Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-сайту фотостудії "Yuraale"

Виконав: студент IV курсу, групи СНз-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Алексєєнко Ю.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_ _» _червня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Алексєєнко Юрій Юлійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-сайту фотостудії "Yuraale"

Керівник роботи Фриз Михайло Євгенович к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444.*

2. Термін подання студентом завершеної роботи 18 червня 2025 р.

3. Вихідні дані до роботи Дані про типи фотосесій, розцінка, приклади фотографій, записування на фотосесію, функціональність

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ. 1.1 Актуальність розробки веб-сайту фотостудії. 1.2. Аналіз аналогів сайтів комерційних фотографів. 1.3 Мета та задачі. 1.4. Технічне завдання. 1.5 Вибір платформи та підхід до реалізації. 1.6. Висновок першого розділу. РОЗДІЛ 2. ПОСТАНОВ-КА ЗАДАЧІ ПРОЕКТУ. 2.1 Вибір засобів реалізації. 2.2. Архітектурна система. 2.3. Опис бази даних. 2.3.1 Використання бази даних. 2.3.1. Використання Entity Framework Core у веб-сайту. 2.3.2. Механізм Міграції у EF Core. 2.3.3. Автентифікація та авторизація в системі. 2.4. Висновок до другого розділу. РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ФОТОСТУДІЇ "YURAALE". 3.1 Архітектура та структура веб-сайту. 3.2. Реалізація функціоналу веб-сайту 3.3 Тестування та перевірка функціональності веб-сайту. 3.3.1 Методи тестування 3.3.2 Результат тестування. 3.3.3 Тестування безпеки. 3.5 Висновок до третього розділу РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ. 4.1. Аналіз умов праці програміста. 4.2. Вимоги до організації робочого місця програміста. 4.3. Електробезпека при роботі з комп'ютерною технікою. 4.4. Загальні вимоги безпеки з охорони праці для користувачів та розробників. 4.5 Висновок до четвертого розділу. Висновок. Перелік джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Повний перелік слайдів у презентації, включаючи перший та останній слайди (з номерами, через крапку).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	доцент кафедри МТ Сенчишин Віктор Степанович	02.06.2025	05.06.2025

7. Дата видачі завдання 07 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	01.03.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	02.03.2025-09.03.2025	*Виконано*
3.	Виконання дослідження щодо аналогів, постановка	10.03.2025-20.03.2025	*Виконано*
	задачі та розроблення технічного завдання, визначення		
	функціоналу		
4.	Оформлення розділу «Аналіз предметної області та	21.03.2025-25.03.2025	*Виконано*
	постановка задачі»		
5.	Оформлення розділу «Проектна частина»	26.03.2025-03.04.2025	*Виконано*
6.	Оформлення розділу «Практична реалізація»	04.04.2025–12.04.2025	*Виконано*
7.	Виконання підрозділу «Безпека життєдіяльності»	13.04.2025–15.04.2025	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	16.04.2025–18.04.2025	*Виконано*
	праці»		
9.	Оформлення повного тексту кваліфікаційної роботи	19.04.2025–24.04.2025	*Виконано*
10.	Нормоконтроль	11.06.2025-13.06.2025	*Виконано*
11.	Перевірка на плагіат	16.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2025	

Студент _____ Алексєєнко Ю.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема: Розробка веб-сайту фотостудії "Yuraale" // Кваліфікаційна робота освітнього рівня «Бакалавр» // Алексєєнко Юрій Юлійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНз-41 // Тернопіль, 2025 // С. 81, рис. 15, табл. 1, кресл. 1, додат. 4, бібліогр. 21.

Ключові слова: веб-сайт, фотостудія, ASP.NET, MVC, база даних, адміністрування, інтерфейс, бронювання.

Кваліфікаційна робота присвячена розробці веб-сайту фотостудії "Yuraale" з метою автоматизації процесів представлення портфоліо, бронювання фотосесій, оплати послуг та управління контентом.

У першому розділі описано аналіз предметної області, розглянуто сучасні інструменти розробки та визначено вимоги до сайту.

У другому розділі представлено архітектуру веб-сайту, UML-діаграми, моделі та структуру бази даних.

У третьому розділі реалізовано функціональні можливості як для клієнтів, так і для адміністратора фотостудії.

Об'єкт дослідження: процес автоматизації діяльності фотостудії.

Предмет дослідження: веб-сайт для керування фотостудією.

ANNOTATION

Subject: Development of the "Yuraale" Photo Studio Website// Qualification work for the Bachelor's Degree // Alekseyenko Yurii Yuliiovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, Group SNz-41 // Ternopil, 2025 // P. 81, fig. 15, tabl. 1, chart. 1, app. 4, bibl. 21.

Keywords: website, photo studio, ASP.NET, MVC, database, administration, interface, booking.

The qualification paper is dedicated to the development of a website for the photo studio “Yuraale” with the aim of automating processes such as portfolio presentation, photo session booking, service payment, and content management.

The first section presents an analysis of the subject area, reviews modern development tools, and outlines the website requirements.

The second section describes the web application architecture, UML diagrams, models, and database structure.

The third section implements the functional capabilities for both clients and the administrator of the photo studio.

Object of research: automation of photo studio operations.

Subject of research: web application for photo studio management.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1. **Помилка! Закладку не визначено.**1.1.

91.2.

101.3.

141.4.

161.5.

18РОЗДІЛ 2.

182.1.

192.2.

222.3.

242.3.1

272.3.2

282.3.3

302.4.

31РОЗДІЛ 3.

Помилка! Закладку не визначено.3.1.

323.2.

333.3.

373.3.1

Методи тестування 35

3.3.2 Результати тестування 36

3.3.3 Тестування безпеки 36

3.4. 403.4.1 40

3.4.2 413.4.3 43

3.4.4 443.5. 48

РОЗДІЛ

4. **Помилка! Закладку не**

визначено.4.1. **Помилка! Закладку не визначено.**4.2. 53
4.3. 544.4. 55
4.5. 57ВИСНОВКИ 56
ПЕРЕЛІК ДЖЕРЕ 58
ДОДАТКИ 60

ВСТУП

Актуальність теми. Сучасний цифровий світ вимагає від бізнесу ефективної присутності в Інтернеті. Особливо це стосується творчих професій, таких як фотографія. Наявність якісного веб-сайту дозволяє фотостудіям демонструвати портфоліо, приймати замовлення онлайн, комунікувати з клієнтами та здійснювати оплату послуг. Це значно спрощує адміністративну роботу та підвищує рівень обслуговування. У зв’язку з цим, розробка веб-сайту для фотостудії "Yuraale" є актуальним завданням, яке дозволить не лише автоматизувати бізнес-процеси, а й підвищити конкурентоспроможність студії на ринку.

Мета і задачі дослідження. Метою кваліфікаційної роботи є розробка веб-сайту для фотостудії з функціоналом бронювання послуг, перегляду портфоліо, зворотного зв’язку та адміністративного управління. Для досягнення поставленої мети необхідно вирішити такі задачі:

- дослідити предметну область та сучасні рішення;
- розробити структуру та архітектуру веб-сайту;
- реалізувати функціональність з використанням технологій ASP.NET MVC;
- забезпечити базовий рівень захисту інформації;
- протестувати працездатність і зручність використання сайту;

Практичне значення одержаних результатів. Результати роботи можуть бути впроваджені у фотостудії для автоматизації основних бізнес-процесів: запису клієнтів, управління знімками, виставлення цін та прийому оплати. Реалізоване рішення легко адаптується для інших сфер діяльності з подібними функціональними вимогами.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ.

1.1. Актуальність розробки веб-сайту фотостудії

У сучасному цифровому світі мати власний сайт – це вже не розкіш, а необхідність, особливо для бізнесу у сфері послуг. Фотостудії – не виняток. Це творчий бізнес, який як ніхто інший потребує зручного способу показати свої роботи, знайти нових клієнтів і налагодити з ними зручний зв'язок.

Більшість людей, обираючи фотостудію, насамперед звертають увагу на портфоліо. Завдяки сучасним веб-технологіям його можна красиво і зручно представити онлайн. Крім того, можливість забронювати зйомку, оплатити послуги або просто написати через сайт – це великий плюс як для клієнтів, так і для самої студії. Це значно спрощує організацію і зменшує навантаження на адміністратора.

Сьогодні існує безліч сучасних фреймворків, наприклад ASP.NET MVC, які дозволяють створювати зручні, надійні та масштабовані сайти [8]. Це відкриває нові можливості для малого й середнього бізнесу – зокрема для фотостудій, які хочуть автоматизувати свою роботу та зробити сервіс ще кращим.

Сайт весільного фотографа Hilario Photography – чудовий приклад того, як можна стильно та ефективно подати себе онлайн. Весь акцент зроблено на емоційних, живих фотографіях, які відкриваються на весь екран і відразу передають атмосферу зйомки. Це дозволяє глядачеві відчути індивідуальність і почерк фотографа. Дизайн сайту – простий, мінімалістичний і дуже зручний. Усе потрібне – портфоліо, інформація про послуги, ціни й контакти – знаходиться буквально в кілька кліків. На рисунку 1.1 видно головну сторінку сайту: вона демонструє, як можна красиво подати фотоконтент, зробити навігацію зрозумілою та водночас не забути про зручність для користувача.

Рисунок 1.1 – Головна сторінка Hilario Photography

Таким чином, подібні приклади демонструють важливість якісного візуального представлення послуг у сфері фотографії. Веб-сайт фотостудії виступає не лише візитною карткою, але й функціональним інструментом для автоматизації взаємодії з клієнтом, формування довіри та підвищення впізнаваємість бренду.

1.2. Аналіз аналогів сайтів комерційних фотографів

Проєктування будь-якого веб-сайту має розпочинатися із визначення чіткої мети, конкретних задач і очікуваного результату, адже саме це забезпечує логічну й послідовну структуру майбутньої системи. Крім того, доцільно здійснити критичний огляд подібних рішень, уже реалізованих у цій сфері. Це дозволяє зрозуміти, на яких аспектах варто зосередити увагу при розробці, а від яких функціональних складових – за потреби – можна цілком відмовитися без шкоди для зручності та ефективності ресурсу. Задля реалізації поставленої мети в межах проєкту було проведено аналіз веб-ресурсів, створених для фотостудій та професійних фотографів. Такий підхід дозволив врахувати практичні рішення,

які вже застосовуються у цій сфері, а також сформувати уявлення про актуальні вимоги до подібних сайтів.

Перший аналог – сайт весільного фотографа «Hilario Photography» [4]. «Головна» сторінка на рисунку 1.1

Другий аналог – фотостудії «PASSAGE STUDIO», м Львів [5]. «Головна» сторінка на рисунку 1.2

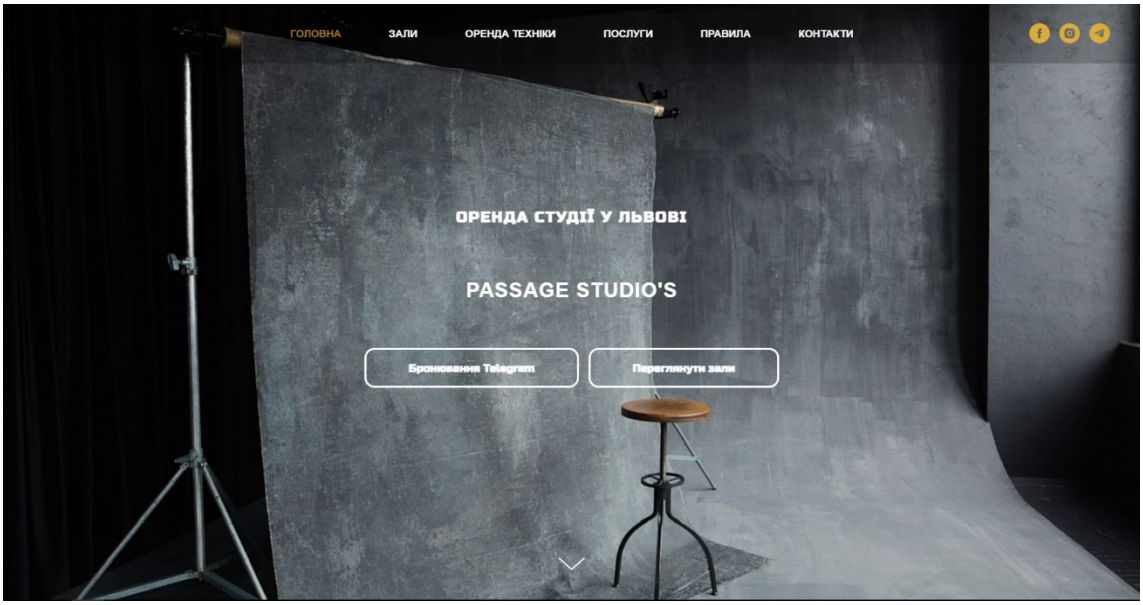


Рисунок 1.2 – Сторінка фотостудії «PASSAGE STUDIO»

Третій аналог – фотостудія «Pudra», м. Київ [6]. «Головна» сторінка на рисунку 1.3

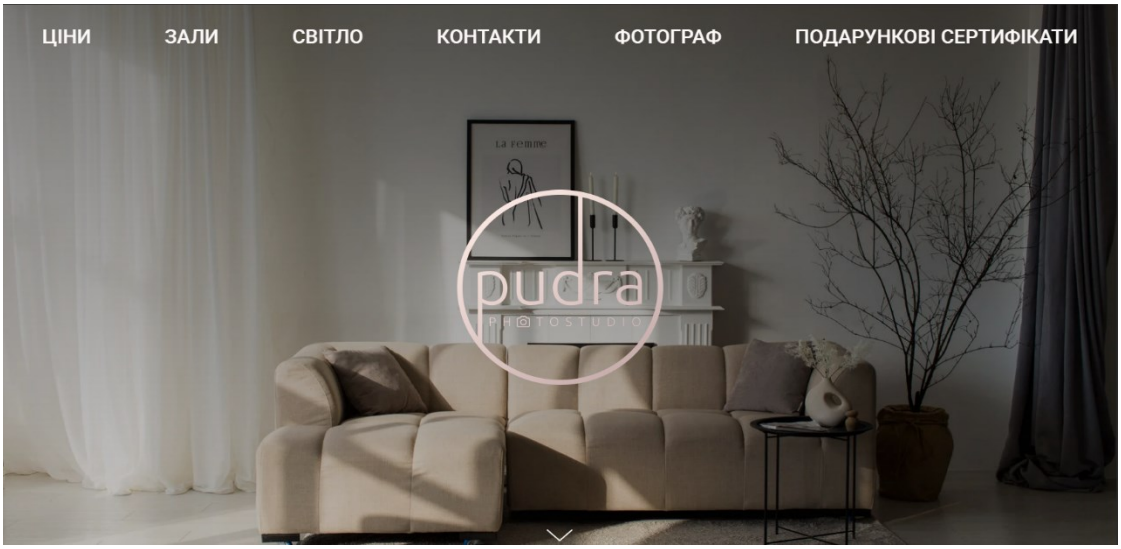


Рисунок 1.3 – Сторінка фотостудії «Pudra»

Четвертий аналог – фотостудія «W Photostudio», м. Київ [7]. «Головна» сторінка на рисунку 1.4

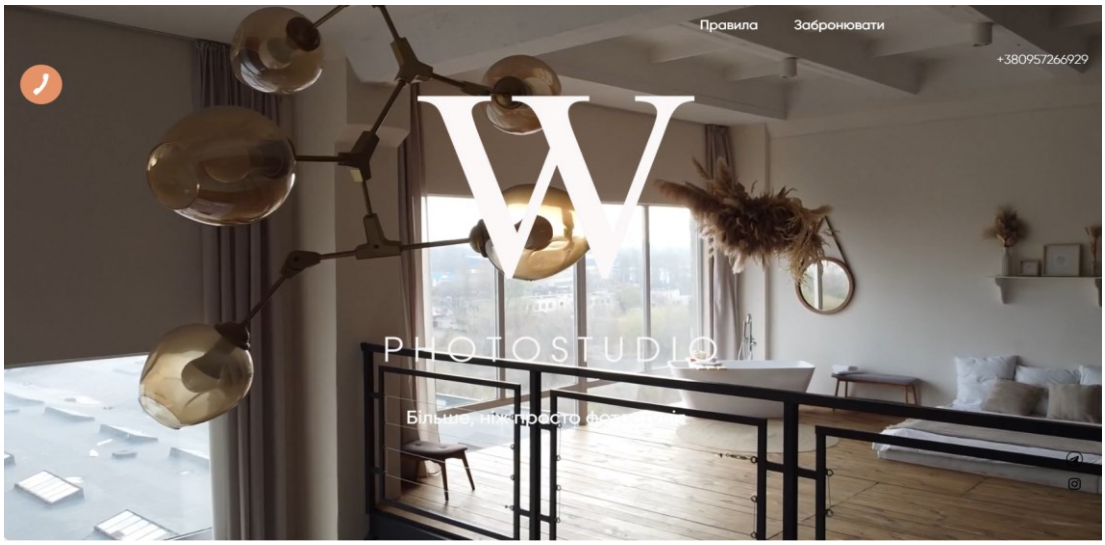


Рисунок 1.4 – Сторінка фотостудії «W Photostudio»

Проаналізувавши перелік сайті фотостудій, було виявлено позитивні та негативні аспекти їх реалізацій. Визначення сильних та слабких сторін допомогло сформувати загальне уявлення про тенденції та помилки, яких варто уникати під час власної розробки веб-сайту.

З недоліків було виявлено:

- Недбале оформлення текстового контенту.
- Велика кількість реклами яка може відволікати користувача та бути зовсім не доречна.
- Проблеми відображення певних сторінок.
- Відсутність функції збільшення для детального перегляду зображень.
- Ускладнений алгоритм оформлення послуг.

З позитивних було виявлено:

- Зручність навігації по веб-сайту.
- Підібрана палітра кольорів сайту.
- Інформаційно та візуально привабливість сторінки.
- Можливість зворотного зв’язку.

Порівняльна характеристика сайтів аналогів представлена в таблиці 1.1

Таблиця 1.1 – Аналіз розглянутих комерційних web-сайтів фотографів

Критерії	**Web-сайт «Hilario Photography»**	**Web-сайт «PASSAGE STUDIO»**	**Web-сайт «Pudra»**	**Web-сайт «W Photostudio»**
Відсутність непотрібної реклами	+	+	+	+
Розподіл по видам фотографії	+	-	-	+
Можливість зворотного зв'язку	+	+	+	+
Зручність	+	+	-	-
Навігація	+	+	-	+
Кольорова гамма	+	+	-	+
Збільшення фото	+	+	+	+
Видима цінова політика	+	+	+	+

На основі виявлених недоліків проаналізованих веб-сайтах було сформовано ключові вимоги до нашого веб-сайту, реалізація яких дозволить суттєво підвищити якість користувацького досвіду:

- виключення нав’язливої та несуттєвої реклами з акцентом лише на релевантні елементи (у разі її наявності);
- використання виключно якісного візуального та текстового контенту;
- регулярне оновлення інформаційних блоків та актуалізація даних;

- узгодженість кольорової палітри із загальною стилістикою сайту для формування естетичного інтерфейсу;
- впровадження інструментів для адміністрування контенту (веб-адміністратор);
- забезпечення високої швидкодії сайту, зокрема швидкого переходу між сторінками та простого оформлення замовлення;

Враховуючі зазначені аспекти дуже сильно полегшується проєктування власного веб-сайту. Його структура та функціональність орієнтовані на усунення недоліків, виявлених у процесі аналізу веб-сайтів аналогів, з метою створення конкурентоспроможного, сучасного та зручного інструменту для взаємодії з клієнтами фотостудії.

1.3. Мета та задачі роботи

Основною ідеєю створення веб-ресурсу стало забезпечення зручного способу донесення інформації до потенційних клієнтів, а також популяризація товарів і послуг, які пропонує фотостудія. Передбачалося не лише представлення творчого портфоліо.

З огляду на те, що значна частина цільової аудиторії активно використовує інтернет у повсякденному житті, було здійснено попередню аналітику ринку. Це дозволило прийняти обґрунтоване рішення щодо необхідності створення власного веб-сайту, що сприятиме розвитку бренду, покращенню комунікації з клієнтами та швидшому залученню нових замовлень. Важливим фактором є також підвищення рівня сервісу, як з точки зору оперативності виконання послуг, так і якості виготовлення продукції.

У результаті аналізу сайтів інших фотостудій було зроблено висновок, що оптимальним рішенням для реалізації даного проекту стане використання зручної та інтуїтивно зрозумілої платформи для керування контентом.

У перспективі користувачі зможуть не лише ознайомитися з інформацією про фотостудію, її команду, перелік послуг і фотозон, але й переглядати приклади робіт, оформляти замовлення та зв’язуватись із адміністратором безпосередньо через сайт. Такий підхід значно спрощує шлях клієнта до замовлення, одночасно підвищуючи впізнаваність бренду у цифровому середовищі.

На основі попереднього аналізу предметної області, а також вивчення типових функцій аналогічних веб-сайтів, було сформовано технічне завдання на розробку веб-сайту фотостудії «Yuraale». Розробка здійснюється як засіб для автоматизації ключових бізнес-процесів фотостудії та для створення сучасного представлення її послуг у цифровому середовищі. Головною метою створення сайту є реалізація інструменту, що дозволить користувачам зручно переглядати портфоліо, ознайомлюватися з описами фотосесій, цінами та доступними датами, а також оформлювати замовлення без необхідності телефонного зв’язку або особистої присутності. Крім того, система повинна надати адміністратору повноцінні можливості для редагування вмісту сайту, перегляду замовлень та керування інформацією без залучення розробників.

Інтерфейс сайту має бути простим для сприйняття користувачем, зручним для користування та добрий вигляд на будь якому пристрою – від телефона до комп’ютера. На основних сторінках публічної частини повинна бути представлена загальна інформація про фотостудію, фотогалереєю з прикладами роботи, опис послуг з цінами, форми для бронювання, а також контактні дані з можливістю написати або подзвонити. Для адміністраторів передбачено окремий розділ який називається адмін-панель, доступ до якого відкривається після входу в систему. Там є можливість керування галереями, змінювати список послуг, слідкувати за замовленнями – з можливістю їх фільтрування, редагування або перегляду деталей. Також є можливість змінювати текст на сайті, що дозволяє швидко оновлювати інформацію без потреби в технічних знаннях.

Розробка сайту буде здійснюватися на основі мови програмування C# із використанням фреймворку ASP.NET MVC. Для зберігання та обробки даних передбачено використання СУБД Microsoft SQL Server, з якою веб-сайту взаємодіятиме через ORM-технологію Entity Framework Core. Інтерфейс користувача реалізовуватиметься із застосуванням Razor Pages і фреймворку Bootstrap для забезпечення адаптивності та відповідності вимогам UX/UI.

У ході реалізації особливу увагу буде приділено захисту даних, обмеженню доступу до адміністративних функцій, перевірці правильності введення даних та забезпеченню стабільної роботи системи за умов звичайного навантаження. Система повинна бути масштабованою, підтримувати розширення функціональності та можливість перенесення на зовнішній хостинг у майбутньому.

1.4. Вибір платформи та підхід до реалізації

Під час попереднього етапу було розглянуто декілька варіантів реалізації веб-сайту для фотостудії, зокрема з використанням професійного фреймворку ASP.NET. Основними критеріями для прийняття рішення були: можливість розширення функціоналу, контроль над архітектурою, гнучкість, безпека, а також досвід у роботі з відповідними технологіями що дає змогу в швидкому виправленню помилок та переписанню функціоналу й логіку.

Як ми раніше й визначились для розробки ми візьмемо фреймворк ASP.NET MVC та дуже зручні технології цього фрейму. Його перевагами є чіткий розподіл логіки на компоненти (Model-View-Controller), надійна система авторизації, інтеграція з Entity Framework для роботи з базами даних, а також висока продуктивність і підтримка від Microsoft.

Цей підхід дозволяє забезпечити гнучке масштабування, простоту обслуговування та адаптацію інтерфейсу під потреби кінцевого користувача. Крім того, обрана платформа дає змогу реалізувати кастомні сценарії взаємодії з

клієнтом, що є складно реалізованим у конструкторах сайтів без додаткових витрат.

1.5. Висновок до першого розділу

У цьому розділі було проведено всебічне дослідження предметної області. Розглянуто причини, чому створення веб-сайту для фотостудії є актуальним у сучасних цифрових умовах. Сформульовано основну мету роботи та визначено ключові завдання, які необхідно вирішити в межах проекту. Також було проаналізовано існуючі аналоги – сайти зі схожим функціоналом – і на основі їх сильних та слабких сторін сформовано вимоги до майбутнього веб-ресурсу як у частині функціоналу, так і з погляду зручності та дизайну.

На основі цього аналізу було складено технічне завдання, яке чітко окреслює структуру майбутнього сайту, принципи його побудови, а також враховує інтереси як кінцевих користувачів, так і співробітників фотостудії. У документі подано обґрунтований вибір технологій, інструментів розробки та платформи. Окрім цього, сформульовано функціональні, якісні та обмежувальні вимоги до системи. Усе це створює необхідне підґрунтя для подальшого етапу роботи – проектування архітектури веб-сайту та побудови структурної моделі системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-САЙТУ ФОТО СТУДІЇ

2.1. Вибір засобів реалізації

На цьому етапі було підібрано найбільш доцільні інструменти, мови програмування, фреймворки та технології, які будуть використані для створення веб-сайту. Вибір здійснюється з урахуванням вимог, визначених у технічному завданні, а також з огляду на можливості сучасних платформ розробки, що добре зарекомендували себе в аналогічних проектах.

В якості основної технологічної основи було обрано фреймворк ASP.NET MVC. Його перевага полягає в тому, що він підтримує принцип розділення логіки на моделі, представлення та контролери, що в результаті сприяє зручному обслуговуванню коду, гнучкому масштабуванню проекту та структурованому підходу до побудови системи. Крім того, ASP.NET MVC добре інтегрується з іншими продуктами Microsoft, зокрема з SQL Server, який буде використовуватись як основна база даних.

Архітектура майбутнього веб-сайту базується на шаблоні Model-View-Controller (MVC), який забезпечує чітке розділення обов’язків між логікою обробки даних, їхнім відображенням і реакцією на дії користувача [15]. Модель (Model) це частина яка відповідає роботі з даними. Представлення (View) формує інтерфейс, а саме представлення методів з контролера, яке можна стилізувати за допомогою файлів стилю, та структуван. Контролер (Controller) є посередником між користувачем і моделлю, він обробляє запити, викликає необхідні методи моделі та передає результат до представлення.

На рисунку 2.1 зображено схему Model-View-Controller.

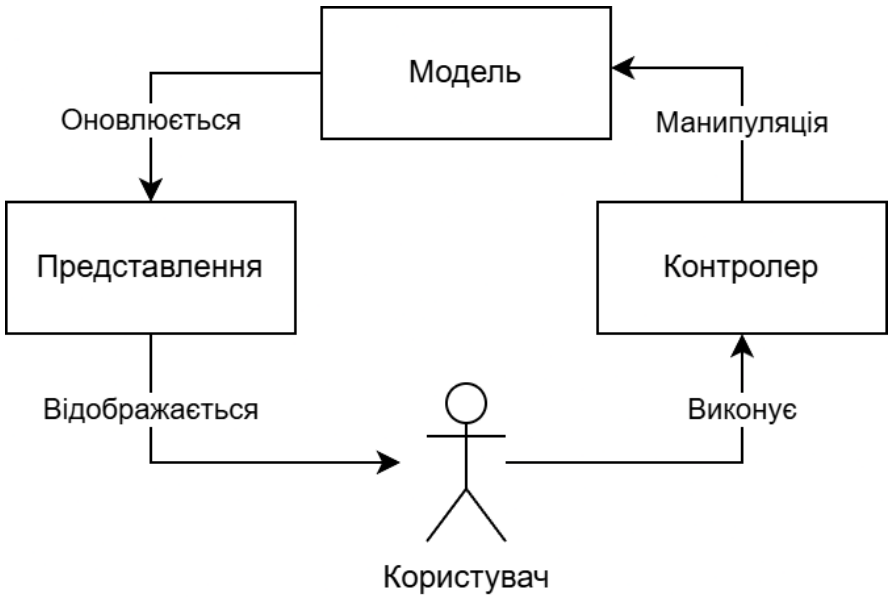


Рисунок 2.1 – Схема роботи архітектурного шаблону

Основною мовою програмування для реалізації проекту було обрано C#, яка є базовою для платформи.NET і дозволяє реалізувати як серверну логіку, так і взаємодію з ORM – Entity Framework Core. Подібне рішення може забезпечити надійність і безпеку при роботі з даними, знижує ймовірність SQL-ін’єкцій, а також спрощує роботу з міграціями та збереженням структур бази даних що полегшує роботу з базою даних, особливо якщо до роботи прийде розробник який не знайомий з синтаксису SQL запитів. Структура бази даних передбачає створення зв’язаних таблиць для зберігання інформації про користувачів, послуги, замовлення, фотоальбоми та повідомлення.

Для побудови інтерфейсу користувача застосовуватиметься Razor Pages – це шаблонна система, що дозволяє створювати динамічні веб-сторінки із вбудованими конструкціями C#. Візуальне оформлення сайту базуватиметься на фреймі Bootstrap, який якраз дає змогу розробити адаптивність до різних форматів моніторів (телефонів, комп’ютерів, планшетів) і дозволяє швидко створити сучасний, зручний та в водночас привабливий інтерфейс що в цьому році є важливою частиною для веб-сайту. Додатково передбачається використання JavaScript для реалізації динамічної поведінки елементів. У разі потреби буде використано jQuery як легкий інструмент для швидкої реалізації

інтерактивних компонентів, та AJAX для функціоналу оновлення інформації та відпрацювання форму чи методів без перезавантаження веб-сторінки.

Особливу увагу в межах розробки буде приділено впровадженню окремих сервісів для зручності користування веб-сайтом як з боку клієнта так і з боку адміністратора. Зокрема, готується реалізація інтеграція з платіжним сервісом для можливості сплати онлайн, та сервісом пошти SMTP, в подальшому можна ввести інтеграцію з календарем гугл для відображення інтерактивно зайнятий час для запобіганню бронюванню на один час двох клієнтів, також інтеграція з сервісами для зберігання файлів, цей момент буде корисний тим що якщо треба людині завантажити матеріал після фотосесії ми знизимо навантаження на наш сайт. Як раніше зазначалось був налаштований сервіс SMTP, що дозволить клієнтам отримувати підтвердження замовлень, а адміністрації – повідомлення про нові заявки. Вибравши повний стек технологій забезпечує баланс між простотою реалізації продуктивності та можливість розширення функціоналу для майбутньої доробки сайту майбутньому та фіксу багів при їх виявленні, що є критично важливим у контексті розвитку веб-сайту.

Визначившись з правильними вимогами до подальшого розвитку проекту можна буде реалізовувати будь який функціонал від простих функцій на сторінках до інтеграції масштабних проектів вимоги до подальшого, а саме обрані засоби реалізації дозволяють у майбутньому розширити функціонал сайту без кардинальних змін у структурі що дуже зіграє для нас роль. Зокрема подібний підхід дає велику змогу над інтеграцією CRM-системами для автоматизованого обліку клієнтів, підключення push-сповіщень для оперативного інформування користувачів, а також реалізація та до опрацювання особистого кабінету користувача для перегляду історії замовлень, статусу обробки, повідомлень та оплати.

У разі подальшого зростання кількості відвідувачів передбачається можливість перенесення застосунку на хмарну платформу,що забезпечить кращу доступність, масштабованість та відмовостійкість. Такий підхід дозволить

фотостудії ефективно адаптуватися до зростаючих вимог і забезпечити якісний сервіс для клієнтів на будь-якому етапі розвитку бізнесу.

2.2. Проектування архітектури веб-сайту

Архітектура веб-сайту фотостудії «Yuraale» розроблена з урахуванням вимог до масштабування проекту в подальшому, модульності та зручності підтримки, та легка передача проекту від розробника до розробника. Система побудована на основі модельного шаблону Model-View-Controller, що забезпечує розділення функціоналу між окремими компонентами. Це дозволяє ефективно організувати процес розробки, спростити тестування окремих частин програми та забезпечити можливість подальшого розширення.

Загальна структура веб-сайту передбачає на кілька основних логічних рівнів:

- Клієнтський рівень (Front-end) – відповідає за взаємодію з користувачем. Тут реалізовано інтерфейс, створений за допомогою Razor Pages, HTML, CSS та JavaScript [10].

- Рівень контролерів (Controller) – обробляє запити від клієнта, визначає, яку модель потрібно використати, і яке представлення слід повернути у відповідь. Контролери є посередниками між інтерфейсом користувача і бізнес-логікою. Простими словами Back-end частина нашого проекту.

- Бізнес-рівень (Model / Services) – містить бізнес-логіку програми, реалізовану у вигляді моделей, сервісів та обробників. Данні моделі нашого проекту будуть мігруватись до бази. У цьому рівні виконуються операції з даними, перевірки, обчислення, фільтрація та взаємодія із зовнішніми сервісами (наприклад, оплата або надсилання листів).

- Рівень доступу до даних (Data Access Layer) – реалізовано з використанням ORM Entity Framework Core. Він дозволяє абстрагуватися від SQL-запитів, забезпечуючи зручну роботу з базою даних у вигляді об'єктів.

- База даних (Database) – зберігає інформацію про користувачів, замовлення, послуги, фотографії, ролі тощо. Всі дані структуровано згідно з нормалізованою схемою, а зв’язки між таблицями реалізовано за допомогою зовнішніх ключів.

- У разі реалізації додаткових можливостей, таких як інтеграція з API платіжних систем або систем масової розсилки, архітектура дозволяє підключати зовнішні сервіси через окремі модулі без порушення основної структури застосунку.

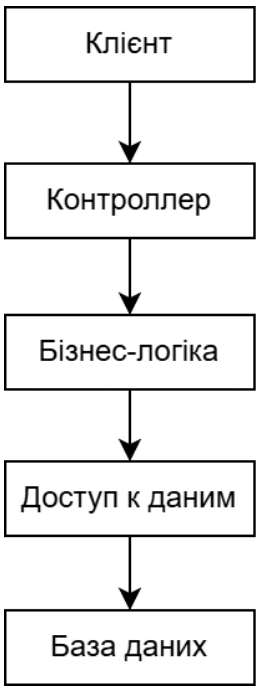


Рисунок 2.2 – Логічна схема архітектури веб-сайту

На рисунку 2.2 наведено загальну логічну схему архітектури веб-сайту, що демонструє взаємозв’язок між його основними компонентами.

2.3. Проектування бази даних

Основою для збереження структурованої інформації у веб-сайту є реляційна база даних, побудована на платформі Microsoft SQL Server із використанням технології Entity Framework Core для мігрування та синхронізації нашої бази та моделей в проекті [16]. У базі даних передбачено декілька логічно пов’язаних таблиць, які відповідають основним сутностям системи.

Серед ключових таблиць можна виділити наступні:

- AspNetUsers – містить облікові записи користувачів, включаючи ім’я, електронну пошту, стан підтвердження, хеш пароля, налаштування двофакторної автентифікації, блокування облікового запису та інші системні атрибути. Пов’язана з таблицями ролей, токенів, логінів і прав доступу.
- Bookings – зберігає інформацію про замовлення клієнтів: дату, контактні дані, кількість людей, обраний пакет послуг, загальну вартість, а також статус та інформацію про користувача, який зробив бронювання.
- Briefs – деталізує запит клієнта на фотозйомку: стиль зйомки, побажання щодо локації, наявність візажиста, референси, примітки тощо. Кожен brief прив’язаний до конкретного бронювання.
- Services – таблиця з описом послуг фотостудії, що включає назву, опис, ціну та доступність.
- BriefServices – проміжна таблиця для реалізації зв’язку "багато до багатьох" між Briefs та Services, тобто вказує, які послуги обрані в рамках кожного brief'a.
- BookingsStatus – довідник статусів бронювання (наприклад, «Очікує», «Підтверджено», «Скасовано»), використовується у зв’язку з таблицею Bookings.

- UploadedPhotos – галерея завантажених фотографій, прив’язана до бронювання та до певної категорії (Galleries). Містить шлях до файлу та дату завантаження.

- Galleries – містить категорії фотографій (альбоми), які можуть бути відображені на головній сторінці портфоліо.

- HomePageContents – зберігає вміст головної сторінки: заголовок, підзаголовок, текст, контактні дані, шлях до банера тощо.

- AspNetRoles, AspNetUserRoles, AspNetUserClaims, AspNetUserLogins, AspNetUserTokens, AspNetRoleClaims – набір таблиць, які реалізують систему автентифікації та авторизації на основі ролей (ASP.NET Identity). Вони дозволяють розмежовувати доступ до функцій системи залежно від ролі користувача.

- __EFMigrationsHistory – службова таблиця, яку автоматично створює Entity Framework для відстеження застосованих міграцій.

Зв’язки між таблицями реалізовано за допомогою зовнішніх ключів, що дозволяє забезпечити цілісність даних та підтримку складених запитів. Вся структура відповідає принципам нормалізації, що зменшує дублювання інформації та покращує ефективність роботи з базою.

На рисунку 2.3 представлено ER-діаграму бази даних, яка демонструє основні сутності системи та взаємозв’язки між ними.

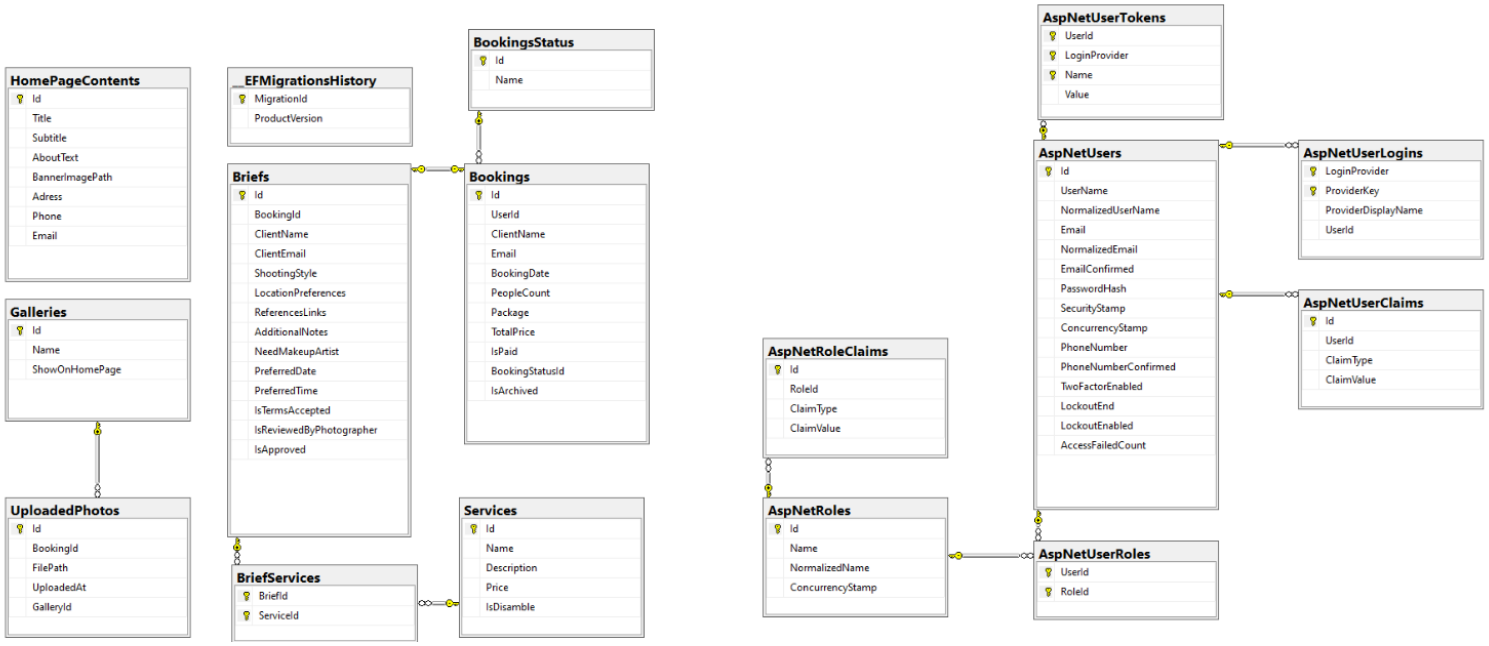


Рисунок 2.3 – ER-модель бази даних

У цій базі даних реалізовано повний процес обробки замовлення – від моменту бронювання клієнтом до створення докладного технічного завдання (Brief) і подальшої публікації результатів фотосесії в онлайн-галереї. Структура БД побудована таким чином, щоб усі об’єкти були зв’язані між собою через первинні та зовнішні ключі. Це дозволяє ефективно виконувати складні запити і забезпечує надійну роботу транзакцій.

Окрема таблиця BriefServices виконує роль проміжної ланки між технічними завданнями та переліком додаткових послуг. Вона дозволяє створювати унікальні набори послуг для кожного клієнта, реалізуючи принцип зв’язку "багато до багатьох". Такий підхід дає змогу легко оновлювати або доповнювати модель без потреби в повній перебудові бази.

Уся система побудована з урахуванням принципів нормалізації до третьої нормальної форми (3NF), що мінімізує надлишковість даних, знижує ризик аномалій при оновленні та забезпечує стабільну логіку відношень.

Блок ASP.NET Identity, представлений таблицями AspNetUsers, AspNetRoles, AspNetUserRoles тощо, інтегрується в загальну модель даних,

дозволяючи розмежовувати рівні доступу до функціональних частин системи – наприклад, розділення прав між адміністраторами, клієнтами та фотографами.

Особливу увагу приділено таблиці HomePageContents, яка дозволяє змінювати основний вміст головної сторінки без втручання у програмний код. Це забезпечує просте адміністрування контенту та є підґрунтям для можливого впровадження CMS-модуля у подальших версіях продукту.

Галереї (Galleries) дозволяють створювати категорії фотографій, які можуть бути представлені на вебсайті як портфоліо. Зв’язок із таблицею UploadedPhotos дає змогу динамічно оновлювати склад зображень у кожній категорії, а також забезпечує багатосторінкову навігацію. Для відстеження застосованих змін у структурі БД використовується таблиця __EFMigrationsHistory, яка створюється автоматично системою Entity Framework Core під час застосування міграцій. Це дозволяє синхронізувати схему БД з об'єктною моделлю додатку, спрощуючи оновлення та розгортання нових версій.

Реалізована структура дає змогу легко масштабувати проєкт, наприклад:

- додати таблиці для історії оплат,
- зберігати лог активності користувачів,
- підтримати мультимовність контенту (через окрему таблицю для мовних ресурсів).

Таким чином, розроблена модель даних повністю відповідає вимогам поставленої задачі, є розширюваною, підтримує сучасні підходи до побудови веб-сайтів і забезпечує надійність та гнучкість системи.

2.3.1 Використання Entity Framework Core у веб-сайту

Для роботи з базою даних у веб-сайту використовується технологія Entity Framework Core (EF Core) – сучасний ORM-фреймворк (Object-Relational

Mapping) від Microsoft, який дозволяє взаємодіяти з реляційними базами даних через об’єктно-орієнтовані моделі замість написання SQL-запитів вручну.

EF Core забезпечує двосторонню синхронізацію між базою даних і моделями C# у програмному коді, що дозволяє:

- створювати, зчитувати, оновлювати та видаляти дані за допомогою LINQ-запитів.
- автоматично створювати або оновлювати схему бази даних на основі змін у класах (через міграції).
- керувати зв’язками між сутностями (один до одного, один до багатьох, багато до багатьох).

На рисунку 2.4 зображено схему роботи EF Core в порівнянні з ORM [15].

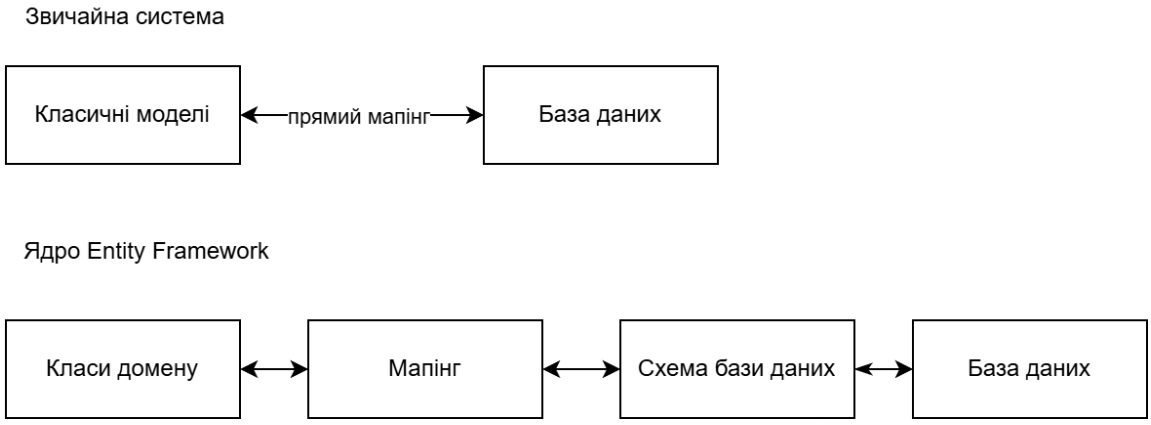


Рисунок 2.4 – Зображено схему роботи EF Core в порівнянні з ORM

У межах проекту було реалізовано підхід Code First, при якому спочатку створюються класи моделей у коді, а вже потім на їх основі генерується структура бази даних.

2.3.2 Механізм міграцій у EF Core

У межах розробки програмного забезпечення важливо забезпечити можливість поетапного оновлення структури бази даних відповідно до змін у

логіці застосунку. У цьому проекті для цього використовується механізм міграцій Entity Framework Core (EF Core), який дозволяє керувати еволюцією схеми БД у контрольований і відтворюваний спосіб [16]. Міграції дають змогу:

- автоматично створювати початкову структуру бази даних;
- додавати, змінювати або видаляти таблиці, поля, зв’язки;
- уникати ручного написання SQL-скриптів;
- зберігати історію змін у вигляді окремих версій, що можуть бути застосовані або відкочені.

2.3.3 Створення та застосування міграцій

Після створення та внесення змін до моделі, будь то додавання нових змінних чи перевисання повної структури, після збереження відкриваємо консоль диспетчера пакунків, та створюємо міграцію або робимо через консоль відкриту в папці нашого застосунку. Після успішної генерації міграції робиться завантаження до самої бази даних.

```
Add-Migration *назва нашої міграції*
Update-Database
```

Усі зміни записуються у службову таблицю __EFMigrationsHistory, що дозволяє відстежувати, які саме міграції вже застосовані. Такий підхід забезпечує високу гнучкість і зручність бо не потрібно вручну писати SQL-скрипти, а усі зміни до структури зберігаються у вигляді контрольованих версій, що особливо зручно під час командної розробки. Міграції легко додавати, редагувати, видаляти та відкатувати. Принцип роботи дуже схожий як з гілками Git, а відновлення структури БД у новому середовищі потребує лише кількох команд, що суттєво прискорює розгортання та супровід проекту.

2.3.4 Автентифікація та авторизація в системі

У сучасних веб-сайтах забезпечення безпечного доступу до функціоналу є критично важливою частиною архітектури. Для цього застосовуються два взаємопов’язані механізми – автентифікація (authentication) та авторизація (authorization). Автентифікація відповідає за підтвердження особи користувача під час входу в систему, тоді як авторизація визначає, які саме ресурси та дії дозволено виконувати конкретному користувачеві залежно від його ролі або прав.

У межах технологічного стеку ASP.NET Core ці механізми реалізуються за допомогою вбудованого фреймворку ASP.NET Core Identity. Це потужна та гнучка система керування обліковими записами, що дозволяє швидко інтегрувати захищену модель доступу до ресурсів. Identity підтримує базові сценарії – реєстрація, вхід, вихід, скидання пароля, підтвердження електронної пошти, двофакторна автентифікація тощо. Облікові записи зберігаються у таблиці AspNetUsers, яка містить ключову інформацію про користувача: ім’я, email, хешований пароль, статус підтвердження акаунта, параметри блокування, а також службові налаштування безпеки. Паролі зберігаються виключно у вигляді хешів із застосуванням криптографічних алгоритмів, що гарантує захист даних навіть у разі втрати даних з бази.

Модель авторизації в ASP.NET Core Identity побудована на основі ролей. Ролі визначаються у таблиці AspNetRoles, а їх прив’язка до користувачів – у таблиці AspNetUserRoles. Це дозволяє реалізувати гнучку політику доступу, за якої певний функціонал буде доступний лише окремим категоріям користувачів – наприклад, адміністраторам адмін-панель, клієнтам особистий кабінет і інший функціонал під них. На рівні коду контроль доступу реалізується за допомогою атрибута [Authorize(Roles = "Admin")], що забезпечує декларативний спосіб захисту маршрутів, методів API або Razor-сторінок. Для розширеної авторизації передбачено використання claims, політик, токенів, а також таблиць для зовнішніх логінів (OAuth), що дозволяє інтегрувати автентифікацію через

сторонні сервіси (Google, Facebook тощо). Усі ці таблиці та механізми повністю інтегровані з базою даних за допомогою Entity Framework Core, що дозволяє зберігати цілісність даних і централізовано керувати міграціями.

Таким чином, застосування ASP.NET Core Identity забезпечує надійну, масштабовану та розширювану систему керування користувачами, яка повністю відповідає сучасним вимогам до безпеки веб-сайтів. Її гнучкість дозволяє ефективно реалізувати як базові сценарії автентифікації, так і складні багаторівневі моделі авторизації.

2.4. Висновок до другого розділу

У даному розділі було виконано проектування інформаційної системи веб-сайту фотостудії з урахуванням вимог функціональності, безпеки та масштабованості. Проведено аналіз основних сутностей предметної області та побудовано реляційну структуру бази даних із використанням платформи Microsoft SQL Server та технології Entity Framework Core. Було розроблено логічну модель бази даних, що включає ключові таблиці, такі як AspNetUsers, Bookings, Briefs, Services, UploadedPhotos та інші, які пов’язані між собою через зовнішні ключі та повністю відповідають принципам нормалізації. ER-діаграма, що представлена у розділі, наочно демонструє ці зв’язки та логіку зберігання інформації. Окрему увагу приділено механізму міграцій у EF Core, який дозволяє ефективно керувати змінами структури бази даних у процесі розробки. Розглянуто теоретичні та практичні аспекти використання ASP.NET Core Identity, що забезпечує надійну систему автентифікації та авторизації користувачів, дозволяючи диференціювати рівні доступу на основі ролей.

Проектована база даних є фундаментом для подальшої розробки логіки застосунку, забезпечує цілісність даних, спрощує обслуговування системи та створює передумови для подальшого розширення функціоналу без необхідності внесення суттєвих змін у її структуру.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ФОТОСТУДІЇ "YURAALE"

3.1. Архітектура та структура веб-сайту

Розробка веб-сайту фотостудії "Yuraale" передбачала створення масштабованого, надійного та зручного для користувача веб-сайту, що поєднує в собі адміністративну та публічну частини. Архітектура програмного продукту побудована з урахуванням сучасних стандартів веб-розробки, використовуючи фреймворк ASP.NET Core та патерн Model–View–Controller (MVC). Застосування архітектури MVC дозволило розділити логіку роботи додатку на три окремі компоненти: модель (робота з даними), представлення (інтерфейс користувача) та контролер (обробка запитів і маршрутизація). Це полегшує тестування, супровід та масштабування системи. Серверна частина сайту обробляє HTTP-запити, виконує авторизацію користувачів, взаємодіє з базою даних через ORM-технологію Entity Framework Core і динамічно формує HTML-відповіді. Клієнтська частина веб-сайту представлена у вигляді сторінок, створених з використанням Razor Pages, які забезпечують гнучке та зручне відображення даних. Інтерфейс ресурсу поділений на зони доступу: публічну (головна сторінка, послуги, портфоліо, форма зворотного зв’язку) та захищену (панель адміністратора, система керування контентом, модуль обробки замовлень). Для реалізації автентифікації та авторизації використовується ASP.NET Core Identity, що забезпечує безпечний доступ до функцій відповідно до ролей користувачів.

Кодова база додатку структурована у вигляді окремих модулів: Controllers, Models, Views, Data, Services, що відповідає принципам чистої архітектури та сприяє логічній організації програмного середовища. Статичні ресурси, такі як стилі, зображення та скрипти, зберігаються у каталозі wwwroot.

Загальна структура проєкту дозволяє легко розширювати функціонал сайту у майбутньому – наприклад, додати онлайн-оплати, календарі доступності або підтримку кількох мов. Вона також забезпечує адаптивність інтерфейсу для коректної роботи на мобільних пристроях, що є критично важливим для цільової аудиторії веб-сайту фотостудії.

3.2. Реалізація функціоналу веб-сайту

У процесі розробки веб-сайту фотостудії "Yuraale" було реалізовано низку ключових функціональних можливостей, які забезпечують ефективну взаємодію користувача з ресурсом, а також зручне адміністрування контенту та замовлень.

Головна мета публічної частини – презентація фотостудії, ознайомлення потенційних клієнтів з послугами та портфоліо, а також надання зручного каналу для зв’язку.

Головна сторінка (Рис. 3.1) містить вітальний блок із заголовком та коротким описом студії, а також слайдер із прикладами робіт. Всю інформацію про надавання послуг клієнт може передивитись в бріфінгу.

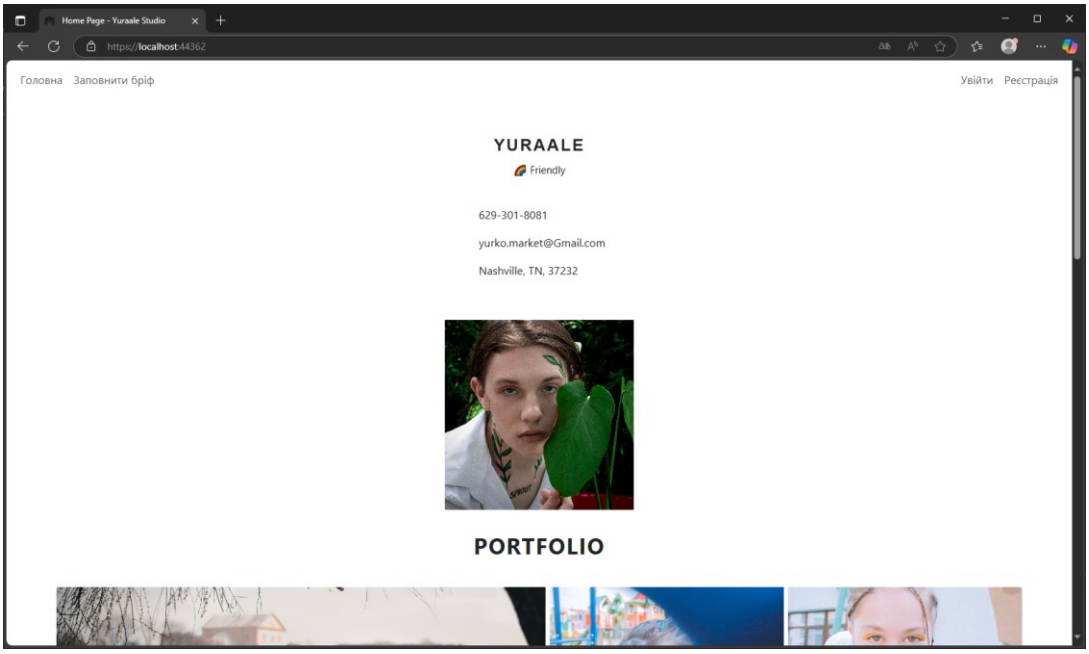


Рисунок 3.1 – Головна сторінка сайту фотостудії

Додатково реалізовано розділ портфоліо, який надає змогу користувачеві переглядати роботи фотографа. Галерею можна структурувати відповідно до типу зйомки: студійна, виїзна, весільна тощо. Кожна галерея містить добірку фотографій, що відображають професійний рівень фотостудії та допомагають клієнтові зорієнтуватися у стилі роботи.

Сторінка бронювання фотосесії реалізована у вигляді інтерактивної форми – бріфу (Рис. 3.2), що дозволяє клієнтові вказати деталі замовлення: ім’я, контактну інформацію, побажання щодо стилю зйомки, бажану локацію, кількість учасників, необхідність візажиста тощо. Ці дані зберігаються у базі даних і надходять адміністратору для подальшого опрацювання. Такий підхід дозволяє уникнути телефонних або ручних домовленостей, централізуючи всі запити в системі.

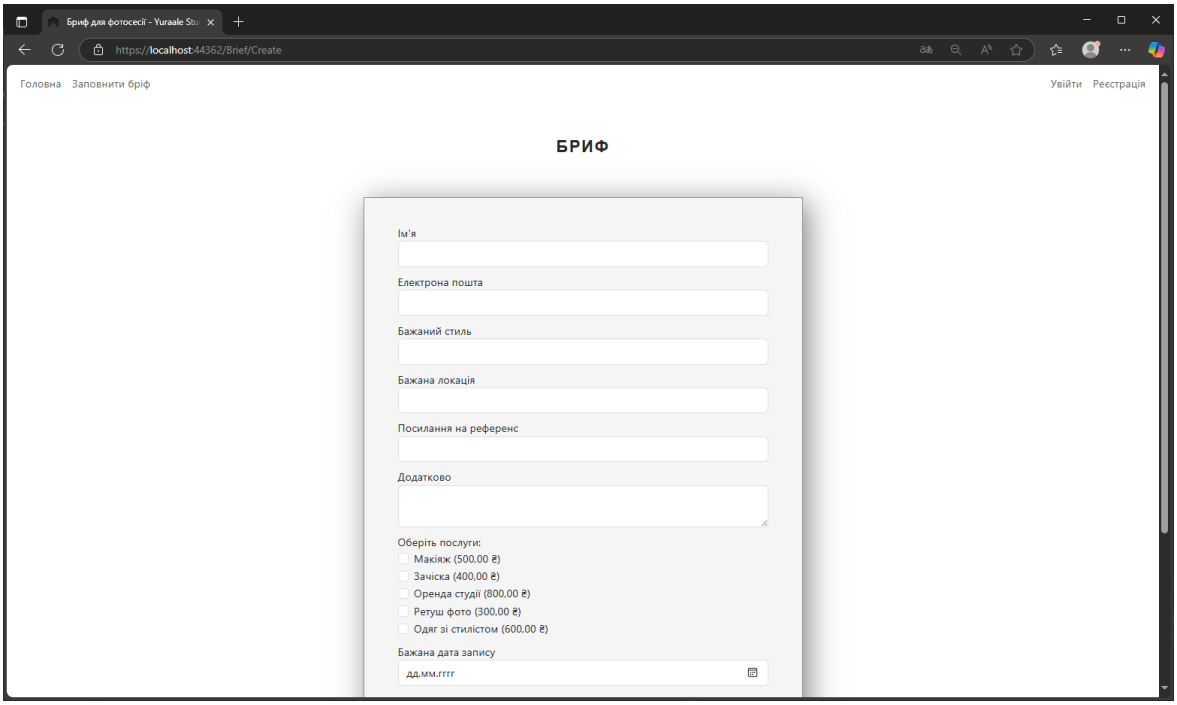


Рисунок 3.2 - Вигляд сторінки створення брифу

Для забезпечення адміністрування реалізовано захищену адмін-панель, доступну лише авторизованим користувачам. Через неї фотограф або адміністратор має змогу:

- додавати, редагувати та видаляти галереї та окремі фото;
- оновлювати опис і вартість послуг;
- переглядати й обробляти заявки на фотосесію;
- затверджувати бриф клієнта та надсилати квитанції на оплату;

Всі дані зберігаються у базі даних Microsoft SQL Server, взаємодія з якою реалізована за допомогою Entity Framework Core, що забезпечує швидку й ефективну роботу з моделями та таблицями.

Фронтенд-частина реалізована за допомогою Razor Views та стандартних HTML/CSS-технологій з адаптивною верстkою, що дозволяє сайту коректно відображатися на різних пристроях, включаючи смартфони та планшети.

Таким чином, реалізований веб-сайт фотостудії "Yuraale" поєднує в собі сучасний дизайн, інтуїтивно зрозумілий інтерфейс та потужну серверну частину, що дозволяє ефективно обробляти запити клієнтів і вести внутрішній облік фотозйомок.

Для підвищення зручності взаємодії з користувачами реалізовано функціонал надсилання повідомлень на електронну пошту. Після створення брифу або підтвердження заявки клієнт отримує відповідне повідомлення з деталями замовлення. Це дозволяє оперативно інформувати користувача про статус бронювання без необхідності ручного контакту. Також впроваджено можливість оплати послуг через платіжну систему LiqPay, що забезпечує швидкий та безпечний розрахунок за фотосесію безпосередньо через сайт [17]. Сторона адміністратора має змогу надсилати квитанцію клієнту після затвердження брифу, яка містить посилання на оплату через LiqPay. На рисунку 3.3 зображено сторінка оплати для клієнта.

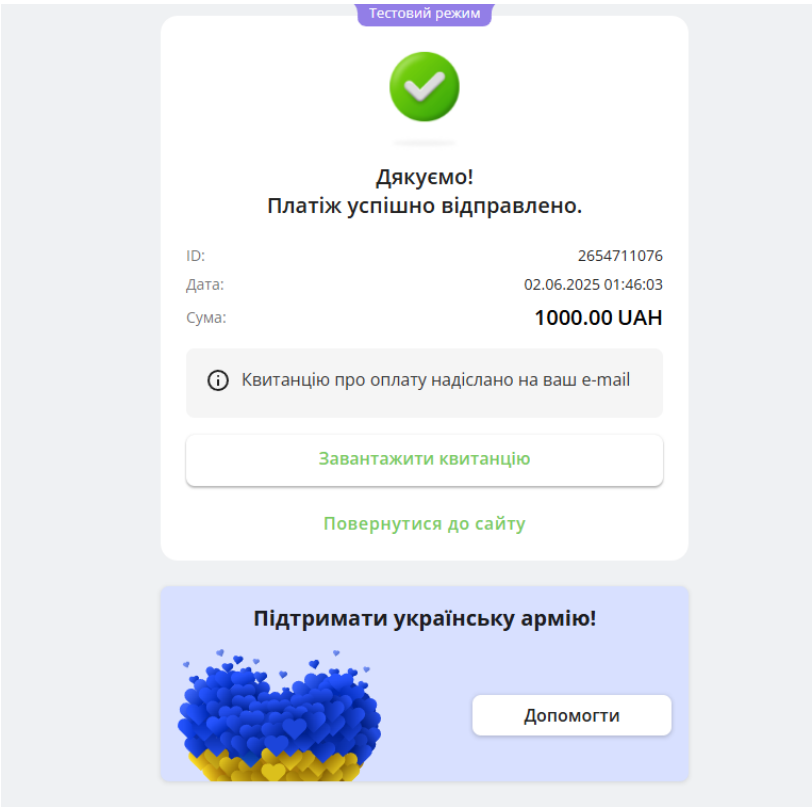


Рисунок 3.3 – Вигляд оплати послуг у клієнта

Та на рисунку 3.4 зображено квитанція яка надійшла нам на пошту після оплати.

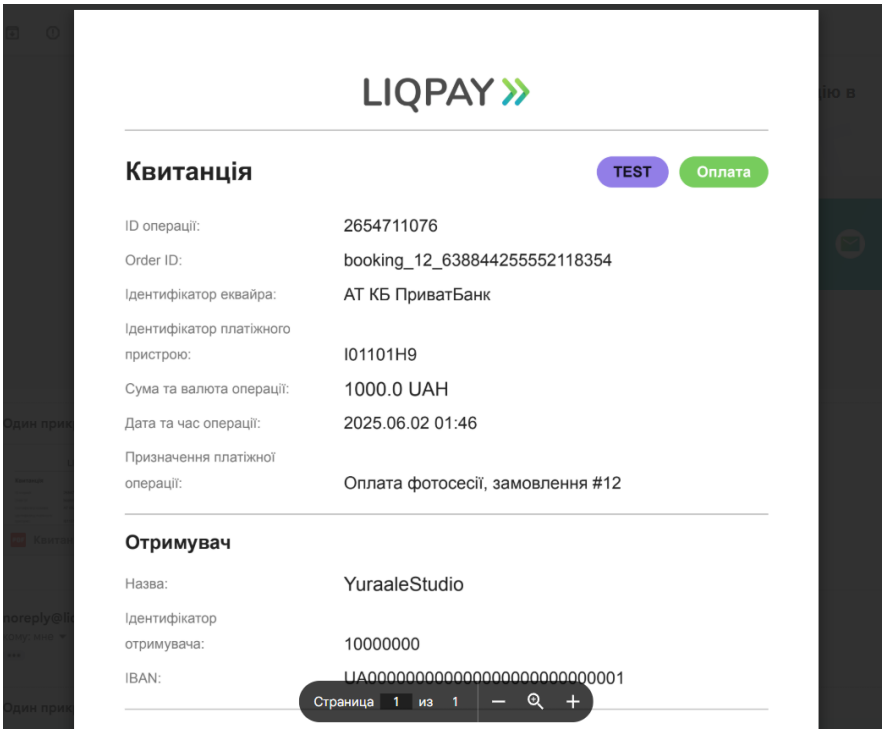


Рисунок 3.4 – Квитанція після оплати послуг

Для управління правами доступу до захищених розділів застосунку використано механізм аутентифікації та авторизації на основі ASP.NET Identity. Це забезпечує безпечне зберігання облікових даних користувачів, дозволяє реалізувати вхід до адмін-панелі лише для авторизованих осіб та підтримує рольову модель доступу, яку за потреби можна масштабувати.

Загалом, комплекс функціональних можливостей веб-сайту спрямований на створення зручного сервісу як для кінцевих користувачів, так і для адміністратора фотостудії, забезпечуючи повноцінний онлайн-цикл взаємодії – від перегляду портфоліо до оплати замовлення.

3.3. Тестування та перевірка функціональності веб-сайту

Після завершення розробки функціональних модулів веб-сайту фотостудії "Yuraale" було проведено комплексне тестування з метою перевірки коректності роботи всіх компонентів системи, забезпечення стабільності, безпеки та зручності користування.

3.3.1 Методи тестування

Для тестування було застосовано ручне функціональне тестування згідно з розробленими сценаріями використання системи. Також використовувалися інструменти розробника браузера для перевірки коректності відображення інтерфейсу на різних розширеннях екрана.

Тестування охоплювало наступні аспекти:

- Коректність заповнення форм: перевірено обробку введених даних у формі брифу, включаючи валідацію електронної пошти, імені, а також обробку незаповнених полів.

- Аутентифікація: протестовано логін адміністратора, перевірено захист доступу до адміністративної панелі без входу.
- Фільтрація і перегляд портфоліо: перевірено навігацію між галереями, відкриття фотографій, динамічне завантаження контенту.
- Створення та обробка бронювань: перевірено сценарій повного циклу: створення клієнтом запиту – підтвердження адміністратором – відображення у базі даних.

3.3.2 Результати тестування

У ході тестування було виявлено низку незначних помилок, таких як:

- некоректне відображення сповіщень при частковому заповненні форми брифу.
- відсутність повідомлення про успішне збереження змін в адмін-панелі.
- дрібні проблеми адаптивності на мобільних пристроях (перекриття елементів на малих екранах).

Усі виявлені недоліки були оперативно усунені. Повторне тестування підтвердило стабільну та коректну роботу всіх модулів системи.

3.3.3 Тестування безпеки

Було здійснено перевірку доступу до захищених сторінок. При спробі перейти в адмін-панель без авторизації користувач автоматично перенаправляється на сторінку входу. Захист реалізовано через механізми авторизації ASP.NET Identity. Крім того, форми введення мають захист від базових атак типу SQL-ін’єкцій завдяки використанню Entity Framework Core, що генерує параметризовані запити до бази даних.

3.4. Технічна реалізація окремих компонентів веб-сайту

У цьому підпункті детально розглянемо, як реалізовано деякі ключові компоненти веб-сайту фотостудії "Yuraale". Особливу увагу приділено обробці брифу, надсиланню пошти, підключенню платіжної системи LiqPay та механізму авторизації користувачів.

3.4.1 Збереження брифу клієнта

Одна з основних функцій сайту – можливість клієнта заповнити бриф, тобто анкету із побажаннями щодо зйомки. Ця форма надсилає дані на сервер, де вони зберігаються в базі даних. Для цього було створено модель Brief, яка відповідає структурі таблиці в базі. Лістинг 3.1 зображено модель бріфу.

Лістинг 3.1 – Модель бріфу

```
public class Brief
{
    public int Id { get; set; }
    public int BookingId { get; set; }
    public Booking? Booking { get; set; }

    [Required]
    [Display(Name = "Ім'я")]
    public string ClientName { get; set; }

    [Required, EmailAddress]
    [Display(Name = "Електрона пошта")]
    public string ClientEmail { get; set; }

    [Display(Name = "Бажаний стиль")]
    public string ShootingStyle { get; set; }

    [Display(Name = "Бажана локація")]
    public string LocationPreferences { get; set; }

    [Display(Name = "Посилання на референс")]
    public string? ReferencesLinks { get; set; }

    [Display(Name = "Додатково")]
    public string? AdditionalNotes { get; set; }
```

```
    public bool NeedMakeupArtist { get; set; }

    [Display(Name = "Бажана дата запису")]
    public DateTime? PreferredDate { get; set; }
    [Display(Name = "Час")]
    public TimeSpan? PreferredTime { get; set; }

    public bool IsTermsAccepted { get; set; }

    public bool IsReviewedByPhotographer { get; set; }
    public bool IsApproved { get; set; }

    public ICollection<BriefService> BriefServices { get; set; } =
new List<BriefService>();

    [NotMapped]
    public List<int> SelectedServiceIds { get; set; } = new();

}
```

Ця модель дозволяє зберігати ім’я клієнта, його електронну адресу, стиль зйомки та інші деталі. Поля Required забезпечують валідацію введених даних.

У контролері реалізовано метод, який обробляє надсилання цієї форми. Метод перевіряє валідність моделі, зберігає її до бази даних за допомогою Entity Framework Core, після чого перенаправляє користувача на сторінку з підтвердженням.

3.4.2 Надсилання поштового повідомлення

Після успішного створення брифу, клієнту автоматично надсилається повідомлення на електронну пошту. Це реалізовано через стандартний клас SmtpClient, який дозволяє взаємодіяти з поштовим сервером. На лістингу 3.2 зображено клас який відповідає за надсилання повідомлень користувачу.

Лістинг 3.2 – Метод класу EmailService

```
public async Task SendEmailAsync(string toEmail, string subject,
string body)
{
    var message = new MimeMessage();
```

```
message.From.Add(MailboxAddress.Parse(_config["SmtpSettings:User"]
));
    message.To.Add(MailboxAddress.Parse(toEmail));
    message.Subject = subject;
    message.Body = new TextPart("html") { Text = body };

    using var smtp = new SmtpClient();
    await smtp.ConnectAsync(
        _config["SmtpSettings:Host"],
        int.Parse(_config["SmtpSettings:Port"]),
        SecureSocketOptions.StartTls
    );

    await smtp.AuthenticateAsync(
        _config["SmtpSettings:User"],
        _config["SmtpSettings:Password"]
    );

    await smtp.SendAsync(message);
    await smtp.DisconnectAsync(true);
}
```

Надсилання реалізується за допомогою SMTP-протоколу через сервер Gmail із увімкненим шифруванням SSL (порт 587)[19]. Це забезпечує безпечну передачу повідомлення та захищає дані користувача під час обміну з поштовим сервером. У майбутньому цей функціонал може бути розширений: наприклад, додаванням HTML-розмітки до листа для оформлення логотипом студії, QR-кодом з деталями замовлення, або прикріпленням PDF-файлу з копією брифу.

На рисунку 3.5 зображено схему роботи протоколу SMTP

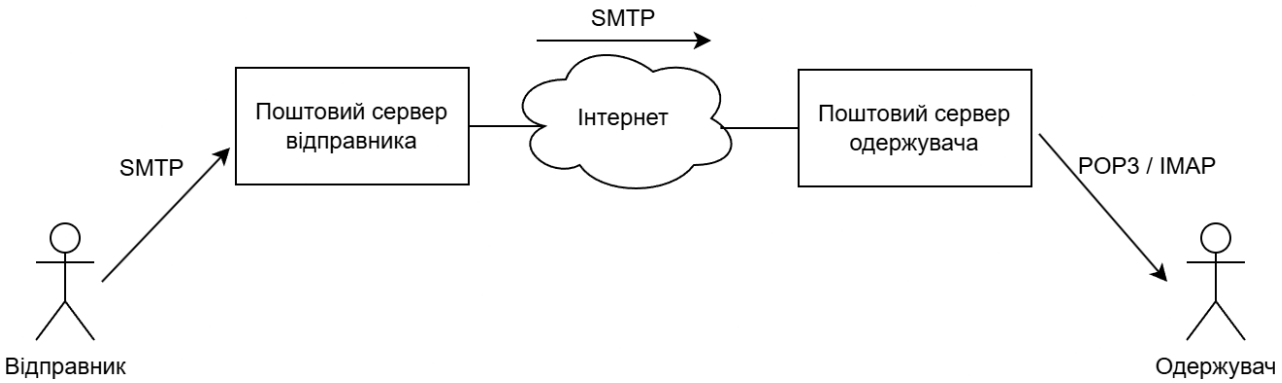


Рисунок 3.5 – Схема роботи протоколу SMTP

3.4.3 Підключення платіжної системи LiqPay

Ще один важливий компонент сайту – можливість онлайн-оплати послуг через LiqPay. Після затвердження заявки адміністратором, клієнт отримує посилання на оплату, яке генерується через API LiqPay. На лістингу 3.3 зображено реалізація сервісу для роботи з LiqPay.

Лістинг 3.3 – Реалізація сервісу для роботи з LiqPay

```
public class LiqPayService
{
    private readonly string _publicKey = "sandbox_i59968869870";
    private readonly string _privateKey =
"sandbox_aVs26K7kQyYF0B7VrR8nEN2ufBn4iv2z5ccbxKfw";

    public (string data, string signature) GeneratePayment(int
bookingId, decimal amount, string email)
    {
        var orderId = $"booking_{bookingId}_{DateTime.Now.Ticks}";
        var json = JsonSerializer.Serialize(new
        {
            version = 3,
            public_key = _publicKey,
            action = "pay",
            amount = amount,
            currency = "UAH",
            description = $"Оплата фотосесії, замовлення
#{bookingId}",
            order_id = orderId,
            result_url =
$"https://localhost:44362/Payment/Success?bookingId={bookingId}",
            server_url =
$"https://localhost:44362/Payment/Callback",
            email = email
        });

        var data =
Convert.ToBase64String(Encoding.UTF8.GetBytes(json));
        var signString = _privateKey + data + _privateKey;
        var signature =
Convert.ToBase64String(SHA1.Create().ComputeHash(Encoding.UTF8.Get
Bytes(signString)));

        return (data, signature);
    }
}
```

Таким чином, створений сервіс LiqPayService дозволяє централізовано та безпечно формувати параметри для ініціалізації платежу. Після отримання пари data і signature, ці значення передаються у HTML-форму, яка автоматично перенаправляє користувача на сторінку LiqPay для здійснення транзакції. Інтеграція з LiqPay дозволила автоматизувати процес оплати фотосесій без залучення адміністратора на цьому етапі. Після здійснення платежу, система отримує зворотній запит (callback) від платіжного сервера на вказаний server_url, що дозволяє зафіксувати статус платежу та оновити бронювання як оплачене.

Крім того, користувач після оплати повертається на сторінку result_url, де відображається повідомлення про успішне завершення оплати, а при необхідності – форма з квитанцією. Завдяки використанню стандартного механізму шифрування та підпису, реалізованого відповідно до вимог API LiqPay, система гарантує захищеність платіжних даних і відповідність сучасним стандартам електронної комерції.

3.4.4Реалізація аутентифікації через ASP.NET Identity

Для захисту адміністративної частини веб-сайту та обмеження доступу до конфіденційної інформації була реалізована система аутентифікації та авторизації користувачів на основі фреймворку ASP.NET Identity [20]. Даний підхід дозволяє швидко розгорнути функціонал реєстрації, входу, керування ролями та безпечного зберігання облікових даних.

У рамках проєкту було підключено базову конфігурацію Identity у файлі Startup.cs, де в методі ConfigureServices вказано конфігурацію служби аутентифікації та збереження користувачів у базі даних (Лістинг 3.4)

Лістинг 3.4 – Конфігурація служби аутефікації.

```
builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
{
    options.SignIn.RequireConfirmedAccount = false;
})
   .AddRoles<IdentityRole>()
```

```
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultUI();
```

Тут ApplicationUser – це клас користувача, який при потребі можна розширити додатковими полями, такими як ім’я, контактний номер чи роль. IdentityRole використовується для визначення типу користувача, зокрема для розмежування прав доступу між адміністраторами та іншими користувачами. Застосування EntityFrameworkStores забезпечує збереження користувачів і ролей у відповідних таблицях бази даних. Додаткові токени, наприклад для скидання пароля або підтвердження пошти, генеруються через DefaultTokenProviders.

Функціонал входу реалізований у відповідному контролері AccountController. Метод входу перевіряє дані, введені користувачем у формі, і виконує аутентифікацію за допомогою сервісу _signInManager. Якщо перевірка пройдена успішно – користувача перенаправляють до панелі адміністратора. Лістинг 3.5 зображено метод входу.

Лістинг 3.5 – Метот авторизації на сайті.

```
public async Task OnGetAsync(string returnUrl = null)
{
    if (!string.IsNullOrEmpty(ErrorMessage))
    {
        ModelState.AddModelError(string.Empty, ErrorMessage);
    }
    returnUrl ??= Url.Content("~/");
    await
HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);
    ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    ReturnUrl = returnUrl;
}

public  async  Task<IActionResult>  OnPostAsync(string  returnUrl  =
null)
{
    returnUrl ??= Url.Content("~/");

    ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

    if (ModelState.IsValid)
    {
```

```
        var result = await
_signInManager.PasswordSignInAsync(Input.Email, Input.Password,
Input.RememberMe, lockoutOnFailure: false);
        if (result.Succeeded)
        {
            _logger.LogInformation("User logged in.");
            return LocalRedirect(returnUrl);
        }
        if (result.RequiresTwoFactor)
        {
            return RedirectToPage("./LoginWith2fa", new { ReturnUrl
= returnUrl, RememberMe = Input.RememberMe });
        }
        if (result.IsLockedOut)
        {
            _logger.LogWarning("User account locked out.");
            return RedirectToPage("./Lockout");
        }
        else
        {
            ModelState.AddModelError(string.Empty, "Invalid login
attempt.");
            return Page();
        }
    }
    return Page();
}
```

У наведеному коді здійснюється перевірка логіна та пароля. У випадку помилки, користувач бачить повідомлення з поясненням. Окрім цього, реалізовано перевірку ролі користувача – доступ до розділів панелі адміністратора дозволено лише тим користувачам, які мають роль Admin. Облікові записи зберігаються у базі даних у спеціальних таблицях, які автоматично створюються фреймворком Identity при ініціалізації міграцій. Паролі зберігаються у вигляді хешованих значень із додаванням "солі", що забезпечує високий рівень захисту персональних даних навіть у разі компрометації бази.

Інтерфейс форми входу реалізований за допомогою Razor Pages, де використовується стандартна Razor-синтаксис форма @Html.BeginForm. Користувач вводить свій email і пароль, після чого система перевіряє відповідність даних та відкриває доступ до адміністративної панелі або відображає повідомлення про помилку.

На рисунку 3.6 зображено схему роботи ASP.NET Identity.

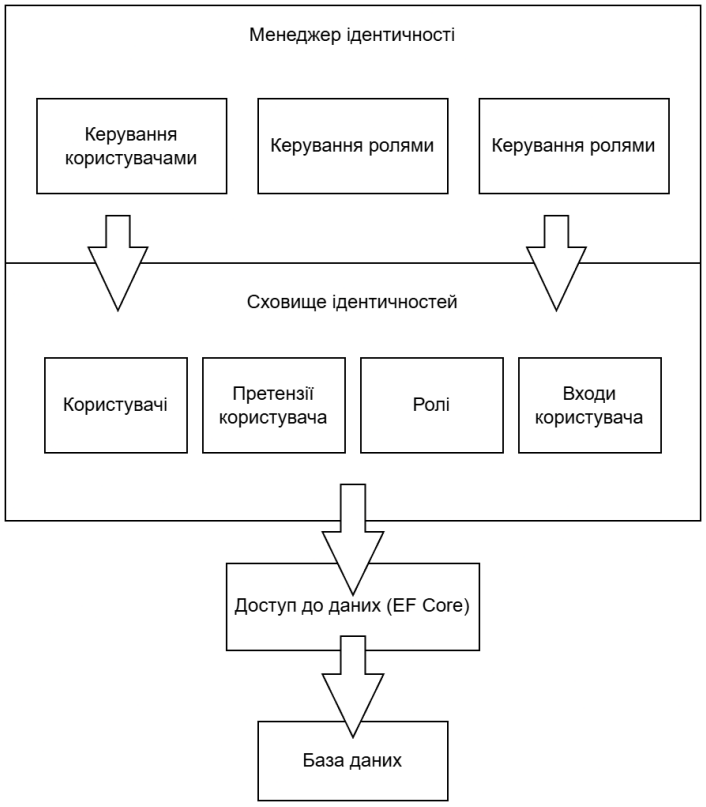


Рисунок 3.6 – Схему роботи ASP.NET Identity

Переваги використання ASP.NET Identity:

- Готовий набір функцій: реєстрація, вхід, зміна пароля, підтвердження email тощо.
- Гнучка рольова модель: можна призначити різні права доступу для різних типів користувачів.
- Безпека: сучасні алгоритми хешування (наприклад, PBKDF2), захист від атак типу brute-force.
- Масштабованість: підтримка зовнішніх провайдерів входу (Google, Facebook, Microsoft) без додаткових змін архітектури.

Завдяки реалізації аутентифікації через ASP.NET Identity адміністрування сайтом стало безпечним, контрольованим та готовим до подальшого розширення

(наприклад, додавання нових ролей, журналів дій, багатофакторної авторизації тощо).

Окрім вищезазначених переваг, ASP.NET Identity забезпечує високу інтеграцію з іншими компонентами платформи ASP.NET, зокрема із системами авторизації через атрибути [Authorize], middleware-компонентами, а також системою Claims-based authentication, що дозволяє реалізовувати гнучкі політики доступу на основі призначених прав. Завдяки цьому розробник має змогу не лише обмежувати доступ до окремих контролерів або сторінок, а й адаптувати систему під складні сценарії, де права користувача залежать від контексту, наприклад, його відділу, посади або типу замовлення. Це особливо корисно у випадках, коли в подальшому планується інтеграція системи з CRM або ERP-рішеннями. Крім того, ASP.NET Identity підтримує інтеграцію з Token-based authentication (наприклад, OAuth2 або JWT), що дозволяє у перспективі реалізувати API-доступ до адмін-функцій або мобільного застосунку без суттєвих змін у наявній архітектурі.

Таким чином, обрана технологія не лише забезпечила надійний захист та зручний механізм аутентифікації на поточному етапі реалізації сайту фотостудії "Yuraale", а й створила міцне підґрунтя для розвитку системи у напрямках інтеграції, масштабування та розширення функціоналу у майбутньому.

3.5. Висновок до третього розділу

У даному розділі було докладно розглянуто основні етапи реалізації функціоналу веб-сайту фотостудії "Yuraale", а також проведено технічний аналіз ключових компонентів системи. Здійснено впровадження форм брифування, обробки замовлень, підключення платіжного сервісу LiqPay, налаштування системи аутентифікації користувачів та реалізацію адміністративної панелі. Проведене тестування підтвердило коректність роботи всіх модулів сайту, а розроблені механізми забезпечили високий рівень автоматизації бізнес-процесів

фотостудії. Система дозволяє зручно керувати контентом, отримувати замовлення від клієнтів у режимі онлайн, автоматично підтверджувати бриф, надсилати сповіщення на пошту та приймати оплату за послуги. Використання сучасних технологій, таких як ASP.NET MVC, Entity Framework Core, ASP.NET Identity та LiqPay API, забезпечило не лише гнучкість і масштабованість розробленого рішення, а й заклало надійне підґрунтя для подальшого розвитку інформаційної системи. Реалізований веб-сайт повністю відповідає функціональним вимогам, задовольняє потреби адміністратора фотостудії та забезпечує зручну взаємодію кінцевого користувача з сервісом.

Таким чином, третій розділ продемонстрував приклад практичного втілення теоретичних знань з веб-програмування, проєктування баз даних, організації безпечного доступу до ресурсів та інтеграції зовнішніх сервісів у сучасні веб-сайту.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1. Аналіз умов праці програміста

Процес розробки веб-сайту фотостудії "Yuraale" виконувався в умовах, характерних для праці програміста, тобто офісного або дистанційного середовища з використанням персонального комп’ютера, засобів зв’язку, програмного забезпечення та периферійного обладнання. Така діяльність є малорухливою, вимагає високої концентрації уваги, інтенсивної роботи органів зору та точності при виконанні великої кількості дрібних моторних дій.

Робочий день розробника зазвичай передбачає тривале перебування в статичному положенні (сидячи), роботу за монітором із застосуванням клавіатури та миші. Це створює комплекс несприятливих умов, які можуть негативно впливати на стан здоров’я працівника. До найбільш поширених факторів ризику відносяться:

- Зорове перенапруження.
- Статичні фізичні навантаження.
- Монотонність та психоемоційне напруження.
- Недостатня рухова активність.
- Електромагнітне випромінювання.
- Недотримання норм мікроклімату.

Незважаючи на те, що праця програміста не відноситься до категорії особливо небезпечних, саме тривалість впливу цих факторів та їхній накопичувальний ефект можуть призвести до професійних захворювань у разі нехтування санітарно-гігієнічними вимогами.

Для підтримання нормального самопочуття працівника та зменшення ризиків, необхідно дотримуватись таких рекомендацій, а саме:

- Організувати регулярні перерви під час роботи – як мінімум по 5-10 хв. після години роботи.
- Забезпечити ергономічність робочого місця.
- Контролювати освітлення.
- Дотримуватись температурного режиму.
- Періодичне провітрювання приміщення.

Дотримання зазначених вимог сприяє збереженню здоров'я програміста, підвищує ефективність його діяльності та зменшує ймовірність виникнення ризиків, пов'язаних із впливом негативних факторів виробничого середовища.

Таблиця 4.1 – Шкідливі та небезпечні фактори при роботі програміста.

№	Шкідливі фактори	Потенційний вплив на здоров'я	Засоби профілактики
1	Тривала робота за комп'ютером	Зорове перенапруження, синдром сухого ока	Регулярні перерви, спеціальні окуляри, зволоження повітря
2	Статичне навантаження, сидяча поза	Біль у спині, порушення постави, варикоз	Ергономічне крісло, регулювання висоти столу, перерви
3	Недостатнє або неправильне освітлення	Швидка втомлюваність, головний біль	Робоче освітлення не менше 300 лк, настільні лампи
4	Електромагнітне випромінювання	Потенційне довгострокове подразнення нервової системи	Зміна старих пристроїв на сучасні сертифаковані моделі
5	Психоемоційне навантаження, стрес	Виснаження, зниження концентрації	Раціональний графік роботи, мікро перерви, чергування завдань
6	Перенавантаження кисті/зап'ястя	Тунельний синдром	Ергономічна мишка, клавіатура
7	Ризик ураження електричним струмом	Опіки, ураження струмом	Заземлення техніки, справна електромережа, захисне вимикання

Дотримання зазначених заходів безпеки дозволяє мінімізувати ризики для здоров’я та життя, та забезпечити комфортні умови праці програмістом під час реалізації проекту. Варто зазначити, що організація робочого місця, мікроклімат, санітарно-гігієнічні умови, а також електробезпека регламентуються низкою нормативно-правовими актами, стандартними та галузевими положеннями, обов’язковими для дотримання в ІТ-сфері та під час офісної діяльності.

Основні нормативні документи щодо безпеки праці:

- НПАОП 0.00-1.28-10 – Порядок опрацювання та затвердження положень про систему управління охороною праці.
- ДСТУ EN 29241-3:2003 – Ергономічні вимоги до офісної роботи з відеотерміналами (VDT). Частина 3: Вимоги до організації робочого місця.
- ДБН В.2.2-3-97 – Будинки і споруди. Будівлі і приміщення навчальних закладів (актуальний для організації робочих місць у навчальних закладах).
- ДБН В.2.5-28:2006 – Природне і штучне освітлення – регламентує норми освітленості для офісних приміщень.

Таким чином, праця програміста хоч і не пов’язана з безпосередньо шкідливими або аварійно-небезпечними умовами, однак вимагає належної організації робочого середовища відповідно до чинних нормативних документів. Недотримання санітарно-гігієнічних норм, вимог ергономіки та електробезпеки може призвести до погіршення стану здоров’я, зниження продуктивності праці та підвищення ризику виникнення професійних захворювань.Врахування вимог безпеки життєдіяльності під час реалізації програмних проєктів, зокрема таких, як розробка веб-сайту фотостудії, є важливою складовою не лише особистої безпеки розробника, а й професійної відповідальності за дотримання стандартів праці у сфері інформаційних технологій.

4.2. Вимоги до організації робочого місця програміста

Раціональна організація робочого місця програміста має важливе значення для забезпечення ефективності праці, зниження рівня втоми, підвищення працездатності та профілактики захворювань, пов’язаних з малорухливою діяльністю та тривалим використанням комп’ютерної техніки. Відповідно до чинних санітарно-гігієнічних вимог, робоче місце програміста має бути організоване з урахуванням принципів ергономіки, безпеки та комфорту.

Робочий стіл повинен мати достатню площу для розміщення монітора, клавіатури, маніпулятора, а також додаткового приладдя чи документації. Поверхня столу бажано має бути матовою, без відблисків, і розташована на оптимальній висоті, що дозволяє підтримувати правильне положення тіла під час роботи. Крісло програміста повинно бути регульованим по висоті та мати підтримку поперекового відділу хребта, що дає змогу мінімізувати навантаження на спину при тривалому перебуванні у сидячій позі. Монітор слід розташовувати на відстані 60–70 см від очей користувача, при цьому верхній край екрана має бути на рівні або трохи нижче лінії зору. Це дозволяє зменшити напруження м’язів шиї та уникнути втоми очей. Клавіатура і мишка повинні розташовуватись таким чином, щоб передпліччя були паралельні підлозі, а зап’ястя – у природному положенні. Для зниження ризику розвитку тунельного синдрому доцільно використовувати спеціальні підставки для кистей. Особливу увагу слід приділити освітленню. Рівень загального освітлення має становити не менше 300 лк, бажано використовувати комбіновану систему освітлення, яка включає загальне і локальне (настільна лампа) світло. Світлові потоки не повинні створювати відблиски на поверхні монітора. Також необхідно забезпечити відповідний мікроклімат у приміщенні: температура повинна бути в межах 18–24 °C, відносна вологість – 40–60 %, а повітря має бути чистим та регулярно оновлюватись шляхом провітрювання.

У разі перевищення допустимого рівня шуму (понад 50 дБ), варто передбачити заходи для акустичної ізоляції – використання звукопоглинальних

матеріалів, перегородок, шумопоглинаючих панелей. Окрім того, важливою умовою є правильне розташування електронного обладнання та кабелів, щоб уникнути механічних пошкоджень і знизити ризик випадкових травм. Для профілактики зорового та фізичного перенапруження рекомендовано кожні 60 хвилин безперервної роботи робити короткі перерви тривалістю 5–10 хвилин. У цей час доцільно виконувати вправи для очей, легку розминку або змінювати вид діяльності. Дотримання режиму праці та відпочинку, а також правильна організація робочого місця є важливими чинниками збереження здоров'я програміста, підвищення його продуктивності та зниження ризику виникнення професійних захворювань.

4.3. Електробезпека при роботі з комп'ютерною технікою

У процесі роботи над кваліфікаційною роботою розробник постійно взаємодіє з комп'ютерною технікою, периферійними пристроями та елементами електроживлення. Незважаючи на те, що сучасні офісні приміщення забезпечені засобами електрозахисту, електробезпека залишається важливим аспектом охорони праці в галузі інформаційних технологій. Основним джерелом потенційної небезпеки є електричний струм, який при певних обставинах може спричинити травми, ураження, термічні опіки або навіть смертельний випадок. Найбільш імовірними причинами виникнення аварійних ситуацій є порушення ізоляції проводів, несправні блоки живлення, механічні пошкодження кабелів, неправильне підключення обладнання або використання несертифікованих подовжувачів і розеток.

Для зниження ризиків ураження електричним струмом робоче місце розробника повинно бути обладнане відповідно до вимог нормативної документації. Усі електроприлади мають бути справними, мати відповідні технічні паспорти, сертифікати якості та регулярно проходити перевірку. Електроживлення комп'ютера, монітора та іншого обладнання повинно

здійснюватися через заземлені розетки або блоки живлення, оснащені захистом від короткого замикання та перевантаження. Особливу увагу слід приділяти стану кабелів – вони не повинні бути перекрученими, розташованими під ногами або натягнутими, адже це може спричинити як електротравму, так і механічне пошкодження самого обладнання. Кабелі повинні бути приховані або прокладені в кабель-каналах. У разі виникнення запаху гару, перегріву блоку живлення чи іншого несправного функціонування техніки, користувач зобов'язаний негайно припинити роботу, вимкнути обладнання з мережі та повідомити відповідального спеціаліста або адміністратора приміщення.

Додатково в робочому приміщенні мають бути передбачені первинні засоби пожежогасіння (наприклад, вогнегасники вуглекислотного типу), доступ до яких має бути вільним та позначеним. Робота з комп'ютерною технікою заборонена у вологому середовищі або при виявленні зовнішніх ушкоджень корпусу пристроїв. Дотримання вимог електробезпеки дозволяє запобігти аварійним ситуаціям, зберегти життя і здоров'я працівника, а також забезпечує надійну та безперебійну роботу обладнання в межах реалізації інформаційних систем.

4.4. Загальні вимоги безпеки з охорони праці для користувачів та розробників веб-сайту

У сучасному цифровому суспільстві веб-сайту є невіддільною частиною як повсякденної діяльності користувачів, так і бізнес-процесів компаній. Саме тому питання безпеки набувають особливої ваги. Забезпечення охорони праці та захисту здоров'я має враховувати не тільки традиційні виробничі ризики, але й нові виклики, пов'язані з використанням інформаційних систем. У контексті кваліфікаційної роботи з розробки сайту фотостудії "Yuraale" це питання стосується як розробника, так і кінцевого користувача, який взаємодіє з веб-сайтом.

Для користувачів, які відвідують сайт, залишають персональні дані у формах замовлення, завантажують зображення або здійснюють онлайн-оплату, особливе значення має цифрова безпека. Передусім, це стосується захисту персональної інформації, фінансових реквізитів та інших чутливих даних. Загальні рекомендації щодо безпечної взаємодії з веб-сайтами включають використання захищених з'єднань (HTTPS), уважне ставлення до введення конфіденційних даних, уникнення підозрілих посилань, активне використання двофакторної аутентифікації, якщо вона доступна, а також регулярне оновлення паролів. Користувачі повинні усвідомлювати ризики фішингових атак, втручання у сесію та можливість несанкціонованого доступу до акаунту у випадку нехтування базовими заходами цифрової гігієни.

З боку розробника система безпеки повинна бути продумана ще на етапі проєктування веб-сайту. Розробник несе відповідальність не лише за коректність роботи функціоналу, але й за безпечність обробки та зберігання даних користувачів. У реалізації сайту фотостудії "Yuraale" були застосовані ключові принципи захисту інформації: обмеження доступу до адмін-панелі через систему ролей, шифрування паролів за допомогою ASP.NET Identity, перевірка вхідних даних через серверну валідацію, використання анти-XSS і анти-CSRF механізмів для запобігання найпоширенішим типам атак.

Крім безпосередньої реалізації програмного захисту, велике значення має також організація роботи розробника, що прямо пов'язана з охороною праці. Це охоплює ергономіку робочого місця, дотримання електробезпеки, захист від електромагнітного випромінювання, регламентовані перерви, належне освітлення та вентиляцію приміщення. Відомо, що порушення санітарно-гігієнічних вимог може призвести до порушення зору, захворювань опорно-рухового апарату, хронічної втоми та психоемоційного вигорання. У зв'язку з цим, розробники веб-сайтів мають дотримуватися встановленого режиму праці, використовувати якісне обладнання, дбати про правильну посадку та робочу позу, а також створювати здорові умови праці навіть при роботі з дому.

Також варто враховувати, що веб-сайт фотостудії надає можливість здійснення онлайн-оплат через платіжну систему LiqPay, що накладає додаткові вимоги до безпеки – як юридичної, так і технічної. Зокрема, у системі реалізовано використання цифрових підписів, контроль цілісності даних, захист конфіденційної інформації через шифрування. Програміст повинен забезпечити перевірку коректності транзакцій, обробку відповідей від API LiqPay, а також зберігання лише необхідної мінімальної кількості платіжних реквізитів (відповідно до вимог PCI DSS). З точки зору правової відповідальності, розробник несе зобов’язання щодо дотримання Закону України "Про захист персональних даних", а також міжнародних стандартів, якщо сайт працює з іноземними користувачами (наприклад, GDPR – General Data Protection Regulation). Система має забезпечити можливість для користувача ознайомитися з політикою конфіденційності та, за необхідності, відкликати згоду на обробку персональних даних.

Підсумовуючи, можна зазначити, що реалізація безпечного веб-сайту передбачає комплексну відповідальність як з боку розробника, який має дотримуватись вимог технічної, правової й організаційної безпеки, так і з боку користувача, який повинен усвідомлено ставитися до захисту власної інформації та дотримуватись елементарних правил цифрової гігієни. Такий підхід дозволяє сформувати цілісне та безпечне середовище взаємодії в межах цифрового продукту, зберігаючи як працездатність розробника, так і довіру клієнтів до застосунку.

4.5. Висновок до четвертого розділу

У межах розділу було проаналізовано основні аспекти забезпечення безпечних умов праці під час реалізації кваліфікаційної роботи з розробки веб-сайту фотостудії "Yuraale". Розглянуто шкідливі та небезпечні фактори, притаманні професійній діяльності програміста, зокрема тривале статичне навантаження, зорове перенапруження, психоемоційний стрес, а також електромагнітне випромінювання та ризики, пов’язані з електроживленням обладнання. Було встановлено, що дотримання вимог охорони праці, організація ергономічного робочого місця, правильний режим праці та відпочинку, забезпечення мікроклімату і відповідного рівня освітлення дозволяють значно знизити рівень професійних ризиків і підвищити ефективність інтелектуальної праці. Особливу увагу приділено електробезпеці при роботі з комп’ютерною технікою, що є обов’язковою умовою збереження здоров’я працівника в умовах офісної або дистанційної праці. Окрім того, охоплено питання цифрової безпеки як для розробника, так і для кінцевого користувача, що взаємодіє з веб-сайтом. Визначено вимоги щодо обробки персональних даних, захисту фінансової інформації, протидії зовнішнім загрозам та дотримання юридичних норм. Таким чином, розробка програмного забезпечення повинна здійснюватися з обов’язковим урахуванням норм охорони праці, електробезпеки, санітарно-гігієнічних стандартів і принципів цифрової етики. Комплексний підхід до безпеки дозволяє не лише уникати порушень, але й забезпечує надійність, стабільність і довіру до створеного веб-сайту як з боку розробника, так і з боку користувача.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було реалізовано повнофункціональний веб-сайт фотостудії "Yuraale", який поєднує сучасний інтерфейс користувача з ефективною внутрішньою логікою адміністрування замовлень, брифування клієнтів та обробки онлайн-платежів. У процесі роботи здійснено повний цикл розробки інформаційної системи: від аналізу предметної області та постановки технічного завдання до впровадження, тестування і забезпечення вимог безпеки.

У першому розділі було проведено аналіз існуючих програмних рішень у сфері онлайн-бронювання фотопослуг, а також виявлено основні недоліки, які враховано при проєктуванні власного веб-сайту. Особливу увагу приділено сучасним вимогам до структури інформаційних систем, архітектурі клієнт-серверної взаємодії та потребам цільової аудиторії.

У другому розділі сформульовано технічне завдання на створення веб-сайту, побудовано структурно-функціональні моделі, а також розроблено ER-модель бази даних для зберігання інформації про послуги, клієнтів, замовлення, фотогалереї та оплату.

У третьому розділі реалізовано всі основні компоненти сайту: публічну частину (портфоліо, форма бронювання), адміністративну панель (керування контентом, підтвердження замовлень), інтеграцію з платіжною системою LiqPay, систему обліку брифів та авторизацію користувачів через ASP.NET Identity. Сайт побудований із використанням технологій ASP.NET MVC, Entity Framework Core та Razor Pages. Результати тестування підтвердили коректну та стабільну роботу системи.

У четвертому розділі з безпеки життєдіяльності та охорони праці досліджено потенційні шкідливі фактори, що виникають під час діяльності розробника. Надано рекомендації щодо ергономіки, електробезпеки та цифрової гігієни, а також розглянуто загальні вимоги безпеки для користувачів веб-сайту.

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено сучасний, функціональний і безпечний веб-сайт для фотостудії "Yuraale", що дозволяє автоматизувати процеси замовлення послуг, покращити клієнтський досвід та надати власнику зручні інструменти для адміністрування.

Отримані результати можуть бути використані як основа для подальшого розширення функціональності системи, зокрема: впровадження багатомовності, інтеграції з календарями бронювання, мобільної адаптації або створення повноцінного веб-порталу з розширеною клієнтською базою.

ПЕРЕЛІК ДЖЕРЕЛ

1. Основи програмування. Курс лекцій для студентів першого рівня вищої освіти за спеціальністю No 121 Інженерія програмного забезпечення/ Уклад.: М.Р. Петрик, О.Ю.Петрик - Тернопіль: ТНТУ 2018- 64 с.

2. Бабак В.П., Куц Ю.В., Мислович М.В., Фриз М.Є., Щербак Л.М. Об’єктно-орієнтована ідентифікація стохастичних шумових сигналів. Київ: Наукова думка, 2024. 240 с. DOI:10.15407/978-966-00-1883-9.

3. V. Babak, A. Zaporozhets, Y. Kuts, M. Fryz, L. Scherbak. Noise signals: Modelling and Analyses. Cham: Springer Nature Switzerland, 2025. 222 с. DOI:10.1007/978-3-031-71093-3.

4. Hilario Photography [електронний ресурс] – URL : https://hilario.co.uk/

5. PASSAGE STUDIO, м Львів [електронний ресурс] – URL : https://artstudiopassage.com/

6. Pudra, м. Київ [електронний ресурс] – URL: https://pudraphotostudio.com/

7. W Photostudio, м. Київ. [електронний ресурс] – URL : https://www.wphotostudio.com.ua/

8. Microsoft Docs — ASP.NET Core Hosting [Електронний ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/.

9. Швець В. О. Технології розробки та захисту баз даних. — К.: КНЕУ, 2020. — 248 с.

10. W3Schools. HTML/CSS/JavaScript Tutorials [Електронний ресурс] – URL: https://www.w3schools.com .

11. Boehm A., Delamater M. Murach's HTML5 and CSS3. — Mike Murach & Associates, 2015. — 672 p.

12. Ермаков С. О. Проєктування та моделювання інформаційних систем: практичний курс UML. — Харків: ХНУРЕ, 2021. — 184 с.

13. 17. Microsoft Learn. Get started with ASP.NET Core MVC [Електронний ресурс]. – 2025. – URL: https://learn.microsoft.com/en-us/aspnet/core/mvc

14. .NET Foundation. ASP.NET Core Architecture e-book [Електронний ресурс]. – 2024. – URL: https://docs.microsoft.com/en-us/dotnet/architecture/

15. Модель-вид-контролер (MVC) [Електронний ресурс] – 2024 – URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер

16. Entity Framework [Електронний ресурс] – 2024 – ULR: https://uk.wikipedia.org/wiki/Entity_Framework

17. Офіційна документація LiqPay API [Електронний ресурс] URL: https://www.liqpay.ua/documentation/en

18. Навчальний посібник «Розробка баз даних та інформаційних систем» Алісейко О.В., Чала Л.Е., Левтеров А.І., Кочуєва З.А., Плєхова Г.А., Бабенко В.О. ХНАДУ 2021 ISBN 978-966-303-775-2.

19. Introduction to Identity on ASP.NET Core [Електронний ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity

20. Чедвік Д., Хришикеш П., Снайдер Т. ASP.NET MVC 4. Розробка реальних веб-застосунків : пер. з англ. – Харків : Вільямс, 2012. – 672 с. ISBN 978-5-8459-1758-4.

21. Коноваленко І. В., Марущак П. О. Платформа .NET та мова програмування C# 8.0 : навч. посіб. – Тернопіль : ФОП Паляниця В. А., 2020. – 320 с. ISBN 978 617 7875 08 5.

22. Freeman A., Sanderson S. ASP.NET MVC 3 з прикладами на C# : пер. з англ., 3-є вид. – Київ : Вільямс, 2012. – 672 с. ISBN 978-5-8459-1758-4.

23. Freeman A. ASP.NET Core MVC 2 з прикладами на C# : 7-ме вид. – Київ : Book-Tower, 2024. – 780 с.

24. Рихтер Джеффрі.CLR via C# : програмування на платформі Microsoft .NET Framework 4.5 / пер. з англ. – К. : Діалектика, 2015. – 896 с. – ISBN 978-5-8459-1782-9.

25. EasyWeek – Програма онлайн-запису для фотографа [електронний ресурс]. – URL: https://easyweek.com.ua/crm-dlya-fotografiv.html

26. EasyWeek – Програмне забезпечення для фотостудії [електронний ресурс]. – URL: https://easyweek.com.ua/special-experiences/programne-zabezpechennya-dlya-fotostudiyi.html

27. Highload.Tech – «Що таке ASP.NET? Принцип функціонування та моделі розробки» [електронний ресурс]. – URL: https://highload.tech/2023/03/23/what-is-asp-net/

28. Філенко А., Пономарьов І. – «Система авторизації та автентифікації ASP.NET Identity» [електронний ресурс]. – URL: https://journals.nmetau.edu.ua/index.php/st/article/view/527

29. Rubarb Digital – Специфіка розробки UX/UI веб-дизайну [електронний ресурс]. – URL: https://rubarbs.com/ua/article/the-specifics-of-the-development-of-ux-ui-web-design

30. PRJCTR (Гараж) – «Що таке UI/UX-дизайн: як стати дизайнером?» [електронний ресурс]. – URL: https://prjctrmag.com/journal/gifts/vyshivanka.html

31. Goldweb Solutions – «Що таке UX і чому користувацький досвід є важливим» [електронний ресурс]. – URL: https://goldwebsolutions.com/uk/blog/shho-take-ux/

32. SEO-Evolution – «Просування сайту фотостудії: стратегії та інструменти» [електронний ресурс]. – URL: https://seo-evolution.com.ua/blog/seo-prodvizhenie/prodvizhenie-sayta-fotostudii

33. Гаєвська Ю., Монтьон І. – «Маркетингові інструменти просування фотостудії» [електронний ресурс]. – URL: https://ur.knute.edu.ua/bitstream/123456789/28263/1/Gaevska-Marketingovi-instrumenty-prosuvannya-fotostudiyi.pdf

34. Kyivstar Business Hub – «Як залучити клієнтів у фотостудію» [електронний ресурс]. – URL: https://bizhub.kyivstar.ua/blog/yak-zaluchyty-kliyentiv-u-fotostudiyu/

35. Dijix – «Аутентифікація ASP.NET Core Web API» [електронний ресурс]. – URL: https://dijix.com.ua/blog/uk/autentifikacziya-asp-net-core-web-api/

ДОДАТКИ

Додаток А

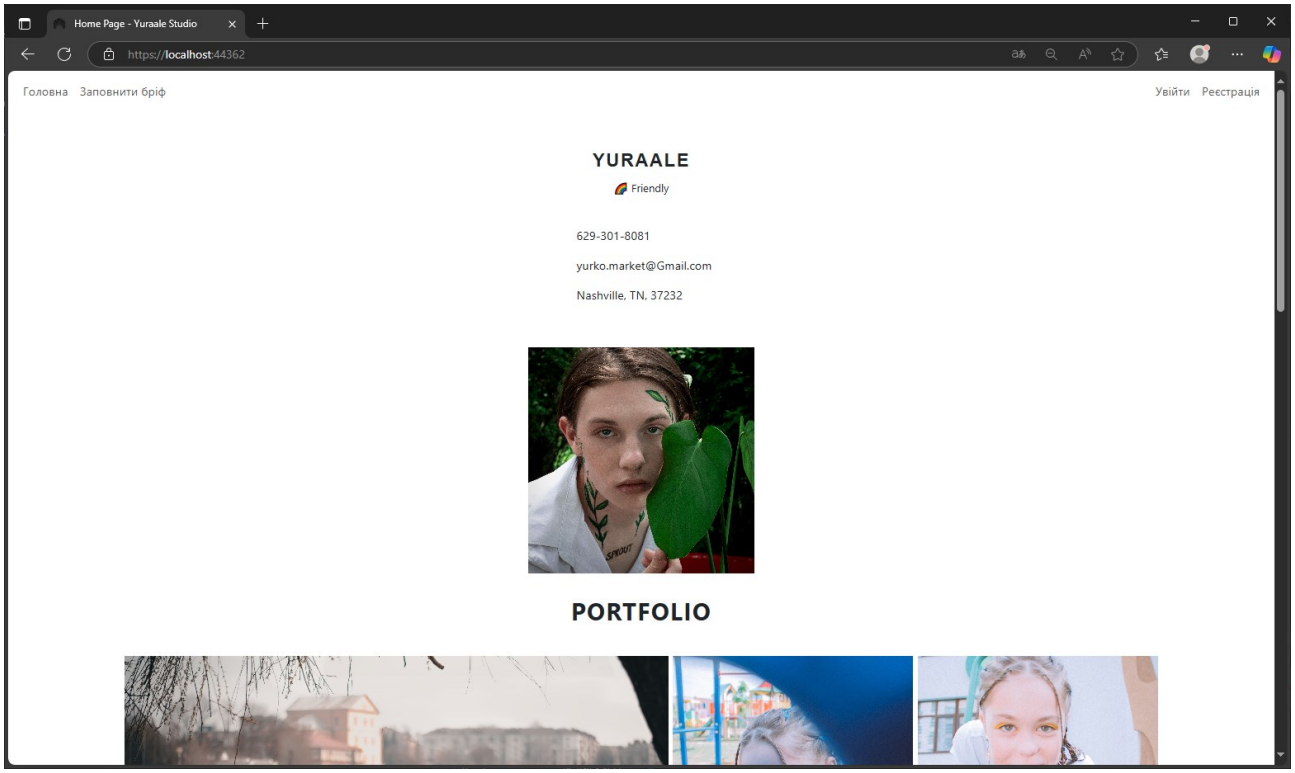
Структура головної сторінки веб-сайту.

```
@model YuraaleStudio.Web.ViewModels.HomePageViewModel
@{
    ViewData["Title"] = "Home Page";
    var address = Model?.Content?.Adress ?? null;
    var encodedAddress = System.Net.WebUtility.UrlEncode(address);
}

<link rel="stylesheet" href="/css/site.css" />

@if (Model.Content != null)
{
    <div class="text-center header__info">
        <h1 class="main_h1">@Model.Content.Title</h1>
        <h4 class="main_h4">@Model.Content.Subtitle</h4>
        <p class="main_about">@Model.Content.AboutText</p>

        <div class="main_contact">
            <div class="main_contact_info">
                <p>
                    <i class="fas fa-phone-alt"></i>
                    <a  href="tel:@Model.Content.Phone.Replace("-",
"").Replace("       ",       "").Replace("(",       "").Replace(")",
"")">@Model.Content.Phone</a>
                </p>
                <p>
                    <i class="fas fa-envelope"></i>
                    <a
href="mailto:@Model.Content.Email">@Model.Content.Email</a>
                </p>
                @if (address != null)
                {
                    <p>
                        <i class="fas fa-map-marker-alt"></i>
                        <a
href="https://www.google.com/maps/search/?api=1&query=@encodedAddr
ess" target="_blank" rel="noopener">
                            @address
                        </a>
                    </p>
                }
            </div>
        </div>

        @if (!string.IsNullOrEmpty(Model.Content.BannerImagePath))
        {
            <div class="img-home-container">
                <img           src="@Model.Content.BannerImagePath"
class="img-home-baner" />
```

```
            </div>
        }
    </div>
}

<h2 class="text-center mt-4 portf_h2">Portfolio</h2>

@foreach (var gallery in Model.Galleries)
{
    <div        id="gallery-@gallery.Id"        class="index_galery
lightgallery">
        <ul>
            @foreach (var photo in gallery.Photos)
            {
                <li>
                    <a  href="@photo.FilePath"  data-lg-size="1600-
1067">
                        <img  class="img-gal"  src="@photo.FilePath"
alt="Фото" />
                    </a>
                </li>
            }
        </ul>
    </div>
}


@section Scripts {
    <script>
        document.querySelectorAll('.lightgallery').forEach(el => {
            lightGallery(el, {
                selector: 'a',
                plugins: [lgZoom, lgThumbnail],
                download: false,
                speed: 200
            });
        });
    </script>
}
```

Додаток Б

Зображення першої сторінки після відкривання веб-сайту



Додаток В

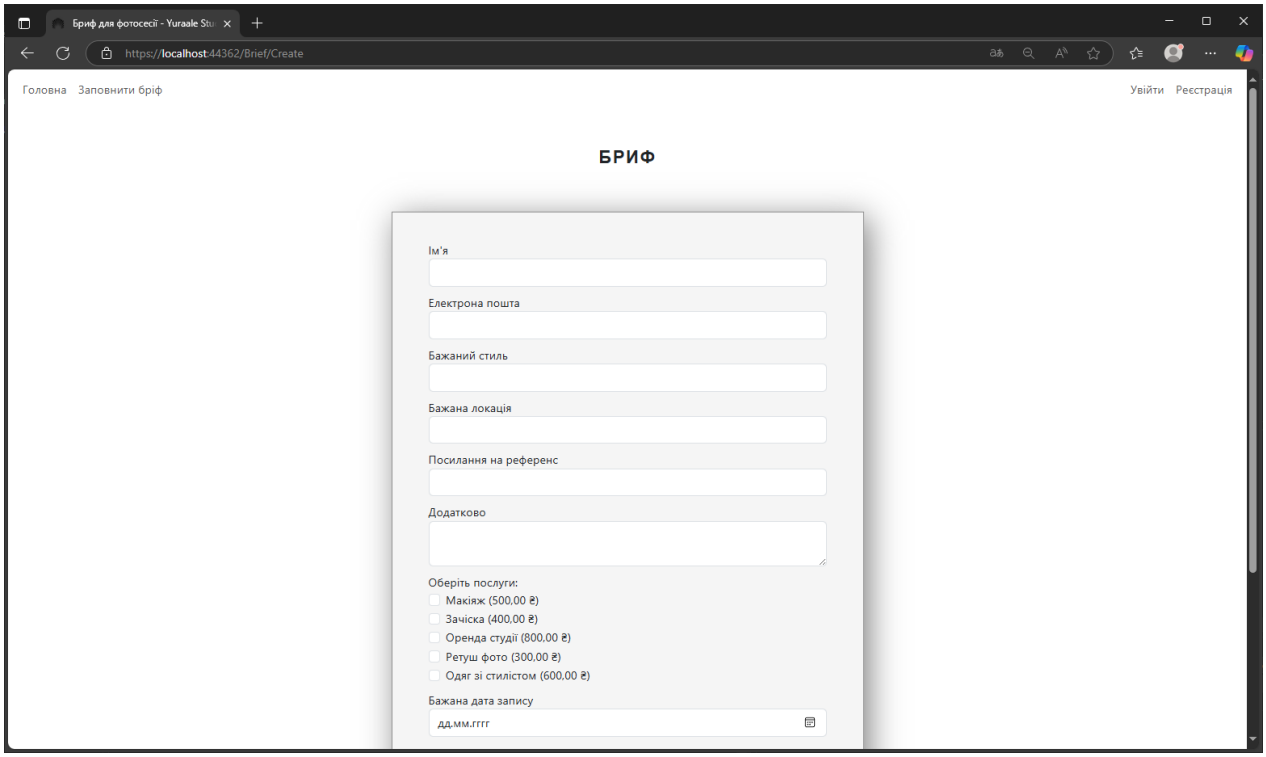
Клас відповідаючий за міграцію з базою даних

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using YuraaleStudio.Web.Models;

namespace YuraaleStudio.Web.Data
{
    public class ApplicationDbContext:
IdentityDbContext<ApplicationUser>
    {
        public
ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
            : base(options) { }

        public DbSet<Booking> Bookings { get; set; }
        public DbSet<Gallery> Galleries { get; set; }
        public DbSet<UploadedPhoto> UploadedPhotos { get; set; }
        public DbSet<HomePageContent> HomePageContents { get; set;
}
        public DbSet<BriefService> BriefServices { get; set; }
        public DbSet<Service> Services { get; set; }
        public DbSet<Brief> Briefs { get; set; }
        public DbSet<BookingStatus> BookingsStatus { get; set; }

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

            modelBuilder.Entity<Brief>().HasKey(b => b.Id);
            modelBuilder.Entity<Service>().HasKey(s => s.Id);

            // Конфигурация для связующей таблицы
            modelBuilder.Entity<BriefService>()
               .HasKey(bs => new { bs.BriefId, bs.ServiceId });

            modelBuilder.Entity<BriefService>()
               .HasOne(bs => bs.Brief)
               .WithMany(b => b.BriefServices)
               .HasForeignKey(bs => bs.BriefId)
               .OnDelete(DeleteBehavior.Cascade);    //    Добавляем
каскадное удаление

            modelBuilder.Entity<BriefService>()
               .HasOne(bs => bs.Service)
               .WithMany(s => s.BriefServices)
               .HasForeignKey(bs => bs.ServiceId)
               .OnDelete(DeleteBehavior.Cascade);
```

```
            modelBuilder.Entity<BookingStatus>().HasData(
                new BookingStatus { Id = 1, Name = "В обробці" },
                new BookingStatus { Id = 2, Name = "Прийнято" },
                new BookingStatus { Id = 3, Name = "Відхилено" }
            );

        }
    }
}
```

Додаток Д

Сторінка створення брифу



Додаток Е

Класс роботи з брифом.

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.SqlServer.Query.Internal;
using YuraaleStudio.Web.Data;
using YuraaleStudio.Web.Models;

namespace YuraaleStudio.Web.Controllers
{
    public class BriefController: Controller
    {
        private readonly BriefRepository _briefRepo;
        private readonly ApplicationDbContext _context;
        private readonly UserManager<ApplicationUser> _userManager;

        private readonly IEmailSender _emailSender;

        public BriefController(
            BriefRepository briefRepo,
            ApplicationDbContext context,
            UserManager<ApplicationUser> userManager,
            IEmailSender emailSender)
        {
            _briefRepo = briefRepo;
            _context = context;
            _userManager = userManager;
            _emailSender = emailSender;
        }

        // GET: /Brief/Create?bookingId=5
        [HttpGet]
        public IActionResult Create(int bookingId)
        {
            ViewBag.Services    =    _context.Services.Where(s    =>
!s.IsDisamble).ToList();
            var brief = new Brief { BookingId = bookingId };
            return View(brief);
        }

        // POST: /Brief/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Brief brief, int[]
SelectedServiceIds)
        {
```

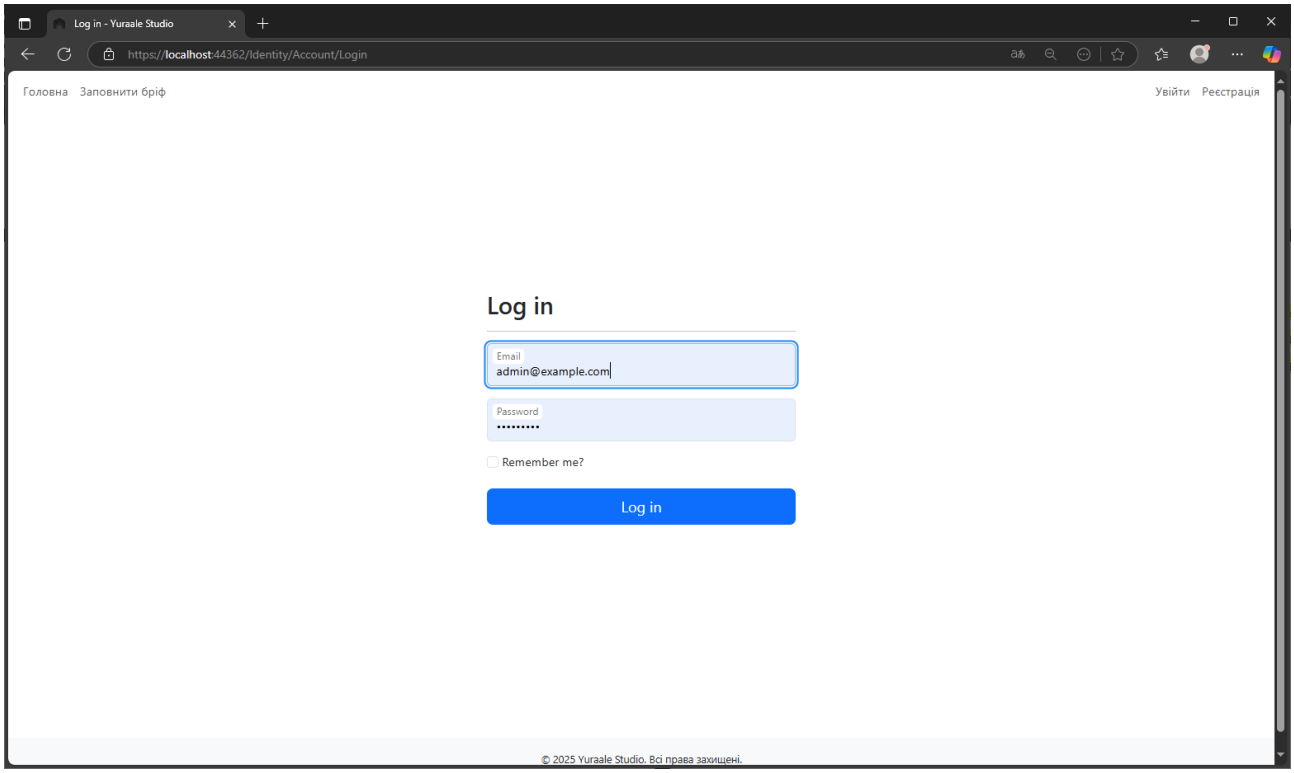
```
            if (!ModelState.IsValid)
            {
                ViewBag.Services = _context.Services.Where(s => 
!s.IsDisamble).ToList();
                return View(brief);
            }

            if (SelectedServiceIds != null && 
SelectedServiceIds.Any())
            {
                brief.BriefServices = SelectedServiceIds.Select(id 
=> new BriefService
                {
                    ServiceId = id,
                    Brief = brief
                }).ToList();
            }

            var selectedServices = await _context.Services
               .Where(s => SelectedServiceIds.Contains(s.Id))
               .ToListAsync();

            brief.BriefServices = selectedServices.Select(service 
=> new BriefService
            {
                ServiceId = service.Id,
                Brief = brief
            }).ToList();

            var booking = new Booking
            {
                ClientName = brief.ClientName,
                Email = brief.ClientEmail,
                BookingDate = DateTime.Now,
                PeopleCount = 1,
                TotalPrice = selectedServices.Sum(s => s.Price),
                Package = "Базовий",
                BookingStatusId = 1,
                IsPaid = false,
                Brief = brief
            };

            var user = await 
_userManager.FindByEmailAsync(brief.ClientEmail);
            if (user != null)
            {
                booking.UserId = user.Id;
            }
            else
            {
                var newUser = new ApplicationUser
                {
                    UserName = brief.ClientEmail,
```

```
                    Email = brief.ClientEmail,
                    EmailConfirmed = false
                };

                /*var                    password                    =
Guid.NewGuid().ToString("N").Substring(0, 10);*/
                var password = "123321";
                var            result            =            await
_userManager.CreateAsync(newUser, password);

                if (result.Succeeded)
                {
                    booking.UserId = newUser.Id;
                    var           token           =           await
_userManager.GeneratePasswordResetTokenAsync(newUser);
                    var                callbackUrl                =
Url.Page("/Account/ResetPassword", null, new { area = "Identity",
code = token, email = newUser.Email }, Request.Scheme);

                    await _emailSender.SendEmailAsync(
                        newUser.Email,
                        "Завершіть реєстрацію на Yuraale Studio",
                        $"Дякуємо  за  бронювання!  Щоб  завершити
реєстрацію,    перейдіть    за    <a    href='{callbackUrl}'>цим
посиланням</a>.");
                }
                else
                {
                    ViewBag.Services            =            await
_context.Services.Where(s => !s.IsDisamble).ToListAsync();
                    ModelState.AddModelError("",    "Не    вдалося
створити обліковий запис.");
                    return View(brief);
                }
            }


            // Додаємо до контексту
            _context.Bookings.Add(booking);
            await _context.SaveChangesAsync();

            return RedirectToAction("Success");
        }



        public IActionResult Success()
        {
            return View();
        }

        [Authorize(Roles = "admin")]
```

```
        public IActionResult Index()
        {
            var briefs = _briefRepo.GetAllBriefs();
            return View(briefs);
        }
    }
}
```

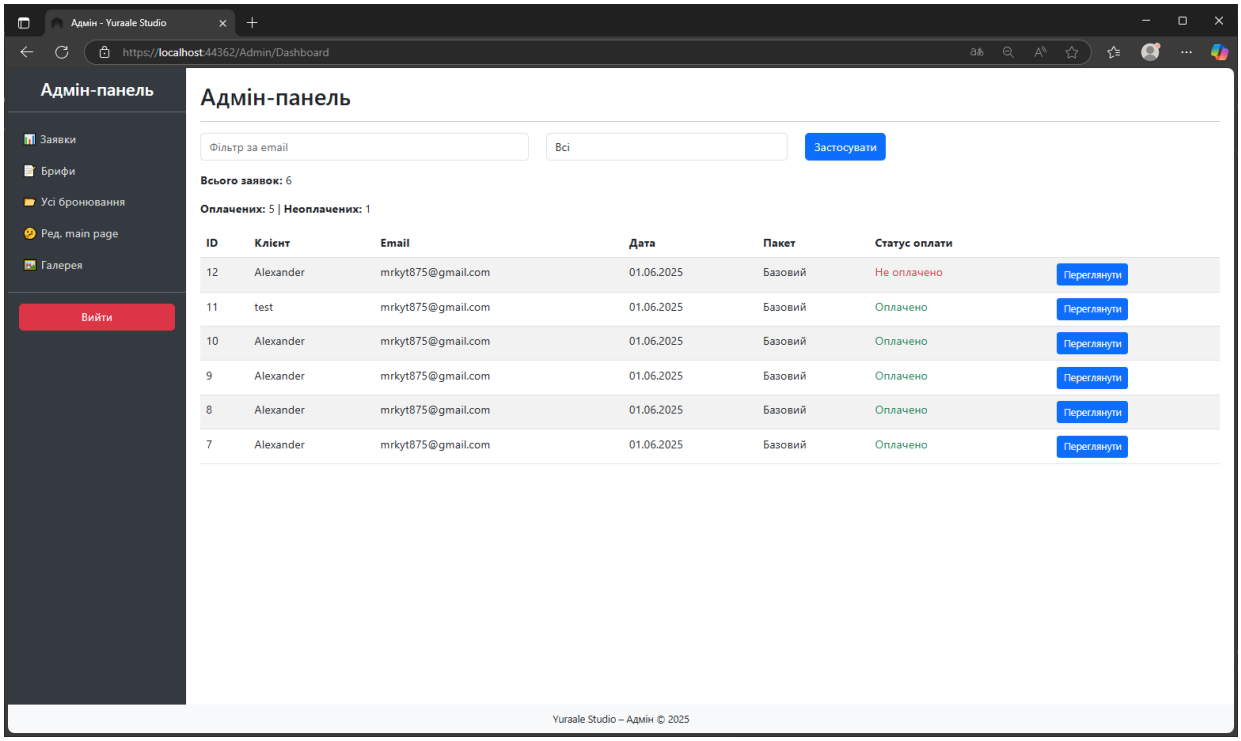
Додаток Ж

Представлення авторизації користувача.



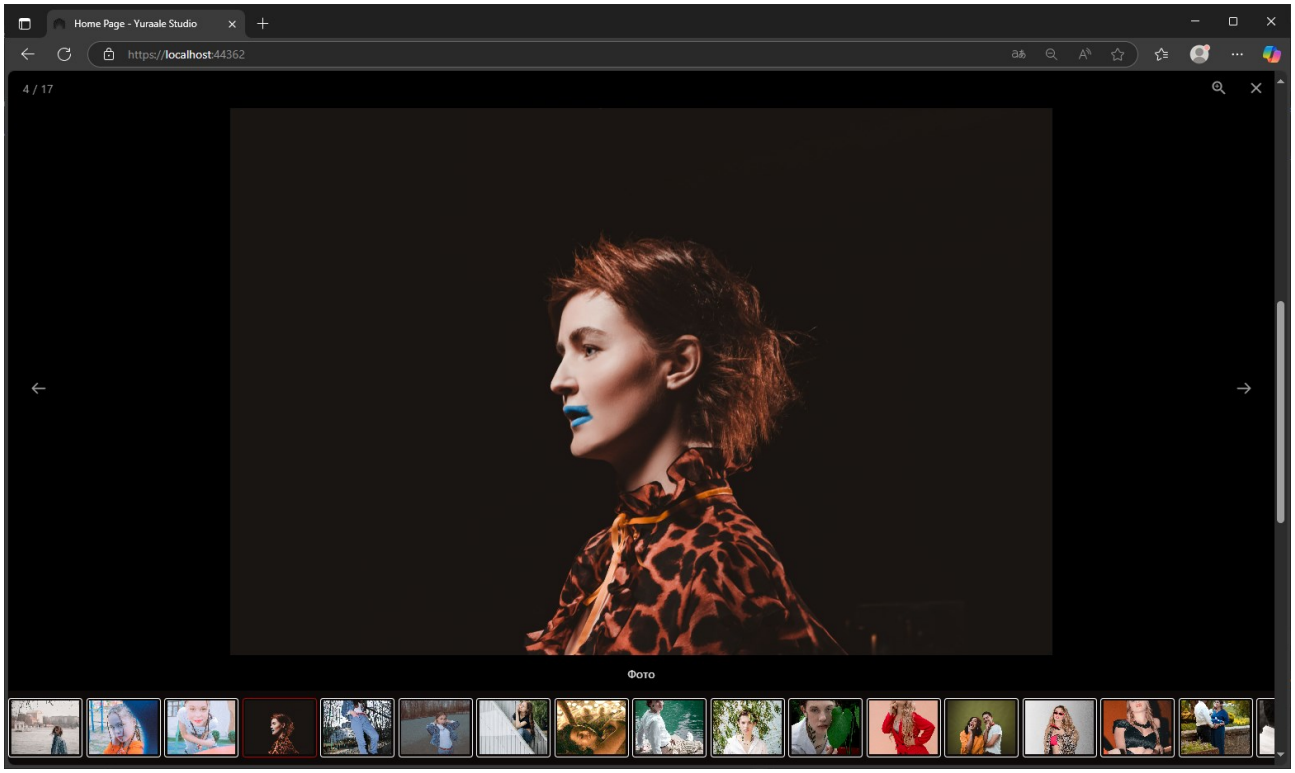
Додаток З

Представлення адмін панелі.



Додаток Л

Реалізація візуальної частини перегляду фотографій.



Додаток М

Класи підключення двох сервісів.

```
using MailKit.Net.Smtp;
using MimeKit;
using MailKit.Security;
using Microsoft.Extensions.Configuration;
using System.Threading.Tasks;

namespace YuraaleStudio.Web.Services
{
    public class EmailService
    {
        private readonly IConfiguration _config;
        public EmailService(IConfiguration config)
        {
            _config = config;
        }

        public async Task SendEmailAsync(string toEmail, string
subject, string body)
        {
            var message = new MimeMessage();

message.From.Add(MailboxAddress.Parse(_config["SmtpSettings:User"]
));
            message.To.Add(MailboxAddress.Parse(toEmail));
            message.Subject = subject;
            message.Body = new TextPart("html") { Text = body };

            using var smtp = new SmtpClient();
            await smtp.ConnectAsync(
                _config["SmtpSettings:Host"],
                int.Parse(_config["SmtpSettings:Port"]),
                SecureSocketOptions.StartTls
            );

            await smtp.AuthenticateAsync(
                _config["SmtpSettings:User"],
                _config["SmtpSettings:Password"]
            );

            await smtp.SendAsync(message);
            await smtp.DisconnectAsync(true);
        }
    }
}


using System;
using System.Security.Cryptography;
```

```
using System.Text;
using System.Text.Json;

namespace YuraaleStudio.Web.Services
{
    public class LiqPayService
    {
        private readonly string _publicKey = "sandbox_i59968869870";
        private readonly string _privateKey =
"sandbox_aVs26K7kQyYF0B7VrR8nEN2ufBn4iv2z5ccbxKfw";

        public (string data, string signature) GeneratePayment(int
bookingId, decimal amount, string email)
        {
            var orderId =
$"booking_{bookingId}_{DateTime.Now.Ticks}";
            var json = JsonSerializer.Serialize(new
            {
                version = 3,
                public_key = _publicKey,
                action = "pay",
                amount = amount,
                currency = "UAH",
                description = $"Оплата фотосесії, замовлення
#{bookingId}",
                order_id = orderId,
                result_url =
$"https://localhost:44362/Payment/Success?bookingId={bookingId}",
                server_url =
$"https://localhost:44362/Payment/Callback",
                email = email
            });

            var data =
Convert.ToBase64String(Encoding.UTF8.GetBytes(json));
            var signString = _privateKey + data + _privateKey;
            var signature =
Convert.ToBase64String(SHA1.Create().ComputeHash(Encoding.UTF8.Get
Bytes(signString)));

            return (data, signature);
        }
    }
}
using Microsoft.AspNetCore.Identity;
using YuraaleStudio.Web.Models;

namespace YuraaleStudio.Web.Services
{
    public static class RoleInitializer
    {
```

```
        public static async Task
InitializeAsync(UserManager<ApplicationUser> userManager,
RoleManager<IdentityRole> roleManager)
        {
            string adminEmail = "admin@yuraale.com";
            string password = "Admin123!";

            if (!await roleManager.RoleExistsAsync("Admin"))
                await roleManager.CreateAsync(new
IdentityRole("Admin"));

            if (!await roleManager.RoleExistsAsync("Client"))
                await roleManager.CreateAsync(new
IdentityRole("Client"));

            if (await userManager.FindByEmailAsync(adminEmail) ==
null)
            {
                var admin = new ApplicationUser { Email =
adminEmail, UserName = adminEmail, EmailConfirmed = true };
                var result = await userManager.CreateAsync(admin,
password);
                if (result.Succeeded)
                {
                    await userManager.AddToRoleAsync(admin,
"Admin");
                }
            }
        }
    }
}
```

Додаток Н

Структура згенерованої бази.