

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ БРОНІЮВАННЯ ПОСЛУГ САЛОНУ
КРАСИ**

**DEVELOPMENT OF A WEB PLATFORM FOR BOOKING BEAUTY
SALON SERVICES**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Петручик Дмитро Олександрович

Керівник:
Гордєєв Олександр Олександрович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Петручику Дмитру Олександровичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка вебплатформи для бронювання послуг салону краси»

Керівник роботи: професор Гордєєв О. О.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: React, Node.js (Express), SQLite, Tailwind CSS, методичні вказівки до виконання кваліфікаційної роботи бакалавра.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): вступ, аналіз предметної області, постановка завдань, специфікація вимог, проектування системи, вибір технологій і алгоритмів, реалізація вебдодатку, тестування та налагодження, розробка бази даних, захист системи, документація для користувача, висновки.

5. Перелік графічного матеріалу: 16 рисунків, 26 лістингів, 3 таблиці.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Гордєєв О. О.</i>		
Специфікація вимог до розробленої системи	<i>Гордєєв О. О.</i>		
Розробка об'єкта проектування	<i>Гордєєв О. О.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	<u> </u> %		
Академічна доброчесність	<i>Гордєєв О. О.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Дмитро ПЕТРУЧИК

Керівник кваліфікаційної роботи _____

Олександр ГОРДЄЄВ

АНОТАЦІЯ

Петручик Д. О. Розробка вебплатформи для бронювання послуг салону краси. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет, Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел та відповідно до структури кваліфікаційної роботи, затвердженої кафедрою.

У роботі досліджено сучасний стан вебплатформ для бронювання послуг салонів краси, проаналізовано особливості функціонування сучасних систем онлайн-запису та визначено основні проблеми існуючих програмних рішень. У процесі виконання роботи було розроблено вебплатформу для бронювання послуг салону краси з використанням React, Node.js (Express) та SQLite. Реалізовано систему онлайн-бронювання послуг, адміністративну панель для керування послугами, майстрами та записами клієнтів, механізм авторизації адміністратора, алгоритм генерації доступних часових слотів та систему скасування записів. У системі реалізовано базові механізми захисту інформації, проведено тестування програмного продукту та підготовлено документацію для встановлення й використання вебзастосунку.

Ключові слова: вебплатформа, онлайн-бронювання, салон краси, React, Node.js, Express, SQLite, REST API, frontend, backend, інформаційна система, веброзробка.

ABSTRACT

Petruchyk D. O. Development of a Web Platform for Booking Beauty Salon Services. Manuscript.

Bachelor's qualification thesis of the Educational and Professional Program "Software Engineering" in specialty 121 "Software Engineering". Lutsk National Technical University, Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three sections of conclusions and a list of sources used in accordance with the structure of the qualification work approved by the department.

The thesis examines the current state of websites for booking beauty salon services, analyzes the features of modern online booking systems, and identifies the main problems of existing software solutions. During the development process, a website for booking beauty salon services using React, Node.js (Express), and SQLite was created. The system includes online booking functionality, an administrative panel for managing services, masters, and client bookings, an administrator authorization mechanism, an algorithm for generating available time slots, and a booking cancellation system. Basic information security mechanisms were implemented, software testing was carried out, and documentation for installation and use of the web application was prepared.

Keywords: web platform, online booking, beauty salon, React, Node.js, Express, SQLite, REST API, frontend, backend, information system, web development.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО СТАНУ ВЕБПЛАТФОРМ ДЛЯ БРОНЮВАННЯ ПОСЛУГ САЛОНУ КРАСИ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ	10
1.1 Аналіз сучасного стану проблеми	10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	14
Висновки до розділу 1	15
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	16
2.1 Аналіз, визначення вимог до програмного забезпечення та проєктування системи	16
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання	25
Висновки до розділу 2	29
РОЗДІЛ 3 РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ БРОНЮВАННЯ ПОСЛУГ САЛОНУ КРАСИ.....	30
3.1 Практична реалізація об'єкта проєктування	30
3.2 Тестування та налагодження інформаційної системи.....	36
3.3 Розробка бази даних.....	41
3.4 Захист інформаційно-комп'ютерної системи.....	45
3.5 Розробка документації для програмного продукту	49
Висновки до розділу 3	52
ВИСНОВКИ.....	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	56

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації сфери обслуговування особливої актуальності набуває автоматизація процесів взаємодії між підприємствами та клієнтами. Однією з найбільш поширених тенденцій є впровадження веборієнтованих систем для онлайн-запису та бронювання послуг. Сфера краси та косметології активно використовує цифрові рішення для підвищення якості обслуговування клієнтів, оптимізації робочих процесів і покращення організації діяльності салонів краси. Використання вебплатформ для бронювання послуг дозволяє автоматизувати запис клієнтів, зменшити навантаження на адміністраторів, забезпечити зручний доступ до інформації про послуги та графік роботи майстрів, а також підвищити конкурентоспроможність закладу.

На сьогодні існує значна кількість вебресурсів і мобільних сервісів для бронювання послуг у сфері краси, проте багато з них мають обмежений функціонал, складний інтерфейс або потребують платної підписки для повноцінного використання. Часто подібні системи не враховують особливостей роботи невеликих салонів краси, мають недостатній рівень захисту даних або не забезпечують зручного адміністрування записів клієнтів. Незважаючи на активний розвиток вебтехнологій, проблема створення доступної, функціональної та зручної інформаційної системи для бронювання послуг салону краси залишається актуальною.

Світові тенденції розвитку інформаційних систем орієнтовані на створення адаптивних вебзастосунків із сучасним інтерфейсом користувача, інтеграцією баз даних, автоматизованим керуванням записами та забезпеченням безпечної обробки інформації. Важливе значення приділяється використанню клієнт-серверної архітектури, вебфреймворків, сучасних засобів розробки інтерфейсу та систем управління базами даних. Такі рішення дозволяють забезпечити швидкий доступ до сервісу з різних пристроїв, спростити процес бронювання та підвищити ефективність роботи підприємств сфери послуг.

Відповідно, актуальність теми кваліфікаційної роботи бакалавра полягає у необхідності створення сучасної вебплатформи для бронювання послуг салону краси, яка забезпечить автоматизацію процесу запису клієнтів, зручне керування послугами та підвищення ефективності роботи персоналу. Розробка подібної системи дозволить покращити якість обслуговування клієнтів, мінімізувати ймовірність помилок під час запису та забезпечити централізоване зберігання інформації про послуги й користувачів.

Метою кваліфікаційної роботи бакалавра є розробка вебплатформи для бронювання послуг салону краси з реалізацією функціоналу онлайн-запису клієнтів, керування послугами, обробки інформації про користувачів та забезпечення захисту даних.

Для досягнення поставленої мети необхідно виконати такі завдання:

- 1) провести аналіз сучасного стану вебплатформ для бронювання послуг салонів краси та визначити основні проблеми предметної області;
- 2) здійснити постановку завдання на кваліфікаційну роботу бакалавра;
- 3) проаналізувати та визначити вимоги до програмного забезпечення і спроектувати систему;
- 4) обґрунтувати вибір засобів, методів і алгоритмів реалізації вебплатформи;
- 5) виконати практичну реалізацію об'єкта проєктування;
- 6) провести тестування та налагодження інформаційно-комп'ютерної системи;
- 7) розробити базу даних для зберігання інформації про користувачів, послуги та записи;
- 8) реалізувати засоби захисту інформаційно-комп'ютерної системи;
- 9) розробити документацію для програмного продукту.

Об'єктом дослідження є процес бронювання послуг у салоні краси з використанням вебтехнологій.

Предметом дослідження є методи та засоби розробки вебплатформ для автоматизації процесу онлайн-бронювання послуг.

Практичне значення отриманих результатів полягає у створенні функціональної вебплатформи, яка може бути використана у діяльності салонів краси для автоматизації процесу запису клієнтів, керування послугами та оптимізації роботи персоналу.

Ця робота пов'язана із сучасними дослідженнями у сфері веброзробки, автоматизації інформаційних систем та цифровізації сфери обслуговування. Результати дослідження можуть бути використані під час подальшого вдосконалення систем онлайн-бронювання та створення аналогічних веборієнтованих програмних продуктів.

РОЗДІЛ 1

АНАЛІЗ СУЧАСНОГО СТАНУ ВЕБПЛАТФОРМ ДЛЯ БРОНЮВАННЯ ПОСЛУГ САЛОНУ КРАСИ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах розвитку цифрових технологій сфера послуг активно впроваджує інформаційні системи для автоматизації процесів обслуговування клієнтів. Особливо це стосується індустрії краси, де салони краси, косметологічні студії та перукарні все частіше використовують вебплатформи для організації запису клієнтів на послуги. Використання вебтехнологій дозволяє значно спростити взаємодію між клієнтом і закладом, автоматизувати процес бронювання, зменшити кількість помилок під час запису та підвищити якість обслуговування.

Проблема автоматизації процесу бронювання послуг є актуальною через постійне зростання кількості клієнтів та потребу в ефективному управлінні робочим графіком майстрів. Традиційні способи запису, зокрема телефонні дзвінки або ведення паперових журналів, мають низку недоліків. До основних проблем можна віднести втрату інформації, накладання записів, складність оперативного редагування графіків та значні витрати часу на обробку замовлень. Саме тому сучасні вебсистеми бронювання стали важливою складовою діяльності підприємств сфери обслуговування.

Аналіз наукових та інформаційних джерел показав, що проблема автоматизації діяльності салонів краси є актуальною в умовах активного розвитку цифрових технологій та веборієнтованих інформаційних систем. У сучасних дослідженнях значна увага приділяється питанням оптимізації процесів онлайн-бронювання, автоматизації керування клієнтськими записами та підвищення ефективності обслуговування користувачів. У роботі Avramets A. та Ziuziun V. розглянуто доцільність створення CRM-систем для автоматизації роботи салонів краси та обґрунтовано необхідність впровадження цифрових

сервісів у сфері обслуговування [2]. Дослідження Blancaflor E., Lopez M., Barrera G. та Halson K. присвячене розробці веборієнтованої системи керування салонними послугами, де особливу увагу приділено організації взаємодії між клієнтом та адміністратором системи [5]. У роботі Choudhary R., Akhtar E., Akhtar M., Choudhary N. R. та Rashid A. досліджено вплив автоматизації на зменшення помилок планування та скорочення часу очікування клієнтів, що підтверджує ефективність використання вебсистем онлайн-запису [13].

Окрему увагу в наукових джерелах приділено використанню сучасних вебтехнологій та інструментів розробки. У статті Ziuziun V., Myrutenko L., Myrutenko O. та Avramets A. розглянуто математичне обґрунтування створення вебплатформи для автоматизації роботи салону краси, що підтверджує перспективність використання вебтехнологій у даній предметній області [10]. У роботі Barat A. проаналізовано можливості використання штучного інтелекту для персоналізації сервісів у beauty-індустрії, що демонструє сучасні тенденції розвитку цифрових систем у сфері послуг [3].

На сьогодні існує велика кількість готових сервісів для бронювання послуг салонів краси. Серед найбільш популярних можна виділити такі системи, як EasyWeek, Beauty Pro, Dashboard Beauty та Altego. Дані вебресурси дозволяють здійснювати онлайн-запис клієнтів, перегляд доступних послуг, управління графіками працівників і ведення клієнтської бази. Однак аналіз існуючих аналогів показав, що більшість систем мають як переваги, так і певні недоліки.

EasyWeek є популярним сервісом для онлайн-бронювання послуг, який використовується підприємствами сфери обслуговування для автоматизації процесу запису клієнтів [7]. Система має сучасний та зручний інтерфейс, підтримує функцію онлайн-запису, автоматичні нагадування про візити, а також календар бронювання. Перевагою сервісу є простота використання, швидке налаштування та підтримка різних типів бізнесу, зокрема салонів краси, косметологічних студій та перукарень. Крім цього, система дозволяє працювати як адміністраторам, так і клієнтам через вебінтерфейс. Водночас значна частина розширених функцій доступна лише у платній версії, що створює певні

обмеження для невеликих закладів. Також система має недостатні можливості індивідуального налаштування інтерфейсу та функціоналу під потреби конкретного салону краси.

Dashboard Beauty є сучасною платформою для автоматизації діяльності салонів краси та організації онлайн-бронювання послуг [6]. Система забезпечує керування записами клієнтів, календарем майстрів, базою послуг та фінансовими операціями салону. Перевагою платформи є сучасний інтерфейс, підтримка онлайн-запису та зручні інструменти адміністрування. Водночас система орієнтована переважно на закордонний ринок, що ускладнює її адаптацію для локальних потреб українських користувачів, а частина розширених функцій доступна лише у платних тарифних планах.

Система Altegio орієнтована на комплексну автоматизацію бізнесу у сфері послуг та забезпечує широкий набір інструментів для управління діяльністю салону краси [1]. Вебплатформа дозволяє здійснювати управління записами клієнтів, вести аналітику роботи персоналу, контролювати фінансові показники та адмініструвати базу користувачів. Основною перевагою системи є високий рівень функціональності, підтримка інтеграції з іншими сервісами та можливість масштабування для великих підприємств. Altegio також забезпечує високий рівень автоматизації внутрішніх процесів, що позитивно впливає на ефективність роботи закладу. Проте система характеризується високою вартістю використання, що може бути недоцільним для невеликих салонів краси. Крім цього, інтерфейс містить велику кількість функцій та елементів керування, через що користування системою може бути складним для персоналу без спеціальної підготовки.

Beauty Pro є більш простою системою для автоматизації роботи невеликих салонів краси та косметологічних студій [4]. Сервіс забезпечує базовий функціонал онлайн-запису, адміністрування клієнтів і ведення інформації про послуги. Основною перевагою системи є доступність, простота використання та швидке освоєння функціоналу навіть для користувачів без значного досвіду роботи з інформаційними системами. Вебресурс дозволяє оптимізувати процес

бронювання та зменшити навантаження на адміністраторів салону. Водночас система має обмежений набір функцій у порівнянні з більш сучасними платформами, недостатній рівень масштабованості та обмежені можливості інтеграції з іншими сервісами. Це знижує ефективність використання системи для великих підприємств або салонів із великою кількістю клієнтів.

Для більш детального аналізу існуючих аналогів варто подати їх порівняльну характеристику (табл. 1.1). Проведений аналіз показує, що існуючі вебсистеми для бронювання послуг салонів краси здебільшого орієнтовані на великі комерційні проєкти та часто містять надлишковий функціонал або потребують значних фінансових витрат. Для невеликих салонів краси актуальною залишається проблема створення доступної та зручної системи, яка забезпечуватиме базові функції бронювання, адміністрування послуг і керування клієнтською базою без складного налаштування та додаткових витрат.

Таблиця 1.1 – Порівняльний аналіз існуючих систем бронювання послуг салону краси

Назва системи	Основні можливості	Переваги	Недоліки
EasyWeek	Онлайн-запис, нагадування клієнтам, календар бронювання	Зручний інтерфейс, швидка робота системи	Частина функцій платна
Dashboard Beauty	Онлайн-бронювання, клієнтська база, фінансовий облік	Великий функціонал	Складний інтерфейс
Altegio	Управління записами, аналітика, адміністрування персоналу	Комплексна автоматизація	Висока вартість
Beauty Pro	Онлайн-запис, ведення клієнтів	Простота використання	Обмежений функціонал

Отже, доцільність створення нової вебплатформи для бронювання послуг салону краси полягає у необхідності розробки простої, доступної та ефективної інформаційної системи, яка поєднуватиме сучасний вебінтерфейс, функціонал онлайн-запису, управління графіком роботи майстрів, зберігання інформації у базі даних та забезпечення захисту персональних даних користувачів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка вебплатформи для бронювання послуг салону краси, яка забезпечуватиме автоматизацію процесу запису клієнтів, зручне управління послугами, оптимізацію роботи персоналу та підвищення якості обслуговування користувачів.

Для досягнення поставленої мети у роботі необхідно виконати такі завдання:

- 1) провести аналіз сучасного стану вебплатформ та інформаційних систем для бронювання послуг салонів краси, визначити їх переваги, недоліки та основні проблеми предметної області;
- 2) здійснити постановку завдання на кваліфікаційну роботу бакалавра та визначити основні функціональні можливості майбутньої системи;
- 3) провести аналіз і визначення вимог до програмного забезпечення, а також виконати проєктування структури інформаційної системи;
- 4) обґрунтувати вибір засобів розробки, методів і алгоритмів вирішення поставленого завдання;
- 5) реалізувати практичну частину вебплатформи для бронювання послуг салону краси із забезпеченням зручного користувацького інтерфейсу;
- 6) провести тестування та налагодження інформаційно-комп'ютерної системи для перевірки стабільності та правильності її роботи;
- 7) розробити базу даних для зберігання інформації про користувачів, послуги, майстрів і записи клієнтів;
- 8) реалізувати засоби захисту інформаційно-комп'ютерної системи для забезпечення безпеки та конфіденційності даних;
- 9) розробити документацію для програмного продукту та підготувати рекомендації щодо його використання.

Практичне значення роботи полягає у створенні функціональної вебплатформи, яка може бути використана у діяльності салонів краси для

автоматизації запису клієнтів, покращення організації роботи персоналу та підвищення ефективності обслуговування користувачів.

Висновки до розділу 1

Отже, у першому розділі було проведено аналіз сучасного стану вебплатформ для бронювання послуг салонів краси та досліджено особливості автоматизації процесу онлайн-запису клієнтів. Розглянуто функціональні можливості популярних сервісів, зокрема Altegio, Beauty Pro, EasyWeek та Dashboard Beauty, визначено їх переваги й недоліки. Встановлено, що існуючі системи часто мають надлишковий функціонал, складний інтерфейс або обмеження безкоштовного використання, що підтвердило актуальність створення власної вебплатформи для автоматизації роботи салону краси.

Також було сформульовано мету та основні завдання кваліфікаційної роботи. Визначено необхідність створення вебплатформи для онлайн-бронювання послуг салону краси з можливістю перегляду послуг і майстрів, вибору доступного часу, створення та скасування записів, а також адміністрування даних системи. Сформовані завдання стали основою для подальшого проєктування, вибору технологій та практичної реалізації інформаційної системи.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до програмного забезпечення та проєктування системи

Вебплатформа для бронювання послуг салону краси потребує детального аналізу вимог до програмного забезпечення, визначення функціональних можливостей системи та проєктування її структури. Основною метою системи є автоматизація процесу онлайн-запису клієнтів, оптимізація роботи адміністраторів і майстрів, а також забезпечення швидкого та зручного доступу користувачів до послуг салону краси.

Для проєктованої вебплатформи доцільно використати клієнт-серверну модель архітектури, яка забезпечує поділ системи на фронтенд та бекенд частини. Такий підхід дозволяє забезпечити незалежність користувацького інтерфейсу від серверної логіки, спростити супровід програмного забезпечення та забезпечити можливість подальшого масштабування системи. Клієнтську частину системи слід реалізувати у вигляді SPA-застосунку з використанням бібліотеки React, а для серверної частини передбачити використання Node.js та Express. Для організації зберігання даних варто використати реляційну базу даних SQLite.

Вибір клієнт-серверної архітектури обумовлений необхідністю забезпечення віддаленого доступу до системи через мережу Internet із використанням протоколу TCP/IP. Взаємодію між клієнтською та серверною частинами доцільно організувати через REST API із використанням HTTP-запитів та формату JSON для обміну даними. У процесі аналізу предметної області визначено дві основні категорії користувачів системи: клієнти салону краси та адміністратори. Для кожної категорії користувачів визначено перелік функціональних можливостей.

До основних функцій користувача належать:

- 1) перегляд списку послуг салону краси;
- 2) перегляд інформації про майстрів;

- 3) онлайн-запис на обрану послугу;
- 4) вибір майстра;
- 5) вибір доступної дати та часу;
- 6) введення контактних даних;
- 7) підтвердження запису;
- 8) скасування запису.

Для адміністративної частини системи слід передбачити такі функціональні можливості:

- 1) авторизацію адміністратора;
- 2) перегляд записів клієнтів;
- 3) зміну статусу записів;
- 4) видалення записів;
- 5) створення та редагування послуг;
- 6) активацію та деактивацію послуг;
- 7) створення та редагування інформації про майстрів;
- 8) прив'язку майстрів до послуг;
- 9) керування відображенням даних на сайті.

Нефункціональні вимоги визначають основні характеристики якості програмного забезпечення, які повинні забезпечувати стабільність, безпечність, продуктивність та зручність використання вебплатформи для бронювання послуг салону краси. Дані вимоги описують умови функціонування системи та встановлюють обмеження щодо її роботи. Під час проєктування системи особливу увагу доцільно приділити швидкодії, захисту даних користувачів, адаптивності інтерфейсу та надійності обробки інформації.

До основних нефункціональних вимог системи належать:

- 1) забезпечення швидкої обробки запитів користувачів та мінімального часу завантаження сторінок;
- 2) забезпечення захисту персональних даних клієнтів і авторизації адміністратора;

- 3) підтримка стабільної роботи системи та запобігання конфліктам під час бронювання;
- 4) адаптивність інтерфейсу для роботи на персональних комп'ютерах, планшетах і мобільних пристроях;
- 5) підтримка роботи у сучасних веббраузерах;
- 6) забезпечення цілісності бази даних за допомогою зовнішніх ключів та SQL-обмежень;
- 7) можливість подальшого масштабування та розширення функціональних можливостей системи;
- 8) модульність і зрозуміла структура програмного коду для спрощення супроводу програмного забезпечення.

Використання клієнт-серверної архітектури, компонентного підходу React та серверної обробки даних на Node.js повинно забезпечити ефективну роботу системи та створити можливість для її подальшого розвитку й удосконалення. Для аналізу вимог до системи варто застосувати метод сценарного аналізу та підхід прецедентів використання. Основним прецедентом роботи системи є сценарій онлайн-бронювання послуги. Користувач повинен послідовно обирати послугу, майстра, дату та час запису, після чого вводити контактні дані та підтверджувати бронювання. Серверна частина системи повинна виконувати перевірку доступності обраного часу та формувати запис у базі даних.

Для моделювання функціональних можливостей вебплатформи доцільно використати UML-діаграму прецедентів використання (рис. 2.1). Дана діаграма відображає взаємодію користувачів із системою та демонструє основні функції, які повинні бути доступні клієнтам і адміністраторам. Основними акторами системи є «Клієнт» та «Адміністратор». Клієнт повинен мати можливість переглядати послуги, здійснювати бронювання та скасовувати записи, тоді як адміністратор повинен виконувати керування послугами, майстрами та записами клієнтів.

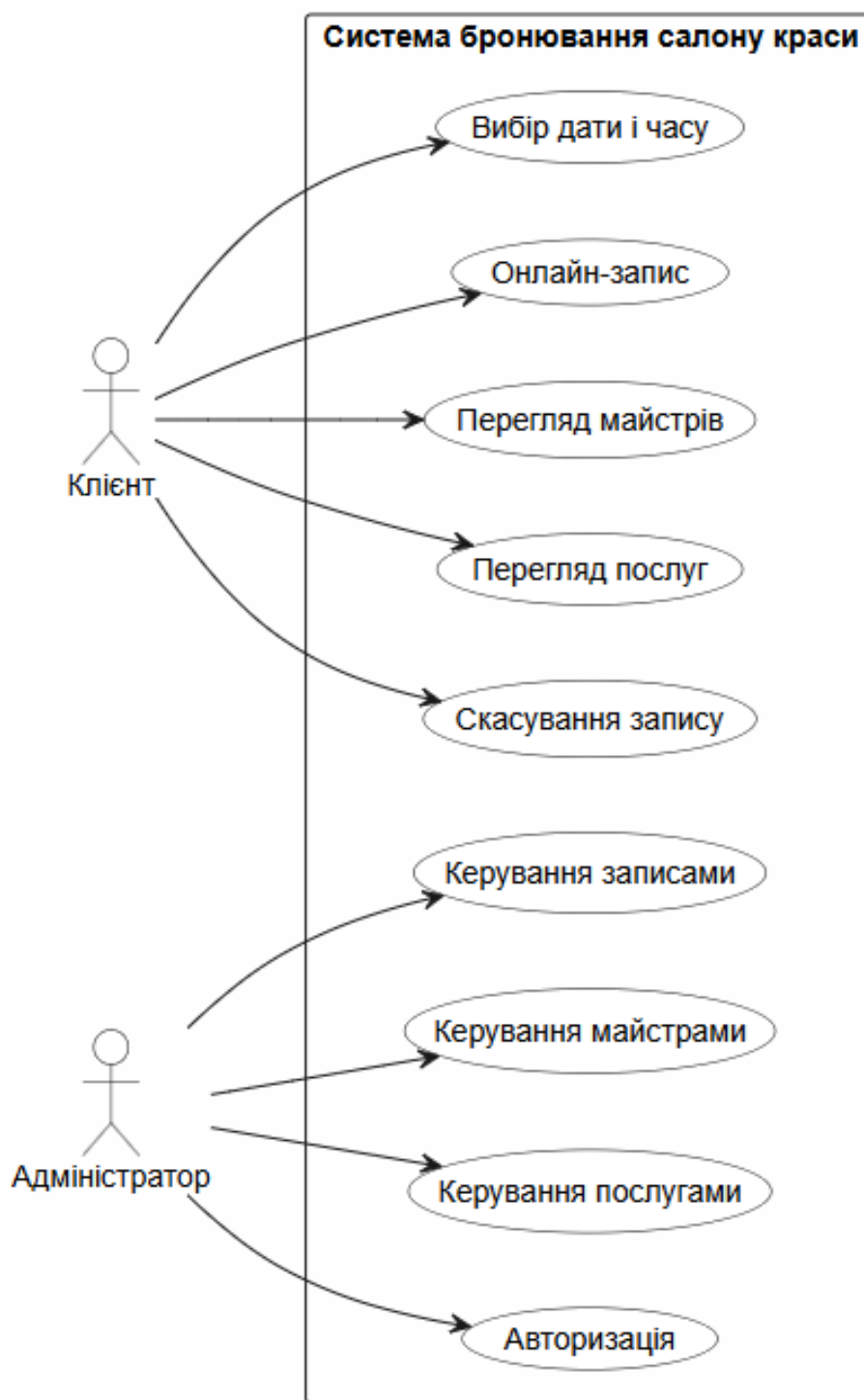


Рисунок 2.1 – UML-діаграма прецедентів використання системи

Для опису логіки роботи системи онлайн-запису слід використати UML-діаграму діяльності (рис. 2.2). Вона демонструє послідовність дій користувача під час бронювання послуги. Процес повинен починатися з вибору послуги та майстра, після чого система має відображати доступні часові слоти. Якщо

обраний час є доступним, користувач повинен ввести контактні дані та підтвердити запис. Після цього система повинна формувати новий запис у базі даних.

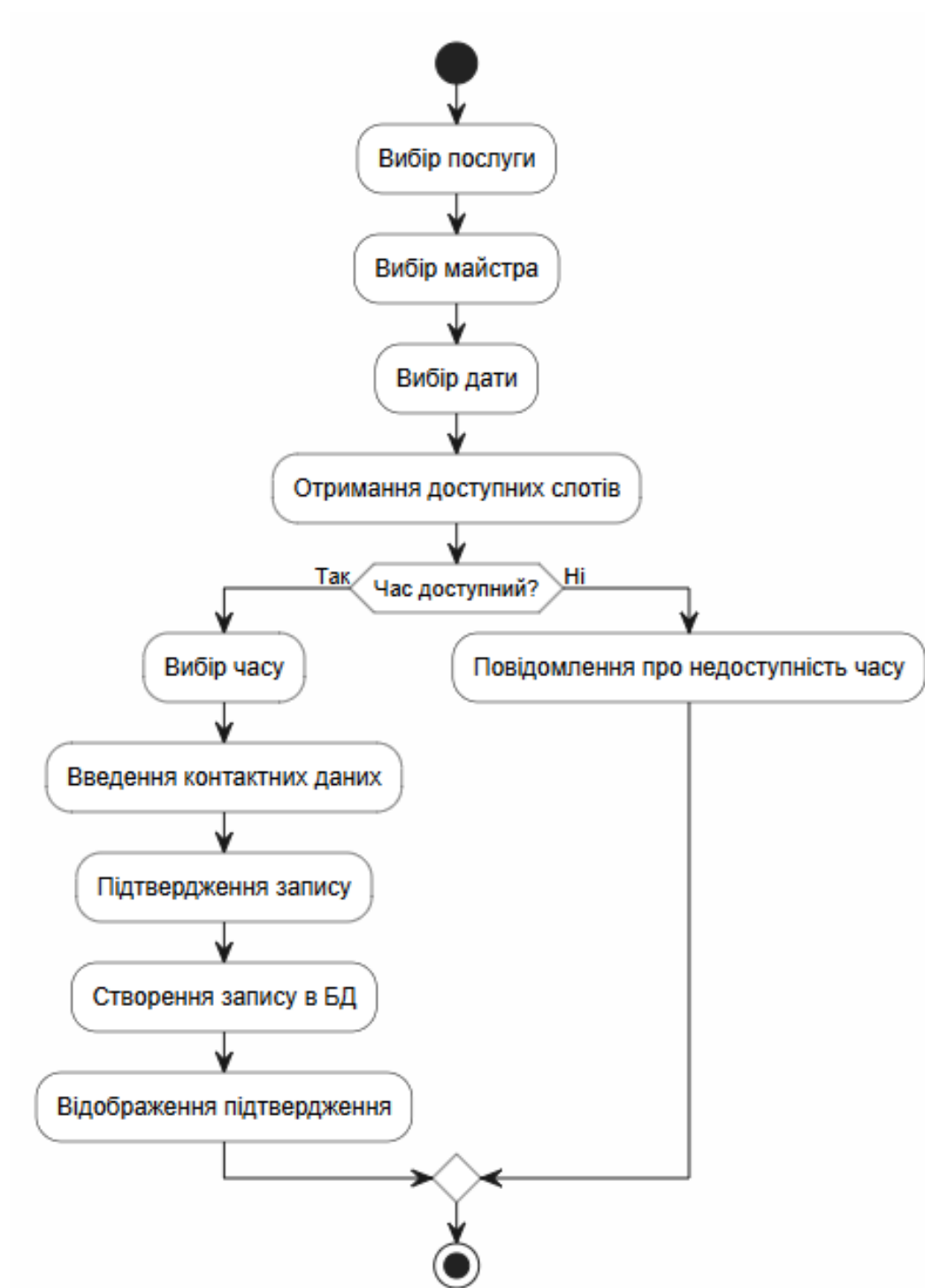


Рисунок 2.2 – UML-діаграма діяльності процесу бронювання

Для моделювання архітектури програмного забезпечення варто використати UML-діаграму компонентів (рис. 2.3). Вона демонструє структуру системи та взаємозв'язок між її основними модулями. У межах проектування передбачається поділ системи на клієнтську та серверну частини. Клієнтська частина повинна взаємодіяти із серверною через REST API, а серверна частина має забезпечувати обробку запитів, бізнес-логіку та взаємодію з базою даних.

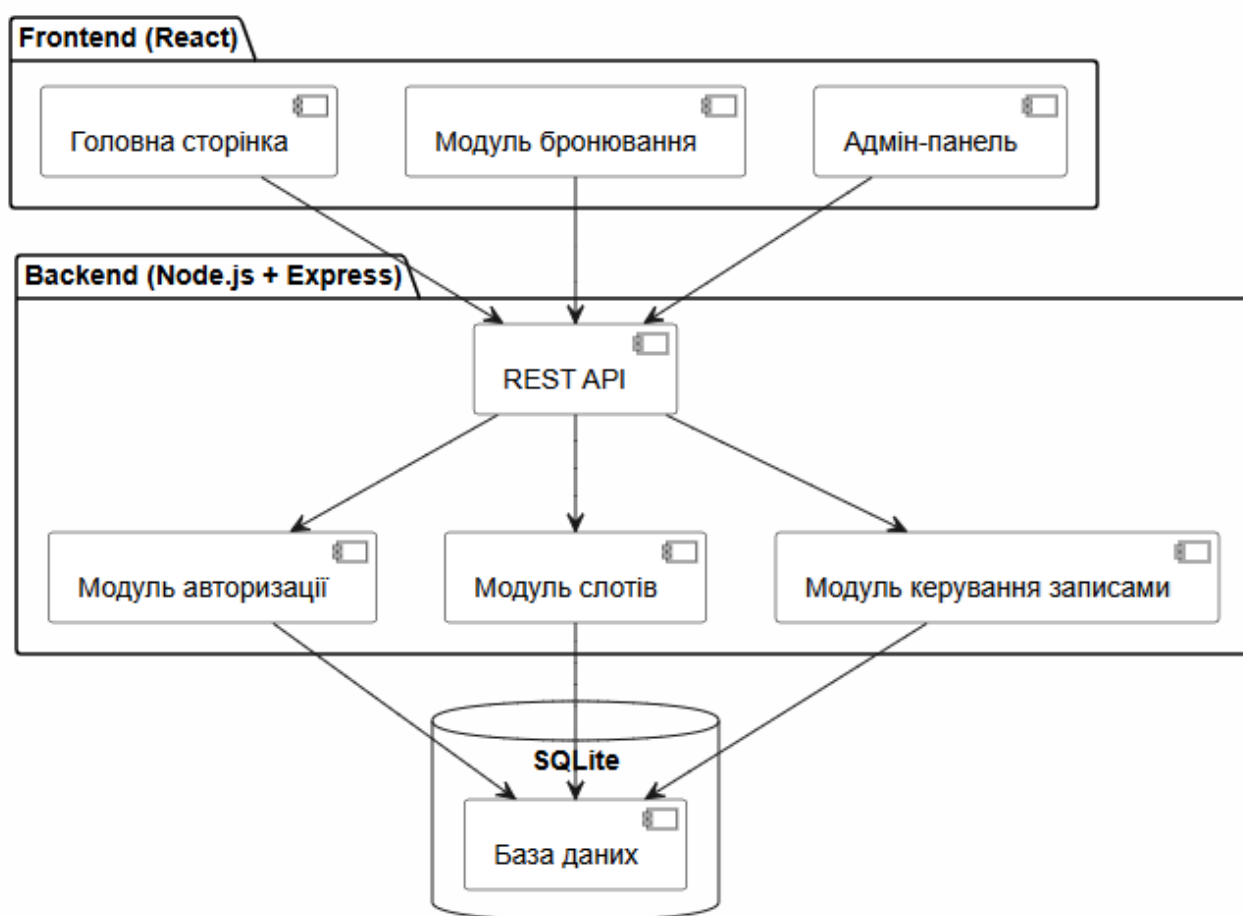


Рисунок 2.3 – UML-діаграма компонентів системи

Для формалізованого опису структури бази даних доцільно використати UML-діаграму класів (рис. 2.4). Вона демонструє основні сутності системи та зв'язки між ними. У структурі бази даних передбачається використання таблиць `services`, `masters`, `bookings`, `working_hours` та `master_services`. Зв'язок між майстрами та послугами слід реалізувати через проміжну таблицю `master_services`, що дозволить одному майстру виконувати декілька послуг.

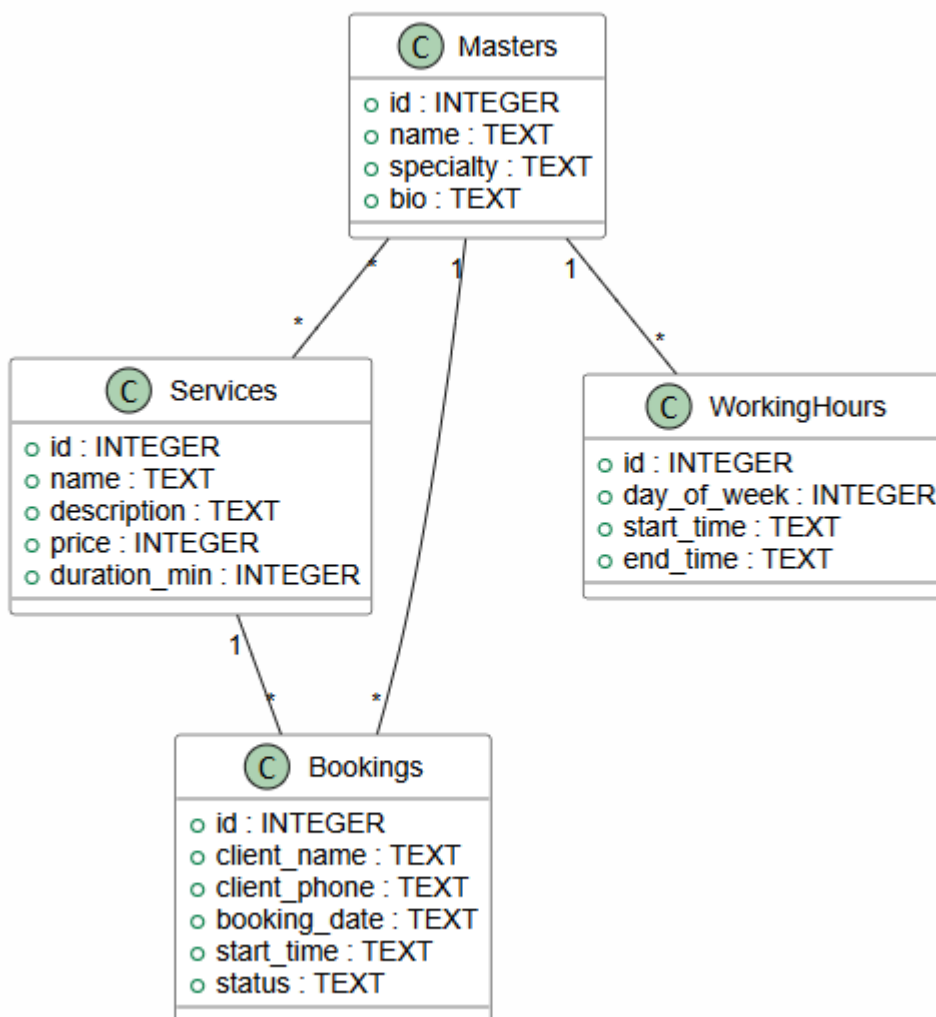


Рисунок 2.4 – UML-діаграма класів бази даних

Для демонстрації взаємодії між клієнтом, інтерфейсом користувача, серверною частиною та базою даних варто використати UML-діаграму послідовності (рис. 2.5). Вона відображає процес отримання доступних слотів часу та створення нового запису в системі. Під час проєктування системи передбачається використання багаторівневої структури програмного забезпечення, у якій клієнтська частина повинна відповідати за взаємодію з користувачем та формування інтерфейсу, а серверна – за обробку запитів, бізнес-логіку та взаємодію з базою даних. Такий підхід дозволить забезпечити чітке розмежування відповідальності між компонентами системи.

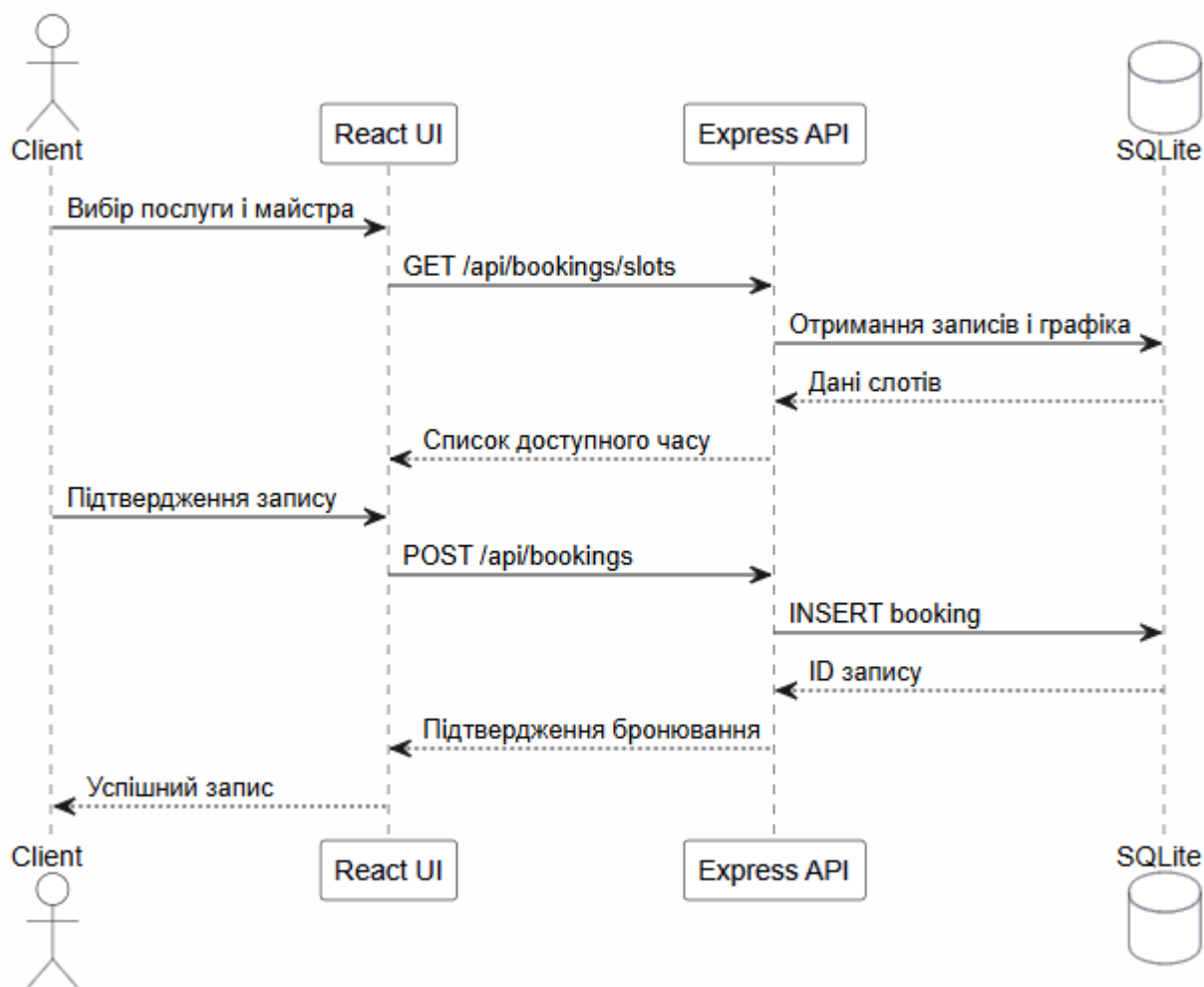


Рисунок 2.5 – UML-діаграма послідовності процесу бронювання

Особливу увагу слід приділити UX/UI-дизайну інтерфейсу користувача. Основною метою є створення простого, зрозумілого та сучасного вебінтерфейсу, який забезпечуватиме комфортну взаємодію користувача із системою. Для створення інтерфейсу доцільно використати бібліотеку Tailwind CSS, що дозволить забезпечити адаптивний дизайн, єдину кольорову стилістику та сучасний вигляд вебзастосунку.

Головна сторінка вебплатформи повинна містити навігаційне меню, інформацію про салон краси, перелік популярних послуг та кнопки швидкого переходу до бронювання. Для сторінки онлайн-запису варто передбачити покрокову систему бронювання, поділену на окремі етапи. Такий підхід повинен забезпечити зручність користування та спростити процес взаємодії користувача

із системою. В адміністративній панелі доцільно передбачити окремий інтерфейс для керування послугами, майстрами та записами клієнтів. Інтерфейс адміністратора повинен містити таблиці записів, форми створення та редагування даних, модальні вікна та засоби фільтрації інформації.

У процесі проєктування системи визначаються основні інформаційні потоки. Користувач через вебінтерфейс повинен надсилати запити на сервер для отримання списку послуг, інформації про майстрів та доступних слотів часу. Серверна частина повинна забезпечувати обробку запитів, взаємодію з базою даних та повернення результатів у форматі JSON. Під час створення запису система повинна виконувати перевірку доступності обраного часу та формувати новий запис у таблиці бронювання.

Важливою складовою системи є забезпечення захисту та безпеки даних. Для адміністративної частини потрібно передбачити механізм авторизації з використанням токенів доступу. Серверна частина повинна перевіряти права доступу користувача перед виконанням адміністративних операцій. У системі також необхідно передбачити перевірку коректності введених даних, валідацію телефонних номерів, перевірку дат та часу запису, а також контроль цілісності бази даних за допомогою зовнішніх ключів та SQL-обмежень.

Для забезпечення цілісності інформації найкраще буде використати реляційну структуру бази даних, яка повинна містити таблиці послуг, майстрів, записів клієнтів, робочих годин та зв'язків між майстрами й послугами. Використання зовнішніх ключів дозволить підтримувати узгодженість даних та запобігати виникненню помилкових записів.

Отже, у результаті аналізу предметної області та проєктування визначено функціональні можливості системи, обґрунтовано вибір клієнт-серверної архітектури та сформовано специфікацію вимог до вебплатформи бронювання послуг салону краси. Сформовані вимоги та спроектована структура системи є основою для подальшої реалізації програмного продукту.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для створення вебплатформи бронювання послуг салону краси необхідно провести аналіз сучасних засобів і технологій розробки веборієнтованих інформаційних систем. Основною метою вибору інструментів є забезпечення високої продуктивності системи, зручності користування, можливості масштабування та спрощення подальшого супроводу програмного забезпечення. Під час вибору технологій необхідно враховувати особливості предметної області, потребу у використанні клієнт-серверної архітектури, роботу з базою даних, авторизацію адміністратора та реалізацію алгоритму онлайн-бронювання.

Для створення клієнтської частини системи варто використати бібліотеку React [14] та мову програмування JavaScript [10]. Даний засіб дозволяє створювати сучасні SPA-застосунки з компонентною архітектурою та високою швидкістю оновлення інтерфейсу. Перевагами React є можливість повторного використання компонентів, ефективне керування станом інтерфейсу, підтримка адаптивного дизайну та швидке оновлення сторінок без повного перезавантаження браузера.

Для організації навігації між сторінками доцільно використати бібліотеку React Router DOM [15]. Даний інструмент дозволяє реалізувати маршрутизацію у SPA-застосунках та забезпечує зручний перехід між сторінками вебплатформи без перезавантаження браузера. Використання React Router дозволяє організувати окремі маршрути для користувацької та адміністративної частини системи.

Для стилізації інтерфейсу слід використати Tailwind CSS [18]. Основною перевагою даного інструменту є можливість швидкого створення адаптивного та сучасного інтерфейсу за допомогою utility-класів. Tailwind CSS дозволяє забезпечити єдину систему стилізації компонентів та спростити підтримку стилів у проєкті.

Для реалізації серверної частини обрано використати платформу Node.js [12] та фреймворк Express.js [9]. Node.js забезпечує ефективну обробку асинхронних запитів та високу продуктивність вебзастосунку. Express.js може бути використаний для створення REST API [16], маршрутизації запитів, обробки даних та організації взаємодії з базою даних. Перевагами Express.js є простота реалізації серверної логіки, висока швидкість роботи та велика кількість доступних модулів.

Для зберігання даних варто використати реляційну базу даних SQLite [17]. Вибір SQLite обумовлений простотою інтеграції, відсутністю необхідності налаштування окремого сервера баз даних та зручністю використання для локальних вебзастосунків. У структурі бази даних передбачається наявність таблиць послуг, майстрів, записів клієнтів, робочих годин і зв'язків між майстрами та послугами.

Для керування станом авторизації адміністратора доцільно використати Context API. Даний підхід дозволяє централізовано керувати токеном доступу та контролювати права доступу до адміністративних маршрутів. Для забезпечення захисту системи слід використати механізм токен-авторизації. Після успішної авторизації система повинна генерувати токен доступу, який може зберігатися у localStorage браузера та використовуватись для виконання адміністративних запитів. Серверна частина повинна перевіряти наявність токена перед виконанням операцій керування даними.

Для вирішення задачі онлайн-бронювання хорошим вибором буде використати алгоритм генерації доступних часових слотів. Основною задачею алгоритму є перевірка доступності часу для запису з урахуванням тривалості послуги, робочого графіка майстра та вже існуючих записів клієнтів. Алгоритм роботи системи бронювання повинен включати такі основні етапи:

- 1) отримання інформації про робочий графік майстра;
- 2) визначення тривалості обраної послуги;
- 3) отримання списку існуючих записів;
- 4) перетворення часу у хвилини;

- 5) перевірку перетину часових інтервалів;
- 6) формування списку доступних слотів;
- 7) повернення результату користувачу.

Для перевірки перетину часових проміжків варто використати алгоритм порівняння часових інтервалів. Алгоритм повинен аналізувати початок і завершення кожного запису та визначати, чи перетинається новий часовий слот із уже зайнятими проміжками часу. Такий підхід дозволить уникнути подвійного бронювання одного й того самого часу. Для моделювання структури серверної частини системи доцільно використати UML-діаграму пакетів (рис. 2.6). Вона повинна демонструвати розподіл функціональності між окремими модулями вебзастосунку.

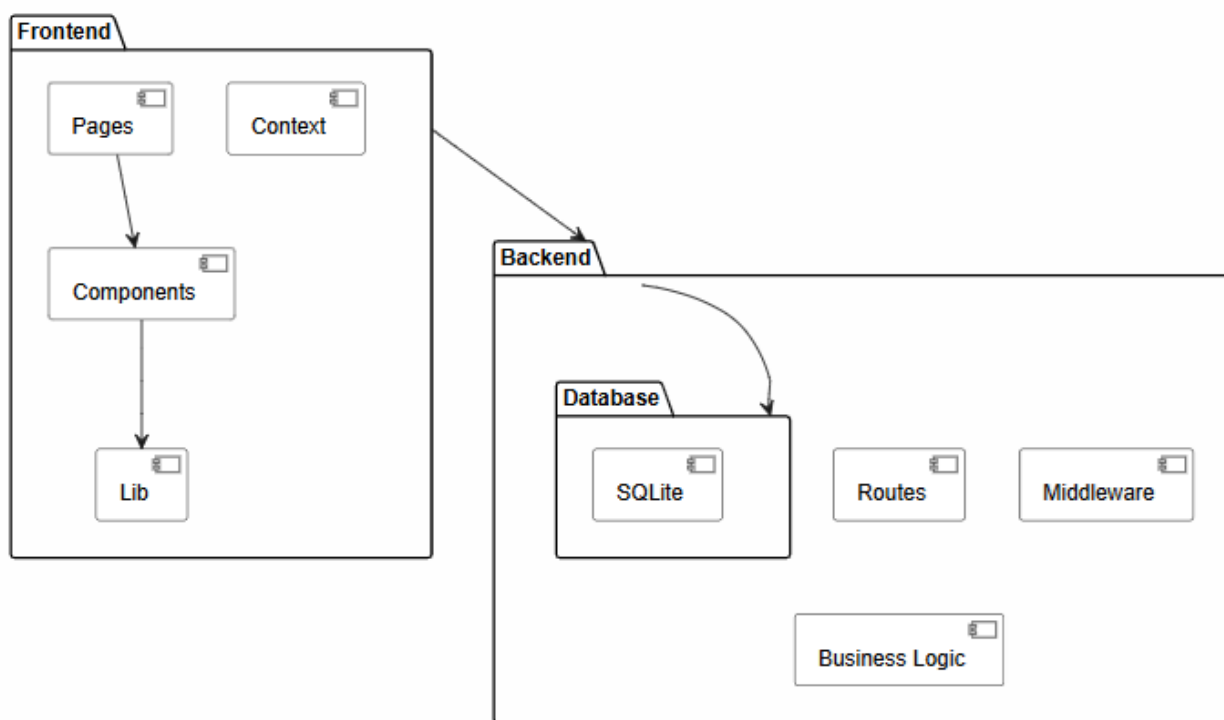


Рисунок 2.6 – UML-діаграма пакетів

Для відображення структури розгортання системи слід використати UML-діаграму розміщення (рис. 2.7).

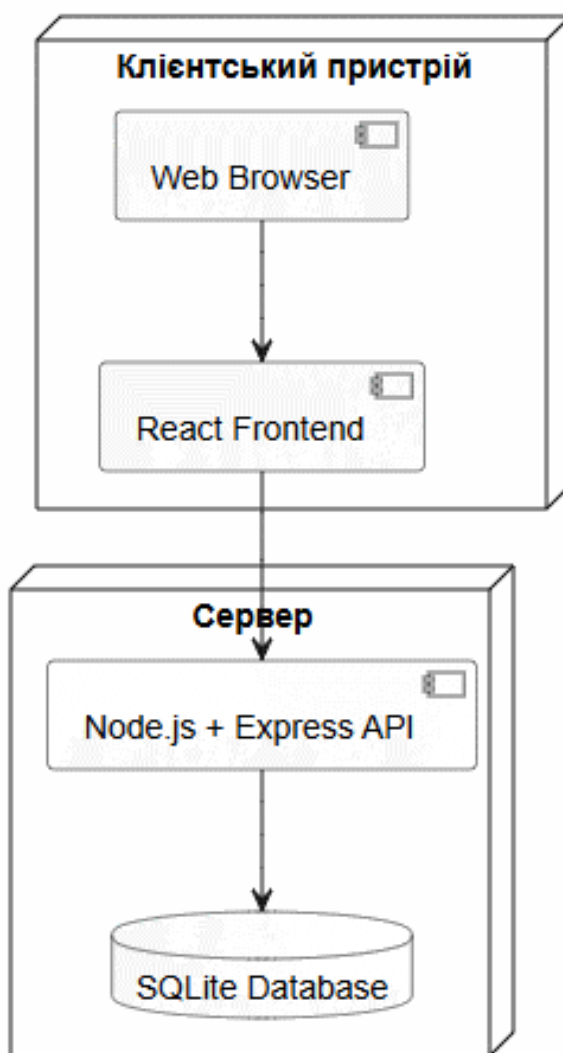


Рисунок 2.7 – UML-діаграма розміщення

Проведений аналіз засобів розробки показав, що React, Node.js, Express.js, SQLite та Tailwind CSS відповідають вимогам до створення вебплатформи бронювання послуг салону краси. Використання зазначених технологій повинно забезпечити створення сучасної інформаційної системи з адаптивним інтерфейсом, зручним процесом онлайн-бронювання та ефективною серверною обробкою даних. Запропоновані алгоритми генерації часових слотів і перевірки конфліктів записів повинні забезпечити коректну роботу системи та підтримку цілісності інформації.

Висновки до розділу 2

У другому розділі було визначено основні вимоги до вебплатформи бронювання послуг салону краси та обґрунтовано доцільність використання клієнт-серверної архітектури. У результаті аналізу предметної області сформовано функціональні та нефункціональні вимоги до системи, визначено категорії користувачів, основні інформаційні потоки та структуру взаємодії між компонентами вебплатформи. Для формалізації структури та логіки роботи системи було використано UML-діаграми прецедентів використання, діяльності, компонентів, класів, пакетів, послідовності та розміщення. Це дозволило описати основні сценарії роботи користувачів, структуру бази даних і взаємодію між клієнтською та серверною частинами системи.

У процесі аналізу засобів розробки було визначено доцільність використання React, Node.js, Express.js, SQLite та Tailwind CSS для створення вебплатформи. Обраний стек технологій відповідає вимогам щодо продуктивності, адаптивності, масштабованості та зручності супроводу програмного забезпечення. Також було визначено основні алгоритми роботи системи онлайн-бронювання, зокрема алгоритм генерації доступних часових слотів і перевірки перетину часових інтервалів. Запропоновані рішення повинні забезпечити коректну роботу системи, уникнення конфліктів під час запису клієнтів та підтримку цілісності даних.

РОЗДІЛ 3

РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ БРОНЮВАННЯ ПОСЛУГ САЛОНУ КРАСИ

3.1 Практична реалізація об'єкта проєктування

Практична реалізація вебплатформи для бронювання послуг салону краси здійснювалась із використанням сучасних вебтехнологій та клієнт-серверної архітектури. Основною метою реалізації було створення зручного вебзастосунку для онлайн-запису клієнтів, керування послугами салону та адміністрування записів через окрему адміністративну панель.

У процесі розробки клієнтської частини було використано бібліотеку React, що дозволило реалізувати SPA-застосунок із компонентною структурою. Серверна частина системи створена на платформі Node.js із використанням Express, а для зберігання інформації застосовано базу даних SQLite. Взаємодія між фронтендом та бекендом реалізована через REST API із використанням HTTP-запитів та формату JSON.

Структура програмного проєкту поділена на дві основні частини: клієнтську (client) та серверну (server). У клієнтській частині містяться React-компоненти, сторінки застосунку, модулі роботи з API та стилізація інтерфейсу. Серверна частина включає маршрути API, middleware авторизації, модулі бізнес-логіки та файли роботи з базою даних.

Одним із основних елементів системи є маршрутизація сторінок вебзастосунку. Для реалізації навігації між сторінками було використано React Router DOM. У системі реалізовано маршрути для головної сторінки, сторінки послуг, сторінки майстрів, онлайн-бронювання та адміністративної панелі (ліст. 3.1).

Лістинг 3.1 – Реалізація маршрутизації вебзастосунку

```

export default function App() {
  return (
    <Routes>
      <Route element={<Layout />}>
        <Route index element={<HomePage />} />
        <Route path="services" element={<ServicesPage />} />
        <Route path="masters" element={<MastersPage />} />
        <Route path="book" element={<BookingPage />} />
        <Route path="cancel" element={<CancelBookingPage />} />
      </Route>
      <Route path="/admin/login" element={<AdminLoginPage />} />
      <Route path="/admin" element={<AdminGuard />}>
        <Route element={<AdminLayout />}>
          <Route index element={<Navigate to="bookings" replace />} />
          <Route path="bookings" element={<AdminBookingsPage />} />
          <Route path="services" element={<AdminServicesPage />} />
          <Route path="masters" element={<AdminMastersPage />} />
        </Route>
      </Route>
    </Routes>
  );
}

```

кінець лістингу 3.1

У системі реалізовано механізм авторизації адміністратора, який забезпечує захист адміністративної панелі від несанкціонованого доступу. Після успішного введення логіна та пароля сервер генерує токен доступу, який використовується для подальших адміністративних операцій (ліст. 3.2).

Лістинг 3.2 – Middleware перевірки авторизації адміністратора

```

if (!header?.startsWith('Bearer ')) {
  return res.status(401).json({
    error: 'Потрібна авторизація'
  });
}
if (!expires || Date.now() > expires) {
  sessions.delete(token);

  return res.status(401).json({
    error: 'Сесія закінчилась. Увійдіть знову.'
  });
}

```

кінець лістингу 3.2

Одним із основних модулів системи є механізм онлайн-бронювання. Користувач має можливість обрати послугу, майстра, дату та доступний часовий слот. Для реалізації цієї функціональності використовується алгоритм генерації доступних часових інтервалів (ліст. 3.3).

Лістинг 3.3 – Алгоритм перевірки перетину часових слотів

```
export function rangesOverlap(startA, endA, startB, endB) {
  return startA < endB && startB < endA;
}
```

кінець лістингу 3.3

Алгоритм формування доступних слотів часу аналізує робочий графік майстра, тривалість послуги та вже існуючі записи клієнтів. У випадку перетину часових проміжків слот не додається до списку доступних для бронювання (ліст 3.4).

Лістинг 3.4 – Генерація доступних слотів часу

```
for (
  let start = workStart;
  start + duration <= workEnd;
  start += SLOT_STEP_MIN
) {
  const end = start + duration;

  const taken = occupied.some((o) =>
    rangesOverlap(start, end, o.start, o.end)
  );

  if (!taken) {
    slots.push(minutesToTime(start));
  }
}
```

кінець лістингу 3.4

Для зберігання інформації про послуги, майстрів і записи клієнтів було створено реляційну базу даних SQLite. Структура бази даних містить таблиці services, masters, bookings, working_hours та master_services (ліст. 3.5).

Лістинг 3.5 – Створення таблиці записів клієнтів

```
CREATE TABLE IF NOT EXISTS bookings (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  service_id INTEGER NOT NULL REFERENCES services(id),
  master_id INTEGER NOT NULL REFERENCES masters(id),
  client_name TEXT NOT NULL,
  client_phone TEXT NOT NULL,
  client_email TEXT,
  booking_date TEXT NOT NULL,
  start_time TEXT NOT NULL,
  status TEXT NOT NULL DEFAULT 'confirmed'
  CHECK (      status IN ('pending', 'confirmed', 'cancelled',
'completed') ),  created_at TEXT NOT NULL DEFAULT (datetime('now')));
```

кінець лістингу 3.5

Для реалізації сучасного та адаптивного інтерфейсу було використано Tailwind CSS. Система підтримує адаптивний дизайн та забезпечує коректне відображення вебплатформи на різних типах пристроїв (ліст. 3.6).

Лістинг 3.6 – Налаштування кольорової теми Tailwind CSS

```
@theme {
  --color-primary: #00a99d;
  --color-primary-dark: #008f85;
  --color-primary-light: #e6f7f5;
  --color-dark: #2d2d2d;
  --color-surface: #ffffff;
  --color-bg: #f5f7f7;
  --color-border: #e2e8e8;
}
```

кінець лістингу 3.6

На рисунку 3.1 представлено головну сторінку розробленої вебплатформи салону краси Lumière Salon. Інтерфейс містить навігаційне меню, основний інформаційний блок та кнопки швидкого переходу до перегляду послуг і онлайн-бронювання.

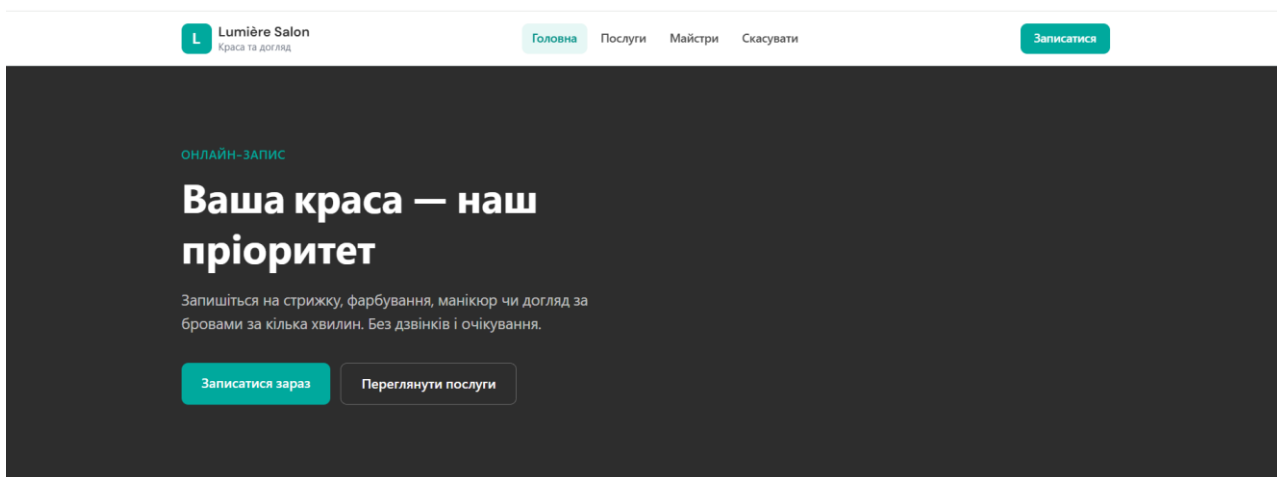


Рисунок 3.1 – Головна сторінка

На рисунку 3.2 зображено сторінку онлайн-запису, на якій користувач може обрати необхідну послугу салону краси. Для кожної послуги відображається її назва, короткий опис, вартість та тривалість виконання.

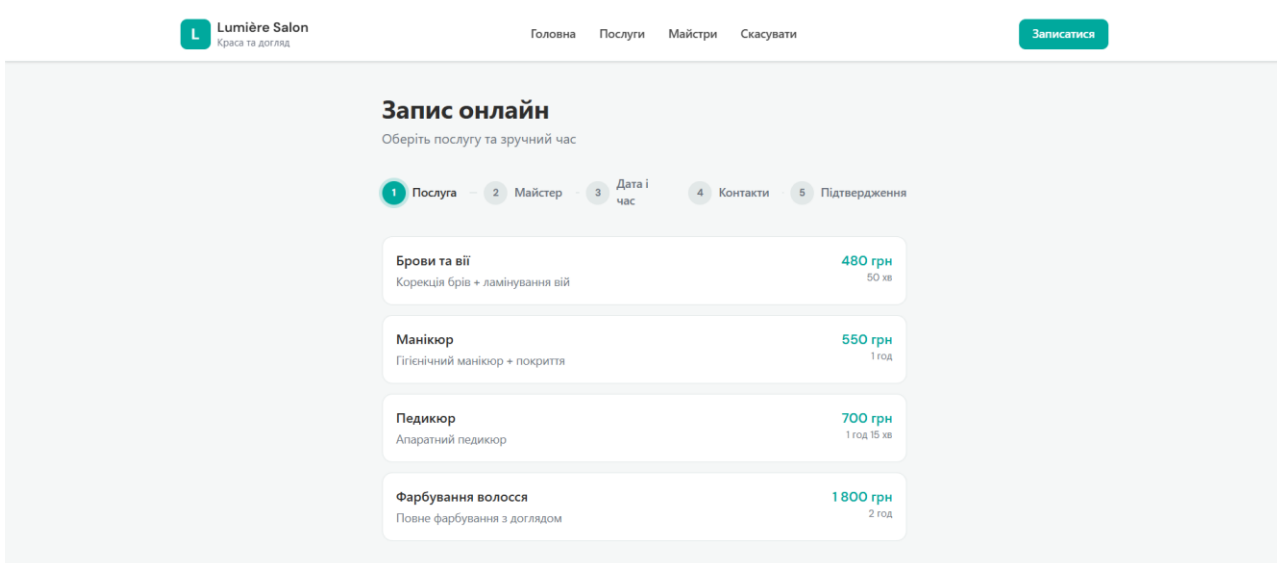


Рисунок 3.2 – Сторінка онлайн-бронювання

На рисунку 3.3 продемонстровано інтерфейс вибору дати та часу бронювання. Система автоматично формує список доступних часових слотів відповідно до графіка роботи майстра та наявних записів.

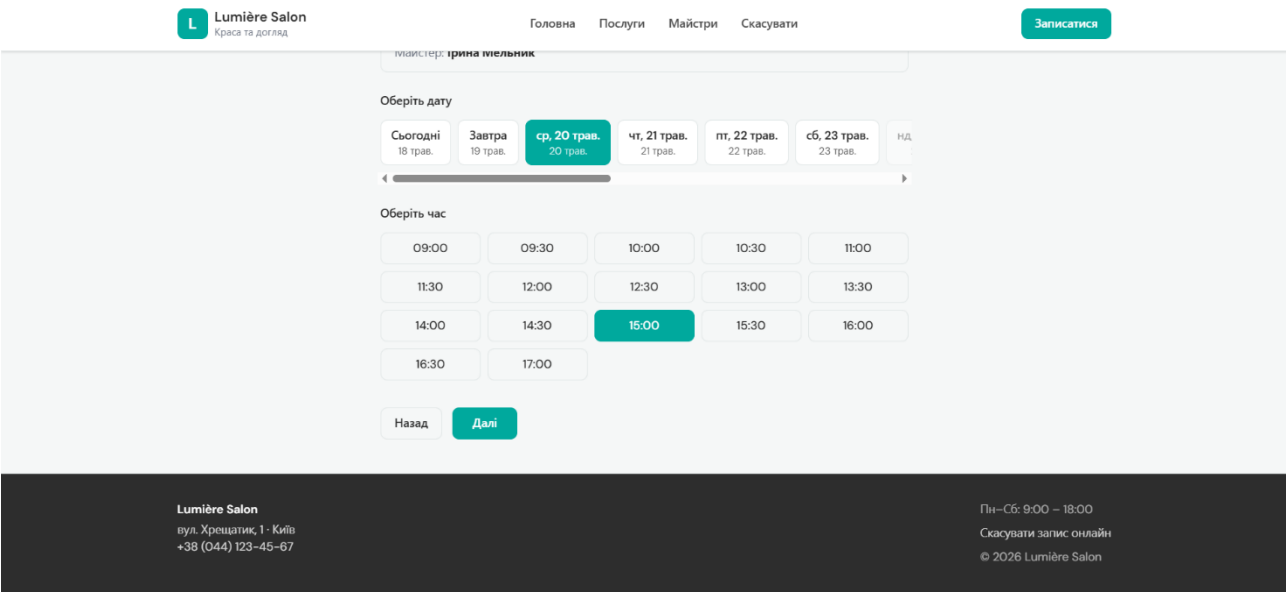


Рисунок 3.3 – Вибір доступного часу для запису

На рисунку 3.4 показано адміністративну панель вебзастосунку, яка використовується для перегляду та керування записами клієнтів. Адміністратор має можливість переглядати інформацію про бронювання, змінювати статус записів та виконувати адміністративні операції.

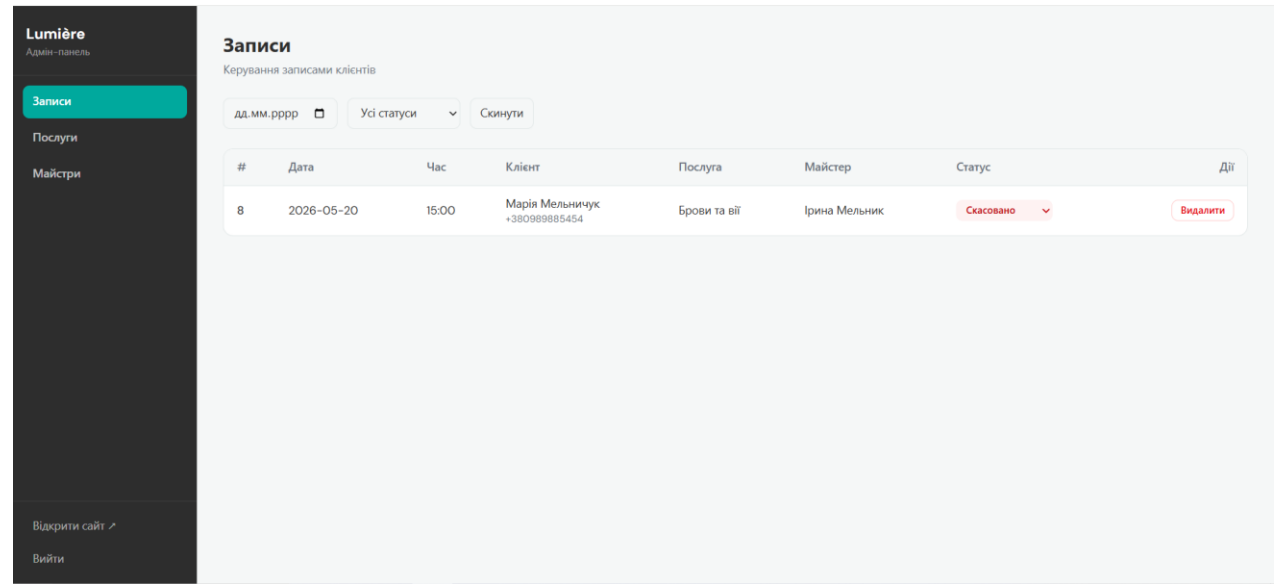


Рисунок 3.4 – Адміністративна панель системи

На рисунку 3.5 представлено форму додавання нової послуги в адміністративній панелі системи. Форма дозволяє вводити назву послуги, опис, вартість, тривалість виконання та змінювати статус активності послуги.

The screenshot displays the 'Lumière' admin interface. On the left is a dark sidebar with navigation links: 'Записи', 'Послуги' (highlighted), and 'Майстри'. The main area is titled 'Послуги' (Services) and 'Каталог послуг салону' (Salon service catalog). It lists several services like 'Брови та вій' (Eyebrows and eyelashes), 'Манікюр' (Manicure), 'Педикюр' (Pedicure), 'Фарбування волосся' (Hair coloring), and 'Жіноча стрижка' (Women's haircut). A modal window titled 'Нова послуга' (New service) is open in the center, containing fields for 'Назва' (Name), 'Опис' (Description), 'Ціна (грн)' (Price in UAH), and 'Тривалість (хв)' (Duration in minutes). It also has a checkbox for 'Активна (видима на сайті)' (Active (visible on site)) and a 'Зберегти' (Save) button. In the background, each service card has 'Редагувати' (Edit) and 'Деактивувати' (Deactivate) buttons.

Рисунок 3.5 – Форма створення нової послуги

Реалізована система забезпечує стабільну роботу, зручний інтерфейс користувача та можливість подальшого розширення функціональних можливостей.

3.2 Тестування та налагодження інформаційної системи

Під час розробки вебплатформи для бронювання послуг салону краси особлива увага приділялась тестуванню та налагодженню інформаційно-комп'ютерної системи. Основною метою тестування було забезпечення стабільної роботи програмного забезпечення, перевірка правильності реалізації функціональних можливостей, виявлення помилок та забезпечення коректної взаємодії між клієнтською та серверною частинами системи.

У процесі розробки застосовувались методи функціонального, модульного та інтеграційного тестування. Функціональне тестування використовувалось для

перевірки коректності роботи основних можливостей вебзастосунку, зокрема онлайн-бронювання, авторизації адміністратора, керування послугами та скасування записів. Модульне тестування дозволило окремо перевіряти роботу серверних функцій і компонентів інтерфейсу користувача. Інтеграційне тестування застосовувалось для перевірки взаємодії між фронтендом, серверною частиною та базою даних. На етапі тестування було перевірено коректність роботи REST API та передачі даних між клієнтською та серверною частинами системи. Особлива увага приділялась перевірці правильності обробки HTTP-запитів, валідації введених даних та обробці помилок. Для тестування API використовувались браузерні інструменти розробника та локальний запуск серверної частини системи.

Одним із основних етапів тестування стала перевірка алгоритму онлайн-бронювання. Під час тестування було перевірено коректність генерації доступних часових слотів, правильність роботи алгоритму перевірки перетину часових інтервалів та неможливість подвійного бронювання одного й того самого часу. Також було протестовано механізм вибору дати та часу запису, створення нових бронювань і функцію скасування записів клієнтів. Результати тестування підтвердили стабільну роботу алгоритму та коректне формування доступного часу для бронювання.

На рисунку 3.6 продемонстровано перевірку доступності майстрів у процесі онлайн-запису. Система автоматично блокує можливість вибору неактивного або недоступного майстра, що унеможливорює створення некоректного бронювання та забезпечує актуальність інформації для користувача.

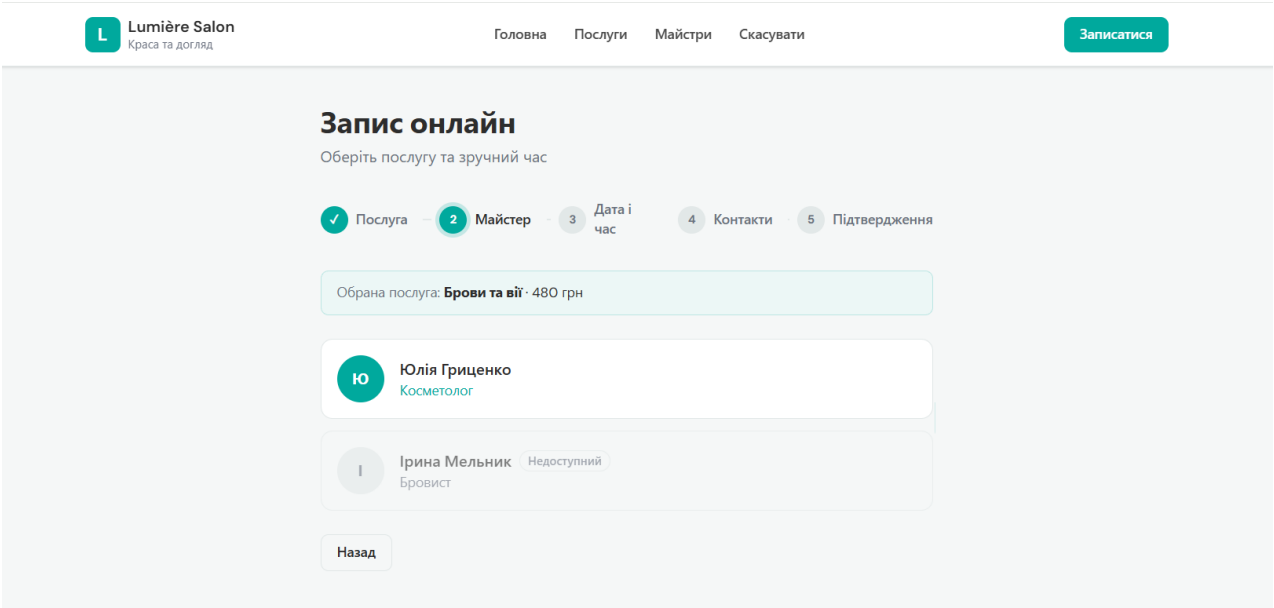


Рисунок 3.6 – Перевірка доступності майстрів під час бронювання.

На рисунку 3.7 представлено механізм перевірки доступності часу для запису. Система автоматично відображає лише вільні часові слоти відповідно до графіка роботи майстра та наявних бронювань, а недоступний час не може бути обраний користувачем.

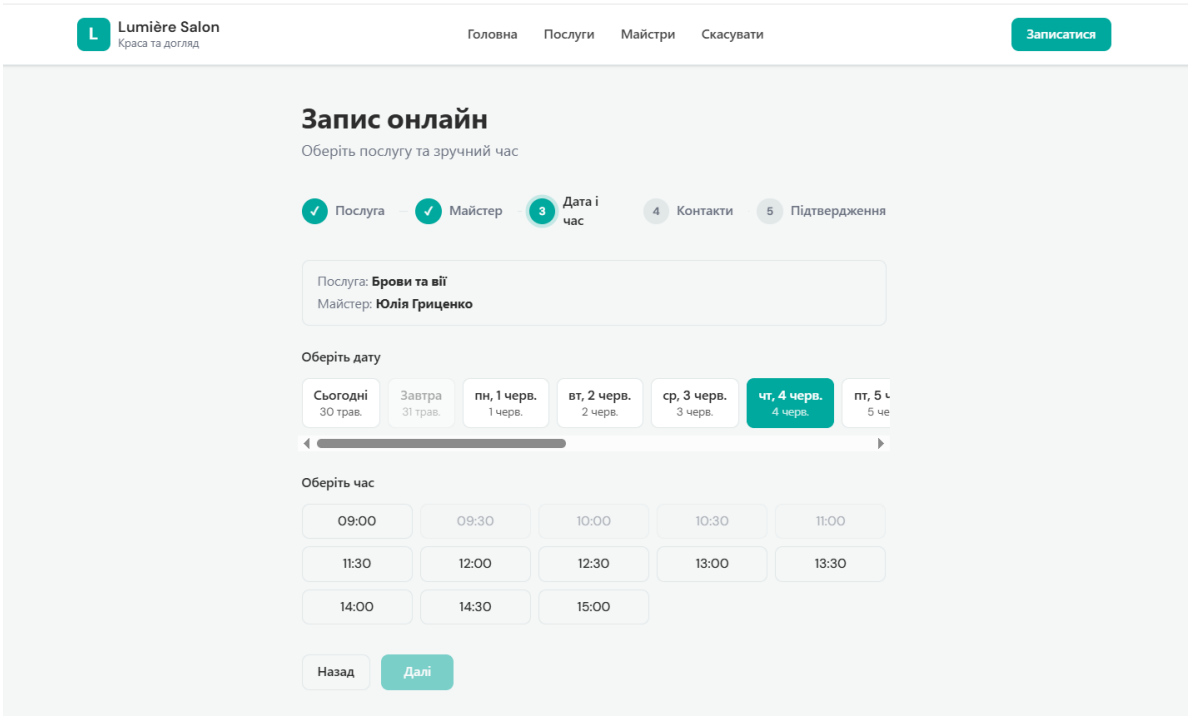


Рисунок 3.7 – Перевірка доступності часових слотів

У процесі тестування серверної частини системи особлива увага приділялась перевірці механізму авторизації адміністратора. Було перевірено правильність входу до адміністративної панелі, роботу токена доступу та захист адміністративних маршрутів від несанкціонованого доступу (рис. 3.8). Крім цього, тестувалась функція автоматичного завершення сесії при використанні недійсного токена та перевірялась неможливість виконання адміністративних операцій неавторизованими користувачами. Проведене тестування підтвердило коректну роботу механізму авторизації та належний рівень захисту адміністративної частини вебзастосунку.

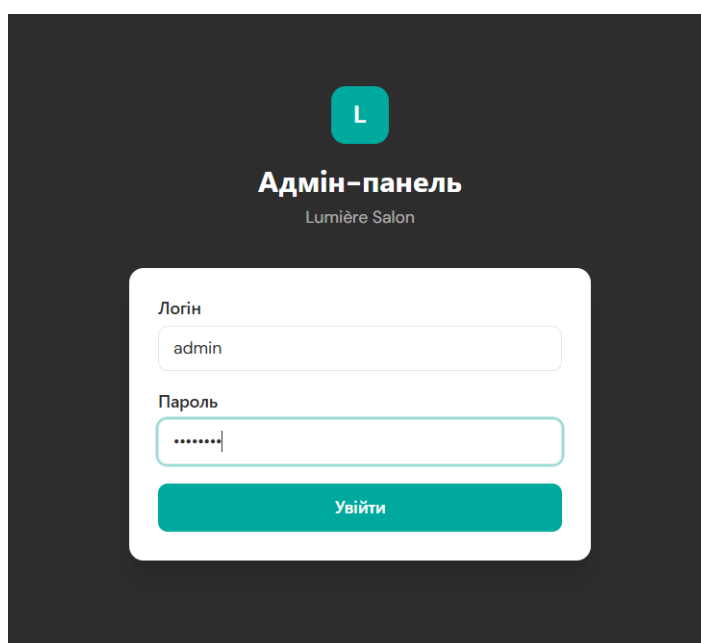


Рисунок 3.8 – Доступ до адміністративного функціоналу можливий лише після авторизації адміністратора

Під час тестування користувацького інтерфейсу перевірялась адаптивність вебзастосунку та коректність відображення сторінок на різних роздільних здатностях екрана (рис. 3.9). Інтерфейс тестувався у браузерях Google Chrome, Mozilla Firefox та Microsoft Edge. Результати тестування показали коректне відображення елементів інтерфейсу та стабільну роботу навігації між сторінками.

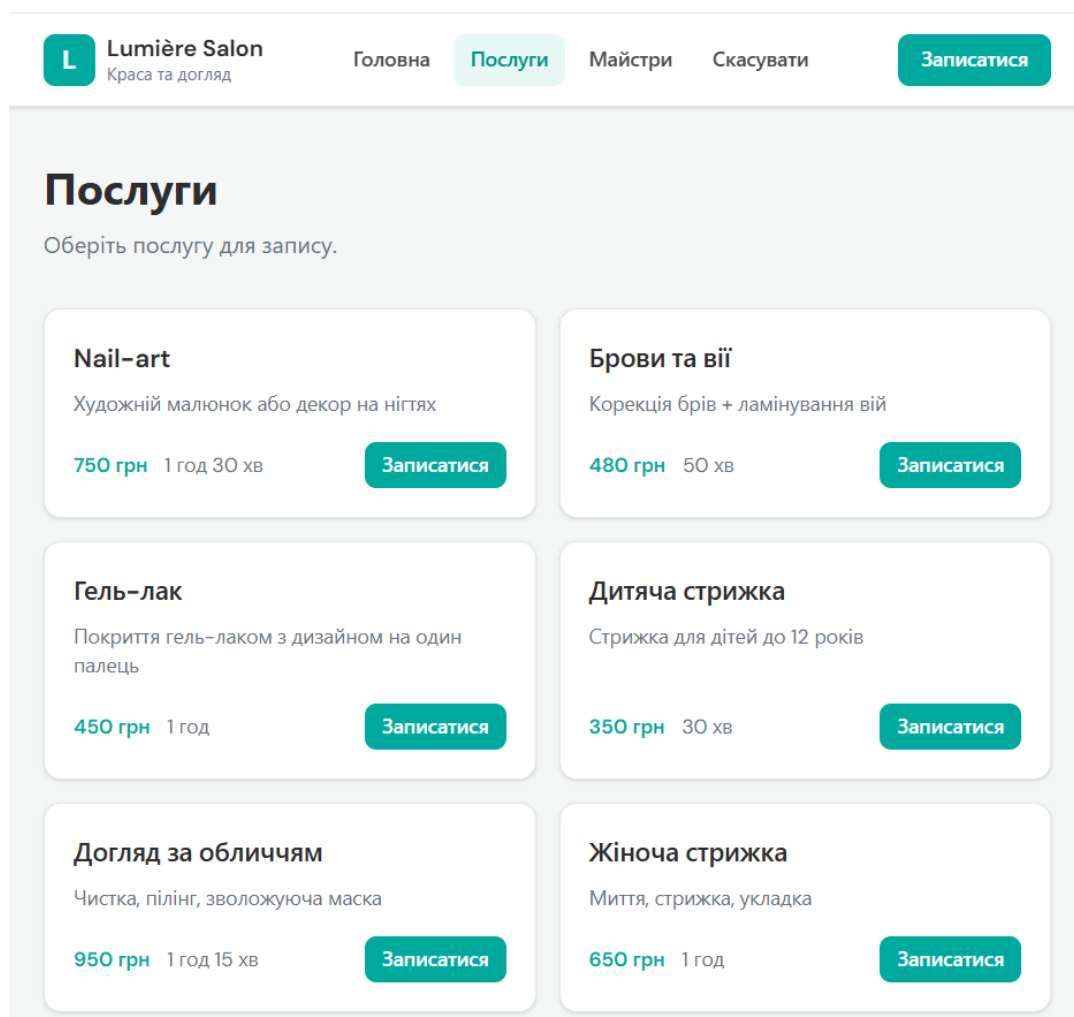


Рисунок 3.9 – Перевірка адаптивності застосунку

У процесі налагодження системи було виявлено та усунено низку недоліків, пов'язаних із перевіркою доступності часових слотів, обробкою помилкових даних та оновленням інтерфейсу після виконання операцій бронювання. Зокрема, було вдосконалено алгоритм перевірки перетину часових проміжків та реалізовано додаткову серверну валідацію введених даних. Результати тестування системи наведені в таблиці 3.1. Під час тестування також перевірялась коректність роботи бази даних SQLite та забезпечення цілісності інформації. Було підтверджено правильність роботи зовнішніх ключів, SQL-обмежень та механізмів перевірки статусів записів.

Таблиця 3.1 – Результати тестування системи

№	Функціональний модуль	Тестова дія	Результат
1	Авторизація адміністратора	Введення правильного логіна і пароля	Успішний вхід
2	Авторизація адміністратора	Введення неправильного пароля	Відображення помилки
3	Онлайн-бронювання	Створення нового запису	Запис створено
4	Онлайн-бронювання	Спроба повторного бронювання того самого часу	Відображення повідомлення про зайнятий слот
5	Скасування запису	Введення правильного номера запису	Запис скасовано
6	Скасування запису	Введення неправильного номера запису	Відображення помилки
7	Керування послугами	Створення нової послуги	Послугу додано
8	Керування майстрами	Редагування інформації про майстра	Дані оновлено

Отже, усі виявлені під час налагодження недоліки були усунені, що дозволило забезпечити надійну та безпечну роботу вебплатформи для бронювання послуг салону краси.

3.3 Розробка бази даних

Для реалізації бази даних було використано реляційну систему SQLite, яка забезпечує просту інтеграцію із серверною частиною застосунку та не потребує встановлення окремого сервера баз даних. Розробка структури бази даних здійснювалась із використанням SQL-запитів та механізму створення таблиць через файл `schema.sql`. Під час проєктування бази даних було враховано необхідність забезпечення цілісності інформації, підтримки зв'язків між таблицями та можливості подальшого розширення функціональності системи.

Основними таблицями бази даних є `services`, яка використовується для зберігання інформації про послуги салону краси, `masters`, що містить інформацію про майстрів, `master_services`, яка реалізує зв'язок між майстрами та послугами, `working_hours` для зберігання графіка роботи майстрів, а також таблиця `bookings`, у якій зберігаються записи клієнтів на послуги салону краси. Для створення

таблиці послуг використовується SQL-запит CREATE TABLE, який формує структуру таблиці та визначає типи даних полів (ліст. 3.7).

Лістинг 3.7 – Створення таблиці послуг

```
CREATE TABLE IF NOT EXISTS services (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  description TEXT,
  price INTEGER NOT NULL,
  duration_min INTEGER NOT NULL,
  is_active INTEGER NOT NULL DEFAULT 1,
  created_at TEXT NOT NULL DEFAULT (datetime('now'))
);
```

кінець лістингу 3.7

Таблиця services містить інформацію про назву послуги, її опис, вартість, тривалість та статус активності. Первинний ключ id використовується для унікальної ідентифікації записів. Для збереження інформації про майстрів створена таблиця masters (ліст. 3.8).

Лістинг 3.8 – Створення таблиці майстрів

```
CREATE TABLE IF NOT EXISTS masters (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  specialty TEXT,
  bio TEXT,
  is_active INTEGER NOT NULL DEFAULT 1,
  created_at TEXT NOT NULL DEFAULT (datetime('now'))
);
```

кінець лістингу 3.8

Для реалізації зв'язку між майстрами та послугами використовується проміжна таблиця master_services. Це дозволяє одному майстру виконувати декілька послуг, а одній послугі бути доступною для кількох майстрів (ліст. 3.9).

Лістинг 3.9 – Створення таблиці зв'язків між майстрами та послугами

```
CREATE TABLE IF NOT EXISTS master_services (
    master_id INTEGER NOT NULL REFERENCES masters(id) ON DELETE CASCADE,
    service_id INTEGER NOT NULL REFERENCES services(id) ON DELETE CASCADE,
    PRIMARY KEY (master_id, service_id)
);
```

кінець лістингу 3.9

Однією з основних таблиць системи є таблиця bookings, у якій зберігається інформація про записи клієнтів (ліст. 3.10). У таблиці bookings зберігається інформація про клієнта, дату бронювання, час початку послуги та статус запису. Для забезпечення цілісності даних використовуються зовнішні ключі (FOREIGN KEY), які пов'язують записи з таблицями послуг та майстрів.

Лістинг 3.10 – Створення таблиці записів клієнтів

```
CREATE TABLE IF NOT EXISTS bookings (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    service_id INTEGER NOT NULL REFERENCES services(id),
    master_id INTEGER NOT NULL REFERENCES masters(id),
    client_name TEXT NOT NULL,
    client_phone TEXT NOT NULL,
    client_email TEXT,
    booking_date TEXT NOT NULL,
    start_time TEXT NOT NULL,
    status TEXT NOT NULL DEFAULT 'confirmed',
    created_at TEXT NOT NULL DEFAULT (datetime('now'))
);
```

кінець лістингу 3.10

Для взаємодії із базою даних серверна частина системи використовує SQL-запити типу SELECT, INSERT, UPDATE та DELETE. Для отримання списку активних послуг використовується запит SELECT (ліст. 3.11).

Лістинг 3.11 – SQL-запит отримання списку послуг

```
SELECT id, name, description, price, duration_min
FROM services WHERE is_active = 1
ORDER BY name;
```

кінець лістингу 3.11

Для створення нового запису клієнта використовується SQL-запит INSERT (ліст. 3.12).

Лістинг 3.12 – SQL-запит створення нового бронювання

```
INSERT INTO bookings (
  service_id,
  master_id,
  client_name,
  client_phone,
  client_email,
  booking_date,
  start_time
)
VALUES (?, ?, ?, ?, ?, ?, ?);
```

кінець лістингу 3.12

Для оновлення статусу запису використовується SQL-запит UPDATE. Для видалення інформації з таблиць використовується SQL-запит DELETE. У серверній частині системи для роботи з базою даних використовується окремий модуль db.js, який забезпечує підключення до SQLite та виконання SQL-запитів (ліст. 3.13).

Лістинг 3.13 – Підключення до бази даних SQLite

```
const db = new DatabaseSync(dbPath);
db.exec('PRAGMA journal_mode = WAL');
db.exec('PRAGMA foreign_keys = ON');
```

кінець лістингу 3.13

Для забезпечення цілісності даних у системі використовуються транзакції. У випадку виникнення помилки всі зміни автоматично скасовуються (ліст. 3.14). Реалізація транзакцій дозволяє забезпечити цілісність і узгодженість даних у базі даних під час виконання критичних операцій. У випадку виникнення помилки всі зміни автоматично скасовуються за допомогою операції ROLLBACK, що запобігає появі пошкоджених або неповних даних у системі.

Лістинг 3.14 – Реалізація транзакції

```
export function withTransaction(fn) {  
  db.exec('BEGIN IMMEDIATE');  
  try {  
    const result = fn();  
    db.exec('COMMIT');  
    return result;  
  } catch (err) {  
    db.exec('ROLLBACK');  
    throw err;  }  
}
```

кінець лістингу 3.14

У результаті розробки бази даних було створено реляційну структуру, яка забезпечує ефективне зберігання інформації, підтримку зв'язків між сутностями та стабільну роботу системи онлайн-бронювання. Використання SQLite та SQL-запитів дозволило реалізувати надійний механізм роботи з даними.

3.4 Захист інформаційно-комп'ютерної системи

Під час розробки вебплатформи для бронювання послуг салону краси особлива увага приділялась забезпеченню захисту інформаційно-комп'ютерної системи. Основною метою реалізації механізмів захисту було забезпечення безпечного доступу до адміністративної частини системи, захист персональних даних клієнтів, контроль прав доступу та запобігання несанкціонованому використанню функціональних можливостей вебзастосунку.

У системі реалізовано механізм авторизації адміністратора на основі токенів доступу. Авторизація використовується для перевірки прав доступу користувача до адмін панелі та функцій керування. Доступ до цієї частини системи мають лише користувачі, які пройшли процедуру входу. Для входу до адміністративної панелі адміністратор вводить логін та пароль. Серверна частина перевіряє правильність введених даних та у випадку успішної перевірки генерує унікальний токен доступу. Згенерований токен використовується для подальшої авторизації запитів до серверної частини системи (ліст. 3.15).

Лістинг 3.15 – Реалізація входу адміністратора

```
export function loginAdmin(username, password) {
  if (username !== ADMIN_USER || password !== ADMIN_PASS) {
    return null;
  }
  const token = crypto.randomBytes(32).toString('hex');

  sessions.set(token, Date.now() + TOKEN_TTL_MS);
  return token;
}
```

кінець лістингу 3.15

У наведеному алгоритмі використовується генерація випадкового токена за допомогою модуля `crypto`. Згенерований токен зберігається у пам'яті сервера разом із часом завершення сесії користувача. Це дозволяє забезпечити контроль активності адміністративних сесій. Для перевірки доступу до адміністративних маршрутів використовується `middleware`-функція `requireAdmin`. Даний механізм перевіряє наявність токена у заголовку HTTP-запиту та контролює термін дії сесії (ліст. 3.16).

Лістинг 3.16 – Перевірка токена авторизації

```
export function requireAdmin(req, res, next) {
  const header = req.headers.authorization;
  if (!header?.startsWith('Bearer ')) {
    return res.status(401).json({
      error: 'Потрібна авторизація'
    });
  }
  const token = header.slice(7);

  const expires = sessions.get(token);

  if (!expires || Date.now() > expires) {
    sessions.delete(token);

    return res.status(401).json({
      error: 'Сесія закінчилась. Увійдіть знову.'
    });
  }
  next();
}
```

кінець лістингу 3.16

У результаті перевірки сервер визначає, чи має користувач право доступу до адміністративних функцій. У випадку відсутності або завершення терміну дії токена сервер повертає повідомлення про необхідність повторної авторизації. Для захисту клієнтської частини системи використовується механізм збереження токена у localStorage. Це дозволяє підтримувати авторизований стан адміністратора після оновлення сторінки браузера (ліст. 3.17).

Лістинг 3.17 – Збереження токена авторизації

```
export function setAdminToken(token) {
  localStorage.setItem(TOKEN_KEY, token);
}
```

кінець лістингу 3.17

У системі також реалізовано перевірку введених користувачем даних. Серверна частина здійснює валідацію номера телефону, імені клієнта, формату дати та часу бронювання. Це дозволяє запобігти збереженню некоректної інформації у базі даних (ліст. 3.18).

Лістинг 3.18 – Перевірка формату номера телефону

```
if (!/^(\\+380|0)\\d{9}$/.test(phone)) {
  return res.status(400).json({
    error: 'Телефон: +380XXXXXXXXX або 0XXXXXXXXX'
  });
}
```

кінець лістингу 3.18

Для забезпечення цілісності інформації у базі даних використовуються зовнішні ключі (FOREIGN KEY) та SQL-обмеження (CHECK). Вони дозволяють контролювати правильність зв'язків між таблицями та допустимі значення полів (ліст. 3.19).

Лістинг 3.19 – Використання SQL-обмеження CHECK

```
status TEXT NOT NULL DEFAULT 'confirmed'
```

```
CHECK (
  status IN (
    'pending',
    'confirmed',
    'cancelled',
    'completed'
  )
)
```

кінець лістингу 3.19

У системі також реалізовано контроль доступності часових слотів для бронювання. Перед створенням нового запису сервер перевіряє, чи не зайнятий обраний проміжок часу іншим клієнтом. Це дозволяє уникнути дублювання записів та конфліктів бронювання (ліст. 3.20).

Лістинг 3.20 – Перевірка доступності часового слоту

```
const taken = occupied.some((o) =>
  rangesOverlap(start, end, o.start, o.end)
);

if (!taken) {
  slots.push(minutesToTime(start));
}
```

кінець лістингу 3.20

У результаті реалізації механізмів захисту було забезпечено базовий рівень безпеки вебзастосунку. Реалізовані засоби авторизації, перевірки токенів, валідації введених даних та контролю цілісності бази даних дозволяють захистити систему від несанкціонованого доступу та помилкових дій користувачів. Використані методи захисту відповідають функціональним особливостям розробленої інформаційно-комп'ютерної системи та забезпечують її стабільну й безпечну роботу.

3.5 Розробка документації для програмного продукту

Для забезпечення коректної роботи вебплатформи бронювання послуг салону краси було підготовлено документацію програмного продукту, яка містить опис системних вимог, інструкції з встановлення та запуску системи, а також довідкові матеріали для користувачів і адміністраторів.

Розроблена інформаційна система є веборієнтованим клієнт-серверним застосунком, який складається з фронтенд та бекенд частини. Клієнтська частина системи реалізована на React із використанням Tailwind CSS, а серверна частина – на Node.js та Express із використанням бази даних SQLite. Для стабільної роботи системи необхідно забезпечити відповідність мінімальним системним вимогам (табл. 3.2).

Таблиця 3.2 – Мінімальні системні вимоги

Параметр	Мінімальні вимоги
Операційна система	Windows 10 / Linux
Процесор	Intel Core i3
Оперативна пам'ять	4 ГБ
Вільне місце на диску	500 МБ
Node.js	Версія 18 або вище
npm	Версія 9 або вище
Веббраузер	Google Chrome, Edge, Firefox
Підключення до Internet	Необхідне

Для запуску програмного продукту необхідно попередньо встановити середовище Node.js, яке містить менеджер пакетів npm [12]. Після цього здійснюється встановлення залежностей проєкту для клієнтської та серверної частини системи. Структура проєкту поділена на два основні каталоги: client – клієнтська частина React-застосунку, server – серверна частина Node.js та Express. Перед запуском системи необхідно встановити всі залежності за допомогою команди (ліст. 3.21).

Лістинг 3.21 – Встановлення залежностей

```
npm install
```

кінець лістингу 3.21

Для запуску серверної частини необхідно перейти до каталогу `server` та виконати команду запуску сервера (ліст. 3.22).

Лістинг 3.22 – Запуск серверної частини

```
cd server  
npm run dev
```

кінець лістингу 3.22

Після запуску серверна частина системи запускається на локальному сервері та забезпечує роботу REST API для взаємодії з клієнтською частиною застосунку. Для запуску клієнтської частини необхідно перейти до каталогу `client` та виконати команду запуску React-застосунку (ліст. 3.23).

Лістинг 3.23 – Запуск клієнтської частини

```
cd client  
npm run dev
```

кінець лістингу 3.23

Після запуску клієнтської частини вебзастосунок стає доступним у браузері за локальною адресою, яка автоматично формується середовищем Vite [19]. У серверній частині системи реалізовано автоматичне створення бази даних SQLite під час першого запуску застосунку. Для цього використовується файл `schema.sql`, який містить SQL-запити створення таблиць бази даних (ліст. 3.24).

Лістинг 3.24 – Підключення та ініціалізація бази даних

```
const db = new DatabaseSync(dbPath);  
db.exec('PRAGMA journal_mode = WAL');  
db.exec('PRAGMA foreign_keys = ON');  
  
const schema = fs.readFileSync(  
  path.join(__dirname, 'schema.sql'),  
  'utf8'  
);  
  
db.exec(schema);
```

кінець лістингу 3.24

Для початкового наповнення бази даних тестовими даними використовується окремий файл `seed.js`. Він створює список послуг, майстрів та графік роботи салону (ліст. 3.25).

Лістинг 3.25 – Запуск початкового наповнення бази даних

```
node seed.js
```

кінець лістингу 3.25

У процесі розробки також було створено довідкові матеріали для користувачів системи. Користувач вебплатформи має можливість переглядати перелік доступних послуг салону краси, обирати майстра, переглядати доступний час для запису, створювати онлайн-бронювання та скасовувати існуючі записи. Інтерфейс користувача реалізований таким чином, щоб забезпечити просту та зрозумілу взаємодію із системою без необхідності додаткового навчання.

Адміністратор системи має доступ до адміністративної панелі, у якій реалізовано функції авторизації, керування послугами та майстрами, перегляду списку бронювань, зміни статусів записів клієнтів і видалення записів. Адміністративний інтерфейс дозволяє централізовано керувати інформацією вебзастосунку та контролювати процес онлайн-бронювання послуг салону краси.

Для забезпечення коректної роботи адміністративної панелі використовується механізм авторизації через токени доступу. Після входу адміністратора токен автоматично зберігається у браузері та використовується для підтвердження прав доступу до адміністративних маршрутів (ліст. 3.26).

Лістинг 3.26 – Збереження токена адміністратора

```
export function setAdminToken(token) {
  localStorage.setItem(TOKEN_KEY, token);
}
```

кінець лістингу 3.26

Під час розгортання системи на сервері необхідно забезпечити встановлення Node.js, запуск серверної частини застосунку та доступність API для клієнтської частини вебзастосунку. База даних SQLite зберігається локально у каталозі data, що спрощує резервне копіювання та супровід системи.

Розроблена документація дозволяє забезпечити правильне встановлення, налаштування та використання програмного продукту. Наявність інструкцій запуску, опису системних вимог та довідкових матеріалів спрощує подальшу експлуатацію вебплатформи для бронювання послуг салону краси.

Висновки до розділу 3

У процесі практичної реалізації вебплатформи було створено клієнтську та серверну частини системи, реалізовано маршрутизацію сторінок, взаємодію з REST API та підключення до бази даних. Для створення вебзастосунку використано React, Node.js, Express.js та SQLite, що дозволило забезпечити стабільну роботу системи та зручну взаємодію користувачів із вебплатформою.

Під час тестування та налагодження системи було перевірено роботу алгоритму онлайн-бронювання, генерацію часових слотів, авторизацію адміністратора, перевірку введених даних та адаптивність інтерфейсу. У результаті тестування виявлені помилки були усунені, що дозволило забезпечити коректну роботу вебзастосунку.

У ході розробки бази даних було створено реляційну структуру для зберігання інформації про послуги, майстрів, графік роботи та записи клієнтів. Для взаємодії з базою даних використано SQL-запити SELECT, INSERT, UPDATE та DELETE, а також реалізовано механізми підтримки цілісності даних за допомогою зовнішніх ключів та SQL-обмежень.

Під час реалізації захисту інформаційно-комп'ютерної системи було впроваджено механізм авторизації адміністратора з використанням токенів доступу, перевірку прав доступу до адміністративних маршрутів та валідацію

введених даних. Це дозволило обмежити доступ до адміністративної частини системи та забезпечити базовий рівень захисту даних.

У процесі створення документації програмного продукту було визначено системні вимоги до роботи вебплатформи, описано процедуру встановлення та запуску системи, а також підготовлено довідкові матеріали для користувачів і адміністратора. Це спрощує процес використання та подальшого супроводу програмного забезпечення.

ВИСНОВКИ

Під час формування постановки завдання було визначено основні цілі та функціональні можливості майбутньої системи. У роботі поставлено завдання реалізувати вебплатформу із підтримкою онлайн-бронювання, керування послугами, майстрами та записами клієнтів, а також забезпечити наявність адміністративної панелі й базових механізмів захисту інформації.

У результаті аналізу сучасного стану вебплатформ для бронювання послуг салонів краси було досліджено популярні сервіси EasyWeek, Dashboard Beauty, Altegio та Beauty Pro. Проведений аналіз дозволив визначити основні функціональні можливості сучасних систем онлайн-запису, їх переваги та недоліки, а також обґрунтувати доцільність створення власного вебзастосунку для автоматизації процесу бронювання послуг салону краси.

У процесі аналізу вимог до програмного забезпечення було визначено функціональні та нефункціональні вимоги до системи, обґрунтовано вибір клієнт-серверної архітектури та виконано UML-моделювання основних компонентів вебзастосунку. Було встановлено вимоги до швидкодії, безпеки, адаптивності інтерфейсу та забезпечення цілісності даних.

Під час вибору засобів, методів і алгоритмів розробки було обрано React для реалізації клієнтської частини, Node.js та Express.js для серверної логіки, SQLite для організації бази даних та Tailwind CSS для створення адаптивного інтерфейсу. Також було реалізовано алгоритм генерації доступних часових слотів із перевіркою перетину часових інтервалів для уникнення конфліктів бронювання.

У процесі практичної реалізації об'єкта проектування було створено вебплатформу Lumière Salon із підтримкою онлайн-запису на послуги салону краси. Реалізовано сторінки перегляду послуг, вибору майстра, бронювання часу, скасування запису та адміністративну панель для керування інформацією системи.

Під час тестування та налагодження інформаційно-комп'ютерної системи було перевірено роботу алгоритму бронювання, генерації часових слотів, механізму авторизації адміністратора та взаємодії клієнтської і серверної частини вебзастосунку. Усі виявлені під час тестування помилки були усунені, що дозволило забезпечити стабільну роботу системи.

У ході розробки бази даних було створено таблиці `services`, `masters`, `master_services`, `working_hours` та `bookings`, які забезпечують зберігання інформації про послуги, майстрів, графік роботи та записи клієнтів. Для роботи з базою даних використано SQL-запити `SELECT`, `INSERT`, `UPDATE` та `DELETE`, а також реалізовано механізми забезпечення цілісності даних.

Під час реалізації механізмів захисту інформаційно-комп'ютерної системи було впроваджено авторизацію адміністратора на основі токенів доступу, серверну валідацію введених даних, захист адміністративних маршрутів та контроль доступу до функцій адміністративної панелі. Це дозволило забезпечити базовий рівень безпеки вебзастосунку.

У процесі розробки документації програмного продукту було сформовано опис системних вимог, структури проєкту, порядку встановлення `Node.js`, `npm`-залежностей, запуску серверної та клієнтської частини системи, а також налаштування бази даних `SQLite`. Підготовлена документація забезпечує можливість швидкого розгортання та подальшого використання вебзастосунку.

Практичне значення отриманих результатів полягає у можливості використання розробленого вебзастосунку для автоматизації діяльності салонів краси та оптимізації процесу онлайн-бронювання послуг. Розроблена система забезпечує зручну взаємодію користувачів із салоном краси, підвищує ефективність роботи персоналу та може бути використана як основа для подальшого розвитку аналогічних інформаційних систем. У подальшому вебзастосунок може бути вдосконалений шляхом інтеграції онлайн-оплати, email-сповіщень, SMS-повідомлень, календарної синхронізації, мобільного застосунку та розширених механізмів захисту інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Altegio – Business automation platform for service companies. Altegio. URL: <https://alteg.io/uk/> (date of access: 10.02.2026).
2. Avramets A., Ziuziun V. Justification for Creating a CRM System to Automate the Operations of a Beauty Salon. 2st International Scientific and Practical Conference “Information Systems and Technology: Results and Prospects”. 2025. P. 167-171. URL: http://kist.ntu.edu.ua/konferencii/54_konf_2025.pdf#page=167 (date of access: 9.02.2026).
3. Barat A. Implementing Artificial Intelligence in the Beauty Industry: Innovative Approaches to Personalization and Optimization of Manicure Services. American Journal of Pedagogical and Educational Research. 2025. Vol. 41. P. 26-30. URL: <https://media.neliti.com/media/publications/678110-implementing-artificial-intelligence-in-af9edecb.pdf> (date of access: 9.02.2026).
4. Beauty Pro – Software for beauty salon automation. Beauty Pro. URL: <https://beautyprosoftware.com/> (date of access: 10.02.2026).
5. Beautyhare: A Design of a Web-Based Management System for Salon Services / E. Blancaflor et al. Proceedings of the 2024 7th International Conference on Computers in Management and Business. 2024. P. 41-47. URL: <https://dl.acm.org/doi/abs/10.1145/3647782.3647789> (date of access: 9.02.2026).
6. Dashboard Beauty - Premium Beauty Supplies. Dashboard Beauty. URL: <https://dashboardbeauty.com/> (date of access: 10.02.2026).
7. EasyWeek – Online booking and business automation system. EasyWeek. URL: <https://easyweek.com.ua/> (date of access: 10.02.2026).
8. Express.js – Fast, unopinionated, minimalist web framework for Node.js. Express.js. URL: <https://expressjs.com/> (date of access: 11.03.2026).
9. JavaScript – JavaScript language documentation. JavaScript. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (date of access: 24.03.2026).
10. Mathematical Justification for the Development of a Web Platform for Beauty Salon Operations / V. Ziuziun et al. 2025 IEEE 5th International Conference

on Smart Information Systems and Technologies (SIST). 2025. P. 1-6. URL: <https://ieeexplore.ieee.org/abstract/document/11139271> (date of access: 9.02.2026).

11. Node.js – JavaScript runtime built on Chrome’s V8 JavaScript engine. Node.js. URL: <https://nodejs.org/> (date of access: 10.03.2026).

12. Npm – Package manager for Node.js. Npm. URL: <https://www.npmjs.com/> (date of access: 07.04.2026).

13. Optimizing Salon Services: Reducing Scheduling Errors and Wait Times through Automation / R. Choudhary et al. Spectrum of Engineering Sciences. 2026. P. 1239-1257. URL: <https://www.thesesjournal.com/index.php/1/article/view/1349/1022> (date of access: 9.02.2026).

14. React – JavaScript library for building user interfaces. React. URL: <https://react.dev/> (date of access: 05.03.2026).

15. React Router – Declarative routing for React applications. React Router. URL: <https://reactrouter.com/> (date of access: 07.03.2026).

16. REST API Tutorial – REST architectural style and API development resource. REST API. URL: <https://restfulapi.net/> (date of access: 12.03.2026).

17. SQLite – Self-contained, high-reliability, embedded SQL database engine. SQLite. URL: <https://www.sqlite.org/> (date of access: 13.03.2026).

18. Tailwind CSS – Utility-first CSS framework for rapidly building modern user interfaces. Tailwind. URL: <https://tailwindcss.com/> (date of access: 09.03.2026).

19. Vite – Next generation frontend tooling. Vite. URL: <https://vitejs.dev/> (date of access 16.04.2026).