Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Веб-застосунок для онлайн-запису клієнтів

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Роман КАНЦІР
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Дар’я ТИМОШЕНКО
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Канцір Роману Богдановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Веб-застосунок для онлайн-запису клієнтів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) документація Next.js; 2) документація TypeScript; 3) документація PocketBase; 4) документація Tailwind CSS і CSS; 5) документація shadcn/ui; 6) документація React; 7) середовище розробки Cursor.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз предметної області;
2) аналіз використовуваних технологій;
3) програмна реалізація;
4) інструкція користувача;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 13 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	27.05.25-30.05.25	
2	Вибір технології розробки та інструменталь-	31.05.25-03.06.25	
	них засобів		
3	Розробка алгоритмічного забезпечення	04.06.25-06.06.25	
4	Розробка та відлагодження програмного	07.06.25-09.06.25	
	забезпечення		
5	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
6	Подання кваліфікаційної роботи керівникові та	12.06.25-13.06.25	
	її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Дар'я ТИМОШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 91 с., 22 рис., 1 табл., 2 дод., 30 джерел.

ООП, МІГРАЦІЯ, АВТОРИЗАЦІЯ, NEXT.JS, POCKETBASE, SPA, TYPESCRIPT, TAILWIND CSS, CRUD, SSR.

Метою кваліфікаційної роботи є створення вебзастосунку для онлайн-запису клієнтів. Розроблена система дозволяє автоматизувати процес бронювання послуг без потреби в реєстрації клієнтів, що значно спрощує взаємодію з сервісом як для бізнесу, так і для користувача.

Основні функціональні можливості вебзастосунку включають: онлайн-запис, повідомлення майстру про нові записи, сторінку для майстра з можливістю налаштування графіку роботи та перегляду записів у форматі календаря, а також адмін панель для створення та керування компанією, додавання майстрів і сервісів, перегляду клієнтів, аналітики та календаря з усіма записами.

У ході виконання кваліфікаційної роботи було проаналізовано існуючі рішення систем онлайн-запису, їхні переваги та недоліки. На основі цього аналізу сформовано перелік функціональних та технічних вимог до майбутнього продукту.

Для реалізації проєкту було використано сучасний стек технологій: Next.js як фронтенд-фреймворк, мову TypeScript для типобезпечної розробки, Tailwind CSS для швидкої стилізації інтерфейсу, а також PocketBase – як легкий та зручний бекенд-сервіс, що надає REST API, авторизацію та базу даних.

ABSTRACT

Bachelor’s thesis: 91 pages, 22 figures, 1 tables, 2 appendices, 30 sources.

OOP, MIGRATION, AUTH, NEXT.JS, POCKETBASE, SPA, TYPESCRIPT, TAILWIND CSS, CRUD, SSR.

The major goal of this thesis is to design and develop a web application for online customer booking, aimed at automating the service reservation process and enhancing the user experience. The proposed solution enables customers to book services online without the need for registration, which significantly simplifies their interaction with the system and improves accessibility for businesses.

The developed web application provides the following core functionalities: the ability to make bookings online, receive automated notifications, manage individual work schedules, and view appointments in a calendar interface. Additionally, the system includes a dedicated admin panel for company management, where administrators can add services and masters, view clients, access analytical data, and manage all bookings from a centralized calendar.

Throughout the thesis, existing solutions for online booking were reviewed and analyzed. Their strengths and weaknesses were evaluated, which allowed for the formulation of a clear set of functional and technical requirements for the developed product.

In order to implement the system, a modern and efficient technology stack was chosen: Next.js as the main frontend framework, TypeScript for type-safe development, Tailwind CSS for fast and responsive user interface styling, and PocketBase as a lightweight backend providing RESTful API, authentication, and data storage capabilities.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ACL – список контролю доступу (англ., Access Control List)

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

BaaS – бекенд як сервіс (англ., Backend as a Service)

CAPTCHA – автоматизований тест для розрізнення людей і ботів (англ., Completely Automated Public Turing test to tell Computers and Humans Apart)

CI/CD – неперервна інтеграція / неперервне розгортання (англ., Continuous Integration / Continuous Deployment)

CRM – система управління взаємовідносинами з клієнтами (англ., Customer Relationship Management)

CRUD – створення, читання, оновлення, видалення (англ., Create / Read / Update / Delete)

CSS – каскадні таблиці стилів (англ., Cascading Style Sheets)

CSR – клієнтська генерація сторінок (англ., Client-Side Rendering)

DRY – не повторюй себе (англ., Don't Repeat Yourself)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

HTTP – протокол передавання гіпертексту (англ., Hypertext Transfer Protocol)

HTTPS – захищений протокол передавання гіпертексту (англ., Hypertext Transfer Protocol Secure)

ICS – формат календарних файлів (англ., iCalendar file format)

IDE – інтегроване середовище розробки (англ., Integrated Development Environment)

ISR – інкрементна генерація статичних сторінок (англ., Incremental Static Regeneration)

JSON – об’єктна нотація JavaScript (англ., JavaScript Object Notation)

JSX – JavaScript + XML – розширення синтаксису для React (англ., JavaScript XML)

JWT – токен на основі JSON (англ., JSON Web Token)

MVP – мінімально життєздатний продукт (англ., Minimum Viable Product)

OAuth2 – протокол відкритої авторизації 2.0 (англ., Open Authorization 2.0)

REST – передача стану представлення (англ., Representational State Transfer)

SaaS – програмне забезпечення як сервіс (англ., Software as a Service)

SDK – набір засобів розробника (англ., Software Development Kit)

SEO – пошукова оптимізація (англ., Search Engine Optimization)

SMS – служба коротких повідомлень (англ., Short Message Service)

SPA – односторінковий застосунок (англ., Single Page Application)

SQL – мова структурованих запитів (англ., Structured Query Language)

SSG – статична генерація сайту (англ., Static Site Generation)

SSR – серверна генерація сторінок (англ., Server-Side Rendering)

TLS – протокол безпеки транспортного рівня (англ., Transport Layer Security)

UI – інтерфейс користувача (англ., User Interface)

URL – уніфікований локатор ресурсу (англ., Uniform Resource Locator)

UTC – всесвітній координований час (англ., Coordinated Universal Time)

UX – досвід користувача (англ., User Experience)

XML – розширювана мова розмітки (англ., eXtensible Markup Language)

ВСТУП

У сучасному цифровому середовищі все більше власників малого бізнесу, салонів краси та незалежних майстрів прагнуть спростити та автоматизувати процес взаємодії з клієнтами. Це особливо актуально для тих, хто працює самостійно та просуває свої послуги Instagram або інші соцмережі, не маючи при цьому власного сайту чи CRM-системи. Онлайн-запис на послуги досі часто здійснюється через ручне листування або телефонні дзвінки, що призводить до втрати клієнтів, плутанини в розкладі та додаткового навантаження на майстра чи адміністратора.

Хоча існує широкий спектр інструментів для онлайн-запису, більшість з них орієнтовані на середній та великий бізнес, що робить їх менш придатними для малих компаній або індивідуальних підприємців. Ці платформи часто вимагають складного налаштування, чи обов'язкової реєстрації клієнтів або мають високу вартість, що може бути обтяжливим для невеликих компаній чи майстрів, які працюють на себе. Через це багато майстрів або ФОП змушені й досі вести запис вручну – через нотатки, месенджери чи по телефону, що займає багато часу та підвищує ймовірність помилок.

У зв’язку з цим виникає потреба у простому, доступному й зрозумілому вебзастосунку, який би дозволив майстрам легко організувати власний запис без залучення сторонніх сервісів чи складних інтеграцій. Актуальність такого підходу полягає в його доступності для найширшого кола користувачів – особливо для тих, хто не має технічного бекграунду або бажає зменшити витрати на впровадження цифрових рішень у своїй діяльності.

Метою кваліфікаційної роботи є створення вебзастосунку для онлайн-запису клієнтів, який забезпечить простий та зрозумілий функціонал, що спростить взаємодію з клієнтами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Процес онлайн-запису

Онлайн-запис – це процес електронної реєстрації клієнтів на послуги через вебзастосунки або мобільні платформи без необхідності фізичної присутності чи телефонних дзвінків. Клієнт самостійно обирає послугу, спеціаліста, зручну дату та час, що значно спрощує процес взаємодії з бізнесом і дозволяє заощадити час обом сторонам.

За останні роки популярність таких систем зросла, оскільки багато компаній прагнуть автоматизувати рутинні процеси та покращити якість обслуговування. Особливу зацікавленість в онлайн-записі проявляють представники малого бізнесу – салони краси, барбершопи, студії манікюру, а також незалежні майстри, які часто не мають власного сайту або спеціалізованої CRM-системи [1].

Ще однією характерною рисою цієї сфери є високі вимоги до простоти інтерфейсу для кінцевого користувача [1, 2]. Багато клієнтів віддають перевагу сервісам, які не змушують проходити реєстрацію або завантажувати додаткові застосунки. Наявність функціонального, адаптивного інтерфейсу та мінімізація кількості дій для запису стають визначальними факторами успішного впровадження таких систем.

До переваг онлайн-запису відносять:

- оптимізацію навантаження на адміністративний персонал;
- зменшення кількості помилок у комунікації;
- підвищення рівня обслуговування та задоволеності клієнтів;
- можливість автоматизації нагадувань та повідомлень.

Такі переваги роблять онлайн-запис ефективним інструментом для підвищення конкурентоспроможності малого бізнесу в сфері послуг.

1.2 Аналіз існуючих рішень

На сьогодні існує велика кількість способів організації запису клієнтів на послуги, кожен із яких має свої переваги та недоліки. Від класичних ручних способів до сучасних цифрових платформ – підходи суттєво відрізняються та вибір залежить від характеру бізнесу, рівня цифрової підготовки та наявних ресурсів [2]. Далі розглянуто основні типові рішення, що застосовуються для управління процесом запису клієнтів.

1.2.1 Ручне ведення записів

На початкових етапах розвитку бізнесу організація запису клієнтів здебільшого здійснюється вручну. До найпоширеніших традиційних методів відносять паперові журнали, нотатки, блокноти, а також фіксація домовленостей через телефонні дзвінки, електронну пошту чи повідомлення в месенджерах [3]. Ці методи виглядають зручними, на перший погляд завдяки своїй простоті, оскільки не потребують додаткових витрат на програмне забезпечення чи спеціальне навчання персоналу.

Ручне ведення записів може залишатися ефективним у випадку невеликої кількості клієнтів, чи незначного робочого навантаження. Наприклад, для незалежного майстра, який приймає кілька клієнтів на день, достатньо фіксувати записи через дзвінки або повідомлення. Водночас із ростом кількості клієнтів та розширенням спектра послуг, такий підхід починає створювати серйозні проблеми.

Серед основних недоліків ручного ведення записів є висока ймовірність помилок через людський фактор, плутанини в графіку та некоректна передача інформації. Частими є випадки подвійного бронювання одного часу для різних клієнтів, або помилковий запис іншої дати чи послуги. Відсутність централізованої системи обліку ускладнює обробку даних, аналіз завантаженості майстрів та планування роботи.

Крім цього, ручний облік не дозволяє автоматизувати нагадування клієнтам про майбутні записи. Це збільшує ризик неявки клієнтів, що, призводить до фінансових втрат для бізнесу. Також ручна робота збільшує навантаження на адміністративний персонал, тому працівникам доводиться витрачати значний обсяг часу на відповіді в месенджерах і телефонні розмови, що знижує загальну продуктивність [3].

У сучасних умовах клієнти очікують від бізнесу можливості швидкого та зручного запису через Інтернет, у будь-який час доби, без необхідності дзвонити або чекати відповіді. Відсутність подібної можливості негативно впливає на імідж компанії, створює враження застарілості її внутрішніх процесів та знижує конкурентоспроможність.

Таким чином, хоча ручне ведення записів історично є простим і безкоштовним рішенням для малого бізнесу, в умовах сучасного ринку він має ряд обмежень, що стримують розвиток компанії та можуть призводити до втрати клієнтів. Це обґрунтовує необхідність впровадження сучасних електронних систем для автоматизації процесу запису.

1.2.2 Використання Google Calendar

Одним із найпоширеніших інструментів для організації розкладу та запису клієнтів є Google Calendar – безкоштовний вебсервіс від компанії Google, що призначений для керування подіями та нагадуваннями. Його популярність обумовлена простотою у використанні, доступністю з будь-якого пристрою та можливістю синхронізації з іншими сервісами Google [4].

Для малого бізнесу чи незалежних майстрів, які не має доступу до спеціалізованих CRM-систем, Google Calendar часто виступає першим кроком у напрямку цифрової організації записів. Наприклад, адміністратор салону чи незалежний майстер може самостійно записувати події вручну, призначаючи кожному клієнту окремий часовий інтервал.

Сервіс також дозволяє додавати описи послуг, контактну інформацію

клієнтів і встановлювати нагадування, що частково систематизує процес обліку.

Водночас, незважаючи на зручність, Google Calendar має низку обмежень, які не дозволяють використовувати його як повноцінний інструмент для управління записами в сфері послуг. Насамперед, сервіс не має вбудовані функції для вибору послуги, майстра чи вартості, або форми для онлайн-запису.

Усі ці параметри потрібно вносити вручну, що створює додаткове навантаження та підвищує ймовірність помилок. Крім того, Google Calendar не передбачає окремих ролей користувачів, тому клієнти не можуть самостійно здійснити запис без використання сторонніх рішень, таких як форми або додаткові сервіси [4].

Ще одним суттєвим недоліком Google Calendar є відсутність інтегрованого інтерфейсу для клієнтів. Тобто користувач не бачать доступних часових слотів і не мають можливості самостійно обирати зручний час для візиту без участі адміністратора.

Для реалізації такого функціоналу потрібно створювати окремі Google-форми, інтегрувати сторонні віджети або користуватися API, що потребує технічної компетенції. Таким чином, рішення втрачає свою простоту й зручність, особливо для невеликих компаній, які не мають часу чи ресурсів для таких налаштувань.

Загалом, Google Calendar (рисунок 1.1) може частково задовольняти базові потреби у веденні розкладу, однак не є ефективним рішенням для системного управління онлайн-записами. Його використання вимагає додаткових налаштувань, не підтримує гнучку логіку вибору послуг або майстрів і не забезпечує зручної взаємодії з клієнтом без технічної адаптації. Це підтверджує необхідність впровадження спеціалізованих рішень, які враховують особливості роботи малого бізнесу та забезпечують повний цикл обслуговування клієнта без залучення сторонніх інструментів.

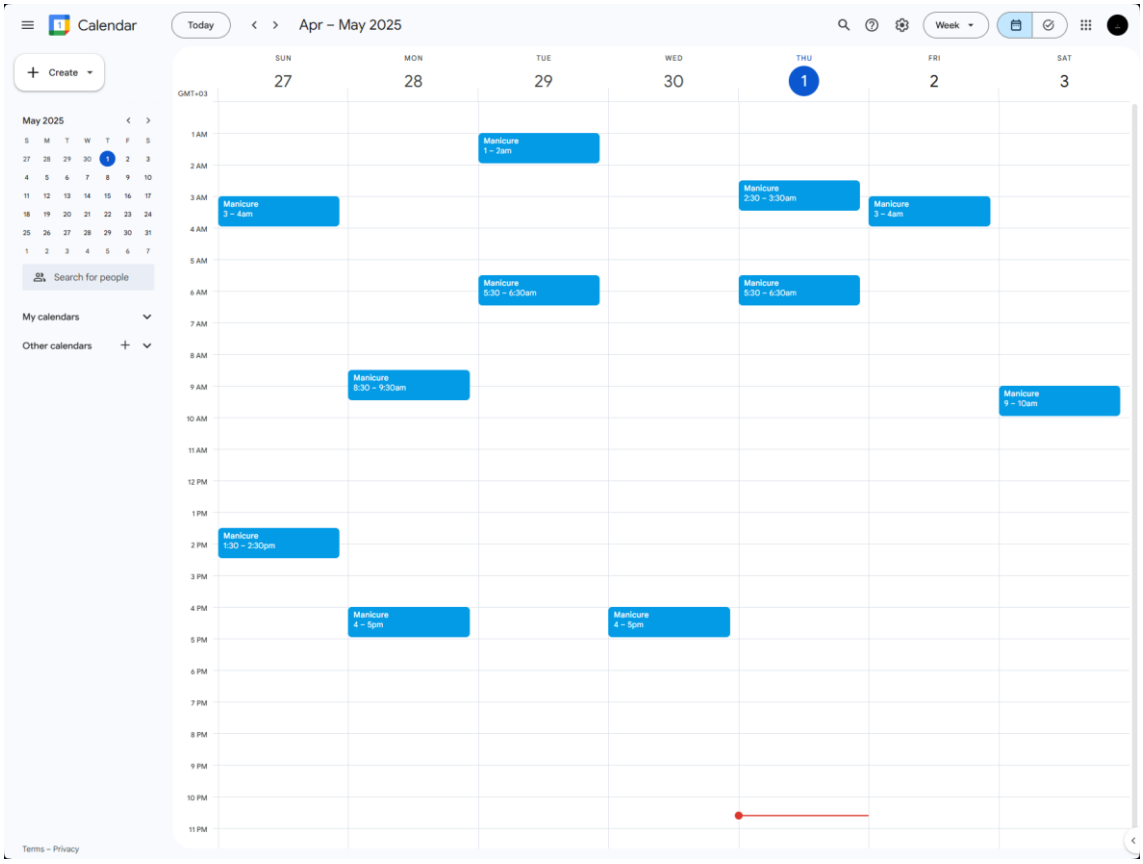

Рисунок 1.1 – Веб-застосунок Google Calendar

Серед недоліків сервісу Google Calendar слід визначити:

- відсутність інтегрованого вибору послуг, тривалості, ціни чи майстра;
- необхідність ручного введення всіх даних;
- низька адаптивність до бізнес-логіки малого сервісного бізнесу;
- обмеження у налаштуванні прав доступу та ролей користувачів, що ускладнює командну роботу;
- відсутність автоматичного нагадування про онлайн-запис для клієнта.

Google Calendar може частково задовольнити потреби окремих фахівців з невеликим клієнтським навантаженням, проте для системного ведення записів у сфері послуг зростаючого бізнесу його функціональності вже буде недостатньо.

1.2.3 Alteg.io

Серед популярних рішень, що спеціалізуються на сфері індустрії краси, значну увагу привертає сервіс Alteg.io – готова SaaS-платформа для автоматизації роботи салонів, студій та інших компаній, що надають послуги. Ця система поєднує функціональність онлайн-запису, CRM-системи, управління персоналом, обліку товарів та фінансового аналізу [5]. Для компаній з сформованими бізнес-процесами та постійним штатом працівників Alteg.io може виступати як комплексне рішення.

Платформа пропонує клієнтам можливість запису онлайн через окрему вебсторінку або віджет який можна розмістити на сайті або в соціальних мережах (рисунок 1.2). Адміністратор або власник бізнесу має доступ у кабінет для керування записами, клієнтською базою, графіками майстрів та фінансами. Система також надсилає SMS або email-нагадування клієнтам, забезпечує ведення аналітики та звітності, що підвищує ефективність управління [5].

Попри широкий функціонал, платформа Alteg.io має ряд обмежень, які роблять її менш зручним для невеликого бізнесу або незалежних майстрів. Насамперед сервіс повністю платний, тому безкоштовного тарифу не передбачено і це може стати фінансовим бар’єром для початківців або тих, хто лише тестує попит на свої послуги.

Щомісячна оплата без можливості ознайомлення з функціоналом до моменту придбання підписки є суттєвим обмеженням в контексті малого бізнесу. Таким чином, користувач змушений платити за комплексний набір можливостей, навіть якщо він фактично хоче використовувати лише базовий функціонал для онлайн-запису.

Крім того, закритість вихідного коду та відсутність можливості кастомізації інтерфейсу або бізнес-логіки робить Alteg.io менш гнучким у порівнянні з індивідуально розробленими рішеннями.

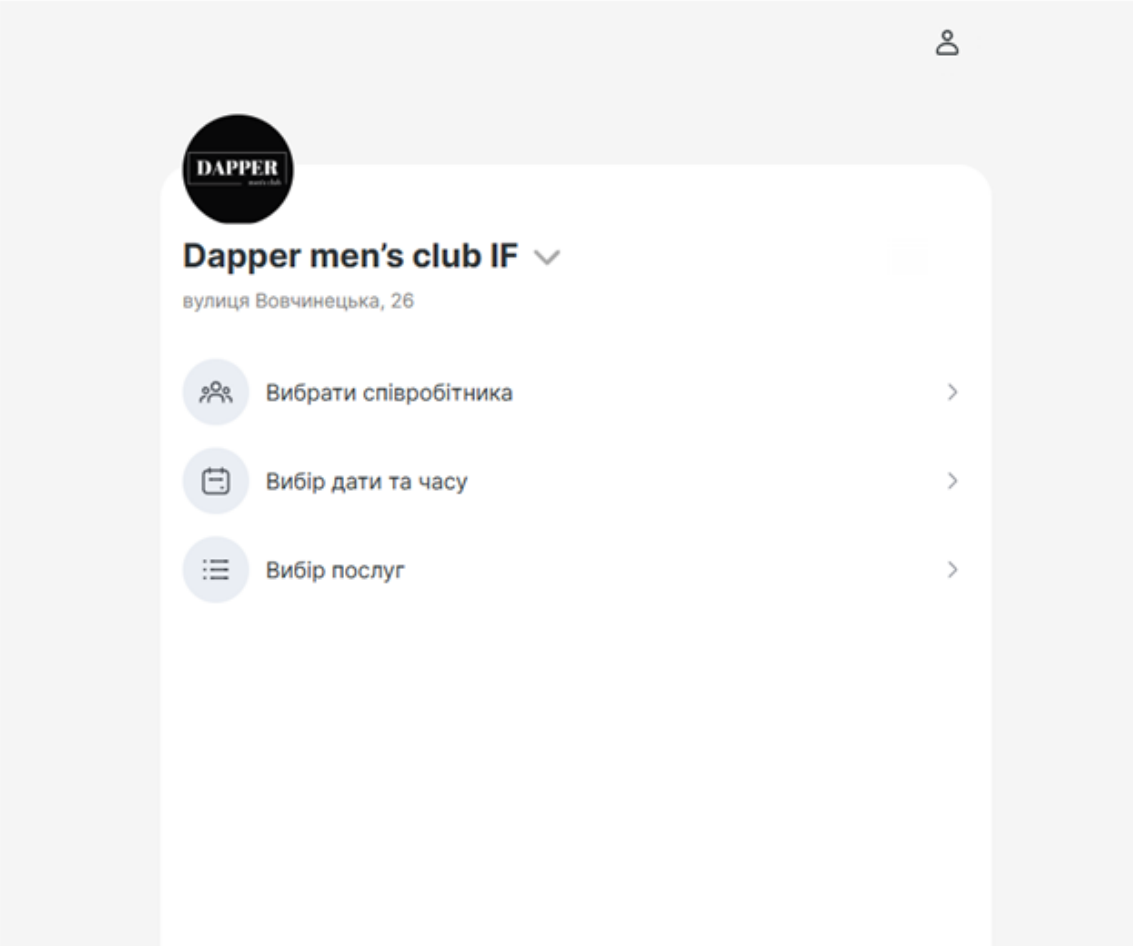

Рисунок 1.2 – Вебзастосунок Alteg.io

У підсумку це функціональна система для комплексного ведення бізнесу у сфері послуг, проте її складність, вартість та надлишковість функцій можуть бути зайвими для малого бізнесу, який шукає саме простий інструмент для онлайн-запису.

З недоліків цього сервісу слід відзначити:

- відсутність безкоштовного тарифу, що робить сервіс недоступним для багатьох представників малого бізнесу;

- складність первинного налаштування, що може створити бар’єр для користувачів без технічної підготовки.

- надлишковість функціоналу для невеликих бізнесів, яким не потрібна повноцінна CRM-система.

Alteg.io є більш доцільним вибором для середніх і великих компаній, які мають налагоджену внутрішню структуру та фінансову спроможність

регулярно сплачувати за підписку. Для невеликих студій або незалежних майстрів це рішення часто є невиправданим з погляду функціональності та вартісними параметрами.

1.2.4 Кастомні рішення

Окремою категорією у сфері організації онлайн-запису становлять індивідуально розроблені (кастомні) системи, що створюються під потреби конкретного бізнесу або окремого фахівця. У цьому випадку система онлайн-запису є складовою частиною або основою повноцінного вебсайту чи мобільного застосунку, створеного для салону краси, барбершопу або незалежного майстра. Такі рішення можуть повністю відповідати стилістиці бренду, враховувати унікальні особливості процесу обслуговування клієнтів і включати лише ті функції, які дійсно потрібні.

Основною перевагою кастомних рішень є їх гнучкість. На відміну від готових платформ з стандартизованим функціоналом, індивідуально розроблені системи дозволяють адаптувати інтерфейс, логіку запису, способи комунікації з клієнтом, системи оплати, інтеграції з соціальними мережами та внутрішній облік. Це особливо актуально для компаній, які прагнуть виділитися на фоні конкурентів та запропонувати клієнтам унікальний користувацький досвід (рисунок 1.3).

Разом із тим, індивідуальна розробка має і суттєві недоліки. Насамперед це висока вартість, яка містить оплату праці розробників, дизайнерів, тестувальників чи системних адміністраторів. Крім того, створення такого рішення займає значний час – від декількох тижнів до кількох місяців, залежно від складності та обсягу функціоналу. Після запуску необхідно забезпечити постійну технічну підтримку, оновлення та виправлення помилок, що також потребує ресурсів.

Ще одним ризиком є залежність від розробника або команди, яка створювала продукт. У разі припинення співпраці чи зміни технологій

підтримка системи може ускладнитись або вимагати суттєвих витрат. Для малого бізнесу це створює ризики та додаткове навантаження, які не завжди виправдовують очікуваний результат.

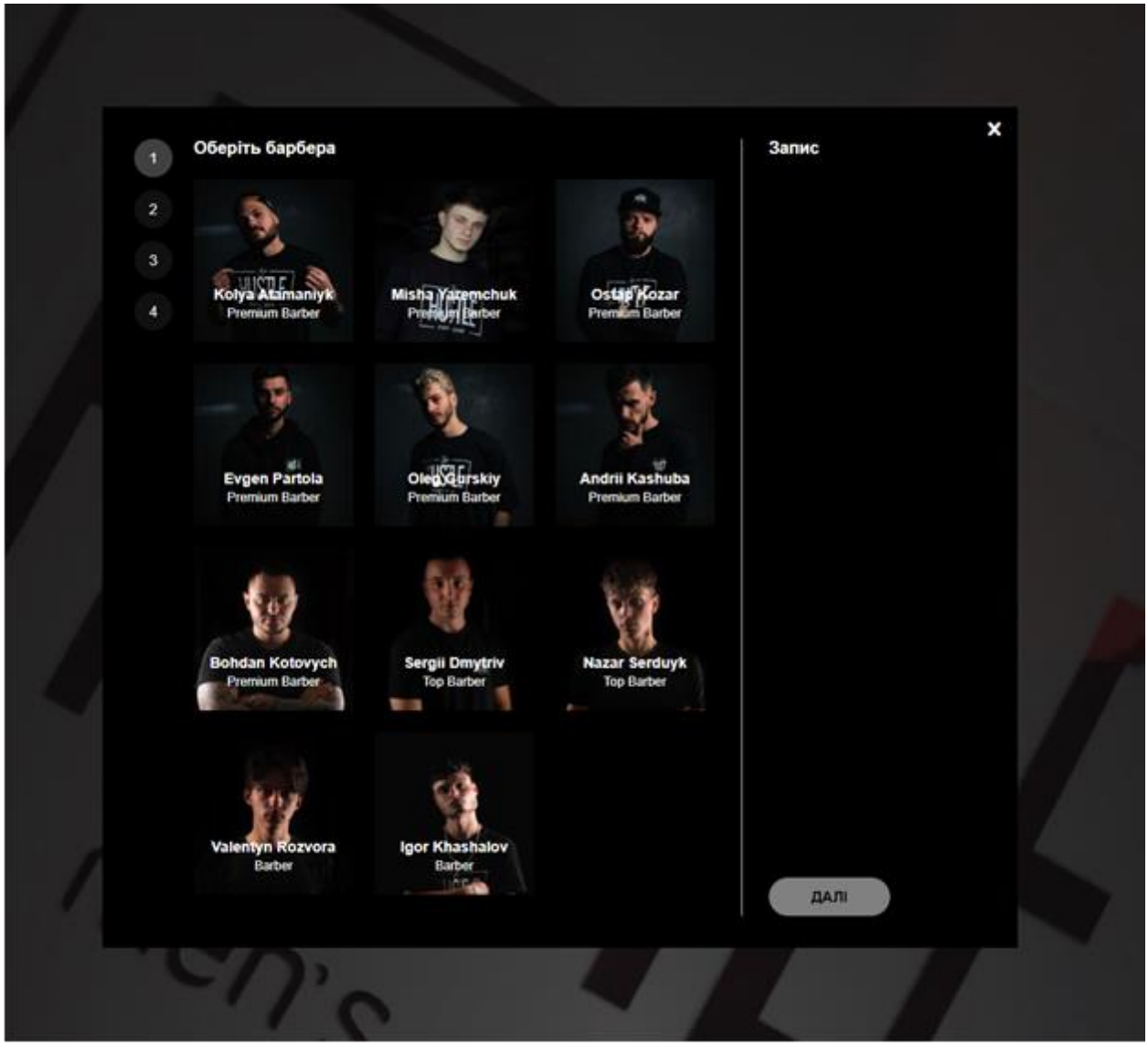

Рисунок 1.3 – Вигляд кастомного рішення для барбершопу

Таким чином, кастомні рішення можуть бути ефективним варіантом для середнього та великого бізнесу, які мають необхідні ресурси для повного циклу розробки, підтримки та масштабування.

Індивідуальна розробка має ряд недоліків:

- висока вартість створення та впровадження;
- тривалий час розробки;

- необхідність технічного супроводу після запуску;
- ризик залежності від однієї команди або фахівця;
- складність у масштабуванні або внесенні змін без участі розробника.
- обмеженість у швидкому реагуванні на зміну бізнес-потреб.

Кастомні рішення вимагають значного фінансування й технічної підтримки, що робить їх доцільними лише за наявності сталого бюджету та довгострокової стратегії розвитку. Для малого бізнесу та незалежних майстрів такі умови часто є недосяжними.

1.2.5 Підсумки аналізу існуючих рішень

Аналіз свідчить, що наявні підходи до організації онлайн-запису мають як переваги, так і низку суттєві обмеження, особливо для потреб малого бізнесу чи незалежного майстра. Ручне ведення записів через нотатки, дзвінки або месенджери залишається поширеним, проте воно досить ненадійне, не підлягають масштабуванню та супроводжуються високим ризиком плутанини. Це рішення не відповідає сучасним очікуванням клієнтів і є лише тимчасовим.

Google Calendar не передбачає вбудованої логіки запису клієнтів, тому доводиться вручну вносити записи, що потребує більшої витрати часу та підвищує ймовірність помилок. Крім того, календар не підтримує автоматичний вибір послуги, майстра чи тривалості запису, що потребує від підприємців додатково налаштовувати інші сервіси чи самостійно вести облік цих даних.

З іншого боку, платформи, як Alteg.io, пропонують великий функціонал, але водночас характеризуються високою вартістю підписки, складністю первинного налаштування та наявністю великої кількості модулів, які не є критично важливими для малих компаній чи незалежних майстрів. Тобто, ці системи переважно орієнтовані на середній та великий бізнес, що призводить до непридатності для підприємців-початківців

Кастомні рішення забезпечують максимальну гнучкість та повну адаптацію під потреби конкретного бізнесу, однак потребують значних фінансових і часових ресурсів, а також подальшої технічної підтримки. Це створює додаткові труднощі для підприємців, що зацікавлені у швидкому старті та в простому рішенню без зайвих витрат.

Таким чином, жодне з розглянутих рішень не є оптимальним для малого бізнесу, якому потрібен простий, зручного, доступний і зрозумілий вебзастосунок для онлайн-запису без додаткових складних налаштувань, авторизацій чи високих витрат. Саме це й визначає актуальність створення спеціалізованого рішення в межах цієї кваліфікаційної роботи..

Результати порівняльного аналізу наведені у таблиці 1.1. Знак «+» означає наявність функціоналу на дослідженому сервісі, знак «-» – відсутність.

Таблиця 1.1 – Порівняльна характеристика існуючих рішень

Рішення	Ручний запис	Google Calendar	Alteg.io	Кастомні рішення
Простота впровадження	+	+	-	-
Функціональність	-	-	+	+
Гнучкість налаштувань	-	-	+	+
Онлайн-запис без адміністратора	-	-	+	+
Підходить для малого бізнесу	+	+	-	-
Потребує технічних навичок	-	-	+	+
Функціональність	-	-	+	+
Велика вартість	-	-	+	+

1.3 Постановка задачі

При реалізації веб-застосунку першочерговим завданням є визначення ролей користувачів. У системі передбачено чотири основні ролі: супер-адміністратор, адміністратор, майстер та звичайний користувач (клієнт). Кожна з цих ролей має власний набір дозволів та функціональних можливостей, що визначають її взаємодію із застосунком.

Супер-адміністратор виконуватиме ключову роль в адмініструванні системи. Він здійснює модерацію цих заявок, може схвалювати або відхиляти їх, а також блокувати раніше підтверджені компанії в разі виявлення порушень. Також супер-адміністратор матиме доступ до перегляду всіх даних у системі, незалежно від того, до якої компанії вони належать.

Адміністратор відповідатиме за управління контентом у межах компаній, які створені ним. До його функціоналу входитиме можливість створення, редагування та видалення компаній, повне управління інформацією про майстрів і послуги (створення, читання, оновлення та видалення), перегляд списку клієнтів, а також керування графіком записів, зокрема переглядом та створенням робочих графіків. Крім того, адміністратор матиме доступ до перегляду аналітики, що надає статистичні дані щодо записів і завантаженості майстрів.

Майстер буде зареєстрованим користувачем, прикріпленим до певної компанії, який надає конкретні послуги клієнтам. У кабінеті майстра буде доступна функціональність для перегляду та управління особистим графіком, встановлення робочих годин, переглядом майбутніх записів та зміни їх статусу. Майстер також буде мати доступ до певної інформації про клієнтів, які зробили запис, та буде отримувати сповіщення про нові бронювання в реальному часі. Після того як адміністратор додає нового майстра до системи, на вказану під час створення облікового запису електронну пошту буде автоматично надіслано лист із логіном, паролем та посиланням для

входу до особистого кабінету.

Звичайний користувач (клієнт) є ключовим елементом у функціональній моделі вебзастосунка, оскільки саме для його потреб реалізовано основні можливості системи. Користувач може вільно переглядати сторінку з деталями компанії де може ознайомитися з переліком послуг, списком майстрів та додатковою інформацією. Для зручності передбачено декілька способів запису: за майстром, за послугою або на певний час. У першому випадку клієнт обирає конкретного спеціаліста та переглядає його доступний розклад; у другому – починає з вибору послуги, після чого система пропонує відповідних майстрів; у третьому – спочатку вказує бажаний час, після чого вже обирає усі інше.

Процес запису не вимагає створення облікового запису, достатньо ввести контактні дані, що значно спрощує взаємодію та знижує поріг входу для нових користувачів. Після завершення візиту клієнт має можливість залишити відгук про отриману послугу, що сприяє підвищенню довіри до системи та покращенню якості обслуговування.

Після успішного створення запису користувач отримає на вказану електронну пошту лист із підтвердженням та деталями бронювання. Для запобігання зловживанню системою, у разі якщо користувач здійснює більше трьох записів протягом однієї години, буде активовано захист у вигляді перевірки CAPTCHA.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Огляд засобів розробки

Під час розробки вебзастосунку було використано сучасний стек технологій, спрямований швидку розробку, масштабованість та зручність користувацького інтерфейсу.

Основу проєкту складає фреймворк Next.js, призначений для створення React-застосунків із підтримкою серверного рендерингу, маршрутизації, API-обробки та Server Actions. Завдяки використанню Next.js вдалося досягти високої продуктивності завдяки можливості комбінованого рендерингу (на стороні сервера і клієнта) та спростити створення динамічного контенту.

Для реалізації адаптивного та сучасного дизайну використовується утилітарна CSS-бібліотека Tailwind CSS. Її набір готових класів дозволяє швидко створювати стилізовані компоненти і використовувати для формування зовнішнього вигляду інтерфейсу.

Доповненням до Tailwind CSS застосовується компонентна бібліотека shadcn/ui, яка надає стандартизовані інтерфейсні елементи, включаючи форми, таблиці та діалоги. Це забезпечує уніфікований вигляд та спрощує створення складних UI-компонентів. Для анімацій використовується бібліотека Framer Motion, що дозволяє додавати динамічність та візуальні ефекти до компонентів без суттєвого навантаження на систему [6].

Обробка форм реалізована за допомогою бібліотеки React Hook Form, яка забезпечує контроль введених даних та валідацію на стороні клієнта [7]. Для сповіщень та повідомлень використовується бібліотека React Hot Toast, що дозволяє у реальному часі інформувати користувача про успішні дії або помилки.

Для зберігання даних та автентифікації використовується серверне рішення PocketBase, що функціонує як сховище даних з підтримкою REST

API. PocketBase надає можливість створювати колекції, керувати користувачами та працювати з файлами, що робить його ефективним інструментом для зберігання структурованих даних.

Для надсилання електронних листів інтегровано сервіс Resend, що дозволяє автоматизувати розсилку повідомлень клієнтам та майстрам.

Задля зручної обробки дат і подій використовується бібліотеки date-fns, Moment.js та React Big Calendar. Це забезпечує створення інтерактивних календарів з підтримкою різних форматів дат, часового поясу та локалізації.

Для інтеграції мап використовується бібліотека @react-google-maps/api, яка надає інструменти для роботи з картографічними сервісами Google Maps. Це забезпечує відображення локацій та зручний візуальний інтерфейс для навігації.

2.2 Середовище для розробки Cursor

Cursor є сучасним інтегрованим середовищем розробки (IDE), створеним на основі Visual Studio Code, яке спеціалізується на підвищенні продуктивності за рахунок використання технологій штучного інтелекту. Середовище підтримує всі ключові функціональні можливості VS Code і також доповнене вбудованим AI-асистентом, що сприяє прискоренню процесу написання, рефакторингу та аналізу програмного коду.

Cursor забезпечує всі основні функції сучасного редактора коду, зокрема підсвічування синтаксису, автодоповнення, пошук і заміну, зручну навігацію між файлами, інтеграцію з Git та терміналом. Крім того, середовище сумісне із широким спектром плагінів і розширень з екосистеми VS Code, що дає змогу адаптувати його під специфічні потреби проєкту. Особливістю Cursor є контекстно-обізнаний AI, який здатен аналізувати всю структуру проєкту, відповідати на запити щодо фрагментів коду та пропонувати нові компоненти або виправляти помилки [8].

До основних можливостей середовища Cursor належать:

- підтримка мов TypeScript, JavaScript, HTML, CSS, а також багатьох інших завдяки сумісності з розширеннями VS Code;
- автоматичне завершення коду з урахуванням контексту проєкту;
- підсвічування помилок у режимі реального часу;
- генерація, пояснення та рефакторинг коду за допомогою AI;
- можливість встановлення та налаштування тем, лінтерів, форматерів та інших інструментів розробника;
- підтримка Snippets, Debugging та інтеграція з терміналом;
- доступ до документації та прикладів безпосередньо з редактора;
- інтеграція з системами керування версіями, зокрема Git.

Cursor значно скорочує час реалізації завдань завдяки автоматизації рутинних операцій, швидкому перемиканню між модулями та оптимізованій роботі з проєктною структурою. Підтримка TypeScript, Tailwind CSS та JSX/TSX робить це середовище особливо ефективним для створення сучасних вебзастосунків на базі React та Next.js.

2.3 TypeScript

TypeScript – це мова програмування, яка виступає надбудовою над JavaScript та розширює його можливості за рахунок впровадження статичної типізації. Представлена компанією Microsoft у 2012 році, вона знайшла широке застосування у розробці масштабованих вебзастосунків. Однією з ключових особливостей TypeScript є компіляція у звичайний JavaScript, що забезпечує виконання програми у будь-якому браузері чи середовищі, які підтримують JavaScript [9].

Головною перевагою TypeScript є статична типізація, яка дозволяє виявляти помилки ще на етапі написання коду, до його виконання [9]. Це значно знижує кількість типових та логічних помилок, особливо у великих проєктах, де важливо дотримуватись структурованості та зрозумілості коду. Окрім того, TypeScript підтримує сучасний синтаксис ECMAScript,

включаючи класи, модулі, async/await, деструктуризацію, а також розширює можливості за допомогою інтерфейсів, дженериків і аліасів типів [10].

В межах даного проєкту TypeScript використовується як основна мова програмування. Його застосування дозволяє чітко структурувати логіку на рівні типів, що полегшує перевірку відповідності вхідних даних та полегшує реалізацію складних елементів, зокрема форм, валідації, авторизації та обробку введених користувачем даних. Це робить проєкт надійнішим і структурованішим, що дуже важливо для сучасних вебзастосунків.

TypeScript поєднується з іншими інструментами, які використовуються у проєкті. У зв'язці з ESLint мова дозволяє здійснювати автоматичний аналіз коду для виявлення помилок, а з React Hook Form забезпечує безпечну та контрольовану роботу з формами [11, 12]. Крім того, мова ефективно інтегрується з Tailwind CSS, сприяючи зручному використанню класів стилізації. У компонентній бібліотеці shadcn/ui типізація дозволяє перевіряти правильність пропсів при створенні та використанні інтерфейсних елементів, що знижує ймовірність помилок у шаблонах та підвищує якість взаємодії з користувачем.

Серед переваг використання TypeScript можна виділити:

- автодоповнення змінних, функцій і пропсів у редакторі;
- раннє виявлення помилок завдяки статичній типізації;
- спрощення командної роботи та масштабування застосунку;
- зручна документація, забезпечена наявністю типів;
- покращена інтеграція з API сторонніх сервісів завдяки типізації відповідей.

TypeScript тісно інтегрований з фреймворком Next.js і фактично є стандартом де-факто для проєктів на його основі. Починаючи з останніх версій, підтримка TypeScript у Next.js є вбудованою та не потребує додаткового встановлення або конфігурації, а лише достатньо додати файли з розширеннями .ts або .tsx, і фреймворк автоматично активує типізацію. Такий підхід сприяє швидкому старту, зручному масштабуванню та

полегшує підтримку коду, що є критично важливим для сучасних складних вебзастосунків.

2.4 Next.js

Next.js – це сучасний фреймворк на базі бібліотеки React, розроблений компанією Vercel. Його головна мета – забезпечити повноцінне середовище для створення продуктивних вебзастосунків, які будуть підтримувати серверний рендеринг (SSR), статичну генерацію (SSG), інкрементальну статичну генерацію (ISR) та клієнтське рендеринг (CSR) [13, 14]. Завдяки такій гнучкості Next.js діє змогу створювати як традиційні SPA, так і багатосторінкові вебсайти з високою продуктивністю й оптимізацією для пошукових систем [14].

Фреймворк надає можливість вибору найкращої стратегії рендерингу залежно від характеру контенту та вимог до проєкту [14]. Так, SSR забезпечує генерацію сторінок на сервері, що корисно для динамічного контенту, тоді як SSG використовується для статичних сторінок з рідкими оновленнями.

Опція ISR дозволяє оновлювати контент без перегенерації всього сайту, що значно підвищує продуктивність. CSR застосовується для інтерфейсів з високою інтерактивністю, які потребують швидкої клієнтської взаємодії. Починаючи з версії 13, у Next.js запроваджено нову архітектуру App Router, що базується на серверних компонентах React [14, 15, 16]. Це забезпечує підвищену продуктивність, покращену структурованість проєкту та більш гнучке управління станом застосунку.

Однією з важливих особливостей Next.js є вбудована підтримка API-маршрутів, що дозволяє реалізовувати серверну логіку без окремого бекенд-середовища [17]. Такий підхід значно полегшує розгортання fullstack-застосунків та мінімізує інфраструктурні залежності. У межах даного проєкту саме ця функціональність використовується для взаємодії з базою

даних PocketBase, реалізації авторизації користувачів та забезпечення внутрішньої логіки.

Next.js автоматично здійснює поділ коду, що зменшує час завантаження сторінок. Використання lazy loading компонентів та попереднє завантаження забезпечує швидкий рендеринг навіть при великому обсязі даних. Додатково підтримка кешування дозволяє зменшити навантаження на сервер та прискорити завантаження контенту для користувачів.

Фреймворк легко інтегрується з сучасними інструментами, як TypeScript, ESLint та Tailwind CSS. Крім того, він оптимізований для роботи з платформою Vercel, що дозволяє автоматизувати процеси деплою та підтримувати високу продуктивність застосунку. Завдяки регулярним оновленням і активній спільноті Next.js залишається актуальним інструментом для створення сучасних вебзастосунків.

До переваг використання Next.js у цьому проєкті можна віднести:

- гнучкий підхід до рендерингу сторінок залежно від бізнес-логіки;
- можливість обробки серверних запитів без окремого серверу;
- хорошу масштабованість застосунку;
- високу швидкодію завдяки оптимізаціям Vercel;
- активну спільноту та регулярні оновлення;
- просту інтеграцію з сучасними інструментами розробки та хостингу.

Next.js дозволяє швидко створювати вебзастосунки завдяки вбудованій маршрутизації, підтримці TypeScript, API-роутам, автоматичній оптимізації продуктивності та інтеграції з актуальними інструментами. Такий підхід дозволяє досягати високої якості вебзастосунків при мінімальних витратах на налаштування інфраструктури та технічну підтримку в майбутньому.

2.5 PocketBase

PocketBase – це компактне та функціональне серверне рішення класу BaaS, яке об'єднує можливості бази даних, автентифікації, файлового

сховища та REST API в одному інструменті [18]. Завдяки простоті у розгортанні та мінімальним інфраструктурним вимогам він дозволяє швидко створювати повноцінні вебзастосунки без потреби в окремому серверному середовищі. Розробка ведеться як open source і його кодова база написана мовою Go.

Збереження даних реалізовано за допомогою вбудованої бази SQLite, яка забезпечує швидкодію та стабільність для невеликих і середніх за масштабом проектів. Обмін даними здійснюється через REST API або WebSocket, який дозволяє реалізувати функціонал у режимі реального часу [18, 19]. Для зручного адміністрування передбачено вбудовану панель керування, яка дає змогу створювати колекції, налаштовувати структуру даних і визначати типи полів, чи встановлювати правила доступу до них.

У межах даного проєкту PocketBase використовується як основна система зберігання даних та інструмент автентифікації. Усі сутності, включно з компаніями, послугами, майстрами та записами, організовані у вигляді колекцій, для кожної з яких задано окремі правила доступу (listRule, createRule, updateRule), що дозволяє реалізувати багаторівневу рольову модель доступу для адміністратора, майстра чи клієнта.

Основні переваги PocketBase:

- простота розгортання у локальному середовищі або на сервері без потреби в складній інфраструктурі;
- підтримка повноцінного REST API для інтеграції з клієнтською частиною;
- реалізація автентифікації через JWT-токени, OAuth2 або підтвердження електронною поштою;
- можливість експорту й імпорту колекцій;
- зручна панель адміністратора без необхідності прямої роботи з SQL.

У поєднанні з Next.js, PocketBase дозволяє швидко створити повноцінний backend-рішення з готовою структурою бази даних, API та автентифікацією, що суттєво скорочує час розробки MVP. Завдяки

поєднанню простоти використання, зручної структури даних і підтримки ключових функцій, PocketBase виявився оптимальним вибором для реалізації даного вебзастосунку з можливістю онлайн-запису.

2.6 Tailwind CSS та shadcn/ui

У процесі створення вебзастосунку особлива увага була приділена реалізації інтерфейсу, що поєднує адаптивність, візуальну узгодженість та зручність користування. Для цього було обрано утилітарний CSS-фреймворк Tailwind CSS та компонентну бібліотеку shadcn/ui, яка базується на рішеннях Radix UI і забезпечує створення доступних і багаторазово використовуваних інтерфейсних елементів [20, 21].

Tailwind CSS є утилітарною бібліотека, яка дозволяє застосовувати стилі безпосередньо в розмітці JSX за допомогою визначених класів [20]. Такий підхід сприяє пришвидшенню верстки і забезпечує стилістичну єдність у межах усього застосунку та мінімізує потребу у написанні окремих CSS-файлів.

До основних переваг Tailwind CSS належать:

- кастомізація дизайну через конфігураційний файл tailwind.config.ts;
- підтримка темної теми, брейкпоінтів, анімацій та псевдокласів;
- зручна інтеграція з фреймворками React та Next.js;
- забезпечення принципу DRY у процесі розробки адаптивного інтерфейсу.

shadcn/ui – це набір готових компонентів, побудованих із використанням Tailwind CSS, TypeScript та Radix UI. Компоненти є повністю типізованими, легко налаштовуються, мають підтримку темної теми, забезпечують доступність згідно з WCAG-стандартами та відповідають сучасним вимогам дизайну. До складу бібліотеки входять кнопки, поля вводу, таблиці, модальні вікна, селектори, календарі, перемикачі та інші UI-елементи [22].

У межах даного проєкту бібліотека shadcn/ui використовується для побудови ключових інтерфейсних блоків: форм бронювання, таблиць записів, системи фільтрації, інтерфейсу перегляду записів. Tailwind CSS, своєю чергою, забезпечує точне стилістичне оформлення компонентів та їх адаптацію під індивідуальні потреби бізнес-логіки.

Комбіноване використання Tailwind CSS і shadcn/ui дозволило:

- підвищити швидкість розробки фронтенду;
- забезпечити уніфікований візуальний стиль усього застосунку;
- створити адаптивний інтерфейс без надмірного дублювання CSS-коду;
- покращити зручність і доступність взаємодії користувача із системою.

Окремою перевагою бібліотеки shadcn/ui є спосіб встановлення компонентів: замість підключення ззовні вони додаються у вигляді окремих локальних файлів. Це дозволяє розробнику вільно змінювати логіку, зовнішній вигляд або поведінку компонентів відповідно до конкретних потреб. Такий підхід забезпечує гнучкість розробки, полегшує кастомізацію та дає змогу уникати обмежень типових бібліотек, у яких компоненти не можна змінювати локально.

2.7 Resend & Google Maps API

У процесі розробки вебзастосунку було реалізовано інтеграцію з зовнішніми сервісами, які значно розширили функціональність системи та підвищили зручність її використання для кінцевих користувачів. Серед таких рішень – хмарний поштовий сервіс Resend та Google Maps API, які забезпечують відповідно автоматизоване надсилання електронних повідомлень та інтерактивну роботу з картами та геоданими.

Resend є сучасним сервісом для надсилання транзакційних електронних листів, що підтримує інтеграцію через REST API та офіційний SDK [23]. До

функціоналу платформи входять шаблонізація листів, логування надсилань, моніторинг за доменами та висока швидкість доставлення [24]. У межах даного проєкту Resend використовується для підтвердження записів клієнтів та запрошенням зареєстрованих майстрів шляхом надсилання email-повідомлень.

Google Maps API – це набір інструментів, розроблених компанією Google, що дозволяє вбудовувати карти, використовувати функції геолокації та реалізовувати пошук місць у вебзастосунках. У межах цього проєкту було використано бібліотеку @react-google-maps/api, яка надає компоненти для інтеграції картографічного функціоналу в архітектуру React-застосунку [25]. Завдяки цьому можливо відображати локації компаній і послуги та розміщувати маркери чи визначення координат за вказаною адресою.

Таким чином, застосування зовнішніх сервісів Resend та Google Maps API суттєво підвищило функціональність вебзастосунку, забезпечивши автоматизацію частини логіки і зручність у роботі з географічною інформацією. Це створює повноцінне середовище, яке є водночас ефективним для адміністраторів та інтуїтивно зрозумілим для клієнтів.

2.8 Vercel

Vercel є сучасною хмарною платформою для хостингу та автоматизованого розгортання вебзастосунків, спеціально оптимізована для фреймворків на основі React, зокрема Next.js [26]. Платформа розроблена однойменною компанією і є офіційно рекомендованим рішенням для розгортання застосунків, створених з використанням Next.js. Vercel забезпечує високу продуктивність, автоматичне масштабування, глобальна мережа доставки контенту, а також зручну інтеграцію з системами керування версіями, такими як GitHub, GitLab або Bitbucket [27, 28, 29].

Однією з ключових переваг платформи є підтримка процесу CI/CD що дозволяє автоматично створювати прев’ю-версії застосунку або оновлювати

продакшн середовище без потреби у ручному втручанні [30]. Такий підхід значно підвищує ефективність розробки, покращує командну взаємодію та зменшує ризики при розгортанні нових версій.

Серед інших можливостей Vercel:

- повна підтримка SSR, SSG та ISR для Next.js;
- швидка доставка контенту через глобальну мережу серверів;
- підтримка HTTPS і підключення кастомних доменів;
- логування та аналітика запитів у реальному часі;
- вбудована система керування середовищами (.env змінні) для staging і production.
- автоматичне відстеження помилок під час збірки та деплою з повідомленнями на пошту або до Git-платформи;
- інтеграція з edge-функціями для обробки запитів на рівні CDN, що покращує швидкість відповіді.

У межах проєкту Vercel використовується як основна платформа для розгортання та хостингу вебзастосунку. Завдяки глибокій інтеграції з Next.js, розгортання відбувається без необхідності додаткового налаштування серверної інфраструктури, що дає змогу зосередитися безпосередньо на реалізації бізнес-логіки. Таким чином, Vercel не лише виконує функції хостингу, а ще й виступає як повноцінне середовище для підтримки та автоматизованого керування життєвим циклом вебзастосунку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура даних у PocketBase

Для реалізації серверної частини проєкту використано PocketBase, який забезпечує зручний доступ, зберігання та обробку даних. Усі сутності застосунку представлені у вигляді колекцій, структура яких ретельно продумана для підтримки ролей, зв’язків між даними, фільтрації та можливості розширення. Застосунок взаємодіє з PocketBase через офіційний JavaScript SDK, а не безпосередньо через REST API. Це дозволяє використовувати зручний об'єктно-орієнтований синтаксис та скоротити обсяг шаблонного коду. SDK забезпечує зручний та зрозумілий спосіб взаємодії з базою даних, дозволяючи легко реалізовувати CRUD-операції, фільтрацію даних та обробку зв’язаних сутностей.

3.1.1 Структура колекцій PocketBase

У PocketBase колекції можуть бути як звичайні, так і системні, наприклад, як users. Кожна колекція має власний набір полів, типів даних, правил доступу (ACL) та зв’язків, що дозволяє точно моделювати структуру застосунку та забезпечувати розмежування прав для різних ролей.

Основні поля колекції «users»:

- id – унікальний ідентифікатор користувача;
- password – пароль користувача, збережений у вигляді хешу;
- tokenKey – службове поле для керування сесіями та токенами доступу;
- email – електронна пошта користувача;
- emailVisibility – булеве поле, що визначає видимість електронної пошти для інших користувачів;

- verified – стан підтвердження електронної адреси;
- username – унікальне ім’я користувача, що може використовуватись як логін;
- name – повне ім’я користувача, яке відображається в інтерфейсі;
- avatar – аватар користувача, що додається у вигляді одного зображення;
- role – роль користувача в системі: Admin або Master;
- super_user – булеве поле, що вказує на наявність прав супер-фдміністратора;
- is_deleted – булеве поле, яке вказує чи був користувач прихований без фізичного видалення;
- created – дата створення користувача;
- updated – дата останнього оновлення користувача.

Колекція «company» містить інформацію про компанії або організації, що надають послуги. Вона використовується для структурування даних, пов’язаних із власниками, майстрами, записами тощо.

Основні поля колекції:

- id – унікальний ідентифікатор компанії;
- logo – логотип компанії у форматі одного зображення;
- name – назва компанії;
- website_link – посилання на зовнішній сайт компанії (необов’язкове поле);
- address – адреса компанії у форматі рядка з координатами;
- masters – посилання на користувачів-майстрів, що працюють у компанії (зв’язок типу many-to-many з колекцією users);
- description – опис компанії;
- owner – зв’язок із користувачем, що володіє компанією (обов’язкове поле);
- phone – номер телефону компанії;
- admin_review – статус модерації, який встановлює супер-

адміністратор: Approved, Rejected, Banned;

- is_deleted – булеве поле, яке вказує чи було компанію деактивовано або приховано без фізичного видалення з бази;
- created – дата створення запису про компанію;
- updated – дата останнього оновлення компанії.

Колекція «master» зберігає інформацію про майстрів, які надають послуги в межах певної компанії. Вона використовується для формування сторінок майстрів, відображення їх графіків, послуг чи рейтингу.

До складу колекції входять такі поля:

- id – унікальний ідентифікатор майстра;
- email – електронна адреса майстра;
- first_name – ім’я майстра;
- last_name – прізвище майстра;
- position – посада або спеціалізація;
- photo – фотографія майстра;
- company – зв’язок із колекцією company, що вказує на компанію, в якій працює майстер (обов’язкове поле);
- user – зв’язок з обліковим записом у колекції users, що дозволяє авторизованому користувачу взаємодіяти з системою (обов’язкове поле);
- services – зв’язок з колекцією service, перелік послуг, які виконує майстер (many-to-many);
- rating_value – числове значення середнього рейтингу, отриманого через відгуки клієнтів;
- is_visible – булеве поле, яке визначає відображення майстра для користувачів;
- is_deleted – булеве поле, яке вказує чи було клієнта деактивовано або приховано без фізичного видалення з бази;
- created – дата створення запису про майстра;
- updated – дата останнього оновлення запису про майстра.

Колекція «service» містить інформацію про послуги, що надаються у

межах певної компанії. Вона використовується для формування списку доступних послуг, які може вибрати клієнт під час онлайн запису.

До складу колекції входять такі поля:

- id – унікальний ідентифікатор послуги;
- price – вартість послуги у числовому форматі;
- company – зв’язок з колекцією company, що вказує, якій компанії належить послуга (обов’язкове поле);
- name – назва послуги (наприклад, стрижка, масаж, манікюр);
- is_visible – булеве поле, що визначає, чи відображається послуга для користувача;
- is_deleted – булеве поле, яке вказує чи було клієнта деактивовано або приховано без фізичного видалення з бази;
- created – дата створення запису про послугу;
- updated – дата останнього оновлення запису про послугу.

Колекція «client» містить деякі дані клієнтів, які здійснюють онлайн-запис на послуги в межах конкретної компанії. Вона використовується для збереження контактної інформації клієнтів та зв’язку з компаніями.

До складу колекції входять такі поля:

- id – унікальний ідентифікатор послуги;
- price – вартість послуги у числовому форматі;
- name – ім’я клієнта;
- phone – номер телефону для зв’язку;
- email – електронна адреса (може використовуватись для сповіщень або ідентифікації);
- company – зв’язок з колекцією company, що визначає, до якої компанії належить клієнт;
- is_deleted – логічне поле, яке вказує на те, чи було клієнта деактивовано або приховано без фізичного видалення з бази;
- created – дата створення запису клієнта;
- updated – дата останнього оновлення запису про клієнта.

Колекція «master_working_hours» використовується для зберігання графіка роботи майстрів за днями тижня. Дозволяє визначити час доступності кожного майстра в певний день, що є основою для побудови системи онлайн-запису.

До складу колекції входять такі поля:

- id – унікальний ідентифікатор запису;
- master – зв’язок з колекцією master, що визначає, якому майстру належать ці години;
- enabled – логічне поле, яке вказує, чи активний запис (наприклад, чи працює майстер у цей день);
- start_time – час початку роботи майстра;
- end_time – час завершення роботи майстра;
- day_of_week_number – номер дня тижня (0 – неділя, 1 – понеділок і т.д.);
- is_deleted – булеве поле, яке вказує, чи було запис деактивовано або приховано без фізичного видалення з бази;
- created – дата створення запису;
- updated – дата останнього оновлення запису.

Колекція «review» використовується для зберігання відгуків клієнтів після візиту. Вона дозволяє оцінити якість наданої послуги та залишити коментар відносно наданої послуги, що сприяє підвищенню прозорості та довіри до сервісу.

До складу колекції входять такі поля:

- id – унікальний ідентифікатор запису;
- booking – зв’язок із записом на послугу (колекція booking), який є джерелом відгуку;
- master – зв’язок з майстром (колекція master), якому залишено відгук;
- rating – числова оцінка (рейтинг), яку залишив клієнт;
- comment – текстовий коментар користувача про послугу;

- is_deleted – булеве поле, яке вказує на те, чи було відгук деактивовано або приховано без фізичного видалення з бази;
- created – дата створення відгуку;
- updated – дата останнього оновлення запису.

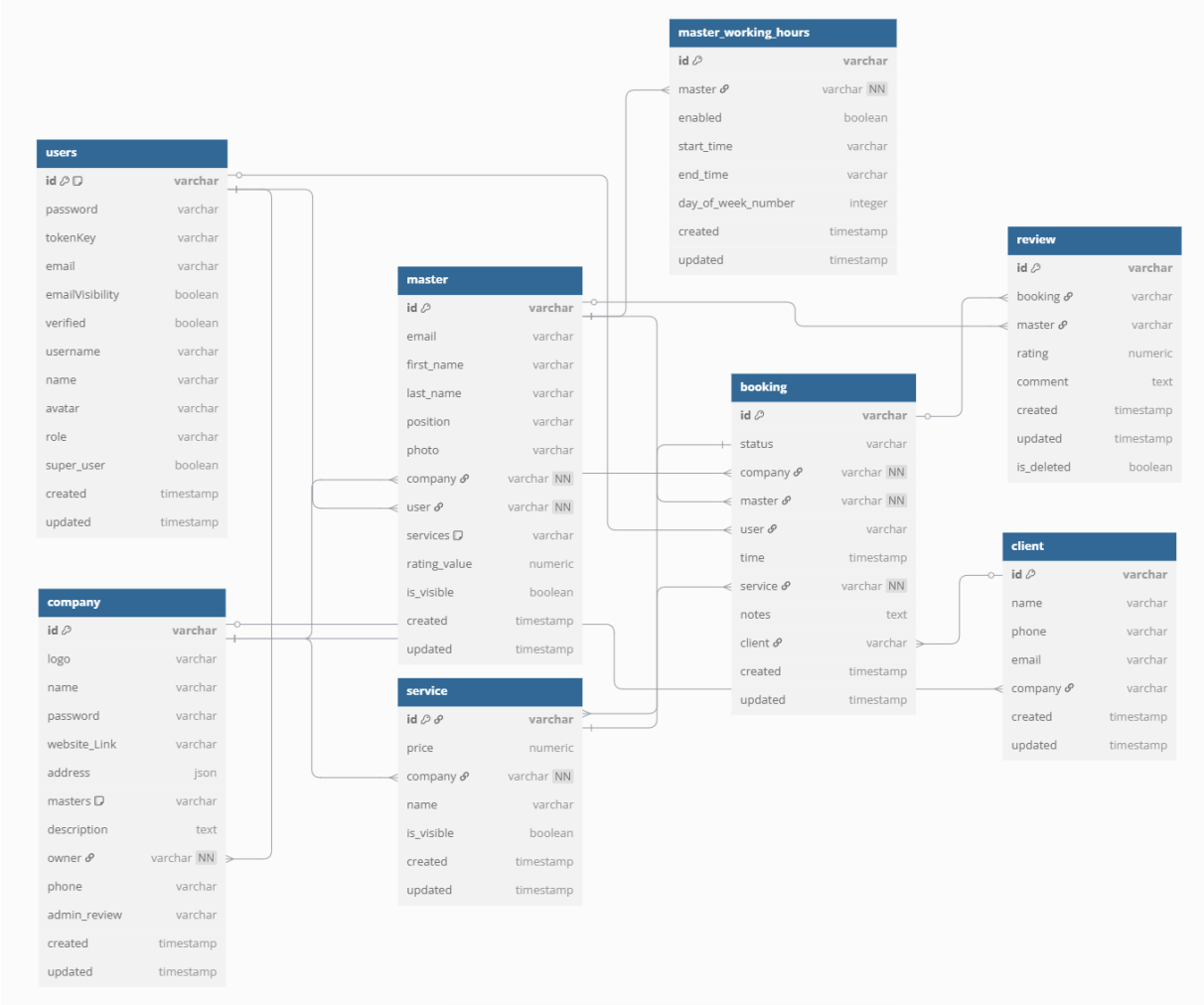

Рисунок 3.1 – Схема бази даних вебзастосунку для онлайн-запису

Завдяки чітко структурованим колекціям, застосунок підтримує ключові функції: автентифікацію користувачів, управління компаніями та майстрами, онлайн-запис, формування розкладу та збирання відгуків. Така модель дозволяє ефективно масштабувати систему, розширювати функціональність та забезпечувати контроль доступу на основі ролей. Кожна колекція логічно пов’язана з іншими через зовнішні ключі, що полегшує агрегацію та фільтрацію даних при реалізації API-логіки.

3.2 API та взаємодія з даними

У межах реалізації вебзастосунку використано сучасну архітектуру маршрутизації – App Router у Next.js, яка дозволяє створювати повноцінні API-ендпоінти безпосередньо всередині проєкту. Всі маршрути реалізовано у директорії app/api, де кожна папка відповідає за певний шлях, а файл route.ts містить обробники HTTP-запитів. У більшості випадків один файл route.ts об’єднує кілька методів (наприклад, GET, POST, PATCH, DELETE) для роботи з відповідною колекцією або ресурсом. Таке групування спрощує підтримку коду та забезпечує логічну організацію функцій, пов’язаних із певною сутністю.

У проєкті API виконує роль посередника між клієнтською частиною та серверною логікою, реалізованою за допомогою PocketBase. Кожен ендпоінт відповідає за виконання конкретної дії з даними – створення, отримання, оновлення або видалення. Для цього використовується офіційний JavaScript SDK PocketBase, який забезпечує зручний доступ до колекцій, підтримку автентифікації, фільтрацію, пагінацію та роботу з пов’язаними записами.

app/api/service/route.ts – обробляє запити для багатьох записів;

app/api/service/[id]/route.ts – обробляє запити однієї послуги за ID.

Кожен route.ts слугує точкою входу для запитів до певної сутності. Замість реалізації всієї логіки безпосередньо в обробнику, запити делегуються до окремих сервісних функцій, що дозволяє зберігати чисту архітектуру та полегшує повторне використання коду. Такий підхід відокремлює рівень контролерів (обробка HTTP) від бізнес-логіки (взаємодія з PocketBase, перевірка ролей, обробка помилок тощо).

App Router автоматично розпізнає серверні маршрути на основі структури файлової системи. Динамічні маршрути (наприклад, [id]) дозволяють створювати гнучкі API-ендпоінти без потреби у конфігурації.

Загалом така структура дозволяє:

- ефективно організувати серверну логіку в межах Next.js-застосунку;

- розділити відповідальність між API та сервісними функціями;
- забезпечити гнучке масштабування застосунку без потреби у зовнішньому бекенді;
- зменшити кількість запитів до PocketBase, за рахунок попередньої обробки та фільтрації на рівні API.

Такий підхід підвищує зручність супроводу проєкту та спрощує додавання нових функцій. Крім того, він дає змогу централізовано керувати перевірками доступу та узгодженістю даних.

3.2.1 Адміністративна логіка (Admin API)

Адміністратор має розширені можливості управління через API, зокрема створення записів, редагування існуючих, модерацію контенту та фільтрацію даних за визначеними критеріями.

Управління компаніями здійснюється через маршрути /api/company та /api/company/[id]. Запит типу POST дозволяє створити нову компанію з передачею даних через об'єкт FormData, що включає такі поля, як назва, адреса, логотип, опис тощо. При оновленні компанії можна змінювати загальні поля, зокрема, поле admin_review, яке доступне для редагування лише супер-адміністратору і використовується для позначення статусу модерації компанії та визначає, чи буде компанія доступна для загального перегляду у системі.

Відповідно до логіки, реалізованої в ендпоінті GET /api/company, лише компанії зі статусом admin_review = "Approved" доступні для звичайних користувачів (лістинг 3.1). Для супер-адміністратора реалізовано спеціальні умови, якщо до запиту додається заголовок x-admin-request: true то система повертає всі записи, включаючи ті, що ще мають інші статуси. Додатково перед виконанням запиту виконується перевірка ролі користувача. Спочатку система зчитує авторизаційний токен з об’єкта pb.authStore, або (якщо токен передано вручну через заголовок Authorization) виконує його розпізнавання.

Далі визначається роль користувача (role) та значення поля super_user. Якщо користувач має роль Admin та є суперкористувачем (super_user = true), йому надається доступ до повного списку компаній, включно з тими, що перебувають у статусі Pending, Rejected або Banned. Звичайні адміністратори (без статусу суперкористувача) можуть переглядати лише компанії, створені ними особисто, а інші користувачі бачать лише затверджені компанії. Таким чином, доступ до даних суворо контролюється на основі ролей, що зберігаються у PocketBase.

Лістинг 3.1 – Метод GET() для отримання компанії

```
export async function GET(request: Request): Promise<Response> {
  const param = new URL(request.url).searchParams;
  const user = await getUserInfo(request, pb);
  const filter = buildCompanyFilter({
    ...user,
    owner: param.get('owner'),
    status: param.get('status'),
    showAllParam: param.get('showAll') === 'true'
  });
  const res = await pb.collection('company')
    .getList(1, parseInt(param.get('limit') || '50'), {
      sort: param.get('sort') || '-created',
      filter,
    });
  return NextResponse.json(res.items.map(mapToCompany));
};
```

Модерація послуг і майстрів також доступна через відповідні маршрути. Послуги створюються через POST /api/service і оновлюються за допомогою PATCH /api/service/[id]. Важливою складовою є поле is_visible, яке адміністратор може змінювати для приховування послуги з публічного показу для клієнта в системі. Це забезпечує контроль над наповненням платформи актуальними і перевіреними послугами.

Аналогічно додаються майстри, через маршрут POST /api/master, де передаються такі параметри, як ім’я, фото, компанія, перелік послуг. Прив’язка послуг до майстра реалізована як масив і передається у полі

services. Оновлювати дані майстра можна через PATCH /api/master/[id], включно з редагуванням видимості is_visible та оновленням переліку послуг. Такий підхід дозволяє адміністрації гнучко керувати відображенням майстрів для клієнтів.

Особливості доступу до адміністративних функцій визначаються на основі ролі користувача, яка зберігається у полі role облікового запису. Для отримання актуальної ролі користувача використовується метод getCurrentUser() (лістинг 3.2). Роль адміністратора має значення "Admin". Логіка перевірки ролі реалізована в окремому сервісному файлі `src/services/auth.ts`, де визначаються поточний користувач та його повноваження.

Лістинг 3.2 – Метод getCurrentUser() для отримання даних про авторизованого користувача

```
getCurrentUser: async () => {
  try {
    const token = auth.getToken();
    if (!token) return null;
    const res = await fetch('/api/auth/user', {
      headers: { Authorization: `Bearer ${token}` },});
    if (res.status === 401) {
      localStorage.removeItem(TOKEN_KEY);
      return null;}
    const data = await res.json();
    return data.user || null;
  } catch {
    return null;
  }
};
```

Таким чином, адміністративна логіка охоплює повний цикл управління основними сутностями системи: компаніями, майстрами, послугами та записами. Вона реалізована через окремі маршрути з підтримкою фільтрації та сортування, що забезпечує ефективне та безпечне управління контентом на платформі.

3.2.2 Логіка майстрів (Master API)

Функціональність, пов’язана з обробкою даних про майстрів, реалізована у вигляді окремого набору API-ендпоінтів, розміщених у директорії app/api/master. Вони дозволяють створювати, оновлювати та отримувати інформацію про майстрів, а також керувати їхнім графіком роботи. Для забезпечення високої якості розробки застосовано модульну архітектуру – вся логіка взаємодії з базою даних винесена до окремого сервісного шару src/services/master.ts, а структура типів строго визначена у файлі src/types/master.ts. Такий підхід гарантує типобезпечність, читаємість та повторне використання коду в межах усього проєкту.

Маршрут для отримання списку всіх майстрів реалізовано через HTTP-запит типу GET. Він підтримує кілька параметрів фільтрації: за компанією (companyId), користувачем (userId) та видимістю (is_visible). Це дозволяє динамічно формувати відповідні списки залежно від ролі користувача та його прав доступу. Наприклад, у публічному інтерфейсі кінцевим користувачам відображаються лише ті майстри, для яких увімкнено прапорець is_visible = true, тоді як адміністратори мають розширений доступ до всієї інформації.

Інформацію про конкретного майстра можна отримати через GET, що знаходиться в /api/master/[id] та робить запит про конкретного майстра за його id. Для перетворення сирого запису з PocketBase у зручний формат застосовується функція mapToMaster(), яка формує повноцінний об’єкт типу Master, зокрема генерує повний URL до фото через метод pb.files.getURL(...).

Створення нового майстра здійснюється через запит POST, який знаходиться в /api/master. Для цього формується об’єкт FormData, до якого додаються поля first_name, last_name, email, position, файл для photo, а також пов’язаний користувач і перелік послуг, який буде доступний для цього майстра. Ця логіка реалізована у функції createMaster(), яка відповідає за валідацію та серіалізацію даних перед відправкою (лістинг 3.3).

Лістинг 3.3 – Метод createMaster() нового майстра у системі

```
export const createMaster = async (data: CreateMasterData):
Promise<Master> => {
  const formData = new FormData();
    Object.entries(data).forEach(([key, value]) => {
    if (value !== null) {
      if (Array.isArray(value)) {
        value.forEach((val) => formData.append(key, val));
      } else {formData.append(key, value);}}
  });
  const response = await fetch('/api/master', {
    method: 'POST', body: formData});
  if (!response.ok) { throw new Error("Failed to create
master");}
  const record = await response.json() as MasterRecord;
  return mapToMaster(record, pb);
};
```

Оновлення інформації про майстра відбувається за допомогою PATCH /api/master/[id]. Як і при створенні, дані передаються через FormData, що дозволяє оновлювати окремі поля без повної заміни запису. Функція updateMaster() вміє обробляти як звичайні поля, так і масиви – наприклад, перелік нових послуг, що прив’язуються до майстра.

Окрім основних CRUD-операцій, реалізовано функцію updateMasterRating(), яка дозволяє оновити поле rating_value для конкретного майстра, наприклад, після залишення клієнтом відгуку. Також доступна утиліта getMasterByUserId(), яка дозволяє знайти майстра за ідентифікатором пов’язаного користувача, що зручно при побудові персонального кабінету.

Видалення майстра здійснюється поетапно: спочатку система перевіряє наявність пов’язаного користувача, і якщо такий існує, він видаляється разом із записом майстра. Вся ця логіка інкапсульована у функції deleteMaster(), яка забезпечує цілісність пов’язаних даних.

Окремо реалізовано логіку управління графіком роботи майстрів. Дані про доступні дні та години зберігаються у колекції master_working_hours. Функція updateWorkingHours() відповідає за зміну графіку роботи (лістинг 3.4). Для отримання графіку використовується маршрут GET

/api/master/[id]/working-hours, який повертає перелік записів з вказанням дня тижня, часу початку та завершення, а також прапорцем enabled, що вказує на активність зміни.

Лістинг 3.4 – Функція updateWorkingHours() для оновлення графіку роботи майстра

```
const updateWorkingHours = async (id: string, data: any):
Promise<any> => {
  const response = await
fetch(`/api/master_working_hours/${id}`, {
    method: 'PATCH',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(data),});
  if (!response.ok) {
    const error = await response.json();
    console.error("Failed to update working hours:", error);
    throw new Error("Failed to update working hours");}
  return response.json();
};
```

Таким чином, Master API забезпечує повний цикл управління інформацією про майстрів у системі, включаючи їх особисті дані, фото, графік роботи та прив’язку до компаній і послуг. Усі операції реалізовані із дотриманням принципів типізації, повторного використання коду та розмежування логіки між API та сервісним шаром.

3.2.3 Клієнтська логіка (Client API)

Клієнтська логіка вебзастосунку охоплює процес запису користувача на послугу, пошук доступних майстрів і послуг, а також можливість залишати відгуки після візиту. Вся ця функціональність реалізована через API-маршрути, доступ до яких не потребує ролі адміністратора.

Однією з ключових функцій є створення нового запису (бронювання). Цей процес реалізується через виклик функції createBooking(), яка виконує кілька послідовних кроків: перевірку доступності обраного часу, створення

або оновлення запису клієнта в базі даних, та створення нового запису у колекції booking. Якщо обраний слот часу зайнятий, функція викидає помилку, що дозволяє відобразити відповідне інформативне повідомлення користувачу.

Для визначення вільних часових слотів використовується функція getAvailableSlots() (лістинг 3.5), яка порівнює графік майстра з уже створеними бронюваннями у вибраний день. Графік зберігається у колекції master_working_hours, а самі онлайн-записи – у booking. Цей підхід дозволяє динамічно формувати доступні інтервали для запису.

Лістинг 3.5 – Функція getAvailableSlots() для отримання доступних слотів часу

```
export const getAvailableSlots = async (id: string, d: Date):
Promise<string[]> => {
  try {
    const day = d.getDay();
    const dateStr = d.toISOString().split("T")[0];
    const wh = await getMasterWorkingHrs(id, day);
    if (!wh?.enabled) return [];
    const booked = getBookedTimeSlots(await
getMasterBookings(id, dateStr));
    const start = getTimeInMinutes(wh.start_time);
    const end = getTimeInMinutes(wh.end_time);
    return TIME_SLOTS.filter(s => isSlotAvailable(s, start, end,
booked));
  } catch {return [];}
};
```

Також реалізовано можливість пошуку послуг і майстрів, прив’язаних до конкретної компанії. Для цього використовуються функції getServicesByCompany() та getMastersByCompany() (лістинг 3.6) , які виконують запити до відповідних ендпоінтів API з параметром ідентифікатора компанії. За необхідності може бути застосовано фільтрацію за параметром видимості (is_visible = true), щоб показати лише перевірених або актуальних фахівців та послуги.

Лістинг 3.6 – Функція getMastersByCompany() для отримання списку майстрів за певною компанією

```
export const getServicesByCompany = async (companyId: string,
signal?: AbortSignal, filterVisible: boolean = false):
Promise<Service[]> => {
  try {
    const url = new
URL(`/api/service?company=${companyId}&sort=name`,
window.location.origin);
    if (filterVisible) {url.searchParams.append('filterVisible',
'true');}
    const response = await fetch(url.toString(), { signal });
    if (!response.ok) {throw new Error("Failed to fetch");}
    const data = await response.json();
    return data;
  } catch (error) {return [];}
};
```

Після завершення послуги користувач має змогу залишити відгук про майстра. Відгуки зберігаються у колекції review і містять оцінку, текстовий коментар, а також зв’язок із записом booking, на основі якого було сформовано відгук. Для створення нового відгуку використовується функція createReview(). Після створення відгуку система автоматично перераховує середній рейтинг майстра шляхом виклику функції updateMasterRating(), яка розраховує нове значення на основі всіх залишених оцінок.

Крім того, для перегляду відгуку за конкретним записом реалізовано функцію getReviewByBookingId(), яка дозволяє отримати один відгук на основі параметра bookingId. А функція getReviewsByMasterId() надає змогу відобразити всі коментарі, залишені конкретному майстру. Таким чином, клієнтський API реалізує повний цикл взаємодії кінцевого користувача з системою.

3.3 Користувацький інтерфейс (UI)

Усі сторінки вебзастосунку структуровані відповідно до маршрутизатора App Router у рамках директорії app/, де кожна вкладена

папка або файл відповідає окремому маршруту та HTTP-шляху. Наприклад, app/page.tsx відповідає за landing page, app/company/[id]/page.tsx – за перегляд конкретної компанії.

Візуальні елементи інтерфейсу реалізовано у вигляді компонентів React, що забезпечують повторне використання UI-блоків, таких як картки майстрів, форми запису тощо. Для побудови адаптивного інтерфейсу застосовано TailwindCSS, який дозволяє швидко та ефективно описувати стилі безпосередньо у JSX-коді. Компоненти з бібліотеки shadcn/ui, зокрема Button, Input, Dialog, Toaster, активно використовуються на всіх сторінках, формах і в модальних вікнах.

Особливу увагу приділено взаємодії з користувачем: у разі успішного створення бронювання, помилки валідації або зміни статусу запису, система відображає відповідні повідомлення через компонент toast. Логіка форм реалізована з використанням react-hook-form, що спрощує управління станом введених даних і дозволяє інтегруватися з бібліотеками валідації, зокрема Zod, для опису та перевірки схеми введених даних.

Також реалізовано умовне відображення компонентів залежно від ролі користувача. Наприклад, кнопки редагування або адміністративного схвалення відображаються лише для адміністраторів. Це підвищує зручність і безпеку використання системи.

Загалом, клієнтська частина застосунку побудована за принципами модульності й адаптивності, із дотриманням сучасних UX-практик. Це дозволяє забезпечити зручну та стабільну роботу вебінтерфейсу на різних типах пристроїв.

3.4 Використання Pocketi як хостингу для PocketBase

Для хостингу вебзастосунку було обрано Pocketi – спеціалізовану хмарну платформу, розроблену саме для розгортання, адміністрування та масштабування інстансів PocketBase. Цей вибір обумовлений зручністю

налаштування, стабільністю роботи та відсутністю необхідності самостійного налаштування серверної інфраструктури.

Процес публікації бекенду був інтуїтивно зрозумілим і не потребував складної конфігурації. Після створення облікового запису в системі Pocketi, достатньо було обрати конфігурацію інстансу, вказати бажану назву та завантажити резервну копію локального проєкту PocketBase.

Сервіс автоматично розгортає бекенд, генерує унікальний URL-адрес для доступу до API, а також надає TLS-сертифікат для безпечної взаємодії через HTTPS. Адміністративна панель дозволяє візуально управляти колекціями, користувачами, медіафайлами та правилами доступу (рисунок 3.2).

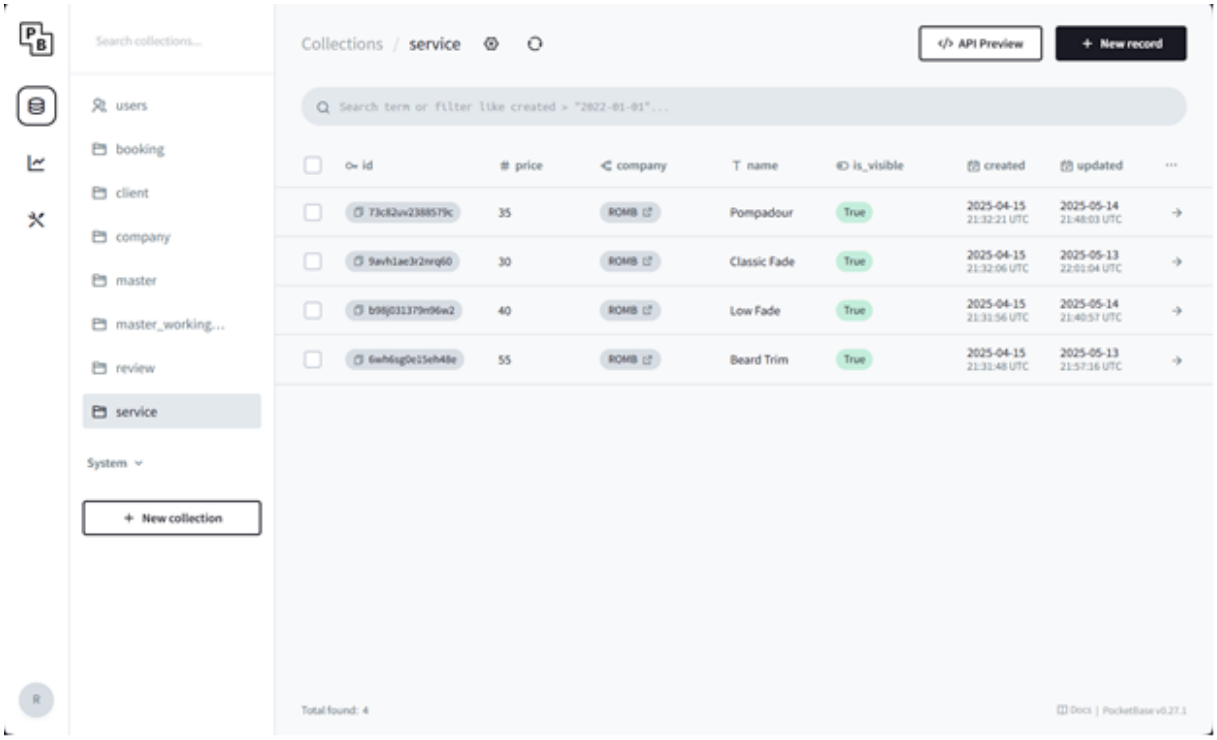

Рисунок 3.2 – База даних вебзастосунку для онлайн-запису клієнтів

З’єднання з Pocketi з клієнтської частини здійснюється дуже просто, через URL інтсансу (наприклад https://pma2dsiyaaxxepkgujkr-152-53-241-106.cloud.pocketi.dev). Цей URL вказується у конфігурації PocketBase SDK у серверних функціях.

3.5 Деплой на Vercel

Фронтенд-частину вебзастосунку, було розгорнуто на хмарній платформі Verce. Дана платформа забезпечує повну інтеграцію з системами керування версіями GitHub, GitLab та Bitbucket, що дозволяє автоматизувати процес розгортання завдяки вбудованій CI/CD-підсистемі.

Після авторизації на Vercel та підключення репозиторію, було обрано гілку для деплою та додано необхідні змінні середовища (POCKETBASE_URL, RESEND_API_KEY, GOOGLE_MAPS_API_KEY, RECAPTCHA_SECRET_KEY) у відповідному розділі налаштувань проєкту. Платформа автоматично виявляє тип застосунку як Next.js і запускає процес збірки з використанням стандартної конфігурації без необхідності ручного втручання.

Завдяки підтримці App Router, усі файли, розміщені в директорії app/api, інтерпретуються як серверні функції (Serverless Functions) і автоматично розгортаються на інфраструктурі Vercel. Результатом є повнофункціональний вебзастосунок із підтримкою SSR (server-side rendering), динамічних API-обробників, масштабованої інфраструктури та можливістю попереднього перегляду (Preview Deployments) при роботі з гілками, відмінними від основної.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Звичайний користувач

Щоб потрапити у вебзастосунок, користувач має перейти за спеціальним посиланням, наприклад /8k9kg9w103l2jv3, яке є унікальним для кожної компанії. Після переходу, користувач потрапляє на головну сторінка компанії де відображається блок з логотипом та назвою компанії а також , доступні варіанти запису та посилання на сторінку деталей про компанію (рисунок 4.1).

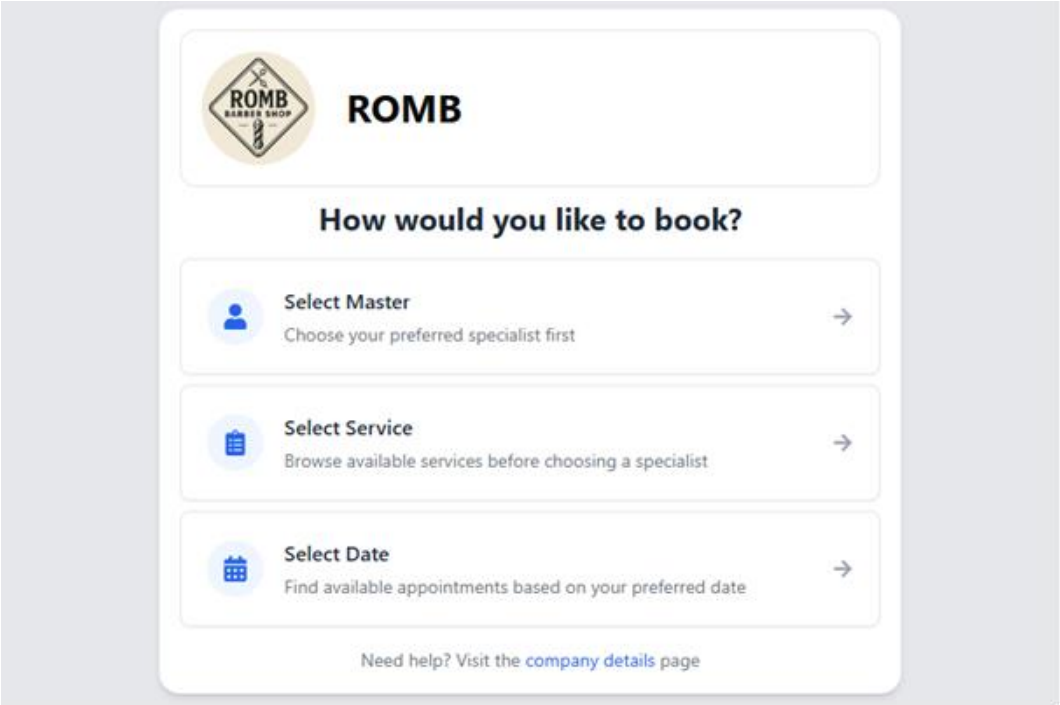

Рисунок 4.1 – Сторінка певної компанії

На екрані відображаються три блоки для вибору способу запису. Запис за майстром – відкриває список доступних майстрів одразу з вибором дати, після чого користувач переходить до вибору послуги. Запис за послугою – дозволяє спочатку обрати конкретну послугу, а потім відповідного майстра та дату. Запис за датою – відкриває календар з слотами доступного часу на

обрану дату, з можливістю вибору послуги та майстра наступними кроками.

У нижній частині сторінки розміщене посилання на сторінку з детальною інформацією про компанію, де вказаний список всіх послуг та відображаються всі майстри цієї компанії. Після натискання на будь-який із варіантів запису користувач переходить до відповідного етапу, наприклад, до вибору майстра (рисунок 4.2).

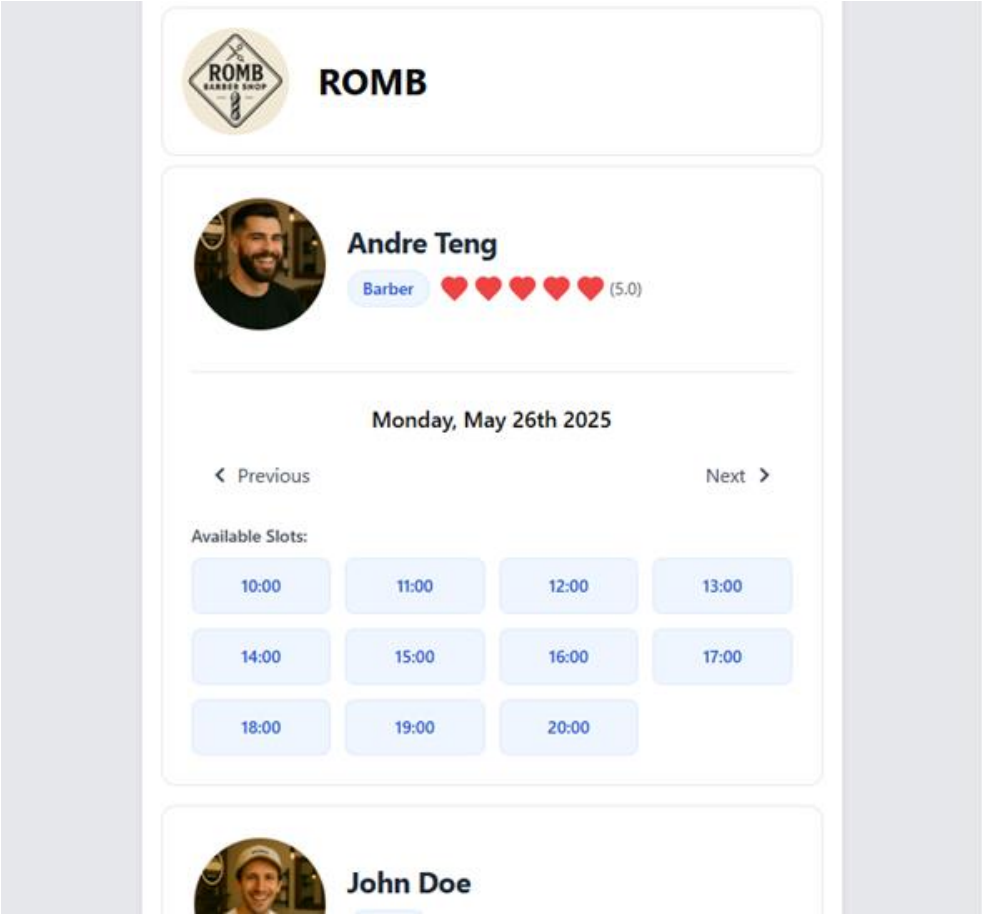

Рисунок 4.2 – Вибір майстра

Для кожного майстра відображаються фотографія, ім’я та прізвище, спеціалізація (позначена міткою), рейтинг у вигляді сердець та числове значення рейтингу. Також є можливість переглянути усі коментарі у спеціальному модальному вікні. Для цього потрібно натиснути на рейтинг майстра.

Під кожним профілем майстра відображається календарний блок з датою перегляду доступного часу, навігаційними кнопками для зміни дати та

списком доступних часових слотів для запису на цю дату. Це дозволяє клієнту одразу бачити розклад майстра на конкретний день та обрати зручний час без додаткових кроків. Після вибору часу користувача буде перенаправлено до кроку вибору послуги, які доступні для обраного майстра (рисунок 4.3).

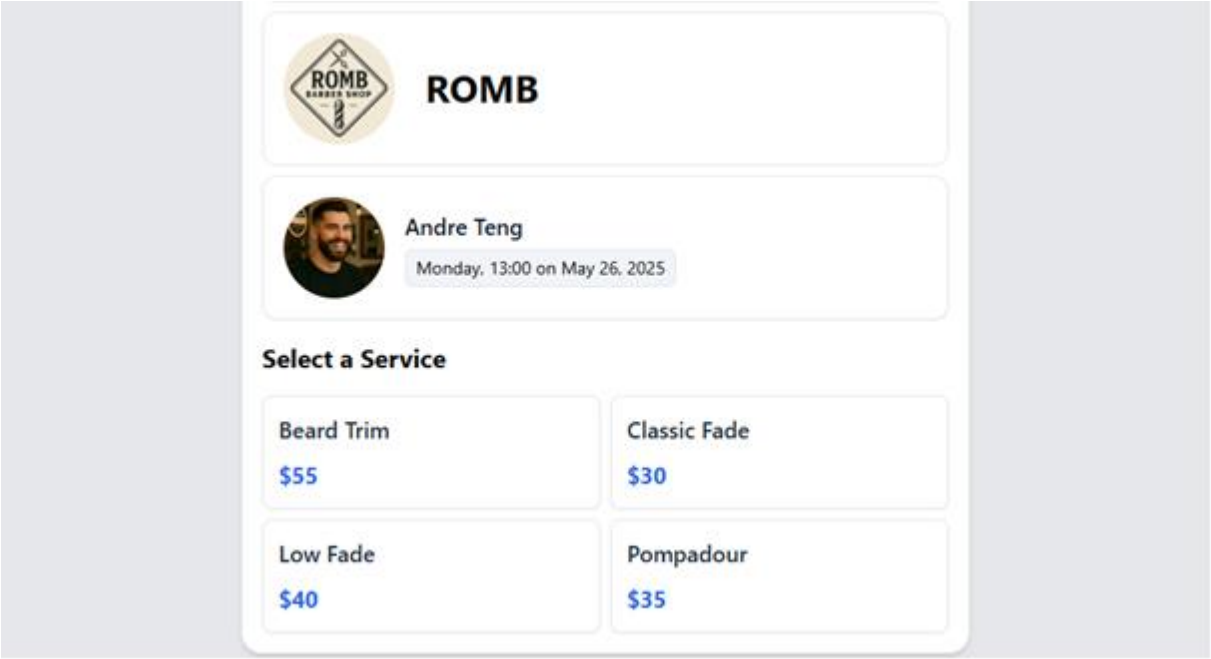

Рисунок 4.3 – Сторінка вибору послуги

У верхній частині інтерфейсу повторно відображається логотип та назва компанії, а також вибраний майстер і обрана дата та час у зручному форматі. Нижче розміщується блок у якому відображається перелік послуг, доступних у даного майстра. Кожна послуга представлена у вигляді окремого інтерактивного елементу з назвою та вартістю. Дані про послуги підтягуються автоматично з бази відповідно до зв’язку майстра з наявними сервісами.

Користувач може обрати одну з послуг, натиснувши на відповідну картку. Після цього він переходить до завершального кроку – підтвердження запису із введенням персональних даних (рисунок 4.4).

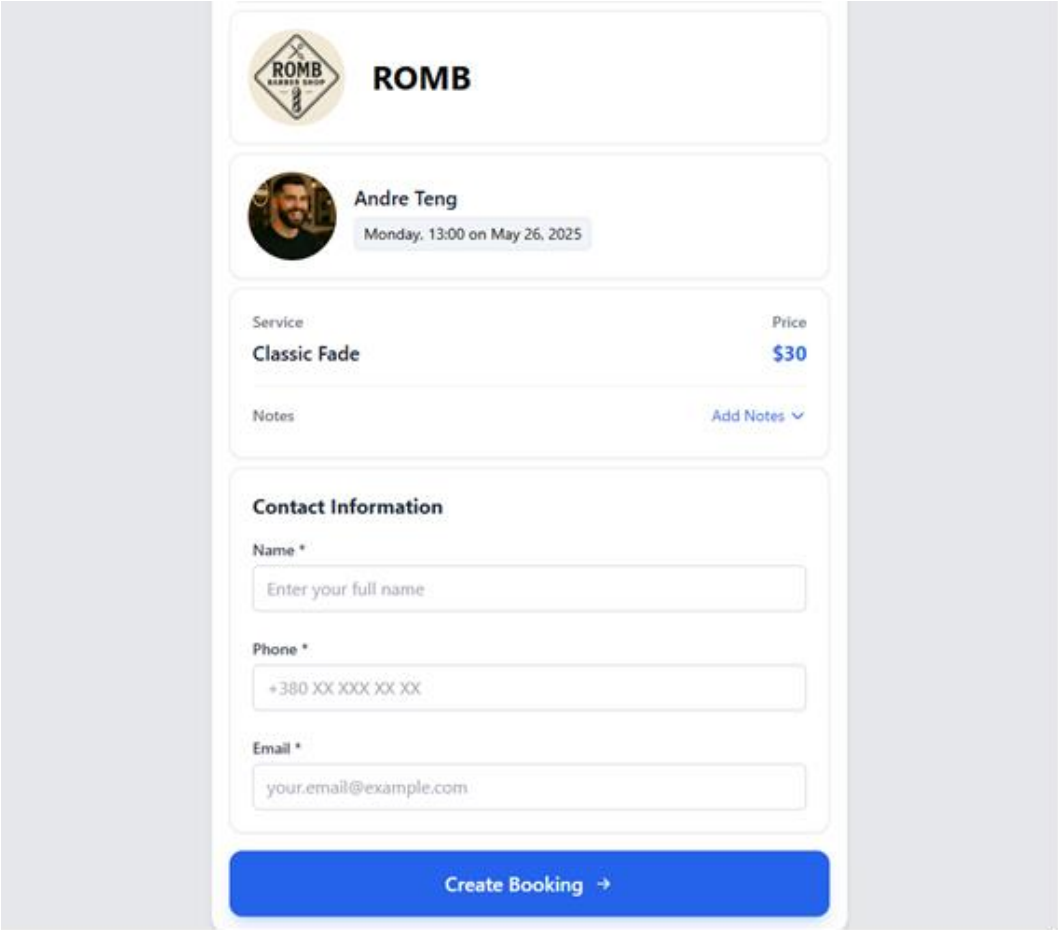

Рисунок 4.4 – Сторінка підтвердження запису на послугу

На екрані відображається узагальнена інформація про запис: компанія, майстер, дата та час та обрана послуга з вказаною вартістю. Додатково доступне поле для внесення коментаря або побажання до запису, який буде відображено майстру та адміністратору. Ще нижче розташований блок в якому користувач заповнює своє ім’я, номер телефону та електронну пошту. Усі ці поля є обов’язковими. Валідація форми виконується безпосередньо в реальному часі, а у випадку помилки – користувачу буде показано відповідне повідомлення.

Після заповнення форми і натискання кнопки Create Booking, відбувається обмін даними з сервером. У випадку успішного створення запису користувач буде перенаправлений на сторінку з детальною інформацією про запис (рисунок 4.5).

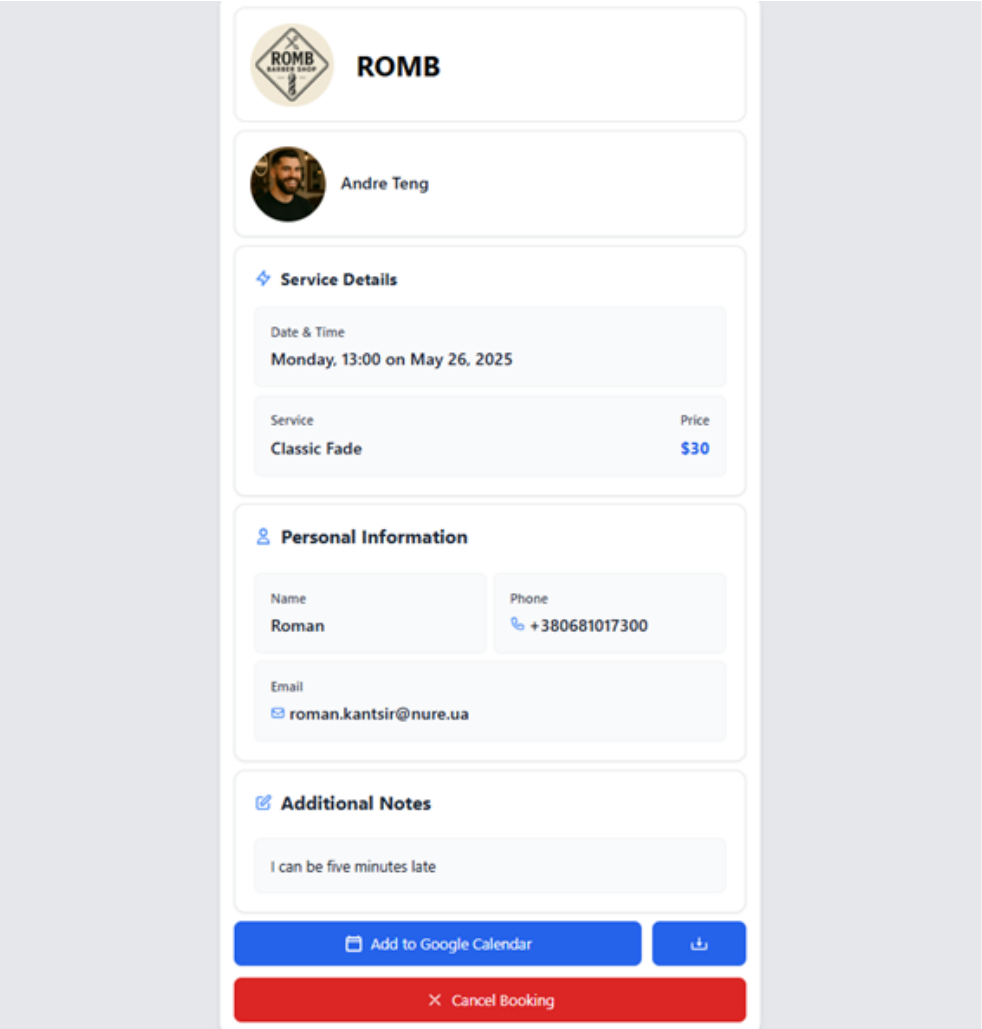

Рисунок 4.5 – Сторінка перегляду підтвердженого запису

На екрані відображається узагальнена інформація про запис: компанія, майстер, обрана дата й час, послуга та її вартість. У секції з персональною інформацією, де вказуються контактні дані клієнта – ім’я, номер телефону та електронна пошта. Якщо користувач залишив додаткові примітки, то вони також будуть виведені окремим блоком.

Крім цього, на сторінці доступна можливість експортувати запис до Google Calendar, що дозволяє зберегти подію з точним часом у особистий календар користувача, що буде додатковим нагадуванням.

Якщо у клієнта змінились плани, то в нижній частині інтерфейсу розміщена кнопка, яка дозволяє користувачу самостійно скасувати запис. Після натискання система відображає відкриває модальне вікно де користувач повинен підтвердити свої дії. Після підтвердження користувача

буде перенаправлено на початкову сторінку компанії.

Після відвідання послуги статус онлайн-запису змінюється на «виконано», і клієнту стає доступною можливість залишити відгук, повернувшись за тим самим посиланням (рисунок 4.6).

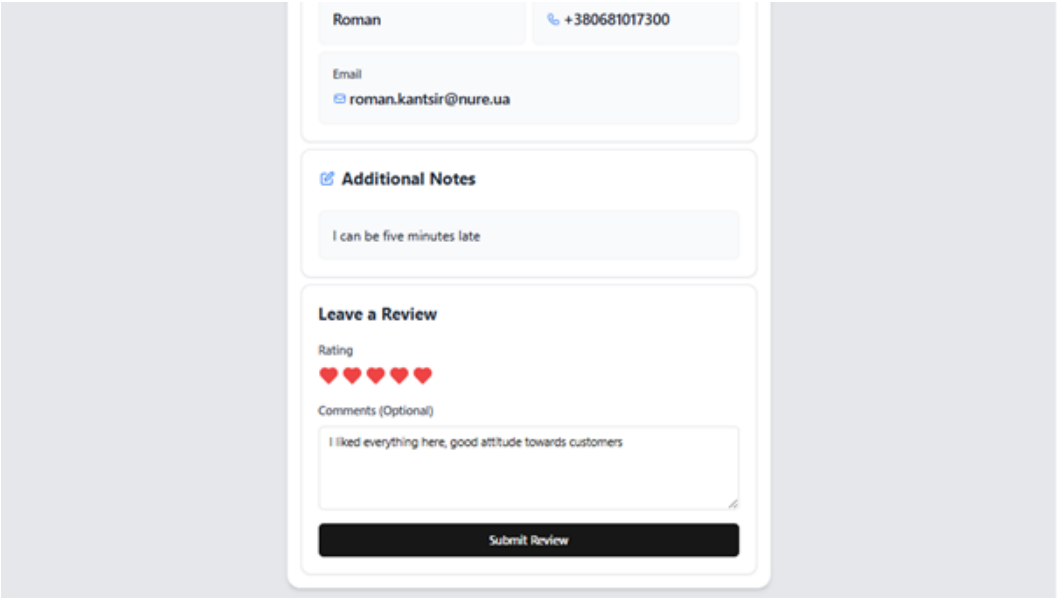

Рисунок 4.6 – Сторінка з формою для відгуку

Відгуки впливають на загальний рейтинг майстра та будуть доступні для інших клієнтів.

4.2 Адміністратор

Для отримання доступу до інтерфейсу адміністратора користувач повинен перейти за посиланням /admin, де необхідно пройти процедуру авторизації. Після успішної автентифікації, на основі встановленої ролі система дозволяє адміністратору потрапити на сторінку з переліком компаній, які були ним створені.

У випадку, якщо адміністратор входить у систему вперше, або ще не має жодної зареєстрованої компанії, тоді автоматично відкривається модальне вікно з формою для створення першої компанії (рисунок 4.7).

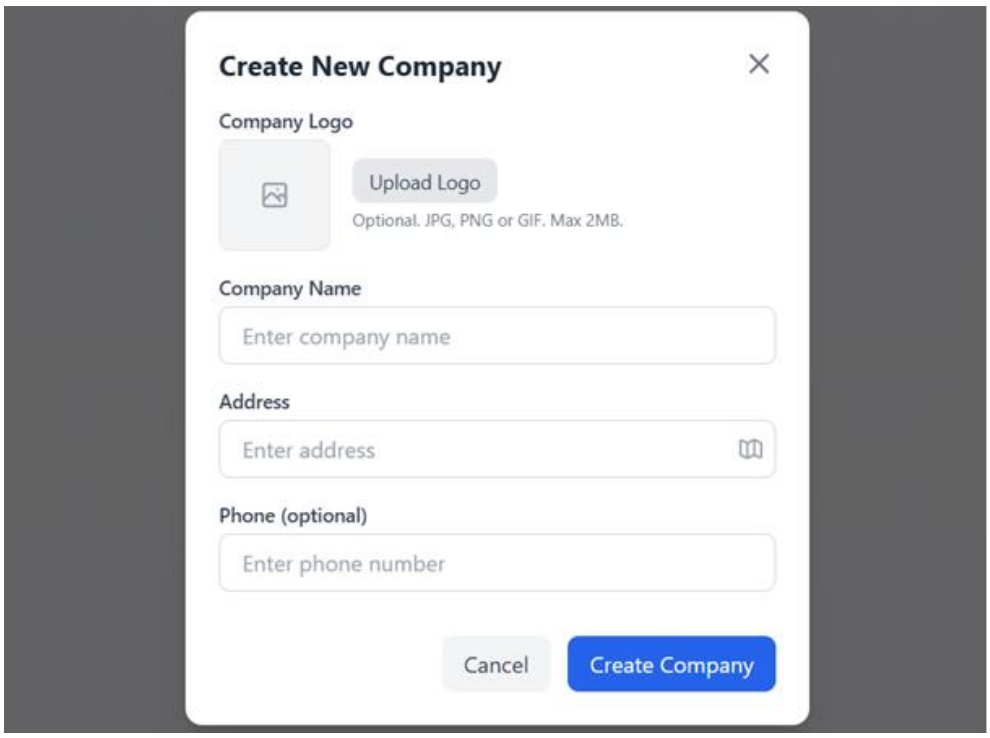

Рисунок 4.7 – Модальне вікно створення компанії

У цьому модальному вікні користувач має ввести усю необхідну інформацію: додати фото або логотип компанії, вказати адресу, яку можна встановити за допомогою спеціального модального вікна з інтегрованою картою, пошуком та детальними даними про обрану точку.

Такий підхід дозволяє одразу розпочати роботу з додавання основних відомостей, необхідних для подальшої взаємодії в системі. Після створення компаній вони відображаються на цій сторінці у вигляді карток (рисунок 4.8).

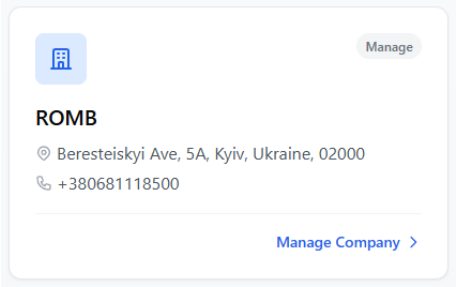

Рисунок 4.8 – Картка компанії

Кожна картка яких містить всю базову інформацію: назву компанії, адресу та контактний номер телефону. Натискання на картку перенаправляє

адміністратора на початкова сторінку компанії (рисунок 4.9).

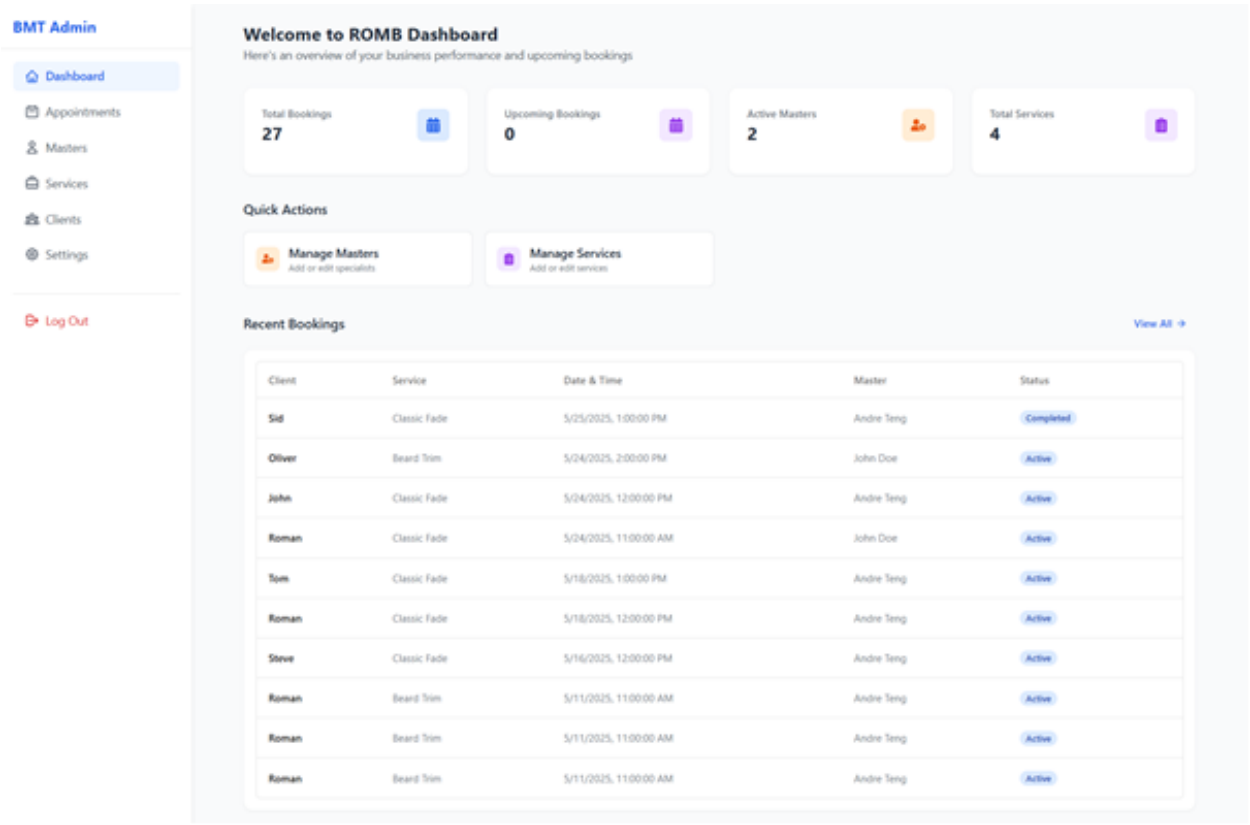

Рисунок 4.9 – Початкова сторінка адміністратора

У верхній частині сторінки відображаються ключові статистичні показники: загальна кількість записів, кількість запланованих записів, кількість активних майстрів, а також загальна кількість створених послуг. Ці показники представлені у вигляді окремих блоків, що дозволяє адміністратору швидко оцінити поточний стан компанії або зрозуміти загальне навантаження на майстрів.

Нижче розміщено блок швидких дій, що містить кнопки для переходу до розділів управління майстрами та послугами. Це дозволяє швидко перейти з панелі до важливих функціональних сторінок і здійснювати основні адміністративні дії, пов’язані з додаванням, редагуванням або видаленням даних.

У нижній частині сторінки розташовано таблицю з останніми записами клієнтів, у якій зазначено ім’я клієнта, обрану послугу, дату та час візиту,

майстра, до якого здійснено запис, а також поточний статус запису. Така структура дозволяє адміністратору швидко відслідковувати актуальну інформацію щодо активності компанії, контролювати процес надання послуг і вчасно реагувати на зміни.

Інтерфейс є лаконічним та інтуїтивно зрозумілим, що забезпечує ефективну роботу навіть за умови великої кількості даних. Усі елементи панелі згруповані логічно, а доступ до розширених функцій реалізовано через переходи у відповідні розділи, зокрема до сторінки управління майстрами (рисунок 4.10).

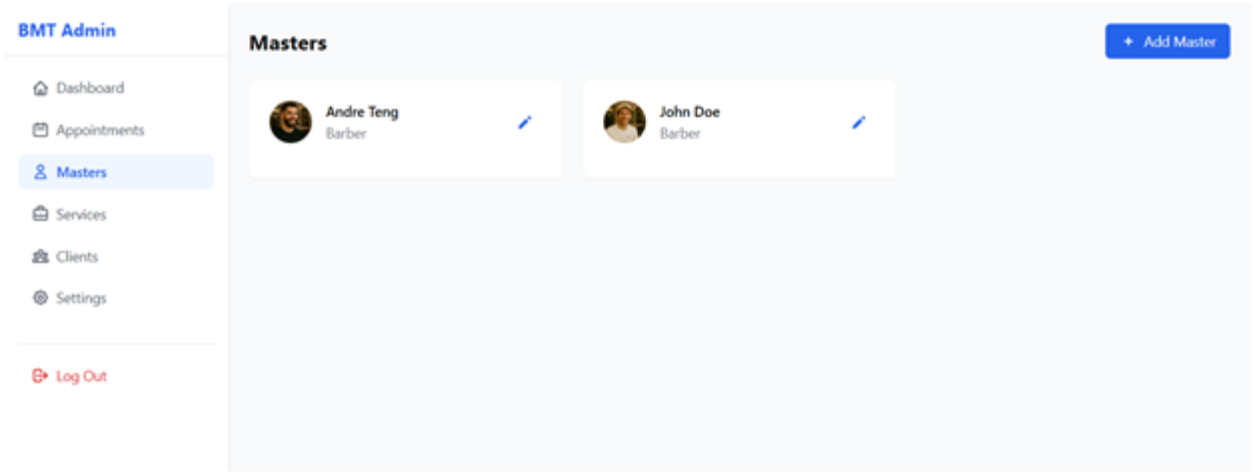

Рисунок 4.10 – Сторінка зі списком майстрів

Ця сторінка містить перелік майстрів, прикріплених до обраної компанії. Кожен майстер представлений у вигляді окремої картки, яка включає його ім’я, фотографію та спеціалізацію. Візуальне оформлення карток дозволяє дуже швидко орієнтуватися в списку майстрів та знаходити потрібного майстра за зовнішнім виглядом чи фахом.

У верхній частині сторінки знаходиться кнопка Add Master, при клікі на яку відкривається модальне вікно для створення нового майстра. У правій частині кожної картки розміщено яскраву іконку редагування, при натисканні на яку відкривається модальне вікно для оновлення даних відповідного майстра (рисунок 4.11).

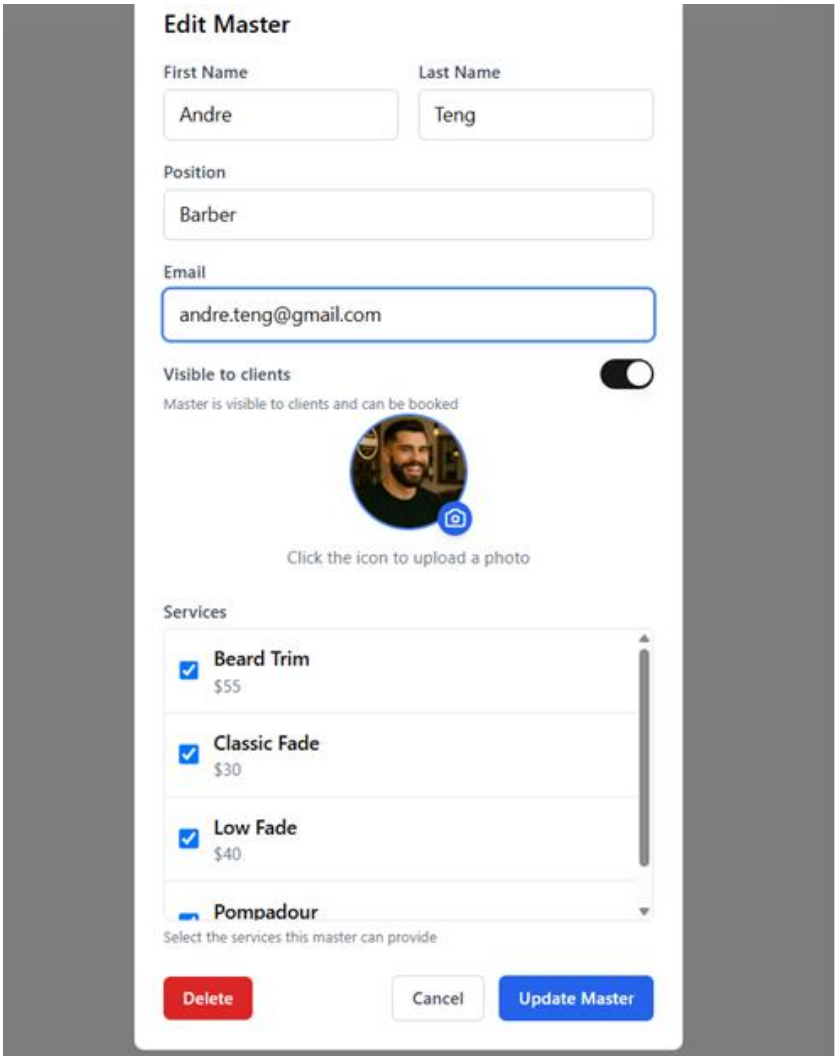

Рисунок 4.11 – Модальне вікно для редагування майстра

Після відкриття модального вікна адміністратор отримує доступ до форми, яка використовується як для створення нового майстра, так і для редагування або видалення вже існуючого. У ній передбачено поля для введення імені, прізвища, електронної пошти та посади. Також можна завантажити фотографію майстра, увімкнути або вимкнути його видимість для клієнтів, що впливає на можливість здійснення запису. У нижній частині форми розміщено список послуг, які адміністратор може призначити конкретному майстру, обравши з доступного переліку. Для завершення дії передбачено кнопки збереження, скасування змін або повного видалення майстра зі списку.

Переходячи до наступної ключової сторінки з послугами, адміністратор

отримує можливість переглядати, створювати та редагувати перелік послуг, доступних у компанії (рисунок 4.12).

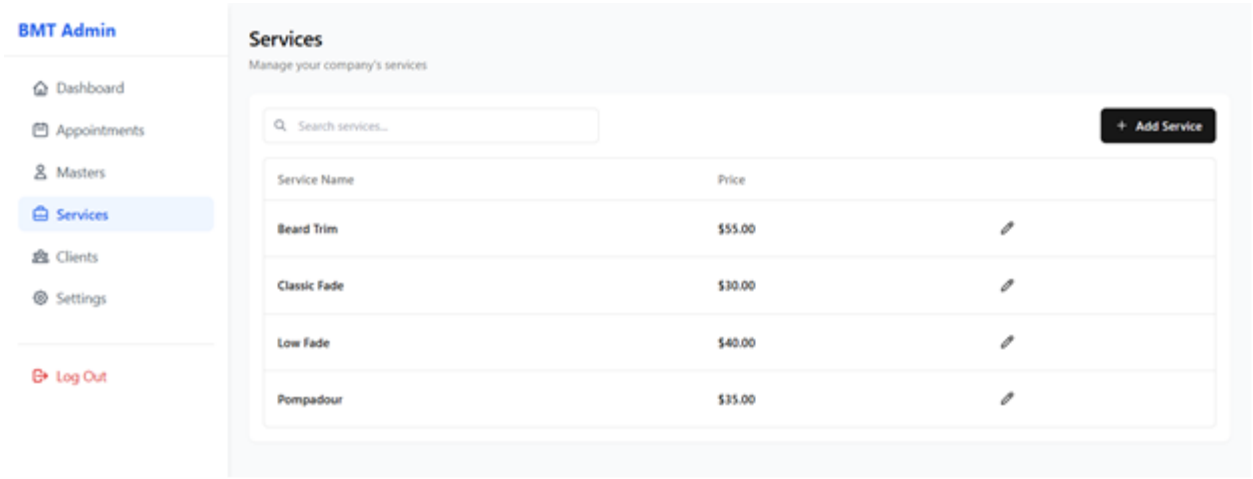

Рисунок 4.12 – Сторінка з переліком послуг

Сторінка дозволяє адміністратору зручно керувати переліком послуг, які додані в межах обраної компанії. Основна частина інтерфейсу представлена у вигляді таблиці, в якій відображаються назви всіх доступних послуг та їхня вартість. Це дає змогу швидко оцінити перелік сервісів та їхню цінову політику.

Кожен рядок таблиці супроводжується іконкою редагування, що відкриває модальне вікно з формою для оновлення відповідної послуги. Це дозволяє адміністратору безпосередньо змінювати ціну або назву без необхідності переходу на інші сторінки.

Для швидкого пошуку передбачено поле фільтрації за назвою, що особливо корисно при наявності великої кількості записів. У правому верхньому куті розміщена кнопка Add Service, натискання на яку відкриває форму створення нової послуги з можливістю вказати назву та ціну. Уся логіка реалізована з урахуванням зручності та мінімізації кількості дій для адміністрування.

Наступним важливим розділом є сторінка Appointments, яка містить інформацію про всі створені клієнтські записи (рисунок 4.13).

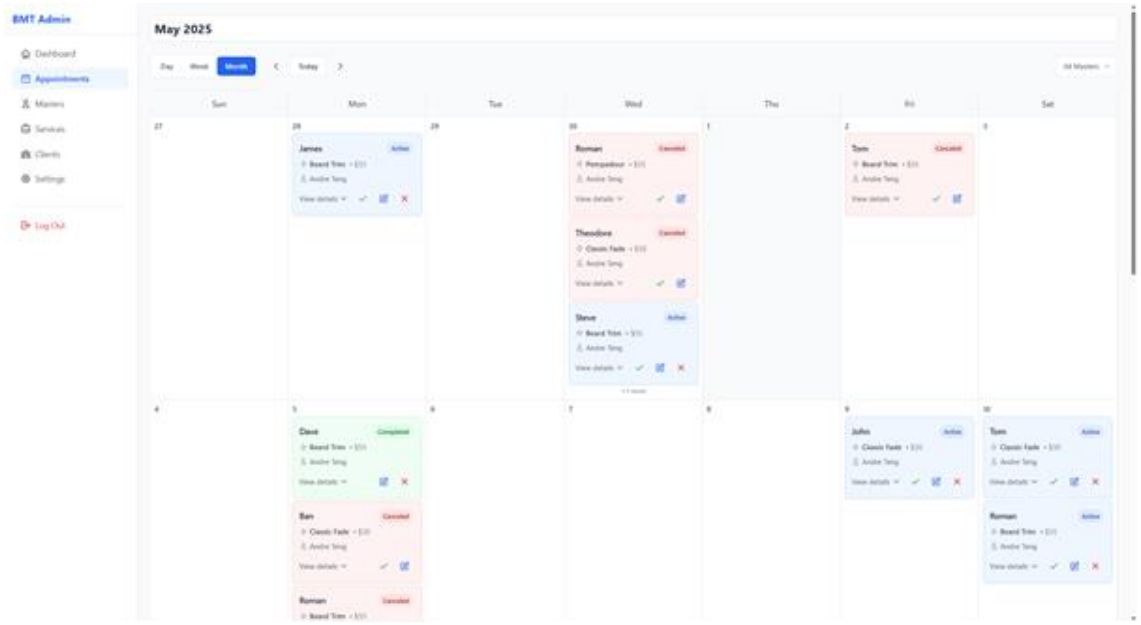

Рисунок 4.13 – Сторінка з відображення усіх записів компанії

На цій сторінці у вигляді зручного календаря відображаються записи до майстрів. Це дозволяє швидко оцінити навантаження та у загальному є дуже важливою функціональністю адміністратора.

4.3 Супер-адміністратор

Для отримання доступу до інтерфейсу супер-адміністратора користувач повинен перейти за посиланням /admin. Після успішної авторизації супер-адміністратор потрапляє на сторінку з переліком компаній (рисунок 4.14).

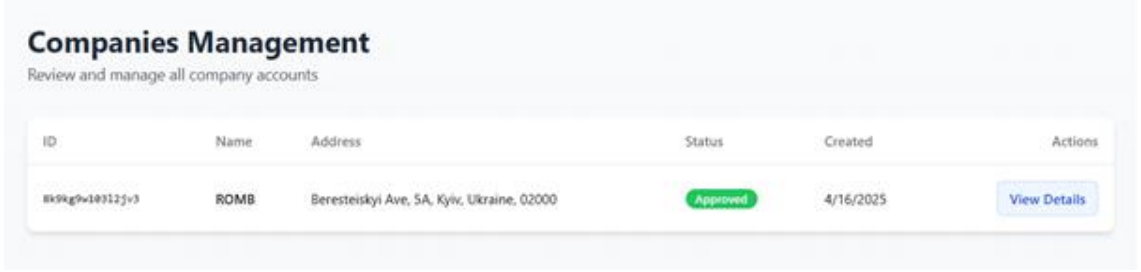

Рисунок 4.14 – Панель керування компаніями для супер-адміністратора

На цій сторінці відображається таблиця з ключовою інформацією про всі компанії: унікальний ідентифікатор (ID), назва компанії, адреса, статус,

дата створення, а також доступ до додаткових дій через кнопку View Details.

Після натискання на кнопку відкривається модальне вікно з розширеною інформацією про компанію (рисунок 4.15).

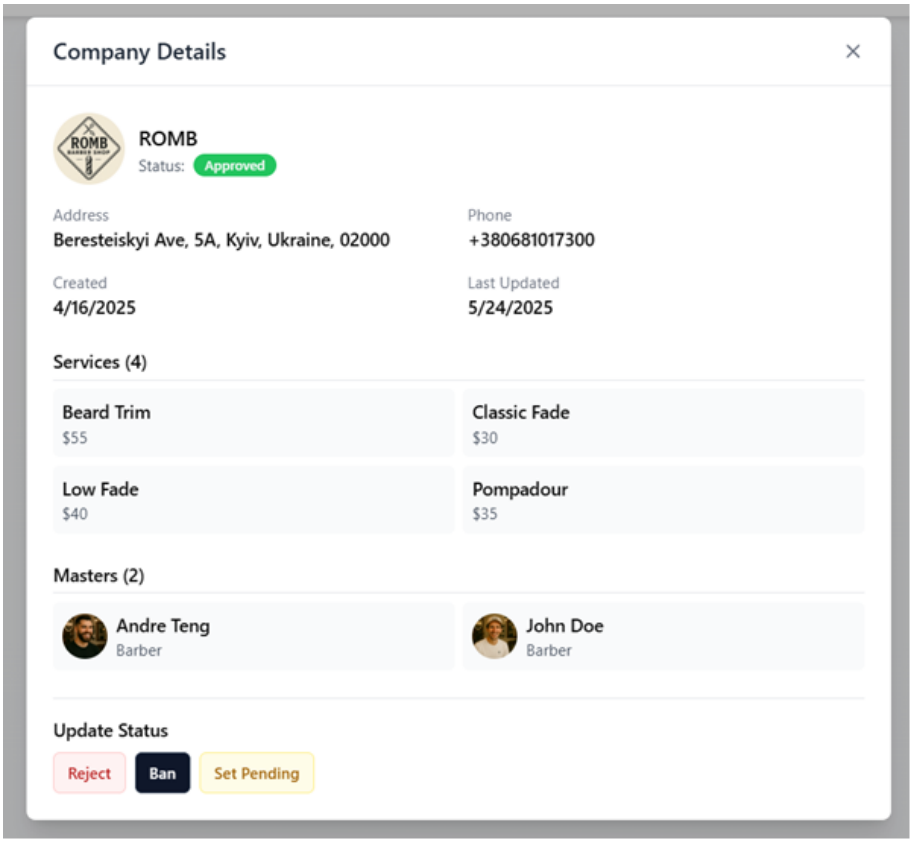

Рисунок 4.15 – Модальне вікно з деталями компанії

У модальному вікні відображається детальна інформація про вибрану компанію. У верхній частині розмішено логотип, назву, поточний статус модерації (наприклад, Approved, Rejected, Banned), адресу, контактний номер телефону, а також дата створення та останнього оновлення профілю компанії.

Нижче подано перелік доступних послуг компанії із зазначенням їхніх цін, а також список майстрів, прикріплених до цієї компанії. У нижній частині модального вікна розташовані кнопки для зміни статусу компанії, що дозволяє супер-адміністратору швидко приймати рішення щодо щодо модерації компанії, що переглядається.

4.4 Майстер

Для отримання доступу до інтерфейсу майстра користувач повинен перейти за посиланням /master, де необхідно пройти процедуру авторизації, через електронну пошту та пароль, що були надіслані на пошту під час створення облікового запису адміністратором. Після чого майстер потрапляє до особистого кабінету, на першу сторінку, яка відповідає за перегляд поточного розкладу та управління записами (рисунок 4.16).

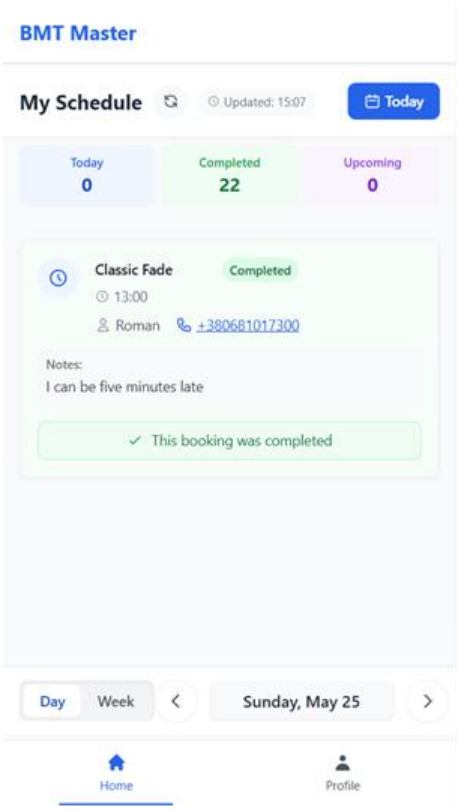

Рисунок 4.16 – Сторінка з відображення записів

На головній сторінці особистого кабінету майстер отримує доступ до зведеної інформації про власний робочий розклад. За замовчуванням відкривається поточна дата, та всі записи відсортовані відповідно до часу їх проведення. Кожен запис представлений у вигляді окремого блоку, який містить назву послуги, час візиту, ім’я клієнта, номер телефону, а також додаткові примітки, якщо вони були вказані під час запису. Поряд із кожним

записом відображається його статус. У нижній частині сторінки реалізовано інструменти для навігації за датами, а також перемикач режимів перегляду – денний, або тижневий. У тижневому режимі записи автоматично групуються відповідно до днів тижня.

Нижнє меню навігації дає змогу швидко перемикатися між основними розділами особистого кабінету. Другий пункт меню відкриває сторінку профіль, на якій майстер може налаштовувати власний робочий графік (рисунок 4.17).

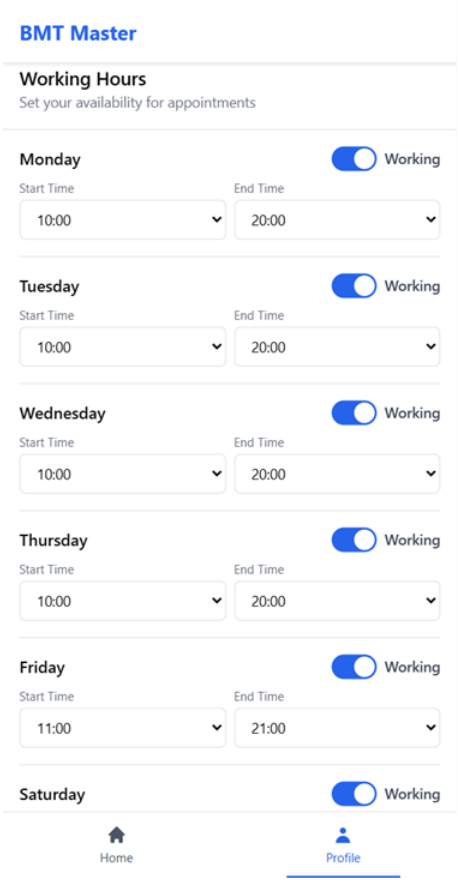

Рисунок 4.17 – Сторінка налаштування робочого часу

У цьому розділі доступна можливість вказати години та дні, у які майстер приймає клієнтів, що дозволяє гнучко керувати власним графіком та формувати розклад для бронювання в межах системи. Ці налаштування безпосередньо впливають на доступність майстра для записів клієнтів.

ВИСНОВКИ

Робота присвячена розробці веб-застосунку для онлайн-запису клієнтів на послуги. Основна мета полягає в автоматизації процесу онлайн-запису без потреби в реєстрації користувачів, що підвищує зручність для клієнтів та ефективність для бізнесу.

Актуальність проєкту зумовлена зростаючим попитом на доступні та прості в користуванні системи онлайн-запису, особливо серед малого бізнесу. Рішення орієнтоване на власників салонів, барбершопів і майстрів, які не мають власних сайтів або CRM.

У процесі дослідження було проаналізовано існуючі сервіси та визначено їхні обмеження та на основі цього сформовано перелік необхідної функціональності.

Розроблений веб-застосунок реалізує:

- підтримку чотирьох ролей (супер-адміністратор, адміністратор, майстер, клієнт);
- онлайн-запис з різними сценаріями (за майстром, послугою або датою);
- керування компаніями, послугами, майстрами, графіками;
- сповіщення, відгуки, календар та аналітику.

Для реалізації застосовано стек: Next.js, TypeScript, Tailwind CSS, PocketBase, Vercel. Бекенд розгорнуто на Pocketi, що забезпечує швидке адміністрування бази.

Результати роботи представлені у рамках п’ятнадцятої міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Канцір Р. Б. Розробка вебзастосунку для онлайн-запису та управління бізнес-послугами [Текст] / Р. Б. Канцір, Д. О. Тимошенко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. – 2025. – 20 р.

2. Wamba S. The Role of Information and Communication Technology in Business Process Automation: A Review of Literature [Текст] / S. Wamba, S. Akter // Journal of Enterprise Information Management. – 2019. – P. 117–129.

3. Lacity M. Robotic Process Automation: The Next Transformation Lever for Business [Текст] / M. Lacity, L. Willcocks // Journal of Information Technology Teaching. – 2017. – P. 110–128.

4. Google Calendar [Електронний ресурс] – Режим доступу : www/ URL: https://developers.google.com/workspace/calendar – 05.06.2025 р. – Загол. з екрану.

5. Altegio [Електронний ресурс] – Режим доступу : www/ URL: https://alteg.io/uk – 05.06.2025 р. – Загол. з екрану.

6. Framer Motion Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://www.framer.com/motion/ – 05.06.2025 р. – Загол. з екрану.

7. React Hook Form Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://www.react-hook-form.com/get-started/ – 05.06.2025 р. – Загол. з екрану.

8. Cursor Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://docs.cursor.com/welcome – 05.06.2025 р. – Загол. з екрану.

9. TypeScript Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://www.typescriptlang.org/docs/ – 05.06.2025 р. – Загол. з екрану.

10. ECMAScript Language Specification [Електронний ресурс] – Режим

доступу : www/ URL: https://tc39.es/ecma262/ – 05.06.2025 р. – Загол. з екрану.

11. ESLint Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://eslint.org/docs/latest/ – 05.06.2025 р. – Загол. з екрану.

12. Goldberg, J. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript [Текст] / J. Goldberg // O'Reilly Media. – 2022. – 432 р.

13. Next.js Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://nextjs.org/docs – 05.06.2025 р. – Загол. з екрану.

14. Building Modern Web Apps with Next.js and Vercel [Електронний ресурс] – Режим доступу : www/ URL: https://www.futurice.com/blog/building-modern-web-apps-next-js-vercel – 05.06.2025 р. – Загол. з екрану.

15. Manning, E. Single Page Applications in Depth [Текст] / E. Manning // Leanpub. – 2020. – 324 р.

16. Mardan, A. SPA Design and Architecture: Understanding Single Page Web Applications [Текст] / A. Mardan // Manning Publications. – 2016. – 275 р.

17. Next.js – API Routes [Електронний ресурс] – Режим доступу : www/ URL: https://nextjs.org/docs/pages/building-your-application/routing/api-routes – 05.06.2025 р. – Загол. з екрану.

18. PocketBase Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://pocketbase.io/docs/ – 05.06.2025 р. – Загол. з екрану.

19. REST API Tutorial [Електронний ресурс] – Режим доступу : www/ URL: https://restfulapi.net/ – 05.06.2025 р. – Загол. з екрану.

20. Tailwind CSS Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://tailwindcss.com/docs – 05.06.2025 р. – Загол. з екрану.

21. Radix UI Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://www.radix-ui.com/docs – 05.06.2025 р. – Загол. з екрану.

22. Shadcn Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://ui.shadcn.com/docs – 05.06.2025 р. – Загол. з екрану.

23. AWS SDK – What is a Software Development Kit? [Електронний ресурс] – Режим доступу : www/ URL: https://aws.amazon.com/what-is/sdk/ –

05.06.2025 р. – Загол. з екрану.

24. Resend Email API Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://resend.com/docs – 05.06.2025 р. – Загол. з екрану.

25. Google Maps Platform Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://developers.google.com/maps/documentation – 05.06.2025 р. – Загол. з екрану.

26. Vercel Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://vercel.com/docs – 05.06.2025 р. – Загол. з екрану.

27. GitHub [Електронний ресурс] – Режим доступу : www/ URL: https://github.com – 05.06.2025 р. – Загол. з екрану.

28. Git Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://git-scm.com/docs – 05.06.2025 р. – Загол. з екрану.

29. Git Version Control [Електронний ресурс] – Режим доступу : www/ URL: https://techrepublic.com/article/version-control-benefits/ – 05.06.2025 р. – Загол. з екрану.

30. CI/CD: Automate your development workflow [Електронний ресурс] – Режим доступу : www/ URL: https://github.com/resources/articles/devops/ci-cd – 05.06.2025 р. – Загол. з екрану.