МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії

програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК САЛОНУ КРАСИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ УПРАВЛІННЯ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувачка _____ **Лада КОВШУН**

«__» _____ 20__ р.

Керівник роботи

ст. викладачка _____ **Марина ФАЛЕНКОВА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувачки

Ковшун Лади

1. Тема кваліфікаційної роботи «Вебзастосунок салону краси з інформаційною системою управління» затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2025 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом є розроблений вебзастосунок салону краси з інформаційною системою управління.

4. Перелік питань, що підлягають розробці:

− дослідження та аналіз існуючих аналогів програмного забезпечення;

− формування вимог до програмного забезпечення;

− визначення архітектури для проєктування програмного забезпечення;

− проєктування програмного забезпечення;

− розробка програмного забезпечення;

− тестування роботи програмного забезпечення;

− проведення аналізу результатів розробки.

5. Перелік графічних матеріалів:

− презентація кваліфікаційної бакалаврської роботи.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «02» січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок салону краси з інформаційною системою управління**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	26.12.2025	02.01.2026	*Виконано*
2.	Огляд літератури та аналіз аналог ПЗ	26.01.2026	06.02.2026	*Виконано*
3.	Складання календарного плану КБР	02.02.2026	06.02.2026	*Виконано*
4.	Аналіз предметної області та визначення вимог	09.02.2026	13.02.2026	*Виконано*
5.	Розробка проєктних рішень	16.02.2026	20.02.2026	*Виконано*
6.	Розробка бази даних	23.02.2026	06.03.2026	*Виконано*
7.	Реалізація бекенду вебзастосунку	09.03.2026	20.03.2026	*Виконано*
8.	Реалізація фронтенду вебзастосунку	20.03.2026	03.04.2026	*Виконано*
9.	Тестування та відлагодження функціоналу	03.04.2026	15.05.2026	*Виконано*
10.	Відгук керівника КБР	15.06.2026	16.06.2026	*Виконано*
11.	Оформлення КБР та презентації	20.05.2026	22.05.2026	*Виконано*
12.	Попередній захист	25.05.2026	25.05.2026	*Виконано*
13.	Рецензування	17.06.2026	18.06.2026	*Виконано*
14.	Завершення оформлення КБР та презентації	01.06.2026	04.06.2026	*Виконано*
15.	Захист кваліфікаційної роботи	22.06.2026	23.06.2026	*Виконано*

Здобувачка _____ **Лада КОВШУН**

«__» _____ 2026 р.

Керівник роботи

Ст. викладачка _____ **Марина ФАЛЕНКОВА**

«__» _____ 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок салону краси з інформаційною системою управління»

Здобувачка 408 гр.: Ковшун Лада

Керівник: ст.викладачка Фаленкова Марина

Актуальність теми кваліфікаційної бакалаврської роботи зумовлена тенденціями спрощення управлінням малого та великого бізнеса за допомогою тематичного вебзастосунку. Вебзастосунок салону краси зорієнтований на бізнес сфери обслуговування, що допоможе в спрощенні актуалізації даних, допоможе в управлінні даною сферою власного бізнесу. Малий та середній бізнес здебільшого потребує в спрощеній та єдиній системі управління, що буле зручним для введення такого бізнесу. Цей застосунок об'єднає усіх користувачів в єдиній базі, таких як: клієнтів салону краси, працівників салонів, власників цього бізнесу. З боку клієнтів спрощена система для вибору категорії послуги та майстра, зручний запис та нагадування в календарі подій про запис на послугу. З боку майстрів, зручний розклад записів, заздалегідь розрахований процент заробітної плати, доступ до перегляду наявних матеріалів для роботи. З боку адміністрації, можливість управління записами, керуванням розкладів роботи та менеджмент системи. Самі ж власники зможуть переглядати звітування роботи свого бізнесу, а саме чи є він вигідним, скільки витрат і який добуток з нього є.

Мета: розробка вебзастосунку салону краси для оптимізації процесу управління.

Об’єкт роботи: процес управління салоном краси з інтегрованою інформаційною системою управління бізнесом.

Предмет роботи: методи та технології розробки вебзастосунку салону краси з інформаційною системою управління.

Кваліфікаційна бакалаврська робота складається з вступу, чотирьох розділів, висновків, переліку джерел посилання та додатку.

У вступі зазначена актуальність теми, головна мета створеного проєкту та огляд поставленої задачі, предмет дослідження кваліфікаційної бакалаврської роботи, а також об’єкт самої роботи.

У першому розділі надається опис аналітичної частини, огляд існуючих застосунків-аналогів застосунків з інформаційної системою управління бізнесом, визначення функціоналу, що наявний, їх переваги та недоліки, зазначається розроблений план виконання поставлених завдань. Наступною вже частиною розділу є формування та опис специфікації вимог до вебзастосунку, що розробляється протягом виконання роботи.

У другому розділі є опис процесу розробки проєктних рішень, що надають факт виконання специфікації вимог до проєкту вебзастосунку. Відбувається процес моделювання об’єкту та предмету дослідження, формуються функціональні та інформаційні моделі, за результатом маємо вирішення поставленої задачі.

У третьому розділі йде детальний опис результату виконаної роботи з конструювання та моделювання вебзастосунку, де розроблено UML-діаграми обрано мови програмування та інтеграція технологій, зазначення основних використаних бібліотек та фреймворків, опис інтерфейсу.

У четвертому розділі демонструється проведена робота над вебзастосунком та його тестування, разом з детальним аналізов результатів тестування.

У висновках зазначений аналіз проведеної роботи та результатів, зазначається додерження основної мети програмного забезпечення, розробленого протягом виконання завдань.

Кваліфікаційна бакалаврська робота викладена на 80 сторінок, вона містить 4 розділи, 47 ілюстрацій, 10 таблиць, 21 джерел в переліку посилань.

Ключові слова: інформаційна система управління, створення вебзастосунку, єдина система управління бізнесом, розробка програмного забезпечення, салон краси.

ABSTRACT

for the Bachelor's Thesis

“Web application for a beauty salon with an information management system”

Student of group 408: Kovshun Lada

Supervisor: Senior Lecturer Falenkova Marina

The relevance of this bachelor’s thesis stems from the trend toward simplifying the management of small and large businesses through specialized web applications. This web application for a beauty salon is designed for the service industry; it will help streamline data updates and facilitate the management of this aspect of the business. Small and medium-sized businesses generally require a simplified and unified management system that is convenient for launching such a business. This application will bring all users together in a single database, including beauty salon clients, salon employees, and business owners. For clients, it offers a simplified system for selecting service categories and stylists, convenient booking, and calendar reminders for service appointments. For stylists, it provides a convenient appointment schedule, pre-calculated salary percentages, and access to view available supplies for work. For the administration, it offers the ability to manage appointments, work schedules, and the system itself. The owners themselves will be able to view reports on their business’s performance, specifically whether it is profitable, what the expenses are, and what the revenue is.

Objective: To develop a web application for a beauty salon to optimize management processes.

Scope of work: The management process of a beauty salon with an integrated business management information system.

Subject of work: Methods and technologies for developing a web application for a beauty salon with a management information system.

This bachelor’s thesis consists of an introduction, four chapters, conclusions, a list of references, and an appendix.

The introduction outlines the relevance of the topic, the main objective of the project, an overview of the task at hand, the subject of the bachelor's thesis, and the focus of the thesis itself.

The first chapter provides a description of the analytical part, an overview of existing applications similar to business management information systems, a definition of their available functionality, their advantages and disadvantages, and outlines the plan developed to accomplish the set tasks. The next part of the chapter involves the formulation and description of the requirements specification for the web application being developed throughout the course of the work.

The second chapter describes the process of developing design solutions that demonstrate compliance with the requirements specification for the web application project. It involves modeling the object and subject of study, creating functional and information models, and ultimately arriving at a solution to the problem at hand.

The third chapter provides a detailed description of the results of the work performed on the design and modeling of the web application, including the development of UML diagrams, the selection of programming languages and technology integration, a list of the main libraries and frameworks used, and a description of the interface.

Chapter four presents the work carried out on the web application and its testing, along with a detailed analysis of the test results.

The conclusions provide an analysis of the work performed and the results, noting that the primary objective of the software developed during the course of the project has been achieved.

The bachelor's thesis consists of 80 pages and contains four chapters, 47 figures, 10 tables, and 21 sources in the reference list.

Keywords: management information system, web application development, integrated business management system, software development, beauty salon.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Огляд застосунків аналогів 7
1.2 Аналіз розроблюваної системи вебзастосунку 12
1.3 Використання CRM-моделі для вебзастосунку з клієнтською базою 15
Висновки до розділу 1 17
2 ЗАСТОСУВАННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ ПРИ ПРОЄКТУЮВАННІ ВЕБЗАСТОСУНКУ САЛОНУ КРАСИ 19
2.1 Дослідження методів створення вебзастосунку 19
2.2 Використання технологій при розробці вебзастосунку салону краси 21
2.3 Специфікація вимог до програмного забезпечення вебзастосунку салону краси «Beauty book» 25
Висновки до розділу 2 32
3 ПРОЄКТУВАННЯ ТА ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ ВЕБЗАСТОСУНКУ 34
3.1 Проєктування вебзастосунку за допомогою розробки UML-діаграм 34
3.1.1 Діаграма використання 35
3.1.2 Діаграма класів 39
3.1.3 Діаграма станів та переходів 42
3.1.4 Діаграма компонентів 44
3.1.5 Діаграма пакетів 45
3.2 Огляд задіяних технологій розробки вебзастосунку 48
3.2.1 Використання технологій front-end розробки 49
3.2.2 Використання технологій back-end розробки 50
3.2.3 Вплив CRM та його використання у розробці 51
Висновки до розділу 3 51
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ 53
4.1 Огляд реалізації користувацького інтерфейсу 53

4.2 Реалізація бізнес-логіки 58
4.2.1 Реалізація головної сторінки вебзастосунку 58
4.2.2 Реалізація реєстрації та запису 60
4.2.3 Реалізація системи аналітики та звітності 64
4.2.4 Реалізація системи повідомлень та бонусів 67
4.2.5 Редагування розкладу та профілю 70
4.3 Тестування вебзастосунку 72
Висновки до розділу 4 75
ВИСНОВКИ 76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 77
ДОДАТОК А Лістинг коду дошки звітності python 79
ДОДАТОК Б Апробація 84

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

3НФ – 3 нормальна форма

ACID – Atomicity, Consistency, Isolation, Durability

API – Application Programming Interface

CRM – Customer Relationship Management

DOM – Document Object Model

JS – JavaScript

MIS – Management Information System

PWA – Progressive Web App

RBAC – Role-Based Access Control

SDLC – Software Development Life Cycle

SMS – Short Message Service

SQL – Structured Query Language

UI – User Interface

UI/UX – User Interface / User Experience

UML – Unified Modeling Language

ВСТУП

Актуальність теми кваліфікаційної роботи бакалавра зумовлена тенденціями спрощення управлінням малого та великого бізнеса за допомогою тематичного вебзастосунку. Вебзастосунок салону краси зорієнтований на бізнес сфери обслуговування, що допоможе в спрощенні актуалізації даних, допоможе в управлінні даною сферою власного бізнесу. Малий та середній бізнес здебільшого потребує в спрощеній та єдиній системі управління, що буле зручним для введення такого бізнесу. Цей застосунок об'єднає усіх користувачів в єдиній базі, таких як: клієнтів салону краси, працівників салонів, власників цього бізнесу. З боку клієнтів, буде спрощена система для вибору категорії послуги та майстра, зручний запис та нагадування в календарі подій про запис на послугу. З боку майстрів. Буде зручний розклад записів, заздалегідь розрахований процент заробітньої плати, доступ до перегляду наявних матеріалів для роботи. З боку адміністрації, буде можливість управління записами, керуванням розкладів роботи та менеджмент системи. Самі ж власники зможуть переглядати звітування роботи свого бізнесу, а саме чи є він вигідним, скільки витрат і який добуток з нього є.

Мета: розробка вебзастосунку салону краси для оптимізації процесу управління.

Відповідно для досягнення визначеної мети необхідно вирішити наступні **завдання:**

− аналіз застосунків-аналогів;

− розробка специфікації вимог до програмного забезпечення;

− проєктування архітектури застосунку;

− розробка дизайну та макету застосунку;

− розробка front-end частини вебзастосунку на базі технології Vue.js, React.js;

− розробка back-end частини вебзастосунку на базі технології Next.js, PostgreSQL, Python;

− тестування програмного забезпечення.

Об'єкт роботи: процес управління салоном краси з інтегрованою інформаційною системою управління бізнесом.

Предмет роботи: методи та технології розробки вебзастосунку салону краси з інформаційною системою управління.

Сфера застосування: вебзастосунок салону краси з інформаційною системою управління можна використовувати як застосунок для запису та управління малим та середнім бізнесом. Застосунок також корисний для власників самого бізнесу, що стає у допомозі у пошуку клієнтів, актуалізації портфоліо майстрів, автоматизації ведення бізнесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд застосунків аналогів

Перед початком розробки, основною задачею, перш за все, є аналіз застосунків-аналогів, що вже наявні на ринку програмного забезпечення. Таким чином, можна визначити основний функціонал, які переваги доцільно буде включити в розробку власного застосунку, які недоліки можна усунути і виявити актуальність. Все це потрібно проаналізувати, щоб таким чином, розробити конкурентоспроможний проєкт, що зацікавить майбутній користувачів та інвесторів. Потрібно приділити увагу не тільки функціоналу, але й інтерфейсу, адже це є «обкладинкою» самого продукту та послуг, що подаються клієнтам. Він повинен бути гнучким та інтуїтивно зрозумілим, поєднувати в собі важливий функціонал, що буде охоплювати основну аудиторію та зацікавлювати більшість.

Для аналізу обраної області розробки, було обрано наступні програмні забезпечення, а саме: Beauty Pro, Altegio та EasyWeek. Перейдемо до найпершого з них, до Beauty Pro.

Beauty Pro

Дане програмне забезпечення охоплює в собі створення комплексної автоматизації бізнес-процесів у сфері надання послуг краси, має аналітичний блок, що дуже спрощує роботу для багатьох власників середнього та великого бізнесу. Нажаль, фунціонал обмежений мобільністю, через необхідність встановлення ПЗ на локальний комп’ютер. Хоча функціонал даного застосунку охоплює в собі автоматизацію багатьох процесів, через наявність єдиного обліку клієнтської бази, що має в собі CRM модель. Це є значною перевагою для використання. Також, наявна спрощеність менеджменту та бухгалтерії, через використання розрахунку заробітної плати для персоналу та аналіз ефективності роботи самого салону, що допомагає в просуванні бізнесу, через маркетинговий підхід [1].

Контроль салонів краси відбувається за наступними напрямками:

− фінансовий менеджмент;

− складський облік матеріалів;

− розрахунок заробітної плати;

− аналіз ефективності маркетингу.

Таблиця 1.1 – Опис застосунку «Beauty Pro» [1]

Розробник	AIHELPS
Архітектура	Client-Server application (Desktop based)
Виробник	Приватна компанія Beauty Pro
Мова реалізації	C#, .NET, SQL Server
Перелік функцій	– попередній запис клієнтів; – контроль фінансових потоків; – звіти про рентабельність бізнесу; – управління базою клієнтів; – ведення детальних карток клієнтів з історією формул та візитів; – автоматичне списання при наданні послуг; – формування понад 50 видів управлінських звітів; – робота в офлайн-режимі без доступу до інтернету.
Переваги	1) висока швидкість та стабільність роботи; 2) глибока аналітика, що дозволяє відстежувати LTV клієнта; 3) наявність локальної бази даних для безпеки інформації; 4) єдина клієнтська база.
Недоліки	1) складність процесу оновлення через десктопну архітектуру застосунку; 2) відсутність повноцінної хмарної гнучкості; 3) застарілий інтерфейс.

Внутрішній інтерфейс застосунку «Beauty Pro» має застарілий стиль оформлення, що є не дуже доцільним для сучасного ринку розробки, але охоплює багатий функціонал бухгалтерського обліку (рис. 1.1).

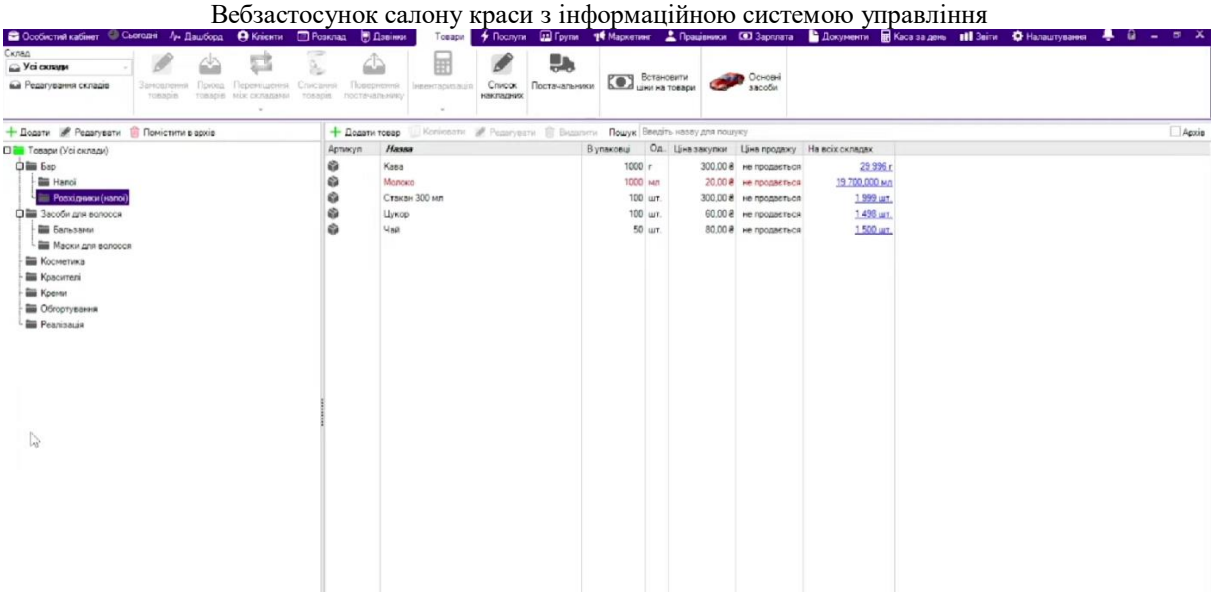

Рисунок 1.1 – Інтерфейс робочої області програмного забезпечення «Beauty Pro»

Altegio

Застосунок містить в собі призначення автоматизації бізнесу сфери послуг з наявним управлінням клієнтським досвідом і, навідмінну від попереднього аналогу, є повністю хмарним, через що спрощується керування цілою мережою салонів краси, тобто орієнтований на великий та середній бізнеси. Цей застосунок також містить в собі . CRM-система, через що охоплює велику кількість користувачів та має функціонал комунікації, що дуже необхідний для роботи. Наприклад, інтеграція месенджерів для спілкування, IP-телефонія та онлайн-запис [2].

Таблиця 1.2 – Опис застосунку «Altegio» [2]

Розробник	Altegio
Архітектура	3-tier SaaS application
Виробник	Міжнародна компанія Altegio
Мова реалізації	PHP / JavaScript

Кінець таблиці 1.2

Перелік функцій	– онлайн-запис через соціальні мережі та власні віджети, наявні в самому застосунку; – інтеграція з IP-телефонією; – – автоматична розсилка SMS та Push-повідомлень про запланований візит; – мобільний застосунок для адміністраторів та клієнтів; – управління лояльністю (бонусні рахунки, абонементи); – CRM-система з історією візитів .
Переваги	1) кросплатформеність; 2) велика кількість готових інтеграцій із зовнішніми сервісами; 3) можливість масштабування для великих мережевих бізнесів; 4) розширений функціонал для персоналу.
Недоліки	1) висока вартість щомісячної підписки порівняно з аналогами; 2) складний інтерфейс з надмірною функціональністю для малих студій; 3) залежність від стабільності роботи хмарного середовища.

На відміну від «Beauty Pro», застосунок «Altegio» має дуже гнучкий та приємний інтерфейс, що привертає клієнтів даного продукту. Розглядаючи клієнтську базу (рис.1.2) можна зручно вносити зміни або переглядати тенденції самих клієнтів, що надає змогу швидкого аналізу.

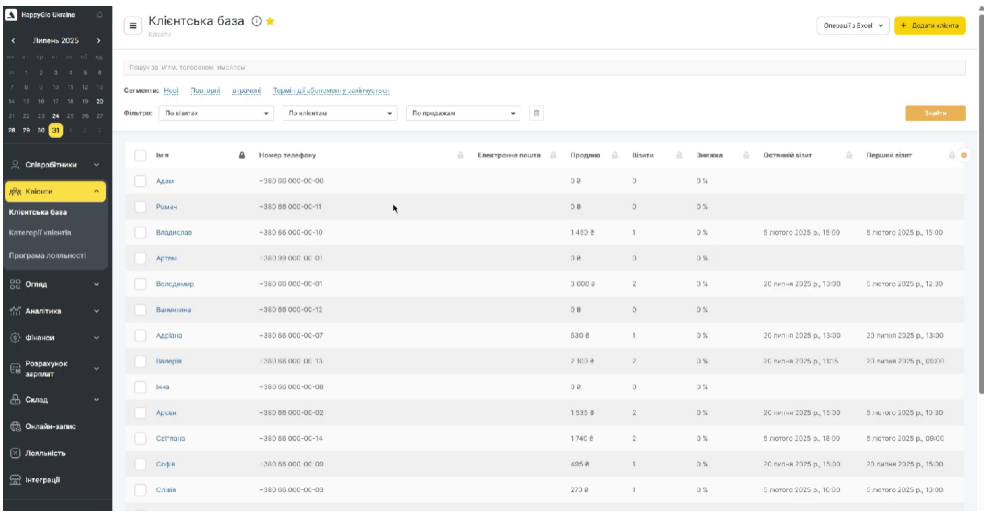

Рисунок 1.2 – Інтерфейс клієнтської бази «Altegio»

EasyWeek

Дане програмне забезпечення вперш за все орієнтоване на швидкий онлайн-запис клієнтів на послуги та керування розкладом персоналу, що значно спрощує буденний менеджмент роботи салону. Має поширення на європейський ринок через швидке налаштування під задані потреби та зручну фіксацію розрахунків клієнтів, що відповідає європейським стандартам. Орієнтир спланований більш на малий та середній бізнес, але охоплює більше основного функціоналу, що допомагає в його роботі. Великою перевагою також є інтеграція платіжних систем, зокрема й українських, наприклад таких як Monobank, що спрощує фіскалізацію чеків [3].

Таблиця 1.3 – Опис застосунку «EasyWeek» [3]

Розробник	EasyWeek
Архітектура	3-tier web application
Виробник	EasyWeek Tech
Мова реалізації	JavaScript (Node.js / React)
Перелік функцій	– персональна вебсторінка для онлайн-запису; – інтеграція з українськими фінтех-сервісами (Monobank); – автоматизація робочого графіка майстрів; – модуль фіскалізації операцій (ПРРО); – керування філіями в одному кабінеті.
Переваги	1) сервери розташовані в ЄС, що забезпечує високий рівень безпеки; 2) мінімалістичний та інтуїтивно зрозумілий інтерфейс застосунку; 3) швидке налаштування віджетів; 4) швидке налаштування персональних кабінетів та керування.
Недоліки	1) складний інтерфейс для новачків через великий функціонал застосунку; 2) платний функціонал для розширених маркетингових можливостей; 3) нецентралізований підхід.

Застосунок «EasyWeek» підхоплює своїм гнучким та зрозумілим інтерфейсом, який є як перевагою, так і недоліком застосунку. В свою чергу, в

даному ПЗ дуже зручне налаштування віджетів (рис.1.3), що є перевагою сучасного застосунку.

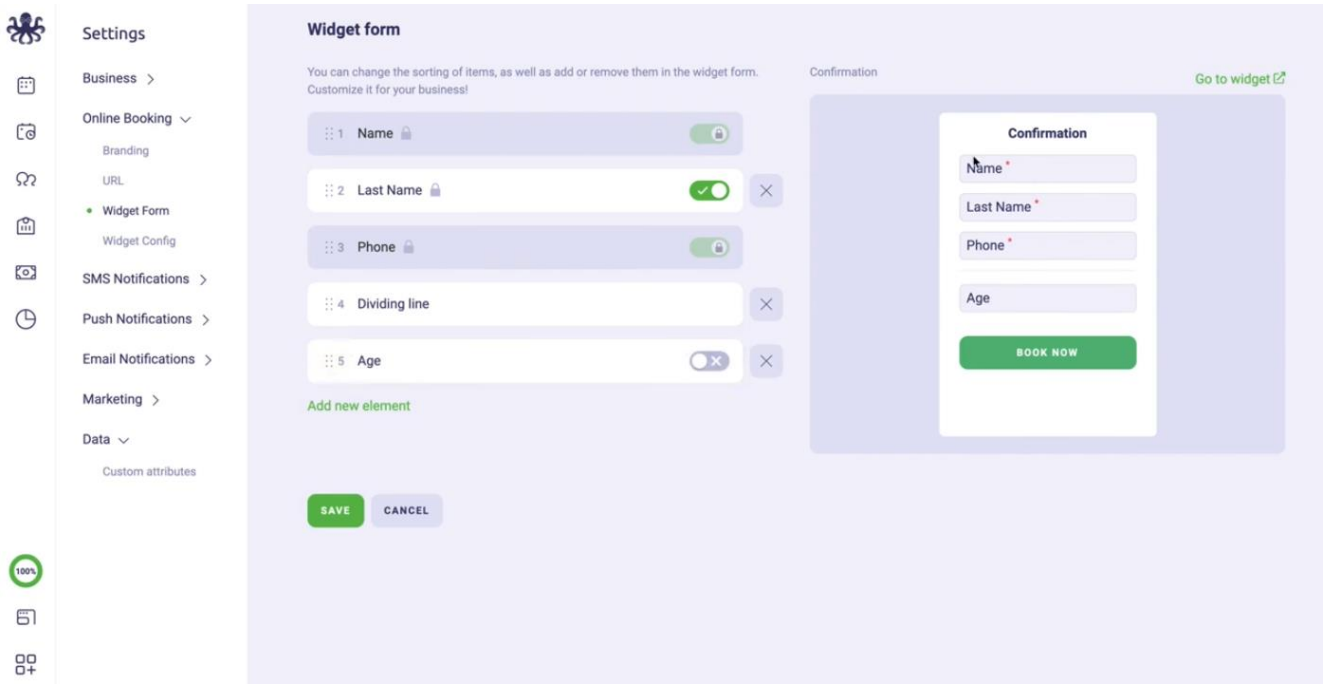

Рисунок 1.3 – Інтерфейс налаштування віджетів «EasyWeek»

Огляд сучасних аналогів застосунків, можна зазначити основних функціонал та призначення, яке в свою чергу використовує CRM-систему та підхід до єдиної об’єднаної бази як клієнтів салонів, так і персоналу.

1.2 Аналіз розроблюваної системи вебзастосунку

Розробка вебзастосунку салону краси для додерження сучасних тенденцій повинна включати у собі інформаційну систему управління, що значно вплине на автоматизацію бізнес-процесів та скорегує регуляцію внутрішніх процесів роботи салону краси для більшої взаємодії роботи з клієнтами та персоналом. Даний застосунок охоплюватиме закриття потреб клієнтів та працівників, створюючи єдину базу роботи, де клієнт отримуватиме потрібні йому послуги в зручному сервісі, зможе оглянути наявний каталог, майстрів, їх роботи та рейтинг, а майстри в свою чергу робочий календар, де зможе корегувати власний графік та мати актуальну інформацію записів, що спрощує також і роботу адміністратора. Адміністратори в свою чергу зможуть виконувати моніторинг роботи працівників,

спрощуватиме ведення керування базою даних клієнтів. А також власників салонів, що в свою чергу економитимуть час на менеджмент та фінансовий облік, бо застосунок охоплюватиме фінансову звітність, рейтингову систему майстрів, облік матеріалів та дохід від бізнесу в одному застосунку.

Система охоплюватиме забезпечення багатокористувацького режиму, матиме кросплатформу для зручної роботи з будь якого браузеру (Chrome, Opera, Firefox та Edge), буде динамічним для підлаштування під мобільні пристрої, доступною у будь-який час.

Таблиця 1.4 – Опис системи, що розробляється

Функції	1) реєстрація та авторизація клієнтів; 2) особистий кабінет клієнта (історія записів); 3) інтерактивний календар для вибору дати та часу; 4) вибір конкретного майстра за рейтингом або послугою; 5) каталог послуг з цінами та описом; 6) система онлайн-оплати (еквайринг); 7) панель адміністратора для управління розкладом; 8) система відгуків та оцінок після надання послуги; 9) модуль складського обліку (залишки лаків, фарб тощо); 10) генерація звітів про доходи за період; 11) аналітика популярності послуг; 12) розрахунок відсоткової ставки зарплати майстрів; 13) управління базою клієнтів (CRM); 14) мультимовність інтерфейсу.
Користувачі	1) власник ; 2) адміністратор; 3) майстер; 4) клієнт.

Кінець таблиці 1.4

Сценарії роботи	1) клієнт відкриває вебзастосунок, авторизується; 2) клієнт обирає необхідну категорію послуг та конкретну процедуру; 3) система пропонує майстрів та доступний час для запису; 4) клієнт підтверджує запис, дані миттєво з'являються в журналі адміністратора та графіку майстра; 5) майстер отримує сповіщення про новий запис і готує робоче місце; 6) після надання послуги адміністратор закриває запис у системі, фіксуючи оплату; 7) система автоматично оновлює фінансову звітність та історію клієнта; 8) адміністратор додає нові послуги або змінює графік роботи майстрів у разі потреби; 9) клієнт залишає відгук про послугу через свій особистий кабінет; 10) адміністратор аналізує щотижневий звіт про доходи та активність клієнтів; 11) власник переглядає аналітику за місяць про прибуток, необхідність оновлення матеріалів та про популярність послуг та рейтинг майстрів.

Для розробки інтерфейсу застосунку (рис. 1.4) використано вебсервіс для створення макетів «Figma», що є дуже популярним серед вебдизайнерів та має широкий функціонал до використання. Інтерфейс має бути простим та зручним для новачків у даній сфері, мати приємні кольори в гамі та мати мультимовність.

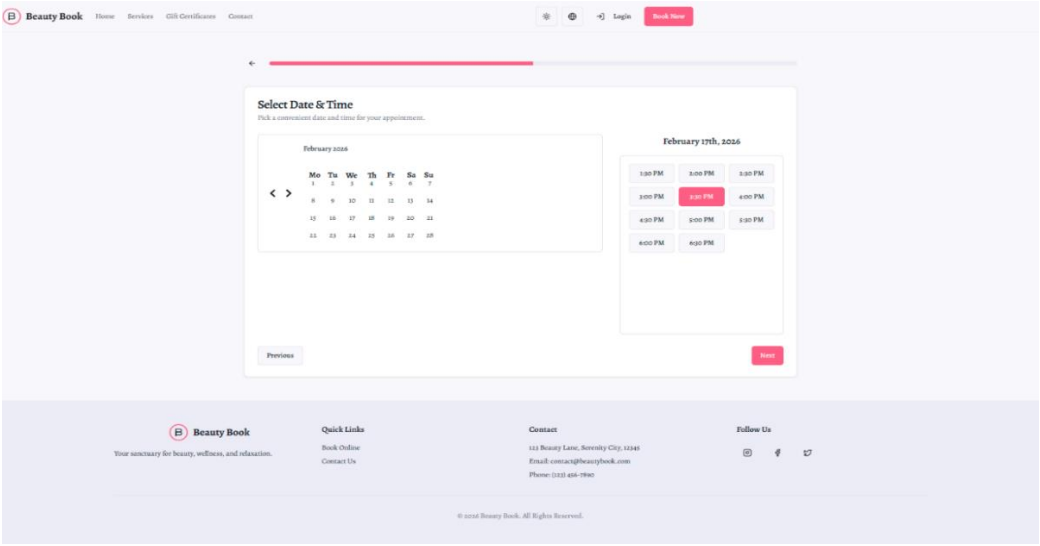

Рисунок 1.4 – Макет інтерфейсу обрання дати запису застосунку

На даному макеті відображено як відбуватиметься запис клієнта, а саме обрання дати та вільного слоту часу. Цей етап буде відбуватися після обрання категорії послуг та майстра, який її виконуватиме.

1.3 Використання CRM-моделі для вебзастосунку з клієнтською базою

На початку багато малих підприємств використовують прості засоби: Excel або Google Sheets для запису інформації про клієнтів, доходи, витрати та графіки, а також Viber, Telegram або інші месенджери для запису та комунікації. Цей спосіб дешевий, зрозумілий і вистачає, коли клієнтів небагато, а процеси ще не потребують складної автоматизації. Наприклад, адміністратор салону може вести таблицю з записами, ручно нагадувати клієнтам про прийом і рахувати виручку вручну. Але з розширенням бізнесу з’являються звичайні проблеми: повторення записів, помилки через ручне введення даних, різні версії таблиць, важкість контролю змін та відсутність швидкого звіту. Обмеження електронних таблиць особливо помітні, коли з ними працює декілька людей одночасно або коли дані потрібно швидко поєднувати з іншими процесами бізнесу.

Сучасні вебзастосунки орієнтовані на керування клієнтоорієнтованими бізнесами потребують в свою чергу інтеграцію інформаційної системи управління, що посідатиме центральне місце розроблюваної системи задля поєднання користувачів: від клієнта до власника. Перш за все, повинна бути реалізація трьох циклів: операційний, аналітичний та колаборативний. Розглянемо окремо кожен із них більш детально.

Операційний цикл повинен бути зорієнтованим на автоматизацію бізнес-процесів, таких як запис клієнта на послугу. Тобто повинне відбуватися бронювання вільного слоту запису клієнта на обрану послугу. Повинна відбуватися швидка обробка звернень, через що використовуватиметься єдина база, де міститиметься потрібна інформація для виконання функціоналу застосунку. Через це відбуватиметься й спрощення ведення обліку.

Аналітичний цикл виконуватиме збір даних про популярність наявних послуг та майстрів, що зможе допомагати в розширенні персоналу за необхідності. Даний

цикл також охоплюватиме, які клієнти найчастіше повертаються (Retention Rate) та на яку середню суму здійснювали оплату послуги, що допоможе у створенні бази лояльності (сертифікати та бонусна система).

Колаборативний цикл здійснює взаємодію між такими категоріями користувачів, як адміністратор та майстер або клієнт та майстер. Прикладом даного циклу є повідомлення про майбутню послугу або зміни із записом, що не потребує особистого повідомлення через дзвінки або смс.

Використати CRM-модель системи у даному вебзастосунку можна використовуючи наступний алгоритм:

– захоплення (capture), де клієнт реєструється та здійснює онлайн-запис, через що система автоматично створює новий об’єкт у базі даних;

– сегментація, допоможе системі автоматично групувати клієнтів на нових та постійних;

– персоналізація, що спростить задачу при повторному записі, де адміністратор бачить, що клієнт любить саме «Манікюр у майстра Олени», і зможе запропонувати персоналізований сервіс;

– утримання для виявлення системою клієнтів, які не приходили понад 30 днів, і запропонує адміністратору надіслати їм нагадування або бонуси;

Для технічної реалізації даної системи підійде 3-tier архітектура, де CRM-модель в свою чергу розподілиться за наступним списком:

– presentation (UI), де інтерфейс застосунку для адміністратора матиме таблиці клієнтів, пошук, фільтри, а клієнт особистий кабінет;

– logic (Application Server) охоплюватиме алгоритми розрахунку програми лояльності, тобто зарахування бонусних балів або сертифікатів, перевірка вільних слотів у базі майстрів;

– data, що є реляційною базою даних, де зберігаються пов’язані таблиці Clients, Orders, Services, Masters.

Даний підхід для розробки дозволить значно зменшити «no-show», через аналітичний підхід та збільшити прибуток, впроваджуючи cross-sell, що допомагає у бізнесі. Таким чином, підвищується швидкість обслуговування, через

автоматизацію багатьох процесів, знижуючи людський фактор на помилку, при записі через адміністратора, а адміністратор в свою чергу матиме швидкий перегляд бази, через що зникає потреба у веденні паперових журналів.

В даному застосунку йде наголошення на конкретні потреби малого бізнесу, що зорієнтований на б’юті сферу обслуговування, матиме легкий та динамічний інтерфейс, та буде поєднувати в собі багатий необхідний функціонал. Таким чином, CRM-система забезпечує клієнтоорієнтований підхід та виконує основний функціонал менеджменту.

Висновки до розділу 1

Перший розділ кваліфікаційної бакалаврської роботи було проведено аналіз предметної області, що охоплює аналіз застосунків аналогів вебзастосунків салонів краси, що мають CRM-систему, що надає змогу визначити переваги та недоліки й винести основний функціонал, що буде втілено в новітньому застосунку. Також, даний аналіз допомагає визначитись в усуненні майбутніх недоліків та перетворити їх в переваги у власному програмному забезпеченні. Було виявлено ключові переваги та критичні недоліки сучасних систем управління бізнесом, що надало змогу сформувати перелік базового та додаткового функціоналу даної системи розробки, який є затребуваним серед кінцевих користувачів (адміністраторів, майстрів та клієнтів). Визначено основну стратегію мінімізації ризиків, перетворюючи виявлені в аналогах помилки (наприклад, перевантаженість інтерфейсу) у конкурентні переваги власного програмного забезпечення.

Через аналіз аналізування застосунків, було обрано основний функціонал вебзастосунку, визначено його архітектуру та майбутній та можливі сценарії роботи, що значно спростить задачу у виконанні. Створено специфікацію вимог програмного забезпечення для подальшої розробки вебзастосунку салону краси. Пророблено роботу з мінімізацією ризиків та шляхи реалізації конкурентоспроможного забезпечення. Розроблено макет однієї з функцій та пророблено роботу з визначенням майбутнього інтерфейсу застосунку.

Розглянуто переваги використання інформаційної системи управління та спрощення функціоналу за допомогою її використання. Розглянуто як вона себе виявлятиме у самому застосунку та чому саме вона є потрібною у даній категорії програмного забезпечення та чому орієнтир впав саме на неї. Дана система допоможе реалізувати оптимізацію функціоналу шляхом автоматизації рутинних операцій (розсилка нагадувань, облік витратних матеріалів) та підвищити якість обслуговування за рахунок персоналізації даних про клієнтів. Інформаційна система управління реалізує бізнес-процеси салону для власника закладу. Було обґрунтовано використання даного підходу через його релевантність для даної категорії програмного забезпечення, де основна орієнтація була поставлена на автоматизацію управління, що зумовлено необхідністю швидкого реагування на запити клієнтів та потребою у централізованому зберіганні інформації, що є ключовим фактором успіху в цифровій трансформації сфери послуг. Даний вид системи широко використовується в багатьох видах застосунках надання послуг, наприклад з продажу товарів, або доставки їжі з кафе чи ресторану, через свій клієнтоорієнтований підхід, що є значною перевагою і для сфери надання послуг краси і значним плюсом як для бізнесу, так і для застосунку.

2 ЗАСТОСУВАННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ ПРИ ПРОЄКТУЮВАННІ ВЕБЗАСТОСУНКУ САЛОНУ КРАСИ

2.1 Дослідження методів створення вебзастосунку

Сучасний підхід розробки вебзастосунків для сфери послуг визначають основні аспекти самої роботи. Однією із головних методологій (SDLC) є прототипування, оскільки традиційні методології, такі як Waterfall, дають поганий результат через застарілість. Каскадна модель представляє собою виконання певної задачі, без умови повернення до неї, що робить її використання недоцільним, адже сфера надання послуг передбачає постійні інтеграції у систему. Через це може бути «зламана» логіка самих записів, через логіку конфліктів, що дуже важко передбачити з першого разу та потребує постійної ітерації.

В свою чергу, ітеративна розробка передбачає своєчасне виявлення помилки в логіці, якщо система зможе пропустити запис до різних майстрів на одну й ту саму процедуру в той самий час, що є доречним до даної сфери. Також, прототипування передбачає створювання на перших етапах розробки low-fidelity макети для розуміння доцільності використання модальних вікон для видображення графіку майстра.

Інтеграція в вебзастосунки надання послуг інформаційної системи управління, таких як MIS (Management Information System) та CRM (Customer Relationship Management) допомагає у вирішенні багатьох бізнес-процесів, зумовлений зокрема на роботу з клієнтами. MIS зорієнтована на обробку та збереження даних однієї сфери послуг (одного салону), використовується для аналізу та подальшої підтримки прийняття управлінських рішень, що в свою чергу допомагає підвищити ефективність та покращити планування. Даний вид інформаційної системи має орієнтованість на внутрішню ефективність, що зумовлює собою автоматичне ведення фінансових звітів та бачення прибутковості тієї, чи іншої категорії послуг [4].

CRM несе в собі управління відносинами з клієнтами, що є дотичним до «клієнт в першу чергу», адже даний вид системи передбачає збереження історії

взаємодії клієнту з салоном. Збереження таких даних відбувається в єдиній картці клієнта, де можна переглянути тенденції повернення до певного майстра, проаналізувати, які послуги були популярними тощо. CR-система є автоматизацією продажів (pipeline) від першого звернення до закриття угоди, що являє собою передбаченого сценарію, де клієнт обирає послугу та отримує її залишаючись задоволеним [5].

UML-моделювання є одним з головних етапів розробки, адже надає візуальне бачення структури роботи застосунку та етапи рішення поставлених на виконання задач. Даний тип моделювання надає змогу до розширення потенціалу розробки, винесення основних класів та їх функції, що полегшують наступні етапи. Даний тип дозволяє розширено розглянути роботу бізнес-логіки та запобігти майбутніх недоцільних рішень.

Проблема більшості вебзастосунків сфери послуг полягають у нечіткому розподілу прав користувачам системи, що спричиняють майбутні проблеми з наданнями прав доступу до певного функціоналу. За допомогою використання Use Case Diagram прецедентів, можна розкрити поділ прав між адміністратором або власником салону та клієнтом, де клієнт в свою чергу не «записується», а проходить через прецеденти «пошук за категорією» та «валідація вільного часу», що зберігає логіку роботи. Адміністратор має права на використання прецедента «редагування розкладу», що спрощує роботу майстрів [6].

В даному аспекту, доцільною є також діаграма діяльності, де є можливість переглянути процес клієнта від «обрання категорії послуги» до «сплати бронювання», де він проходить через підтвердження даного вибору.

База даних для сфери послуг повинна бути приведена до третьої нормальної форми (3NF), що надає гарантію, що при зміні ціни на послугу, вона зміниться лише в одному місці, а не в кожному минулому записі клієнта. Це збереже цілісність даних, що забезпечить неможливість випадкового видалення з системи майстра, якщо у нього на сьогодні є активні запису клієнтів [7].

Відношення знаходиться в третій нормальній формі (3NF), коли воно відповідає умовам першої та другої нормальних форм, а кожен неключовий

атрибут не залежить транзитивно від первинного ключа. У контексті досліджуваної предметної області, це означає, що дані про майстра, клієнта та послугу потрібно розбити на окремі сутності, які пов'язані лише через ідентифікатори (foreign keys). Завдяки виокремленню сутності «Послуги» в окрему таблицю, зміна вартості сервісу здійснюється атомарно (в одному записі). Оскільки транзакційна таблиця «Записи» лише посилається на ідентифікатор послуги, система уникає необхідності масового оновлення історичних даних, що мінімізує обчислювальні витрати та ризик розсинхронізації інформації.

2.2 Використання технологій при розробці вебзастосунку салону краси

Для ефективної роботи салону краси необхідні такі ключові елементи: швидка та зручна взаємодія користувача з системою, наявність різних рівнів доступу та функціоналу для різних ролей (наприклад, адміністратора, майстра, клієнта), можливість запису на послуги, ефективне управління розкладом роботи майстрів, інструменти для ведення бази клієнтів (CRM), аналітичні звіти та можливість інтеграції з іншими сервісами. З огляду на це, обраний стек технологій є методологічно виправданим. SPA-підхід (Single Page Application) є оптимальним для інтерфейсів, де користувач часто взаємодіє з динамічними елементами, такими як календарі, форми для введення даних, історія відвідувань та персональні кабінети. Контейнеризація ж дозволяє створити ізольовані середовища для кожного компонента системи: веб-інтерфейсу, серверної частини (API), фонових обчислень (воркера) та бази даних, що забезпечує стабільність та масштабованість. Дослідження в галузі single-page architecture також свідчать про те, що такий підхід значно полегшує процес створення вебзастосунків з багатим функціоналом та складним інтерфейсом [8].

Для реалізації односторінкового застосунку (SPA) [9] інтерфейсу салону краси були обрані популярні JavaScript-фреймворки – React або Vue.js. Такий вибір є цілком виправданим, оскільки обидва підходи базуються на принципі компонентної архітектури. Згідно з документацією React, інтерфейс формується з окремих, багаторазово використовуваних блоків – компонентів, які можуть бути

вкладені один в одного. Vue.js також пропонує подібний підхід, дозволяючи розбивати користувацький інтерфейс на незалежні та повторно застосовні частини. Крім того, Vue.js робить особливий наголос на реактивній моделі управління станом, що означає автоматичне оновлення відображення при зміні даних. Це є ключовим фактором для вебзастосунку салону краси, адже типові елементи інтерфейсу, такі як картки послуг, форми для запису, календарі, списки замовлень, історія відвідувань та блоки з бонусами, повторюються для різних категорій користувачів: гостей, клієнтів, майстрів та адміністраторів.

У проєктах, побудованих на React та Next.js логічно розглядати як окремий шар, який відповідає за публічну частину системи. Такий підхід є особливо вигідним для сторінок, де важливі SEO-оптимізація, швидкий перший рендер та підтримка кількох мов. Прикладами таких сторінок можуть бути каталоги послуг, розділи з цінами, відгуки чи промо-матеріали. Офіційна документація Next.js [10] підтверджує його можливості в області серверного рендерингу (SSR), генерації статичних сайтів (SSG), використання Metadata API для покращення SEO та зручності шарингу, а також вбудованої підтримки багатомовної маршрутизації (i18n). Таким чином, для внутрішніх кабінетів користувача доцільно дотримуватися класичного SPA-підходу, в той час як для публічних, чутливих до SEO сторінок, варто активно застосовувати функціонал Next.js [11].

Використання TypeScript виступає як розширення JavaScript, що додає сувору типізацію. Це означає, що він допомагає виявляти помилки, пов'язані з типами даних, ще під час написання коду, а не під час виконання. Крім того, TypeScript [12] пропонує покращені інструменти для розробників, які полегшують роботу з великими проєктами. Емпіричні дослідження підтверджують, що використання статичних систем типів для JavaScript-коду дозволяє виявити значну кількість потенційних проблем ще до того, як програма буде запущена. Це особливо цінно для систем, де користувачі записуються на послуги. У таких системах форми містять різноманітну інформацію, таку як дата, час, назва послуги, ім'я майстра, контактні дані та статуси замовлень. Всі ці дані повинні бути коректними та узгодженими як на стороні браузера (фронтенд), так і на стороні

сервера (бекенд). Для розробників, які використовують React, бібліотека Formik спрощує процес роботи зі станом форми, її валідації та надсилання даних. Для розробників на Vue, VeeValidate надає схожі можливості, включаючи відстеження змін у полях форми, збір усіх помилок валідації та можливість проводити валідацію як синхронно, так і асинхронно. Коли ці інструменти використовуються разом із загальноприйнятими методами клієнтської валідації, це призводить до зменшення кількості некоректних запитів, які надходять на сервер, і покращує досвід користувача, надаючи йому чіткий зворотний зв'язок про помилки.

Обрання Node.js слугує основою для розробки основного API. Його архітектура, побудована на асинхронності та подієвій моделі (event-driven), що робить його ідеальним для створення масштабованих мережевих сервісів. Ключовим елементом є цикл подій (event loop), який дозволяє ефективно обробляти операції введення-виведення (I/O) без блокування основного потоку виконання. Це особливо важливо для вебзастосунку сфери надання послуг, де типове навантаження полягає не в складних обчисленнях, а в паралельній обробці численних коротких I/O-операцій. До таких операцій належать: перевірка доступності ресурсів (слотів), реєстрація клієнтів, оновлення статусу замовлень, обробка транзакцій, надсилання сповіщень, взаємодія з системами управління взаємовідносинами з клієнтами (CRM) та запити до баз даних. В контексті цього API, фреймворк Express пропонує мінімалістичний та гнучкий підхід, тоді як NestJS надає більш структуровану, TypeScript-орієнтовану платформу для побудови масштабованих серверних додатків [13].

Серверна частина включає в себе функціонал авторизації та RBAC [14]. Це відповідає побудові на ролях: гість, клієнт, майстер, адміністратор та власник – кожен з них має свої унікальні права доступу та бачить лише ту інформацію, яка йому призначена. RBAC, згідно з рекомендаціями NIST, означає, що контроль доступу здійснюється на основі ролей, які призначаються користувачам, та відповідних їм дозволів. Використання засобів, такі як guards та механізми авторизації, спеціально розроблені для перевірки умов доступу, ролей та списків контролю доступу (ACL) під час обробки запитів. Таким чином, саме серверний

шар відіграє ключову роль у централізованому управлінні тим, хто має право переглядати календар, хто може списувати матеріали, хто отримує доступ до фінансової аналітики, а хто може вносити зміни до довідників послуг чи програм лояльності.

Основною системою управління базами даних (управління) для даного проєкту є PostgreSQL є технічно обґрунтованим вибором. Офіційна документація PostgreSQL [15] підкреслює його статус як надійної об'єктно-реляційної бази даних, яка фокусується на забезпеченні цілісності даних та коректності виконання операцій. Зокрема, детально описані ACID-властивості транзакцій (Atomicity, Consistency, Isolation, Durability), механізми ізоляції для паралельного доступу та механізм write-ahead logging, що гарантує надійність збереження даних. Для інформаційної системи управління салоном краси це є критично важливим, оскільки такі дані, як записи на послуги, інформація про платежі, складські операції, історія відвідувань клієнтів, розрахунок бонусів та фінансові звіти, вимагають абсолютної узгодженості та точності. Крім того, підтримка типів даних json та jsonb надає гнучкість у зберіганні напівструктурованих даних, таких як payload-и від зовнішніх API, службові метадані, параметри маркетингових акцій або журнали інтеграційних процесів.

Для розгортання дана архітектура поєднується з платформою Docker та Docker Compose. Використання контейнеру є дотичним, бо контейнер пакує все необхідне для запуску застосунку в ізольованому середовищі, а Compose дає змогу керуватистеком сервісів, мереж і томів із єдиного YAML-файлу та запускати багатоконтейнерні застосунки. Для системи салону краси це використання надасть змогу вебклієнту, API, Python-воркері PostgreSQL можливість бути описаним як єдина інфраструктурна одиниця зпрогнозованим життєвим циклом. У цьому ж контурі доречно використовувати CI/CD через сервіс GitHub Actions, який офіційно позиціонується як платформа для автоматизації build, test і deployment pipeline. Таким чином, збірка образів, запуск тестів, створення артефактів і контрольований деплой можуть виконуватися автоматично післязмін у репозиторії [16].

2.3 Специфікація вимог до програмного забезпечення вебзастосунку салону краси «Beauty book»

СПЕЦИФІКАЦІЯ ВИМОГ ДЛЯ ВЕБЗАСТОСУНКУ САЛОНУ КРАСИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ УПРАВЛІННЯ

1. Призначення та межі проєкту

1.1 Призначення системи

Вебзастосунок салону краси з інформаційною системою управління призначений для надання клієнтам швидкого та корректного доступу до б’юті-послуг не відвідуючи фізичного салону. Система забезпечує автоматизацію процесу онлайн-запису, перегляд категорій послуг, портфоліо майстрів, управління графіком візитів, а також доступ до програми лояльності та пропозицій.

1.2 Погодження, ухвалені в програмній документації

– специфікація розроблена на основі вимог замовника (власника салону) та стандартів захисту персональних даних (GDPR, Закону України «Про захист персональних даних»);

– ухвалено використання сучасних технологій веброзробки для забезпечення стабільної роботи на різних типах пристроїв (кросплатформність);

– визначено перелік пріоритетних функцій для запуску MVP (мінімально життєздатного продукту), що включає онлайн-бронювання та базу клієнтів.

1.3 Межі проєкту

– проєкт охоплює розробку клієнтської частини (інтерфейс для користувачів), модуля роботи для майстрів та адміністративної панелі для керування процесами роботи салону;

– інтеграція з зовнішніми сервісами сповіщень (Telegram API) для нагадування про візити;

– передбачена робота над адаптивним дизайном застосунку;

– не охоплює розробку або обслуговування внутрішнього обладнання салону (касові апарати, термінали).

2. Загальний опис

2.1 Сфера застосування

Вебзастосунок використовується для автоматизації діяльності салону краси та надання послуг онлайн клієнтам закладу. Застосунок підтримує функції електронного запису на процедури, вибору конкретного спеціаліста, перегляду актуального прайс-листа, а також управління графіком роботи персоналу та базою клієнтів.

2.2 Характеристики користувачів

– клієнти салону – фізичні особи, які використовують застосунок для ознайомлення з переліком послуг, запису на візити, керування власними бронюваннями та участі в програмах лояльності;

– майстри – персонал салону (перукарі, косметологи, майстри манікюру тощо), які використовують систему для перегляду свого розкладу, відмітки виконаних робіт та ведення карток клієнтів;

– адміністратор – персонал, який має доступ до панелі управління для редагування графіків усіх майстрів, додавання нових послуг, формування звітів та моніторингу активності клієнтів;

– власник – фізична особа підприємець, який має доступ до фінансової звітності та керуванням салонів та моніторингом клієнтів.

Рівень підготовки є коли користувачі (клієнти та майстри) мають базові навички роботи з вебзастосунками, а адміністратори мають середній рівень технічної підготовки для роботи з внутрішніми системами управління.

3. Функції системи

3.1 Авторизація та аутентифікація

3.1.1 Опис функції

Дозволяє клієнтам, майстрам та адміністраторам входити в систему за допомогою логіну (email) та пароля.

3.1.2 Вхідна і вихідна інформація

– вхідна: номер email та пароль, одноразовий код підтвердження ;

– вихідна: токен сесії (JWT), роль користувача (клієнт/майстер/адмін), профіль користувача.

3.1.3 Функціональні вимоги

– підтримка підтвердження входу через код на email;

– шифрування паролів у базі даних за допомогою алгоритму bcrypt;

– тимчасове блокування можливості входу після 5 невдалих спроб для захисту взлому профілю.

3.2 Перегляд каталогу послуг та історії візитів

3.2.1 Опис функції

Клієнт може переглядати доступні послуги, ціни та інформацію про свої минулі та майбутні записи до салону.

3.2.2 Вхідна і вихідна інформація

– вхідна: ID користувача, фільтри (категорія послуги, ПІБ майстра, діапазон дат);

– вихідна – список записів (дата, час, назва послуги, майстер, вартість, статус візиту).

3.2.3 Функціональні вимоги

– фільтрація історії за статусом візиту (заплановано, виконано, скасовано);

– можливість клієнту переглянути деталі майбутнього запису;

– оновлення статусу запису в реальному часі.

3.3 Бронювання послуг та онлайн-оплата

3.3.1 Опис функції

Користувач може здійснювати онлайн-запис на вибрані послуги до конкретних майстрів, вносити передоплату або повну оплату за послуги.

3.3.2 Вхідна і вихідна інформація

– вхідна: ID послуги, ID майстра, дата та час візиту, реквізити платіжної картки (у разі оплати), сума;

– вихідна: статус бронювання/транзакції (успіх/помилка), ID запису, електронний чек.

3.3.3 Функціональні вимоги

– автоматична перевірка доступності вибраного часового слота в графіку майстра перед підтвердженням;

– встановлення лімітів на кількість одночасних записів для одного користувача;

– підтвердження онлайн-оплати через систему.

4. Вимоги до інформаційного забезпечення

4.1 Джерела і зміст вхідної інформації

– дані користувачів (ПІБ, контактний номер телефону, історія відвідувань) надаються через інтерфейс реєстрації клієнта;

– інформація про оплати та онлайн-транзакції надходить із підключених платіжних шлюзів (наприклад, LiqPay, Portmone);

– графіки роботи та професійні дані майстрів вносяться адміністратором через внутрішню інформаційну систему управління.

4.2 Нормативно-довідкова інформація

– категорії послуг та актуальний прайс-лист (стрижки, фарбування, косметологія тощо);

– статусів запису (очікує підтвердження, підтверджено, виконано, скасовано);

– база даних майстрів та їхніх кваліфікацій (категорії майстрів, перелік послуг).

4.3 Вимоги до способів організації, збереження та ведення інформації

– дані про клієнтів, записи та фінансові звіти зберігаються в реляційній базі даних (PostgreSQL);

– регулярне автоматичне резервне копіювання всієї бази даних (кожної ночі о 3:00);

– шифрування персональних даних клієнтів та конфіденційної інформації за допомогою стандарту AES-256.

5. Вимоги до технічного забезпечення

– сервери – хмарні сервери з можливістю вертикального та горизонтального масштабування (наприклад, AWS EC2, Google Cloud або DigitalOcean);

– мережа – стабільне з'єднання з мережею інтернет та підтримкою захищеного протоколу HTTPS;

– клієнтські пристрої – персональні комп'ютери, планшети та смартфони з актуальними версіями браузерів (Google Chrome, Opera).

6. Вимоги до програмного забезпечення

6.1 Архітектура програмної системи

– модульна архітектура – розділення системи на окремі сервіси для авторизації користувачів, керування записами (бронювання), обробки платежів та адміністративної панелі;

– взаємодія – використання RESTful API для забезпечення обміну даними між фронтенд-частиною та сервером.

6.2 Системне програмне забезпечення

– ОС серверів – дистрибутиви сімейства Linux (Ubuntu 20.04 LTS або новіші версії);

– вебсервер – використання Nginx як зворотного проксі-сервера та для балансування навантаження.

6.3 Мережне програмне забезпечення

– безпека – використання протоколу HTTPS із діючим SSL/TLS сертифікатом для шифрування трафіку;

– інтерактивність – підтримка миттєвого оновлення статусу записів та графіків майстрів без перезавантаження сторінки.

6.4 Програмне забезпечення ведення інформаційної бази

– управління – реляційна система керування базами даних PostgreSQL версії 13 або новішої;

– керування схемою – використання інструментів автоматичної міграції даних (Flyway або Liquibase).

6.5 Мова і технологія розробки

– frontend: бібліотека React, мова TypeScript для типізації, CSS для швидкої розробки адаптивного дизайну;

– backend – середовище виконання Node.js;

– база даних – мова SQL для взаємодії з PostgreSQL;

– інструменти розгортання – контейнеризація за допомогою Docker та оркестрація через Kubernetes для автоматизації масштабування.

7. Вимоги до зовнішніх інтерфейсів

7.1 Інтерфейс користувача

– адаптивність – дизайн, що коректно відображається на екранах від 320x480 (смартфони) до 1920x1080 (монітори ПК);

– локалізація – інтуїтивно зрозумілий інтерфейс із підтримкою української та англійської мов.

7.2 Апаратний інтерфейс

Не потребує спеціалізованого обладнання; стабільно працює на стандартних персональних комп'ютерах, планшетах та мобільних пристроях.

7.3 Програмний інтерфейс

– інтеграція – REST API для взаємодії з платіжними сервісами та системами розсилок (SMS, месенджери);

– документація – наявність Swagger/OpenAPI специфікації для інтеграції з внутрішніми системами обліку салону.

7.4 Комунікаційний протокол

безпека – використання протоколу HTTPS для всіх мережевих запитів;

реальний час – застосування WebSocket для миттєвого оновлення розкладу майстрів та статусів записів.

8. Властивості програмного забезпечення

8.1 Доступність

Час безперебійної роботи системи становить не менше 99.9% (допускається не більше 8 годин технічного простою на рік).

8.2 Супроводжуваність

– модульність полягає структура коду дозволяє легко оновлювати окремі функції (наприклад, додавання нової категорії послуг) без зупинки всієї системи;

– документованість є повний опис API та коментарі до коду для швидкої передачі проєкту іншим розробникам.

8.3 Переносимість

Повна сумісність із сучасними хмарними платформами, такими як AWS, Azure або Google Cloud.

8.4 Продуктивність

– швидкість – час відповіді сервера не перевищує 1 секунди при навантаженні до 1000 одночасних користувачів;

– масштабність – здатність обробляти до 10 000 операцій бронювання або запитів на перегляд розкладу за годину.

8.5 Надійність

– відновлення – автоматичне відновлення працездатності системи після програмного збою протягом 5 хвилин;

– резервування – автоматичне щоденне створення копій бази даних.

8.6 Безпека

– захист – впроваджено механізми захисту від SQL-ін’єкцій, XSS та CSRF атак;

– шифрування – використання стандарту AES-256 для захисту персональних даних клієнтів;

– аудит – проведення регулярних перевірок системи на вразливості.

9. Інші вимоги

– логування – ведення журналу всіх дій адміністраторів та майстрів для можливості аудиту змін у розкладі;

– конфіденційність – повна відповідність вимогам GDPR та законодавству про захист персональних даних;

– масштабованість – архітектурна можливість розширення функціоналу, наприклад, для впровадження системи онлайн-продажу косметичних товарів (E-commerce модуль).

Висновки до розділу 2

Другий розділ кваліфікаційної бакалаврської роботи було призначено аналізу та методологічному обґрунтуванню розробки, що надало змогу визначити основні тенденції сучасного вебзастосунку сфери надання послуг. Зроблено висновок стосовно використання методології прототипування, її переваги в даному контексті та основні переваги, що сприятиме забезпеченню гнучкості системи. Визначено основну роль та необхідність використання інформаційних систем управління MIS та CRM для забезпечення ефективності бізнесу. Таким чином зазначено, що MIS забезпечує автоматизацію фінансової звітності, що спрощує підтримку управлінських рішень бізнесу. CRM вже в свою чергу робить основну тенденцію на клієнтоорієнтований підхід системи взаємодії салону та клієнту, а також ефективність проведення маркетингових рішень. Було обрано потенційний стек технологій, що вирішуватимуть основні задачі вебзастосунку та забезпечитимуть оптимізацію інтерактивних систем.

Поставлено перевагу використання SPA (React/Vue) та TypeScript для забезпечення швидкого відгуку інтерфейсу та високу надійність коду завдяки статичній типізації, що критично для складних форм запису, які можуть бути наявні при роботі самого салону. Для розробки бекенд частини запропоновано Node.js (NestJS) та PostgreSQL, використання подієвої моделі (Event Loop) дозволить ефективно обробляти велику кількістькоротких I/O-операцій (перевірка слотів, транзакції тощо). PostgreSQL в свою чергу гарантує цілісність даних (ACID) та гнучкість завдяки підтримці JSON. Використання RBAC для реалізації контролю доступу на основі стандартів NIST забезпечуватиме чітке розмежування прав між клієнтом, майстром таадміністратором та власником, що гарантуватиме безпеку конфіденційних даних. Впровадження CI/CD через GitHub Actions автоматизує

процеси тестування та деплою, знижуючи ризик помилок при ручному тестуванні вебзастосунку.

Після проведення попередніх кроків, була розроблена специфікація вимог до програмного застосунку проєкту «Beauty book», який розробляється, що підтверджує готовність до реалізації MVP, що та містить в собі аспекти кросплатформності, адаптивність, високу доступність та безпеку даних (AES-256, HTTPS). Розглянуто значний потенціал у розробці даного вебзастосунку, за допомогою використання сучасних підходів та інструментарію, який цьому сприятиме.

Приділено UML-моделюванню як інструменту відображення структури системи та бізнес-процесів. Використання діаграм прецедентів та діяльності дозволяє чітко визначити ролі користувачів, розмежувати їхні права доступу та змоделювати сценарії взаємодії із системою, що зменшує ризик логічних помилок у подальшій реалізації. У контексті проєктування бази даних доведено необхідність приведення її до третьої нормальної форми , що гарантує цілісність, узгодженість та ефективність обробки даних. Такий підхід мінімізує дублювання інформації, спрощує оновлення даних та знижує ризик виникнення аномалій.

3 ПРОЄКТУВАННЯ ТА ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙВ РОЗРОБЦІ ВЕБЗАСТОСУНКУ

3.1 Проєктування вебзастосунку за допомогою розробки UML-діаграм

Для створення конкурентоспроможного застосунку, потрібно візуалізувати нагальний функціонал та розробити концепцію роботи, проаналізувавши аналоги застосунків та з’ясувавши, які проблеми він вирішує. Таким чином, на даному етапі проєктування ПЗ, допомагає використання UML-діаграми, що мають широкий спектр типів та покривають відображення багатьох етапів розробки.

UML-діаграми поділяються на три типи: структурні, поведінкові та поведінкові взаємодії. Деякі з них є тимчасовим рішенням для відображення концепції та можливої комунікації клієнта з системою, а також вони допомагають у розумінні замовником етапи розвитку системи, що розробляється. Розглянемо детальніше кожен з типів діаграм.

Спершу розглянемо структурні види діаграм, до яких відносяться:

– діаграма класів;
– діаграма компонентів;
– діаграма об’єктів;
– діаграма композитних структур;
– діаграма розгортання;
– діаграма пакетів.

Структурні типи діаграм, здебільшого, відображають архітектуру самого застосунку, які класи та компоненти він має, як розгортається проєкти та які пакети наявні. Це робиться в більшій мірі для самих розробників, щоб розуміти з чим доведеться працювати та як саме в подальшому розвивати функціонал, маючи основні відомості про застосунок, а також допомагає новим членам команди інтегруватися в новий проєкт.

Типи поведінкових діаграм:

– діаграма активностей;
– діаграма використання;

- діаграма кінцевих автоматів.

Поведінкові діаграми розкривають поведінку програмного забезпечення згідно з вимогами замовника або клієнта. Вони допомагають при автоматизації частих процесів та відображають реакцію системи на внесені рішення. Це спрощує передбачення можливих проблем та запобігає ним у подальшій розробці.

До поведінкових діаграм взаємодії відносяться наступні:

- діаграма послідовностей;
- діаграма схем взаємодії;
- діаграма комунікацій.

Даний тип діаграми відповідає за відображення поведінки програмного забезпечення та його внутрішніх та зовнішніх протоколів, формує ієрархію системи та використовує графи, для детальнішого розгляду проблеми. Вони є популярними у використанні, але не завжди доречними, адже здебільшого мають тимчасове значення.

3.1.1 Діаграма використання

Діаграма використання (або діаграма прецедентів) – це тип діаграм, що відображають функціональні вимоги до системи та демонструє взаємодію користувача із функціоналом системи. Таким чином, охоплюється основний функціонал застосунку без використання детальної документації та звернення до великої кількості коду. Вони надають змогу побачити сценарій використання, що піде у відповідь клієнту системи [17].

Короткий сценарій використання:

Клієнт заходить на головну сторінку салону краси, ознайомлюється з інформацією на сайті, переглядає послуги та портфоліо майстрів. Він може шукати потрібну йому послугу та майстра та ознайомитись із деталями за потрібним пошуковим запитом.

Головний (успішний) сценарій використання:

Клієнт салону вперше знайомиться із застосунком та має бажання переглянути інформацію про салон краси та ознайомитись із послугами, який салон

в свою чергу надає. Даний тип користувачів переходить у розділ послуг, обирає потрібну йому послугу , обирає майстра, обирає дату та час, на який потрібно записатися. Потім користувач вводить данні для передоплати обраної послуги. Після цього в профілі клієнта з'являється інформація про майбутній запис (рис. 3.1).

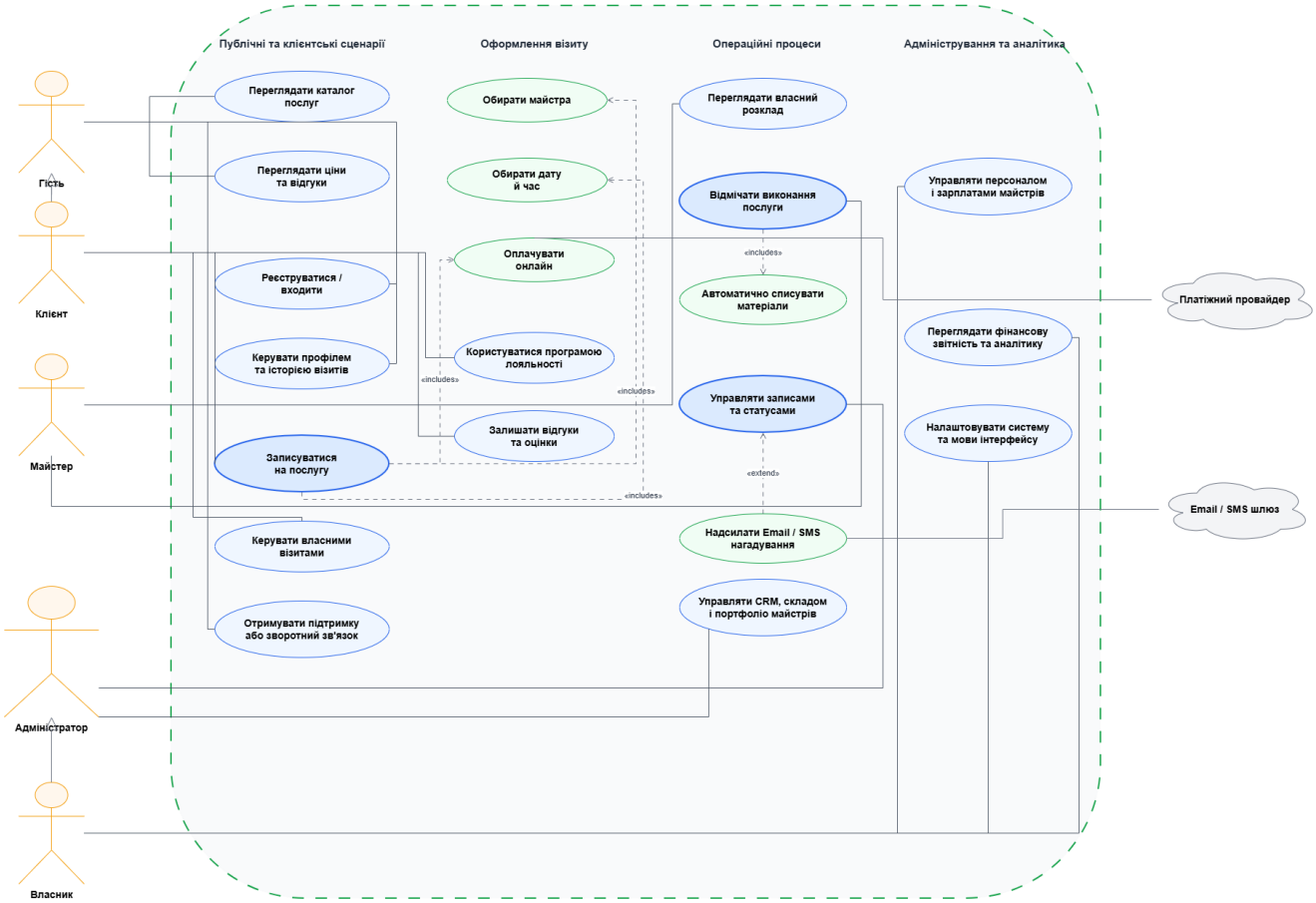

Рисунок 3.1 – Діаграма використання вебзастосунку салону краси

На діаграмі використання можна побачити наступних акторів: гість, клієнт, майстер, адміністратор та власник. Всі вони є користувачами системи та взаємодіють із нею. Як зазначено раніше, клієнт має можливість створити запис на послугу та отримати зворотній зв'язок – інформацію про запис у профілі. Актор зазначений гостем без реєстрації не може здійснити запис, поки не зареєструється, але може отримати підтримку.

Майстер має можливість відмітити виконання послуги, після виконання послуги. Після чого, матеріали, що використовуються під час цієї процедури спишуться автоматично із системи.

Адміністратор може керувати портфоліо майстрів та їх розкладом. Має доступ до управління записом та його статусом, або надсилати смс-нагадування про майбутній запис клієнту, щоб він точно про нього не забув.

Власник салону краси має можливість управляти персоналом та їх заробітною платою, а також переглядати фінансову звітність за місяць, де буде розглянута повна система нарахування заробітної платні та витрат. Таким чином формується й звітність аналітики – які послуги мають популярність у салоні і т.д.

Таблиця 3.1 – Повна форма сценарію використання запису клієнта

Usecase section	**Comment**
Use Case Name	Вести прийом клієнта та закривати візит
Scope	System
Level	User-goal
Primary Actor	Майстер салону краси
Stakeholders and interests	− майстер – хоче мати актуальну інформацію про послуги, матеріали та оплату; − клієнт – очікує якісну послугу, прозорий розрахунок та чек; − адміністратор – контролює дотримання стандартів і облік матеріалів.
Preconditions	− існує підтверджений запис зі статусом Confirmed; − майстер авторизований у кабінеті та має доступ до модуля прийому; − необхідні матеріали відмічені як доступні на складі.
Success guarantee	− візит позначено як виконаний (Done); − оплата проведена та зафіксована у системі платежів; − списання матеріалів відображене в інвентаризації; − клієнту надіслано електронний чек і рекомендації.
Main Success Scenario	1) клієнт прибуває та повідомляє про запис; 2) майстер авторизується; 3) обирає потрібний запис у списку прийомів; 4) система показує деталі послуг та примітки; 5) майстер фіксує початок візиту;

Кінець таблиці 3.1

	6) відмічає виконані або додає додаткові послуги. 7) система перераховує тривалість і вартість. 8) майстер фіксує використані матеріали. 9) система списує матеріали зі складу. 10) майстер ініціює оплату. 11) клієнт проводить оплату, статус → Paid; 12) система змінює статус на Done, генерує чек і надсилає рекомендації.
Extensions	– додаткова послуга – система перевіряє час і оновлює заявку; – запізнення > 15 хв – система пропонує скоротити або перенести; – матеріалу немає – система пропонує аналог, повідомляє адміністратора; – платіжний шлюз недоступний – повтор/інший спосіб; у разі збою створюється відкладений рахунок; – оплата сертифікатом – списується номінал, розраховується доплата.
Special Requirements	– інтерфейс адаптований для планшетів; – підтвердження оплати онлайн ≤30 секунд у 90% випадків; – аудиторський журнал змін по візиту і списанню матеріалів.
Technology and Data Variations List	– оплата – готівка, картка (POS), онлайн-еквайринг; – списання матеріалів – шаблони або ручний вибір; – чек: друк або email/SMS; – робота – вебверсія або планшетна PWA.
Frequency of Occurrence	Для кожного завершеного візиту; у середньому 15–30 разів на день для популярних майстрів.
Miscellaneous	– стандартизований шаблон рекомендацій; – процедура повторного списання при частковому поверненні коштів; – політика зберігання електронних чеків для податкової звітності.

За даним сценарієм, можемо зрозуміти, що майстер, як завершаючий етап надання послуги користувачу. Він виконує послугу, після чого вона є завершеною

та у звітності додалася заробітна платня за її виконання. Також виконання даного сценарію можна побачити на діаграмі ведення цього прийому (рис. 3.2).

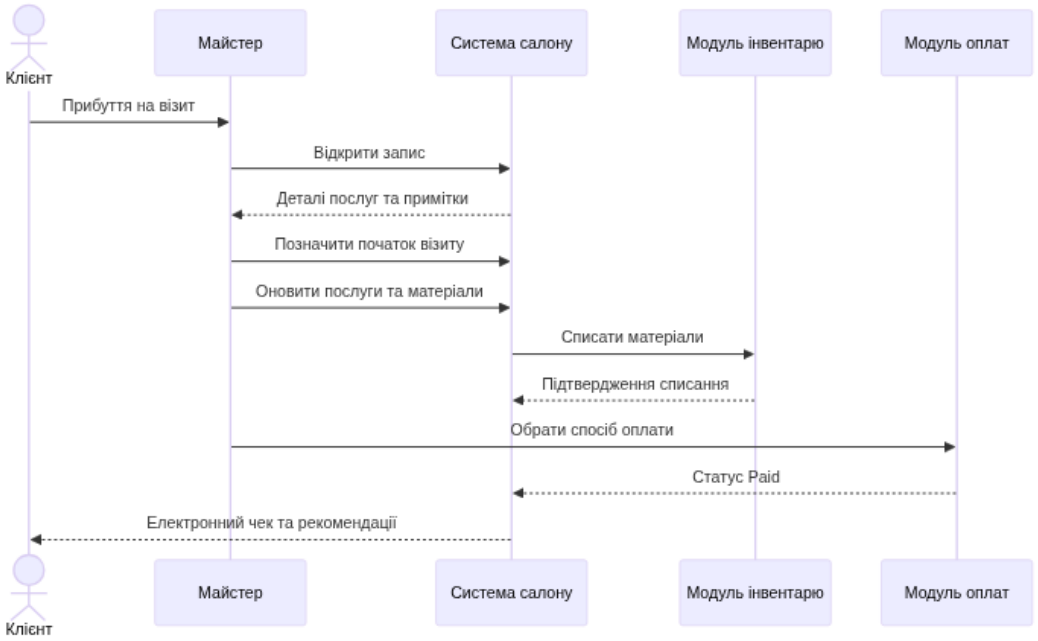

Рисунок 3.2 – Діаграма ведення прийому

Таким чином, розглянуто основний функціонал та головний сценарій використання застосунку, що надає система у відповідь на запит та яким чином це відбувається. Надані діаграми підтверджують сценарій використання.

3.1.2 Діаграма класів

Діаграма класів застосовується для відображення принципу об’єктно-орієнтованого моделювання системи, що розробляється (рис. 3.3). Даний тип діаграм демонструє класи та їх зв’язки у застосунку, яке місце вони посідають. Класи в свою чергу мають назву та низку атрибутів, що допомагає з’ясувати, за що саме він відповідає та як працює.

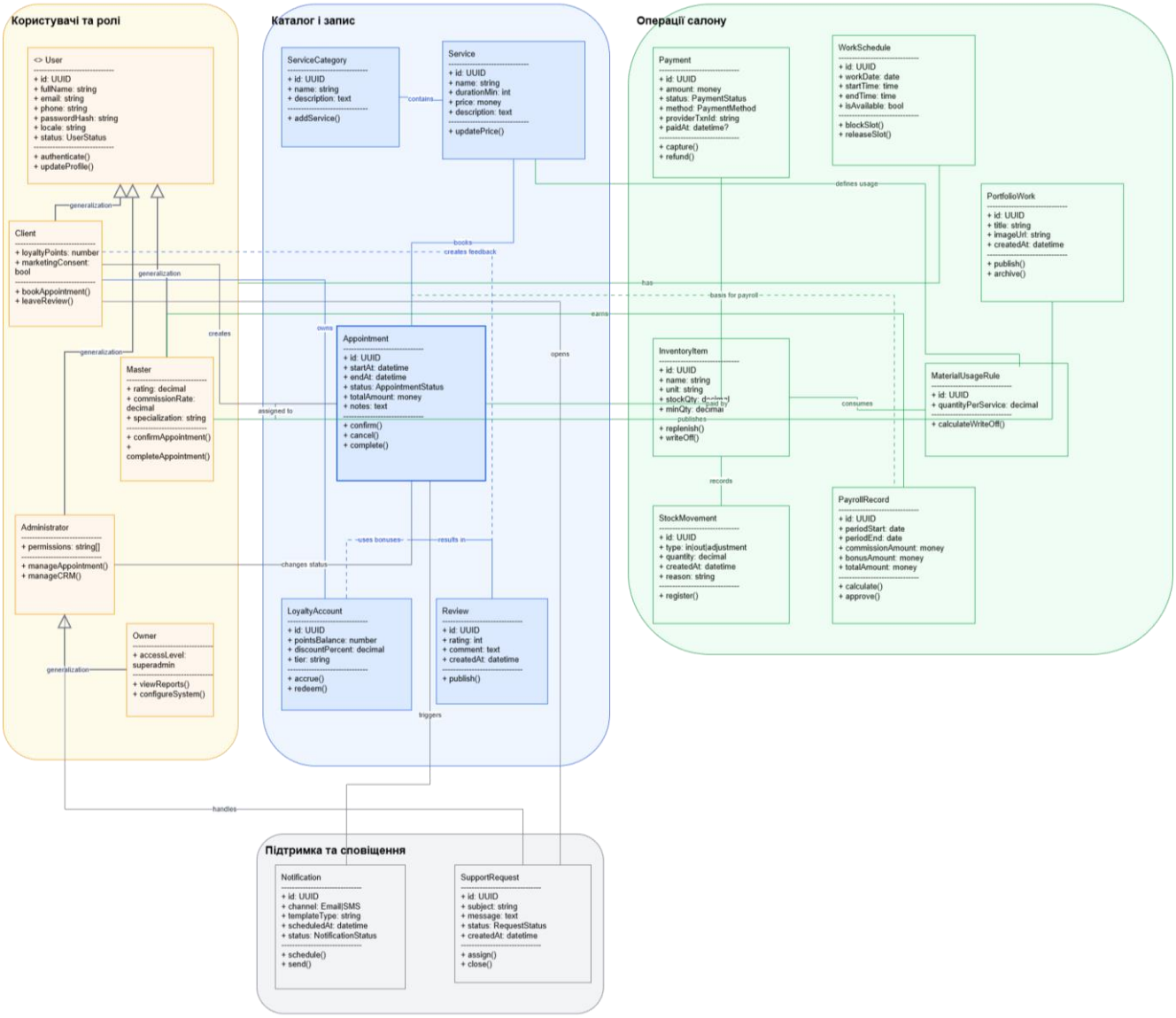

Рисунок 3.3 – Діаграма класів

На даній діаграмі можна спостерігати поділення класів та їх категорій. Клас користувачів має підкласи клієнту, майстра, адміністратора та власника. Усі користувачі мають атрибути ім’я, даних телефону та електронної пошти, що робиться для зворотного зв’язку. Клієнт може також мати сертифікат, що надає йому бонуси, може переглядати статус свого запису. Майстри поділяються за спеціалізацією, мають власний рейтинг та комісійні за пророблену роботу. Адміністратор має можливості менеджера системи, а власник в свою чергу є суперадміном, що може генерувати звітність системи.

Таблиця 3.2 – Класи та їх характеристики

Клас	**Компоненти (атрибути)**	**Методи**	**Призначення класу**
User	id, fullName, email, phone, passwordHash, locale, status	authenticate(), updateProfile()	Базовий клас для зберігання облікових даних та профілю користувача.
Client	loyaltyPoints, marketingConsent	bookAppointment(), leaveReview()	Клас клієнта, що дозволяє здійснювати записи та керувати програмою лояльності.
Master	rating, commissionRate, specialization	confirmAppointment(), completeAppointment()	Клас майстра, що містить дані про кваліфікацію та керує статусом виконання робіт.
Administrator	permissions	manageAppointment(), manageCRM()	Клас адміністратора для операційного керування записуми та клієнтською базою.
Owner	accessLevel	viewReports(), configureSystem()	Клас власника з повним доступом до звітів та налаштувань системи.
ServiceCategory	id, name, description	addService()	Класифікатор послуг за категоріями.
Service	id, name, durationMin, price, description	updatePrice()	Опис конкретної послуги салону з її параметрами.
Appointment	id, startAt, endAt, status, totalAmount, notes	confirm(), cancel(), complete()	Центральний клас запису на послугу, що координує взаємодію клієнта та майстра.
Payment	id, amount, status, method, providerTxnId, paidAt	capture(), refund()	Фінансовий клас для проведення оплат за надані послуги.
LoyaltyAccount	id, pointsBalance, discountPercent, tier	accrue(), redeem()	Управління бонусними балами та знижками конкретного клієнта.
Review	id, rating, comment, createdAt	publish()	Клас для зберігання та публікації відгуків клієнтів.
WorkSchedule	id, workDate, startTime, endTime, isAvailable	blockSlot(), releaseSlot()	Керування робочим часом та доступністю майстрів.
PortfolioWork	id, title, imageUrl, createdAt	publish(), archive()	Зберігання прикладів робіт майстрів для візуального портфоліо.
InventoryItem	id, name, unit, stockQty, minQty	replenish(), writeOff()	Облік витратних матеріалів та залишків на складі.
StockMovement	id, type, quantity, createdAt, reason	register()	Історія руху товарів (надходження, списання, коригування).

Кінець таблиці 3.2

MaterialUsageRule	id, quantityPerService	calculateWriteOff()	Правила списання матеріалів залежно від обраної послуги.
PayrollRecord	id, periodStart, periodEnd, commissionAmount, bonusAmount, totalAmount	calculate(), approve()	Розрахунок заробітної плати та комісійних виплат персоналу.
Notification	id, channel, templateType, scheduledAt, status	schedule(), send()	Система сповіщень (Email/SMS) за розкладом.
SupportRequest	id, subject, message, status, createdAt	assign(), close()	Обробка запитів до служби підтримки від користувачів.

Таким чином, можна зрозуміти концепцію самої системи та яким чином вона буде працювати. Від користувачів до запису та системних операцій салону, що покриватиме основні його потреби. Можна зазначити, що даний тип системи забезпечить економію через самостійне автоматичне ведення бухгалтерського обліку, що додержується принципу інформаційної системи управління салону.

3.1.3 Діаграма станів та переходів

Діаграми станів та переходів, або State-Transition Diagrams відображають життєвий цикл програмного забезпечення, його системи або об’єкта. Тобто, відбувається візуалізація підстави, за якої може змінюватися стан та як воно впливає, наприклад очікування дії та її виконання, зміна стану. За допомогою даних діаграм відбувається опис складних процесів, що спрощує комунікацію між розробниками та замовником ПЗ [18].

Розглянемо три варіанти станів та переходів вебзастосунку салону краси:

1) життєвий цикл запису клієнта (рис. 3.4);
2) повний шлях клієнтського візиту (рис. 3.5);
3) операційний цикл салону (рис.3.6).

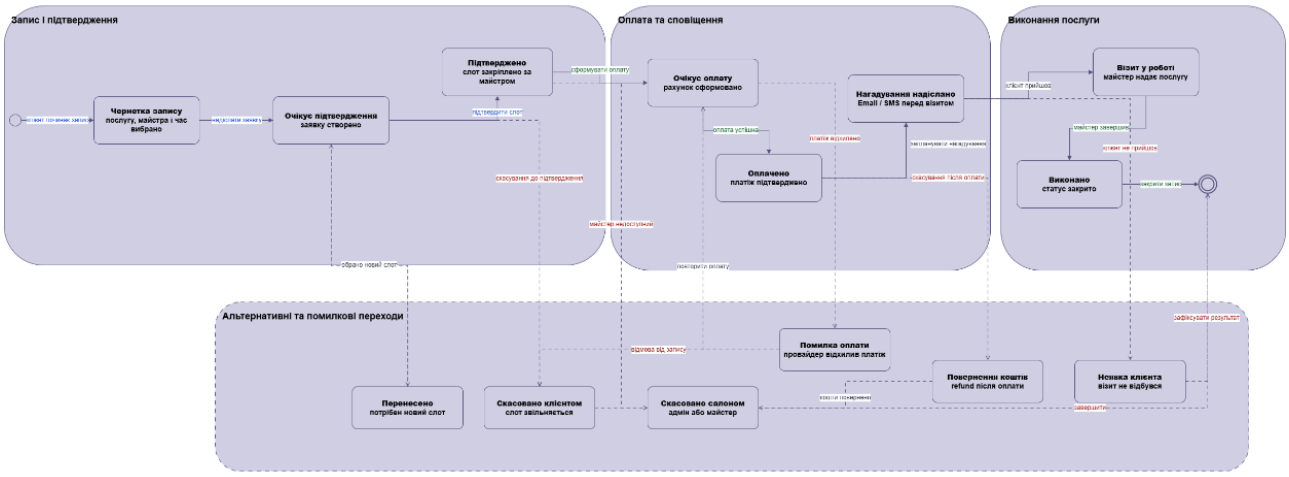

Рисунок 3.4 – Діаграма станів та переходів життєвого циклу запису клієнта

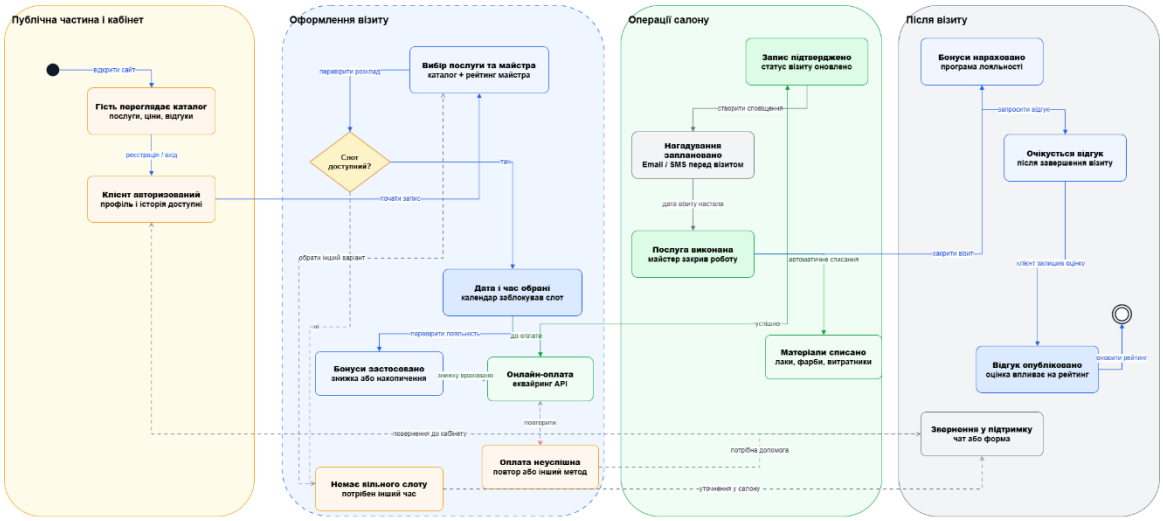

Рисунок 3.5 – Діаграма станів та переходів повного шляху клієнтського візиту

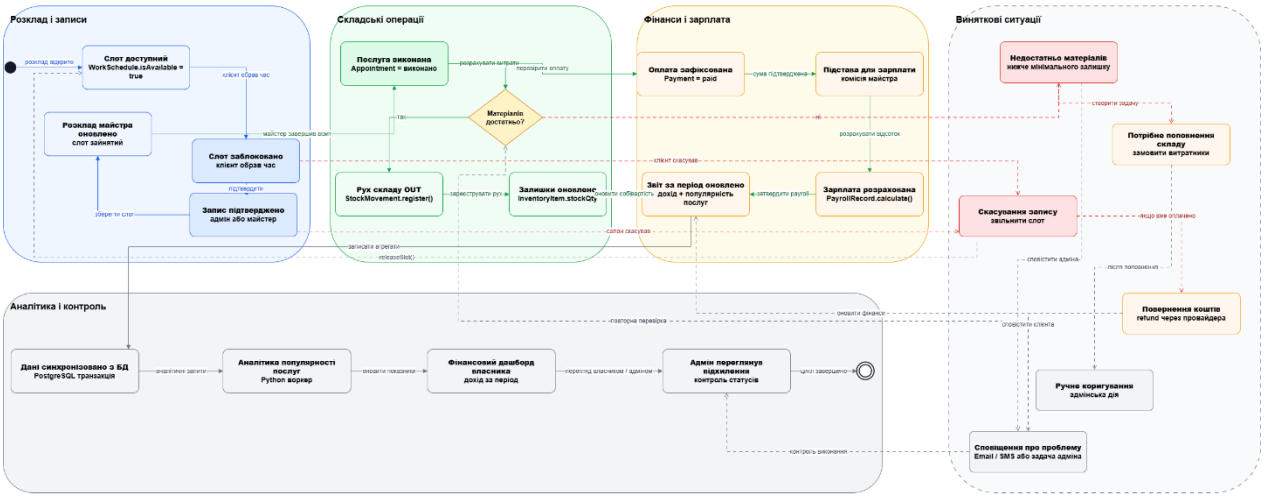

Рисунок 3.6 – Діаграма станів та переходів операційного циклу салону

Можна зазначити, що даний спосіб моделювання надає можливість огляду повного циклу життя системи та відобразити її поведінку при відпрацюванні тієї чи іншої дії, що значно спрощує можливі помилки при розробці та покращує комунікативні навички в команді. За допомогою даних діаграм розглянуто умови, за яких відбувається зміна стану об’єкту та до чого це може призвести.

3.1.4 Діаграма компонентів

Діаграма компонентів є одним з головних ключових моментів розробки. В свою чергу, вона відображає складові програмної системи: які бібліотеки містить в собі проєкт, з яких модулів складається, які файли та компоненти й їх взаємодію. Це допомагає зрозуміти архітектуру застосунку, визначити певні залежності та зробити план розгортання проєкту, розглянувши його складові. Розглянемо діаграму компонентів вебзастосунку салону краси та яке місце відіграє в ньому інформаційна система управління та які її прояви з боку архітектурних рішень (рис. 3.7).

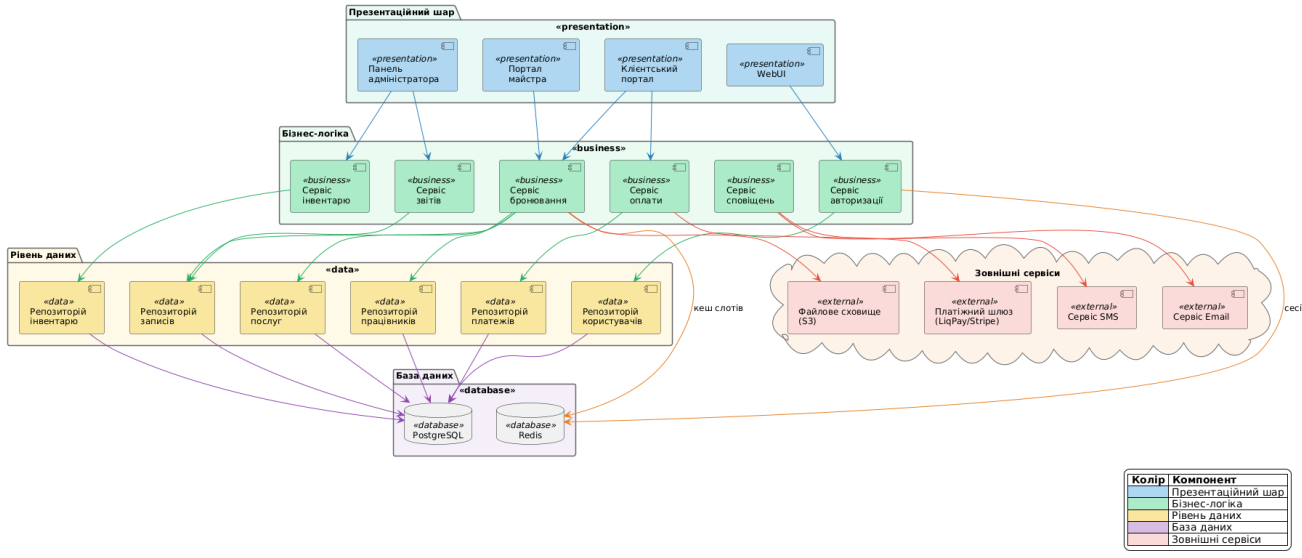

Рисунок 3.7 – Діаграма компонентів застосунку

На даній діаграмі продемонстровано, що компоненти діляться на декілька шарів:

1) презентаційний;
2) бізнес-логіка;

3) рівень даних;

4) база даних;

5) зовнішні сервіси.

Кожен з шарів зображає певний аспект функціоналу. Презентаційний шар вбирає в себе всю клієнтську частину застосунку (UI/UX), для взаємодії користувача із системою. Тобто, робочу панель адміністратора, де він може керувати розкладом, або сторона майстра, де він може його переглядати. Клієнти в свою чергу можуть здійснювати запис на категорію послуг та обраного майстра (за наявності вільного слоту запису).

Бізнес-логіка включає в себе роботу сервісів, що виконують основний функціонал застосунку: сервіс інвентарю, звітів, бронювання, оплати, сповіщень та авторизації, що є головним чинником збереження даних клієнта. В свою чергу, відбувається взаємодія з даними та їх збереженням: наявність слота, матеріалів тощо.

Таким чином зазначимо як відбувається робота застосунку, а саме шляхи взаємодії компонентів між собою. Клієнт бронює та сплачує за послугу – відбувається взаємодія клієнтського та бізнес шарів. З шаром бізнес-логіки працюють зовнішні сервіси – платіжний шлюз або смс-повідомлення для нагадування про запис. Також, бізнес-логіка взаємодіє з шаром даних та базою даних, для збереження наданих вхідних даних про послугу.

3.1.5 Діаграма пакетів

Діаграма пакетів (рис. 3.8) використовується для візуалізації та покращення розуміння структури системи, відображення залежностей та керування просторами імен, задля збереження цілісності проєкту, використовуючи одне й те саме ім’я в різних пакетах. Також це надає змогу спростити архітектуру, через розбиття коду на логічні модулі, що допомагає зробити структуру проєкту організованою та зручною в розумінні, а також надає змогу швидко орієнтуватися в проєкті та знаходити помилки та вразливості.

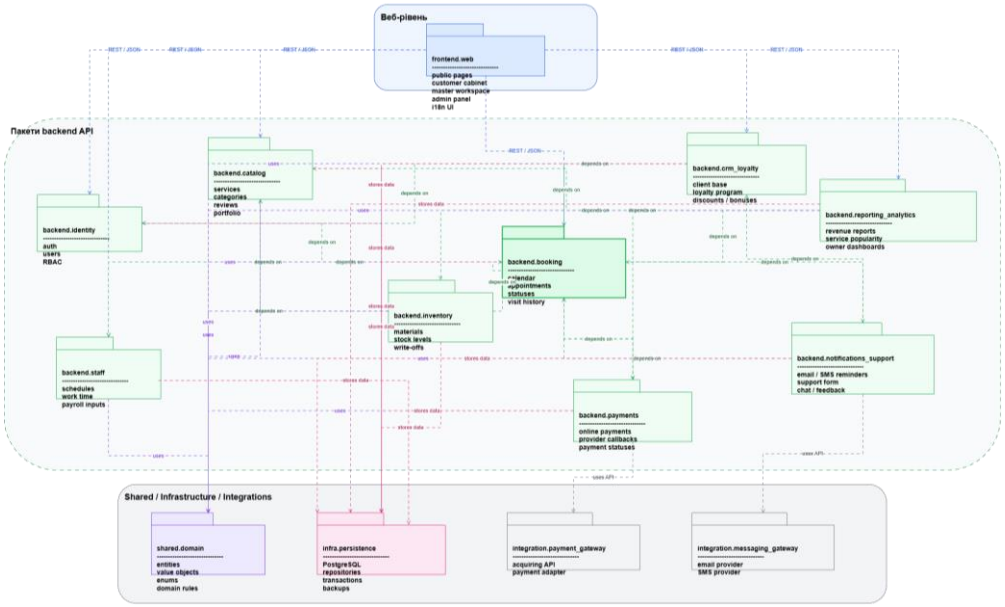

Рисунок 3.8 – Діаграма пакетів вебзастосунку салону краси

1. Веб-рівень – відповідає за інтерфейс користувача та взаємодію з клієнтом:

1) frontend.web:

– public pages (публічні сторінки);

– customer cabinet (кабінет клієнта);

– master/admin panels (панелі майстрів та адміністратора).

2. Рівень Backend API – бізнес-логіка системи, розбита на доменні підсистеми (мікросервіси або модулі моноліту):

1) backend.identity:

– auth (автентифікація);

– users (користувачі);

– RBAC (керування доступом на основі ролей).

2) backend.catalog:

– services (послуги);

– categories (категорії);

– reviews (відгуки);

– portfolio (портфоліо робіт).

3) backend.staff:

– schedules (розклади роботи);

– work time (робочий час);

– payroll inputs (дані для розрахунку заробітної плати).

4) backend.inventory:

– materials (матеріали/товари);

– stock levels (рівні запасів);

– write-offs (списання).

5) backend.booking:

– calendar (календар);

– appointments (записи/візити);

– statuses (статуси записів);

– visit history (історія візитів).

6) backend.payments:

– online payments (онлайн-платежі);

– provider callbacks (зворотні виклики від платіжних провайдерів);

– payment statuses (статуси платежів).

7) backend.crm_loyalty:

– client base (клієнтська база);

– loyalty program (програма лояльності);

– discounts / bonuses (знижки та бонуси).

8) backend.reporting_analytics:

– revenue reports (звіти про доходи);

– service popularity (популярність послуг);

– owner dashboards (дашборди для власника бізнесу).

9) backend.notifications_support:

– email / SMS reminders (нагадування через Email/SMS);

– support form (форма підтримки);

– chat / feedback (чат та зворотний зв'язок).

3. Рівень спільного використання, інфраструктури та інтеграцій – нижній рівень, який забезпечує збереження даних, системні утиліти та зв'язок із зовнішніми сервісами.

1) shared.domain:

– entities (сутності);
– value objects (об'єкти-значення);
– enums (переліки);
– domain rules (доменні правила).

2) infra.persistence:
– PostgreSQL (налаштування/драйвер БД);
– repositories (репозиторії для доступу до даних);
– transactions (керування транзакціями);
– backups (резервне копіювання).

3) integration.payment_gateway:
– acquiring API (API еквайрингу);
– payment adapter (адаптер платежів).

4) integration.messaging_gateway:
– email provider (провайдер електронної пошти);
– SMS provider (SMS-провайдер).

3.2 Огляд задіяних технологій розробки вебзастосунку

Для розробки вебзастосунку салону краси з інформаційною системою управління обрано та задіяно відповідний стек технологій, для отримання кращого програмного рішення:

– мови програмування: JavaScript / TypeScript, Python, SQL;
– frontend: бібліотека React, мова TypeScript для типізації;
– backend – середовище виконання Node.js;
– база даних – PostgreSQL;
– інструменти розгортання – контейнеризація за допомогою Docker та оркестрація через Kubernetes для автоматизації масштабування.

Використання даного стеку технологій витікає з огляду сучасного ринку технологій веброзробки, що доведено аналізом даної області програмування. Використання таких сучасних мов програмування, як JavaScript, TypeScript, Python та SQL надають можливість використання фреймворків та бібліотек для рішення

поставлених завдань проєкту спрямоване на високий результат. Також зазначимо використання інформаційної системи управління, за допомогою використання CRM, що додержує принцип використання єдиної бази управління бізнес-процесами, та відображає все в єдиній системі, що надає змогу автоматизації менеджменту та є гарним рішенням для вебзастосунку салону краси, адже відбувається взаємодія клієнта з представниками надання послуг даної сфери.

3.2.1 Використання технологій front-end розробки

Використання таких сучасних мов програмування як JavaScript та TypeScript для розробки користувацької частини застосунку надає великий набір інструментарію для отримання бажаного результату як замовника, так і розробників. За використання потужної бібліотеки React від компанії Meta, можна скористатися можливістю концепції компонентів, які потім можна буде використовувати повторно в інтерфейсі.

Оскільки вебзастосунок салону краси матиме значний обсяг вхідних даних, з’являється необхідність у використанні менеджеру форм, який надаватиме їх перевірку та спростить валідацію. Таким інструментом є Formik, який може керувати станом форми, перевіряє введені дані за допомогою валідації та використовує обробку сабмітів.

Інструменти Vue.js в свою чергу має двостороннє зв’язування даних та має схожий аналог Vee-Validate та є інтегрованою в Vue.js, може працювати зі звичайними та кастомними інпутами та мають вбудовані правила перевірки, на відміну від Formik, що потребує підключення зовнішніх бібліотек таких як Yup.

React має можливість Virtual DOM, за допомогою чого створює віртуаьну копію сторінки, перевіряє зміни в сторінці до оновлення, та виправляє тільки те, що було замінено, що пришвидшує та мінімізує оновлення та не вантажить зайвий процес. Таким чином можна зазначити, що використання React зручним для використання реалізації користувацького інтерфейсу, що спрощує мінімізацію багів на клієнтській частині інтерфейсу.

3.2.2 Використання технологій back-end розробки

Технології back-end розробки наповнені потужними реалізації серверної частини застосунку. Одним з таких є мова програмування Python, що має чистий синтаксис та використовується для роботи з аналізом даних, що є поза конкуренцією на сучасному ринку. В випадку з вебзастосунком салону краси, за допомогою даної мови програмування відбудеться реалізація звітності та аналітики, що автоматизує рутинні бізнес-процеси системи [19].

SQL є зручною мовою взаємодії з базами даних та є популярною вже багато років у сфері розробки сучасного програмного забезпечення. Здебільшого, використання даної мови є у реляційних базах даних. Прикладом реляційної бази даних може стати таблиця «користувачі», де є дані користувача, на яку процедуру він записався або який сертифікат має. За допомогою SQL створюється запит до даної таблиці, щоб звести певні дані та є необхідним для будь-якого застосунку без виключення. PostgreeSQL є тією самою реляційною базою даних, що має збереження даних у вигляді таблиць. Її використання є перевагою через ACID-відповідність, що забезпечує завершує операцію транзакції або повністю, або повністю її відхиляє, що гарантує збереження цілісності. Також відмітимо гнучкість, через можливість роботи із JSON-форматом, що надає змогу роботи як в NoSQL.

Використання Node.js дозволяє роботу JavaScript на стороні серверу, що значно покращує реалізацію full-stack розробки. Він використовує архітектуру асинхронну та працює в одному потоці, не створюючи багато процесів для навантаження системи, що спрощує малі запити до застосунку та своєчасне виконання [20].

Для розгортання вебзастосунку використовуватиметься Docker, що є потужним інструментом DevOps. За допомогою контейнерізації застосунку, відбувається його «ізоляція», що вирішує проблеми роботи на локальному та хмарному сервері. Він створює спільну віртуальну мережу для роботи між контейнерами та керує збереженням даних за допомогою прив’язки.

3.2.3 Вплив CRM та його використання у розробці

Для відтворення інформаційної системи управління застосунком виникає потреба реалізації власних потреб, що зумовлює собою кастомну систему. Було визначено створення такого типу системи на базі CRM. З цього з'являється реалізація бізнес-логіки за допомогою коду, що зумовлює собою моделювання сутностей бази даних, автоматизації бізнес-процесів та потреба в рольовій моделі доступу.

Головним в даній частині є рівень бази даних, адже для реалізації інформаційною системи управління потрібно реалізувати бізнес-логіку, що можна розглядіти через зв'язки. Зв'язки, що відображають дану систему, здебільшого є «один до багатьох» або «багато до багатьох».

Тоді виникає потреба у використанні патерну стану, адже він найбільше сприяє реалізації бізнес-логіки. Таким чином, не можна буде записатися на послугу, оминувши етап обрання майстра. Використання RBAC для перевірки ролей та надання доступу, що зберігає потребу в регулюванні збереження даних бізнесу. В свою чергу використання елементів фільтрації, сегментації та сортування збереже швидкий доступ. Використання дошок-аналітики спрятиме візуалізації процесів та їх стан, шо спростить роботу менеджменту з клієнтами.

На рівні інфраструктури виникає потреба в Kubernetes що збереже критично важливі процеси під час розгортання. Таким чином буде реалізоване оновлення без зупинки процесів системи, автоматичне генерування звітів кожен певний проміжок часу, що уникне потреби оновлення вручну [21].

Висновки до розділу 3

В даному розділі пророблена робота з проєктуванням вебзастосунку салону краси з інформаційною системою управління. Використана розробка UML-діаграм для візуалізації концепцій роботи застосунку, його основного функціоналу та визначення його компонентів та класів, яке місце в архітектурі підлягає та як взаємодіє клієнт із постачальником послуг краси.

Використано діаграму використання для відображення сфери застосування функціоналу, показано кращий сценарій використання користувачами системи, показано акторів та їх можливості виконання своїх робочих потреб. Продемонстровано діаграму з веденням прийому клієнта для закріплення. Також, продемонстровано діаграму класів, що моделює повну систему, зв’язки між класами, під які категорії вони підпадають та які атрибути мають класи, що робить їх унікальними.

Діаграми станів та переходів надали змогу розглянути життєвий цикл запису клієнта, що є основним функціоналом застосунку, що надає послуги. Показано повний шлях клієнтського візиту, від запису клієнта на послугу, до закриття сесії запису після її виконання. Також зазначено операційний цикл салону, що огортає в собі також фінансову звітність та аналітику, складські операції та регулювання розкладу.

Діаграми пакетів та компонентів закріпили модулі роботи застосунку, з яких шарів він складається та яке місце в цьому відіграє інформаційна система управління. Особисто діаграма пакетів допомагає в поділені системи на рівні для спрощення розробки.

Зроблено огляд задіяних технологій розробки, що сприятимуть отриманню бажаного результату при кінцевому результаті проєкту. Зазначена потреба у використанні Docker та оркестрації через Kubernetes для автоматизації масштабування.

Визначено шляхи реалізації інтегрції в застосунок кастомної інформаційної системи управління для додерження CRM, через що виникатиме автоматизація бізнес-процесів салону та вирішуватиметься питання з генерацією звітності за певний протяжок часу в залежності робочої необхідності.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

Вебзастосунок «Beauty Book» є цілковитим застосунком салону краси з підтримкою інформаційною системи управління, що вважається значною перевагою, адже використовується в цілях бізнесу надання послуг. Даний застосунок має 8 головних сторінок: від головної сторінки до профілю користувача. Реалізовано наступні типи користувачів: гість, користувач (він же потенційний клієнт), майстер, адміністратор та власник. Роль гість є має обмежені можливості – він може переглядати інформацію, але поки не зареєстрований, здійснити бонус він не може. Для користувачів системи є нагадування про майбутній запис, через «дзвоник», можливість додати собі нагадування через інтеграцію з гугл календарем. Адміністратор та власник мають схожі права та функціонал, але тільки власник може генерувати фінансову звітність та аналітику.

4.1 Огляд реалізації користувацького інтерфейсу

Для створення mockup інтерфейсу використано сучасний вебсервіс Figma, що налічує багато готових рішень, інструментів для роботи та постійний доступ до проєкту, за умови стабільного інтернет-з’єднання. Для гостя системи доступні для перегляду всі сторінки вебзастосунку, але без реєстрації немає доступу до зарахування бонусів та запису на послугу. Вебзастосунок є мультимовним – можна обрати англійську або українську мову (рис. 4.1) та змінювати теми, наприклад на світлу (рис. 4.2) або на темну (рис.4.3).

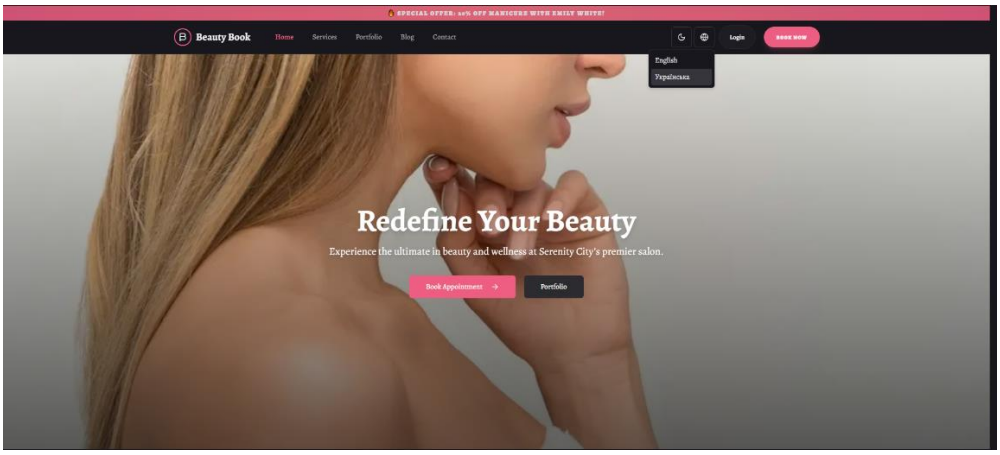

Рисунок 4.1 – Зміна мови інтерфейсу

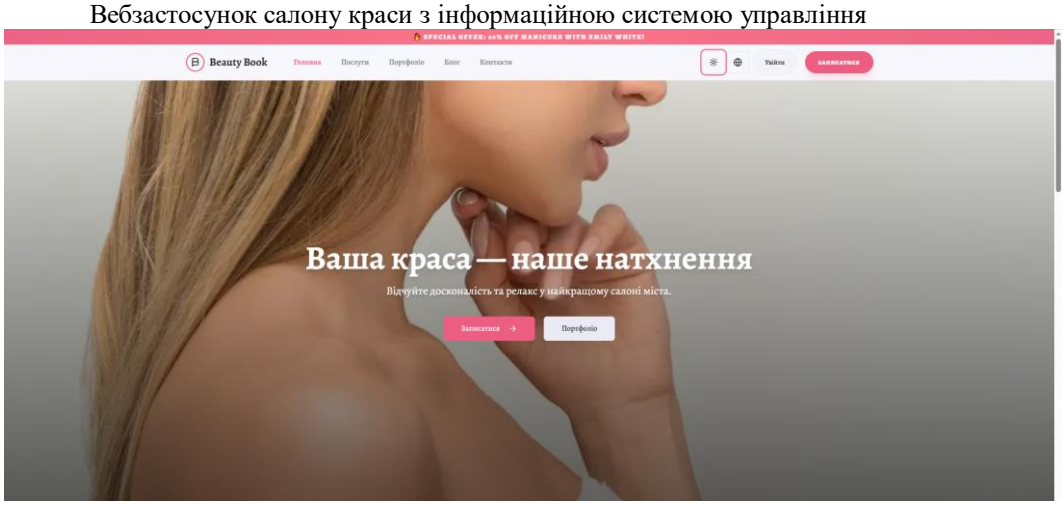

Рисунок 4.2 – Світла тема інтерфейсу

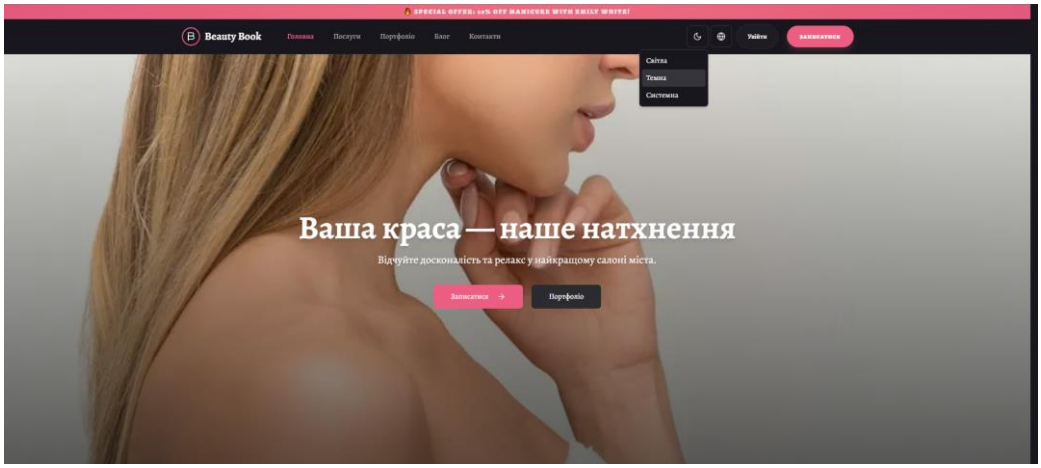

Рисунок 4.3 – Темна тема інтерфейсу

Також гостю системи доступна кнопка «увійти». При натиснені на неї запропоновано ввести логін і пароль, якщо колись був зареєстрований користувач, увійти за допомогою гугл акаунту, або зареєструватися (рис. 4.4).

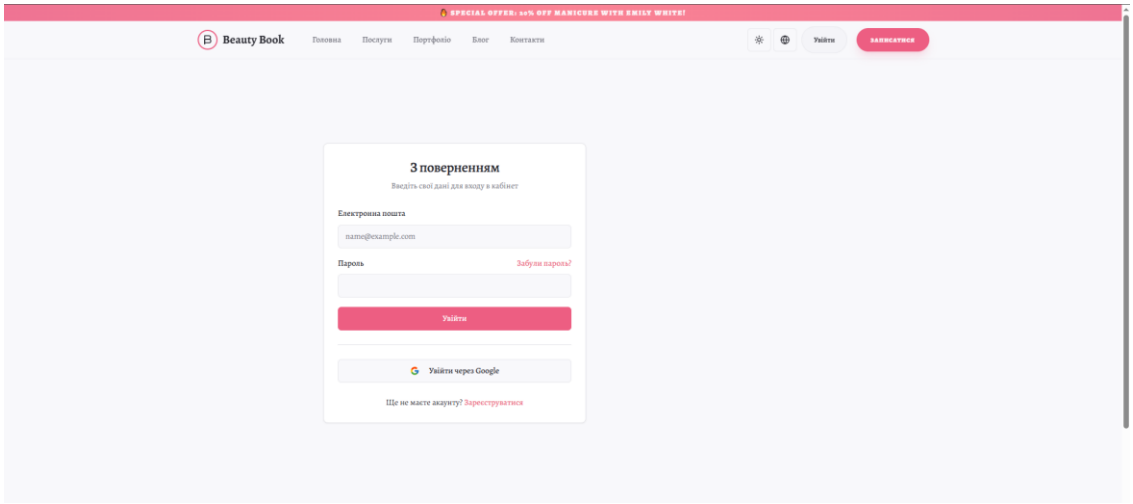

Рисунок 4.4 – Вікно входу

У вікні реєстрації потрібно ввести ім’я, електронну пошту та пароль (рис.4.5). Також, запропоновано виконати реєстрацію через гугл акаунт, що є зручним для багатьох, адже більшість не хоче витрачати багато часу на заповнення даних, а має бажання натиснути лише одну кнопку.

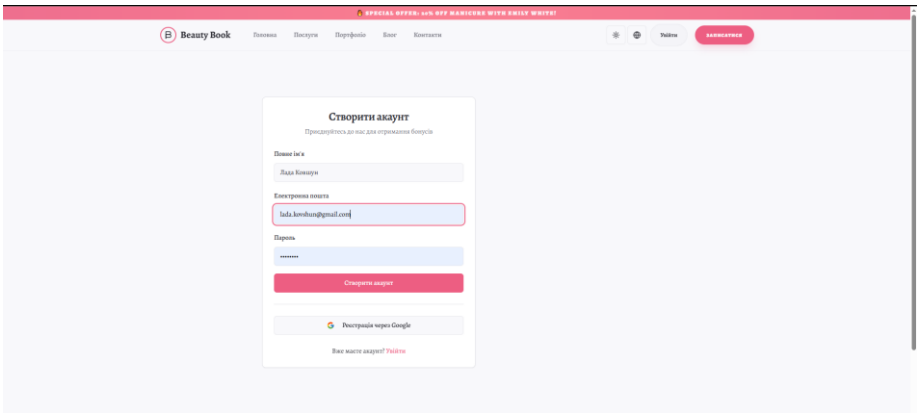

Рисунок 4.5 – Вікно реєстрації

Вже зареєстрований користувач має змогу для детального перегляду застосунку:

– перегляд послуг, можливість їх швидкого пошуку за категорією або назвою, перегляд актуальних цін (рис.4.6);

– перегляд портфоліо кожного з майстрів, їх рейтинг у системі, можливість запису до конкретного майстру підчас перегляду його портфоліо, огляд цін на його послуги (рис.4.7);

– перегляд блогу та новин салону (рис.4.8);

– контактна інформація та зворотній зв’язок (рис.4.9);

– профіль користувача (рис.4.10);

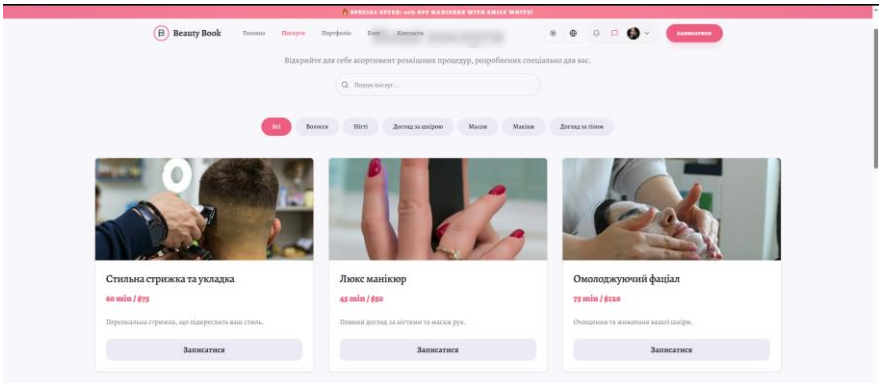

Рисунок 4.6 – Сторінка послуг

На сторінці портфоліо можна переглянути майстрів та категорію послуг, у якій вони працюють. Також можна переглянути їх рейтинг, що відображається одразу. Майстрів можна шукати за іменами або їх категорією послуг.

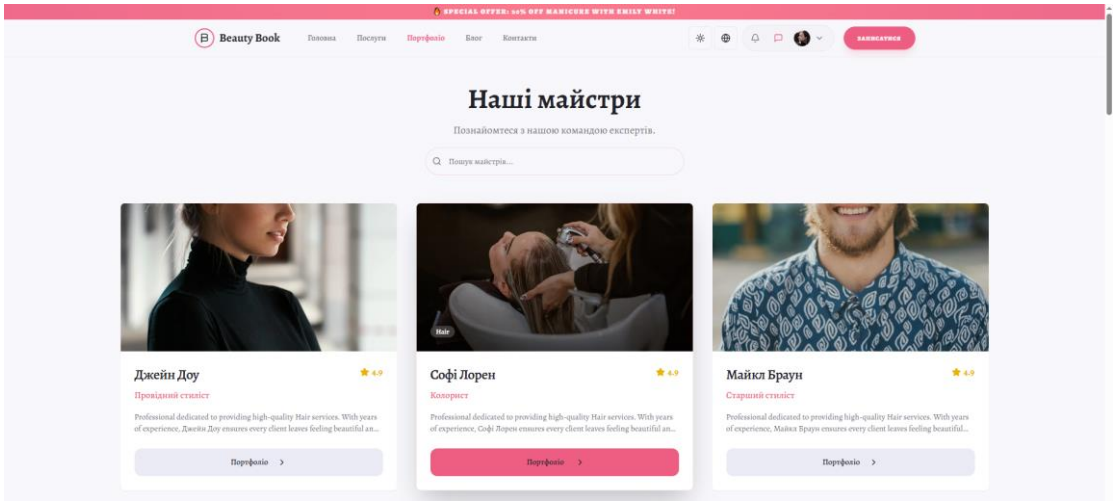

Рисунок 4.7 – Сторінка портфоліо

Розділ «блог» обіймає в собі можливості перегляду статей та порад, які можуть бути необхідні для клієнта. Це привертає увагу та затримує клієнта на сайті та для нього це корисне – адже там розміщуються актуальні новини про знижки на послуги.

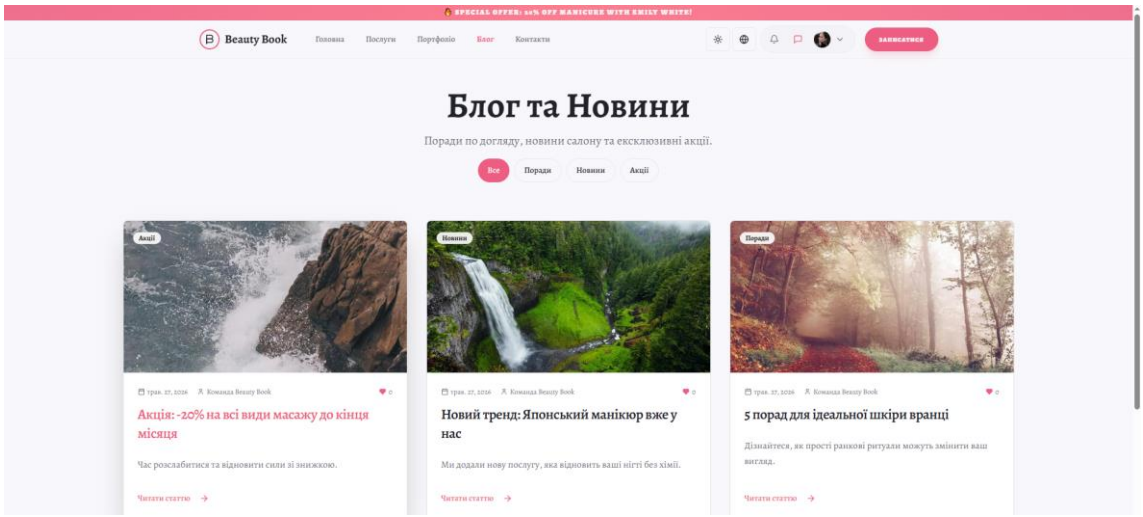

Рисунок 4.8 – Сторінка блогу та новин

Також, користувач може переглянути статтю, залишити коментар або поставити «вподобайку» до неї, що створює тісний зв’язок між клієнтом та надавачем послуг у сфері краси. Такий прийом є зручним для привернення клієнтів.

На сторінці «контакти», користувач може переглянути контактну інформацію салону краси, записати зауваження та переглянути актуальний графік роботи самого салону. Це є також одним з показником довіри для клієнтів салону, бо вони бачать, що про них дбають.

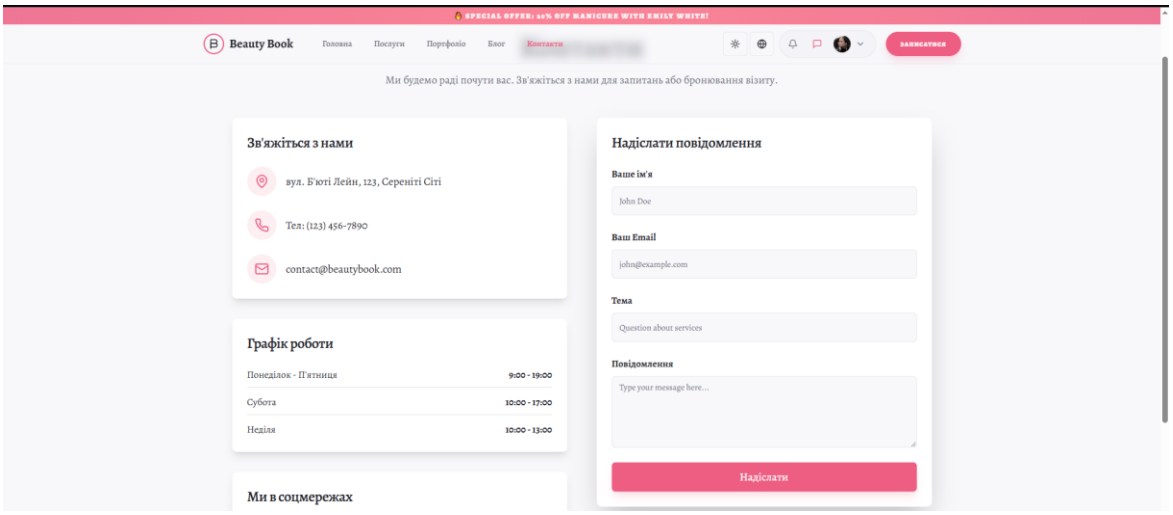

Рисунок 4.9 – Сторінка контактів

На сторінці профілю користувача, можна переглянути фото, змінити його, переглянути інформацію про майбутні записи, бонусну систему, систему сповіщень, що є, а також отримати реферальний код для запрошення друга, після якого можна буде отримати бонус.

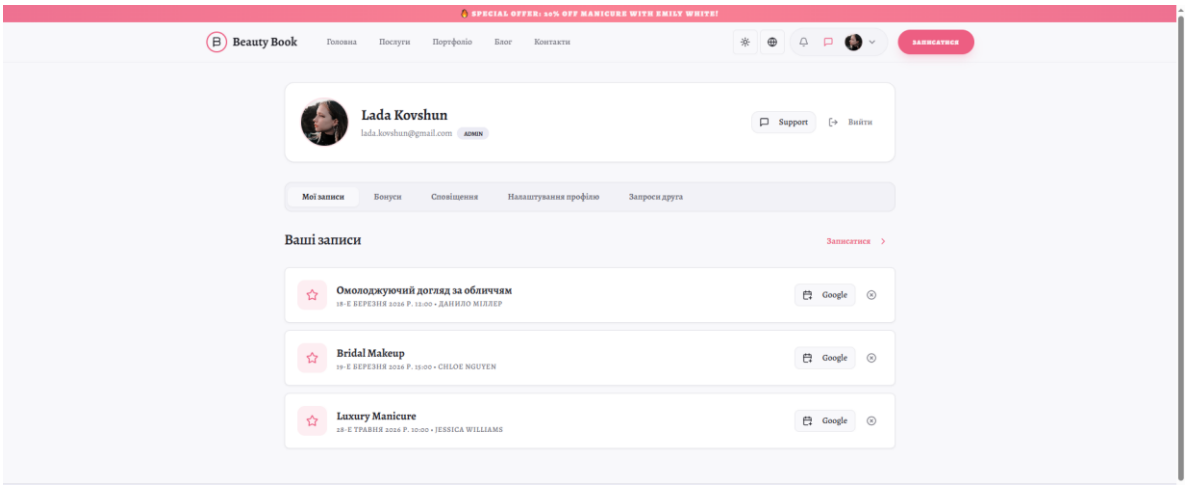

Рисунок 4.10 – Сторінка профілю

Таким чином, можна зазначити, що зазначений вище інтерфейс має потужну базу та є цілковито конкурентоспроможним на сучасному ринку, що робить його

ще привабливішим для багатьох клієнтів. Більшою перевагою є мультимовність, адже заохочує також європейських клієнтів.

4.2 Реалізація бізнес-логіки

Для усіх користувачів системи салону краси реалізована бізнес-логіка застосунку, що додержується типу інформаційної системи управління. Використання даної системи заохочує клієнтів, адже вона має клієнтоорієнтований підхід та автоматизує процеси салону краси, що закриває велику необхідність як малого, так і великого бізнесу.

4.2.1 Реалізація головної сторінки вебзастосунку

Головна сторінка (4.11) – це обличчя салону краси, тому вона повинна об’єднувати в собі стильний та простий дизайн, лозунг салону, відгуки задоволених клієнтів, зручний та адаптивний інтерфейс, щоб притягнути якомога більше клієнтів до салону. У верхній частині повинно розміщуватись навігаційне меню, щоб зручно пересуватися вебзастосунком.

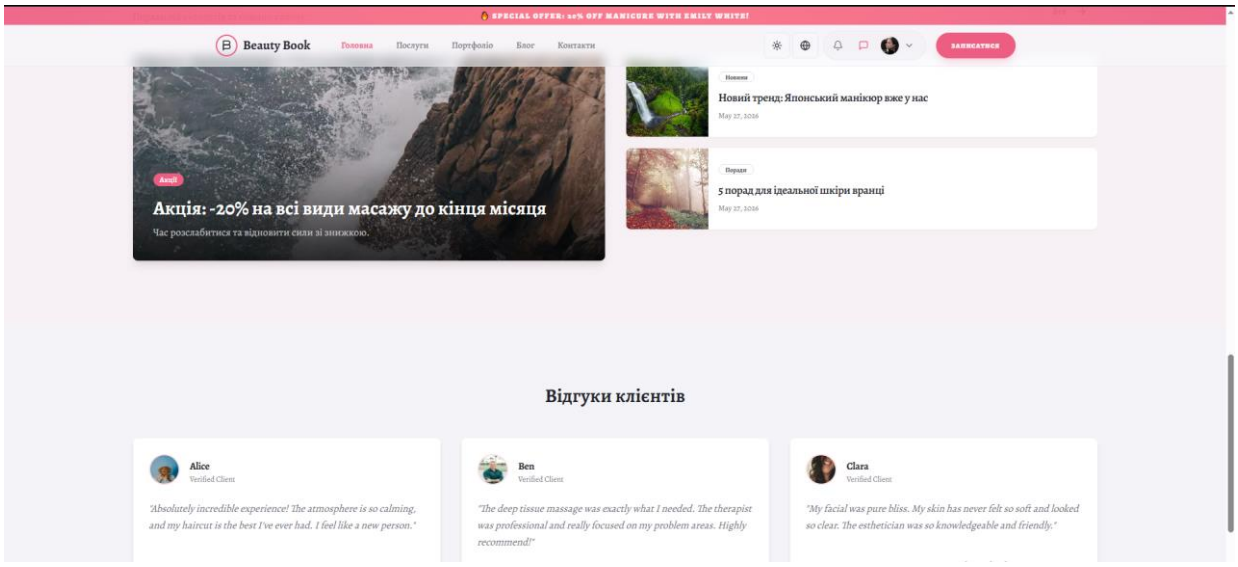

Рисунок 4.11 – Головна сторінка

Розглянемо код головної сторінки вебзастосунку (рис.4.12). Можна побачити, що йде використання useEffect для використання React Query та

звернення до бази даних відбуваються через використання HTTP-запитів до Node.js API, який збирає данні з PostgreSQL (4.13).

```tsx
import { useMemo, useState, useEffect } from 'react';

export default function Home() {
  const params = useParams();
  const lang = (params?.lang as Locale) || 'en';
  const dictionary = useDictionary(lang);

  const [rawPosts, setRawPosts] = useState<BlogPost[]>([]);
  const [postsLoading, setPostsLoading] = useState<boolean>(true);

  useEffect(() => {
    async function fetchBlogPosts() {
      try {
        setPostsLoading(true);
        const response = await fetch(`/api/blog?lang=${lang}`);
        if (response.ok) {
          const data = await response.json();
          setRawPosts(data);
        }
      } catch (error) {
        console.error('Error fetching blog posts:', error);
      } finally {
        setPostsLoading(false);
      }
    }

    if (lang) {
      fetchBlogPosts();
    }
  }, [lang]);

  const latestPosts = useMemo(() => {
    if (!rawPosts || rawPosts.length === 0) return [];
    return [...rawPosts]
      .sort((a, b) => {
        const dateA = a.createdAt ? new Date(a.createdAt).getTime() : 0;
        const dateB = b.createdAt ? new Date(b.createdAt).getTime() : 0;
        return dateB - dateA;
      })
      .slice(0, 3);
  }, [rawPosts]);

  if (!dictionary || !dictionary.home_page) {
    return (
      <div className="flex items-center justify-center min-h-screen">
        <Loader2 className="h-12 w-12 animate-spin text-primary/20" />
      </div>
    );
```

Рисунок 4.12 – Код головної сторінки салону краси

```ts
import { NextResponse } from 'next/server';
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

export async function GET(request: Request) {
  try {
    const { searchParams } = new URL(request.url);
    const lang = searchParams.get('lang') || 'en';

    const posts = await prisma.blogPost.findMany({
      where: {
        lang: lang,
      },
      orderBy: {
        createdAt: 'desc', // Сортування на рівні БД, щоб полегшити роботу фронтенду
      },


    return NextResponse.json(posts);
  } catch (error) {
    console.error('Database error:', error);

    return NextResponse.json(
      { error: 'Internal Server Error', details: 'Failed to fetch blog posts' },
      { status: 500 }
    );
  }
}
```

Рисунок 4.13 – API Route головної сторінки

Головна сторінка відповідає складеній специфікації вимог та викрнує свій основний функціонал, як привітальна ознайомча частина вебзастосунку, що надає інформацію про салон та можливість запису.

4.2.2 Реалізація реєстрації та запису

Однією з найважливіших частин кожного вебзастосунку зазвичай є функціонал реєстрації/авторизації користувача в системі. В залежності від ролі користувача в системі, змінюються права доступу до неї. Незареєстрований користувач (гість) не може записатися на послугу, поки не зареєструється, або надіслати повідомлення у чат підтримки одного з адміністраторів. Майстри мають змогу редагувати власний профіль у портфоліо, а власник та адміністратор мають права доступу до дошки аналітики та звітності. Розглянемо реалізацію функцію asinc, яка за це відповідає (рис.4.14).

```
1
2   import { NextResponse } from 'next/server';
3   import { PrismaClient } from '@prisma/client';
4   import bcrypt from 'bcryptjs';
5
6   const prisma = new PrismaClient();
7
8   export async function POST(request: Request) {
9     try {
10      const { name, email, password } = await request.json();
11
12      const hashedPassword = await bcrypt.hash(password, 10);
13
14      let targetRole = 'Client';
15      if (email.toLowerCase() === "marina.falenkova@gmail.com") targetRole = 'Owner';
16      else if (email.toLowerCase() === "lada.kovshun@gmail.com") targetRole = 'Admin';
17      else if (email.toLowerCase() === "specialist@beautybook.com") targetRole = 'Specialist';
18
19      const user = await prisma.user.create({
20        data: {
21          name: name || email.split('@')[0],
22          email: email,
23          password: hashedPassword,
24          role: targetRole,
25          rewardPoints: 0
26        }
27      });
28
29      return NextResponse.json({ id: user.id, name: user.name, email: user.email, role: user.role });
30    } catch (error) {
31      return NextResponse.json({ message: "Користувач вже існує або сталася помилка БД" }, { status: 400 });
32    }
33  }
```

Рисунок 4.14 – Asinc для реєстрації користувача

Розглянемо як виглядає профіль користувача (рис. 4.15). В свою чергу профіль містить як ім’я та аватар користувача (яку він може змінювати) так і контактну інформацію про записи та отримання бонусів (наприклад, за реєстрацію

нараховані автоматично або за відвідування послуги без відміни запису), сповіщення які надходили та можливість налаштування профілю. Таким чином, front-end частина сторінки використовує React-хуки useState та useEffect для завантаження даних Node.js (рис.4.16).

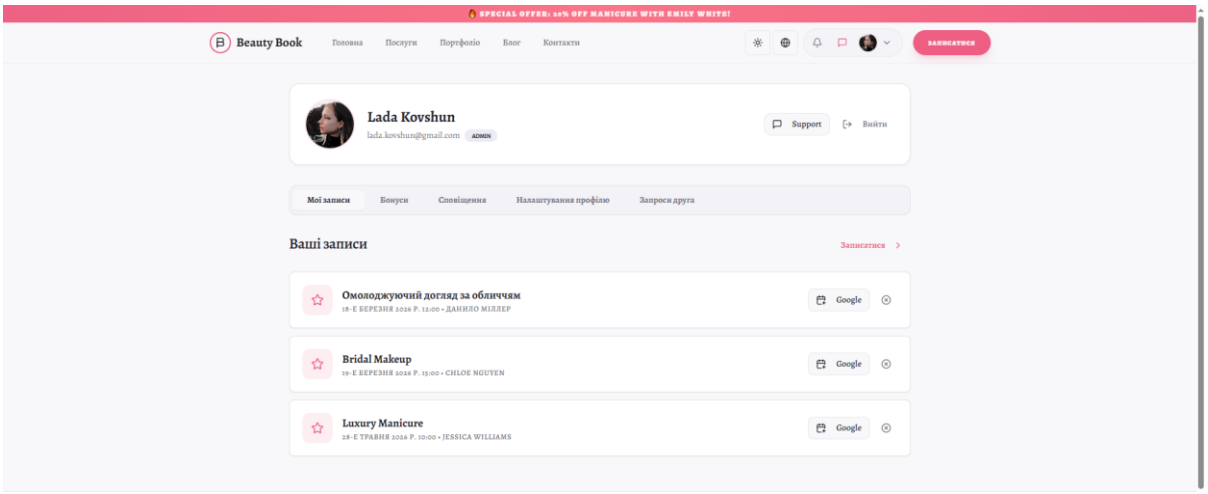

Рисунок 4.15 – Сторінка профілю користувача

```
export default function AccountPage() {
  const params = useParams();
  const lang = (params?.lang as Locale) || 'en';
  const dictionary = useDictionary(lang);
  const { toast } = useToast();
  const fileInputRef = useRef<HTMLInputElement>(null);

  const [user, setUser] = useState<{ id: string; email: string } | null>(null);
  const [userProfile, setUserProfile] = useState<UserProfile | null>(null);
  const [appointments, setAppointments] = useState<Appointment[]>([]);
  const [notifications, setNotifications] = useState<Notification[]>([]);
  const [isLoading, setIsLoading] = useState(true);

  const [fullName, setFullName] = useState('');
  const [photoURL, setPhotoURL] = useState('');
  const [isUpdating, setIsUpdating] = useState(false);

  const dateLocale = lang === 'uk' ? uk : enUS;

  useEffect(() => {
    async function fetchAccountData() {
      try {
        const authRes = await fetch('/api/auth/me');
        if (!authRes.ok) {
          setUser(null);
          setIsLoading(false);
          return;
        }
        const sessionData = await authRes.json();
        setUser({ id: sessionData.id, email: sessionData.email });

        const [profileRes, appointmentsRes, notificationsRes] = await Promise.all([
          fetch('/api/user/profile'),
          fetch('/api/user/appointments'),
          fetch('/api/user/notifications')
        ]);

        if (profileRes.ok) {
          const profile: UserProfile = await profileRes.json();
          setUserProfile(profile);
          setFullName(profile.name || '');
          setPhotoURL(profile.photoURL || '');
        }
        if (appointmentsRes.ok) setAppointments(await appointmentsRes.json());
        if (notificationsRes.ok) setNotifications(await notificationsRes.json());

      } catch (error) {
        console.error("Помилка завантаження даних акаунту:", error);
```

Рисунок 4.16 – Код сторінки профілю

Здійснити запис на послугу можна декількома способами: на головній, у послугах, через портфоліо майстра. Розглянемо стандартний запис через головну сторінку (рис. 4.17). Потрібно здійснити заступні етапи:

– натиснути на кнопку «Записатися»;
– обрати процедуру;
– обрати майстра;
– обрати дату та час;
– підтвердити та оплатити (рис. 4.18).

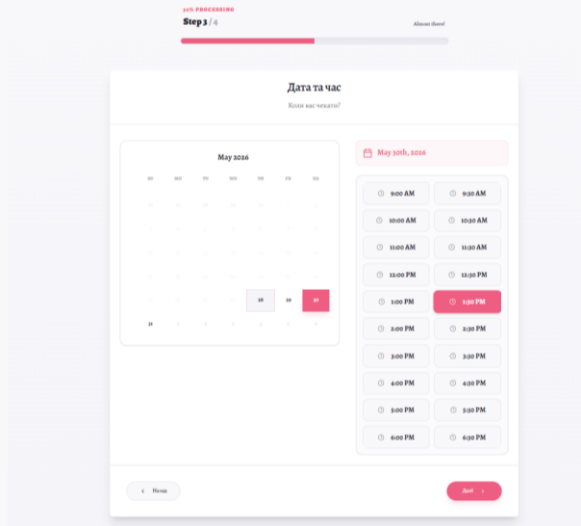

Рисунок 4.17 – Запис на послугу. Обрання дати та часу

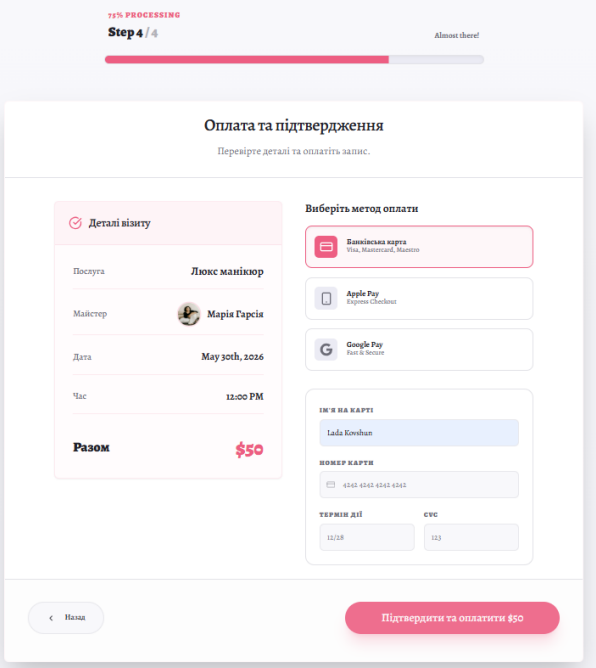

Рисунок 4.18 – Підтвердження та оплата

Після того як оплата здійснена можна буде додати нагадування про запис до гугл календарю (рис. 4.19) та переглянути у профілі. Таким чином повний цикл запису на послугу здійснюється без великого витрачення часу на це (рис. 4.20).

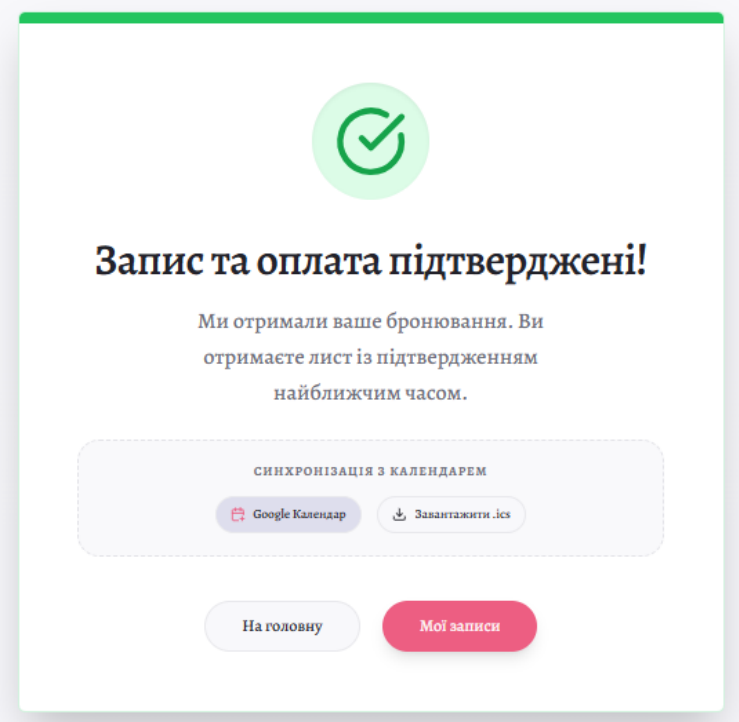

Рисунок 4.19 – Синхронізація з гугл календарем

```
export async function POST(request: Request) {
  try {
    const body = await request.json();
    const {
      serviceId,
      serviceName,
      specialistId,
      specialistName,
      date,
      price,
      paymentMethod,
      lang
    } = body;

    const userId = request.headers.get('x-user-id');

    if (!userId) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
    }

    if (!serviceId || !specialistId || !date) {
      return NextResponse.json({ error: 'Missing required fields' }, { status: 400 });
    }

    await new Promise((resolve) => setTimeout(resolve, 2000));

    const isUk = lang === 'uk';
    const formattedDate = new Date(date).toLocaleString(isUk ? 'uk-UA' : 'en-US', {
      dateStyle: 'medium',
      timeStyle: 'short',
    });

    const bookingNotificationTitle = isUk ? "Оплату отримано & Запис підтверджено" : "Payment Received & Booking Confirmed";
    const bookingNotificationMessage = isUk
      ? `Ви успішно оплатили ${serviceName} до ${specialistName} на ${formattedDate}. Чекаємо на вас!`
      : `You successfully paid for ${serviceName} with ${specialistName} on ${formattedDate}. See you soon!`;

    const rewardNotificationTitle = isUk ? "Нараховано бонуси" : "Bonuses Earned";
    const rewardNotificationMessage = isUk
      ? "Ви отримали +100 балів за ваше бронювання. Дякуємо, що обираєте нас!"
      : "You've earned +100 reward points for your booking. Thank you for choosing us!";

    const result = await prisma.$transaction(async (tx) => {

      const appointment = await tx.appointment.create({
        data: {
          userId,
          serviceId,
          serviceName,
```

Рисунок 4.20 – Код функцій запису

Даний функціонал забезпечує коректну роботу запису та автоматизує головний бізнес-процес салону, а саме – здійснення запису до салону на певну категорію послуг, які наявні. Для новітніх салонів такий функціонал є необхідним та відображає їх мету забезпечити клієнта наданням власних послуг.

4.2.3 Реалізація системи аналітики та звітності

Для реалізації аналітики та звітності системи використано аналітичні можливості скриптів на python, що автоматизує фінансовий облік бізнесу та відображає MIS та CRM для реалізації інформаційної системи управління (рис. 4.21).

```
app = FastAPI(title="Beauty Book MIS Analytics API")

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

def calculate_financials(df: pd.DataFrame) -> dict:
    if df.empty:
        return {}

    total_revenue = float(df['price'].sum())

    df['commission'] = df['price'] * 0.3
    df['tax'] = df['commission'] * 0.2
    df['net_pay'] = df['commission'] - df['tax']

    spec_group = df.groupby(['specialist_id', 'specialist_name']).agg(
        count=('id', 'count'),
        revenue=('price', 'sum'),
        commission=('commission', 'sum'),
        tax=('tax', 'sum'),
        net_specialist_pay=('net_pay', 'sum')
    ).reset_index()

    specialist_table = spec_group.to_dict(orient='records')


    total_commission = float(df['commission'].sum())
    total_taxes = float(df['tax'].sum())
    material_costs = total_revenue * 0.1
    net_profit = total_revenue - total_commission - total_taxes - material_costs

    def categorize(name):
        if 'Hair' in name: return 'Hair'
        if 'Nail' in name: return 'Nails'
        if 'Facial' in name: return 'Skincare'
        return 'Other'

    df['category'] = df['service_name'].apply(categorize)
    cat_group = df.groupby('category').size().to_dict()
    pie_data = [{'name': k, 'value': v} for k, v in cat_group.items()]

```

Рисунок 4.21 – Частина коду автоматизації фінансового звіту салону

У коді наявно прописані алгоритми розподілу фінансової системи салону, (детальніше в додатку А), які включають в собі:

– комісійні – де SUM(price) * 0.3 це автоматичний розрахунок заробітної плати майстрів (30% від виторгу);

– податкове навантаження – де (SUM(price) * 0.3) * 0.2 це утримання податку (20% від нарахованого доходу);

– собівартість матеріалів та чистий прибуток – що включає в собі обчислення витратної частини (material_costs = total_revenue * 0.1) та виведення фінального KPI салону краси – net_profit;

– потокова генерація звітів – де використання io.StringIO() та оператора yield відображає системну оптимізацію під великі масиви даних (Big Data / Enterprise).

З боку користувача наявний доступ до панелі керування, де є повна інформація про популярні послуги, прибуток, записи, до яких майстрів найчастіше записувалися клієнти та діаграма тенденції записів (рис. 4.22 та рис. 4.23).

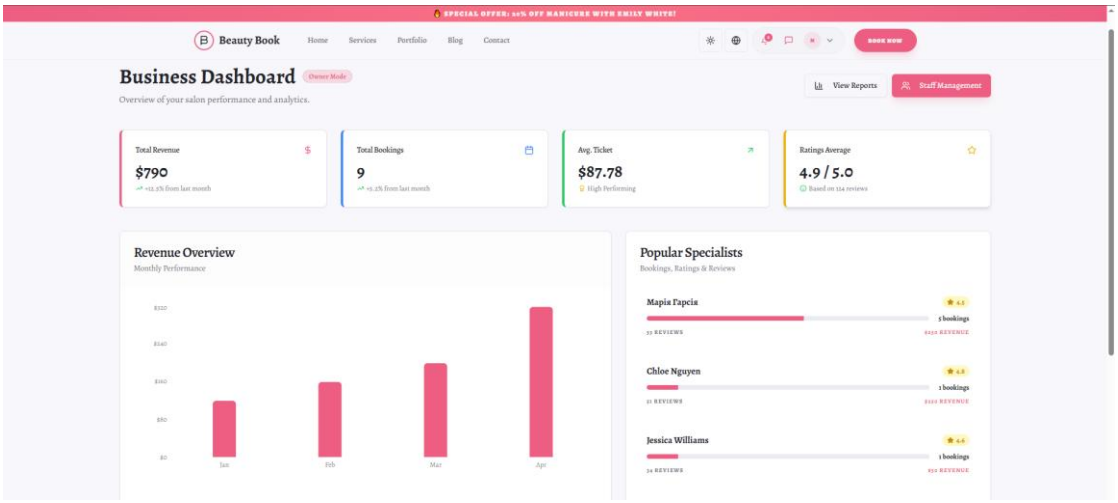

Рисунок 4.22 – Бізнес-панель аналітики: доступ власника

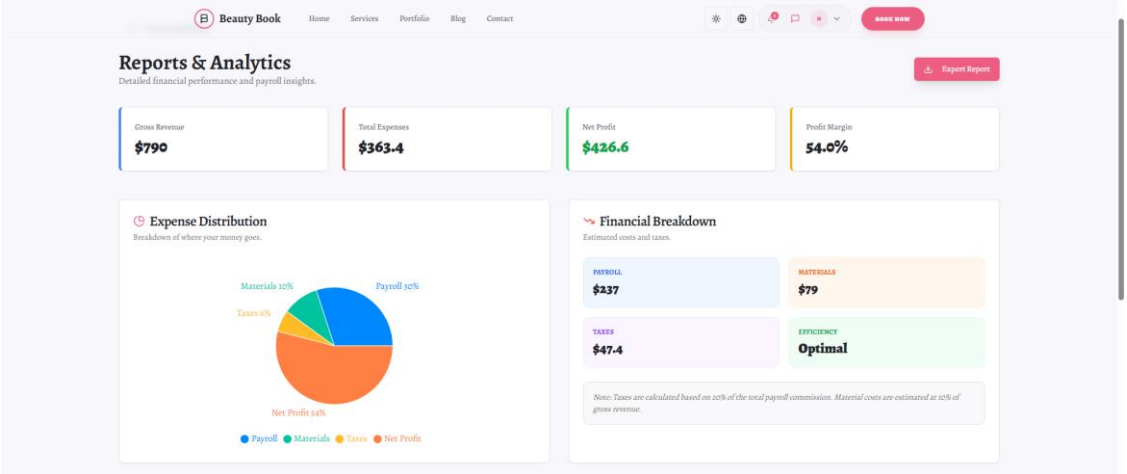

Рисунок 4.23 – Фінанси та аналітика: доступ власника

Також є розрахунок заробітної плати працівника – це спрощує нарахування та розуміння власником прибутковості даного бізнесу (рис. 4.24). Там зазначені податки на кожного працівника, комісійний внесок та чиста заробітна плата майстра. Також зазначена кількість записів. Повний звіт можна завантажити в формат «.csv» та переглянути (рис. 4.25).

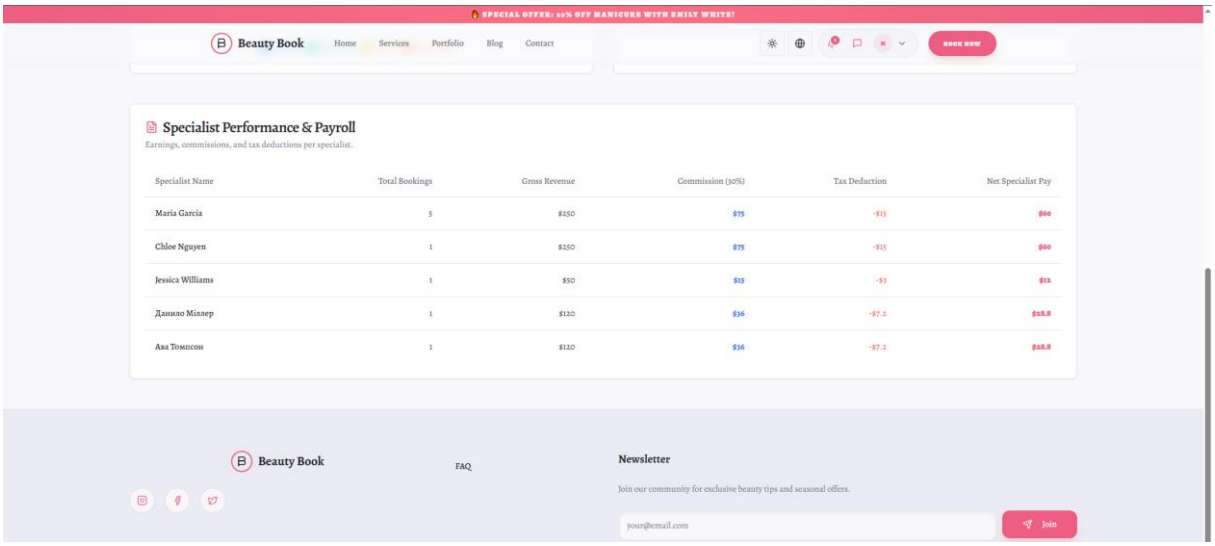

Рисунок 4.24 – Розрахунок заробітної плати

	A
1	Financial Report - Beauty Book
2	Generated At,"29/05/2026, 13:58:46"
3	
4	Financial Summary
5	Gross Revenue,"$790.00"
6	Payroll (Commission),"$237.00"
7	Material Costs,"$79.00"
8	Estimated Taxes,"$47.40"
9	Net Profit,"$426.60"
10	
11	Specialist Performance & Payroll
12	Specialist Name,"Total Bookings","Gross Revenue","Commission (30%)","Tax Deduction","Net Specialist Pay"
13	Maria Garcia,"5","250.00","75.00","15.00","60.00"
14	Chloe Nguyen,"1","250.00","75.00","15.00","60.00"
15	Jessica Williams,"1","50.00","15.00","3.00","12.00"

Рисунок 4.25 – Фінансовий звіт

Таким чином автоматизована бухгалтерська робота салону краси, що значно спрощує виконання бізнес-процесів салону та покращує його роботу. Таким чином є чиста аналітика системи в цілому про його популярність та прибуток, що є

значною перевагою при використанні інформаційної системи управління у сфері бізнесу.

4.2.4 Реалізація системи повідомлень та бонусів

У вебзастосунку салону краси можна отримувати сповіщення від системи про нарахування бонусів або нові пропозиції, що діють на даний момент за допомогою «дзвоника» у верхній частині застосунку (рис. 4.26). Там можна переглянути усі сповіщення, перейти за ними. Історія цих сповіщень зберігається у профілі користувача, що є дуже зручним рішенням. Логіка отримання списку сповіщень (рис. 4.27) відображає в собі отримання списку та відображення його користувачеві та маніпуляції з ним (відмітити як прочитане).

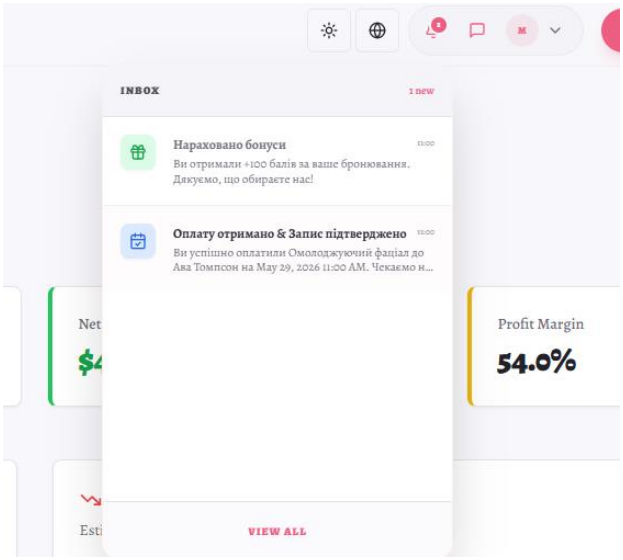

Рисунок 4.26 – Сповіщення користувача «дзвоник»

```
20
21  app.get('/api/v1/notifications', async (req: Request, res: Response) => {
22    const userId = req.query.user_id as string;
23
24    if (!userId) {
25      return res.status(400).json({ error: 'user_id is required' });
26    }
27
28    try {
29      const queryText = `
30        SELECT id, title, message, type, read, created_at as "createdAt"
31        FROM notifications
32        WHERE user_id = $1
33        ORDER BY created_at DESC;
34      `;
35      const result = await pool.query(queryText, [userId]);
36      return res.json(result.rows);
37    } catch (error) {
38      console.error('Error fetching notifications:', error);
39      return res.status(500).json({ error: 'Internal server error' });
40    }
41  });
42
```

Рисунок 4.27 – Код отримання списку сповіщень

Якщо є якісь питання, користувач має змогу написати одному з працівників салону краси та звернутися за допомогою (рис. 4.28 та 4.29), що є дуже клієнтоорієнтованим підходом.

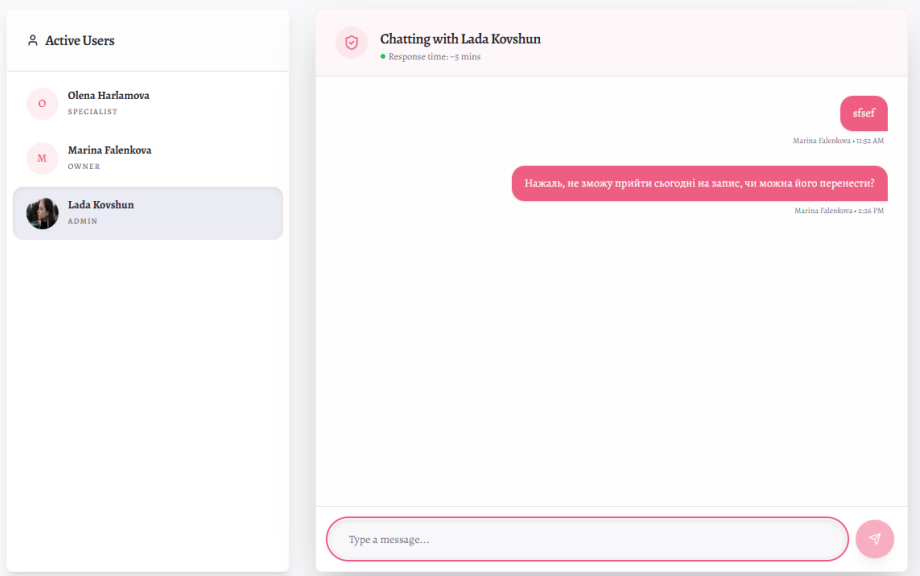

Рисунок 4.28 – Чат з адміністратором

```
//SOCKET.IO

io.on('connection', (socket: Socket) => {
  console.log(`User connected: ${socket.id}`);

  socket.on('join_chat', (chatId: string) => {
    socket.join(chatId);
    console.log(`Socket ${socket.id} joined room: ${chatId}`);
  });


  socket.on('send_message', async (data: {
    chatId: string,
    senderId: string,
    senderName: string,
    text: string,
    isAdmin: boolean
  }) => {
    const { chatId, senderId, senderName, text, isAdmin } = data;

    try {
      // 1. Перевіряємо/створюємо чат у базі (аналог setDoc з merge)
      await pool.query(
        `INSERT INTO chats (id, last_update) VALUES ($1, NOW())
         ON CONFLICT (id) DO UPDATE SET last_update = NOW()`,
        [chatId]
      );


      const result = await pool.query(
        `INSERT INTO chat_messages (chat_id, sender_id, sender_name, text, is_admin)
         VALUES ($1, $2, $3, $4, $5)
         RETURNING id, created_at`,
        [chatId, senderId, senderName, text, isAdmin]
      );

      const savedMessage = {
        id: result.rows[0].id,
        chatId,
        senderId,
        senderName,
        text,
        isAdmin,
        createdAt: result.rows[0].created_at
      };


      io.to(chatId).emit('receive_message', savedMessage);
```

Рисунок 4.29 – Код отримання та надсилання повідомлень

Система нарахування бонусів наявна як в сповіщеннях так і в профілю користувача (рис. 4.30). Користувач може переглянути яка в нього кількіть бонусів та за потреби їх обміняти (рис. 4.31). Це підтримка системи лояльності – адже клієнтів це привертає. Бонуси можна отримати за реєстрацію або успішний запис на послугу.

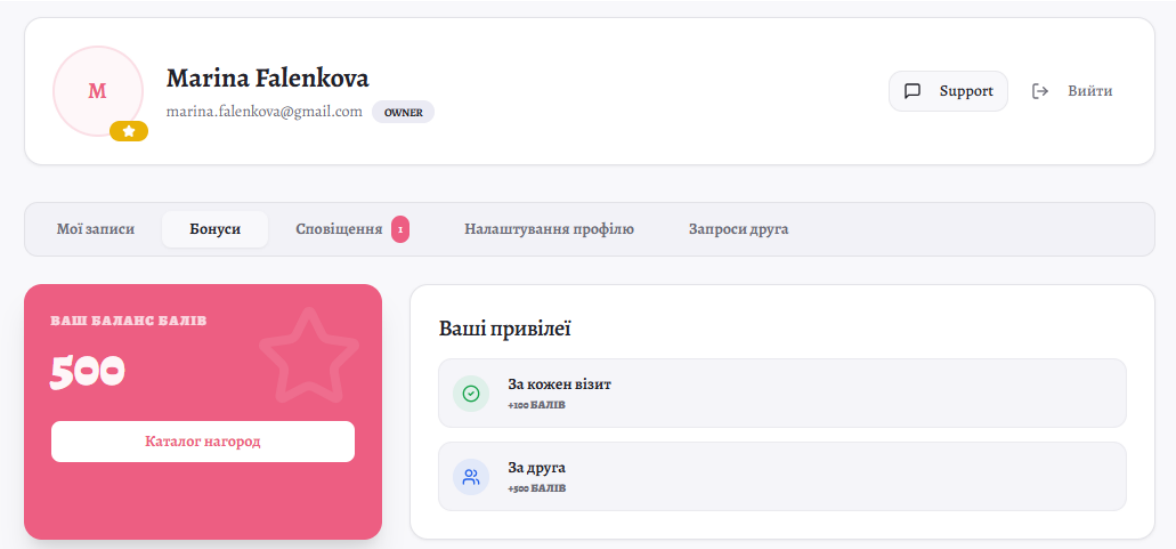

Рисунок 4.30 – Розділ «бонуси» у профілі користувача

```
34
35  const client = await pool.connect();
36
37  try {
38    await client.query('BEGIN');
39
40    const userCheck = await client.query(
41      'SELECT reward_points FROM users WHERE id = $1 FOR UPDATE',
42      [userId]
43    );
44
45    if (userCheck.rowCount === 0) {
46      await client.query('ROLLBACK');
47      return res.status(404).json({ error: 'User not found' });
48    }
49
50    const currentPoints = userCheck.rows[0].reward_points;
51
52    if (currentPoints < pointsNeeded) {
53      await client.query('ROLLBACK');
54      return res.status(400).json({ error: 'Not enough points' });
55    }
56
57    const updateResult = await client.query(
58      `UPDATE users
59       SET reward_points = reward_points - $1
60       WHERE id = $2
61       RETURNING reward_points as "rewardPoints"`,
62      [pointsNeeded, userId]
63    );
```

Рисунок 4.31 – Код отримання пулу для ізольованої транзакції

Даний функціонал вебзастосунку салону краси є перевагою для бізнесу, адже орієнтований на утримання клієнтів у даній сфері послуг, що в свою чергу є однією з переваг інформаційної системи управління бізнесом та аспектом приваблення нових клієнтів.

4.2.5 Редагування розкладу та профілю

Іноді є випадки, коли розклад одного з працівників потребує змін, а працівник в свою чергу не може цього зробити без повідомлення про це керівництва. Керівник або адміністратор салону краси в свою чергу повинні відредагувати графік працівника. Для цього в панелі керування створене «Персонал та графіки», щоб вчасно мати змогу ввести корективи (рис.4.32).

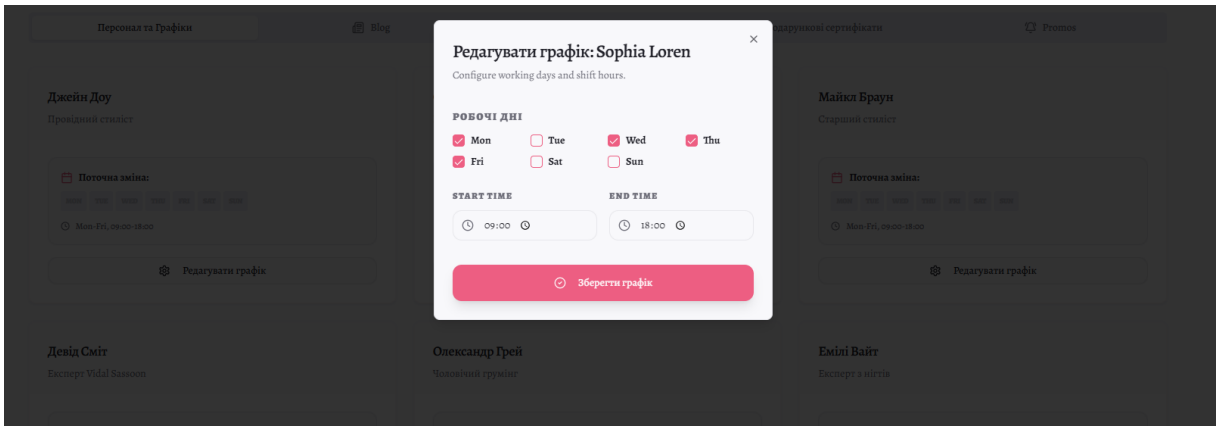

Рисунок 4.32 – Розділ панелі керування графіком

Через REST API для отримання повного розкладу конкретного майстра та пакетного оновлення (UPSERT) робочих днів можна внести зміни у розклад (рис. 4.33). Таким чином забезпечена безпека та коректність збереження даних у системі та запобігає помилок при запиті до бази даних.

```
29  app.put('/api/v1/specialists/:id/schedule', async (req: Request, res: Response) => {
30    const { id } = req.params;
31    const { schedule } = req.body;
32    const client = await pool.connect();
33    try {
34      await client.query('BEGIN');
```

Рисунок 4.33 – Збереження розкладу

Також можна редагувати профіль користувача (рис. 4.34), наприклад, фото профілю або ім’я. Фото можна завантажити як з локального пристрою, так і за допомогою url зображення аватару (рис. 4.35). Обмеження в 1 мб розміру файлу не навантажуватиме базу даних та є зручним.

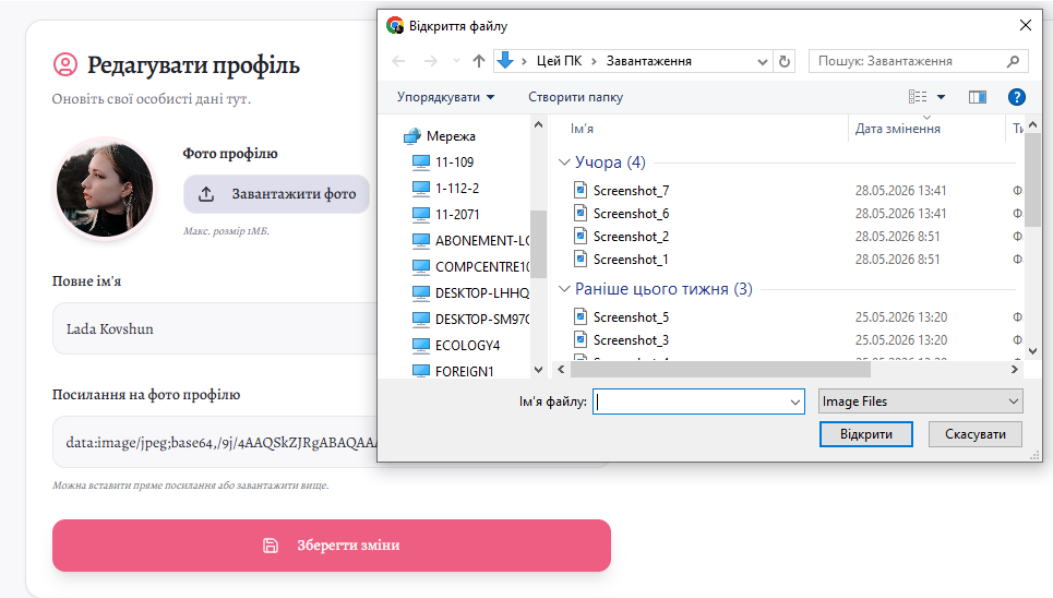

Рисунок 4.34 – Розділ редагування профілю користувача

```
app.put('/api/v1/users/:id', async (req: Request, res: Response) => {
  const { id } = req.params;
  const { fullName, photoURL } = req.body;

  if (!fullName) {
    return res.status(400).json({ error: 'Full name is required' });
  }

  try {
    const result = await pool.query(`
      INSERT INTO users (id, full_name, photo_url, updated_at)
      VALUES ($1, $2, $3, CURRENT_TIMESTAMP)
      ON CONFLICT (id)
      DO UPDATE SET
        full_name = EXCLUDED.full_name,
        photo_url = EXCLUDED.photo_url,
        updated_at = CURRENT_TIMESTAMP
      RETURNING id, full_name as "fullName", photo_url as "photoURL"
    `, [id, fullName, photoURL]);

    return res.json({
      success: true,
      message: 'Profile updated successfully',
      user: result.rows[0]
    });
  } catch (error) {
    console.error('Profile update failed:', error);
    return res.status(500).json({ error: 'Failed to update profile' });
  }
});
```

Рисунок 4.35 – Код збереження ім’я та фото аватару профілю

Таким чином, забезпечена велика робота з збереженням даних та їх оновленням, що є частиною роботи з реляційною базою даних та реалізацією

CRUD операцій на сервері. Це є необхідністю при роботі з будь-яіким застосунком та невід’ємною його частиною.

4.3 Тестування вебзастосунку

Один з головних завершуючих етапів розробки є тестування застосунку. Пов’язано це з потребою коректної роботи та запобігання помилок у системі, що наявна та забезпечення їх виправлення. Результати тестування вебзастосунку наявні в таблицях 4.1–4.4 для відображення успішних сценаріїв тестування системи.

Таблиця 4.1 – Тест-кейс «Редагування профілю»

Елемент прецеденту	**Опис**
Назва прецеденту	Редагування особистих даних профілю користувача
Діючі особи	Авторизований користувач (клієнт/майстер/адмін), Система
Мета	Актуалізація імені та фотографії профілю в базі даних
Передумова	Користувач авторизований в системі та знаходиться в особистому кабінеті.
Успішний сценарій	1) користувач переходить на вкладку «Редагувати профіль» (TabsContent value="profile"); 2) вносить зміни в поле «Ім'я та прізвище» (fullName) або вказує нове посилання на фото (photoURL); 3) натискає кнопку «Зберегти зміни»; 4) система валідує дані, виконує PUT-запит до БД PostgreSQL через API (/api/v1/users/:id); 5) система повертає статус 200 OK, оновлює інтерфейс та показує іконку успіху (CheckCircle2).
Результат	Зміни профілю успішно збережено в таблиці users.
Розширення (Помилки)	Обов'язкове поле «Повне ім'я» залишено порожнім – система блокує відправку форми на рівні HTML5 (required) або API повертає помилку 400 Bad Request. – виводиться повідомлення про помилку – результат є зміни в БД не внесені.

Таблиця 4.2 – Тест-кейс «Завантаження фото профілю»

Елемент прецеденту	**Опис**
Назва прецеденту	Завантаження локального файлу фотографії профілю
Діючі особи	Авторизований користувач, Система
Мета	Зміна аватарки шляхом завантаження медіафайлу з пристрою
Передумова	Користувач відкрив сторінку налаштувань профілю.
Успішний сценарій	1) користувач натискає кнопку «Завантажити» (Upload); 2) система відкриває діалогове вікно вибору файлу на пристрої; 3) користувач обирає зображення формату .jpg/.png розміром до 1 МБ; 4) система створює тимчасовий локальний URL (URL.createObjectURL) для миттєвого відображення в компоненті Avatar; 5) після збереження форми файл надсилається на сервер, а згенерований лінк записується в PostgreSQL;
Результат	Нове фото відображається в інтерфейсі, шлях оновлено.
Розширення (Помилки)	Розмір обраного файлу перевищує 1 МБ – система перевіряє file.size у клієнтському коді – спливає вікно з попередженням: «Файл занадто великий! Максимум 1МБ.» – процес завантаження переривається – результат попереднє фото залишається незмінним.

Таблиця 4.3 – Тест-кейс «Редагування розкладу»

Елемент прецеденту	**Опис**
Назва прецеденту	Управління робочим розкладом майстрів салону
Діючі особи	Користувач (Менеджер / Адміністратор), Система
Мета	Встановлення або зміна робочих годин та вихідних днів для спеціалістів
Передумова	Користувач має роль Admin або Owner та перейшов у розділ керування розкладом.

Кінець таблиці 4.3

Успішний сценарій	1) користувач обирає потрібного спеціаліста з бокової панелі; 2) система робить GET-запит та відображає поточну сітку розкладу на 7 днів із бази даних; 3) користувач за допомогою чекбоксів вмикає/вимикає робочі дні та змінює час (workStart, workEnd). 4) натискає кнопку «Зберегти зміни». 5) система робить пакетний PUT-запит, виконуючи операцію UPSERT (ON CONFLICT DO UPDATE) в таблиці specialist_schedules. 6) екран оновлюється, відображаючи статус «Збережено!».
Результат	Нові часові інтервали майстра записані в PostgreSQL, клієнти тепер бачать оновлений час для запису.
Розширення (Помилки)	Збій з'єднання з базою даних під час збереження – бекенд викликає ROLLBACK для транзакції у випадку помилки SQL – система виводить повідомлення «Transaction failed, schedule not saved» – результат старі дані розкладу не пошкоджено, зміни не застосовано.

Таблиця 4.4 – Тест-кейс «Перегляд аналітики застосунку»

Елемент прецеденту	**Опис**
Назва прецеденту	Перегляд та аналіз бізнес-показників (Дашборд)
Діючі особи	Користувач (Власник / Адміністратор), Система
Мета	Отримання актуальної інформації про дохід, відвідуваність та популярність майстрів
Передумова	Користувач авторизований, має права доступу Admin або Owner.
Успішний сценарій	1) користувач переходить на головну сторінку Dashboard (/dashboard); 2) система перевіряє роль користувача через API профілю; 3) система паралельно викликає ендпоінт аналітики /api/v1/dashboard/stats; 4) PostgreSQL на сервері миттєво рахує суму доходів, середній чек та групує топ-5 майстрів; 5) система рендерить картки з метриками та будує інтерактивні графіки Recharts.

Кінець таблиці 4.4

Результат	Користувач бачить точні фінансові показники салону в реальному часі.
Розширення (Помилки)	Користувач не має адміністративних прав (роль User або Guest) – система обчислює значення canAccess = false – екран очищується, і замість графіків рендериться інтерфейс «Access Denied» (Доступ заборонено) – активується кнопка «Return to Account» для повернення у звичайний кабінет– результат конфіденційні фінансові дані не завантажуються в браузер.

Згідно з виконаним тестуванням застосунку можна зазначити про коректність його роботи та приступити до релізу даного застосунку. Функціонал доданий до застосунку відповідає цілі його створення в контексті сфери використання у бізнесі салону краси.

Висновки до розділу 4

Наведено повний огляд розробленого застосунку салону краси з використанням за основу інформаційної системи управління застосунком. Розглянуто основний функціонал застосунку та наведено короткий код для відображення логіки його роботи та зазначення переваг використаних технологій розробки.

Приділено значну увагу розробленої частини керування, такий як: керування персоналом (його розкладом) керування нарахування бонусів користувачам, керування аналітикою та звітністю вебзастосунку салону краси. Виконано огляд частини сповіщень про нарахування бонусів та звернення до чату підтримки з надходженням повідомлень до адміністратора салону. Користувацький інтерфейс застосунку є зручним та лаконічним, що відповідає потребам клієнту даної сфери.

Виконано тестування застосунку та зазначено, що робота вебзастосунку є більш ніж коректною, дані зберігаються та надсилаються запити на базу даних. Усі CRUD операції виконуються згідно з підтримки реляційною бази даних, що є необхідним для даного вебзастосунку.

ВИСНОВКИ

У ході виконання кваліфікаційної бакалаврської роботи створено вебзастосунок салону краси, що є сучасним рішенням для автоматизації бізнес процесів за умовою використання інформаційної системи управління за прикладом CRM системи, що є гарним рішенням для малого та середнього бізнесу надання послуг у сфері краси.

Задля досягнення мети поставленого напочатку, виконано наступні завдання:

− проаналізовано застосунків-аналогів;

− розроблена специфікації вимог до програмного забезпечення;

− спроєктовано архітектуру застосунку;

− розроблено дизайн та макет застосунку;

− розроблено front-end частину вебзастосунку на базі технології Vue.js, React.js;

− розроблено back-end частину вебзастосунку на базі технології Next.js, PostgreSQL, Python;

− виконано тестування розробленого програмного забезпечення.

Дослідження предметної області розробки допомогли у вирішенні завдань та реалізації конкурентоспроможного рішення вебзастосунку для салону краси на сучасному ринку розробки та сприяли складанню специфікації вимог задля сприяння покращення та вирішення недоліків аналогових застосунків для покращення застосунку. Результати дослідження розглянуто на XXVIII Всеукраїнській науково-практичній конференції «МОГИЛЯНСЬКІ ЧИТАННЯ – 2025:досвід та тенденції розвитку суспільства в Україні: глобальний, національний тарегіональний аспекти» (додаток Б).

Виконано огляд інструментарію для додержання виконання поставлений завдань та розроблено UML-діаграми для відображення функціональних та архітектурних рішень в розробці. Протестовано вебзастосунок салону краси, що продемонструвало гарні сценарії виконання його роботи. Результатом виконання роботи став вебзастосунок салону краси з інформаційною системою управління.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Beauty Pro : вебсайт. URL: https://beautyprosoftware.com/ (дата звернення: 18.04.2026).

2. Altegio : вебсайт. URL: https://alteg.io/en/ (дата звернення: 19.04.2026).

3. EasyWeek : вебсайт. URL: https://easyweek.io/ (дата звернення: 01.05.2026).

4. Kristiadi S., Putra R. P. Prototype development for online reservation system in barbershop and salon industry // *2019 International Conference on Information Management and Technology (ICIMTech)*. 2019. P. 20 – 21. DOI: 10.1109/ICIMTech.2019.8843836.

5. Wei J. T., Lee M. C., Chen H. K., Wu H. H. Customer relationship management in the hairdressing industry: an application of data mining techniques // *Expert Systems with Applications*. 2013. Vol. 40, No. 18. P. 128–140. DOI: 10.1016/j.eswa.2013.07.053.

6. Blancaflor E., Lacson B., Balajadia J., Hilario M. Beautyhare: a design of a web-based management system for salon services // *Proceedings of the 7th International Conference on Computers in Management and Business*. 2024. P. 41–47. DOI: 10.1145/3647782.3647789.

7. Sutarman A., Ariyanti M., Suhariyanto S. Improving a quality of services through information system model of academic services based on customer relationship management at university // *International Journal of Engineering and Advanced Technology*. 2019. Vol. 8, No. 6. P. 250–255. DOI: 10.35940/IJEAT.F8561.088619.

8. Krishna B., Kumar S., Singh A. Enhancing SEO in single-page web applications in contrast with multi-page applications // *IEEE Access*. 2024. Vol. 12. P. 197–204. DOI: 10.1109/ACCESS.2024.3355740.

9. Next.js documentation : website. URL: https://nextjs.org/docs (last excess: 16.04.2026).

10. Gao Z., Bird C., Barr E. T. To type or not to type: quantifying detectable bugs in JavaScript // *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 2017. P. 158–169. DOI: 10.1109/ICSE.2017.75.

11. Tilkov S., Vinoski S. Node.js: using JavaScript to build high-performance network programs // *IEEE Internet Computing*. 2010. Vol. 14, No. 6. P. 80–83. DOI: 10.1109/MIC.2010.145.

12. Ferraiolo D. F., Sandhu R., Gavrila S., Kuhn D. R., Chandramouli R. Proposed NIST standard for role-based access control // *ACM Transactions on Information and System Security*. 2001. Vol. 4, No. 3. P. 224–274. DOI: 10.1145/501978.501980.

13. PostgreSQL documentation : website. URL: https://www.postgresql.org/docs/ (last excess: 18.02.2026).

14. Zulkifli A. Accelerating database efficiency in complex IT infrastructures: advanced techniques for optimizing performance, scalability, and data management in distributed systems // *ResearchGate*. 2024. URL: https://www.researchgate.net/publication/386218911 (date of application: 16.02.2026).

15. Docker Compose documentation : website. URL: https://docs.docker.com/compose/ (last excess: 19.02.2026).

16. GitHub Actions documentation : website. URL: https://docs.github.com/en/actions (last excess: 20.02.2026).

17. Booch G., Rumbaugh J., Jacobson I. *The Unified Modeling Language User Guide*. 2nd ed. Upper Saddle River : Addison-Wesley Professional, 2005. 504 p. DOI: https://doi.org/10.5555/1076335.

18. Fowler M. *Patterns of Enterprise Application Architecture*. Boston : Addison-Wesley Professional, 2002. 560 p. DOI: https://doi.org/10.5555/577055.

19. Van Rossum G., Drake F. L. *The Python Language Reference Manual*. Bristol : Network Theory Ltd., 2011. 140 p. DOI: https://doi.org/10.5555/2125553.

20. Flanagan D. *JavaScript: The Definitive Guide*. 7th ed. Sebastopol : O'Reilly Media, 2020. 704 p. ISBN: 978-1-491-95202-3.

21. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes: Lessons learnt from three generations of container management systems with Kubernetes. *Queue*. 2016. Vol. 14, No. 1. P. 70–93. DOI: https://doi.org/10.1145/2898442.2898444.

ДОДАТОК А

Лістинг коду дошки звітності python

```
import io
import os
import datetime
import pandas as pd
from fastapi import FastAPI, Depends, HTTPException
from fastapi.responses import StreamingResponse
from sqlalchemy import create_engine, Column, Integer, String, Float, DateTime, text
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, Session

app = FastAPI(title="Beauty Book MIS Analytics API (PostgreSQL)")

POSTGRES_lada = os.getenv("DB_USER", "lada_admin")
POSTGRES_LADA04 = os.getenv("DB_PASSWORD", "vK9_sP24_Lada04")
POSTGRES_HOST = os.getenv("DB_HOST", "db-beautybook-prod.c123456789.eu-central-
1.rds.amazonaws.com")
POSTGRES_PORT = os.getenv("DB_PORT", "5432")
POSTGRES_DB = os.getenv("DB_NAME", "beauty_book_db")

DATABASE_URL =
f"postgresql://{POSTGRES_lada}:{POSTGRES_LADA04}@{POSTGRES_HOST}:{POSTGRES_PORT}/{POSTGRES
_DB}"

engine = create_engine(
    DATABASE_URL,
    pool_size=10,
    max_overflow=20,
    pool_pre_ping=True
)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

class OrderRecord(Base):
    __tablename__ = "orders"

    id = Column(Integer, primary_key=True, index=True)
    specialist_id = Column(Integer, nullable=False, index=True)
    specialist_name = Column(String(100), nullable=False)
```

```
    service_name = Column(String(150), nullable=False)
    price = Column(Float, nullable=False)
    # Використовуємо TIMESTAMP з часовим поясом для серверів Enterprise-рівня
    created_at = Column(DateTime(timezone=True), server_default=text('NOW()'))

Base.metadata.create_all(bind=engine)

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

def calculate_financials(df: pd.DataFrame) -> dict:
    if df.empty:
        return {}

    total_revenue = float(df['price'].sum())

    df['commission'] = df['price'] * 0.3

    df['tax'] = df['commission'] * 0.2

    df['net_pay'] = df['commission'] - df['tax']

    spec_group = df.groupby(['specialist_id', 'specialist_name']).agg(
        count=('id', 'count'),
        revenue=('price', 'sum'),
        commission=('commission', 'sum'),
        tax=('tax', 'sum'),
        net_specialist_pay=('net_pay', 'sum')
    ).reset_index()

    specialist_table = spec_group.to_dict(orient='records')



    total_commission = float(df['commission'].sum())
    total_taxes = float(df['tax'].sum())

    material_costs = total_revenue * 0.1
```

```
    net_profit = total_revenue - total_commission - total_taxes - material_costs

    def categorize(name):
        if 'Hair' in name: return 'Hair'
        if 'Nail' in name: return 'Nails'
        if 'Facial' in name: return 'Skincare'
        return 'Other'

    df['category'] = df['service_name'].apply(categorize)
    cat_group = df.groupby('category').size().to_dict()
    pie_data = [{'name': k, 'value': v} for k, v in cat_group.items()]

    return {
        "total_revenue": total_revenue,
        "total_commission": total_commission,
        "total_taxes": total_taxes,
        "material_costs": material_costs,
        "net_profit": net_profit,
        "specialist_table": specialist_table,
        "pie_data": pie_data
    }

@app.get("/api/analytics/dashboard")
def get_dashboard_analytics(db: Session = Depends(get_db)):
    # Завантаження сирих даних з PostgreSQL
    query_result = db.query(OrderRecord).all()
    if not query_result:
        return {"message": "Дані відсутні"}

    data = [{
        "id": r.id,
        "specialist_id": r.specialist_id,
        "specialist_name": r.specialist_name,
        "service_name": r.service_name,
        "price": r.price
    } for r in query_result]

    df = pd.DataFrame(data)
    analytics_result = calculate_financials(df)
```

```
        analytics_result["trend_charts"] = [
            {"date": "Пн", "appointments": 12, "revenue": 4500},
            {"date": "Вт", "appointments": 19, "revenue": 7200},
            {"date": "Ср", "appointments": 15, "revenue": 5800},
            {"date": "Чт", "appointments": 22, "revenue": 9100},
            {"date": "Пт", "appointments": 28, "revenue": 12000},
            {"date": "Сб", "appointments": 35, "revenue": 16500},
            {"date": "Нд", "appointments": 14, "revenue": 5000},
        ]

        return analytics_result

    @app.get("/api/analytics/download-report")
    def download_large_report():

         sql_query = "SELECT id, specialist_id, specialist_name, service_name, price,
created_at FROM orders"
        chunk_size = 5000
        def csv_streamer():
            output = io.StringIO()
            first_chunk = True

            with engine.connect() as conn:
               for chunk in pd.read_sql_query(sql_query, con=conn, chunksize=chunk_size):

                    chunk['commission'] = chunk['price'] * 0.3
                    chunk['tax'] = chunk['commission'] * 0.2
                    chunk['net_pay'] = chunk['commission'] - chunk['tax']
                    chunk['material_cost'] = chunk['price'] * 0.1
                    chunk['net_profit_per_order'] = chunk['price'] - chunk['commission']
- chunk['tax'] - chunk['material_cost']

                                    if 'created_at' in chunk.columns:
                                                        chunk['created_at']     =
pd.to_datetime(chunk['created_at']).dt.strftime('%Y-%m-%d %H:%M:%S')

                                    chunk.to_csv(output, index=False, header=first_chunk)
                    first_chunk = False

                    yield output.getvalue()
```

```
            output.seek(0)
            output.truncate(0)

    return StreamingResponse(
        csv_streamer(),
        media_type="text/csv",
                                    headers={"Content-Disposition":       "attachment;
filename=beauty_book_postgres_report.csv"}
    )
```

ДОДАТОК Б

Апробація

Бондаренко С. В., Ковшун Л. Д. Інформаційна система управління, інтегрована у вебзастосунок мережі салону краси: «МОГИЛЯНСЬКІ ЧИТАННЯ – 2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» XXVIII Всеукраїнська науково-практична конференція ТЕЗИ ДОПОВІДЕЙ ТЕХНІЧНІ НАУКИ Миколаїв, 10 – 14 листопада 2025 року м. Миколаїв, Україна ЧНУ ім. Петра Могили с. 145

УДК 004.42

Бондаренко С. В.
викладачка кафедри інженерії програмного забезпечення
Чорноморський національний університет ім. Петра Могили, м. Миколаїв, Україна
Ковшун Л. Д.
здобувачка першого (бакалаврського) рівня вищої освіти,
Чорноморський національний університет імені Петра Могили, м. Миколаїв, Україна

ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ, ІНТЕГРОВАНА У ВЕБЗАСОСУНОК МЕРЕЖІ САЛОНІВ КРАСИ

У сучасних умовах розвитку ринку послуг краси управління мережею салонів вимагає ефективної організації бізнес-процесів, точного планування та контролю діяльності кожного структурного підрозділу. Зростання кількості клієнтів, розширення асортименту послуг і необхідність підтримання єдиних стандартів якості зумовлюють потребу у створенні спеціалізованих інформаційних систем управління, які забезпечують централізований облік, координацію й контроль роботи салонної мережі.

Сучасна інформаційна система управління мережею салонів краси [1, 2] має на меті автоматизувати ключові напрями діяльності підприємства: запис клієнтів, ведення бази даних, управління персоналом, облік матеріалів, фінансів та контролю показників ефективності. Завдяки централізованому зберіганню даних система дозволяє забезпечити єдину політику обслуговування клієнтів, узгоджене управління філіями та швидке прийняття управлінських рішень.

Важливою перевагою інформаційної системи є можливість автоматизації процесів бронювання послуг і робочих місць, що усуває дублювання записів і помилки під час планування графіків. Інтеграція із модулем управління [2, 3] персоналом дозволяє формувати робочі розклади, контролювати навантаження майстрів, відстежувати продуктивність працівників і вчасно реагувати на зміни попиту [4].

Окрему роль відіграє модуль обліку матеріалів і фінансів, який забезпечує контроль залишків, планування закупівель, розрахунок витрат і прибутків. Це сприяє оптимізації витрат та підвищенню фінансової прозорості діяльності підприємства.

Модуль звітності та аналітики [5] системи формує показники ефективності (KPI) для керівництва, надає дані про завантаження персоналу, середній чек, кількість відвідувачів, динаміку доходів і