

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Вебсистема бронювання послуг салону краси»

Виконав студент групи ІПЗ-22-2
БОРОДАЧОВ Микита Андрійович
Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна
Рецензент:
ЛОКТИКОВА Тамара Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«27» лютого 2026 р.

**ЗАВДАННЯ
на кваліфікаційну роботу**

Здобувач вищої освіти: **БОРОДАЧОВ Микита Андрійович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Вебсистема бронювання послуг салону краси»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**
Вихідні дані для роботи: об'єктом дослідження є процес автоматизації запису клієнтів, управління розкладом майстрів та обробки бронювань у сфері послуг краси. Предметом дослідження є методи та програмні засоби розробки вебсистем, моделювання даних, реалізації бізнес-логіки бронювання та автоматизації сповіщень користувачів на основі сучасних вебтехнологій.
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	01.03.2026р.	01.03.2026р.
2	Чижмотря О.Г.	01.03.2026р.	01.03.2026р.
3	Чижмотря О.Г.	01.03.2026р.	01.03.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебсистеми для бронювання послуг салону краси та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить текстову частину, викладену на 80 сторінках друкованого тексту. Список використаних джерел містить 32 найменування і займає 3 сторінки. У роботі наведено 32 ілюстрації. Також у роботі є 18 сторінок додатків. Загальний обсяг роботи – 98 сторінок.

Головною метою роботи є створення сучасної вебсистеми для онлайн-бронювання послуг салону краси, яка забезпечує зручне керування бронюваннями користувачами, швидкий пошук вільних часових слотів, інтерактивну взаємодію між клієнтами та майстрами, а також надає можливості для автоматизації процесу запису та оптимізації роботи персоналу салону.

У першому розділі проведено аналіз напрямків використання інформаційних технологій, сформульовано постановку задачі та виконано огляд існуючих програмних рішень для сервісів бронювання, зокрема українських платформ Appointer та Anolla, а також європейських Planity та Shore. На основі порівняльного аналізу обрано архітектуру системи та обґрунтовано вибір технологічного стеку: .NET Core для серверної частини, React.js для клієнтської частини, SQLite як базу даних та Entity Framework Core для роботи з нею.

У другому розділі проведено детальний аналіз функціональних та нефункціональних вимог до системи, визначено ролі користувачів (клієнт та майстер) та їхні права доступу. Розроблено діаграми варіантів використання, активності, класів та компонентів.

					Житомирська політехніка.26.121.05.000 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Бородачов М. А.			Вебплатформа бронювання послуг салону краси. Пояснювальна записка			Літ.	Арк.	Аркуші	
Керівник		Чижмотря О. Г.								3	98
								ФІКТ, гр. ІПЗ-22-2			
Рецензент		Локтікова Т. М.									
Зав. каф.		Вакалюк Т. А.									

Спроектовано структуру реляційної бази даних SQLite, яка включає всім основних таблиць: Users, Masters, Services, Bookings, MasterServices, Schedules, Payments та Reviews. Реалізовано функціонал авторизації з використанням JWT-токенів, бонусну систему, систему сповіщень та налаштування інтерфейсу (тема, мова, валюта).

У третьому розділі детально розглянуто порядок встановлення та налаштування вебсистеми, описано структуру інтерфейсу, включаючи головну сторінку, сторінку послуг, сторінку майстрів, кабінет клієнта, кабінет майстра, сторінку створення бронювання, сторінку оплати та сторінку профілю. Проведено комплексне тестування основних компонентів системи, яке підтвердило коректність валідації введених даних, ефективність контролю доступу користувачів до різних ресурсів і стабільність роботи API за різних сценаріїв використання.

Висновки містять ключові результати аналізу вимог до програмного продукту та описують етапи розробки системи, включаючи створення технічного завдання, розробку моделі бази даних, реалізацію серверної та клієнтської частин, а також тестування програмного забезпечення. У додатках розміщено лістинги програмного коду основних модулів системи.

КЛЮЧОВІ СЛОВА: REACT.JS, .NET CORE, ENTITY FRAMEWORK CORE, SQLITE, JWT, REST API, БРОНЮВАННЯ, САЛОН КРАСИ, МАЙСТЕР.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ABSTRACT

The bachelor's qualification thesis consists of a web system for booking beauty salon services and an explanatory note. The explanatory note contains a textual part presented on 80 pages of printed text. The list of references includes 32 sources and occupies 3 pages. The work contains 32 illustrations. There are also 18 pages of appendices. The total volume of the work is 97 pages.

The main goal of the work is to create a modern web system for online booking of beauty salon services, which provides convenient management of bookings by users, quick search for free time slots, interactive interaction between clients and masters, as well as provides opportunities for automating the booking process and optimizing the work of salon staff.

In the first chapter, the analysis of directions for using information technologies was carried out, the problem statement was formulated, and a review of existing software solutions for booking services was performed, in particular the Ukrainian platforms Appointer and Anolla, as well as the European Planity and Shore. Based on a comparative analysis, the system architecture was chosen and the choice of technological stack was justified: .NET Core for the server part, React.js for the client part, SQLite as a database, and Entity Framework Core for working with it.

In the second chapter, a detailed analysis of functional and non-functional requirements for the system was carried out, user roles (client and master) and their access rights were defined. Use case diagrams, activity diagrams, class diagrams and component diagrams were developed.

The structure of a relational SQLite database was designed, which includes eight main tables: Users, Masters, Services, Bookings, MasterServices, Schedules, Payments and Reviews. The functionality of authorization using JWT tokens, a bonus system, a notification system and interface settings (theme, language, currency) was implemented.

In the third chapter, the procedure for installing and configuring the web system was considered in detail, the interface structure was described, including the

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

home page, services page, masters page, client cabinet, master cabinet, booking creation page, payment page and profile page. Comprehensive testing of the main components of the system was carried out, which confirmed the correctness of input data validation, the effectiveness of user access control to various resources and the stability of the API operation under various usage scenarios.

The conclusions contain the key results of the analysis of requirements for the software product and describe the stages of system development, including the creation of technical specifications, development of a database model, implementation of the server and client parts, as well as software testing. The appendices contain listings of the program code of the main system modules.

KEYWORDS: REACT.JS, .NET CORE, ENTITY FRAMEWORK CORE, SQLITE, JWT, REST API, BOOKING, BEAUTY SALON, MASTER.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	8
ВСТУП	9
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	12
1.1. Постановка задачі	12
1.2. Аналіз аналогів програмного продукту.....	15
1.3. Вибір архітектури веборієнтованої системи.....	21
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	24
Висновки до першого розділу	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ.....	30
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	30
2.2. Розробка бази даних системи	35
2.3. Проєктування та реалізація алгоритмів роботи системи.....	38
2.4. Реалізація функціоналу системи	43
Висновки до другого розділу.....	48
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СЕРВІСОМ	50
3.1. Порядок встановлення та налаштування параметрів системи.....	50
3.2. Структура інтерфейсу вебзастосунку	53
3.3. Тестування роботи програмного забезпечення	67
Висновки до третього розділу	72
ВИСНОВКИ.....	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	78
ДОДАТКИ.....	81

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ПЗ – програмне забезпечення

СКБД – система керування базами даних

SQL – Structured Query Language (структурована мова запитів)

API – Application Programming Interface (прикладний програмний інтерфейс)

REST – Representational State Transfer (передача репрезентативного стану)

URL – Uniform Resource Locator (уніфікований локатор ресурсу)

ORM – Object-Relational Mapper (об'єктно-реляційний маппер)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

DOM – Document Object Model (об'єктна модель документа)

DTO – Data Transfer Object (об'єкт передачі даних)

JWT – JSON Web Token (вебтокен у форматі JSON)

UI/UX – User Interface / User Experience (інтерфейс користувача / досвід користувача)

PWA – Progressive Web Application (прогресивний вебзастосунок)

CSV – Comma-Separated Values (текстовий формат для зберігання табличних даних)

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У сучасних умовах розвитку сфери послуг, зокрема індустрії краси, автоматизація процесів запису клієнтів стає невід'ємною складовою успішного бізнесу. Традиційні методи ведення запису, такі як телефонні дзвінки, паперові журнали або обмін повідомленнями в месенджерах, часто призводять до помилок, втрати часу, подвійних бронювань та незадоволення клієнтів. Використання вебсистеми для онлайн-бронювання дозволяє оптимізувати роботу адміністраторів, зменшити навантаження на персонал, надати клієнтам зручний інструмент самостійного планування візитів, автоматично надсилати нагадування, що зменшує кількість неявок до сорока відсотків, та підвищити загальну ефективність салону краси.

Сучасні системи бронювання базуються на клієнт-серверній архітектурі, використовують вебтехнології для забезпечення доступності з будь-якого пристрою та інтегруються з календарями, платіжними системами, CRM та сервісами сповіщень. Однак існуючі рішення часто мають надлишковий функціонал, високу вартість підписки або складність налаштування, що робить актуальним створення власної спеціалізованої вебсистеми. Крім того, більшість наявних платформ не мають повноцінної української локалізації, що ускладнює їх використання для україномовних клієнтів та майстрів.

Мета і завдання дослідження. Метою роботи є аналіз предметної області, проєктування та програмна реалізація вебсистеми для онлайн-бронювання послуг салону краси, що забезпечує автоматизацію процесу запису клієнтів, оптимізацію роботи персоналу та оперативне сповіщення про зміни у розкладі.

Для досягнення поставленої мети необхідно провести аналіз теоретичних і практичних вимог до вебсистем бронювання послуг салону краси, визначити структуру та архітектуру системи, що відповідає сучасним стандартам, оцінити існуючі аналогічні вебрішення і визначити їх сильні та слабкі сторони, вибрати та обґрунтувати оптимальні інструменти і технології для розробки платформи, розробити ключові компоненти системи, алгоритми

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

функціонування і користувацький інтерфейс, а також впровадити, протестувати та оптимізувати вебзастосунок для забезпечення його надійності та продуктивності.

Об'єкт і предмет дослідження. Об'єктом дослідження є процес автоматизації запису клієнтів, управління розкладом майстрів та обробки бронювань у сфері послуг краси. Досліджується взаємодія між клієнтами та майстрами, а також способи оптимізації цієї взаємодії за допомогою інформаційних технологій. Предметом дослідження є методи та програмні засоби розробки вебсистем, моделювання даних, реалізації бізнес-логіки бронювання та автоматизації сповіщень користувачів. Розглядаються сучасні підходи до побудови вебзастосунків з використанням архітектури REST API, мікросервісів та клієнт-серверної моделі.

Методи дослідження. У процесі розробки вебзастосунку застосовуються методи об'єктно-орієнтованого проєктування, що дозволяє створити гнучку та масштабовану архітектуру системи. Використовуються методи сучасного вебпрограмування, зокрема розробка односторінкових додатків з використанням бібліотеки React.js та створення серверної частини на платформі .NET Core з використанням ASP.NET Core Web API. Для проєктування інтерфейсів користувача застосовуються принципи UI/UX дизайну, що забезпечують інтуїтивно зрозумілу та зручну взаємодію з платформою.

Технології взаємодії з базами даних реалізовано через ORM Entity Framework Core, що дозволяє абстрагуватися від конкретної СКБД та працювати з даними у вигляді об'єктів. Інтеграція зі сторонніми API, зокрема платіжною системою LiqPay, забезпечує можливість онлайн-оплати послуг. Використання актуального технологічного стеку, а саме .NET Core, React, SQLite, JWT та BCrypt, дозволяє досягти високого рівня продуктивності, масштабованості та безпеки системи, що є необхідною умовою для її практичного використання в умовах сучасного інформаційного середовища.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Практичне значення отриманих результатів. Розроблена вебсистема може бути впроваджена в салонах краси будь-якого розміру – від невеликих студій до великих мереж. Вона дозволяє автоматизувати процес запису клієнтів, зменшити навантаження на персонал, підвищити задоволеність клієнтів за рахунок зручного онлайн-запису та системи сповіщень. Крім того, система може бути легко адаптована для інших сфер послуг, таких як медичні клініки, перукарні, косметологічні кабінети, SPA-центри, фітнес-клуби та інші заклади, що працюють за попереднім записом.

Завдяки модульній архітектурі, функціонал системи може бути розширено для додавання нових можливостей, зокрема адміністративної панелі, інтеграції з Google Calendar, Telegram-ботом та іншими сервісами.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Розробка вебсистеми для бронювання послуг салону краси є важливим завданням, яке спрямоване на створення зручної та ефективної онлайн-платформи для запису клієнтів до салонів краси, перукарень, косметологічних кабінетів, студій манікюру та SPA-центрів. Суть проєкту полягає у забезпеченні користувачів можливістю легко переглядати послуги та майстрів, створювати та керувати бронюваннями, а також отримувати сповіщення про статус записів. Актуальність таких систем зростає з кожним роком, оскільки все більше клієнтів віддають перевагу онлайн-запису замість традиційних телефонних дзвінків. За даними досліджень, салони краси, які впроваджують онлайн-бронювання, збільшують кількість клієнтів у середньому на 30% та зменшують кількість неявок на 40% завдяки автоматичним нагадуванням.

Основними елементами сайту є система реєстрації та авторизації, яка дозволяє розмежувати доступ для різних категорій користувачів: клієнтів, майстрів, адміністраторів та власників бізнесу. Крім того, ключову роль відіграють механізми управління послугами, розкладом майстрів, бронюваннями, а також система автоматичних сповіщень. Наприклад, майстер бачить лише свій розклад та запити до нього, тоді як адміністратор має доступ до всіх бронювань та може керувати каталогом послуг.

Для реалізації сайту необхідно виконати низку технічних завдань. Насамперед потрібно визначити стек технологій, який забезпечить ефективну розробку, підтримку та масштабованість проєкту. Важливо також обрати відповідний фреймворк, що дозволить прискорити процес створення сайту. При виборі технологій враховувалися такі критерії, як простота освоєння, наявність документації, розмір спільноти, продуктивність, безпека та вартість впровадження. Оптимальним рішенням став стек .NET Core для бекенду та

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

React.js для фронтенду, які забезпечують високу продуктивність та гнучкість розробки.

Наступним етапом є розробка структури бази даних. Вона повинна охоплювати інформацію про користувачів, майстрів, послуги, бронювання, платежі та відгуки. Одночасно потрібно передбачити надійні засоби безпеки, зокрема шифрування персональних даних. База даних спроектована таким чином, щоб мінімізувати дублювання інформації та забезпечити цілісність даних на рівні зовнішніх ключів. Нормалізація бази даних дозволяє уникнути аномалій при додаванні, оновленні та видаленні записів, що особливо важливо для системи з великою кількістю взаємопов'язаних сутностей.

Особливу увагу слід приділити користувацькому інтерфейсу. Його дизайн має бути інтуїтивно зрозумілим, адаптивним для різних типів пристроїв і доповнений інтерактивними елементами, що забезпечують комфортну взаємодію з платформою. При розробці інтерфейсу застосовувалися принципи UI/UX дизайну, зокрема ієрархія інформації, контрастність, єдиність стилю та зворотний зв'язок про дії користувача. Усі ключові дії, такі як створення бронювання або скасування запису, супроводжуються спливаючими підказками та діалогами підтвердження, що запобігає випадковим помилкам.

Функціональна частина сайту включає реалізацію механізмів реєстрації та авторизації користувачів, перегляду послуг та майстрів, створення, редагування й скасування бронювань, а також впровадження системи сповіщень для інформування клієнтів про статус їхніх записів. Унікальною особливістю системи є бонусна програма лояльності, яка автоматично нараховує бали за кожну виконану послугу. Клієнт може використовувати накопичені бали для оплати до 30% вартості наступних послуг, що стимулює повторні візити та підвищує лояльність до салону.

Для управління контентом необхідно створити адміністративну панель, яка дозволить контролювати послуги, майстрів, бронювання та модерувати відгуки. Безпека системи повинна бути посилена захистом від можливих атак,

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

таких як SQL-ін'єкції та XSS, а також впровадженням електронної верифікації користувачів. У розробленій системі безпека реалізована на кількох рівнях: на рівні бази даних (параметризовані запити), на рівні API (JWT-токени, перевірка ролей, CORS-політика) та на рівні клієнта (валідація введених даних, захист від XSS через екранування). Паролі користувачів зберігаються виключно у вигляді хешів із сіллю, що унеможлиблює їх відновлення навіть при компрометації бази даних.

Завершальним етапом є налаштування середовища для розгортання сайту та забезпечення його стабільної роботи. Підсумовуючи, основними завданнями є аналіз існуючих аналогів і їх можливостей, підготовка даних для наповнення системи, побудова структури бази даних, реалізація ролей користувачів із відповідними рівнями доступу, створення та налаштування адміністративної панелі, проектування архітектури системи, а також комплексне тестування перед запуском. Для забезпечення безперебійної роботи системи після розгортання передбачено механізм моніторингу стану сервера та автоматичне створення резервних копій бази даних.

Результатом реалізації проєкту стане сучасна онлайн-платформа для бронювання послуг салону краси, що відповідає актуальним потребам користувачів. Завдяки продуманій архітектурі, зручному інтерфейсу та широкому функціоналу, система забезпечуватиме комфортну взаємодію між клієнтами та майстрами. Використання надійного технологічного стеку, ефективної логіки на сервері та захищених API гарантує стабільну, швидку й безпечну роботу всіх компонентів платформи. Розроблена система є повністю готовою до впровадження в реальному бізнес-середовищі та може бути адаптована для інших сфер послуг, таких як медичні клініки, перукарні, косметологічні кабінети, фітнес-клуби та інші заклади, що працюють за попереднім записом. Модульна архітектура дозволяє легко додавати новий функціонал, наприклад, інтеграцію з Google Calendar, Telegram-ботом або SMS-розсилками, без необхідності переписувати існуючий код.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

Перш ніж розпочати розробку або вдосконалення програмного продукту, важливо провести аналіз існуючих рішень на ринку. Це дозволяє оцінити доступні можливості, визначити сильні та слабкі сторони конкурентів, а також знайти ідеї для покращення власного сервісу. Аналіз аналогів допомагає зрозуміти поточний стан галузі, виявити тенденції та сформуванати конкурентоспроможну стратегію розвитку продукту.

Під час аналізу ринку сервісів для онлайн-бронювання послуг салонів краси було виокремлено чотири провідні платформи: Appointer (Україна), (рис. 1.1). Anolla (Україна), Planity (Франція/Європа) та Shore (Німеччина/Європа). Нижче наведено детальний розгляд кожного з них, з акцентом на функціональні можливості, переваги та недоліки.

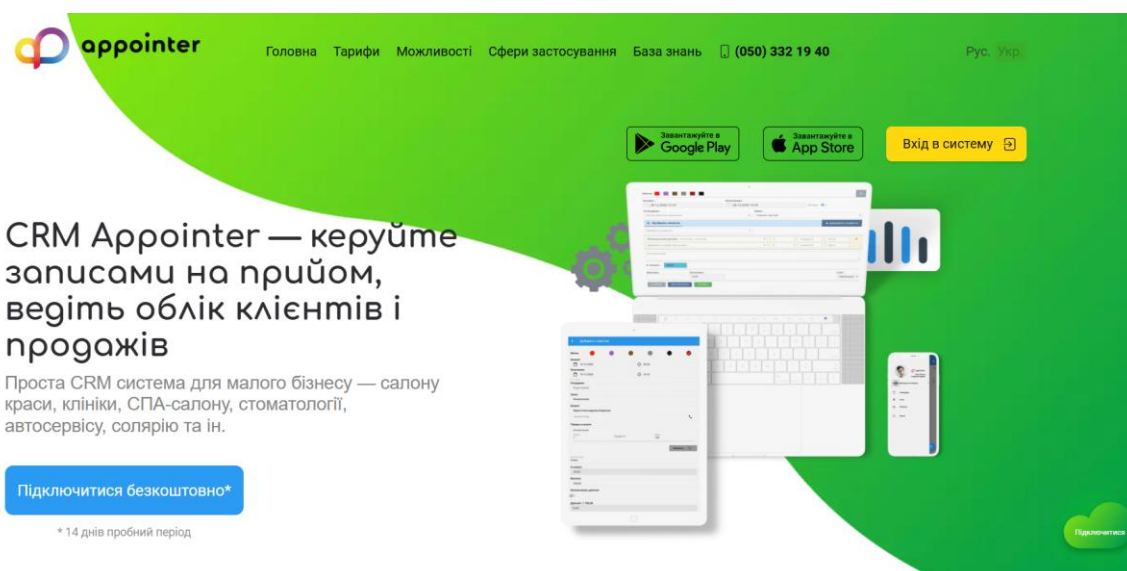


Рис. 1.1. Сервіс Appointer (Україна)

Appointer є українською програмою для управління записами на прийом, обліку клієнтів та ведення продажів, орієнтованою на салони краси, клініки, масажні кабінети, солярії, стоматології та автосервіси [1].

Платформа пропонує електронний календар записів з наочним відображенням зайнятості спеціалістів та залів, а також єдину базу клієнтів з детальною статистикою, яка включає історію замовлень, прибуток по кожному клієнту та кількість нових і втрачених клієнтів. Крім того, сервіс забезпечує онлайн-запис через вебсайт та соціальні мережі Facebook,

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Instagram, сповіщення клієнтам через SMS та Telegram з нагадуванням про візит, автоматичний розрахунок заробітної плати співробітників, облік складу та витратних матеріалів.

До переваг Appointer належать повна українська локалізація, широка географія клієнтів, потужний аналітичний функціонал, інтеграція з IP-телефонією та можливість автоматичного розрахунку зарплат. Основними недоліками є відсутність інтеграції з Google Calendar та обмежена інформація про наявність мобільного додатку для клієнтів [27]. Схожий функціонал у платформи Anolla (рис.1.2).

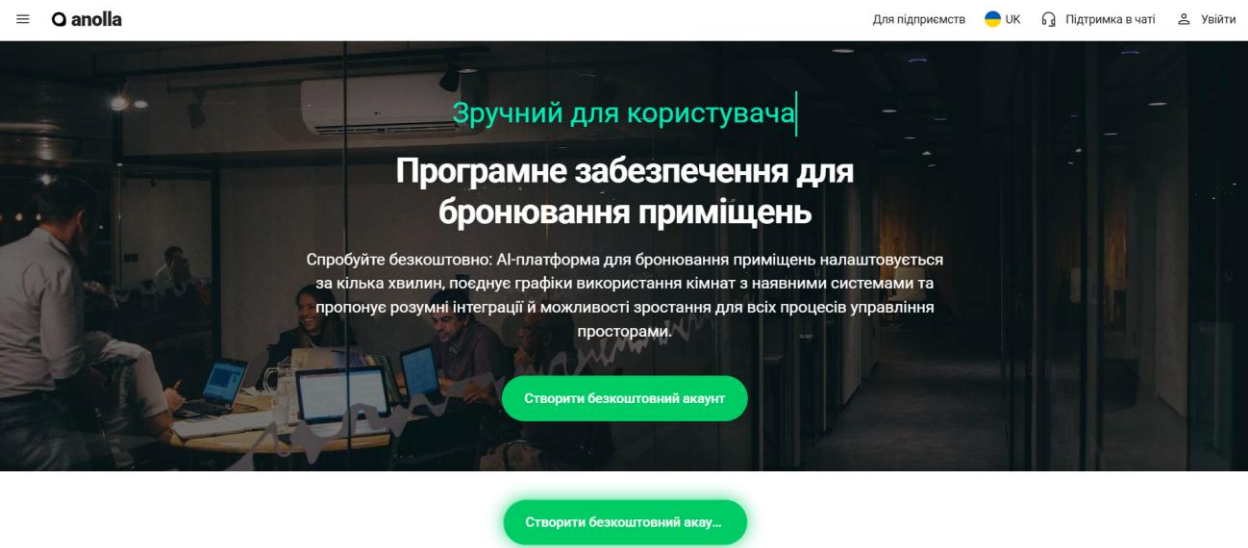


Рис. 1.2. Сервіс Anolla (Україна)

Anolla є сучасною українською платформою для автоматизації салонів краси, яка активно розвивається та пропонує зручний і сучасний інтерфейс, орієнтований насамперед на потреби сфери краси [2]. Платформа надає безкоштовний тариф з базовим функціоналом, що дозволяє опрацьовувати до п'ятдесяти записів на місяць, що є привабливою пропозицією для невеликих салонів та стартапів. Anolla інтегрується з Instagram, надаючи можливість запису клієнтів через Direct повідомлення, що значно розширює канали комунікації з аудиторією.

Крім того, система забезпечує автоматичні SMS та Email-нагадування клієнтам про майбутні візити, що зменшує кількість неявок. Важливою перевагою є повна українська локалізація та технічна підтримка українською

мовою, що робить платформу доступною для широкого кола користувачів без мовного бар'єру.

Разом з тим, Anolla має певні обмеження. Безкоштовна версія пропонує лише базовий функціонал, а для доступу до розширених можливостей необхідно переходити на платний тариф. Платформа не забезпечує складного обліку матеріалів, що може бути критичним для великих салонів з великою кількістю витратних матеріалів. Також Anolla поступається міжнародним аналогам у популярності, що може впливати на довіру потенційних клієнтів та обмежувати інтеграцію з іншими сервісами.

Незважаючи на ці недоліки, Anolla залишається конкурентоспроможним рішенням на українському ринку завдяки своїй орієнтації на локального користувача та зручному інтерфейсу. Закордонним аналогом у платформи Anolla є Planity (рис.1.3).

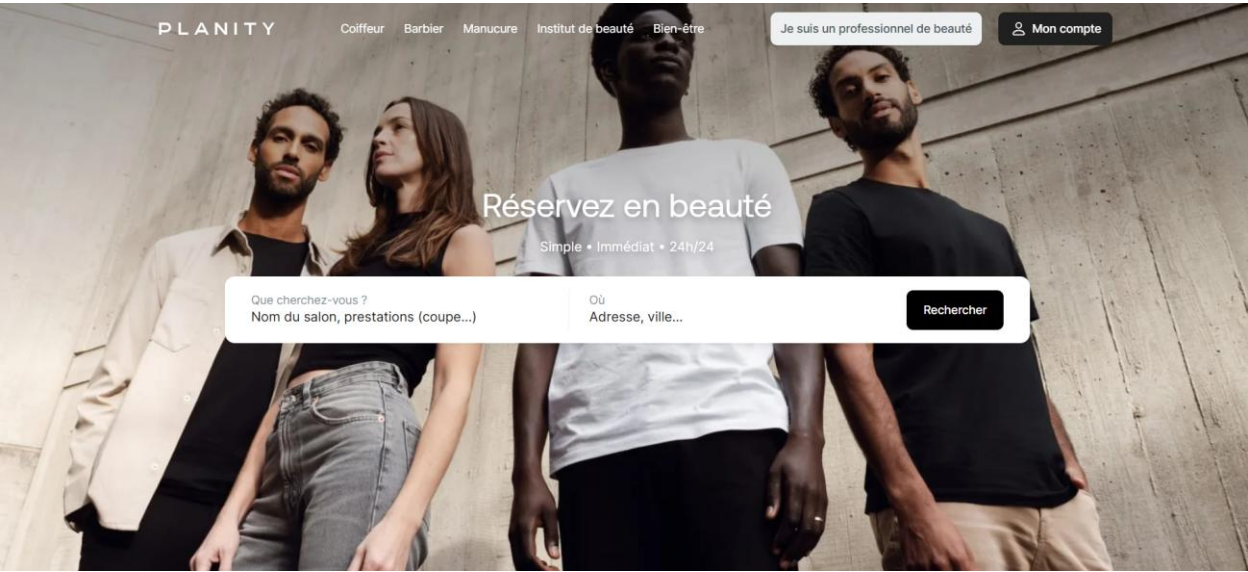


Рис. 1.3. Сервіс Planity (Франція)

Planity є інноваційним європейським рішенням All-in-One для салонів краси, перукарень та барбершопів, яке позиціонується як операційна система для індустрії краси [3]. Платформа надає онлайн-календар для керування записами, потужну платформу для бронювання з понад чотирмастами мільйонами заброньованих зустрічей та близько десятима мільйонами користувачів, касову систему для обробки платежів, інструменти для управління клієнтами та маркетингу, а також автоматичні SMS-нагадування,

які рятують три з чотирьох записів від неявок. Крім того, Planity забезпечує цілодобову онлайн-бронь, адже п'ятдесят відсотків бронювань відбуваються поза робочим часом салонів, та надає підтримку клієнтів і допомогу в налаштуванні.

До переваг Planity належить статус одного з лідерів ринку в Європі, особливо у Франції та Німеччині, а також величезна база користувачів, що створює додатковий потік клієнтів для салонів, які підключаються до платформи. Важливою перевагою є відсутність додаткових комісій за заброньовані зустрічі – салони не платять додаткових зборів, що робить сервіс фінансово вигідним. Також Planity пропонує потужні маркетингові інструменти для просування послуг.

Основними недоліками Planity є відсутність української локалізації, що ускладнює використання платформи українськими майстрами та клієнтами, платна підписка, яка може бути недоступною для невеликих салонів, та орієнтація на європейський ринок, що знижує її привабливість в Україні. Незважаючи на ці обмеження, Planity залишається одним із найпотужніших рішень у сфері бронювання послуг краси в Європі. Конкуруючим сервісом може виступати Shore (рис.1.4).

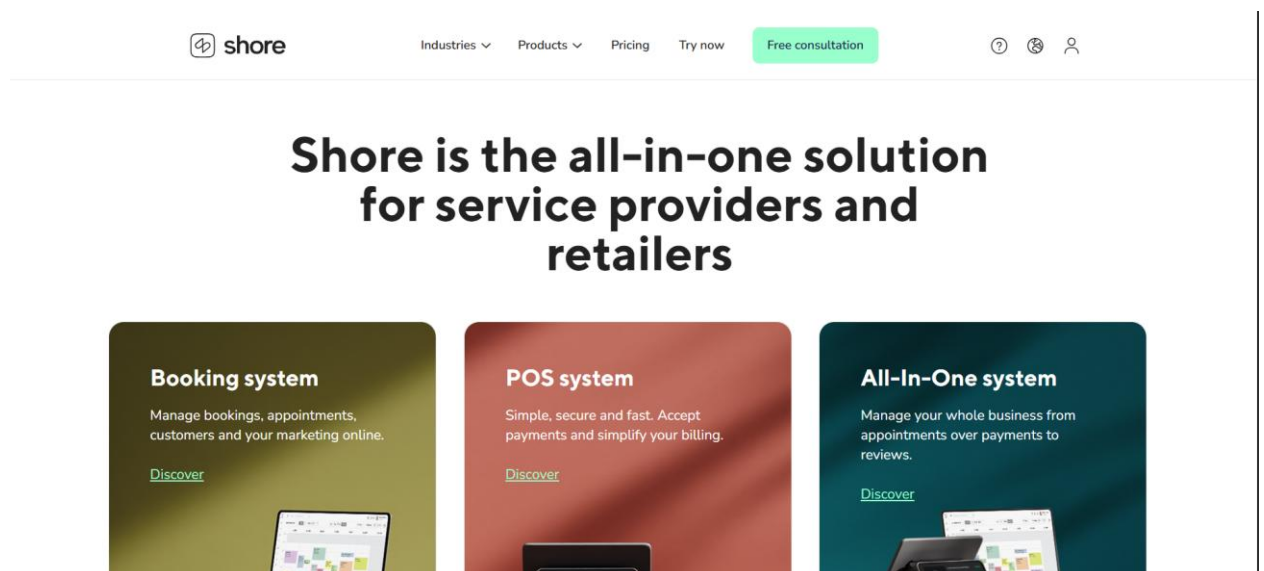


Рис. 1.4. Сервіс Shore (Німеччина)

Shore є хмарним програмним забезпеченням з Мюнхена, призначеним для оптимізації операцій салонів краси та косметичних студій [4]. Компанія

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

була заснована в 2012 році та має команду зі ста сорока двох співробітників. Платформа пропонує онлайн-систему запису 24/7 з автоматичними нагадуваннями для зменшення кількості неявок, управління клієнтами з централізованим зберіганням всієї інформації та історії бронювань, а також інструменти для маркетингу, зокрема таргетовані email та SMS-кампанії. Крім того, Shore надає допомогу у створенні професійного вебсайту та інтеграцію з соціальними мережами, безпечну обробку платежів через Shore Pay та POS-систему, а також аналітику бізнесу для отримання цінної інформації про діяльність.

До переваг Shore належать німецька якість та надійність, відповідність європейським нормам захисту даних GDPR, інтегрована платіжна система, яка спрощує прийом оплат, та допомога у створенні вебсайту, що особливо корисно для невеликих салонів без власних ІТ-фахівців.

Основними недоліками Shore є відсутність української локалізації, що ускладнює роботу з платформою для україномовних користувачів, платна підписка, яка може бути значною для малого бізнесу, та менша популярність в Україні у порівнянні з локальними рішеннями, такими як Appointer та Anolla. Незважаючи на ці обмеження, Shore є надійним європейським рішенням з широкими можливостями для автоматизації салонів краси.

Для систематизації отриманої інформації та визначення сильних та слабких сторін кожної платформи, було проведено порівняльний аналіз за ключовими критеріями.

Таблиця 1.1

Порівняльний аналіз систем онлайн-бронювання послуг

№	Критерій порівняння	Appointer (Україна)	Anolla (Україна)	Planity (Франція)	Shore (Німеччина)	Мій проєкт
1	Країна походження	Україна	Україна	Франція	Німеччина	Україна
2	Українська локалізація	Повністю українською	Повністю українською	Відсутня	Відсутня	Повністю українською

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

продовж. табл. 1.1

3	Онлайн-запис 24/7	Присутній	Присутній	Присутній	Присутній	Присутній
4	Мобільний додаток для клієнтів	Немає даних	Відсутній	Присутній	Немає даних	PWA (вебдодаток)
5	Автоматичні нагадування	SMS, Telegram	SMS, Email	SMS	Email, SMS	Email, SMS
6	Інтеграція з соцмережами	Facebook, Instagram	Instagram (Direct)	Немає даних	Facebook, Instagram	Заплановано
7	Управління розкладом майстрів	Присутнє	Присутнє	Присутнє	Присутнє	Присутнє
8	CRM (клієнтська база)	Присутня	Присутня	Присутня	Присутня	Присутня
9	Облік складських запасів	Присутній	Відсутній	Немає даних	Немає даних	Заплановано
10	Інтеграція з IP-телефонією	Присутня	Відсутня	Немає даних	Немає даних	Відсутня
11	Фінансова звітність та аналітика	Детальна	Базова	Присутня	Присутня	Поглиблена
12	Безкоштовний тариф	Немає даних	Присутній (обмежений)	Немає даних	Немає даних	Не передбачено
13	Технічна підтримка українською	Присутня	Присутня	Відсутня	Відсутня	Присутня

Висновок за таблицею. Проведений порівняльний аналіз показує, що на українському ринку представлені як локальні рішення (Appointer, Anolla), так і європейські (Planity, Shore). Appointer вирізняється широким функціоналом (облік складу, інтеграція з IP-телефонією, автоматичний розрахунок зарплат), тоді як Anolla пропонує сучасний інтерфейс та безкоштовний тариф. Planity та Shore є потужними європейськими гравцями, але не мають української

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

локалізації та підтримки. Мій проєкт орієнтується на українського користувача з повною локалізацією та PWA-підходом, але на початковому етапі поступається конкурентам у кількості інтеграцій. Для підвищення конкурентоспроможності доцільно розвивати функціонал з урахуванням сильних сторін Appointer (облік складу) та європейських систем (аналітика, інтеграції).

1.3. Вибір архітектури веборієнтованої системи

Вибір архітектури веборієнтованої системи є важливим етапом розробки, оскільки він визначає загальну структуру, продуктивність, масштабованість та зручність подальшої підтримки проєкту. Від правильно обраної архітектури залежить ефективність роботи всіх компонентів системи та їх взаємодія між собою. Неправильний вибір архітектури на початкових етапах може призвести до значних труднощів при масштабуванні системи, ускладнити додавання нового функціоналу та збільшити витрати на підтримку програмного забезпечення.

У цьому проєкті було прийнято рішення використовувати RESTful API архітектуру, оскільки вона дозволяє стандартизувати взаємодію між клієнтською та серверною частинами, забезпечуючи гнучкість у розвитку та підтримці. Даний підхід спрощує інтеграцію нових функцій, а також дає можливість легко масштабувати систему при збільшенні навантаження [30]. RESTful API сприяє покращенню структурованості коду, забезпечує чітке розділення відповідальностей між компонентами системи та спрощує обмін даними між клієнтом і сервером, уніфікуючи формат запитів і відповідей [14]. Це, своєю чергою, полегшує розробку та подальшу підтримку системи, оскільки всі запити до ресурсів виконуються за однаковими принципами з використанням стандартних HTTP-методів, таких як GET для отримання даних, POST для створення нових ресурсів, PUT для оновлення існуючих та DELETE для їх видалення.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Серверну частину реалізовано на платформі .NET Core з використанням ASP.NET Core Web API у вигляді монолітної архітектури з модульним поділом коду за функціональними областями. Такий підхід є виправданим для даного проекту, оскільки забезпечує просту та гнучку структуру, яка дозволяє швидко налаштовувати маршрути, middleware та логіку обробки запитів [15]. Код організовано у вигляді окремих контролерів, кожен з яких відповідає за певну сутність системи. Зокрема, AuthController забезпечує реєстрацію, авторизацію та управління обліковими записами користувачів з використанням JWT-токенів для захисту API-ендпоінтів. BookingsController реалізує повний цикл роботи з бронюваннями, включаючи створення, підтвердження, скасування та зміну статусу записів. MastersController надає доступ до інформації про майстрів салону, включаючи їх спеціалізацію, досвід та рейтинг. ServicesController відповідає за управління каталогом послуг, їх цінами та тривалістю. Така модульна організація дозволяє легко знаходити та змінювати потрібні частини коду, не порушуючи роботу інших компонентів системи.

ASP.NET Core Web API забезпечує високу продуктивність завдяки асинхронній моделі обробки запитів, що дозволяє ефективно використовувати ресурси сервера при одночасній роботі багатьох користувачів [7]. Вбудований механізм dependency injection спрощує тестування окремих компонентів та полегшує підтримку коду. Крім того, ASP.NET Core має вбудовану підтримку CORS, що дозволяє налаштувати безпечну взаємодію між клієнтською частиною, яка працює на окремому порту [22] [26].

Клієнтська частина реалізована за клієнт-серверною архітектурою і побудована з використанням бібліотеки React.js. Вона відповідає за відображення інтерфейсу користувача та взаємодію з сервером через REST API за допомогою HTTP-запитів. Для стилізації інтерфейсу використовується бібліотека Material-UI, яка надає набір готових компонентів, що відповідають принципам сучасного дизайну [6]. Це дозволяє створити візуально привабливий та зручний інтерфейс без необхідності писати велику кількість

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

CSS-коду. Компоненти Material-UI є адаптивними, що забезпечує коректне відображення сторінок як на настільних комп'ютерах, так і на мобільних пристроях.

Архітектурно клієнтська частина відповідає патерну Model-View-ViewModel, де модель відповідає за отримання і збереження даних з сервера, логіка інкапсульована у кастомних React-хуках, а сам інтерфейс реалізовано за допомогою React-компонентів. Такий підхід забезпечує модульність, підтримуваність та тестованість клієнтської частини. Кастомні хуки, такі як useAuth для управління авторизацією, useBooking для роботи з бронюваннями та useSettings для налаштувань інтерфейсу, дозволяють перевикористовувати логіку між різними компонентами та спрощують тестування [31]. Кожна сторінка вебзастосунку реалізована як окремий компонент, що відповідає за відображення відповідного маршруту.

Для зберігання даних використовується легковагова реляційна база даних SQLite, яка інтегрується з серверною частиною через Entity Framework Core [8]. Вибір SQLite обумовлений простотою розгортання – база даних зберігається у вигляді одного файлу beautybooking.db, що не потребує окремого сервера та спрощує резервне копіювання [9]. Це особливо важливо для невеликих та середніх проєктів, де адміністрування окремого сервера баз даних є надлишковим. Entity Framework Core виконує роль ORM, що дозволяє працювати з даними у вигляді об'єктів C#, автоматизує створення міграцій для оновлення схеми бази даних та виконує оптимізовані SQL-запити без необхідності писати їх вручну [23].

Для забезпечення безпеки паролів користувачів використовується бібліотека BCrypt, яка забезпечує надійне хешування з використанням солі. Це унеможливляє відновлення оригінальних паролів навіть у разі компрометації бази даних, оскільки кожен пароль хешується індивідуально з додаванням випадкової послідовності символів. BCrypt також є адаптивним алгоритмом, що дозволяє збільшувати складність обчислень з часом для компенсації зростання обчислювальної потужності атакуючих систем.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Авторизація користувачів реалізована з використанням JWT-токенів [10]. Після успішної авторизації сервер генерує унікальний токен, який зберігається на стороні клієнта у localStorage та додається до заголовків кожного наступного запиту до захищених ендпоінтів API [29]. Такий підхід дозволяє уникати зберігання стану сесії на сервері, що спрощує масштабування системи. Токен містить інформацію про користувача, зокрема його ідентифікатор та роль, що дозволяє належним чином обмежувати доступ до різних функцій системи. Для різних типів користувачів, а саме для звичайних клієнтів та майстрів, використовуються окремі ключі зберігання токенів (beauty_token_client та beauty_token_master), що дозволяє одночасно бути авторизованим в обох ролях та перемикатися між ними без повторного введення облікових даних.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки серверної частини було обрано платформу .NET Core з використанням фреймворку ASP.NET Core Web API. Цей вибір зумовлений кількома важливими факторами. По-перше, .NET Core є крос-платформенною платформою, що дозволяє запускати розроблений застосунок на операційних системах Windows, Linux та macOS без необхідності змінювати код. По-друге, ASP.NET Core забезпечує високу продуктивність завдяки асинхронній моделі обробки запитів, що особливо важливо для вебзастосунків з великою кількістю одночасних користувачів. По-третє, фреймворк має вбудовані механізми для роботи з JWT-авторизацією, CORS, dependency injection, що спрощує реалізацію типових завдань та зменшує кількість шаблонного коду. Крім того, .NET Core має велику спільноту розробників та детальну документацію, що полегшує пошук рішень для типових проблем та прискорює процес розробки. Додатковою перевагою є вбудована підтримка логування через Microsoft.Extensions.Logging, що дозволяє відстежувати помилки та стан роботи застосунку в реальному часі. Також ASP.NET Core надає можливість

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

легко налаштовувати обробку винятків за допомогою middleware, що дозволяє централізовано керувати помилками та повертати клієнту стандартизовані відповіді у форматі JSON. Крім того, наявність вбудованих засобів тестування, зокрема інтеграції з xUnit та Moq, дозволяє створювати модульні тести для перевірки коректності роботи контролерів та сервісів без необхідності запуску всього застосунку. Це значно підвищує надійність коду та спрощує виявлення помилок на ранніх етапах розробки.

Для хешування паролів користувачів використовується бібліотека BCrypt.Net. Ця бібліотека реалізує алгоритм адаптивного хешування, який автоматично генерує випадкову сіль для кожного пароля, що унеможливорює використання заздалегідь обчислених таблиць хешів для підбору паролів. Крім того, BCrypt дозволяє регулювати обчислювальну складність хешування, що дає змогу збільшувати час обчислення хешу в міру зростання продуктивності комп'ютерів, зберігаючи стійкість до атак навіть через багато років після впровадження системи [11]. Варто зазначити, що BCrypt є стандартом де-факто для зберігання паролів у багатьох мовах програмування завдяки своїй надійності та простоті використання. На відміну від простих хеш-функцій, таких як MD5 або SHA-1, BCrypt спеціально розроблений для повільного обчислення, що ускладнює атаки перебором. Крім того, бібліотека забезпечує зворотну сумісність: при оновленні work factor старі хеші залишаються працездатними, а нові паролі хешуються з новими налаштуваннями. Це дозволяє поступово підвищувати складність хешування без необхідності примусової зміни всіх паролів користувачів. У разі компрометації бази даних злоумисник отримує лише хеші, відновлення яких потребує величезних обчислювальних ресурсів, що робить атаку практично неможливою.

Клієнтська частина реалізована за допомогою бібліотеки React.js [5]. Цей фреймворк був обраний через його компонентний підхід, що дозволяє ефективно керувати інтерфейсом, забезпечує повторне використання коду та спрощує розширення функціональності. Використання віртуального DOM дозволяє оптимізувати рендеринг сторінки, змінюючи лише ті частини

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу, які дійсно змінилися, що позитивно впливає на продуктивність та плавність анімацій. Для стилізації інтерфейсу застосовано бібліотеку Material-UI, яка надає великий набір готових компонентів, що відповідають принципам Material Design. Це дозволяє створювати сучасний та візуально привабливий інтерфейс без необхідності розробляти власні компоненти з нуля, що значно пришвидшує процес розробки. Компоненти Material-UI є адаптивними та доступними, що забезпечує коректне відображення на різних пристроях та відповідає сучасним стандартам доступності вебконтенту. React також підтримує концепцію односпрямованого потоку даних (unidirectional data flow), що значно спрощує налагодження та передбачення поведінки застосунку. Завдяки цьому будь-яка зміна стану призводить до передбачуваного оновлення інтерфейсу, а помилки легше локалізувати. Крім того, велика екосистема React включає такі інструменти, як React DevTools для браузера, що дозволяє в реальному часі аналізувати компоненти, їхні props, state та продуктивність рендерингу. Це особливо корисно при оптимізації складних інтерфейсів з великою кількістю динамічних елементів. Для маршрутизації в застосунку використовується React Router, який забезпечує декларативну навігацію між сторінками без перезавантаження браузера, що створює відчуття роботи з нативним застосунком.

Для зберігання даних у проєкті обрано легковагову реляційну базу даних SQLite. Цей вибір зумовлений простотою розгортання та адміністрування. База даних SQLite зберігається у вигляді одного файлу (beautybooking.db), що не потребує встановлення та налаштування окремого сервера баз даних. Це особливо важливо для невеликих та середніх проєктів, де витрати на адміністрування окремого сервера є надлишковими. Крім того, SQLite підтримує стандартні SQL-запити, транзакції та обмеження цілісності, що забезпечує надійне зберігання даних. Незважаючи на свою легковаговість, SQLite забезпечує достатню продуктивність для типових операцій вебзастосунку, таких як створення, читання, оновлення та видалення записів, особливо при невеликій кількості одночасних користувачів. Серед додаткових

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

переваг SQLite варто відзначити вбудовану підтримку зовнішніх ключів та каскадних операцій, що дозволяє автоматично підтримувати цілісність зв'язків між таблицями. Наприклад, при видаленні користувача можна автоматично видалити всі його бронювання або встановити значення NULL у відповідних полях. Крім того, SQLite підтримує тригери, які дозволяють виконувати автоматичні дії при зміні даних, наприклад, оновлювати бонусний баланс або логувати зміни статусу бронювань. Важливою перевагою є також відсутність необхідності в налаштуванні прав доступу на рівні СКБД – вся безпека реалізується на рівні файлової системи та застосунку. Для резервного копіювання достатньо просто скопіювати файл beautybooking.db, що робить процес відновлення даних максимально простим.

Для взаємодії з базою даних використовується ORM Entity Framework Core. Ця бібліотека дозволяє працювати з даними у вигляді об'єктів C#, що позбавляє розробника від необхідності писати SQL-запити вручну. Entity Framework Core автоматично генерує оптимізовані SQL-запити на основі LINQ-виразів, що знижує ймовірність помилок та прискорює розробку. Крім того, Entity Framework Core підтримує механізм міграцій, який дозволяє автоматично оновлювати схему бази даних при зміні моделей даних без втрати існуючої інформації [32]. Це особливо корисно на етапах розробки та супроводу проєкту, коли структура даних може змінюватися. Entity Framework Core також забезпечує механізм відкладених запитів (lazy loading), коли пов'язані дані завантажуються тільки в момент першого звернення до них, що дозволяє оптимізувати продуктивність при роботі з великими обсягами даних. Окрім того, ORM підтримує жадібне завантаження (eager loading) через метод Include(), яке дозволяє за один запит отримати основну сутність разом з усіма пов'язаними даними. Це особливо актуально для сторінки бронювання, де потрібно відобразити інформацію про клієнта, майстра та послугу одночасно. Важливою функцією є також автоматичне відстеження змін (change tracking): Entity Framework Core запам'ятовує початковий стан об'єктів при завантаженні, а при виклику SaveChanges() оновлює лише ті поля, які реально

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

змінилися. Це зменшує кількість даних, що передаються до бази даних, та підвищує продуктивність оновлень.

Для виконання HTTP-запитів до серверної частини з клієнтської використовується вбудований механізм `fetch` разом з бібліотекою `axios`, яка надає зручний API для роботи з асинхронними запитами, підтримує перехоплювачі запитів та відповідей, що дозволяє централізовано обробляти помилки авторизації та додавати токени до заголовків запитів [17]. Для управління станом авторизації та маршрутизацією використовуються вбудовані засоби `React`, зокрема контексти та хуки, що дозволяє створювати приватні маршрути, доступні лише авторизованим користувачам. Додатково `axios` підтримує скасування запитів через `AbortController`, що дозволяє переривати застарілі запити при швидкій зміні параметрів фільтрації або при переході користувача на іншу сторінку. Це запобігає ситуаціям, коли відповідь від сервера приходить вже після того, як відповідний компонент був знищений, що могло б спричинити помилки в консолі. У системі реалізовано централізований перехоплювач відповідей, який автоматично обробляє помилки авторизації: якщо сервер повертає статус 401 (Unauthorized), користувач перенаправляється на сторінку входу, а його токен видаляється з локального сховища. Це забезпечує єдиний підхід до обробки неавторизованого доступу без необхідності дублювати логіку в кожному компоненті. Крім того, `axios` дозволяє налаштовувати базову URL-адресу для всіх запитів, що спрощує зміну адреси API при розгортанні системи на різних середовищах (розробка, тестування, продакшен).

Висновки до першого розділу

У першому розділі визначено основні завдання та технічні вимоги для розробки вебсистеми бронювання послуг салону краси. Проведено аналіз існуючих українських та європейських рішень із визначенням їх функціональності, переваг і недоліків на основі порівняння з аналогами, зокрема `Appointer`, `Anolla`, `Planity` та `Shore`. Встановлено, що більшість існуючих систем або не мають української локалізації, або мають

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

надлишковий функціонал та високу вартість підписки, що підтверджує актуальність створення власної спеціалізованої вебсистеми.

Архітектура системи обрана з урахуванням вимог проєкту. Серверна частина реалізована на платформі .NET Core з використанням ASP.NET Core Web API, що забезпечує високу продуктивність, крос-платформеність та можливість масштабування. Клієнтська частина побудована на бібліотеці React.js за клієнт-серверною моделлю, що дозволяє створювати динамічний та інтерактивний інтерфейс користувача.

Для реалізації системи вибрано наступний стек технологій. Як система керування базами даних обрано SQLite – легковагову реляційну базу даних, яка зберігається у вигляді одного файлу та не потребує окремого сервера, що спрощує розгортання та адміністрування. Для взаємодії з базою даних використовується ORM Entity Framework Core, який дозволяє працювати з даними у вигляді об'єктів C# та автоматизує створення міграцій. Для побудови користувацького інтерфейсу використовується React.js разом з бібліотекою компонентів Material-UI, що забезпечує сучасний та адаптивний дизайн. Для забезпечення безпеки паролів застосовується бібліотека BCrypt, а для авторизації – JWT-токени, які дозволяють розмежувати доступ для різних типів користувачів (клієнтів та майстрів). Такий вибір технологій забезпечує високу продуктивність, гнучкість розробки та належний рівень безпеки системи.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Аналіз вимог охоплює визначення та уточнення як функціональних, так і нефункціональних вимог до системи, що дозволяє сформулювати повне уявлення про те, які завдання має вирішувати платформа та яким критеріям якості вона повинна відповідати [19]. Функціональні вимоги описують ключові завдання та можливості системи, які необхідно реалізувати для забезпечення її цільового призначення та відповідності очікуванням користувачів [20]. Визначення цих вимог є критично важливим етапом, оскільки саме на їх основі будується вся подальша архітектура системи, обираються технології та розробляються конкретні програмні рішення.

У межах сервісу передбачено наявність двох основних ролей користувачів. Майстер є працівником салону краси, який безпосередньо надає послуги. Він має доступ до власного розкладу, може підтверджувати запити на бронювання, відзначати виконані записи та скасовувати бронювання. Клієнт є кінцевим споживачем послуг салону краси, наділеним правами для запису на послуги, перегляду історії власних записів, редагування профілю та використання бонусної системи. Неаутентифікований користувач має обмежений набір доступних йому функцій.

Важливо зазначити, що розмежування прав доступу між ролями реалізовано як на рівні клієнтської частини, так і на рівні серверної. На клієнтській частині окремі маршрути захищені за допомогою механізму PrivateRoute, який перевіряє наявність відповідного JWT-токену в localStorage та перенаправляє неавторизованих користувачів на сторінку входу. На серверній частині кожен контролер перевіряє роль поточного користувача, отриману з JWT-токену, та надає доступ лише до дозволених операцій. Наприклад, майстер не може отримати доступ до бронювань іншого майстра, оскільки в методі GetMasterBookings контролера BookingsController

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

виконується перевірка, чи ідентифікатор майстра в запиті збігається з ідентифікатором авторизованого користувача.

Для неаутентифікованого користувача система надає можливість переглядати доступні послуги, ознайомлюватися з детальною інформацією про кожну з них, включно з ціною та тривалістю, а також переглядати профілі майстрів. Також доступна функція пошуку послуг. Окрім того, користувач може пройти реєстрацію або авторизацію через електронну пошту та пароль. Такий набір функцій дозволяє потенційним клієнтам ознайомитися з послугами салону та рівнем майстрів ще до створення облікового запису, що підвищує довіру до платформи та стимулює реєстрацію.

Після авторизації клієнт отримує доступ до розширеного функціоналу, що дозволяє створювати нові бронювання з перевіркою доступності часу, переглядати та скасовувати власні записи, а також змінювати свій профіль. Клієнт може переглядати всі наявні послуги, отримувати сповіщення про статус записів, користуватися бонусною системою, накопичувати бали та використовувати їх для отримання знижок. Крім того, доступна можливість налаштування інтерфейсу, зокрема вибір темної теми, зміна мови та валюти. Цей набір функцій робить платформу зручною та персоналізованою, дозволяючи кожному клієнту налаштувати її під свої потреби.

Майстер має доступ до перегляду власного розкладу, підтвердження або скасування запитів на бронювання, зміни статусу виконання послуги (відмітка про виконання). Майстер не має доступу до налаштувань системи або управління іншими користувачами. Таке обмеження прав майстра є свідомим архітектурним рішенням, оскільки основне завдання майстра – надання послуг та управління власним графіком роботи, а не адміністрування системи.

Для візуалізації функціональних вимог та взаємозв'язків між акторами і системними функціями побудовано діаграму варіантів використання, яку представлено на рис. 2.1 [18].

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

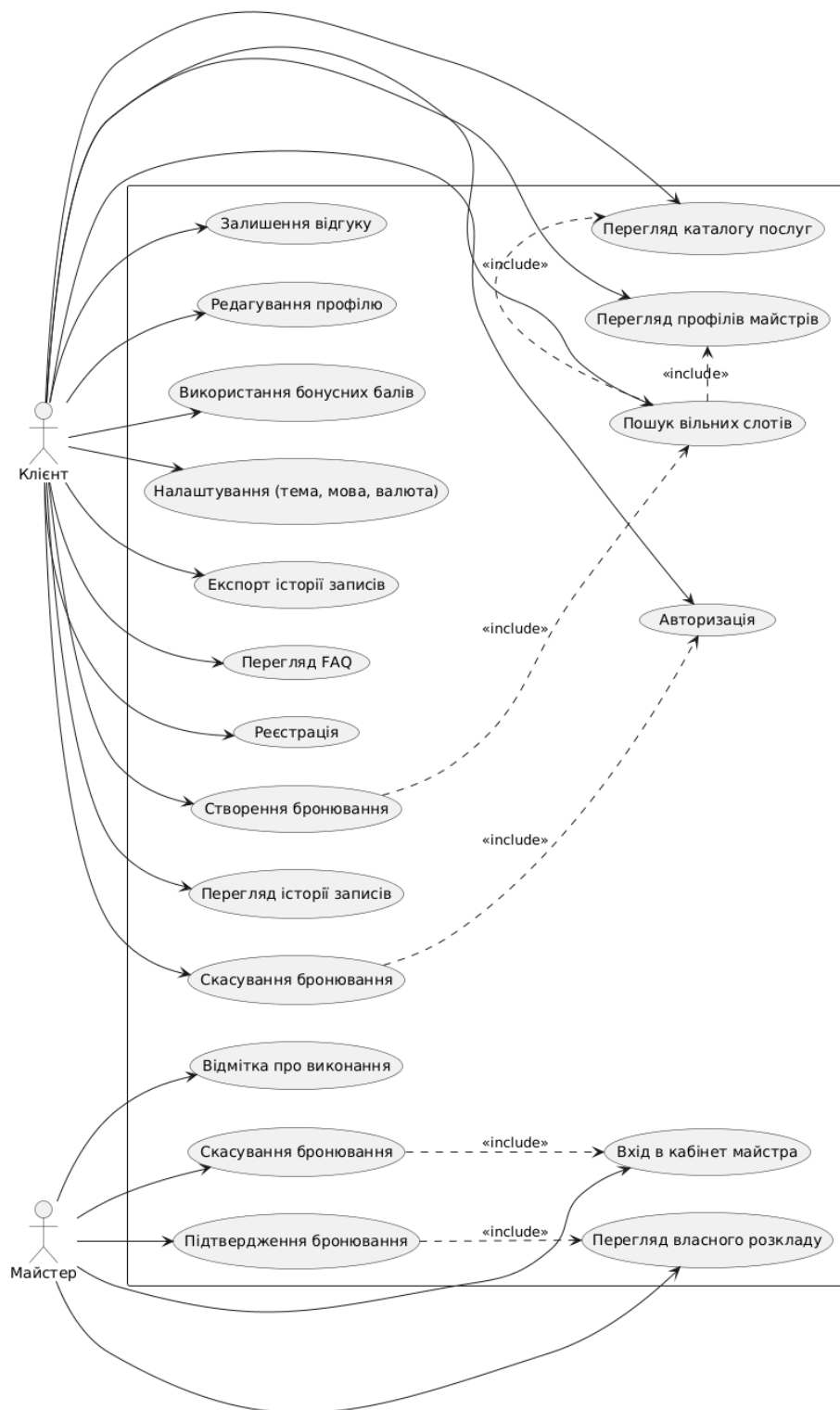


Рис. 2.1. Діаграма варіантів використання

Нефункціональні вимоги описують властивості та показники якості системи, що впливають на її ефективність, надійність і зручність використання [21]. Вони охоплюють кілька ключових аспектів. Однією з основних вимог є сприйняття та зручність використання. Інтерфейс користувача має бути

інтуїтивно зрозумілим, не вимагати спеціальної підготовки від користувачів та бути адаптованим до роботи на мобільних пристроях, що забезпечує доступність і комфорт у взаємодії з системою.

Продуктивність і масштабованість також є критично важливими. Система повинна забезпечувати час відгуку не більше двох секунд для основних операцій і до п'яти секунд для складних запитів. Водночас вона має підтримувати стабільну роботу за умови одночасного використання щонайменше однією тисячі користувачів без помітного зниження швидкодії.

Ще одним важливим аспектом є безпека. Усі дані користувачів повинні передаватися через захищені канали, а паролі зберігатися у вигляді хешів з використанням сучасних криптографічних алгоритмів, зокрема BCrypt. Крім того, програмне забезпечення має регулярно оновлюватися з метою усунення потенційних вразливостей. Останнім компонентом є сумісність. Система повинна коректно працювати в усіх основних сучасних веббраузерах, зокрема Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, а також підтримувати різні операційні системи, що гарантує широку доступність і зручність для користувачів із різними технічними платформами.

Архітектура серверної частини застосунку побудована за модульним принципом, що забезпечує гнучкість, масштабованість та легкість у підтримці й розширенні системи. Кожен функціональний блок винесений в окремий модуль, який містить у собі пов'язані компоненти: контролери, сервіси, сутності, DTO та інші допоміжні елементи. Такий підхід дозволяє, наприклад, замінити модуль роботи з базою даних без внесення змін до модуля бізнес-логіки, що підвищує стійкість системи до змін. Це дозволяє інкапсулювати логіку кожного домену в межах модуля, водночас відкриваючи необхідні функціональності для взаємодії з іншими модулями. Діаграму класів системи наведено на рис. 2.2.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

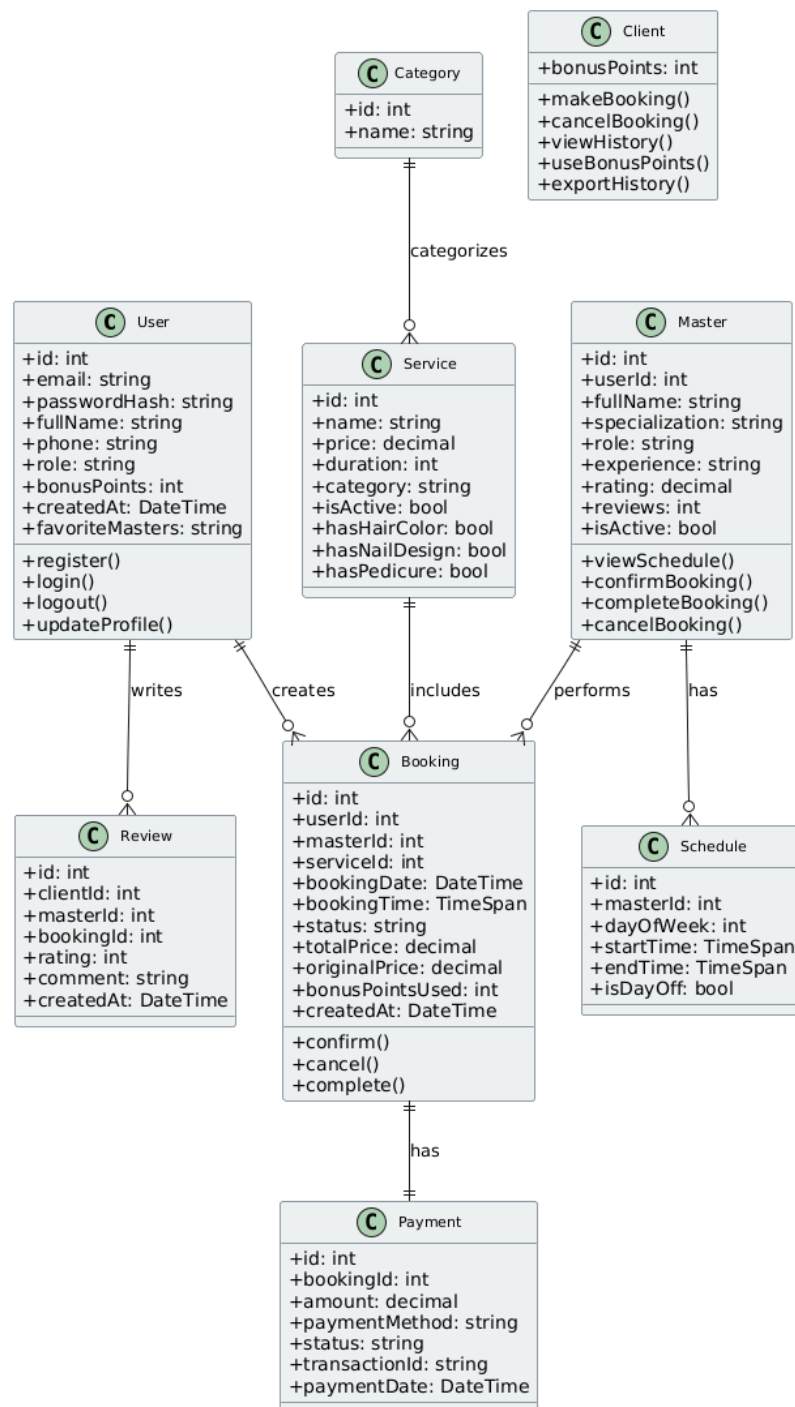


Рис. 2.2. Діаграма класів

Клас User є основною сутністю, що представляє користувача системи. Він містить ідентифікатор, електронну пошту, хеш пароля, повне ім'я, телефон, роль (Client або Master), бонусні бали, дату створення та список улюблених майстрів. Користувач має методи для реєстрації, входу, виходу та оновлення профілю.

Класи Client та Master є похідними ролями користувача. Client має бонусні бали та методи для створення бронювання, скасування бронювання,

перегляду історії, використання бонусів та експорту історії. Master має спеціалізацію, рейтинг, досвід та методи для перегляду розкладу, підтвердження бронювання, відмітки про виконання та скасування бронювання.

Клас Service представляє послугу салону з назвою, ціною, тривалістю, категорією та додатковими опціями. Клас Booking є центральною сутністю, що пов'язує користувача, майстра та послугу. Він містить дату, час, статус, ціни та кількість використаних бонусів. Класи Schedule, Review, Category та Payment є допоміжними сутностями для зберігання розкладу майстрів, відгуків, категорій послуг та інформації про платежі відповідно.

Зв'язки між класами є наступними. Користувач створює багато бронювань та пише багато відгуків. Майстер виконує багато бронювань та має власний розклад. Послуга входить до багатьох бронювань. Категорія групує багато послуг. Кожне бронювання має один платіж.

2.2. Розробка бази даних системи

Для зберігання всієї інформації, що обробляється сервісом, використовується легковагове рішення SQLite, яке є однією з найпоширеніших реляційних систем керування базами даних з відкритим вихідним кодом. Вибір SQLite обумовлений простотою розгортання та адміністрування, оскільки база даних зберігається у вигляді одного файлу beautybooking.db, що не потребує встановлення та налаштування окремого сервера баз даних. Це особливо важливо для невеликих та середніх проєктів, де витрати на адміністрування окремого сервера є надлишковими.

SQLite використовується для зберігання складної та зв'язаної моделі даних, яка включає облікові записи користувачів, профілі майстрів, каталог послуг, бронювання, платежі, відгуки та графіки роботи. Жорсткі обмеження цілісності, такі як зовнішні ключі, обмеження NOT NULL та унікальності, гарантують консистентність даних, а індекси забезпечують ефективну роботу

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

запитів. У результаті платформа отримує надійну, продуктивну та масштабовану основу для збирання і обробки даних.

Для вебсистеми бронювання послуг салону краси було спроектовано вісім реляційних таблиць, які повністю покривають потреби зберігання даних платформи. До цих таблиць належать Users, Masters, Services, Bookings, Schedules, Reviews та Payments. Діаграму бази даних наведено на рис. 2.3.

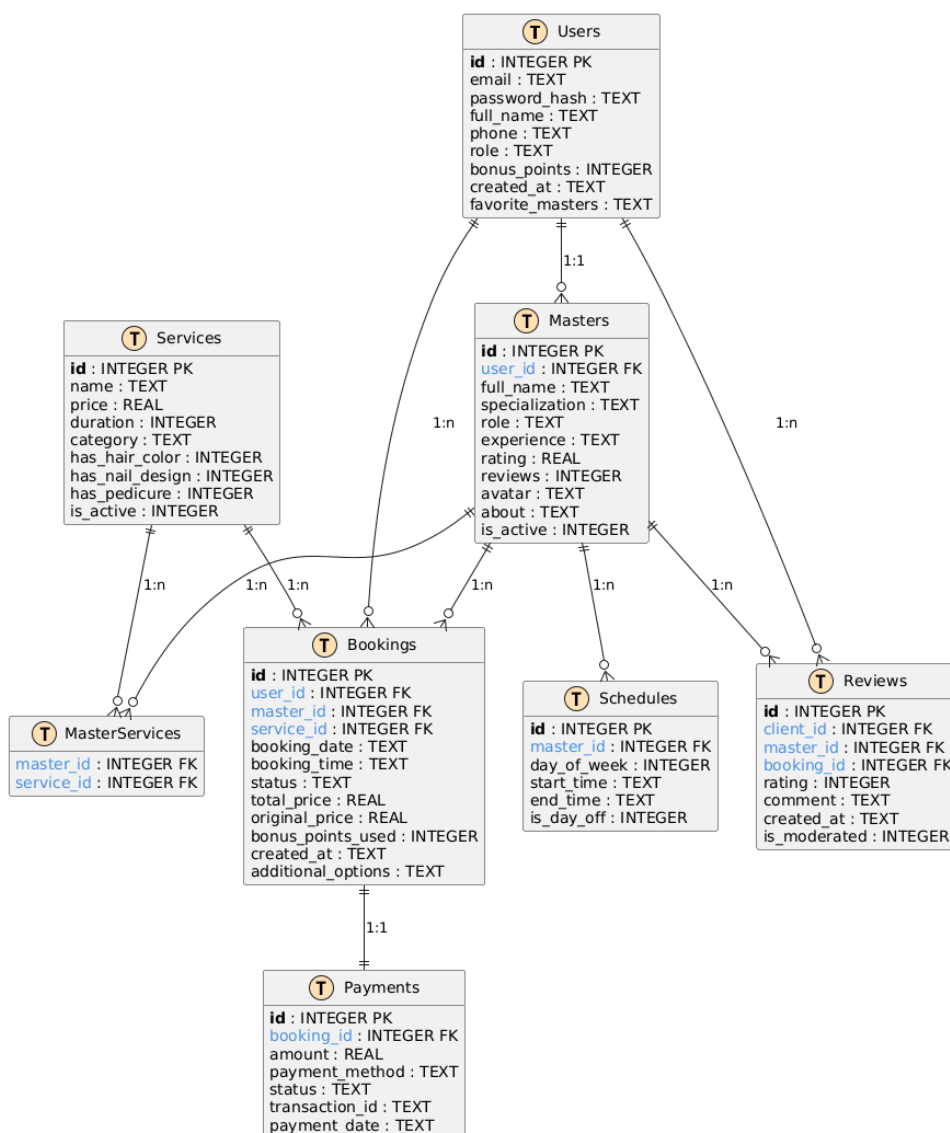


Рис. 2.3. Діаграма бази даних

Таблиця Users є основною таблицею, що зберігає облікові записи користувачів системи. Вона містить унікальний ідентифікатор користувача, електронну пошту, хеш пароля, повне ім'я, номер телефону, роль, яка може бути Client або Master, кількість бонусних балів, дату реєстрації та список

улюблених майстрів у форматі JSON. Саме на основі цієї таблиці відбувається авторизація користувачів та розмежування їх прав доступу.

Таблиця Masters містить детальну інформацію про майстрів салону краси. Вона зв'язана з таблицею Users через поле user_id відношенням один-до-одного, оскільки кожен майстер є зареєстрованим користувачем системи. Ця таблиця зберігає повне ім'я майстра, спеціалізацію, посаду, досвід роботи, рейтинг, кількість відгуків, посилання на аватар, текстовий опис та статус активності. На основі цих даних формується профіль майстра, який бачать клієнти при виборі спеціаліста для запису.

Таблиця Services зберігає каталог послуг, які надаються в салоні краси. Вона містить назву послуги, ціну, тривалість у хвилинах, категорію, а також логічні поля, що визначають наявність додаткових опцій, зокрема фарбування волосся, дизайн манікюру та педикюру. Це дозволяє динамічно відображати різні форми для запису залежно від обраної послуги.

Таблиця MasterServices є проміжною таблицею, яка реалізує зв'язок багато-до-багатьох між майстрами та послугами. Вона містить лише два поля – зовнішні ключі на майстра та на послугу. Завдяки цій таблиці один майстер може надавати багато різних послуг, а одна послуга може виконуватися багатьма майстрами, що відповідає реальній організації роботи салону краси.

Таблиця Schedules відповідає за зберігання графіків роботи майстрів. Вона містить день тижня, час початку та закінчення роботи, а також прапорець, що вказує, чи є день вихідним. На основі цієї таблиці та існуючих бронювань система генерує доступні часові слоти для запису клієнтів, виключаючи вже зайняті проміжки часу.

Таблиця Bookings є центральною таблицею системи, яка фіксує всі створені бронювання. Вона містить зовнішні ключі на користувача-клієнта, майстра та обрану послугу, дату та час запису, статус бронювання, який може бути Pending, Confirmed, Completed або Cancelled, загальну та оригінальну ціну, кількість використаних бонусів, дату створення та додаткові опції у

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

форматі JSON. Саме ця таблиця забезпечує основний функціонал системи – створення, підтвердження, виконання та скасування записів.

Таблиця Payments зберігає інформацію про фінансові транзакції, пов'язані з бронюваннями. Вона зв'язана з таблицею Bookings відношенням один-до-одного, оскільки кожне бронювання може мати лише один платіж. Таблиця містить суму платежу, спосіб оплати, який може бути Cash, Card або Online, статус платежу, ідентифікатор транзакції від платіжної системи та дату платежу.

Таблиця Reviews зберігає відгуки клієнтів про отримані послуги та майстрів. Вона містить зовнішні ключі на клієнта, майстра та бронювання, оцінку за шкалою від одного до п'яти, текст відгуку, дату створення та прапорець, що вказує, чи пройшов відгук модераторів. Ця таблиця дозволяє формувати рейтинг майстрів та надавати зворотний зв'язок від клієнтів.

Усі вісім таблиць пов'язані між собою за допомогою зовнішніх ключів, що забезпечує цілісність даних на рівні бази даних. Таблиця Users є центральною для зв'язку з Masters, Bookings та Reviews. Таблиця Masters пов'язана з MasterServices, Schedules, Bookings та Reviews. Таблиця Services пов'язана з MasterServices та Bookings. Таблиця Bookings пов'язана з Payments та Reviews. Така структура забезпечує повне покриття всіх бізнес-процесів системи бронювання послуг салону краси.

2.3. Проектування та реалізація алгоритмів роботи системи

Алгоритми мають ключове значення для забезпечення ефективної та коректної роботи системи, сприяючи вирішенню різноманітних завдань – від обробки запитів користувачів до управління потоками даних та їхньої обробки. Для моделювання алгоритмів роботи системи та відображення основних етапів взаємодії користувача з платформою для бронювання послуг салону краси були створені діаграми активності, які дозволяють детально описати послідовність дій у системі та ілюструють, як система реагує на різні дії користувачів. Діаграма активності є потужним інструментом для

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмоторя О.Г.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

відображення послідовності операцій, що відбуваються в межах бізнес-процесів. Вона показує, як різні дії взаємодіють між собою та в якому порядку вони виконуються. На рис. 2.4 представлено діаграму активності при першому вході на сайт.



Рис. 2.4. Діаграма активності при першому вході на сайт

Користувач починає взаємодію з системою, відкриваючи головну сторінку вебзастосунку. На цьому етапі йому пропонується три основні варіанти подальших дій: вхід в існуючий обліковий запис, реєстрація нового користувача або продовження роботи без авторизації в режимі гостя. Кожен з цих варіантів має власну послідовність кроків та умови успішного завершення.

У разі, якщо користувач обирає режим гостя, він отримує доступ до перегляду каталогу послуг та профілів майстрів, однак при спробі створення бронювання система вимагатиме авторизації, пропонуючи перейти до входу або реєстрації. Такий підхід дозволяє потенційним клієнтам ознайомитися з переліком послуг та рівнем майстрів перед тим, як прийняти рішення про створення облікового запису, що підвищує довіру до платформи та стимулює реєстрацію.

Якщо користувач вже має обліковий запис, він обирає опцію входу. Система відображає форму для введення електронної пошти та пароля. Після

відправлення форми запит надходить на сервер, де виконується перевірка наявності користувача з вказаною електронною поштою в таблиці Users. Якщо користувач знайдений, система за допомогою BCrypt порівнює введений пароль із збереженим хешем у базі даних, оскільки паролі ніколи не зберігаються у відкритому вигляді.

У разі успішної верифікації сервер генерує JWT-токен, який містить інформацію про ідентифікатор користувача, його електронну пошту, повне ім'я та роль. Токен повертається клієнту та зберігається в localStorage браузера для подальшої автентифікації запитів до захищених API-ендпоінтів. Залежно від ролі користувача – Client або Master – система виконує перенаправлення до відповідного кабінету: для клієнта це сторінка з переліком його записів та можливістю створення нових бронювань, а для майстра – сторінка з розкладом та списком запитів на бронювання. Якщо ж введені дані не відповідають жодному користувачеві, система відображає повідомлення про помилку та пропонує спробувати ще раз або скористатися відновленням пароля.

У випадку, коли користувач не пам'ятає пароль, він може ініціювати процес його скидання. Для цього необхідно вказати адресу електронної пошти, на яку буде надіслано лист з унікальним посиланням для створення нового пароля. Після переходу за цим посиланням користувач потрапляє на захищену сторінку, де може встановити новий пароль. Новий пароль перед збереженням хешується за допомогою BCrypt та оновлюється в базі даних. Такий механізм підвищує безпеку системи, оскільки паролі не передаються у відкритому вигляді, а процес відновлення вимагає доступу до електронної пошти користувача, що унеможливорює несанкціоновану зміну пароля.

Якщо ж користувач не має облікового запису, він обирає опцію реєстрації. Система відображає форму, в якій необхідно вказати електронну пошту, пароль, повне ім'я та номер телефону. Після заповнення даних виконується їх валідація як на стороні клієнта, так і на стороні сервера. Клієнтська валідація перевіряє правильність формату електронної пошти,

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

довжину пароля (не менше шести символів) та заповненість усіх обов'язкових полів.

Серверна валідація додатково перевіряє унікальність електронної пошти в таблиці Users, оскільки дублювання email-адрес не допускається. Якщо користувач з такою електронною поштою вже існує, система повертає відповідну помилку. Якщо всі дані введено коректно, сервер виконує хешування пароля за допомогою BCrypt, що забезпечує надійне зберігання паролів навіть у разі компрометації бази даних.

Після цього створюється новий запис у таблиці Users з роллю Client, а також нараховуються стартові бонусні бали в розмірі п'ятдесяти одиниць, які новий клієнт може використати для отримання знижки на перший запис. Після успішної реєстрації система автоматично виконує вхід користувача, генеруючи JWT-токен, зберігаючи його в localStorage та перенаправляючи користувача до його кабінету.

Усі описані дії повторюються циклічно до моменту успішного завершення процесу входу або реєстрації, при цьому кожна невдала спроба супроводжується відповідним повідомленням про помилку, що допомагає користувачеві зрозуміти причину та виправити її. Система після кожної невдалої спроби очищає форму або підсвічує поля, які потребують виправлення, що покращує користувацький досвід.

Крім того, для зручності користувачів реалізовано можливість відновлення пароля через електронну пошту, що дозволяє уникнути блокування облікового запису у разі втрати доступу. Завдяки інтуїтивно зрозумілим підказкам та валідації в реальному часі навіть нові користувачі без труднощів проходять процес реєстрації та авторизації. Такий підхід забезпечує зручний, безпечний та інтуїтивно зрозумілий процес автентифікації для всіх категорій користувачів системи. На рис. 2.5 представлено діаграму активності дій з бронюваннями.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

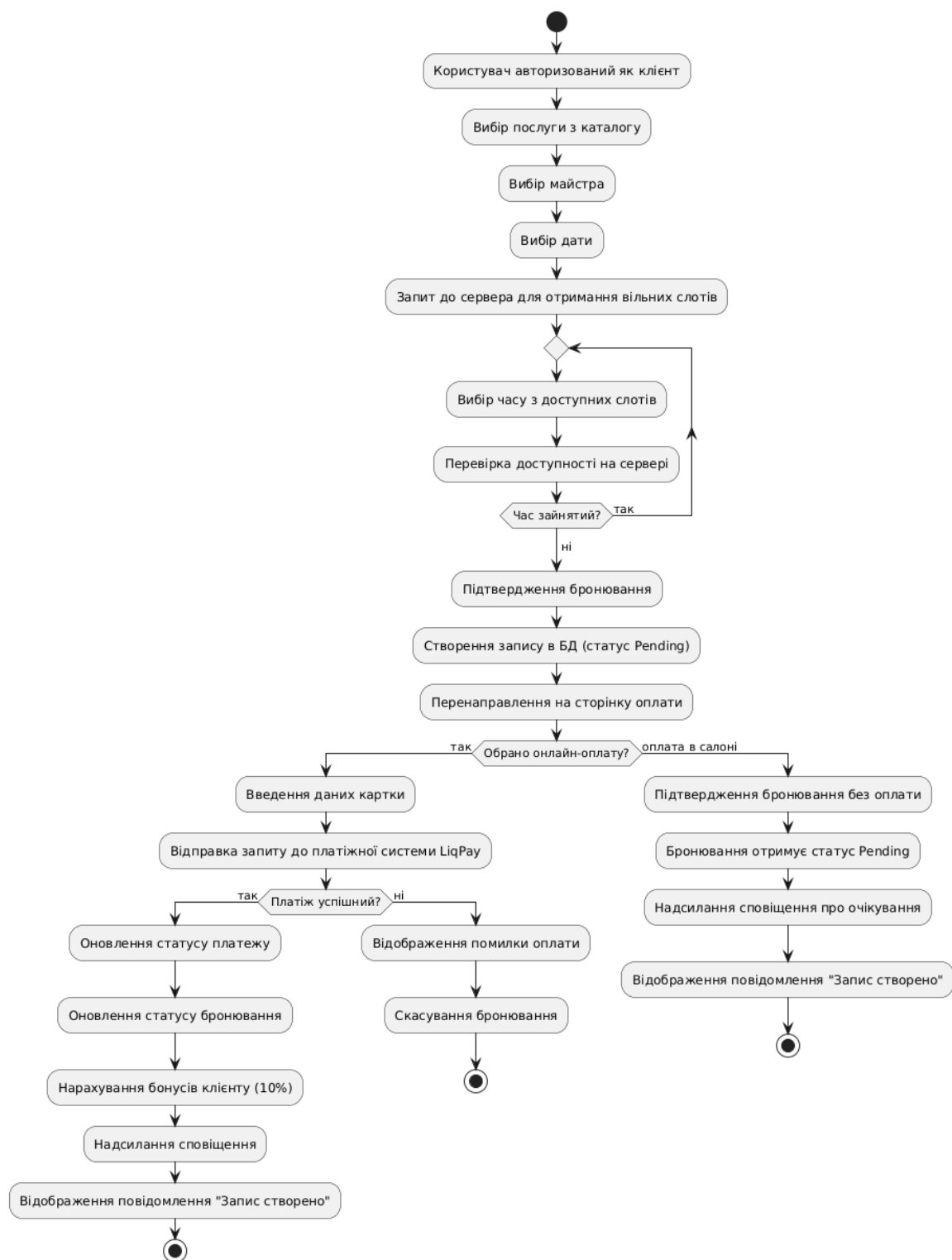


Рис. 2.5. Діаграма активності дій з бронюваннями

Після успішної авторизації клієнт отримує доступ до функціоналу створення бронювань. Процес починається з вибору послуги з каталогу, після чого система пропонує список майстрів, які надають обрану послугу. Клієнт обирає майстра, потім обирає зручну дату. Система відправляє запит на сервер для отримання вільних часових слотів на обрану дату, враховуючи графік

роботи майстра з таблиці Schedules та вже існуючі бронювання з таблиці Bookings. Якщо обраний час є вільним і не суперечить правилам системи, клієнт підтверджує бронювання. Система створює новий запис у таблиці Bookings зі статусом Pending, після чого клієнт переходить до сторінки оплати. Після успішної оплати бронювання зберігається в системі, і клієнт отримує сповіщення про успішне створення запису. У разі скасування бронювання процес переривається, і запис не створюється.

Майстер, у свою чергу, після входу до свого кабінету переглядає список усіх бронювань, призначених на нього. Для кожного бронювання зі статусом Pending майстер може підтвердити його, змінивши статус на Confirmed, або скасувати. Після виконання послуги майстер змінює статус бронювання на Completed, що також впливає на нарахування бонусних балів клієнту. У разі зміни статусу бронювання система автоматично оновлює відображення в кабінеті клієнта та, за необхідності, надсилає сповіщення.

2.4. Реалізація функціоналу системи

Перед початком розробки було спроектовано базу даних за допомогою графічних засобів, зокрема сервісу draw.io [28]. У візуальному редакторі створено макет майбутньої бази даних, визначено основні таблиці, їх поля, типи даних та налаштовано зв'язки між сутностями, такі як один-до-одного, один-до-багатьох та багато-до-багатьох.

На основі створеного макету бази даних реалізовано відповідну кодову структуру з використанням Entity Framework Core, який виконує роль ORM для роботи з базою даних SQLite у середовищі .NET Core. Кожна таблиця в макеті відповідає окремій сутності у коді, для чого створено класи, які описують структуру таблиць, їхні поля, типи, первинні ключі, зв'язки та інші обмеження. Entity Framework Core забезпечує автоматичне створення та виконання міграцій, що дозволяє легко оновлювати схему бази даних при зміні моделей даних без втрати існуючої інформації.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Роботу над проєктом розпочато з реалізації функціоналу авторизації, реєстрації та інших дій, пов'язаних безпосередньо з користувачем. Для забезпечення належного рівня безпеки та зручності цього процесу обрано механізм авторизації на основі JSON Web Token. У реалізації передбачено використання access токена, який має короткий термін дії та використовується для авторизації при кожному запиті до захищених маршрутів API. Access токен зберігається в localStorage на стороні клієнта, що дозволяє швидко додавати його до заголовків запитів.

Для різних типів користувачів, а саме для клієнтів та майстрів, використовуються окремі ключі зберігання токенів – beauty_token_client та beauty_token_master, що дозволяє одночасно бути авторизованим в обох ролях та перемикатися між ними без повторного введення облікових даних. Для хешування паролів використовується бібліотека BCrypt.Net, яка забезпечує надійне зберігання паролів за рахунок соління та адаптивного хешування, що унеможливорює відновлення оригінальних паролів навіть у разі компрометації бази даних.

Після створення основних дій з користувачем наступним кроком стала розробка повноцінного функціоналу для роботи з бронюваннями. Цей функціонал включає створення, перегляд, скасування бронювань, а також управління розкладом майстрів. Інтерфейс для роботи з бронюваннями спроектовано максимально зручним та інтуїтивно зрозумілим. При створенні бронювання клієнт обирає послугу, майстра, дату та час, причому система автоматично перевіряє доступність обраного часу, враховуючи графік роботи майстра з таблиці Schedules та вже існуючі бронювання з таблиці Bookings. Якщо обраний час є зайнятим, система повідомляє про це та пропонує обрати інший вільний проміжок часу. Крім того, при виборі сьогоднішньої дати система не відображає часові слоти, які вже минули, що запобігає спробам запису на час, який неможливо використати.

Для забезпечення зручної взаємодії між клієнтською та серверною частинами системи було розроблено REST API, документацію до якого

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

згенеровано за допомогою Swagger. На рис. 2.6 представлено структуру API-ендпоінтів розробленої вебсистеми.

Auth	^
Bookings	^
GET /api/Bookings/master/{masterId}	^
PUT /api/Bookings/{id}/confirm	^
PUT /api/Bookings/{id}/complete	^
PUT /api/Bookings/{id}/cancel	^
GET /api/Bookings/upcoming	^
GET /api/Bookings/history	^
Masters	^
GET /api/Masters	^
Profile	^
GET /api/Profile/{id}	^
PUT /api/Profile/{id}	^

Рис. 2.6. Swagger-документація API вебсистеми бронювання послуг салону краси

API містить наступні групи ендпоінтів. Група Auth відповідає за автентифікацію та управління користувачами, включаючи реєстрацію, вхід в систему та призначення ролей. Група Bookings реалізує повний цикл роботи з бронюваннями: отримання записів майстра, підтвердження, відмітку про виконання, скасування, а також отримання майбутніх та історичних записів. Групи Masters, Services та Profile надають ендпоїнти для роботи з майстрами, послугами та профілем користувача відповідно. Група TempBookings використовується для тимчасового зберігання бронювань перед оплатою. Така структура API забезпечує чітке розділення відповідальностей та спрощує розширення функціоналу системи в майбутньому.

Для підвищення зручності інформування користувачів реалізовано систему сповіщень. Коли бронювання отримує статус Confirmed, тобто підтверджено майстром, система автоматично додає нагадування в кабінет клієнта. Клієнт може переглядати всі активні нагадування у своєму кабінеті, а при скасуванні або виконанні запису нагадування автоматично видаляється. Такий підхід підвищує зручність моніторингу записів, адже клієнт завжди має

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

актуальну інформацію про свої майбутні візити без необхідності самостійно відстежувати їх.

Майстер має доступ до власного кабінету, де відображаються всі бронювання, призначені на нього. Для кожного бронювання зі статусом Pending майстер може підтвердити його, змінивши статус на Confirmed, або скасувати. Після виконання послуги майстер змінює статус бронювання на Completed. У разі зміни статусу бронювання система автоматично оновлює відображення в кабінеті клієнта та оновлює список нагадувань. Клієнт, у свою чергу, може скасувати бронювання зі статусом Pending або Confirmed, але не може скасувати вже виконаний запис.

З метою зменшення кількості запитів до сервера та підвищення чутливості інтерфейсу на клієнтській стороні використовується автоматичне кешування даних. Дані, отримані з API, зберігаються в кеші, що дозволяє уникнути повторних запитів до сервера при перегляді одних і тих самих сторінок. Кабінет майстра автоматично оновлюється кожні десять секунд, що дозволяє майстру бачити нові бронювання без необхідності оновлювати сторінку вручну. Такий підхід забезпечує більш плавну взаємодію з інтерфейсом та зменшує відчутну затримку для користувача.

У клієнтській частині реалізовано розширені налаштування інтерфейсу. Користувач може змінювати тему оформлення, обираючи між світлою та темною темами, що особливо актуально для роботи в темний час доби. Також доступна зміна мови інтерфейсу між українською та англійською, що робить платформу доступною для ширшої аудиторії. Крім того, реалізовано можливість зміни валюти відображення цін.

Ціни на послуги зберігаються в базі даних у гривнях, але при зміні валюти на долар США або євро система автоматично конвертує ціни за актуальним курсом, що дозволяє клієнтам зручніше орієнтуватися у вартості послуг. Для клієнтів реалізовано бонусну систему. За кожний виконаний запис клієнт отримує десять відсотків бонусів від вартості послуги. Накопичені бонуси можна використовувати для отримання знижки на наступні записи, при

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

цьому максимальна знижка бонусами обмежена тридцятьма відсотками від вартості послуги.

Це стимулює клієнтів повертатися до салону та робити повторні записи. Крім того, клієнт може експортувати свою історію записів у форматі CSV, що дозволяє зберігати дані про відвідування на власному пристрої або використовувати їх для аналізу.

Для зручності користувачів на сторінці профілю розміщено блок з найближчими вихідними днями салону, а також розділ з часто задаваними питаннями, де клієнти можуть знайти відповіді на типові запитання щодо скасування записів, нарахування бонусів, зміни часу запису та доступних способів оплати. Це зменшує навантаження на адміністраторів та дозволяє клієнтам самостійно знаходити потрібну інформацію.

Слід зазначити, що кожна з описаних вище операцій, як на клієнтській, так і на серверній стороні, супроводжується обробкою помилок, перевітками та валідацією вхідних даних, зокрема перевіркою формату електронної пошти, довжини пароля, унікальності записів тощо.

Всі дані зберігаються в базі даних із дотриманням вимог надійності та безпеки. У ряді випадків доступ до певних операцій обмежено і надається лише окремим групам користувачів відповідно до їх ролей у системі, наприклад, майстер не має доступу до профілю клієнта, а клієнт не може підтверджувати власні бронювання.

Подібні обмеження реалізовані і на клієнтській частині, де окремі сторінки або функції доступні лише авторизованим користувачам з відповідними правами доступу. Якщо користувач без авторизації намагається отримати доступ до захищеної сторінки, система автоматично перенаправляє його на сторінку входу з подальшим поверненням на запитовану сторінку після успішної автентифікації. На рис. 2.7 представлено діаграму компонентів системи.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

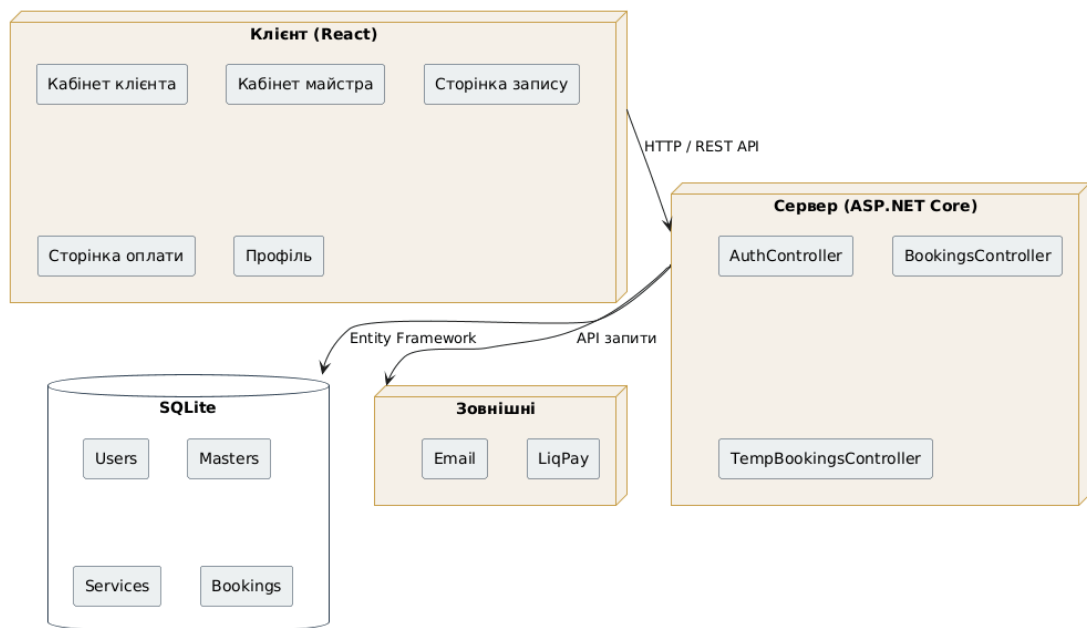


Рис. 2.7. Діаграма компонентів

Усі функціональні можливості були реалізовані з урахуванням сучасних технологій, стандартів безпеки, принципів масштабованості та кращих практик розробки вебзастосунків, що забезпечує надійну та зручну роботу системи в цілому.

Висновки до другого розділу

У другому розділі було проведено детальний аналіз функціональних і нефункціональних вимог до вебсистеми бронювання послуг салону краси. Визначено ролі користувачів, а саме клієнт та майстер, та їхні права доступу, що забезпечує належний рівень безпеки та персоналізації. Для візуалізації логіки роботи було використано діаграми активностей, які відображають ключові бізнес-процеси, зокрема створення та обробку бронювань. Діаграма класів дозволила проілюструвати структуру системи та взаємозв'язки між її основними модулями.

Було описано структуру реляційної бази даних SQLite з чітко визначеними таблицями, полями та зв'язками, зокрема таблицями Users, Masters, Services, Bookings, Schedules, Payments та Reviews. Розглянуто питання безпечної авторизації користувачів за допомогою JWT-токенів, що забезпечує захищений контроль сесій. Особливу увагу приділено оптимізації

взаємодії з даними: реалізовано автоматичне кешування даних на клієнтській стороні, періодичне оновлення кабінету майстра, перевірку доступності часових слотів з урахуванням графіку роботи та існуючих бронювань, а також автоматичне оновлення нагадувань при зміні статусу записів.

Таким чином, результати другого розділу формують стійке технічне та концептуальне підґрунтя для реалізації платформи, що відповідає актуальним вимогам до зручності, продуктивності, масштабованості та безпеки.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СЕРВІСОМ

3.1. Порядок встановлення та налаштування параметрів системи

Для забезпечення стабільної та коректної роботи вебсистеми бронювання послуг салону краси необхідно провести належне розгортання всіх її компонентів та налаштування взаємодії між ними. Оскільки серверна частина реалізована на платформі .NET Core, а клієнтська – на React.js, процес встановлення передбачає налаштування середовища для роботи обох частин застосунку. Для роботи системи необхідно мати встановлений .NET SDK для запуску бекенд-сервера та Node.js для збірки та запуску фронтенд-частини.

Серверна частина за замовчуванням працює на порту 5137, а клієнтська – на порту 3000. Для зберігання даних використовується файл бази даних SQLite beautybooking.db, який створюється автоматично при першому запуску застосунку завдяки механізму міграцій Entity Framework Core. Для запуску системи в режимі розробки достатньо виконати команду dotnet run у папці з проєктом бекенду та npm start у папці з проєктом фронтенду.

Після успішного запуску обох частин застосунку користувач може звертатися до вебсистеми через будь-який сучасний веббраузер (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) за адресою <http://localhost:3000>. Клієнтська частина, побудована на React.js, автоматично надсилає асинхронні HTTP-запити до серверної частини на порт 5137, використовуючи бібліотеку axios. У відповідь сервер, реалізований на ASP.NET Core Web API, повертає необхідні дані у форматі JSON, зокрема інформацію про послуги, майстрів, доступні часові слоти, створені бронювання, історію записів та бонусний баланс. Уся взаємодія між клієнтською та серверною частинами виконується через REST API з використанням JWT-токенів для автентифікації захищених маршрутів.

Токен зберігається у localStorage браузера та автоматично додається до заголовка Authorization кожного запиту, що дозволяє серверу ідентифікувати користувача та розмежовувати права доступу відповідно до його ролі (клієнт або майстер). Для забезпечення безпеки передачі даних між компонентами

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

системи налаштовано CORS-політику, яка дозволяє запити лише з довірених джерел (зокрема з порту 3000, де працює клієнтська частина). У разі потреби розгортання системи на продакшен-середовищі (хостингу або хмарній платформі, наприклад Microsoft Azure, AWS або DigitalOcean) необхідно додатково виконати низку важливих дій.

По-перше, слід налаштувати змінні середовища для підключення до бази даних, оскільки у виробничому середовищі замість локального файлу SQLite доцільно використовувати більш потужну СКБД, наприклад PostgreSQL або Microsoft SQL Server, особливо при великій кількості одночасних користувачів. По-друге, необхідно налаштувати проксування запитів за допомогою вебсервера (Nginx або IIS) таким чином, щоб клієнтська та серверна частини працювали на одному домені (наприклад, api.сайт.com для API та сайт.com для фронтенду), що усуває проблеми з CORS та спрощує налаштування SSL-сертифікатів. По-третє, обов'язково слід забезпечити використання HTTPS-протоколу для захисту каналів передачі даних, оскільки через мережу передаються персональні дані користувачів (електронна пошта, номер телефону, історія записів), а також здійснюється взаємодія з платіжною системою LiqPay під час онлайн-оплати послуг.

Належне налаштування SSL-сертифіката (наприклад, безкоштовного Let's Encrypt) запобігає перехопленню конфіденційної інформації зловмисниками. Крім того, для забезпечення безперебійної роботи системи в довгостроковій перспективі рекомендується налаштувати автоматичне створення резервних копій бази даних (наприклад, щоденне резервне копіювання файлу beautybooking.db або дам্পів SQL для віддаленої СКБД), а також налаштувати моніторинг доступності сервера за допомогою інструментів типу UptimeRobot або вбудованих засобів хмарних платформ.

Після завершення налаштування продакшен-середовища доцільно виконати навантажувальне тестування для перевірки стабільності роботи системи під піковими навантаженнями, а також перевірити коректність роботи всіх функціональних модулів (реєстрація, авторизація, створення бронювань,

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

оплата, сповіщення, бонусна система) у реальних умовах експлуатації. Для візуалізації архітектури та взаємодії між основними елементами створено діаграму розгортання, представлену на рис. 3.1.

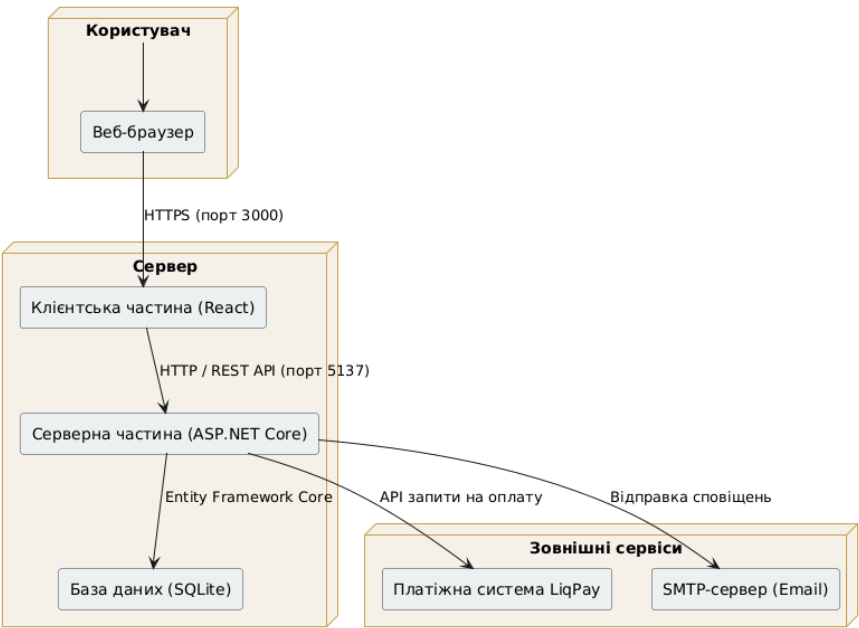


Рис. 3.1. Діаграма розгортання

Користувачі взаємодіють із платформою за допомогою веббраузера, що є основним інструментом доступу до клієнтського інтерфейсу. Цей інтерфейс реалізовано за допомогою React-застосунку, який виконується на стороні клієнта, забезпечуючи відображення сторінок та обробку дій користувача. Через нього здійснюються HTTP-запити до API серверної частини для отримання та відправлення даних.

Запити, що надходять від клієнтської частини, передаються до бекенд-сервера, на якому розгорнутий застосунок на основі ASP.NET Core Web API. Цей застосунок відповідає за всю бізнес-логіку системи, обробку HTTP-запитів, авторизацію користувачів за допомогою JWT-токенів, валідацію вхідних даних, а також взаємодію з базою даних та зовнішніми сервісами. Доступ до окремих ендпоінтів API обмежено відповідно до ролей користувачів: клієнти не мають доступу до функціоналу майстрів, а майстри не мають доступу до функціоналу клієнтів.

Для зберігання структурованих даних використовується легковагова реляційна база даних SQLite, яка зберігається у вигляді одного файлу

beautybooking.db. Бекенд-застосунок звертається до цієї бази через Entity Framework Core, виконуючи необхідні запити для обробки даних про користувачів, майстрів, послуги, бронювання, платежі та відгуки. Завдяки використанню ORM, більшість запитів генеруються автоматично, що спрощує розробку та зменшує ймовірність помилок.

Для зберігання зображень, зокрема аватарів майстрів, які додають адміністратори, застосовується хмарне сховище, однак у поточній версії системи цей функціонал не є критичним, оскільки зображення можуть зберігатися локально. Платформа також інтегрується з платіжною системою LiqPay для обробки онлайн-оплат [12]. Також із зовнішнім поштовим сервісом SMTP для надсилання системних електронних листів, зокрема сповіщень про зміну статусу бронювань [13].

Уся система організована як сукупність взаємопов'язаних компонентів: клієнтської частини, серверної частини, бази даних та зовнішніх сервісів [13]. Такий підхід забезпечує модульність, легкість у масштабуванні та підтримці, а також дозволяє незалежно оновлювати окремі компоненти без ризику порушення роботи всієї платформи. Використання контейнеризації через Docker спрощує розгортання системи в різних середовищах, забезпечуючи ізоляцію кожного компонента та зручне управління залежностями [25].

3.2. Структура інтерфейсу вебзастосунку

Інтерфейс є важливою складовою будь-якого проєкту, оскільки він визначає зручність користування та взаємодію користувачів із платформою. У цьому розділі представлено структуру інтерфейсу розробленого сервісу, яка базується на сучасних принципах дизайну та UI/UX практиках. Візуальне оформлення виконано в теплих бежевих та золотистих тонах, що створює атмосферу затишку та відповідає тематиці салону краси. Інтерфейс є повністю адаптивним, що забезпечує коректне відображення як на настільних комп'ютерах, так і на мобільних пристроях та планшетах.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Після запуску вебзастосунку та переходу на головну сторінку користувач потрапляє на сайт, де йому пропонується ознайомитися з переліком доступних послуг. Головна сторінка містить кілька ключових блоків. У верхній частині сторінки розташована шапка сайту, яка містить логотип салону, навігаційне меню з посиланнями на сторінки "Послуги", "Майстри", "Про нас", "Контакти" та кнопку "Запис".

Для неавторизованих користувачів у шапці також відображаються кнопки "Вхід" та "Реєстрація". Після авторизації ці кнопки замінюються на аватар користувача з випадаючим меню, через яке можна перейти до кабінету, профілю або вийти з системи. На рис. 3.2 зображено головну сторінку вебсистеми бронювання послуг салону краси.

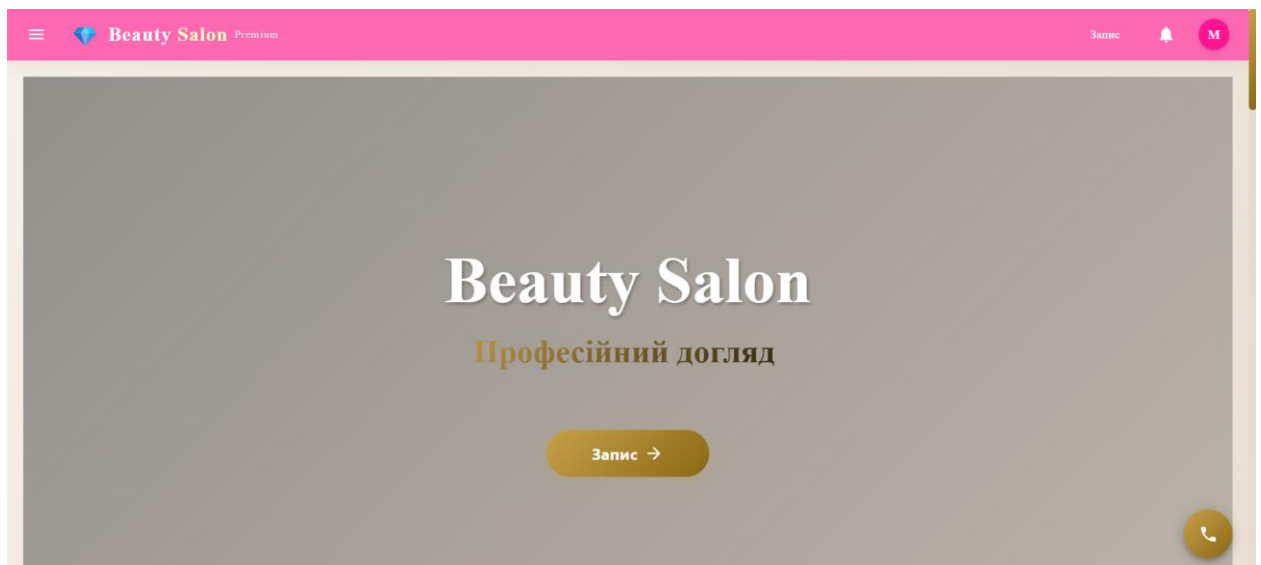


Рис. 3.2. Головна сторінка вебсистеми бронювання послуг салону краси

Основна частина головної сторінки містить каталог послуг, який представлений у вигляді сітки карток. Кожна картка містить назву послуги, її тривалість, короткий опис, ціну та кнопку "Записатися". Каталог послуг динамічно завантажується з сервера через API, що дозволяє легко оновлювати перелік послуг без необхідності змінювати код клієнтської частини. Нижче каталогу послуг розташований блок з акційною пропозицією, зображений на рис. 3.3 де відображається знижка на перший візит та зворотний відлік часу до закінчення акції.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

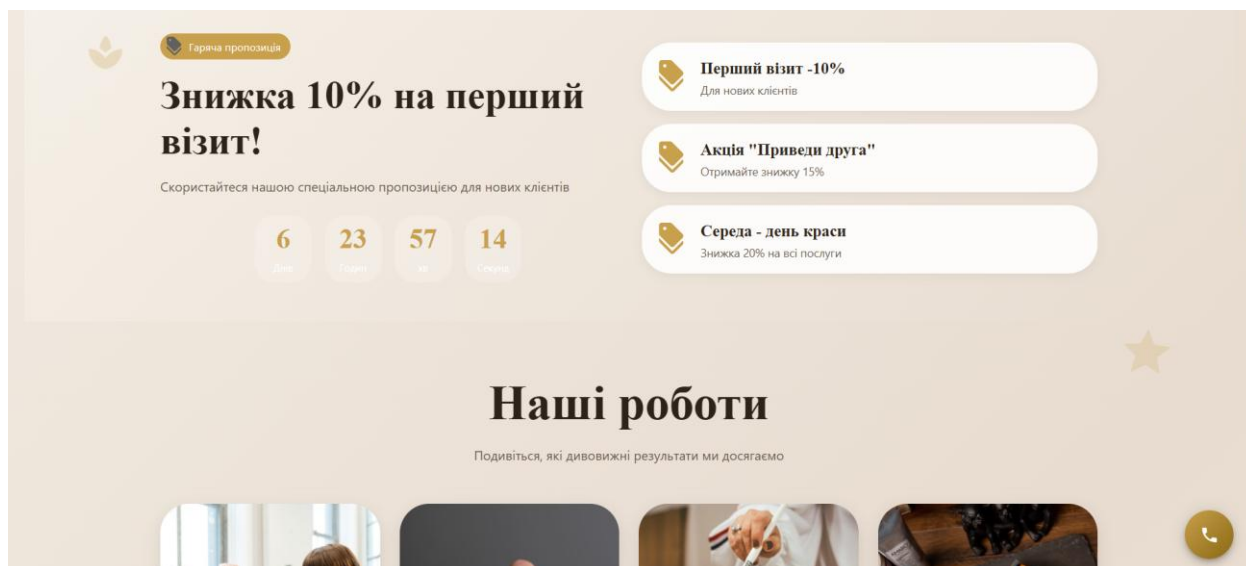


Рис. 3.3. Блок з акційною пропозицією

На сторінці послуг, яку зображено на рис. 3.4, представлено детальний каталог усіх доступних послуг салону краси. На відміну від головної сторінки, де послуги представлені стисло, тут кожна картка містить розширену інформацію: назву послуги, тривалість, детальний опис, ціну та кнопку для переходу до запису. Каталог динамічно завантажується з сервера та може бути відфільтрований за категоріями, що дозволяє клієнту швидко знайти потрібну послугу. Картки послуг мають ефект наведення – при наведенні курсора картка трохи підіймається та з'являється тінь, що створює відчуття інтерактивності. Ціни на послуги відображаються в обраній користувачем валюті, що робить сторінку зручною для міжнародних клієнтів.

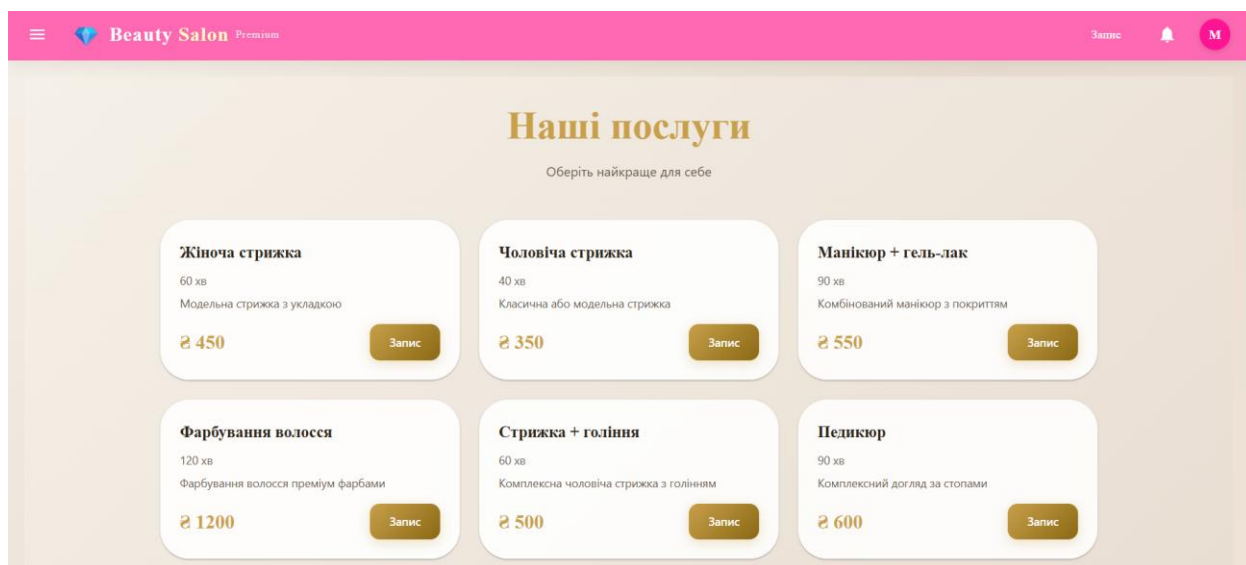


Рис. 3.4. Сторінка послуг

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка майстрів, яку зображено на рис. 3.5, містить профілі всіх спеціалістів салону краси. Кожен майстер представлений у вигляді картки, яка містить аватар, повне ім'я, посаду, спеціалізацію, досвід роботи, рейтинг та кількість відгуків. Користувач може відфільтрувати майстрів за спеціалізацією або знайти конкретного майстра за допомогою пошуку за ім'ям. На картці майстра також присутня іконка у вигляді серця, яка дозволяє додати майстра до списку улюблених. Після додавання іконка змінює колір, а майстер з'являється у відповідному розділі на сторінці профілю. Натискання на картку майстра відкриває модальне вікно з детальною інформацією, включно з описом професійних навичок, сертифікатами та портфоліо робіт, що допомагає клієнту зробити усвідомлений вибір.

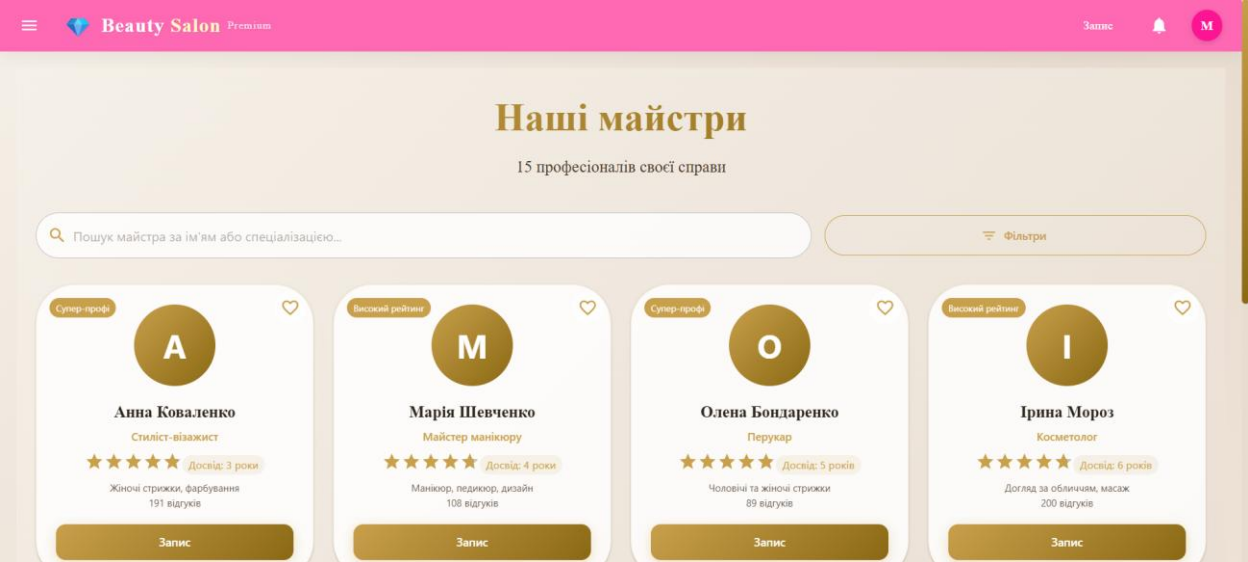


Рис. 3.5. Сторінка майстрів

Сторінка "Про нас", яку зображено на рис. 3.6, містить інформацію про салон краси, його історію, цінності та переваги. Тут розповідається про те, коли було засновано салон, які цілі ставить перед собою команда та чому клієнти обирають саме цей заклад. Окремий блок присвячено команді майстрів, де перераховані всі спеціалісти салону з коротким описом їхніх досягнень. Також на сторінці представлено переваги роботи з салоном, зокрема використання професійної косметики, сертифіковані майстри, гарантія якості, зручний графік роботи та комфортна атмосфера. Ця сторінка

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

формує довіру до закладу та допомагає новим клієнтам скласти позитивне враження ще до першого візиту.

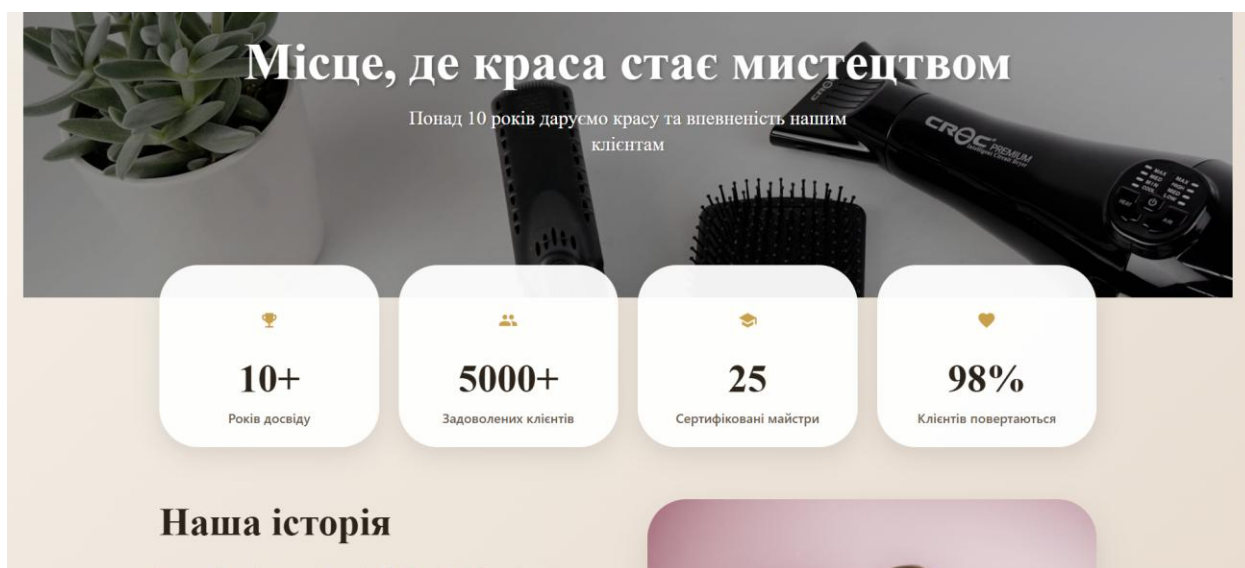


Рис. 3.6. Сторінка «Про нас»

На сторінці контактів, яку зображено на рис. 3.7, розміщено всю необхідну інформацію для зв'язку з салоном краси. Тут вказано фізичну адресу салону з можливістю побудови маршруту через Google карти, номери телефонів для дзвінків та месенджерів, електронну пошту для офіційних звернень та графік роботи з розбивкою на будні та вихідні дні. Також на сторінці розміщено форму зворотного зв'язку, через яку клієнт може надіслати повідомлення адміністрації салону. Після відправки форми повідомлення надходить на електронну пошту салону, а клієнт отримує підтвердження про успішну відправку [24]. Додатково на сторінці представлені посилання на соціальні мережі, що дозволяє клієнтам стежити за новинами та акціями салону. Для зручності користувачів контактна інформація продубльована у нижньому колонтитулі кожної сторінки сайту, що дозволяє швидко знайти необхідні дані незалежно від місцезнаходження на платформі. Крім того, усі номери телефонів виконані у вигляді інтерактивних посилань, що дозволяє клієнтам здійснити дзвінок одним натисканням на мобільних пристроях без необхідності копіювати номер вручну.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

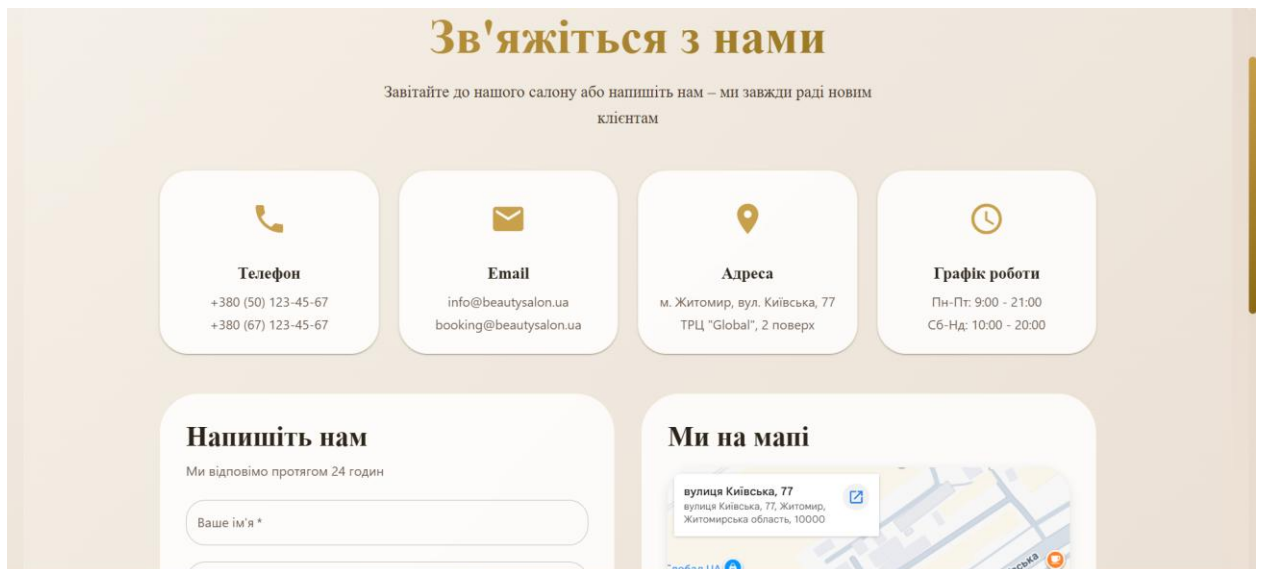


Рис. 3.7. Сторінка контактів

Головна сторінка містить шапку сайту з логотипом, навігаційним меню та кнопками входу та реєстрації, основну частину з каталогом послуг та підвал з контактною інформацією. Каталог послуг відображається у вигляді карток, кожна з яких містить назву послуги, ціну, тривалість та кнопку для переходу до сторінки запису. На рис. 3.8 зображено сторінку реєстрації.

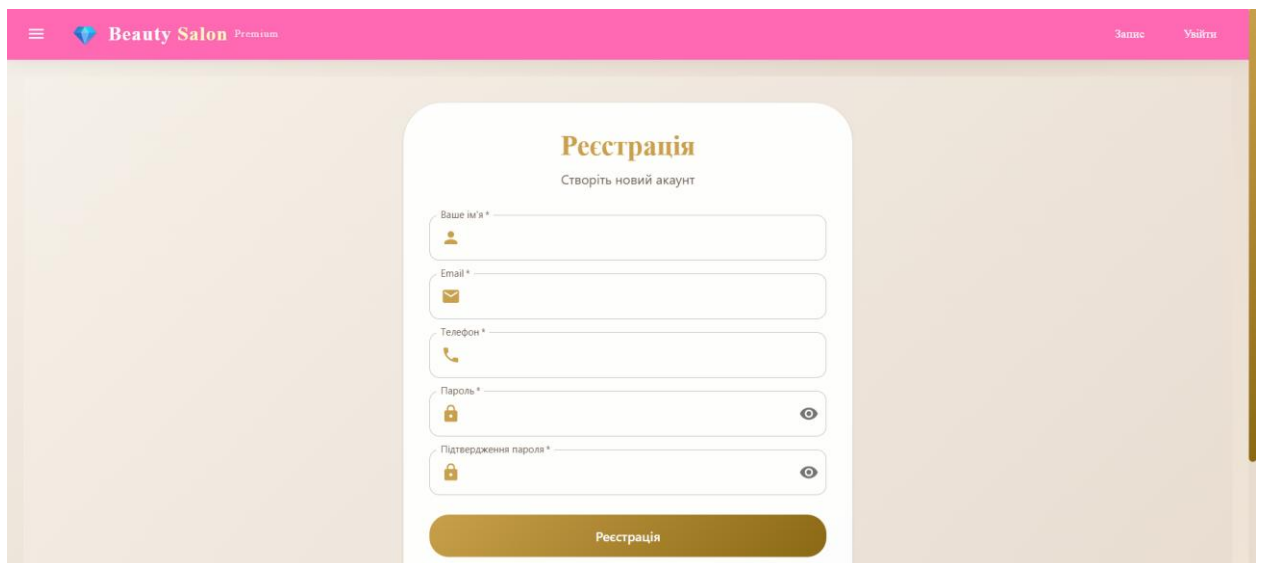


Рис. 3.8. Сторінка реєстрації

На сторінці входу користувач вводить електронну пошту та пароль. Після відправки форми система перевіряє відповідність введених даних записам у базі даних. Залежно від ролі користувача – Client або Master – система перенаправляє його до відповідного кабінету [16]. Після успішної авторизації в шапці сайту з'являється аватар користувача та випадаюче меню

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

з посиланнями на кабінет, профіль та вихід з системи. Після входу клієнта відкривається його кабінет, який на рис. 3.9.

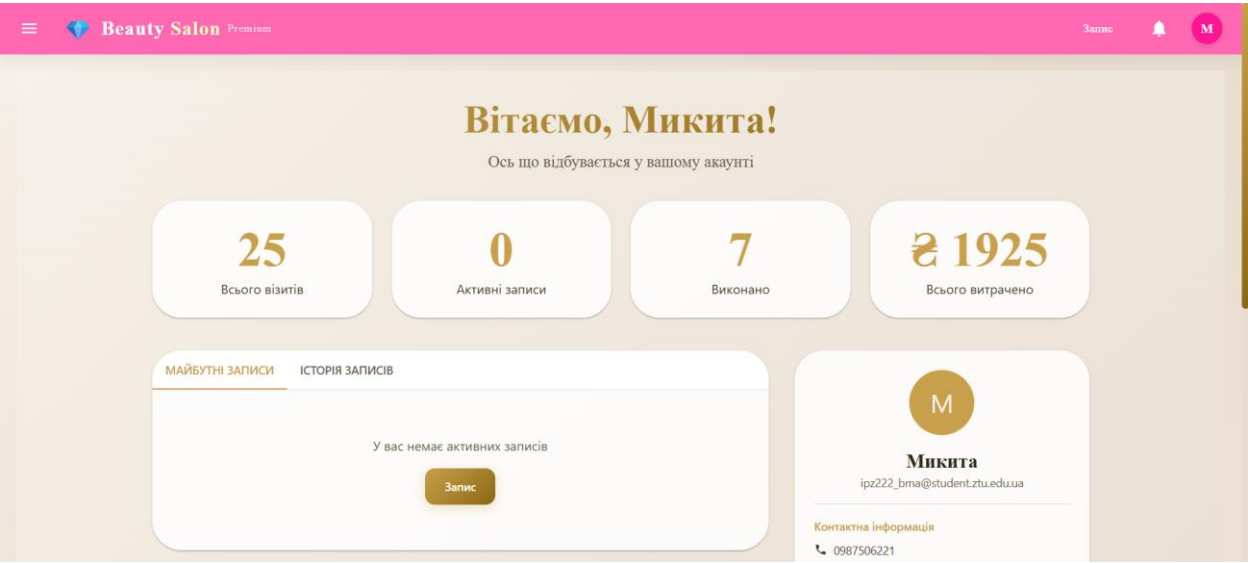


Рис. 3.9. Кабінет користувача

Кабінет клієнта складається з двох основних вкладок: майбутні записи та історія записів. У вкладці майбутніх записів відображаються всі бронювання, які мають статус Pending або Confirmed. Для кожного запису клієнт може переглянути назву послуги, ім'я майстра, дату, час, суму та статус.

Клієнт може скасувати бронювання, яке ще не виконано, натиснувши на кнопку видалення. У вкладці історії відображаються виконані та скасовані записи. При скасуванні бронювання система відображає діалогове вікно з підтвердженням дії, що запобігає випадковому видаленню запису. Крім того, якщо на скасоване бронювання були нараховані бонусні бали, вони автоматично списуються з балансу користувача. Скасувати запис можна лише за певний час до початку процедури – система не дозволяє скасовувати бронювання менш ніж за дві години до призначеного часу, що дає змогу майстру ефективніше планувати свій робочий графік. Праворуч на сторінці розташована інформація про профіль користувача, його контактні дані, кількість бонусних балів та блок з активними нагадуваннями про майбутні запити. На рис. 3.10 зображено кабінет майстра.

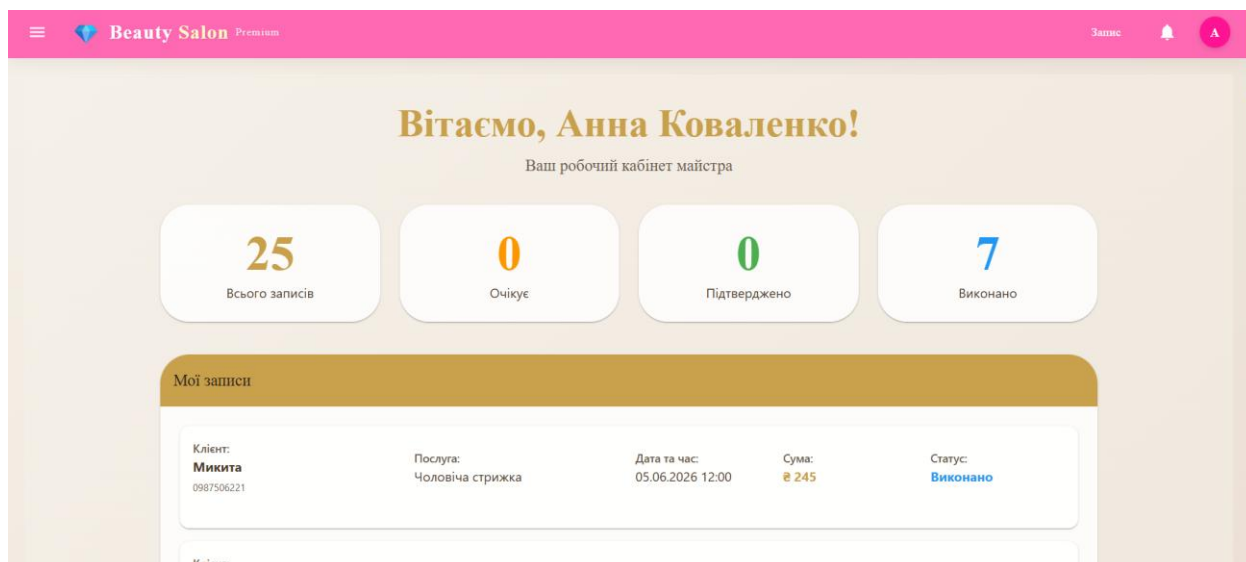


Рис. 3.10. Кабінет майстра

Кабінет майстра призначений для працівників салону краси. Тут відображаються всі бронювання, призначені на поточного майстра. Для кожного запису майстер бачить ім'я клієнта, обрану послугу, дату, час, суму та статус. Якщо статус бронювання є Pending, майстер може підтвердити його, натиснувши відповідну кнопку. Якщо статус Confirmed, майстер може відзначити запис як виконаний. Майстер також може скасувати бронювання. Сторінка оновлюється автоматично кожні десять секунд, що дозволяє майстру бачити нові запити без необхідності перезавантажувати сторінку.

На рис. 3.11 зображено сторінку створення бронювання.

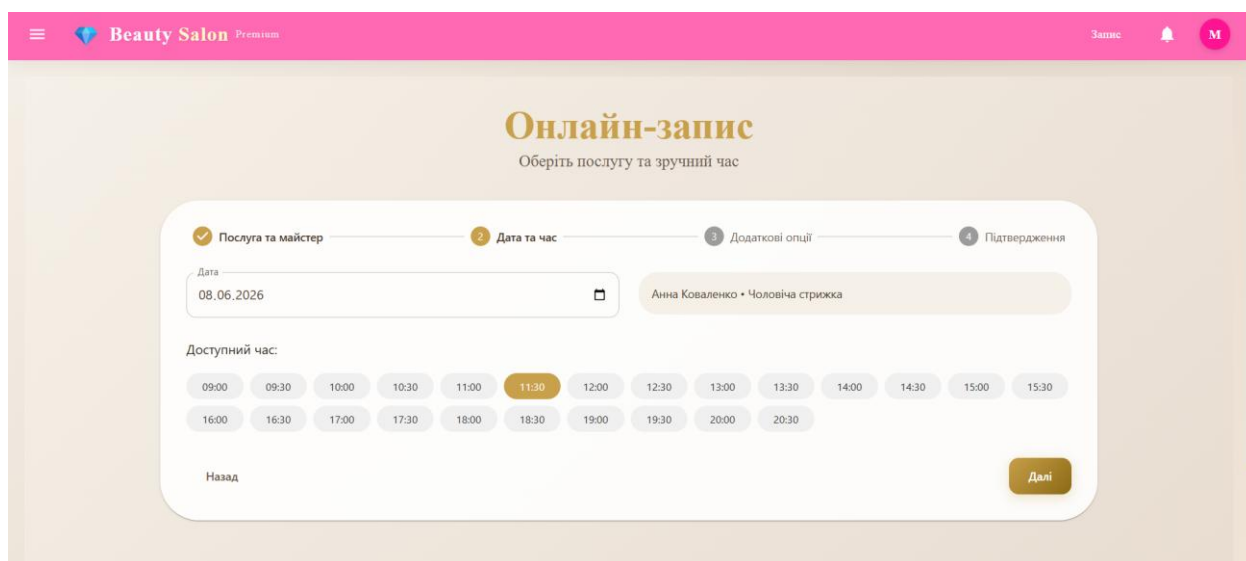


Рис. 3.11. Сторінку створення бронювання

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Створення бронювання відбувається в кілька кроків. На першому кроці клієнт обирає послугу з каталогу, після чого система відображає список майстрів, які надають обрану послугу. Клієнт обирає майстра та переходить до наступного кроку. На другому кроці клієнт обирає зручну дату для запису. Система автоматично перевіряє доступність часу, враховуючи графік роботи майстра та вже існуючі бронювання. На третьому кроці, якщо обрана послуга має додаткові опції, система відображає відповідні форми для їх вибору. На четвертому кроці клієнт підтверджує бронювання та переходить до сторінки оплати. На рис. 3.12 зображено сторінку оплати.

Рис. 3.12. Сторінку оплати

На сторінці оплати, клієнт може обрати спосіб оплати: онлайн карткою через платіжну систему LiqPay або оплату готівкою або карткою безпосередньо в салоні після надання послуги. Ліворуч на сторінці відображаються деталі створюваного бронювання, зокрема назва обраної послуги, ім'я майстра, дата та час запису, а також вартість послуги. Якщо клієнт має бонусні бали, у цьому блоці також відображається інформація про доступну кількість балів та максимально можливу знижку, яка обмежена тридцятьма відсотками від вартості послуги.

Для використання бонусів клієнт повинен активувати відповідний перемикач, після чого стає доступним поле для введення кількості балів, яку він бажає використати. Система автоматично обмежує максимальну кількість

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

балів, які можна застосувати, таким чином, щоб знижка не перевищувала тридцять відсотків від вартості послуги. Наприклад, якщо вартість послуги становить 1000 гривень, максимальна знижка бонусами не може перевищувати 300 гривень, що відповідає 300 бонусним балам, оскільки один бонус дорівнює одній гривні знижки. Після вибору кількості балів система перераховує остаточну суму до сплати та відображає розмір отриманої знижки.

Після натискання кнопки підтвердження система створює бронювання зі статусом Pending, що означає очікування підтвердження з боку майстра. Якщо обрано онлайн-оплату, відбувається з'єднання з платіжною системою LiqPay. Клієнт перенаправляється на захищену сторінку платіжної системи, де підтверджує платіж. Після успішної оплати платіжна система повертає клієнта на сайт, а бекенд отримує сповіщення про успішну транзакцію. У цьому випадку статус бронювання оновлюється, залишаючись у статусі Pending до моменту підтвердження майстром.

Після успішного створення бронювання (незалежно від способу оплати) система нараховує клієнту бонусні бали в розмірі десяти відсотків від вартості послуги. Нарахування відбувається за такою логікою: базова вартість послуги множиться на 0,1, а отримане значення округлюється до цілого числа. Наприклад, якщо вартість послуги становить 350 гривень, клієнт отримує 35 бонусних балів. Якщо вартість послуги становить 550 гривень, клієнт отримує 55 бонусних балів. Якщо клієнт використовував бонуси для отримання знижки, нарахування відбувається від оригінальної вартості послуги, а не від суми після знижки. Це стимулює клієнтів повертатися до салону, оскільки кожен наступний візит стає дешевшим за рахунок накопичених бонусів.

Накопичені бонуси зберігаються в таблиці Users у полі BonusPoints. Клієнт може переглянути поточну кількість бонусів у своєму кабінеті або на сторінці профілю. При створенні нового бронювання система автоматично пропонує використати доступні бонуси, якщо вони є. Бонуси не можна використати частково – клієнт самостійно обирає кількість балів для

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

застосування, але в межах доступного ліміту та з урахуванням максимальної знижки. Бонуси не підлягають виведенню готівкою та діють виключно в межах платформи для оплати послуг салону. Детальну інформацію можна переглянути у профілі користувача рис. 3.13.

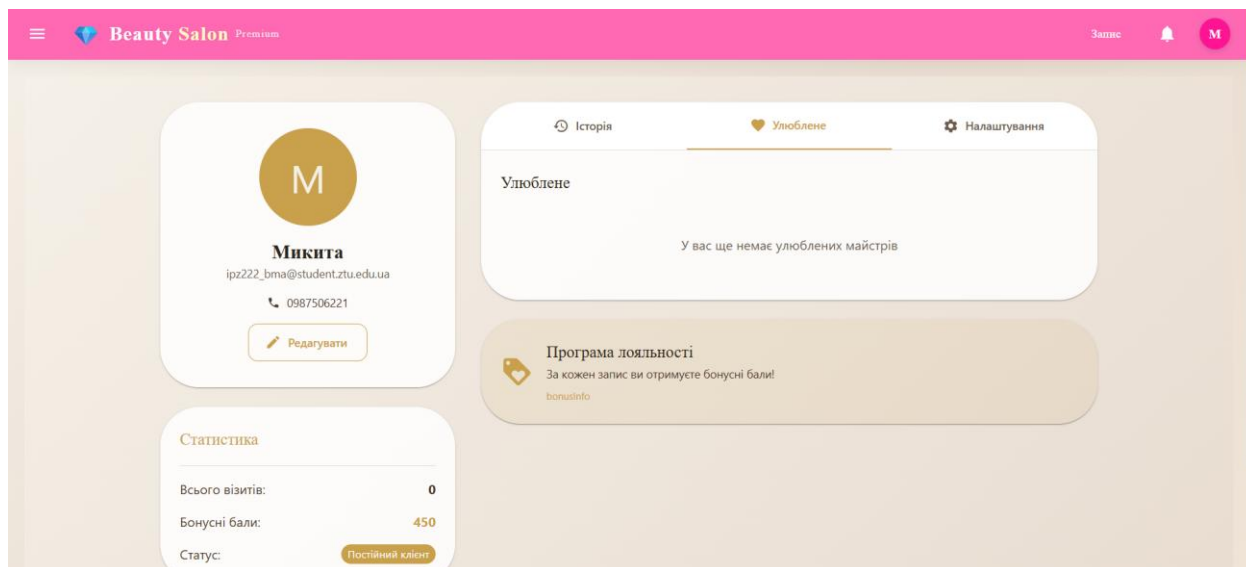


Рис. 3.13. Профіль користувача

Сторінка профілю користувача є центральним місцем для керування обліковим записом клієнта. Вона складається з кількох основних розділів: інформація про користувача, статистика, історія записів, улюблені майстри та налаштування. Ліворуч на сторінці розташований блок із аватаром користувача, його повним іменем, електронною поштою та номером телефону. Поруч з аватаром передбачено можливість його зміни – після натискання на іконку камери користувач може завантажити нове зображення, яке після обробки зберігається на сервері.

Тут же розміщена кнопка "Редагувати", після натискання на яку користувач може змінити своє ім'я та номер телефону, рис. 3.14. У розділі статистики користувач може переглянути загальну кількість відвідувань, сумарну вартість виконаних записів та загальну кількість накопичених бонусних балів за весь час користування платформою. Після внесення змін та підтвердження система оновлює відповідні поля в таблиці Users та зберігає актуальні дані в localStorage для подальшого використання.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

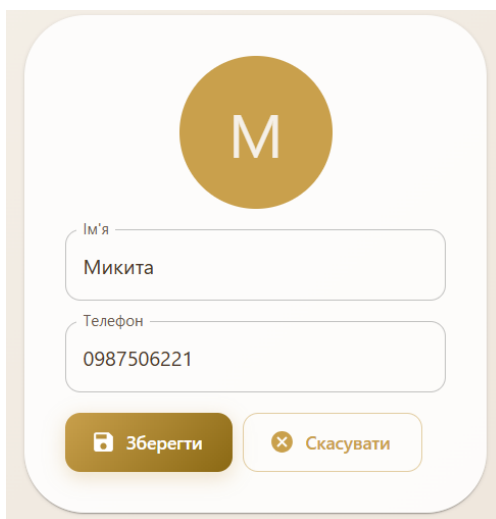


Рис. 3.14. Редагування інформації користувача

На вкладці налаштувань, яку зображено на рис. 3.15, користувач може налаштувати інтерфейс відповідно до своїх уподобань. Тут доступні три основні параметри: тема оформлення, мова інтерфейсу та валюта відображення цін.

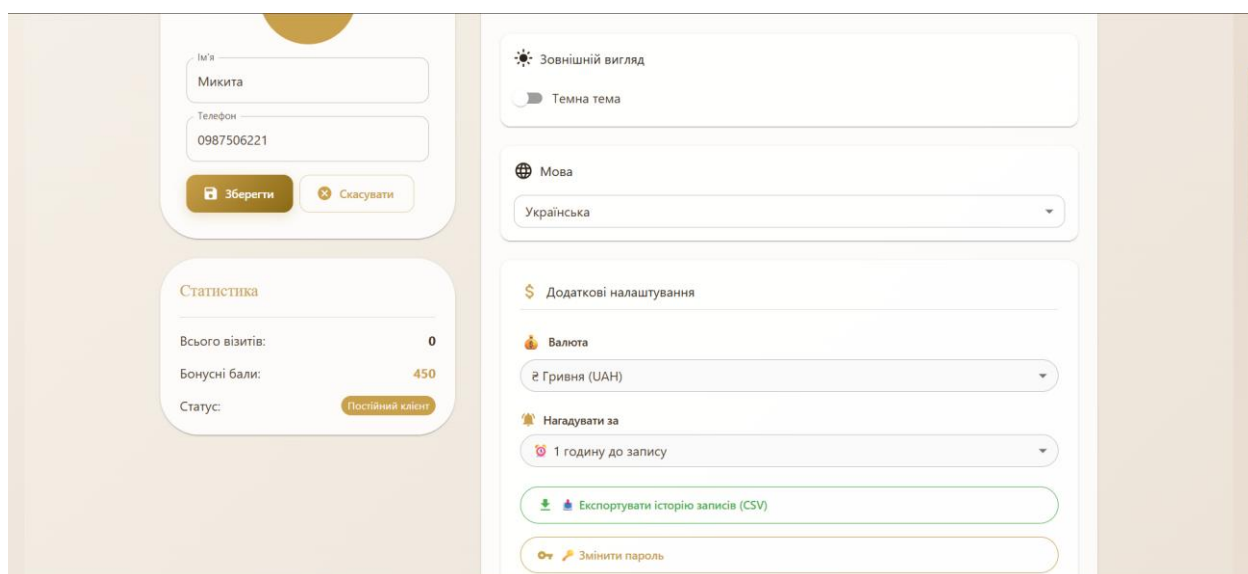


Рис. 3.15. Налаштування інтерфейсу

Першим параметром є тема оформлення. Користувач може обрати між світлою та темною темами. При виборі темної теми система змінює кольорову гаму всього інтерфейсу: фон стає темним, текст – світлим, а акцентні елементи залишаються золотистими для збереження візуальної ієрархії. Це особливо актуально для використання в темний час доби або для користувачів, які

віддають перевагу темному оформленню інтерфейсів. Налаштування теми застосовується до всіх сторінок сайту та зберігається між сесіями.

Другим параметром є мова інтерфейсу. Користувач може обрати українську або англійську мову. При зміні мови система динамічно перекладає всі текстові елементи інтерфейсу, включаючи назви послуг, кнопки, повідомлення та інші тексти. Переклади реалізовано через контекст React, що дозволяє централізовано керувати всіма текстовими рядками застосунку. Вибір мови зберігається в налаштуваннях користувача та автоматично застосовується при наступних візитах.

Третім параметром є валюта відображення цін. Користувач може обрати одну з трьох валют: гривня, долар США або євро. Всі ціни на послуги зберігаються в базі даних у гривнях, а при зміні валюти система автоматично конвертує їх з використанням заздалегідь визначених курсів. На момент розробки системи встановлено такі курси: один долар США дорівнює сорока одній гривні, а один євро дорівнює сорока чотирьом гривням. При зміні валюти всі ціни на сторінках послуг, у каталозі, в історії записів та на сторінці оплати перераховуються миттєво. Символ валюти також змінюється відповідно до обраної валюти: для гривні використовується знак ₴, для долара – знак \$, для євро – знак €.

У розділі улюблених майстрів, який зображено на рис. 3.16, відображаються майстри, яких користувач додав до списку обраного. Додати майстра до улюблених можна на сторінці майстрів, натиснувши на відповідну іконку у вигляді серця поруч із профілем майстра. При натисканні іконка змінює колір, а ідентифікатор майстра додається до масиву FavoriteMasters у записі користувача в таблиці Users. Цей масив зберігається у форматі JSON, що дозволяє легко додавати та видаляти елементи.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

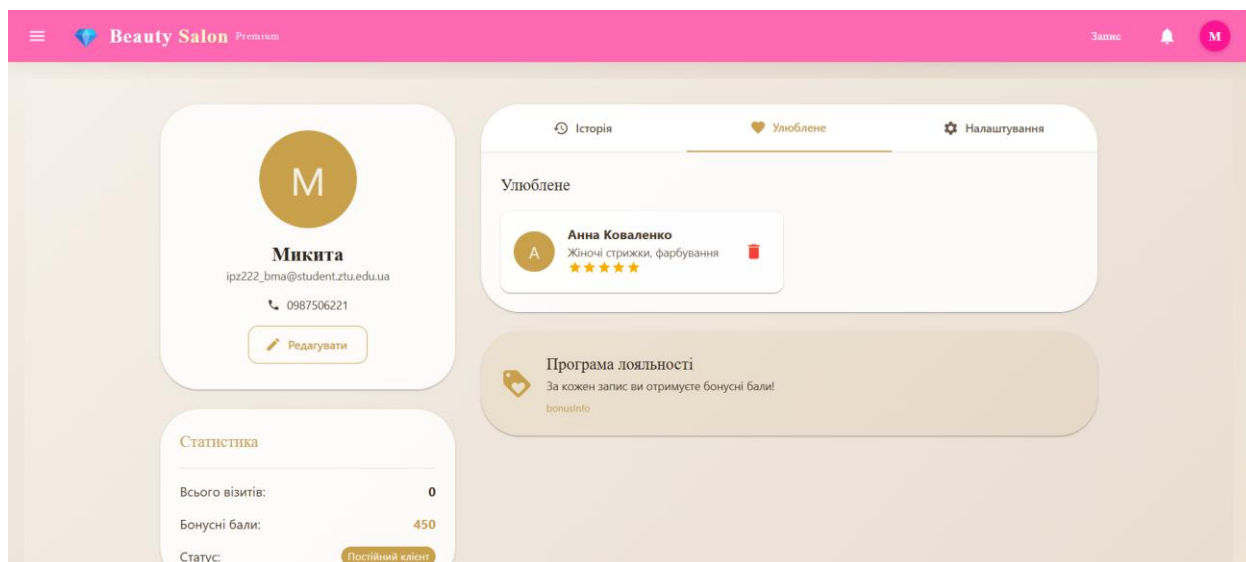


Рис. 3.16. Розділ налаштувань «Улюблене»

У списку улюблених майстрів кожен запис містить аватар майстра, його повне ім'я, спеціалізацію, рейтинг та кнопку для видалення з обраного. Натискання на картку майстра перенаправляє користувача на сторінку запису до цього майстра, що дозволяє швидко створити бронювання без необхідності повторного пошуку. Якщо користувач видаляє майстра з обраного, ідентифікатор майстра видаляється з масиву FavoriteMasters, а картка зникає зі списку. Всі зміни в списку улюблених майстрів синхронізуються між сторінкою профілю та сторінкою майстрів за допомогою події storage, що дозволяє оновлювати інтерфейс без перезавантаження сторінки.

На сторінці профілю також розміщено блоки з корисною інформацією, які зображено на рис. 3.17. Перший блок містить перелік найближчих вихідних днів салону, зокрема державні свята, такі як День Конституції України 28 червня, День Незалежності України 24 серпня та Новорічні свята 1-7 січня. Ця інформація допомагає клієнтам планувати візити з урахуванням графіка роботи салону.

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

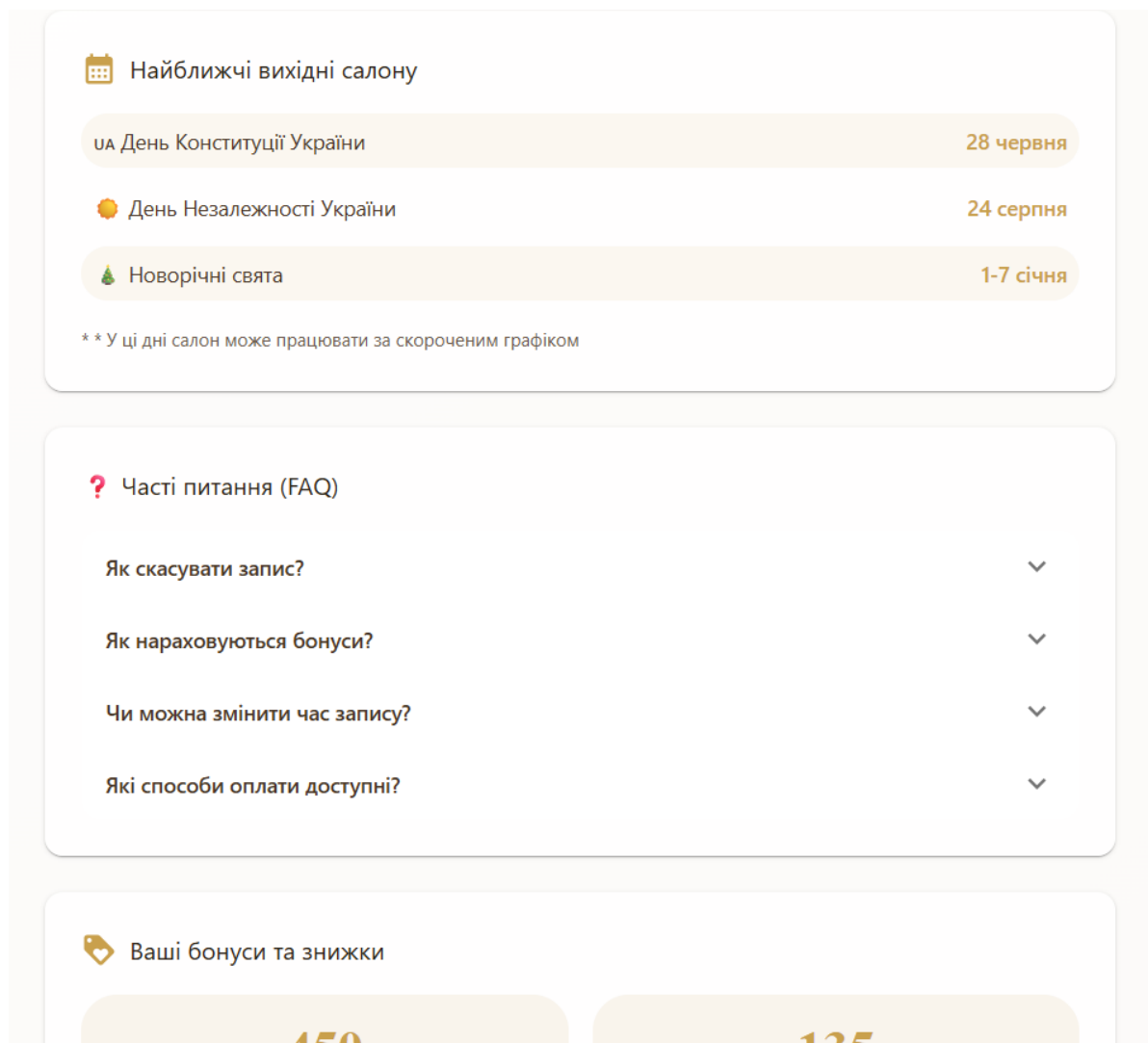


Рис. 3.17. Блок з корисною інформацією

3.3. Тестування роботи програмного забезпечення

Розглянемо процес тестування ключових аспектів розробленої платформи, який охоплює валідацію даних під час виконання різних операцій, авторизацію користувачів для здійснення певних запитів, а також перевірку коректності роботи основного функціоналу системи. Тестування проводилося як на стороні клієнта, так і на стороні сервера, з метою забезпечення стабільності, безпеки та коректності роботи всіх компонентів вебсистеми.

Валідація даних є одним із критично важливих етапів тестування, оскільки від неї залежить цілісність даних у системі та безпека платформи. У розробленій системі валідація реалізована на двох рівнях: на стороні клієнта (миттєва перевірка при введенні даних) та на стороні сервера (обов'язкова перевірка перед збереженням у базі даних). Далі розглянемо типові помилки,

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

яких може припуститися користувач під час взаємодії з платформою, та реакцію системи на такі дії.

У разі реєстрації нового користувача система виконує перевірку унікальності електронної пошти. Якщо обліковий запис із вказаною електронною адресою вже існує, система виводить повідомлення "Користувач з таким email вже існує" під формою реєстрації, що зображено на рис. 3.18. Аналогічно, під час авторизації, якщо введені дані не відповідають жодному користувачеві, система повідомляє про невірний email або пароль. При цьому поля форми не очищаються повністю – лише підсвічуються червоним кольором, що дозволяє користувачеві швидко виправити помилку. Крім того, на стороні клієнта реалізовано перевірку формату електронної пошти: якщо користувач вводить адресу без символу "@" та домену, система відображає попередження "Введіть коректну електронну пошту" ще до відправлення запиту на сервер.

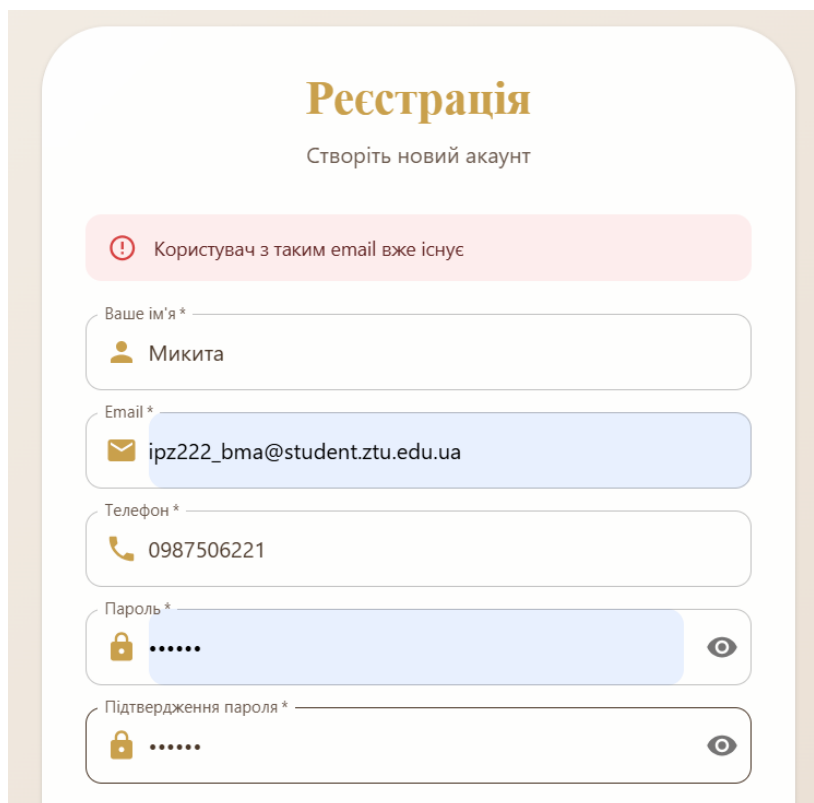


Рис. 3.18. Помилка при реєстрації в разі існуючого email

Перевірка пароля під час реєстрації включає кілька критеріїв: довжина пароля має становити не менше шести символів, а також обов'язкове

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

співпадіння пароля та його підтвердження. Якщо довжина пароля менше шести символів, під полем введення пароля з'являється повідомлення "Пароль повинен містити не менше 6 символів". Якщо пароль та його підтвердження не співпадають, система відображає повідомлення "Паролі не співпадають". Ці повідомлення з'являються в реальному часі під час введення тексту, що дозволяє користувачеві одразу бачити та виправляти помилки без додаткових запитів до сервера.

При створенні бронювання система виконує низку перевірок для забезпечення коректності запису. На рис. 3.19 зображено ситуацію, коли користувач намагається записатися на час, який вже зайнятий іншим клієнтом. У цьому випадку система блокує можливість вибору зайнятого часу – відповідний чип стає сірим, неактивним та не реагує на натискання. Над сіткою часу відображається попередження "Часи, позначені сірим, вже зайняті". Якщо користувач все ж намагається обрати зайнятий час через інструменти розробника, сервер при відправці форми повертає помилку "Цей час вже зайнятий. Оберіть інший час."

Рис. 3.19. Зайнятий час при створенні бронювання

При виборі сьогоднішньої дати система автоматично приховує часові слоти, які вже минули. Наприклад, якщо поточна година 14:30, користувач не побачить у списку доступних часів 14:00 та попередні години. Це запобігає спробам запису на час, який неможливо використати. При виборі минулої дати календар блокує таку можливість – дні, що передують поточному, стають неактивними та виділяються сірим кольором.

Якщо користувач спробує перейти за посиланням, доступ до якого мають лише авторизовані користувачі, система перенаправляє його на сторінку входу. Наприклад, при спробі переходу на сторінку /dashboard без наявності JWT-токену в localStorage, спрацьовує захисний маршрут (PrivateRoute), який перевіряє наявність токену та, за його відсутності, виконує перенаправлення на /login. Після успішної авторизації користувач повертається на сторінку, яку намагався відкрити.

Під час тестування бонусної системи перевіряється коректність нарахування балів після виконання запису. При створенні бронювання на суму 350 гривень клієнт отримує 35 бонусних балів, що відповідає десяти відсоткам від вартості. При використанні бонусів система обмежує максимальну знижку тридцятьма відсотками. На рис. 3.20 зображено обмеження використання бонусних балів.

Оформлення запису

Деталі запису

Послуга: Чоловіча стрижка

Майстер: Анна Коваленко

Дата та час: 2026-06-07 о 09:30

Сума: ₾ 350

Ваші бонуси: 485 балів

☒ Використати бонуси для знижки

Доступно до 105 балів (макс. 30% від вартості)

105

Спосіб оплати

Оплата онлайн

☐ Безпечна оплата карткою Visa/Mastercard через LiqPay

Оплата в салоні

☒ Готівка або карта в день візиту

☒ onsitePaymentOption1

☒ onsitePaymentOption2

Вартість послуги: ₾ 350

Знижка за бонуси: - ₾ 105

До сплати: ₾ 245

Рис. 3.20. Обмеження використання бонусів

При скасуванні бронювання з нього списуються раніше нараховані бонуси в розмірі десяти відсотків від вартості. Якщо у клієнта недостатньо балів, баланс обнуляється. Система відображає спливаюче вікно з

підтвердженням скасування, в якому зазначається кількість балів, що буде списано. На рис. 3.21 зображено діалогове вікно підтвердження скасування бронювання з інформацією про списання бонусів.

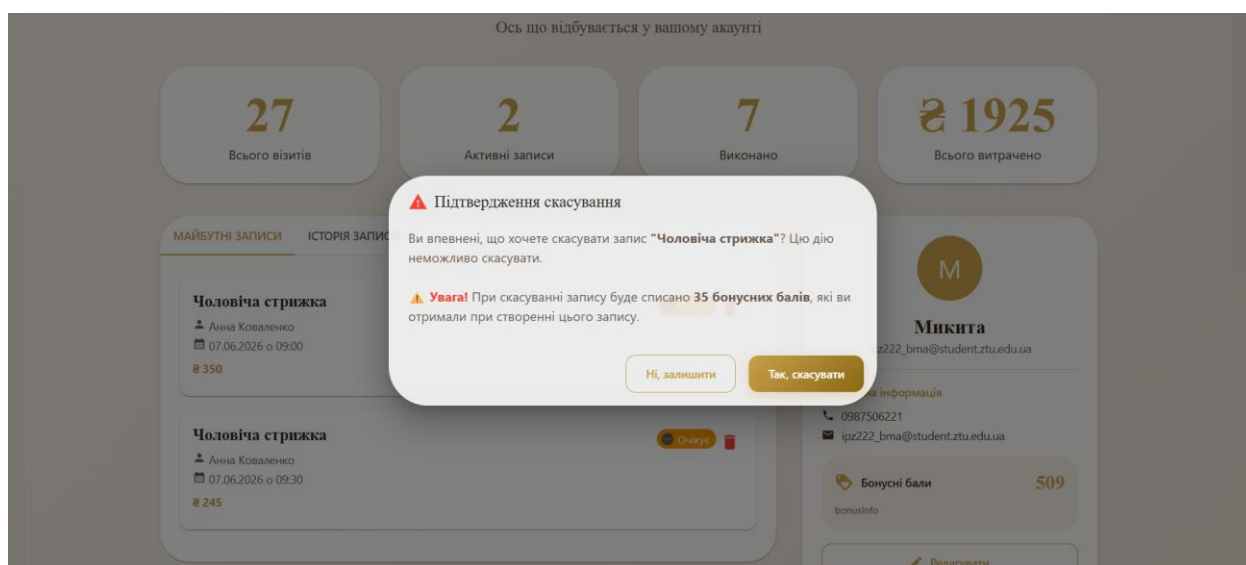


Рис. 3.21. Підтвердження скасування бронювання зі списанням бонусів

При зміні пароля система перевіряє, чи введений старий пароль відповідає поточному. У разі невідповідності відображається повідомлення "Невірний старий пароль". Якщо новий пароль занадто короткий, система повідомляє "Пароль повинен містити не менше 6 символів". Після успішної зміни пароля всі активні сесії користувача стають недійсними, що вимагає повторного входу в систему. Це забезпечує додатковий рівень безпеки у разі компрометації облікового запису.

Функція відновлення пароля перевіряє, чи існує користувач із вказаною електронною поштою. Якщо користувача не знайдено, система відображає повідомлення "Користувача з такою електронною поштою не знайдено". Якщо користувач знайдений, на вказану адресу надсилається лист з унікальним посиланням для скидання пароля. Посилання діє протягом однієї години, після чого стає недійсним, що запобігає несанкціонованому доступу.

Окрім клієнтської валідації, на стороні сервера реалізовано обов'язкову перевірку всіх вхідних даних. Досвідчений користувач може свідомо змінити дані форми перед відправкою через інструменти розробника або вручну підмінити параметри в URL-рядку. Наприклад, при спробі змінити

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

ідентифікатор бронювання в запиті на скасування, сервер перевіряє, чи належить це бронювання поточному користувачеві. Якщо ні, запит відхиляється з помилкою доступу. Аналогічно перевіряються всі параметри: наявність обов'язкових полів, відповідність типів даних, коректність значень (наприклад, дата не може бути в минулому, час повинен відповідати формату), а також існування пов'язаних сутностей у базі даних (чи існує майстер, чи існує послуга, чи не зайнятий час). У разі виявлення будь-якої невідповідності сервер повертає статус 400 Bad Request з детальним описом помилки, а клієнтська частина відображає відповідне повідомлення користувачеві.

Висновки до третього розділу

У третьому розділі було детально розглянуто порядок встановлення та налаштування вебсистеми бронювання послуг салону краси. Розгортання системи передбачає запуск серверної частини на платформі .NET Core з використанням ASP.NET Core Web API, який за замовчуванням працює на порту 5137, та клієнтської частини на React.js, яка виконується на порту 3000. Для роботи системи необхідно мати встановлений .NET SDK для запуску бекенд-сервера та Node.js для збірки та запуску фронтенд-частини. База даних SQLite зберігається у вигляді одного файлу beautybooking.db, який створюється автоматично при першому запуску завдяки механізму міграцій Entity Framework Core. Для запуску системи в режимі розробки достатньо виконати команду dotnet run у папці з проєктом бекенду та npm start у папці з проєктом фронтенду.

Окрему увагу приділено роботі з бронюваннями, включно з усіма аспектами їх створення, перегляду та скасування. При створенні бронювання клієнт обирає послугу з каталогу, майстра, зручну дату та час. Система автоматично перевіряє доступність обраного часу, враховуючи графік роботи майстра та вже існуючі бронювання, а також не відображає часові слоти, які вже минули, якщо обрано сьогоднішню дату. Клієнт може скасувати бронювання зі статусом Pending або Confirmed, але не може скасувати вже

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

виконаний запис. Майстер, у свою чергу, має доступ до власного розкладу, може підтверджувати бронювання, відзначати їх як виконані та скасовувати.

Розглянуто кабінет клієнта та кабінет майстра, їх функціональні можливості та особливості. Кабінет клієнта містить дві основні вкладки: майбутні записи та історія записів. У вкладці майбутніх записів відображаються всі бронювання зі статусом Pending або Confirmed, де клієнт може переглянути деталі запису або скасувати його. У вкладці історії відображаються виконані та скасовані записи. Кабінет майстра призначений для працівників салону краси, де відображаються всі бронювання, призначені на поточного майстра. Для кожного запису майстер бачить ім'я клієнта, обрану послугу, дату, час, суму та статус. Сторінка оновлюється автоматично кожні десять секунд, що дозволяє майстру бачити нові запити без необхідності перезавантажувати сторінку.

Крім того, було детально описано структуру інтерфейсу інших ключових сторінок. Сторінка послуг містить детальний каталог усіх доступних послуг із можливістю фільтрації за категоріями. Сторінка майстрів містить профілі всіх спеціалістів салону з можливістю пошуку, фільтрації за спеціалізацією та додавання до списку улюблених. Сторінка "Про нас" містить інформацію про салон, його історію, цінності та переваги. Сторінка контактів містить адресу, номери телефонів, електронну пошту, графік роботи та форму зворотного зв'язку.

Було проведено комплексне тестування основних компонентів системи, яке підтвердило коректність валідації введених даних, ефективність контролю доступу користувачів до різних ресурсів і стабільність роботи API за різних сценаріїв використання. Тестування показало, що система коректно обробляє помилки при реєстрації, коли користувач з такою електронною поштою вже існує, та при авторизації, коли введено невірний email або пароль. Перевірено, що при виборі зайнятого часу для бронювання система блокує таку можливість та відображає відповідне попередження. Також підтверджено, що

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

при виборі сьогоднішньої дати система автоматично приховує часові слоти, які вже минули, а при виборі минулої дати календар блокує таку можливість.

Особливу увагу приділено тестуванню бонусної системи. Підтверджено, що при створенні бронювання клієнт отримує бонуси в розмірі десяти відсотків від вартості послуги. Перевірено, що при використанні бонусів система обмежує максимальну знижку тридцятьма відсотками від вартості послуги. Також підтверджено, що при скасуванні бронювання з нього списуються раніше нараховані бонуси, а у разі недостатньої кількості балів баланс обнуляється. Тестування зміни пароля підтвердило коректність перевірки старого пароля, вимог до нового пароля та анулювання всіх активних сесій після успішної зміни. Функція відновлення пароля перевіряє існування користувача з вказаною електронною поштою та надсилає лист з унікальним посиланням, яке діє протягом однієї години.

Усі результати тестування підтвердили, що розроблена вебсистема бронювання послуг салону краси є стабільною, безпечною та готовою до експлуатації в реальних умовах. Вона здатна надійно обслуговувати як клієнтів, які бажають записатися на послуги, так і майстрів, які керують своїм розкладом. Інтуїтивно зрозумілий інтерфейс, адаптивний дизайн та широкий функціонал роблять платформу зручною для всіх категорій користувачів.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно розроблено сучасну вебсистему для онлайн-бронювання послуг салону краси, яка відповідає актуальним вимогам як з точки зору функціональності, так і технічної реалізації. Розроблена платформа забезпечує зручне керування бронюваннями для клієнтів, ефективне управління розкладом для майстрів та автоматизацію процесу запису, що дозволяє оптимізувати роботу салону краси, зменшити навантаження на персонал та підвищити загальну ефективність бізнесу.

У першому розділі роботи було проведено ґрунтовний аналіз предметної області та сформульовано основні завдання і технічні вимоги до розробки вебсистеми. Виконано огляд існуючих програмних рішень для сервісів бронювання, зокрема українських платформ Appointer та Anolla, а також європейських Planity та Shore. На основі порівняльного аналізу визначено їхні сильні та слабкі сторони, що дозволило обґрунтувати актуальність створення власної спеціалізованої системи з повною українською локалізацією та адаптацією до потреб салонів краси. Архітектура системи обрана з урахуванням вимог проєкту: серверна частина реалізована на платформі .NET Core з використанням ASP.NET Core Web API, що забезпечує високу продуктивність, крос-платформеність та безпеку, а клієнтська частина побудована на бібліотеці React.js з використанням Material-UI для створення сучасного та адаптивного інтерфейсу. Для зберігання даних обрано легковагову базу даних SQLite з ORM Entity Framework Core, що спрощує розробку та підтримку. Обґрунтовано використання BCrypt для хешування паролів та JWT-токенів для авторизації, що забезпечує належний рівень безпеки системи. Визначено мінімальні та рекомендовані вимоги до апаратного забезпечення.

У другому розділі було проведено детальний аналіз функціональних і нефункціональних вимог до системи. Визначено дві основні ролі користувачів – клієнт та майстер, а також їхні права доступу, що забезпечує належний

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

рівень безпеки та персоналізації. Для візуалізації логіки роботи створено діаграми варіантів використання, активності та класів, які відображають ключові бізнес-процеси, зокрема реєстрацію, авторизацію, створення та обробку бронювань. Розроблено структуру реляційної бази даних SQLite, яка включає вісім основних таблиць: Users, Masters, Services, Bookings, MasterServices, Schedules, Payments та Reviews. Кожна таблиця має чітко визначені поля, типи даних та зв'язки, що забезпечує цілісність та консистентність даних. Реалізовано механізм перевірки доступності часових слотів, який враховує графік роботи майстра з таблиці Schedules та вже існуючі бронювання з таблиці Bookings, а також автоматично приховує минулі часи при виборі сьогоднішньої дати. Впроваджено бонусну систему, за якою клієнт отримує десять відсотків бонусів від вартості виконаної послуги, при цьому максимальна знижка бонусами обмежена тридцятьма відсотками. Реалізовано систему сповіщень, яка автоматично додає нагадування в кабінет клієнта при підтвердженні бронювання та видаляє їх при скасуванні або виконанні запису. Клієнтська частина підтримує зміну теми оформлення (світла/темна), мови інтерфейсу (українська/англійська) та валюти відображення цін (гривня, долар США, євро) з автоматичною конвертацією. Особливу увагу приділено оптимізації роботи з даними: кабінет майстра автоматично оновлюється кожні десять секунд, що дозволяє майстру бачити нові бронювання без необхідності оновлювати сторінку вручну.

У третьому розділі детально розглянуто порядок встановлення та налаштування вебсистеми. Описано структуру інтерфейсу, яка включає головну сторінку, сторінку послуг, сторінку майстрів, сторінку "Про нас", сторінку контактів, сторінку реєстрації та входу, кабінет клієнта, кабінет майстра, сторінку створення бронювання, сторінку оплати та сторінку профілю з налаштуваннями. Проведено комплексне тестування основних компонентів системи, яке підтвердило коректність валідації введених даних, ефективність контролю доступу користувачів до різних ресурсів і стабільність роботи API за різних сценаріїв використання. Тестування показало, що

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

система коректно обробляє помилки при реєстрації, авторизації, створенні бронювань, зміні пароля та відновленні доступу. Перевірено, що при виборі зайнятого часу система блокує таку можливість, а при виборі сьогоднішньої дати автоматично приховує минулі часові слоти. Підтверджено коректність роботи бонусної системи, зокрема нарахування балів після виконання послуги, обмеження максимальної знижки та списання балів при скасуванні бронювання. Особливу увагу приділено тестуванню серверної валідації, яка захищає систему від спроб несанкціонованого доступу через підміну даних у запитах. Усі результати тестування підтвердили готовність платформи до експлуатації в реальних умовах.

Отже, виконана робота охоплює всі основні етапи розробки складної вебсистеми – від аналізу та проектування до впровадження та тестування. Розроблена система є масштабованою, безпечною, зручною у використанні та адаптованою до потреб сучасного ринку послуг краси. Вона забезпечує повний цикл бронювання: від перегляду послуг та вибору майстра до створення запису, оплати та отримання сповіщень. Завдяки використанню сучасних технологій, таких як .NET Core, React, SQLite, JWT та BCrypt, система демонструє високу продуктивність, надійність та безпеку. Отримані результати мають практичну цінність і можуть бути використані для подальшого розвитку функціоналу, зокрема додавання адміністративної панелі, розширення звітності, інтеграції з додатковими платіжними системами та месенджерами, а також впровадження системи лояльності з більш гнучкими умовами нарахування бонусів.

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Appointer [Електронний ресурс] – Режим доступу до ресурсу: <https://appointer.com.ua/>
2. Anolla [Електронний ресурс] – Режим доступу до ресурсу: <https://anolla.com/>
3. Planity [Електронний ресурс] – Режим доступу до ресурсу: <https://www.planity.com/>
4. Shore [Електронний ресурс] – Режим доступу до ресурсу: <https://www.shore.com/>
5. Офіційна документація React.js [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
6. Material-UI Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/>
7. ASP.NET Core Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/aspnet/core/>
8. Entity Framework Core Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/ef/core/>
9. SQLite Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.sqlite.org/docs.html>
10. Introduction to JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction>
11. BCrypt.Net Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/BcryptNet/bcrypt.net>
12. LiqPay API Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.liqpay.ua/documentation>
13. Nodemailer documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://nodemailer.com/>
14. Learn REST API design [Електронний ресурс] – Режим доступу до ресурсу: <https://www.restapitutorial.com/>

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

15. What is middleware? [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.redhat.com/en/topics/middleware/what-is-middleware>
16. React Router Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/>
17. Axios Getting Started [Електронний ресурс] – Режим доступу до ресурсу:
<https://axios-http.com/docs/intro>
18. Як будувати UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/40575/>
19. Аналіз вимог [Електронний ресурс] – Режим доступу до ресурсу:
<https://visuresolutions.com/uk/alm-guide/requirement-analysis>
20. Що таке функціональні вимоги: приклади, визначення [Електронний ресурс] – Режим доступу до ресурсу: <https://visuresolutions.com/uk/alm-guide/functional-requirements>
21. Нефункціональні вимоги: приклади, типи, підходи [Електронний ресурс] – Режим доступу до ресурсу:
<https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/>
22. Cross-Origin Resource Sharing [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS>
23. SQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.w3schools.com/sql/>
24. FormData [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/API/FormData>
25. Офіційна документація Docker [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>
26. Microsoft Docs. (2026). Enable Cross-Origin Requests (CORS) in ASP.NET Core. Microsoft Learn. [Електронний ресурс]. Режим доступу:
<https://learn.microsoft.com/aspnet/core/security/cors>

		Бородачов М. А.			Житомирська політехніка. 26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

27. OAuth 2.0 для доступу до Google API [Електронний ресурс] – Режим доступу до ресурсу:
<https://developers.google.com/identity/protocols/oauth2>
28. Get started with draw.io [Електронний ресурс] – Режим доступу до ресурсу: <https://www.drawio.com/doc/>
29. Microsoft Docs. (2025). Configure JWT bearer authentication in ASP.NET Core. Microsoft Learn. [Електронний ресурс]. Режим доступу: <https://learn.microsoft.com/aspnet/core/security/authentication/configure-jwt-bearer-authentication>
30. Microsoft Docs. (2025). Best practices for RESTful web API design. Microsoft Learn. [Електронний ресурс]. Режим доступу: <https://learn.microsoft.com/azure/architecture/best-practices/api-design>
31. React Documentation. (2025). Built-in React Hooks. React.dev. [Електронний ресурс]. Режим доступу: <https://react.dev/reference/react/hooks>
32. Microsoft Q&A. (2026). Using EF Core Migrations across multiple machines. Microsoft Learn. [Електронний ресурс]. Режим доступу: <https://learn.microsoft.com/en-gb/answers/questions/>

ДОДАТКИ

		Бородачов М. А.			Житомирська політехніка.26.121.05.000 - ПЗ	Арк.
		Чижмотря О.Г.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

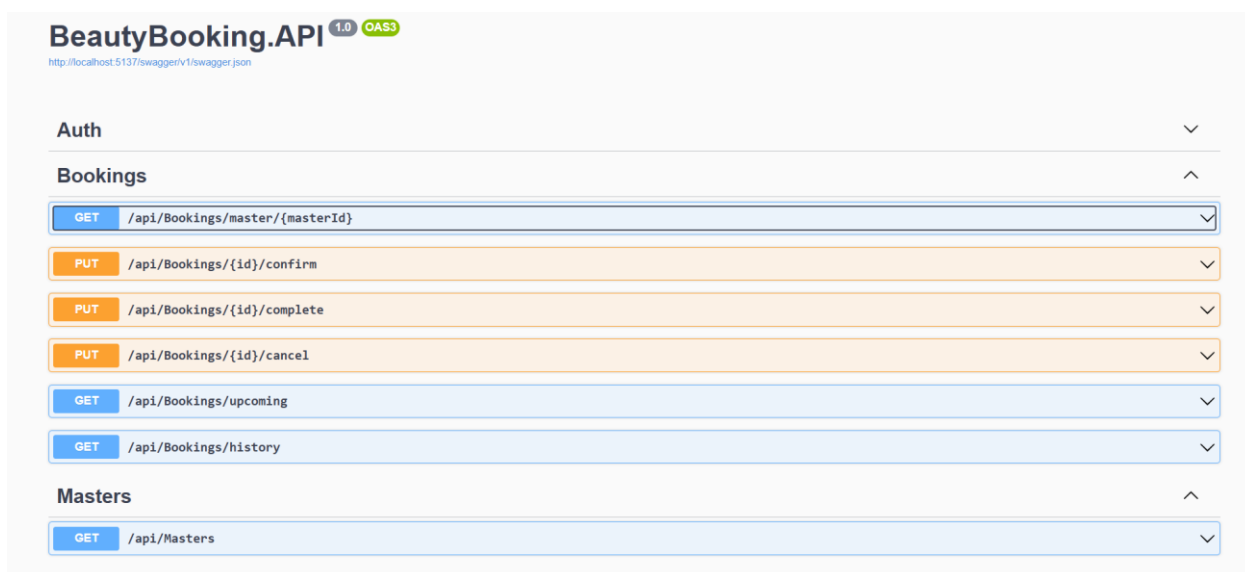


Рис. А.1. Swagger-документація API вебсистеми бронювання послуг салону краси

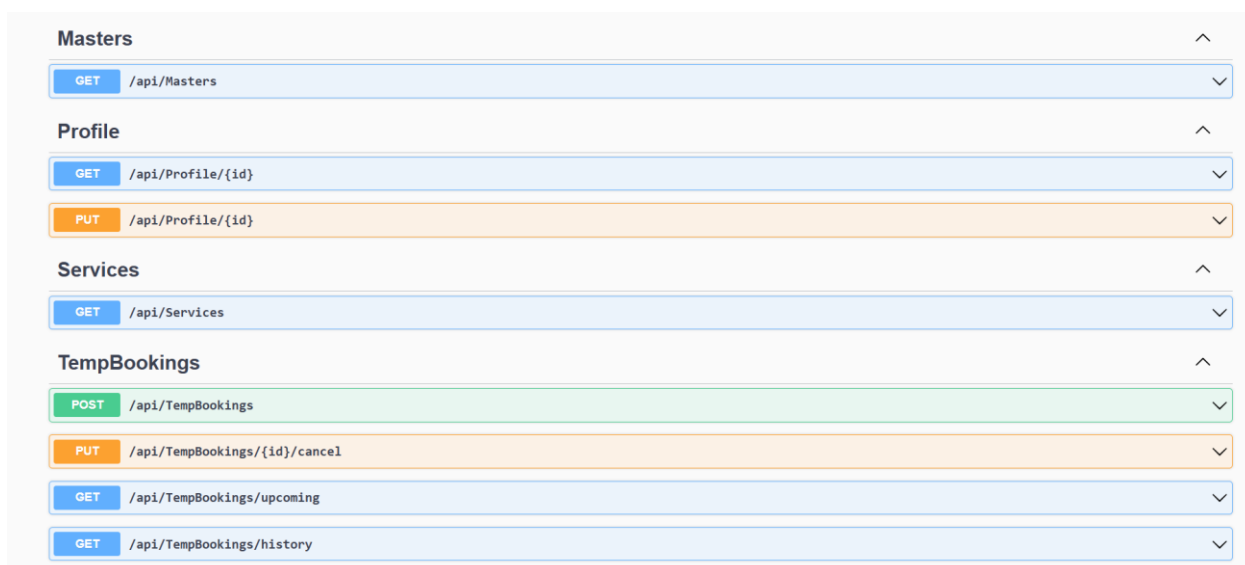


Рис. А.2. Swagger-документація API вебсистеми бронювання послуг салону краси

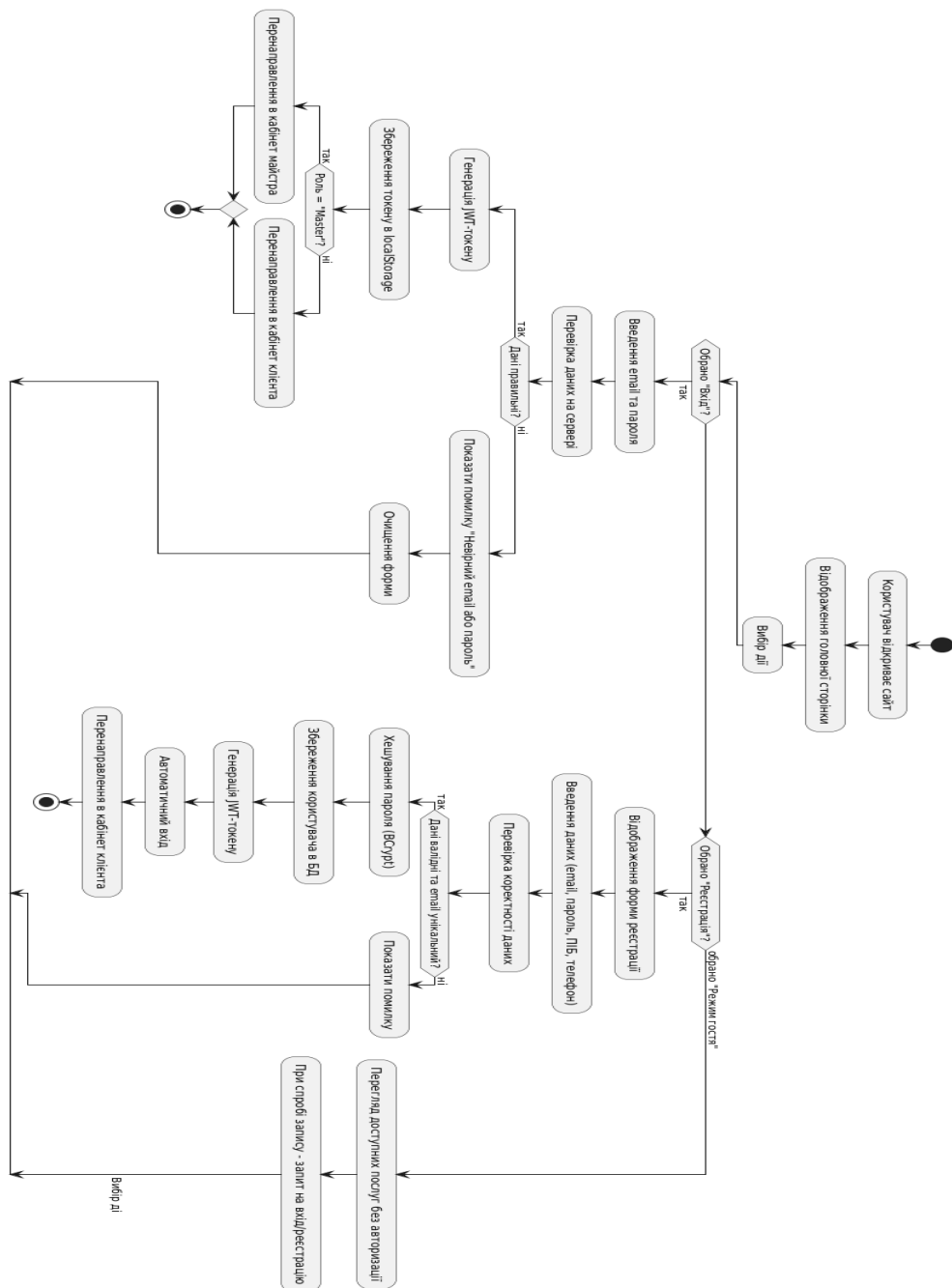


Рис. Б.1. Діаграма активності при першому вході на сайт

Глосарій

API (Application Programming Interface) – інтерфейс прикладного програмування; набір визначень та протоколів, що дозволяє різним програмним компонентам, зокрема клієнтській частині та серверу, взаємодіяти між собою.

Актор (Actor) – у мові UML це сутність, якою може бути людина, зовнішня система або пристрій, що взаємодіє з програмною системою, але не є її частиною. У розробленій системі акторами виступають Клієнт та Майстер.

Axios – HTTP-клієнт на основі Promise для браузеру та Node.js, що використовується для виконання асинхронних запитів до API з клієнтської частини застосунку.

Варіант використання (Use Case) – опис послідовності дій, які може виконувати система у відповідь на дії актора для досягнення певної цілі. Наприклад, "Створення бронювання" або "Підтвердження запису майстром".

Деплоймент (Deployment) – процес розгортання програмного забезпечення; перенесення програмного коду на робочий сервер або хмарну платформу для його безпосереднього використання кінцевими користувачами.

DTO (Data Transfer Object) – об'єкт передачі даних; шаблон проєктування, який використовується для передачі даних між підсистемами програми, наприклад між сервером та клієнтом, без розкриття внутрішньої структури бази даних.

Express.js – мінімалістичний вебфреймворк для Node.js, що використовується для створення серверної частини додатку та REST API. У даній роботі не використовувався, натомість застосовано ASP.NET Core.

JWT (JSON Web Token) – відкритий стандарт для створення токенів доступу, що використовується для аутентифікації користувачів у вебдодатках. У системі застосовуються окремі токени для клієнтів та майстрів.

Node.js – середовище виконання JavaScript на сервері, що дозволяє створювати високопродуктивні вебдодатки. У даній роботі не використовувався, натомість серверну частину реалізовано на .NET Core.

ORM (Object-Relational Mapping) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, дозволяючи працювати з даними як з об'єктами. У роботі використано Entity Framework Core.

React.js – JavaScript-бібліотека з відкритим кодом для створення інтерфейсів користувача, використана для розробки клієнтської частини системи у вигляді односторінкового додатку.

REST API – архітектурний стиль для побудови вебсервісів, що використовує HTTP-методи (GET, POST, PUT, DELETE) для роботи з ресурсами. Уся взаємодія між клієнтською та серверною частинами побудована на цьому принципі.

SPA (Single Page Application) – односторінковий додаток, який завантажує єдину HTML-сторінку та динамічно оновлює її вміст без перезавантаження. Клієнтська частина системи реалізована саме як SPA.

TypeScript – типізована надбудова над JavaScript, що додає статичну типізацію та покращує підтримку великих проєктів. Використовувався для розробки серверної частини на .NET Core (C# має вбудовану типізацію).

Фронтенд (Frontend) – клієнтська сторона користувацького інтерфейсу програмно-апаратної частини сервісу; те, з чим безпосередньо взаємодіє користувач у браузері. У системі фронтенд реалізовано на React.js.

Лістинг коду моделі даних Booking.cs:

```
namespace BeautyBooking.API.Models
{
    public class Booking
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int MasterId { get; set; }
        public int ServiceId { get; set; }
        public DateTime BookingDate { get; set; }
        public TimeSpan BookingTime { get; set; }
        public string Status { get; set; } = "Pending";
        public decimal TotalPrice { get; set; }
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
        public string? AdditionalOptions { get; set; }

        // Додаткові поля для бонусів
        public decimal OriginalPrice { get; set; } // Оригінальна ціна без знижки
        public int BonusPointsUsed { get; set; } // Використані бонуси
    }
}
```

Лістинг коду моделі даних User.cs:

```
using System.ComponentModel.DataAnnotations;

namespace BeautyBooking.API.Models
{
    public class User
    {
        public int Id { get; set; }

        [Required, EmailAddress]
        public string Email { get; set; } = string.Empty;

        [Required]
        public string PasswordHash { get; set; } = string.Empty;

        [Required]
        public string FullName { get; set; } = string.Empty;
    }
}
```

```

[Required, Phone]
public string Phone { get; set; } = string.Empty;

public string Role { get; set; } = "User";
public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

// Улюблені майстри
public string FavoriteMasters { get; set; } = "";

// Бонусні бали
public int BonusPoints { get; set; } = 0;
}
}

```

Лістинг коду моделі даних Master.cs:

```

using System.Text.Json.Serialization;

namespace BeautyBooking.API.Models
{
    public class Master
    {
        public int Id { get; set; }

        public int? UserId { get; set; }

        public string FullName { get; set; } = string.Empty;
        public string Specialization { get; set; } = string.Empty;
        public string Role { get; set; } = string.Empty;
        public string Experience { get; set; } = string.Empty;
        public decimal Rating { get; set; } = 5.0m;
        public int Reviews { get; set; } = 0;
        public string? Avatar { get; set; }
        public string? About { get; set; }
        public bool IsActive { get; set; } = true;
    }
}

```

```

    [JsonIgnore]
    public User? User { get; set; }
}
}

```

Лістинг коду AuthController.cs:

Демонструє реалізацію безпеки – JWT-токени, хешування паролів BCrypt, ролі користувачів.

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using BeautyBooking.API.Data;
using BeautyBooking.API.Models;
using Microsoft.AspNetCore.Authorization;

```

```

namespace BeautyBooking.API.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthController : ControllerBase
    {
        private readonly ApplicationDbContext _context;
        private readonly IConfiguration _configuration;

        public AuthController(ApplicationDbContext context, IConfiguration configuration)
        {
            _context = context;
            _configuration = configuration;
        }

        [HttpGet("me")]
        [Authorize]

```

```

public async Task<IActionResult> GetCurrentUser()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null)
    {
        return Unauthorized(new { message = "Користувача не знайдено" });
    }

    var userId = int.Parse(userIdClaim.Value);
    var user = await _context.Users.FindAsync(userId);

    if (user == null)
        return NotFound(new { message = "Користувача не знайдено" });

    return Ok(new
    {
        user.Id,
        user.Email,
        user.FullName,
        user.Phone,
        user.Role,
        user.CreatedAt,
        user.BonusPoints,
        user.FavoriteMasters
    });
}

[HttpPost("register")]
public async Task<IActionResult> Register([FromBody] RegisterRequest request)
{
    if (await _context.Users.AnyAsync(u => u.Email == request.Email))
    {
        return BadRequest(new { message = "Користувач з таким email вже існує" });
    }
}

```

```

string passwordHash = BCrypt.Net.BCrypt.HashPassword(request.Password);

var user = new User
{
    Email = request.Email,
    PasswordHash = passwordHash,
    FullName = request.FullName,
    Phone = request.Phone,
    CreatedAt = DateTime.UtcNow,
    Role = "User",
    FavoriteMasters = "[]",
    BonusPoints = 50
};

_context.Users.Add(user);
await _context.SaveChangesAsync();

var token = GenerateJwtToken(user);

return Ok(new {
    token,
    user = new {
        user.Id,
        user.Email,
        user.FullName,
        user.Phone,
        user.Role,
        user.CreatedAt,
        user.BonusPoints
    }
});
}

[HttpPost("login")]
public async Task<IActionResult> Login([FromBody] LoginRequest request)

```

```

{
    var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);

    if (user == null || !BCrypt.Net.BCrypt.Verify(request.Password, user.PasswordHash))
    {
        return BadRequest(new { message = "Невірний email або пароль" });
    }

    var token = GenerateJwtToken(user);

    return Ok(new {
        token,
        user = new {
            user.Id,
            user.Email,
            user.FullName,
            user.Phone,
            user.Role,
            user.CreatedAt,
            user.BonusPoints
        }
    });
}

```

```

[HttpPost("setmasterrole")]
public async Task<IActionResult> SetMasterRole([FromBody] SetMasterRoleRequest
request)
{
    var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);
    if (user == null)
        return NotFound(new { message = "Користувача не знайдено" });

    user.Role = "Master";
    await _context.SaveChangesAsync();
}

```

```

        return Ok(new { message = $"Користувачу {request.Email} присвоєно роль Master"
    });
    }

    [HttpPost("setuserrole")]
    public async Task<IActionResult> SetUserRole([FromBody] SetMasterRoleRequest
request)
    {
        var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);
        if (user == null)
            return NotFound(new { message = "Користувача не знайдено" });

        user.Role = "User";
        await _context.SaveChangesAsync();

        return Ok(new { message = $"Користувачу {request.Email} присвоєно роль User" });
    }

    [HttpGet("fix-name")]
    public async Task<IActionResult> FixName()
    {
        var user = await _context.Users.FirstOrDefaultAsync(u => u.Email ==
"anna@beauty.com");
        if (user == null)
        {
            return NotFound(new { message = "Користувача не знайдено" });
        }

        user.FullName = "Анна Коваленко";
        await _context.SaveChangesAsync();

        return Ok(new { message = "Ім'я виправлено!", fullName = user.FullName });
    }

    // Зміна паролю

```

```

[HttpPost("change-password")]
[Authorize]
public async Task<IActionResult> ChangePassword([FromBody] ChangePasswordRequest
request)
{
    try
    {
        var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
        if (userIdClaim == null)
        {
            return Unauthorized(new { message = "Користувача не знайдено" });
        }

        var userId = int.Parse(userIdClaim.Value);
        var user = await _context.Users.FindAsync(userId);

        if (user == null)
        {
            return NotFound(new { message = "Користувача не знайдено" });
        }

        // Перевіряємо старий пароль
        if (!BCrypt.Net.BCrypt.Verify(request.OldPassword, user.PasswordHash))
        {
            return BadRequest(new { message = "Невірний старий пароль" });
        }

        // Хешуємо новий пароль
        user.PasswordHash = BCrypt.Net.BCrypt.HashPassword(request.NewPassword);
        await _context.SaveChangesAsync();

        return Ok(new { message = "Пароль успішно змінено" });
    }
    catch (Exception ex)
    {

```

```

        return StatusCode(500, new { error = ex.Message });
    }
}

private string GenerateJwtToken(User user)
{
    var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
    var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    var claims = new[]
    {
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Email, user.Email),
        new Claim(ClaimTypes.Name, user.FullName),
        new Claim(ClaimTypes.Role, user.Role)
    };

    var token = new JwtSecurityToken(
        issuer: _configuration["Jwt:Issuer"],
        audience: _configuration["Jwt:Audience"],
        claims: claims,
        expires: DateTime.UtcNow.AddDays(7),
        signingCredentials: credentials
    );

    return new JwtSecurityTokenHandler().WriteToken(token);
}

}

public class RegisterRequest
{
    public string Email { get; set; } = string.Empty;
    public string Password { get; set; } = string.Empty;
    public string FullName { get; set; } = string.Empty;

```

```

        public string Phone { get; set; } = string.Empty;
    }

    public class LoginRequest
    {
        public string Email { get; set; } = string.Empty;
        public string Password { get; set; } = string.Empty;
    }

    public class SetMasterRoleRequest
    {
        public string Email { get; set; } = string.Empty;
    }

    public class ChangePasswordRequest
    {
        public string OldPassword { get; set; } = string.Empty;
        public string NewPassword { get; set; } = string.Empty;
    }
}

```

Лістинг коду Створення бронювання BookingsController.cs:

```

// ===== МЕТОД ДЛЯ СТВОРЕННЯ ЗАПИСУ =====
[HttpPost]
[Authorize]
public async Task<IActionResult> CreateBooking([FromBody] CreateBookingDto
bookingDto)
{
    try
    {
        var currentUserId = GetCurrentUser();
        if (currentUserId != bookingDto.UserId)
        {
            return Forbid();
        }
    }
}

```

```

var master = await _context.Masters.FindAsync(bookingDto.MasterId);
if (master == null)
{
    return BadRequest(new { message = "Майстра не знайдено" });
}

var service = await _context.Services.FindAsync(bookingDto.ServiceId);
if (service == null)
{
    return BadRequest(new { message = "Послугу не знайдено" });
}

var existingBooking = await _context.Bookings
    .FirstOrDefaultAsync(b => b.MasterId == bookingDto.MasterId
        && b.BookingDate == bookingDto.BookingDate
        && b.BookingTime == TimeSpan.Parse(bookingDto.BookingTime)
        && b.Status != "Cancelled");

if (existingBooking != null)
{
    return BadRequest(new { message = "Цей час вже зайнятий. Оберіть інший час."
});
}

var booking = new Booking
{
    UserId = bookingDto.UserId,
    MasterId = bookingDto.MasterId,
    ServiceId = bookingDto.ServiceId,
    BookingDate = bookingDto.BookingDate,
    BookingTime = TimeSpan.Parse(bookingDto.BookingTime),
    Status = "Pending",
    TotalPrice = bookingDto.TotalPrice,
    OriginalPrice = bookingDto.OriginalPrice,

```

```

        BonusPointsUsed = bookingDto.BonusPointsUsed,
        CreatedAt = DateTime.UtcNow
    };

    _context.Bookings.Add(booking);
    await _context.SaveChangesAsync();

    return Ok(new {
        success = true,
        bookingId = booking.Id,
        message = "Запис успішно створено"
    });
}
catch (Exception ex)
{
    return StatusCode(500, new { error = ex.Message });
}
}

[HttpGet("master/{masterId}")]
[Authorize]
public async Task<IActionResult> GetMasterBookings(int masterId)
{
    try
    {
        var currentUserId = GetCurrentUser();
        var currentRole = GetCurrentUserRole();

        if (currentRole != "Master" || currentUserId != masterId)
        {
            return Forbid();
        }

        var bookings = await _context.Bookings
            .Where(b => b.MasterId == masterId)

```

```

        .Select(b => new
        {
            b.Id,
            b.UserId,
            b.MasterId,
            b.ServiceId,
            b.BookingDate,
            b.BookingTime,
            b.Status,
            b.TotalPrice,
            b.OriginalPrice,
            b.BonusPointsUsed,
            b.CreatedAt,
            ServiceName = _context.Services.Where(s => s.Id == b.ServiceId).Select(s =>
s.Name).FirstOrDefault() ?? "Невідомо",
            ClientName = _context.Users.Where(u => u.Id == b.UserId).Select(u =>
u.FullName).FirstOrDefault() ?? "Клієнт",
            ClientPhone = _context.Users.Where(u => u.Id == b.UserId).Select(u =>
u.Phone).FirstOrDefault() ?? ""
        })
        .ToListAsync();

var sortedBookings = bookings
    .OrderByDescending(b => b.BookingDate)
    .ThenBy(b => b.BookingTime.ToString())
    .ToList();

return Ok(sortedBookings);
}
catch (Exception ex)
{
    return StatusCode(500, new { error = ex.Message });
}
}

```