначити точки інтеграції та очікувані дані, що робить процес інтеграції більш ефективним та зменшує ризик помилок.

Діаграма прецедентів, також відома як діаграма варіантів використання, є одним з типів UML-діаграм, що використовується для моделювання та візуалізації функціональних вимог до системи. Вона показує взаємодію між акторами (користувачами або зовнішніми системами) та прецедентами (функціоналом) системи.

Основні елементи діаграми прецедентів [10]:

– актор – це сутність, що взаємодіє з системою, але не є її частиною. Це може бути людина, інша система або пристрій. Актори поділяються на первинних, які ініціюють взаємодію, та вторинних, які реагують на неї. І хоча актор може представляти клас користувачів, він не обов’язково є конкретним користувачем. Навіть один користувач може виконувати різні ролі в межах системи, зображуються у вигляді фігурок людей;

– сценарій використання (прецедент) визначає очікувану поведінку системи та відповідає на питання про те, що система робить. Він являє собою набір можливих функцій, дій або завдань і зображується у вигляді еліпса з назвою дії в ньому. Прецедент вказує, що має трапитись, але не відповідає на запитання, як це має статись;

– у діаграмах прецедентів використовуються різні типи відношень для опису зв’язків між акторами, прецедентами та іншими елементами діаграми. Ці відношення допомагають чітко визначити та зрозуміти функціональні можливості системи.

У діаграмі варіантів використання існує 4 типи зв’язків:

– асоціація (Association) – це зв’язок між актором та прецедентом, який позначається суцільною лінією без напису. Незалежно від типу зв’язку, кожен актор повинен бути пов’язаний принаймні з одним (а можливо з декількома) варіантами використання. Також кілька акторів можуть бути пов’язані з одним варіантом використання;

– розширення (Extend) демонструє додатковий функціонал або можливий, але не обов’язковий, варіант поведінки системи. Базовий прецедент має свою логіку сам по собі і може існувати незалежно від розширюючого прецеденту. Відношення розширення позначається пунктирною лінією зі звичайним вказівником на базовий прецедент і містить стереотип (напис) «extend». Розширюючий прецедент активується лише у випадку виконання певної умови;

– включення (Include) демонструє, що поведінка одного прецеденту включається як складова частина послідовності дій іншого прецеденту. Це показує, що базовий варіант використовується для виконання операції. На відміну від зв’язку розширення, включений варіант у зв’язку include є обов’язковим для базового. Використання включення допомагає уникнути дублювання однакових прецедентів і додає функціональність, яка не вказана в базовому. Відношення позначається пунктирною лінією зі стрілкою та стереотипом «include», що вказує на включений варіант;

– генералізація (Generalization), також відома як успадкування або узагальнення, описує відношення між батьківським і дочірнім прецедентами. У цьому відношенні генералізація позначається суцільною лінією з трикутним незафарбованим вказівником, що вказує на прецедент-предок.

Відповідно до зазначених вимог було створено діаграми прецедентів для адміністратора вебзастосунка (рисунок 2.2), адміністратора готелей (рисунок 2.3) та користувача (рисунок 2.4).

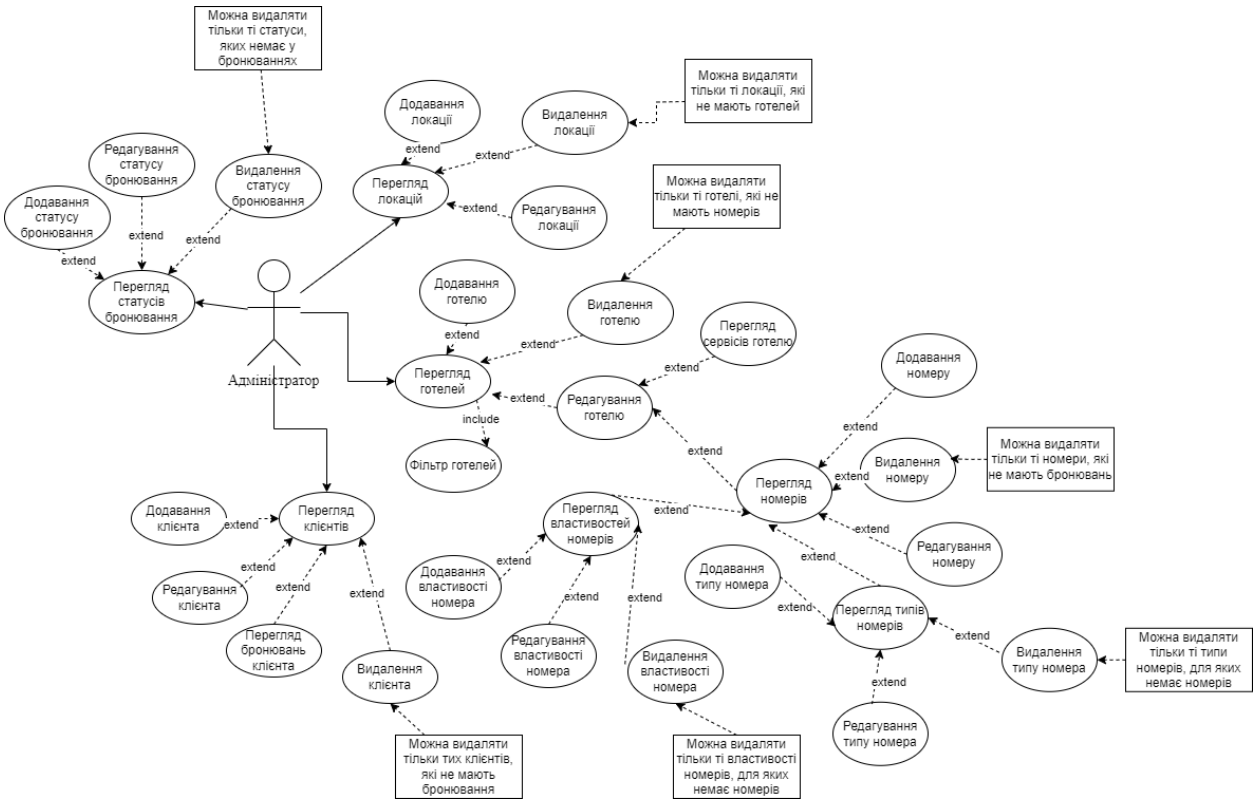

Рисунок 2.2 – Діаграма прецедентів адміністратора проєкту

Діаграми було створено із використанням draw.io – це безоплатне онлайн програмне забезпечення для створення діаграм, що дозволяє створювати потокові діаграми, діаграми процесів, діаграми UML, ER, мережеві діаграми тощо.

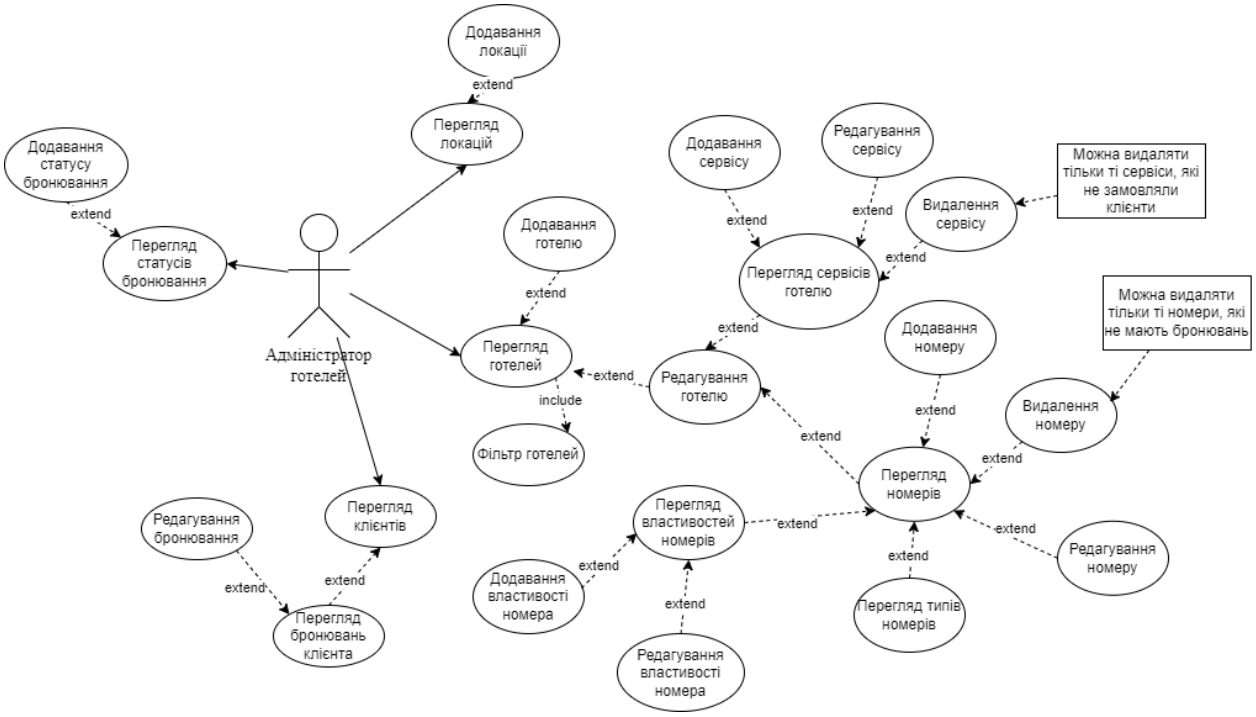

Рисунок 2.3 – Діаграма прецедентів адміністратора готелей

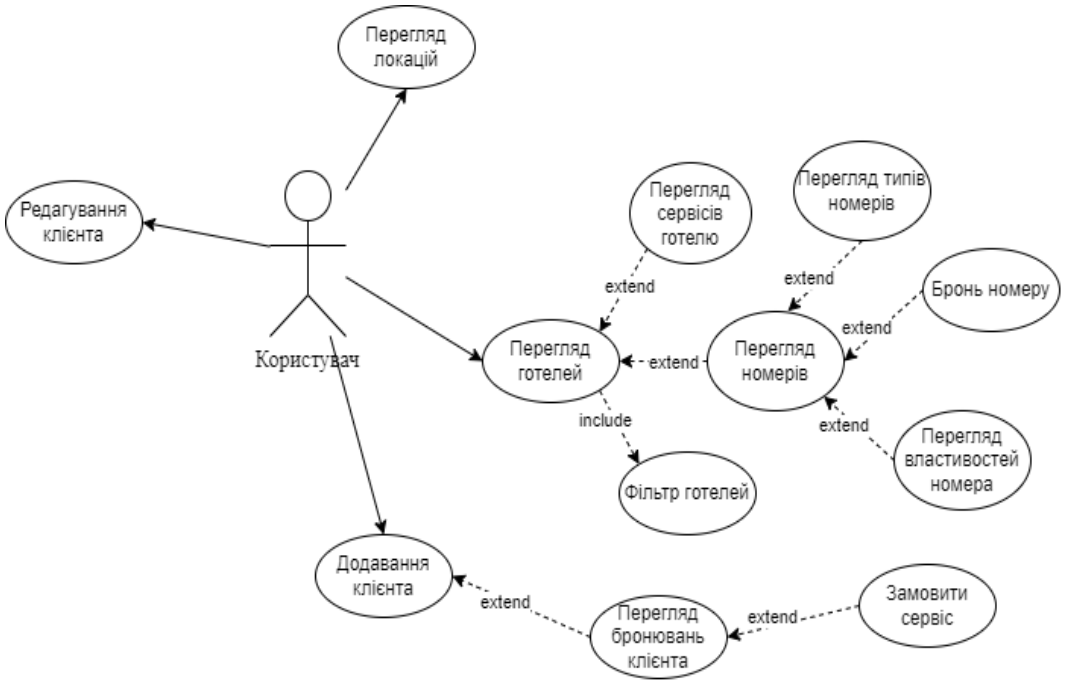

Рисунок 2.4 – Діаграма прецедентів клієнта

Найбільша кількість функціональних можливостей має адміністратор проєкту, адже фактично він відповідає за довідникову інформацію, що стосується всіх, він фактично має право втрутитися майже на будь-якому етапі, щоб допомогти розв’язати проблему, якщо вона раптом виникне в інших користувачів готелей має більше обмежень, у порівнянні з адміністратором проєкту, адже він вже не з системою працює, а використовує її для відповідних потреб бізнесу.

Клієнт також має можливість використовувати цей вебзастосунок та переглядати власну інформацію, а також замовляти наявні послуги готелей.

2.1.2 Діаграма бази даних

У будь-якому вебзастосунку, особливо такому як система бронювання готелів, ефективне управління даними є однією з ключових складових успіху. База даних є основним інструментом для зберігання та організації інформації, необхідної для функціонування системи. Вона містить набір таблиць, кожна з яких вмішує дані про конкретний аспект системи, наприклад, клієнтів, готелі, бронювання тощо. Взаємозв’язки між таблицями визначають способи, якими дані пов’язані між собою, що дозволяє здійснювати зв’язані запити та отримувати повну картину інформації. Діаграма бази даних виступає як інструмент візуалізації цієї структури, допомагаючи розуміти, як дані організовані та як вони взаємодіють між собою.

Microsoft SQL Server – це реляційна система керування базами даних (СКБД), розроблена компанією Microsoft. Вона надає набір інструментів для зберігання та управління даними, а також для роботи з ними. SQL Server дозволяє створювати бази даних, визначати їхню структуру, виконувати різноманітні операції з даними (додавання, зміну, видалення), а також проводити аналіз та оптимізацію роботи з базами даних [12].

Ця платформа підтримує різні мови програмування, такі як SQL, T-SQL, а також різноманітні інструменти розробки, такі як SQL Server Management

Studio (SSMS) для адміністрування та розробки баз даних, а також різноманітні сервіси та технології, наприклад, забезпечення безпеки даних, резервне копіювання та відновлення, реплікація даних, забезпечення високої доступності та інші.

Microsoft SQL Server використовується у багатьох організаціях та підприємствах для управління великими обсягами даних, а також вебзастосунків, корпоративних систем та інших інформаційних систем. Він є однією з найпоширеніших реляційних систем управління базами даних у світі (рисунок 2.5).

Рисунок 2.5 – Стандартний інтерфейс MS SQL Server

Основні функції MS SQL Server [13]:

– зберігання та керування даними: SQL Server дозволяє зберігати різні типи даних, включаючи текстові рядки, числа, дати, зображення та багато іншого. Він також надає функції для створення, читання, оновлення та видалення даних (CRUD);

– безпека даних: SQL Server пропонує широкий спектр функцій безпеки для захисту ваших даних від несанкціонованого доступу, модифікації та видалення;

– аналітика та звітування: SQL Server включає вбудовані інструменти для створення інтерактивних звітів та панелей моніторингу, які допомагають вам отримувати глибокі знання з ваших даних;

– масштабованість: SQL Server можна масштабувати від невеликих локальних серверів до великих хмарних розгортань, щоб відповідати потребам вашого бізнесу;

– інтеграція: SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Office 365 та Azure, що спрощує спільну роботу та аналітику даних.

MS SQL Server використовується у широкому колі галузей, включаючи:

– фінанси: зберігання та аналіз фінансових даних, таких як транзакції, рахунки та звіти;

– охорона здоров’я: зберігання та керування медичними записами, даними про пацієнтів та результатами досліджень;

– роздрібна торгівля: відстеження продажів, управління запасами та аналіз поведінки клієнтів;

– виробництво: управління ланцюжками постачання, відстеження виробництва та контроль якості;

– державне управління: зберігання та аналіз даних про громадян, податки та державні послуги.

Відповідно до зазначених переваг, було створено діаграму бази даних та її реалізовано (фінальний варіант представлено на рисунку 2.6).

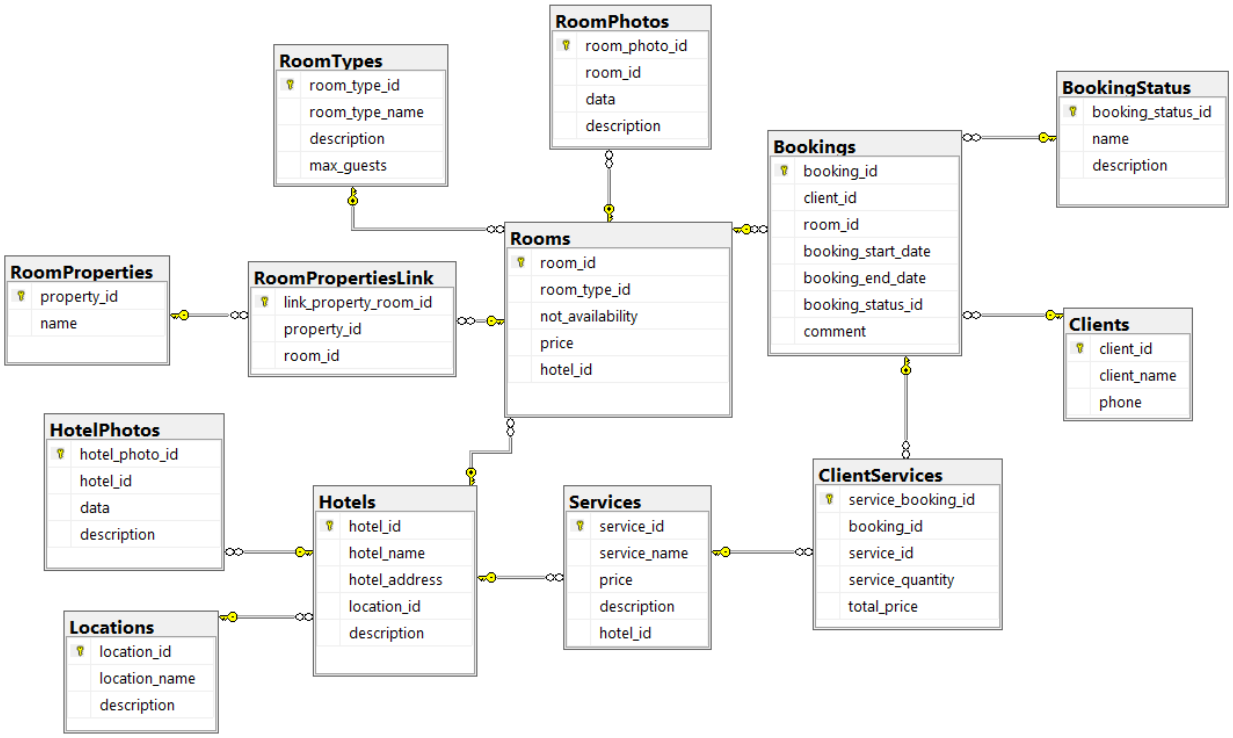

Рисунок 2.6 – Діаграма бази даних

2.2 Опис моделі даних

Згідно з розробленими діаграмами прецедентів та бази даних було створено базу даних, яка містить таблиці, опис яких наведено у таблиці 2.1.

Таблиця 2.1 – Таблиці бази даних вебзастосунку DreamHotel

Номер	**Таблиця**	**Опис**
1	**2**	**3**
1	Bookings	Таблиця для зберігання інформації про бронювання номерів.
2	BookingStatus	Таблиця для зберігання статусів бронювання.

Продовження таблиці 2.1

1	2	3
3	Clients	Таблиця для зберігання інформації про клієнтів.
4	ClientServices	Таблиця для зберігання інформації про послуги, що надаються клієнтам.
5	HotelPhotos	Таблиця для зберігання фотографій готелів.
6	Hotels	Таблиця для зберігання інформації про готелі.
7	Locations	Таблиця для зберігання інформації про розташування.
8	RoomPhotos	Таблиця для зберігання фотографій номерів.
9	RoomProperties	Таблиця для зберігання властивостей номерів.
10	RoomPropertiesLink	Таблиця для зв’язку властивостей з номерами.
11	Rooms	Таблиця для зберігання інформації про номери готелю.
12	RoomTypes	Таблиця для зберігання типів номерів.
13	Services	Таблиця для зберігання інформації про послуги, що надаються готелем.

Детальна інформація про кожну з таблиць бази даних міститься у таблицях 2.2 - 2.14, наведених нижче.

Таблиця 2.2 – Таблиця BookingStatus (статус бронювання)

Ключ	Атрибут	Тип	Опис
✓	booking_status_id	int	Унікальний ідентифікатор статусу бронювання.
	name	nvarchar(100)	Назва статусу бронювання.
	description	nvarchar(300)	Опис статусу бронювання.

Таблиця 2.3 – Таблиця Services (послуги готелю)

Ключ	Атрибут	Тип	Опис
✓	service_id	int	Унікальний ідентифікатор послуги.
	service_name	nvarchar(150)	Назва послуги.
	price	money	Вартість послуги.
	description	nvarchar(max)	Опис послуги.
	hotel_id	int	Ідентифікатор готелю, до якого відноситься послуга.

Таблиця 2.4 – Таблиця RoomTypes (типи номерів)

Ключ	Атрибут	Тип	Опис
✓	room_type_id	int	Унікальний ідентифікатор типу номера.
	room_type_name	nvarchar(100)	Назва типу номера.
	description	nvarchar(200)	Опис типу номера.
	max_guests	smallint	Максимальна кількість гостей, яку може розмістити цей тип номера.

Таблиця 2.5 – Таблиця RoomProperties (властивості номерів)

Ключ	Атрибут	Тип	Опис
✓	property_id	int	Унікальний ідентифікатор властивості.
	name	nvarchar(100)	Назва властивості.

Таблиця 2.6 – Таблиця RoomPhotos (фото номерів)

Ключ	Атрибут	Тип	Опис
✓	room_photo_id	int	Унікальний ідентифікатор фотографії номера.
	room_id	int	Ідентифікатор номера, до якого належить фотографія.
	data	varbinary(max)	Дані фотографії (зображення).
	description	nvarchar(250)	Опис фотографії номера.

Таблиця 2.7 – Таблиця Locations (розташування)

Ключ	Атрибут	Тип	Опис
✓	location_id	int	Унікальний ідентифікатор розташування.
	location_name	nvarchar(150)	Назва розташування.
	description	nvarchar(max)	Опис розташування.

Таблиця 2.8 – Таблиця Hotels (готелі)

Ключ	Атрибут	Тип	Опис
✓	hotel_id	int	Унікальний ідентифікатор готелю.
	hotel_name	nvarchar(100)	Назва готелю.
	hotel_address	nvarchar(300)	Адреса готелю.
	location_id	int	Зовнішній ключ, що посилається на ідентифікатор розташування.
	description	nvarchar(max)	Опис готелю.

Таблиця 2.9 – Таблиця HotelPhotos (фото готелю)

Ключ	Атрибут	Тип	Опис
✓	hotel_photo_id	int	Унікальний ідентифікатор фотографії готелю.
	hotel_id	int	Зовнішній ключ, що посилається на ідентифікатор готелю.
	data	varbinary(max)	Дані фотографії готелю.
	description	nvarchar(250)	Опис фотографії готелю.

Таблиця 2.10 – Таблиця Clients (клієнти)

Ключ	Атрибут	Тип	Опис
✓	client_id	int	Унікальний ідентифікатор клієнта.
	client_name	nvarchar(250)	Ім’я клієнта.
	phone	nchar(20)	Номер телефону клієнта.

Таблиця 2.11 – Таблиця ClientServices (надані послуги)

Ключ	Атрибут	Тип	Опис
✓	service_booking_id	int	Унікальний ідентифікатор послуги клієнта.
	booking_id	int	Ідентифікатор бронювання, з яким пов’язана послуга.
	service_id	int	Ідентифікатор послуги, яку замовив клієнт.
	service_quantity	int	Кількість послуг, що замовлено.
	total_price	money	Загальна вартість послуги.

Таблиця 2.12 – Таблиця Bookings (бронювання)

Ключ	Атрибут	Тип	Опис
✓	booking_id	int	Унікальний ідентифікатор бронювання.
	client_id	int	Ідентифікатор клієнта, що здійснив бронювання.
	room_id	int	Ідентифікатор номера, який було заброньовано.
	booking_start_date	date	Дата початку бронювання.
	booking_end_date	date	Дата закінчення бронювання.
	booking_status_id	int	Ідентифікатор статусу бронювання.
	comment	nvarchar(300)	Коментар до бронювання (необов’язково).

Таблиця 2.13 – Таблиця RoomPropertiesLink (властивості конкретних номерів)

Ключ	Атрибут	Тип	Опис
✓	link_property_room_id	int	Унікальний ідентифікатор зв’язку між властивістю та номером.
	property_id	int	Ідентифікатор властивості.
	room_id	int	Ідентифікатор номера, з яким пов’язана властивість.

Таблиця 2.14 – Таблиця Rooms (номери)

Ключ	Атрибут	Тип	Опис
✓	room_id	int	Унікальний ідентифікатор номера.
	room_type_id	int	Ідентифікатор типу номера, до якого належить цей номер.
	not_availability	smallint	Показник доступності номера (0 - доступний, 1 - недоступний).
	price	money	Ціна за проживання в цьому номері.
	hotel_id	int	Ідентифікатор готелю, до якого належить цей номер.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Реалізація функціонала вебзастосунка

Після успішної реалізації діаграм у розділі 2, наступним кроком буде розробка вебзастосунка з використанням ASP.NET. Цей вебзастосунок буде ключовим компонентом проєкту, оскільки він буде забезпечувати зручний та ефективний інтерфейс для користувачів.

Важливим аспектом розробки вебзастосунка буде забезпечення його зв’язку з базою даних. Це означає, що система повинна взаємодіяти з базою даних для отримання, збереження та оновлення інформації про доступне житло, додаткові послуги та інші важливі дані. Забезпечення надійного та ефективного зв’язку з базою даних є ключовим аспектом успішної реалізації вебзастосунка.

Крім того, важливо враховувати потреби різних категорій користувачів під час розробки інтерфейсу вебзастосунка. Для цього можна розділити функціонал за ролями, наприклад, адміністраторами проєкту, адміністрацією готелю та звичайними клієнтами. Кожна з цих ролей матиме свої унікальні можливості та інтерфейс, що дозволить забезпечити зручність та ефективність використання для кожного типу користувачів.

Таким чином, розробка вебзастосунка буде включати не лише створення інтерфейсу, але й розподіл функціонала за ролями, забезпечення зв’язку з базою даних та створення зручного та комфортного інтерфейсу для кінцевих користувачів.

3.1.1 Створення проєкту за шаблоном ASP.NET

Проєкт створено в інтегрованому середовищі розробки Visual Studio 2022 [14]. Для реалізації вебзастосунка було обрано платформу ASP.NET.

ASP.NET є фреймворком розробки вебзастосунків, розроблений компанією Microsoft. Він надає розробникам зручний і потужний інструментарій для створення вебзастосунків, які можуть працювати на різних платформах, включно з Windows, Linux і macOS [15, 16].

Переваги ASP.NET:

– масштабованість: ASP.NET дозволяє легко масштабувати вебзастосунки для відповіді на висхідний обсяг трафіку та вимог користувачів;

– швидкодія: завдяки вбудованим оптимізаціям та підтримці технологій, таких як Just-In-Time компіляція, ASP.NET може забезпечити високу продуктивність вебзастосунків;

– безпека: ASP.NET має вбудовану систему безпеки, яка допомагає уникнути таких проблем, як атаки SQL-ін’єкцій та перехоплення даних;

– інтеграція з іншими продуктами Microsoft: оскільки ASP.NET розроблений Microsoft, він має глибоку інтеграцію з іншими продуктами, такими як SQL Server, Azure та Visual Studio;

– підтримка мов програмування: ASP.NET підтримує різні мови програмування, такі як C#, VB.NET, F#, що дає розробникам можливість вибирати мову, яка найбільш підходить для їх потреб.

Недоліки ASP.NET:

– вартість: деякі інструменти та функціонал ASP.NET можуть бути дорогими, особливо для невеликих команд або стартапів;

– блокування постачальника: використання ASP.NET зазвичай зв’язує вас з екосистемою Microsoft, що може ускладнити перехід до інших платформ або технологій у майбутньому;

– великий розмір: вебзастосунки, розроблені на ASP.NET, можуть мати великі розміри через велику кількість вбудованих бібліотек і функціоналу;

– системні вимоги: для розгортання ASP.NET додатків може знадобитися специфічне програмне забезпечення та обладнання, що може вимагати додаткових витрат та зусиль.

Враховуючи всі переваги та недоліки цієї платформи, було створено проєкт «WebApplicationDreamHotel» із використанням шаблону проєктування MVC.

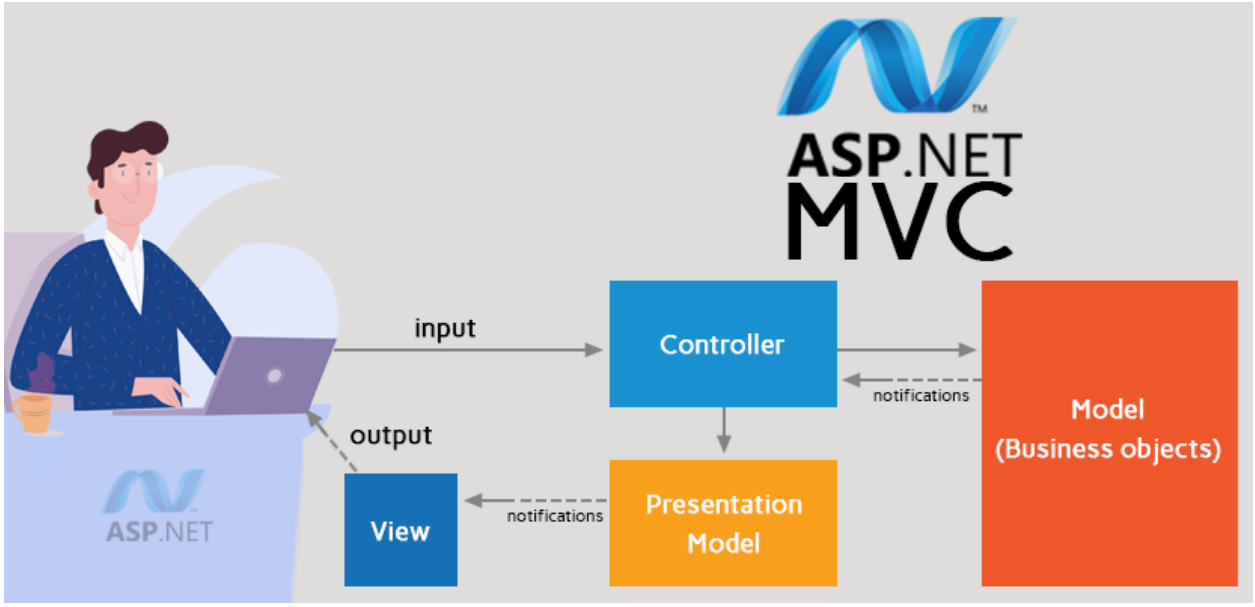

Рисунок 3.1 – Схема ASP.NET MVC

MVC (Model-View-Controller) – це популярний шаблон проєктування для розробки програмного забезпечення, особливо для вебзастосунків (рисунок 3.1). Він розділяє програму на три основні компоненти [17]:

– модель (Model): модель являє собою бізнес-логіку програми, дані та стан системи. Вона відповідає за обробку даних, доступ до бази даних, виконання операцій із даними та валідацію. Модель незалежна від інтерфейсу користувача;

– вид (View): вид відповідає за представлення даних користувачеві. Він відображає інформацію, яка знаходиться в моделі у вигляді, зрозумілому для користувача. Вид не має бізнес-логіки і зазвичай включає шаблони, HTML, CSS та інші елементи для створення інтерфейсу користувача;

– контролер (Controller): контролер приймає вхідні дані від користувача через інтерфейс користувача, обробляє їх і взаємодіє з моделлю, здійснюючи відповідні операції. Він реагує на дії користувача, викликає відповідні методи моделі та відображення, і повертає результати користувачеві.

Переваги використання MVC:

– розділення відповідальності: MVC чітко розділяє відповідальність між трьома компонентами, що робить код більш організованим, зрозумілим та керованим;

– простота тестування: кожен компонент MVC можна тестувати окремо, що спрощує тестування та налагодження програми;

– гнучкість: MVC легко розширювати та модифікувати, що робить його гарним вибором для складних вебзастосунків;

– повторне використання коду: компоненти MVC можна повторно використовувати в різних частинах програми, що економить час та зусилля розробників.

3.1.2 Підключення бази даних до проєкту

Для зручної роботи із базою даних використаємо Entity Framework – це набір технологій в рамках платформи .NET, який надає зручний і потужний інструментарій для роботи з базами даних з використанням об’єктно-орієнтованого підходу [18]. У контексті C#, Entity Framework дозволяє розробникам працювати з базами даних, не використовуючи прямої роботи з

SQL-запитами, а замість цього використовувати об’єктно-орієнтовані концепції, такі як класи та об’єкти.

Основні концепції, пов’язані з Entity Framework:

– code first: це підходить до розробки, коли розробники спочатку створюють класи для своїх моделей даних, а потім Entity Framework автоматично генерує схему бази даних на основі цих класів;

– database first: цей підхід передбачає визначення моделей даних на основі наявної бази даних. Entity Framework автоматично створює класи для цих моделей даних;

– model first: цей підхід включає роботу з графічним інструментом для визначення моделей даних. Потім Entity Framework генерує код класів на основі цих моделей.

Entity Framework дозволяє використовувати LINQ (Language Integrated Query) для написання запитів до бази даних, що робить роботу з даними більш зручною та ефективною. Він також надає можливості для використання відносин між об’єктами, мапінгування даних і роботи з транзакціями (рисунок 3.2).

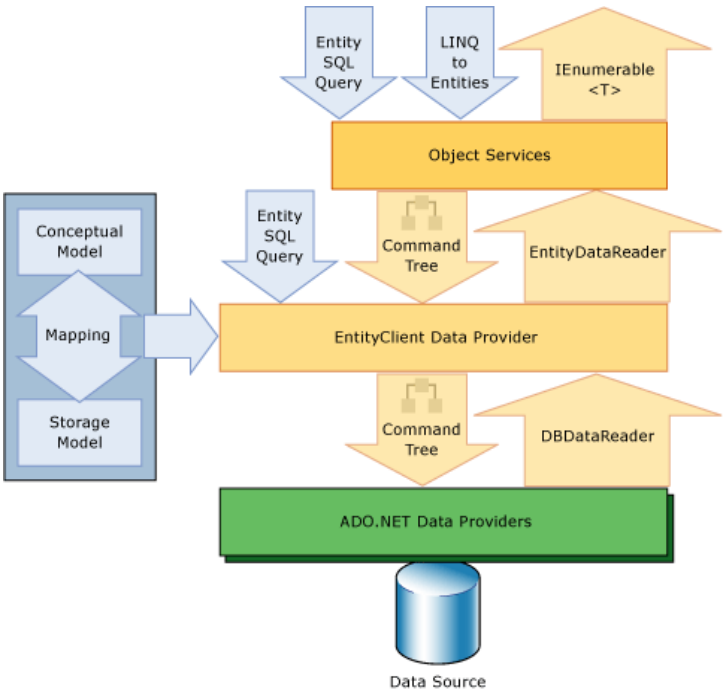

Рисунок 3.2 – Архітектура Entity Framework

Для використання технології, потрібно підключити NuGet пакети – це невеликі пакети, що містять код, бібліотеки та інші ресурси, необхідні для розробки програмного забезпечення на платформі .NET [19]. Вони слугують зручним способом спільного використання, повторного використання та розповсюдження коду, а також полегшують процес розробки та інтеграції.

Для роботи з наявною БД MS SQL Server додаємо три пакети: Microsoft.EntityFrameworkCore.SqlServer – надає функціональність Entity Framework для роботи з MS SQL Server; a Microsoft.EntityFrameworkCore.Tools та Microsoft.EntityFrameworkCore (рисунок 3.3) необхідні для створення класів за базою даних, тобто для reverse engineering.

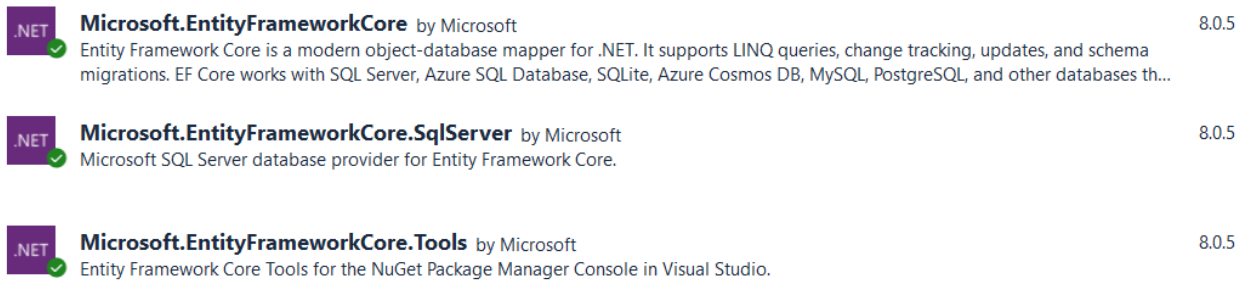

Рисунок 3.3 – Підключені NuGet пакети Microsoft.EntityFrameworkCore

Для подальшого підключення бази даних використаємо підхід Database First, тобто створення моделей за попередньо створеною базою даних. З цією метою, перейдемо до Package Manager Console (це інструмент, призначений для управління пакетами NuGet) та виконати команду:

Scaffold-DbContext «Server=DESKTOP-T2KTETO\SQLEXPRESS; Database=DreamHotel; Trusted_Connection=True; TrustServerCertificate=True;» Microsoft.EntityFrameworkCore.SqlServer

Розбір команди:

– Scaffold-DbContext – ця команда використовується для автоматичного генерації класів DbContext у проєкті C# на основі наявної бази даних Microsoft SQL Server.

– Server=DESKTOP-T2KTETO\SQLEXPRESS – вказує на ім’я сервера та екземпляр SQL Server.

– Database=DreamHotel – вказує на назву бази даних, до якої потрібно підключитися (DreamHotel).

– Trusted_Connection=True – використовує інтегровану автентифікацію Windows для підключення до бази даних. Тобто, для підключення використовується обліковий запис користувача Windows, під яким запущена програма.

– TrustServerCertificate=True – вказує на довіру до сертифіката сервера бази даних. Зазвичай це потрібно лише у випадках, коли використовується самопідписаний сертифікат.

– Microsoft.EntityFrameworkCore.SqlServer – цей параметр вказує на пакет NuGet, який необхідний для роботи з базами даних Microsoft SQL Server в Entity Framework Core.

Команда Scaffold-DbContext проаналізувала структуру бази даних DreamHotel та автоматично згенерувала класи DbContext та класи моделей, які відповідають таблицям та стовпчикам у проєкту C#. Відповідно до шаблону проєктування MVC, переносимо створені моделі у теку Models (рисунок 3.4). Важливо, що назви таблиць маємо у множині (Hotels, Clients тощо), а класи було створено із назвами в однині (Hotel, Client тощо).

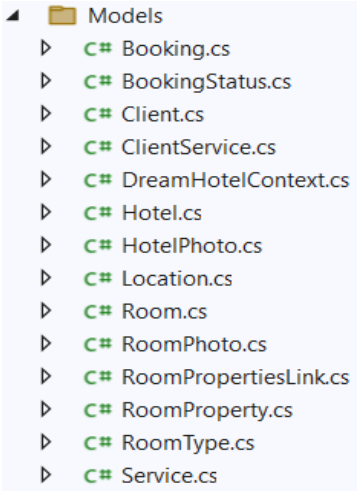

Рисунок 3.4 – Класи моделей, створені відповідно до бази даних

Для підключення до бази даних, необхідно задати параметри підключення. Для цього додаємо новий параметр ConnectionStrings у файлі appsettings.json та внести зміни у файл Program.cs для налаштування сервісу доступу до бази даних у вебзастосунку.

Для створення контролера для класу моделі Entity Framework використаємо Scaffolded Item – це термін, який використовується в контексті розробки програмного забезпечення, зокрема в середовищі розробки Visual Studio та при роботі з різними шаблонами проєктів.

У контексті Visual Studio Scaffolded Item (або просто Scaffold) – це автоматично створений або згенерований файл або набір файлів, які додаються до вашого проєкту на основі певного шаблону чи шаблону коду. Scaffolded Items можуть включати контролери, представлення, моделі, ресурси, конфігураційні файли та інші складові вебзастосунку.

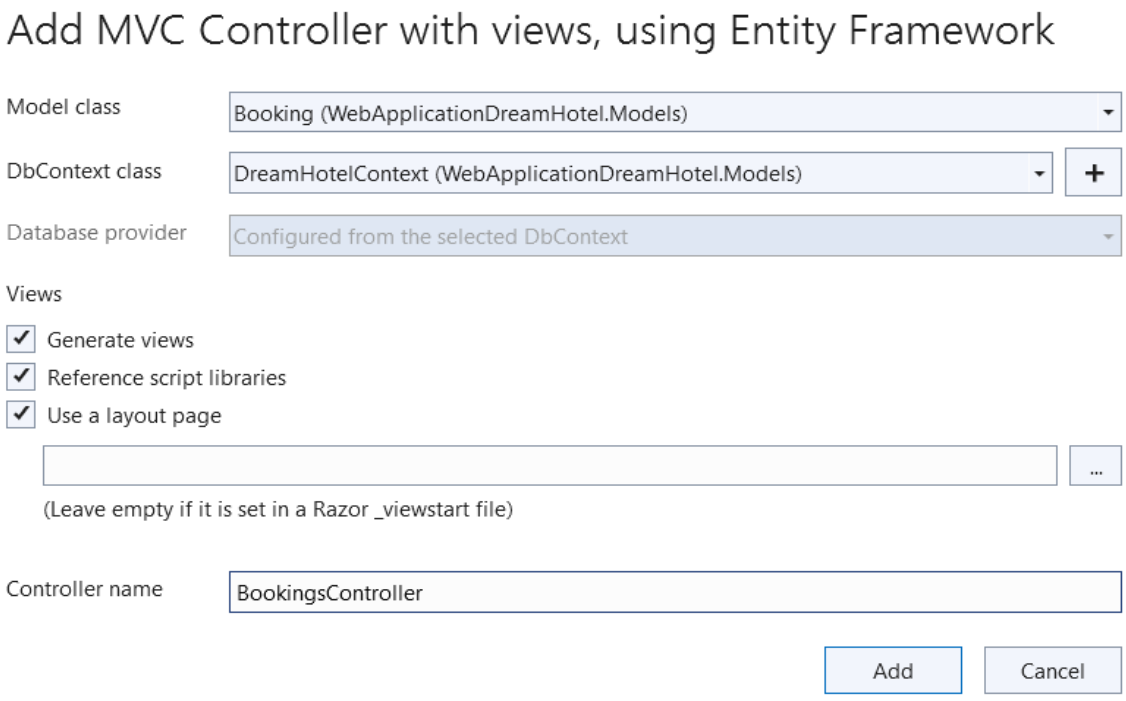

Рисунок 3.5 – Приклад налаштування створення контролера для моделі

Наприклад, якщо створюється новий контролер у проєкті ASP.NET MVC за допомогою шаблону «Empty Controller», Visual Studio автоматично згенерує файл контролера з методами дій та іншими необхідними

відомостями. Цей файл контролера буде розгорнутим (scaffolded) елементом вебзастосунка. Замість того, щоб вручну створювати кожен файл або компонент, можна скористатися Scaffolded Items (рисунок 3.5), які надаються Visual Studio, щоб автоматично згенерувати необхідні складові за певними шаблонами.

Важливо зауважити, що крім контролера, назва якого буде згенерована у множині (наприклад, Booking => BookingsController), також буде створено базові відображення.

3.1.3 Запити до бази даних

LINQ (Language Integrated Query) – це мовна конструкція в C#, яка дозволяє писати запити до джерел даних, таких як колекції об’єктів, бази даних, XML-документи та інші джерела даних, безпосередньо у коді програми [20]. Одним з найпоширеніших використань LINQ є створення запитів до бази даних з використанням Entity Framework.

LINQ не є окремою мовою, а інтегрована в синтаксис C#, що робить запити до даних більш природними та читабельними. Це значно спрощує роботу з даними в C# порівняно з традиційними методами.

Основні можливості LINQ:

– фільтрація: відбір даних, що відповідають певним критеріям;

– сортування: впорядкування даних за одним або декількома полями;

– проєкція: вибір та перетворення даних для отримання необхідних результатів;

– об’єднання: комбінування даних з декількох джерел;

– агрегація: виконання операцій над даними, таких як підрахунок, сумування або знаходження середнього значення.

Переваги використання LINQ:

– простота та читабельність: запити LINQ написані за допомогою синтаксису, схожого на SQL, що робить їх зрозумілими та легкими для вивчення;

– потужність та гнучкість: LINQ пропонує широкий спектр операцій для роботи з даними, що робить його придатним для розв'язання складних задач;

– тип-безпечність: LINQ забезпечує тип-безпечність, що допомагає запобігти помилкам під час роботи з даними;

– інтеграція з іншими технологіями C#: LINQ легко інтегрується з іншими мовними конструкціями C#, що робить його універсальним інструментом для розробки.

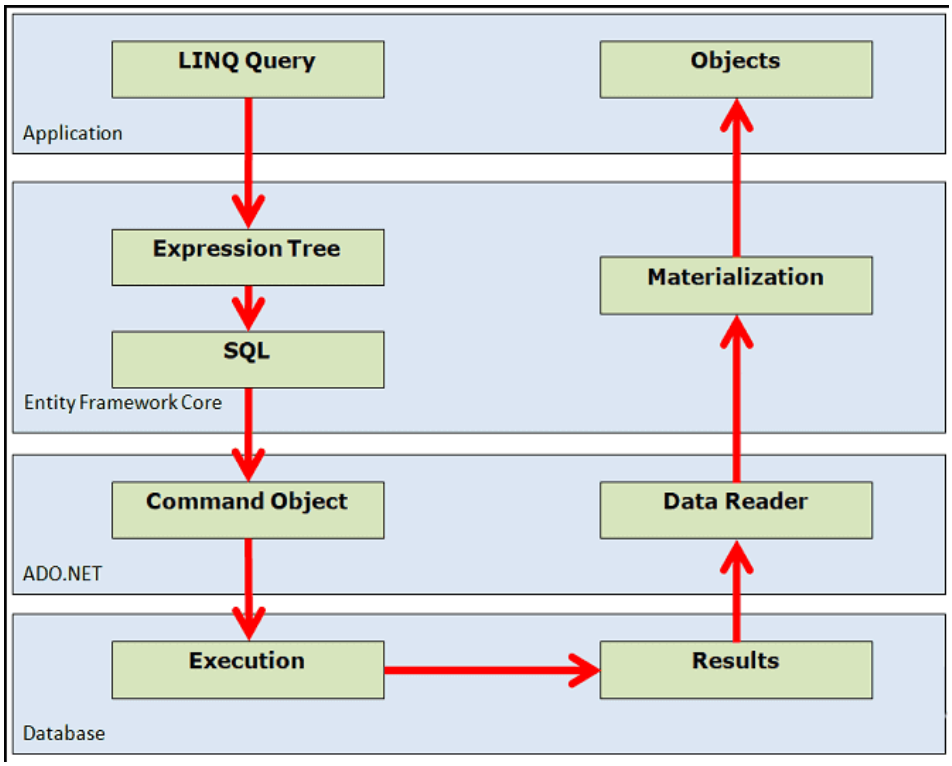

Рисунок 3.6 – Схема роботи запиту LINQ до бази даних

Отже, саме LINQ використано у проєкті для взаємодії зі створеною базою даних (рисунок 3.6). Далі наведено 10 базових прикладів використання таких запитів у вебзастосунку.

В усіх запитах _context – це змінна, яка представляє об’єкт контексту Entity Framework. Async наприкінці назви вказує на те, що це асинхронний метод.

1. _context.RoomProperties.FirstOrDefaultAsync(m =>m.PropertyId == id)

RoomProperties – це об’єкт контексту, який відповідає за взаємодію із таблицею RoomProperties у базі даних.

FirstOrDefaultAsync – цей метод шукає перший елемент у колекції, який відповідає певній умові, і повертає його. Якщо такого елемента не знайдено, він повертає значення null.

m => m.PropertyId == id – це лямбда-вираз, який визначає умову, за якою потрібно шукати елемент. Він отримує елемент m з колекції RoomProperties і перевіряє, чи значення його властивості PropertyId дорівнює змінній id.

Цей код ідентичний до виклику SQL-запиту:

SELECT TOP 1 * FROM RoomProperties WHERE property_id = @id.

2. _context.Add(roomProperty)

roomProperty – це об’єкт RoomProperty (відповідає рядку з таблиці RoomProperties у базі даних), який містить дані нового запису.

Add(roomProperty) – це метод, який використовується для додавання нового об’єкта до контексту. При цьому у саму базу даних, нова інформація ще не потрапляє.

Цей код ідентичний до виклику SQL-запиту:

INSERT INTO RoomProperties (name, property_id) VALUES (@propertyName, @property_id).

3. _context.SaveChangesAsync()

Контекст Entity Framework відстежує зміни, які ви вносите до об’єктів, що представляють дані в базі даних. Ці зміни не збережуться в базі даних, доки ви не викличете метод SaveChanges() або SaveChangesAsync().

Виклик методу SaveChanges() виконує наступні дії:

– перевіряє всі зміни, внесені до контексту;
– генерує SQL-запити, необхідні для збереження цих змін;

– відправляє SQL-запити до бази даних;

– оновлює стан об’єктів в контексті, щоб відобразити збережені зміни.

4. _context.RoomProperties.FindAsync(id)

Код виконує асинхронний запит до бази даних, шукаючи конкретний запис в таблиці RoomProperties. Цей запис визначається значенням змінної id, яка містить первинний ключ шуканого запису. Якщо знайдено запис із таким ідентифікатором, метод FindAsync повертає об’єкт RoomProperty, який містить дані цього запису. Якщо ж запису з таким ідентифікатором не знайдено, метод FindAsync повертає значення null.

Цей код ідентичний до виклику SQL-запиту:

SELECT * FROM RoomProperties WHERE property_id = @id.

5. _context.Update(roomProperty)

Update(roomProperty) – це метод, який використовує Entity Framework для відстеження того, що об’єкт roomProperty призначений для оновлення бази даних. Сам викликне зберігає дані в базі даних. Він лише повідомляє Entity Framework, що властивості об’єкта roomProperty містять нові значення, які потрібно зберегти.

За умови зміни назви властивості цього номеру, код може бути перетворено на SQL-запит вигляду:

UPDATE RoomProperties SET name = @newPropertyName WHERE property_id = @roomPropertyId.

6. _context.RoomProperties.Remove(roomProperty)

Remove(roomProperty) – це метод, який використовує Entity Framework для відстеження того, що об’єкт roomProperty призначений для видалення з бази даних. Сам виклик не видаляє запис з бази даних. Він лише повідомляє Entity Framework, що об’єкт roomProperty представляє запис, який потрібно видалити.

Цей код ідентичний до виклику SQL-запиту:

DELETE FROM RoomProperties WHERE property_id = @propertyId.

7. _context.RoomProperties.Any(e => e.PropertyId == id);

Any(e => e.PropertyId == id) – це метод LINQ, який використовується для перевірки, чи існує хоча б один елемент в колекції RoomProperties, де значення властивості PropertyId дорівнює значенню змінної id. Метод Any повертає значення true, якщо знайдеться хоча б один запис, що відповідає умові, і значення false, якщо жодного запису не знайдеться

Цей код ідентичний до виклику SQL-запиту:

SELECT EXISTS

(SELECT 1 FROM RoomProperties WHERE PropertyId = @id).

8. _context.Bookings.Include(b=>b.BookingStatus).Include(b=>b.Client) .Include(b => b.Room)

Bookings – властивість контексту, яка представляє DbSet<Booking>. DbSet – це тип колекції, що використовується для роботи з таблицею Bookings в базі даних.

Include(...) – це метод розширення LINQ, який використовує Entity Framework для завантаження пов’язаних даних з іншої таблиці для кожного об’єкта Booking в результаті запиту.

Отже, весь цей код виконує один запит до бази даних, який отримує дані з таблиці Bookings, а також пов’язані дані з таблиць BookingStatus, Client, і Room для кожного запису в таблиці Bookings. Це допомагає уникнути виконання окремих запитів для отримання пов’язаних даних пізніше.

Цей код аналогічний до виклику SQL-запиту:

SELECT b.*, bs.*, c.*, r.*

FROM Bookings b

INNER JOIN BookingStatus bs ON b.booking_status_id = bs.booking_status_id INNER JOIN Client c ON b.client_id = c.client_id

INNER JOIN Room r ON b.room_id = r.room_id.

9. _context.Hotels.Include(h=>h.Location).Where(h=>h.Location .LocationName.StartsWith(locationName));

Hotels – властивість контексту, яка представляє DbSet<Hotel>. DbSet – це тип колекції, що використовується для роботи з таблицею Hotels в базі даних.

.Where(...) – цей метод LINQ Where фільтрує результати запиту, залишаючи лише готелі, де значення поля LocationName в пов’язаній таблиці Location починається із символів змінної locationName.

Цей код аналогічний до виклику SQL-запиту:

SELECT h.*, l.* FROM Hotels h INNER JOIN Location l ON h.location_id = l.location_id WHERE l.location_name LIKE @locationName + ‘%’.

10. _context.Services.Include(s => s.Hotel).OrderBy(s=>s.Price);

Services – властивість контексту, яка представляє DbSet<Service>. DbSet – це тип колекції, що використовується Entity Framework для роботи з таблицею Services в базі даних.

.OrderBy(s => s.Price): Метод OrderBy вказує на те, що результати запиту повинні бути відсортовані за значенням поля Price в класі Service.

Цей код аналогічний до виклику SQL-запиту:

SELECT s.*, h.* FROM Services s INNER JOIN Hotel h ON s.hotel_id = h.hotel_id ORDER BY s.price ASC.

3.1.4 Дизайн вебзастосунка

У вебзастосунку вже реалізовано базовий функціонал для виведення категорій книг та додавання до них нових книг. Застосунок використовує декілька представлень, які фактично нагадують вебсторінки.

Якщо виникне потреба застосувати до кожної вебсторінки стилі з зовнішнього файлу CSS, то доведеться під’єднати цей файл у кожному представленні. Аналогічно, якщо виникне потреба створити спільне для всіх

вебсторінок меню або інші загальні елементи інтерфейсу, наприклад, футер, то знову ж таки доведеться прописувати всі ці елементи в кожному представленні. Такий підхід не є оптимальним, адже при внесенні змін до цих загальних елементів доведеться редагувати код у кожному представленні, яких може бути досить багато.

Набагато кращим рішенням є використання майстер-сторінок. За замовчуванням в ASP.NET MVC вже є майстер-сторінка, яка називається _Layout.cshtml і знаходиться в теці Views/Shared. Цю майстер-сторінку можна трохи змінити під потреби свого проєкту.

Використання майстер-сторінок має ряд переваг:

– повторне використання коду: стилі, меню та інші загальні елементи інтерфейсу описуються лише один раз у майстер-сторінці, після чого вони автоматично включаються у всі представлення;

– зниження складності коду: структура коду стає чіткішою та логічною, адже загальні елементи інтерфейсу описуються в одному місці;

– зручність внесення змін: при оновленні загальних елементів інтерфейсу нам потрібно редагувати лише код майстер-сторінки, а не кожного представлення окремо.

Спочатку підключаються стилі бібліотеки Bootstrap, яка за замовчуванням вже є в проєкті в теці wwwroot / lib / bootstrap / dist / css /.

Бібліотека Bootstrap – це набір інструментів для розробки вебінтерфейсів із використанням HTML, CSS і JavaScript [21]. Bootstrap надає готові шаблони, компоненти та стилізацію, що допомагають зробити вебсайти більш адаптивними та зручними для користувачів на різних пристроях.

Після секції head на майстер-сторінці йде створення меню. Для створення меню тут застосовуються стандартні класи Bootstrap. Далі в основній частині йде виклик методу RenderBody() – за допомогою цього методу в це місце буде підставлятися розмітка вже конкретних представлень.

При створенні нових представлень не було вказано конкретний Layout – тому, використовується типова майстер-сторінка (рисунок 3.7)

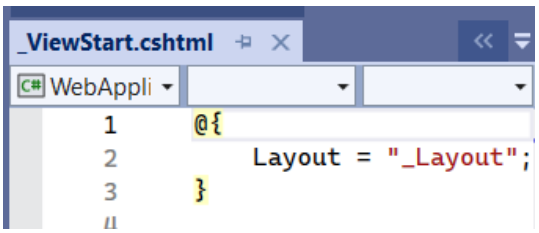

Рисунок 3.7 – Фрагмент коду

Для зміни теми bootstrap буде перейдено за посиланням на сайт та обрано тему Minty (рисунок 3.8).

Рисунок 3.8 – Тема Minty bootstrap

Після чого було завантажено 2 файли (рисунок 3.10) та під'єднано їх до проєкту у відповідній теці.

Після додавання файлів до проєкту, необхідно прописати їх використання у коді файлу _Layout.cshtml (рисунок 3.9).

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - DreamHotel</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap_minty.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/WebApplicationDreamHotel.styles.css" asp-append-version="true" />
</head>
```

Рисунок 3.9 – Файли для підключення теми в проєкті

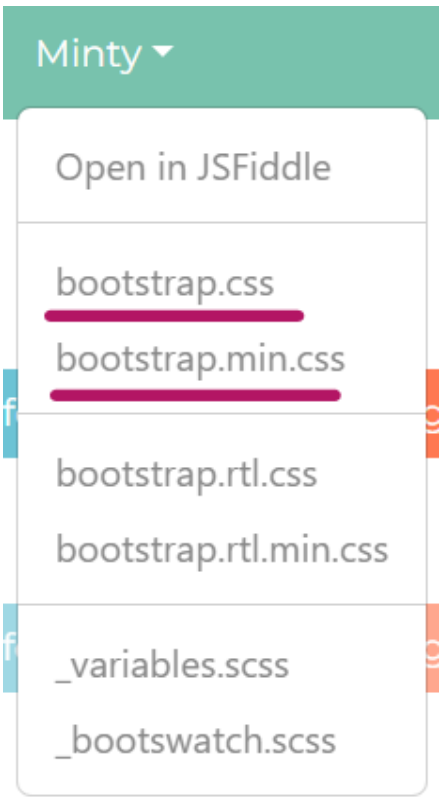

Рисунок 3.10 – Файли для підключення теми в проєкті

Також звернемо увагу на локалізацію – автоматично у всіх відображення надписи будуть англійською мовою, проте проєкт орієнтується на україномовну аудиторію, тому варто виконати допрацювання.

Для зміни написів полів моделей не потрібно робити у кожному з відображень, необхідно використовувати атрибут Display, наприклад:

```
[Display(Name = "Опис")]
public string Description { get; set; }
```

Крім цього атрибуту, також можна використовувати і інші, наприклад:

```
[Required]
[Compare("Password", ErrorMessage = "Паролі не співпадають")]
[Display(Name = "Підтвердження паролю")]
[DataType(DataType.Password)]
public string PasswordConfirm { get; set; }
```

Звідки отримуємо:

– [Required]: Цей атрибут вказує, що поле є обов’язковим для заповнення. Використання атрибута гарантує, що значення поля не буде null або порожнім.

– [Compare("Password", ErrorMessage = "Паролі не співпадають")]: Цей атрибут порівнює значення поточного поля з іншим полем, вказаним в параметрі ("Password" у вашому випадку). Він використовується, наприклад, для підтвердження пароля, де поле "PasswordConfirm" порівнюється з полем "Password". Якщо значення не збігаються, буде відображено повідомлення про помилку, яке ви задали у параметрі ErrorMessage.

– [Display(Name = "Підтвердження пароля")]: Цей атрибут використовується для відображення іншої назви поля в текстовому віджеті (наприклад, у формі). У вашому випадку, замість стандартної назви поля ("PasswordConfirm"), буде відображена назва "Підтвердження пароля".

– [DataType(DataType.Password)]: Цей атрибут вказує, що поле містить пароль. Він використовується для підказки клієнту, що поле містить конфіденційну інформацію (наприклад, пароль), та може впливати на його відображення у деяких контекстах, наприклад, при використанні HTML-поля вводу.

3.1.5 Розподіл на ролі

ASP.NET Identity – це система управління автентифікацією і авторизацією для застосунків на платформі ASP.NET [22]. Кілька переваг використання ASP.NET Identity:

– готові рішення: ASP.NET Identity надає готові рішення для основних задач, пов’язаних з управлінням користувачами, такими як реєстрація, вхід, відновлення пароля тощо. Це дозволяє швидко впроваджувати

функціональність автентифікації та авторизації без необхідності писати код з нуля;

– безпека: ASP.NET Identity має вбудовану підтримку для різних методів безпеки, таких як хешування паролів, двофакторна аутентифікація, блокування облікових записів після невдалих спроб входу тощо. Це допомагає забезпечити захист від зловмисників та небажаних втручань у систему;

– розширюваність: ASP.NET Identity дозволяє легко розширювати та налаштовувати функціональність автентифікації та авторизації відповідно до потреб конкретного проєкту;

– інтеграція з іншими сервісами: ASP.NET Identity добре інтегрується з іншими службами Microsoft, такими як Azure Active Directory, що дозволяє використовувати їхні можливості для управління користувачами та безпекою.

Загалом, використання ASP.NET Identity дозволяє розробникам ефективно впроваджувати функціональність автентифікації та авторизації у свої вебзастосунки, забезпечуючи високий рівень безпеки та зручності для користувачів (рисунок 3.11).

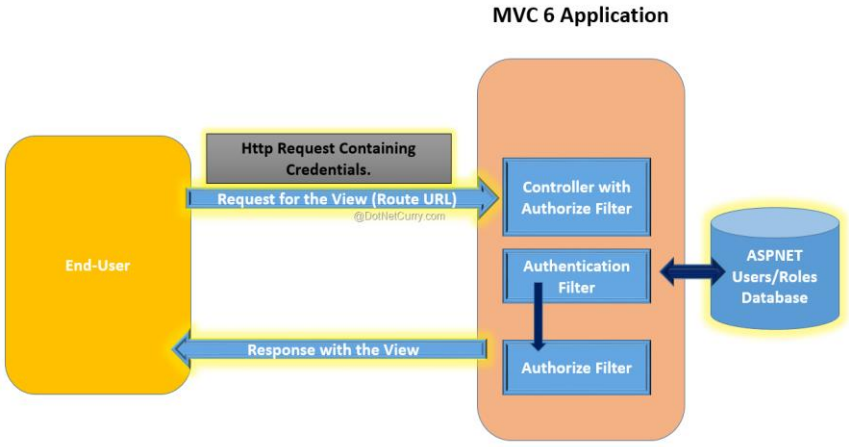

Рисунок 3.11 – Реалізація автентифікації користувачів в ASP.NET MVC 6

Для реалізації розподілу на ролі, використаємо саме ASP.NET Identity. Для цього створено новий клас User (у теці Models), який представляє користувача та успадковується від класу IdentityUser, переймаючи все його

властивості. Для взаємодії з MS SQL Server через ASP.NET Core Identity додамо в проєкт через Nuget пакет Microsoft.AspNetCore.Identity.EntityFrameworkCore.

Оскільки для реалізації потрібна ще одна база даних, то необхідно додати клас контексту даних IdentityContext (підхід Model First). Крім того, клас контексту даних має успадковуватися не від DbContext, а від IdentityDbContext. Далі обов’язково виконуємо підключення авторизації та реєстрації до проєкту. Для встановлення доступу до контролера та/або його метода потрібно використовувати атрибут Authorize(наприклад, [Authorize(Roles ="admin")].

3.1.6 Подальші варіанти розвитку

На будь-якому етапі проєкту вебзастосунка важливо враховувати його подальше розвиток з кількох ключових причин. Перш за все, розробникам потрібно передбачити масштабованість системи, зокрема розширення функціоналу та зростання кількості користувачів без втрати продуктивності. Використання модульної архітектури або мікросервісів дозволяє легше додавати нові компоненти та покращувати систему по мірі росту бізнесу.

Також важливо планувати заходи з безпеки та захисту даних з самого початку, щоб уникнути потенційних загроз у майбутньому. Крім того, ефективне управління UX та інтеграція з іншими системами дозволяють підтримувати конкурентоспроможність продукту і забезпечувати швидке реагування на потреби користувачів та ринку. Урахування цих аспектів дозволяє економити ресурси та забезпечує більш плавний та ефективний розвиток вебзастосунка в майбутньому.

Один із ключових аспектів бізнесу – аналіз. Вкрай необхідно мати можливість проаналізувати наявні дані, на основі яких розробити стратегію подальшого розвитку. Наприклад, в подальшому для аналіза даних

(кластерізація користувачів, аналіз відгуків, динамічне ціноутворення) можна застосувати модифікований рекурентний метод достовірної нечіткої кластеризації [23]. Переваги цього методу у тому, що запропонований модифікований рекурентний метод достовірної нечіткої кластеризації, що використовує оптимізаційну процедуру на основі косяків риб, є ефективним і простим у числовій реалізації способом знаходження глобальних екстремумів у складних функціях кластеризації даних. Він поєднує ройові та еволюційні алгоритми, що дозволяє ефективно вирішувати задачі кластеризації великих масивів даних.

Ще однією проблемою користувачів, що бронюють готелі є недостовірні зображення. Одним з можливих варіантів допрацювання вебзастосунка може бути аналіз зображень, які додає адміністрація готелю або власники житла. Один з варіантів, яким можна скористатися це спосіб, протестований на аналізі медичних зображень як об’єктів досліджень [24]. Окрім цього, також можна реалізувати інтеграцію з різними джерелами, для аналізу більшої кількості даних, а не тільки тих .які надав власник. Оскільки кожне джерело може мати свою структуру даних і формати представлення, то для розв’язання цієї проблеми можна використати методи семантичної інтеграції [25], що дозволяють створювати єдине уявлення про ці дані, адже вони допомагають визначати еквівалентність атрибутів і стандартизувати їх представлення.

3.2 Опис основного функціонала вебзастосунка

Відповідно до поставленої задачі було розроблено застосунок для бронювання готелей.

При початковому запуску вебзастосунка опиняємося на базовій сторінці (рисунок 3.12), з якої можемо виконати вхід або реєстрацію.

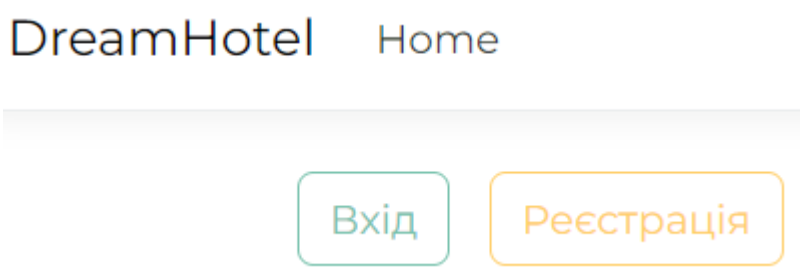

Рисунок 3.12 – Базова сторінка вебзастосунка

За умови вибору реєстрації, перейдемо до відповідної сторінки (рисунок 3.13), де будь-який новий користувач може створити собі акаунт. Автоматично йому буде надано роль «client», тобто максимально обмежені права, спрямовані на пошук готелю, бронювання та замовлення додаткових послуг.

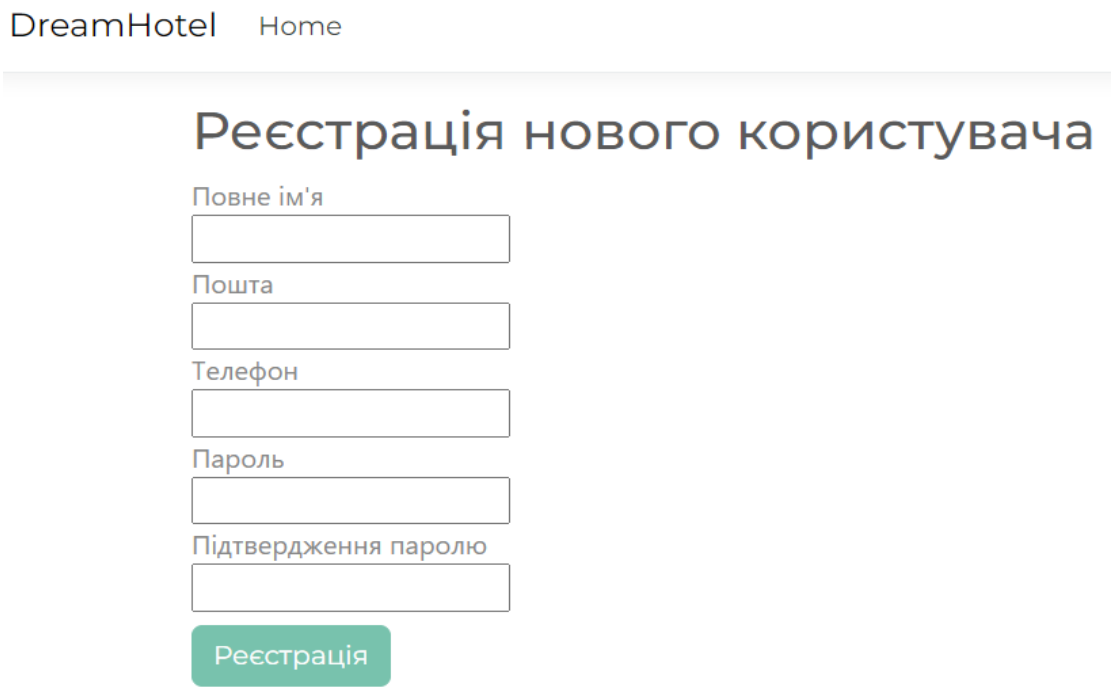

Рисунок 3.13 – Сторінка реєстрації нового користувача

Якщо акаунт вже є, то можна увійти під відповідними даними та продовжити роботу (рисунок 3.14). За потреби, можна «запам’ятати» дані, щоб кожен раз не вводити повторно.

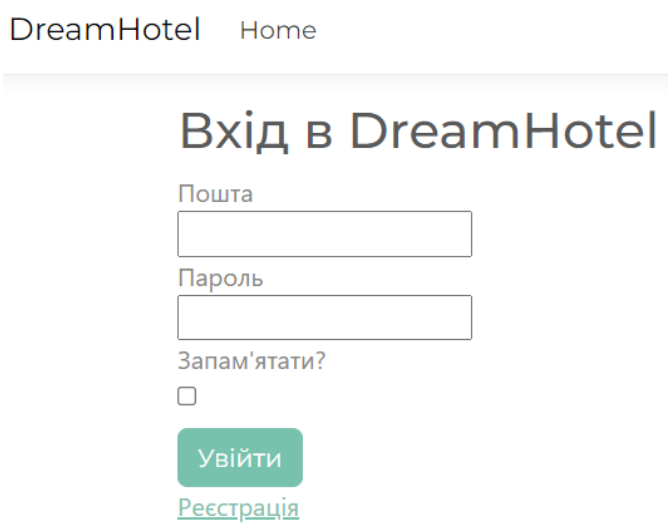

Рисунок 3.14 – Сторінка входу у вебзастосунок

В системі наразі є три варіанти ролей: адміністратор вебзастосунка («admin»), адміністратор готелей («hotelAdmin») та клієнт («client»). Найбільша повнота «влади» саме у адміністратора вебзастосунка, тому розглядатимемо роботу із застосунком на його прикладі.

Після входу у систему, адміністратор вебзастосунка має можливість побачити весь доступний йому функціонал (рисунок 3.15).

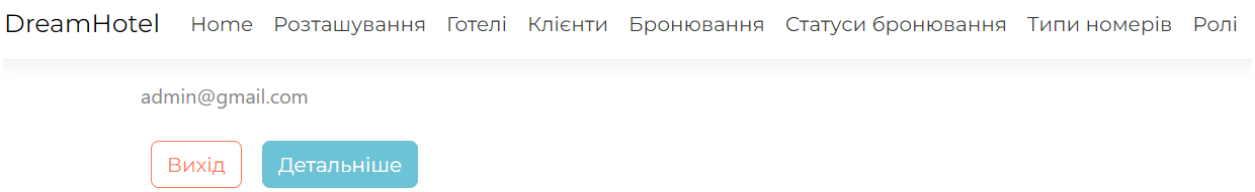

Рисунок 3.15 – Початковий вигляд інтерфейсу адміністратора

Саме адміністратор вебзастосунка має можливість надати вищий рівень доступу для користувача. Для цього потрібно перейти до вкладки «Ролі», після чого буде відображено список ролей (рисунок 3.16), потім перейти за посиланням «Список користувачів для зміни ролей», звідки (рисунок 3.17) можна для конкретного користувача змінити його доступ до даних.

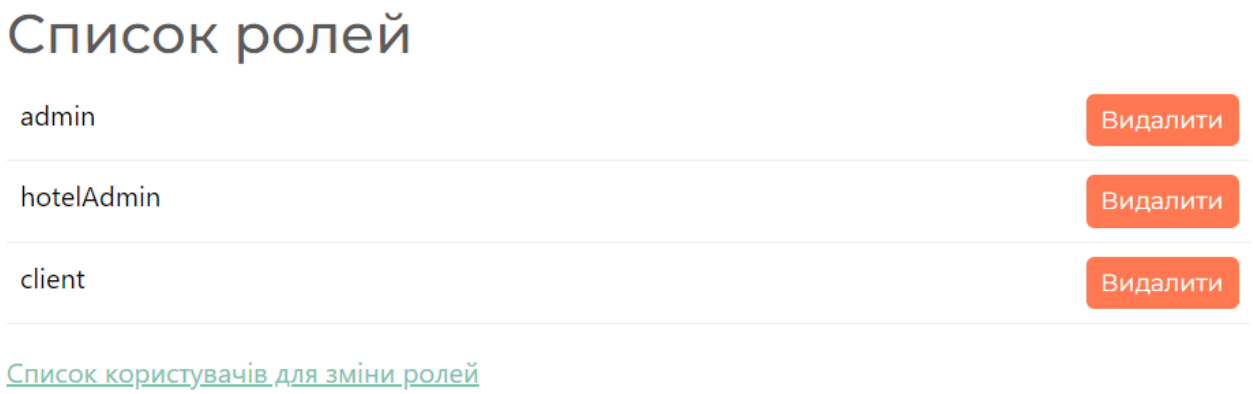

Рисунок 3.16 – Список ролей

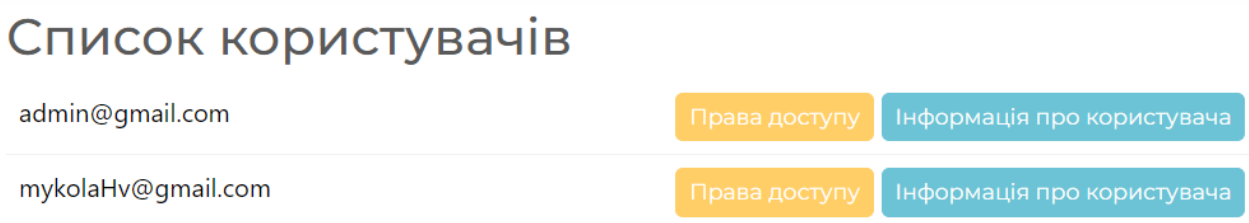

Рисунок 3.17 – Список користувачів для зміни їх ролей

Функціонал вебзастосунка було реалізовано відповідно до діаграми прецедентів, створеної у розділі 2, тому не будемо детально зупинятися на всіх можливостях системи, розглянемо все на роботі з готелями. Звернемо увагу на основу сторінку пошуку готелей для броні – список готелей із фільтрами (рисунок 3.18).

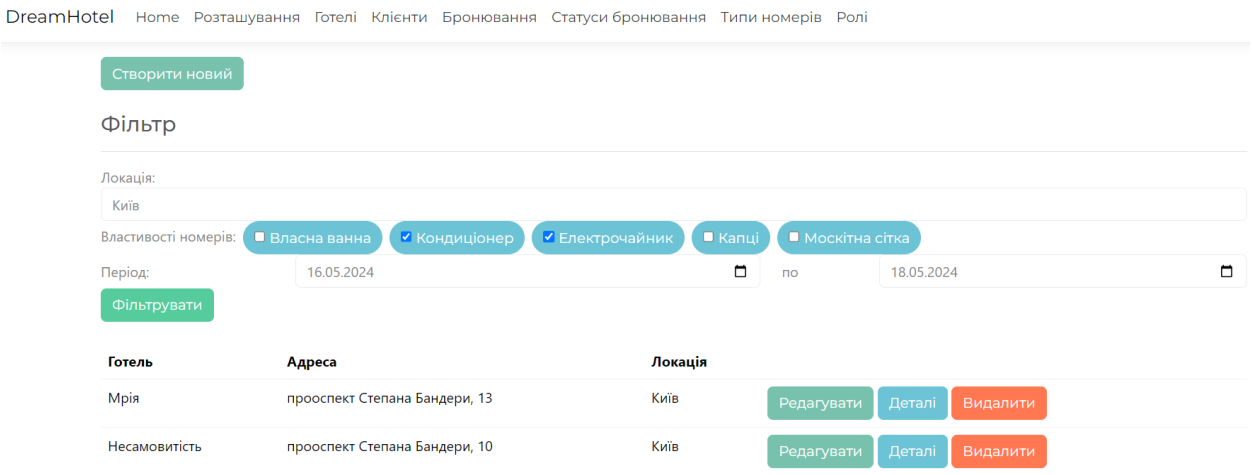

Рисунок 3.18 – Список готелей

У фільтрі є можливість вказати локацію (розташування), обрати властивості номерів, які мають бути доступні та бажаний період бронювання. Далі зі списку обираємо готель, який зацікавив та переглядаємо про нього детальнішу інформацію.

У детальній інформації про конкретний готель (рисунок 3.19) можна переглянути список номерів (реалізовано аналогічно до готелей), сервіси готелю (з можливістю перейти до їх додавання) та фото готелю, з можливістю перейти до їх додавання.

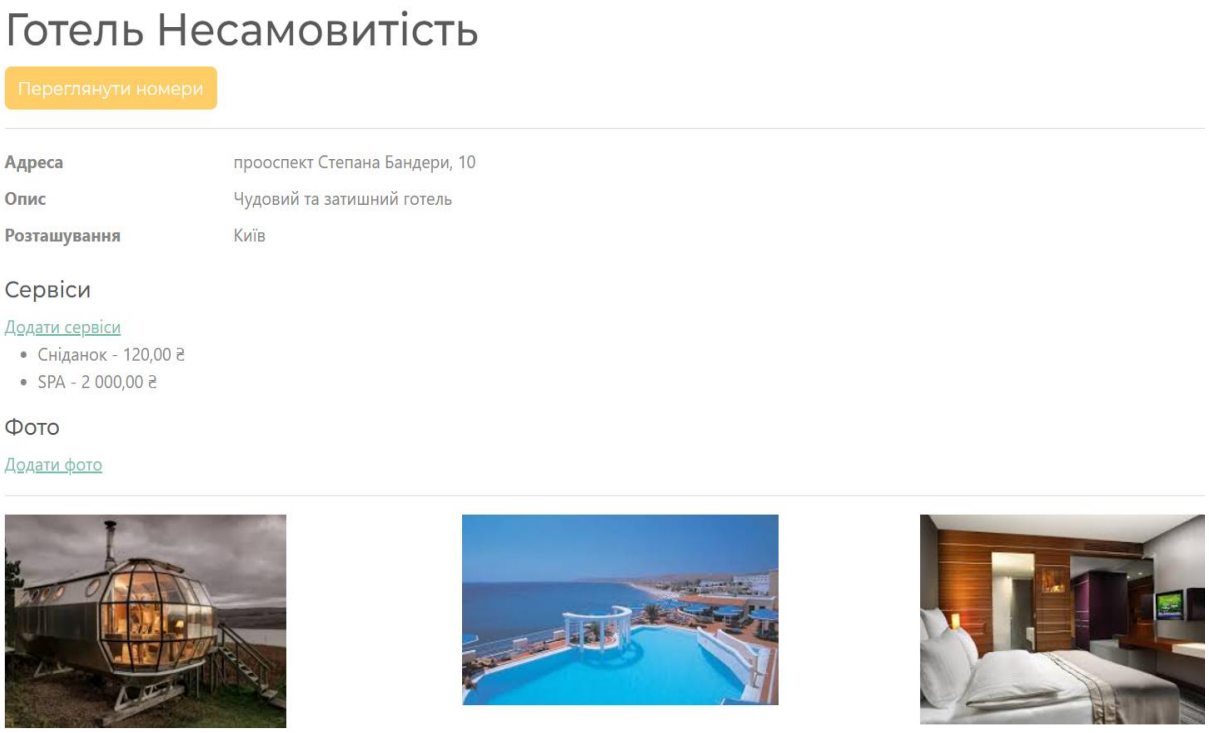

Рисунок 3.19 – Детальна інформація про готель

Створення (рисунок 3.20), редагування інформації (рисунок 3.21) та видалення (рисунок 3.22) відбувається за класичною схемою з усіма необхідними перевірками.

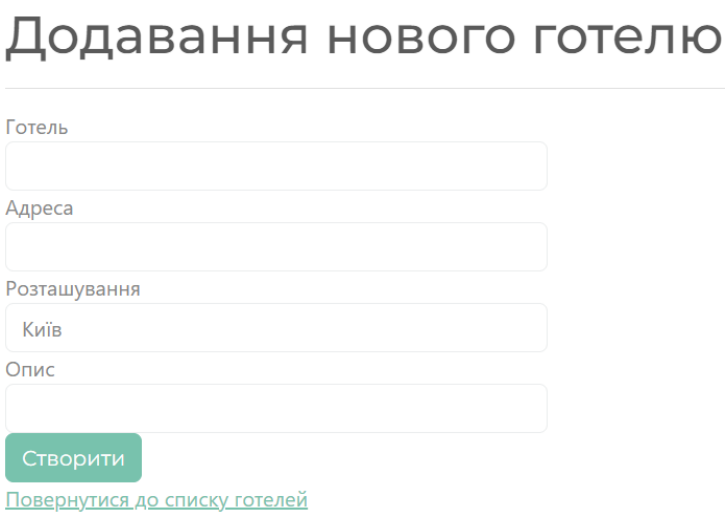

Рисунок 3.20 – Сторінка додавання нового готелю

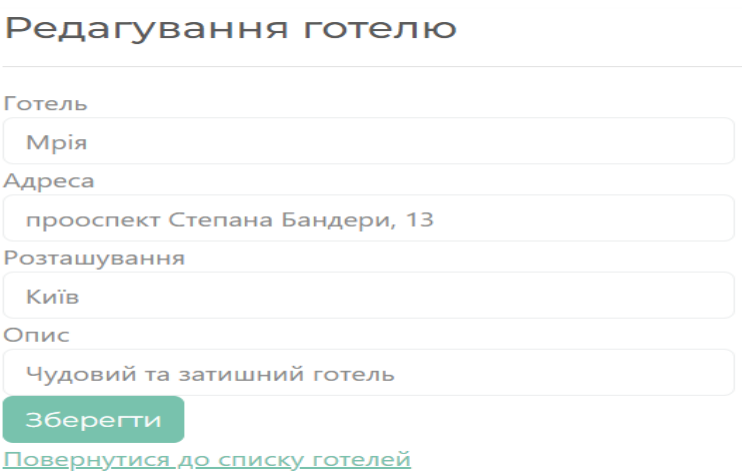

Рисунок 3.21 – Сторінка редагування готелю

Зрештою, отримали повнофункціональний вебзастосунок, який дає можливість бронювати готелі/житло або ж бути власником, який з цього буде заробляти.

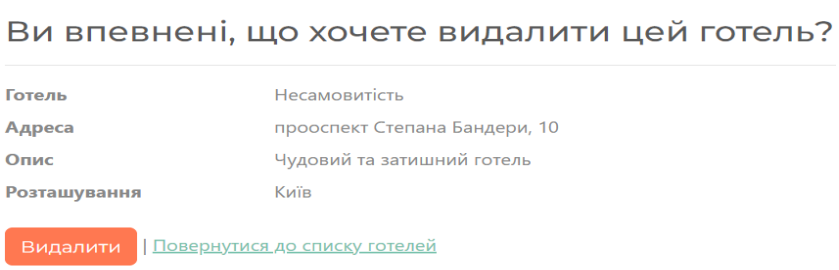

Рисунок 3.22 – Сторінка видалення готелю

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено дослідження найбільших конкурентів на ринку бронювання житла, таких як Booking.com та Airbnb. Отримані дані вказують на те, що розроблений вебзастосунок для онлайн бронювання житла є важливим та потужним інструментом, який може значно спростити процес пошуку та бронювання житла для мандрівників. Вебзастосунок також може стати корисним для власників житла, які зможуть пропонувати свої послуги широкому колу користувачів через цю платформу.

У процесі реалізації було застосовано різноманітні технології, що дозволило краще ознайомитися з ними та підвищити рівень володіння. Цей досвід може бути важливим для подальшого професійного розвитку та зростання як фахівця в області програмування та розробки вебзастосунків.

Всупереч досягнутому успіху, у розробленого вебзастосунка є потенціал для подальшого розвитку. Деякі напрямки подальшого розвитку проєкту можуть включати:

– покращення інтерфейсу: Робота над інтерфейсом з метою підвищення зручності та привабливості для кінцевих користувачів;

– створення мобільного застосунку: Розробка мобільного додатку для зручного бронювання житла через мобільні пристрої;

– розширення функціонала для власників житла: Створення модуля звітності, який дозволить власникам житла вести облік доходів та витрат, а також аналізувати прибутковість у різні періоди часу;

– монетизація проєкту: Розгляд можливостей отримання прибутку через рекламу, партнерські програми або комісійні від бронювань.

Загалом, розроблений вебзастосунок є лише початком для подальшого розвитку та вдосконалення. Потенціал для зростання та вдосконалення існує, і важливо продовжувати працювати над проєктом з метою забезпечення його успішного та стабільного функціонування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kozlovskyi, Y. (2020). Основні сфери застосування інформаційних систем і технологій у туризмі. Вісник Київського національного університету культури і мистецтв. Серія: Туризм, 3(1), 128–136. URL: https://doi.org/10.31866/2616-7603.3.1.2020.207516 (дата звернення: 01.05.2024).

2. Статистичний щорічник України за 2022 рік. Державна служба статистики України: Відповід. за випуск О.А. Вишневська. – Київ, 2023. С. 387. URL: https://ukrstat.gov.ua/druk/publicat/kat_u/2023/zb/11/year_22_u.pdf (дата звернення: 01.05.2024).

3. Історія створення та маркетингові прийоми. URL: https://bazilik.media/booking-istoriia-stvorennia-ta-marketynhovi-pryjomy/ (дата звернення: 01.05.2024).

4. Наш посібник із оренди та здачі в оренду нерухомості на Airbnb. URL: https://drealty.com.ua/nash-posibnyk-iz-orendy-ta-zdachi-v-orendu-neruhomosti-na-airbnb/ (дата звернення: 03.05.2024).

5. Sparx Systems. Conceptual Data Model. URL: https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/conceptual_data_model__.html (дата звернення: 03.05.2024).

6. Unified Modeling Language. У Вікіпедії. URL: https://en.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 06.05.2024).

7. Visual Paradigm. What is Communication Diagram?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-communication-diagram/ (дата звернення: 07.05.2024).

8. LinkedIn Learning. Software Design: Modeling with UML. URL: https://www.linkedin.com/learning/software-design-modeling-with-uml (дата звернення: 11.05.2024).

9. UML Diagrams. Use Case Diagrams. URL: https://www.uml-diagrams.org/use-case-diagrams.html (дата звернення: 11.05.2024).

10. Sourcemaking. Modeling for System Integration. URL: https://sourcemaking.com/uml/modeling-for-system-integration (дата звернення: 04.05.2024).

11. Dou.ua. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 09.05.2024).

12. Microsoft. SQL Server. URL: https://www.microsoft.com/en-us/sql-server (дата звернення: 13.05.2024).

13. Microsoft. What is SQL Server?. URL: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16 (дата звернення: 13.05.2024).

14. Microsoft. Visual Studio. URL: https://visualstudio.microsoft.com/vs/preview/ (дата звернення: 13.05.2024).

15. Вікіпедія. ASP.NET. URL: https://uk.wikipedia.org/wiki/ASP.NET (дата звернення: 14.05.2024).

16. GeeksforGeeks. Introduction to ASP.NET. URL: https://www.geeksforgeeks.org/introduction-to-asp-net/ (дата звернення: 15.05.2024).

17. Brander. MVC. URL: https://brander.ua/technologies/mvc (дата звернення: 16.05.2024).

18. Entity Framework Tutorial. What is Entity Framework?. URL: https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx (дата звернення: 16.05.2024).

19. Microsoft. NuGet. URL: https://learn.microsoft.com/en-us/nuget/ (дата звернення: 12.05.2024).

20. TekTutorialsHub. LINQ to Entities Tutorial. URL: https://www.tektutorialshub.com/linq-to-entities/linq-to-entities-tutorial/ (дата звернення: 11.05.2024).

21. Bootstrap. URL: https://getbootstrap.com/ (дата звернення: 20.05.2024).

22. Microsoft. Getting Started with ASP.NET Identity. URL: https://learn.microsoft.com/en-us/aspnet/identity/overview/getting-started/ (дата звернення: 18.05.2024).

23. А.Ю. Шафроненко, Є.В. Бодянський, Д.О. Руденко. (2023) Модифікований рекурентний метод достовірної нечіткої кластеризації з використанням оптимізаційної процедури на основі косяків риб. //Information Processing Systems, 2023, Issue 1(172), pp.92-96 URL: https://journal-hnups.com.ua/index.php/soi/article/view/1385/1253 (дата звернення: 24.05.2024).

24. Amer Abu-Jassar, Diana Rudenko, & Hitham Abdalla. (2024). Digital medical image as an object of processing and analysis. Multidisciplinary Journal of Science and Technology, 4(1), pp.28–35. URL: https://mjstjournal.com/index.php/mjst/article/view/692 (дата звернення: 24.05.2024).

25. Руденко Д.О. (2017) Метод семантичної інтеграції локально незалежних даних URL: https://openarchive.nure.ua/server/api/core/bitstreams/c8e75eab-132e-4fd6-abd0-075f69522bdf/content (дата звернення: 24.05.2024).