

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Вебсистема управління барбершопом»

Виконав студент групи ВТ-22-2
КОЛЕСНИК Олексій Андрійович

Керівник роботи:
ПОНТКІВСЬКИЙ Владислав Ігорович

Рецензент:
МОРОЗОВ Андрій Васильович

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Колесник Олексій Андрійович**

Керівник роботи: **Піонтківський Владислав Ігорович**

Тема роботи: **«Вебсистема управління барбершопом»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **Об'єктом дослідження є процес розробки та впровадження системи для управління барбершопом. Предметом дослідження є створення системи для автоматизованого управління барбершопом.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Піонтківський В. І.	02.03.2026р.	02.03.2026р.
2	Піонтківський В. І.	06.03.2026р.	06.03.2026р.
3	Піонтківський В. І.	24.04.2026р.	24.04.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка містить 126 сторінок, з них 87 сторінок основного тексту, 33 ілюстрацій, 13 таблиць, 1 додаток та 30 використаних літературних джерел. Система містить модулі для управління всіма процесами барбершопу, включаючи розклад та персонал.

Метою випускної кваліфікаційної роботи бакалавра є розробка вебсистеми для автоматизації управління барбершопом, що забезпечить задоволення бізнес-потреб, підвищення ефективності організації роботи та оптимізацію процесів обслуговування клієнтів.. Система реалізована у вигляді вебзастосунку з використанням монолітної архітектури на базі Laravel для серверної частини та Vue.js з TypeScript для клієнтської частини. Для організації взаємодії між частинами застосовано Inertia.js, що дозволяє реалізувати SPA-підхід без окремого API. Як СУБД використано PostgreSQL, що забезпечує надійне зберігання та обробку даних.

У роботі виконано аналіз предметної області та існуючих аналогів, на основі чого визначено вимоги до системи. Обґрунтовано вибір клієнт-серверної архітектури, розроблено структуру бази даних, визначено основні сценарії використання та логіку взаємодії між модулями системи.

Клієнтська частина забезпечує зручний інтерфейс для запису на послуги, перегляду майстрів та управління інформацією про барбершоп. Серверна частина реалізує бізнес-логіку, обробку запитів та взаємодію з базою даних.

КЛЮЧОВІ СЛОВА: ВЕБСИСТЕМА, БАРБЕРШОП, АВТОМАТИЗАЦІЯ, LARAVEL, VUE.JS, TYPESCRIPT, INERTIA.JS, POSTGRESQL, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, СЛОТ.

					ІПЗ.КР.Б – 121 – 26 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсистема управління барбершопом Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Колесник О.А.					3	126
Керівник.		Піонтківський В.І						
Рецензент		Морозов А.В.				Житомирська політехніка, група ВТ-22-2		
Зав. каф.		Вакалюк Т.А						

ABSTRACT

The bachelor's qualification thesis consists of a software complex and an explanatory note. The explanatory note consists of 126 pages, including 87 pages of main text, 32 figures, 13 tables, 1 appendices, and 30 references. The system includes modules for managing all barbershop processes, including scheduling and personnel management.

The goal of the bachelor's qualification thesis is to develop a web system for automating barbershop management, which ensures the satisfaction of business needs, increases the efficiency of organizational processes, and optimizes customer service workflows. The system is implemented as a web application using a monolithic architecture based on Laravel for the backend and Vue.js with TypeScript for the frontend. Inertia.js is used to connect the components, enabling a SPA approach without a separate API. PostgreSQL is used as the database management system, ensuring reliable storage and processing of data.

The work includes an analysis of the subject area and existing solutions, based on which system requirements were defined. A client-server architecture was chosen, the database structure was designed, and the main use cases and interaction logic between system modules were defined.

The client-side provides a convenient interface for booking services, viewing barbers, and managing barbershop information. The server-side implements business logic, request handling, and interaction with the database.

KEYWORDS: WEB SYSTEM, BARBERSHOP, AUTOMATION, LARAVEL, VUE.JS, TYPESCRIPT, INERTIA.JS, POSTGRESQL, CLIENT-SERVER ARCHITECTURE, SLOT.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ВИЗНАЧЕННЯ АРХІТЕКТУРИ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1. Визначення та постановка задачі (проблематика)	9
1.2. Аналіз аналогів програмного продукту	14
1.3. Визначення архітектури програмного продукту	23
1.4. Обґрунтування вибору засобів реалізації та вимоги до програмного забезпечення	26
Висновки до першого розділу	30
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ	32
2.1. Визначення варіантів використання та вимог системи	32
2.2. Розробка бази даних системи	39
2.3. Проєктування та реалізація алгоритмів роботи системи.....	50
2.4. Реалізація системи.....	57
Висновки до другого розділу	60
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ	61
3.1. Порядок встановлення та налаштування системи.....	61
3.2. Структура інтерфейсу	64
3.3. Тестування роботи програмного комплексу.....	78
Висновки до третього розділу	81
ВИСНОВКИ	83
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	85
ДОДАТКИ	88

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CMS – Система керування контентом
CSS – Каскадні таблиці стилів
HTML – Мова розмітки гіпертексту
HTTP – Протокол передачі гіпертексту
JS – Мова програмування JavaScript
TS – Мова програмування TypeScript
UI – Користувацький інтерфейс
UX – Користувацький досвід
ООП – Об'єктно-орієнтоване програмування
ПЗ – Програмне забезпечення
СУБД – Система управління базами даних
SPA – Односторінковий застосунок
SSR – Серверний рендеринг
SEO – Пошукова оптимізація
MVC – Архітектурний шаблон Model-View-Controller
ORM – Об'єктно-реляційне відображення
REST – Архітектурний стиль взаємодії вебсервісів
CRUD – Створення, читання, оновлення та видалення даних
KPI – Ключові показники ефективності
SMTP – Протокол передавання електронної пошти
SQL – Мова структурованих запитів до баз даних

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми кваліфікаційної роботи обумовлена цифровою відсталістю малого та середнього бізнесу в сфері послуг краси. Переважна більшість барбершопів чи перукарень використовують загальнодоступні та дешеві цифрові продукти для автоматизації деяких процесів, але для масштабування, конкурентоспроможності та підвищення ефективності їх роботи критично необхідна цифровізація всіх особливостей бізнес-процесів в цій сфері. Це напряду впливає на ефективність управління персоналом, зокрема в економії часу, зменшення людських зусиль та зосередженні співробітників на вирішенні найважливіших бізнес-задач. Пріоритетними напрямками для автоматизації в 2026 році є пошук та залучення нових клієнтів, і покращення користувацького досвіду (як клієнтів так і працівників). Досягнути цього результату можна зробивши пріоритетним напрямком локального SEO та розвитку в соціальних мережах, а також автоматизації більшості процесів, що дозволить бізнесу звільнити ресурси, необхідні для забезпечення максимальної конверсії та повернення клієнтів. Характерною рисою цієї сфери є наявність клієнтської бази у бізнесу, що дозволить мені як розробнику швидко випустити на ринок MVP продукту, і поступово його розвивати відповідно до потреб ринку та ключових стейкхолдерів. Така модель розвитку програмного продукту дуже ефективна в довгостроковій перспективі, за рахунок гнучкого масштабування, та в короткостроковій, тому що продукт навіть на ранніх стадіях розвитку вже буде приносити користь.

Метою випускної кваліфікаційної роботи бакалавра є розробка вебсистеми для автоматизації управління барбершопом, що забезпечить задоволення бізнес-потреб, підвищення ефективності організації роботи та оптимізацію процесів обслуговування клієнтів.

Для досягнення визначеної мети необхідно виконати наступні **завдання**:

- Провести аналіз області застосування та огляд існуючих на ринку конкурентів.
- Визначити функціонал продукту, та потенційні напрямки для розширення

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- Спроекувати архітектуру та засоби для розробки сайту.
- Спроекувати базу даних.
- Розробити модулі адміністрування, керування розкладом, контентом та аналітикою.
- Налаштувати систему CI/CD для зручності подальшого вдосконалення продукту
- Провести тестування, оцінити показники продуктивності та провести розгортання.

Об'єктом дослідження є дослідження інформаційних технологій управління бізнес-процесами перукарень.

Предметом дослідження є використання Web-орієнтованих технологій для розробки CMS перукарні з модулями онлайн-бронювання та аналітики..

Створений продукт може повноцінно використовуватися як офіційний сайт існуючого барбершопу. Сайт покриває весь ключовий функціонал існуючих конкурентів.

Апробація результатів роботи. За результатами досліджень у випускній кваліфікаційній роботі, було опубліковано тези: Колесник О. А., Піонтківський В. І. Розробка інформаційної системи для автоматизації бронювання управління та аналітики у сфері послуг. Комп'ютерні технології: інновації, проблеми, рішення : тези доповідей VIII Всеукр. наук.-техн. конф. (м. Житомир, 2–3 груд. 2025 р.).

Структура та обсяг проекту. Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 30 найменувань (3 стор.), 1 додатка (38 стор.), 13 таблиць та 32 ілюстрації по тексту. Повний обсяг роботи становить 126 сторінок, з них 87 сторінок основного тексту.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ВИЗНАЧЕННЯ АРХІТЕКТУРИ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Визначення та постановка задачі (проблематика)

Випускна кваліфікаційна робота бакалавра присвячена розробці веб-системи для управління барбершопом, яка розглядається як рішення для комплексних проблем автоматизації діяльності барбершопу. Основним призначенням системи є оптимізація процесу взаємодії між клієнтом і закладом, а також забезпечення адміністрації сучасними інструментами для ефективного управління бізнес-процесами. У сучасних умовах розвитку сфери послуг особливого значення набуває швидка та зручна комунікація з клієнтами, автоматизація внутрішніх операцій, зменшення впливу людського фактору та забезпечення прозорого господарсько-фінансового контролю. Саме тому впровадження подібних інформаційних систем є необхідною умовою підвищення конкурентоспроможності бізнесу [3].

На практиці значна частина барбершопів використовує комбінований підхід, за якого інформаційна складова представлена власним веб-сайтом або сторінками в соціальних мережах, а функція онлайн-запису реалізується через універсальні сторонні сервіси бронювання. Незважаючи на відносну простоту та дешевизну впровадження, такий підхід має низку суттєвих недоліків, які стають особливо помітними зі зростанням конкуренції на ринку.

Однією з основних проблем є дублювання процесів адміністрування. Будь-які зміни в переліку послуг, цінах, графіку роботи майстрів або контактній інформації часто необхідно вносити одночасно як на власному веб-ресурсі, так і в сторонній системі запису. Це збільшує витрати часу на підтримку інформації, підвищує ризик виникнення помилок та призводить до появи неактуальних або суперечливих даних на різних платформах.

Додатковим недоліком є обмежена масштабованість таких рішень. Універсальні сервіси бронювання орієнтовані на широкий спектр бізнесів і, як правило, не враховують специфіку роботи конкретного барбершопу. У результаті

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

виникають труднощі з реалізацією індивідуальних бізнес-процесів, інтеграцією додаткових модулів, персоналізацією користувацького досвіду та впровадженням нових функцій. Крім того, дані про клієнтів, записи та діяльність персоналу часто розподілені між декількома системами, що ускладнює їх аналіз та централізоване управління.

У зв'язку з цим актуальним є створення єдиної інтегрованої веб-системи, яка поєднує інформаційний веб-сайт, механізми онлайн-запису, інструменти управління персоналом та засоби адміністрування в межах одного програмного продукту. Такий підхід дозволяє усунути дублювання, забезпечити узгодженість даних, спростити підтримку системи та створити основу для її подальшого розвитку й масштабування в майбутньому.

Реалізована система повинна створити єдине цифрове середовище, яке дозволить автоматизувати повний цикл діяльності закладу – від залучення клієнтів через SEO-оптимізацію та інформаційний контент до організації запису на послуги, управління персоналом та ведення фінансового обліку. Використання сучасних підходів до розробки вебсистеми забезпечує високу швидкість роботи, доступність сервісу та зручність користування як для клієнтів, так і для персоналу. Завдяки впровадженню інтуїтивно зрозумілого інтерфейсу користувач отримає можливість швидко знайти необхідну інформацію, обрати майстра, послугу та зручний час для запису, без необхідності здійснення телефонних дзвінків або особистого відвідування закладу. Такий підхід значно підвищує рівень сервісу, зменшує навантаження на адміністраторів і сприяє збільшенню кількості записів.

Оскільки дана сфера бізнесу дуже конкурентна, особливу увагу варто приділити підвищенню впізнаваності бренду серед потенційних клієнтів та покращенню позицій сайту у пошукових видачах. Система передбачає використання гнучких інструментів для SEO-оптимізації, включаючи налаштування метаданих, структури сторінок та контенту, що дозволяє ефективно конкурувати у цифровому середовищі. Крім того, реалізація сучасного UI/UX-дизайну забезпечує привабливий зовнішній вигляд сайту та логічну навігацію, що

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

мінімізує час, необхідний користувачу для виконання основних дій, таких як бронювання послуг або перегляд інформації.

Функціональна структура системи базується на взаємодії декількох ключових модулів, кожен з яких орієнтований на конкретну групу користувачів. Клієнтська частина відповідає за забезпечення швидкого та зручного доступу до послуг, реалізуючи покроковий процес бронювання з можливістю вибору майстра, послуги та часу візиту. Адміністративна частина системи реалізується у вигляді CMS, яка надає широкі можливості для управління інформаційним наповненням сайту, редагування переліку послуг, прайс-листів, даних про майстрів, а також контролю за актуальністю контенту. Такий підхід дозволяє оперативно реагувати на зміни в діяльності закладу та підтримувати інформацію в актуальному стані.

Окрім управління контентом, адміністратор здійснює керування персоналом барбершопу. Система дозволяє створювати облікові записи співробітників, надавати їм доступ до внутрішніх функцій платформи відповідно до визначених ролей та контролювати їхню активність. Завдяки цьому кожен майстер отримує доступ до власного робочого кабінету, де може переглядати розклад записів, інформацію про клієнтів та керувати власним графіком роботи. Централізоване управління користувачами забезпечує належний рівень безпеки, спрощує адміністрування системи та дозволяє ефективно організовувати роботу персоналу в межах єдиного програмного середовища.

Окремий модуль хочу присвятити для управління діяльністю персоналу, зокрема барберів, які отримають можливість самостійно формувати власний робочий графік, визначати перелік послуг, а також вести персональні сторінки або блоги з портфоліо виконаних робіт. Це сприяє підвищенню мотивації працівників, розвитку їхнього професійного бренду та покращенню взаємодії з клієнтами. Водночас система забезпечує централізований контроль за розкладом, що дозволяє уникати накладок у записах та забезпечує вирішення адміністративних питань.

Ще однією функцією адміністративного модуля є контроль за розкладом роботи персоналу та всіма створеними записами клієнтів. Адміністратор має

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

доступ до загального календаря бронювань, що дозволяє відстежувати завантаженість майстрів, контролювати коректність сформованих графіків та своєчасно виявляти можливі конфлікти в розкладі. У разі виникнення непередбачених обставин, таких як відсутність майстра, зміна графіка роботи або технічні проблеми, адміністратор може оперативно втручатися в процес обслуговування, скасовувати або переносити записи клієнтів, а також інформувати їх про внесені зміни. Такий механізм забезпечує безперервність роботи закладу, підвищує якість сервісу та сприяє ефективному вирішенню організаційних питань.

Важливим елементом системи є ведення клієнтської бази, яка містить структуровану інформацію про користувачів, включаючи контактні дані, історію відвідувань та перелік отриманих послуг. Наявність такої інформації відкриває широкі можливості для персоналізації сервісу, покращення якості обслуговування та впровадження програм лояльності. Крім того, це дозволяє здійснювати аналіз поведінки клієнтів, визначати їхні потреби та адаптувати пропозиції відповідно до отриманих даних.

Система також включатиме аналітичний модуль, який забезпечить збір, обробку та візуалізацію ключових показників ефективності діяльності закладу. До таких показників належать кількість записів, завантаженість майстрів, популярність окремих послуг, фінансові показники результативності та динаміка розвитку бізнесу.

Аналіз цих даних дозволяє власникам та адміністраторам приймати обґрунтовані управлінські рішення, оптимізувати використання ресурсів та формувати стратегію подальшого розвитку бізнесу.

Фінансовий блок системи забезпечить автоматичний облік доходів і витрат, що дозволить формувати актуальну картину господарської діяльності закладу. Підтримка різних моделей співпраці з майстрами, зокрема відсоткової та орендної, робить систему універсальним інструментом для різних форматів бізнесу. Це сприяє підвищенню прозорості фінансових операцій і спрощує процес контролю бюджетом.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Додатково планую передбачити системну реалізацію механізмів сповіщення, які інформують клієнтів та персонал про нові записи, зміни у графіку або майбутні візити за допомогою SMS або сучасних месенджерів. Це дозволяє зменшити кількість пропущених записів, підвищити дисципліну клієнтів і забезпечити більш ефективну організацію роботи закладу. Впровадження таких функцій позитивно вплине на загальний рівень сервісу та задоволеність користувачів.

Процес розробки системи передбачає поетапну реалізацію всіх компонентів, починаючи з формування мети, цілей та функціональних вимог для основних груп користувачів, серед яких клієнти, адміністратори та майстри. Наступним етапом є проектування структури бази даних, яка описує всі ключові сутності системи, включаючи послуги, розклади, фінансові транзакції та контент. Після цього здійснюється розробка архітектури клієнтської та серверної частин із урахуванням вимог до продуктивності, безпеки та масштабованості.

Особлива увага приділяється реалізації клієнтської частини системи з фокусом на швидкість завантаження сторінок та відповідність принципам Mobile First, що забезпечує коректну роботу на різних пристроях. Серверна частина включає механізми безпечної автентифікації, обробки запитів та взаємодії з базою даних. Після завершення етапу розробки здійснюється інтеграція основних модулів системи, включаючи онлайн-запис, CMS, аналітичний блок та систему сповіщень.

Завершальним етапом є комплексне тестування системи, яке включає перевірку функціональності, кросбраузерності, стабільності роботи під навантаженням та відповідності сучасним стандартам веброботки. Це дозволяє виявити можливі помилки, забезпечити коректну роботу всіх компонентів і підготувати систему до впровадження в реальних умовах експлуатації.

У результаті виконання роботи буде створено цілісну, надійну та масштабовану інформаційну систему, яка дозволить автоматизувати ключові бізнес-процеси барбершопу, підвищити ефективність управління та покращити якість обслуговування клієнтів. Використання розробленої платформи забезпечить зручний інструмент для формування гнучких графіків роботи персоналу,

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

оперативного управління контентом, отримання актуальної аналітичної інформації та організації процесу запису на послуги. У підсумку це сприятиме стабільному розвитку бізнесу, підвищенню його конкурентоспроможності та зменшенню витрат на адміністрування без необхідності кардинальної зміни програмного забезпечення в майбутньому.

1.2. Аналіз аналогів програмного продукту

Оскільки ніша розроблюваного продукту передбачає пряму конкуренцію на ринку послуг, важливим етапом дослідження є аналіз існуючих аналогів та потенційних конкурентів. Такий підхід до аналізу дозволяє визначити сильні та слабкі сторони вже реалізованих рішень, виявити ключові функціональні можливості, які користуються попитом серед користувачів, а також сформуванню уявлення про рівень цифровізації подібних бізнесів у регіоні. Аналіз конкурентів також дає змогу оцінити підходи до організації онлайн-сервісів, взаємодії з клієнтами та управління внутрішніми процесами закладу. Особливий інтерес становлять механізми запису на послуги, представлення інформації про майстрів та способи залучення нових клієнтів.

Особливу увагу доцільно приділити локальним лідерам ринку, оскільки саме вони формують основні стандарти якості обслуговування та рівень сервісу що задовільняє очікування клієнтів, а результати такого аналізу слугують основою для формування унікальної торгової пропозиції для системи та дозволяють обґрунтовано визначити напрями її подальшого функціонального розвитку.

В якості першого прикладу було обрано барбершоп «КОД», який є одним із найвідоміших закладів у місті Житомир та має декілька локацій. Заклад позиціонує себе як сучасний простір, що поєднує класичні традиції барберингу з актуальними трендами у сфері стилю та догляду за обличчям. Барбершоп надає широкий спектр послуг, включаючи чоловічі стрижки, оформлення бороди, укладку волосся та інші супутні процедури, орієнтуючись на створення індивідуального образу для кожного клієнта. Важливою особливістю є також наявність декількох філій у місті, що свідчить про успішність бізнес-моделі та стабільний попит на послуги (рис. 1.1).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

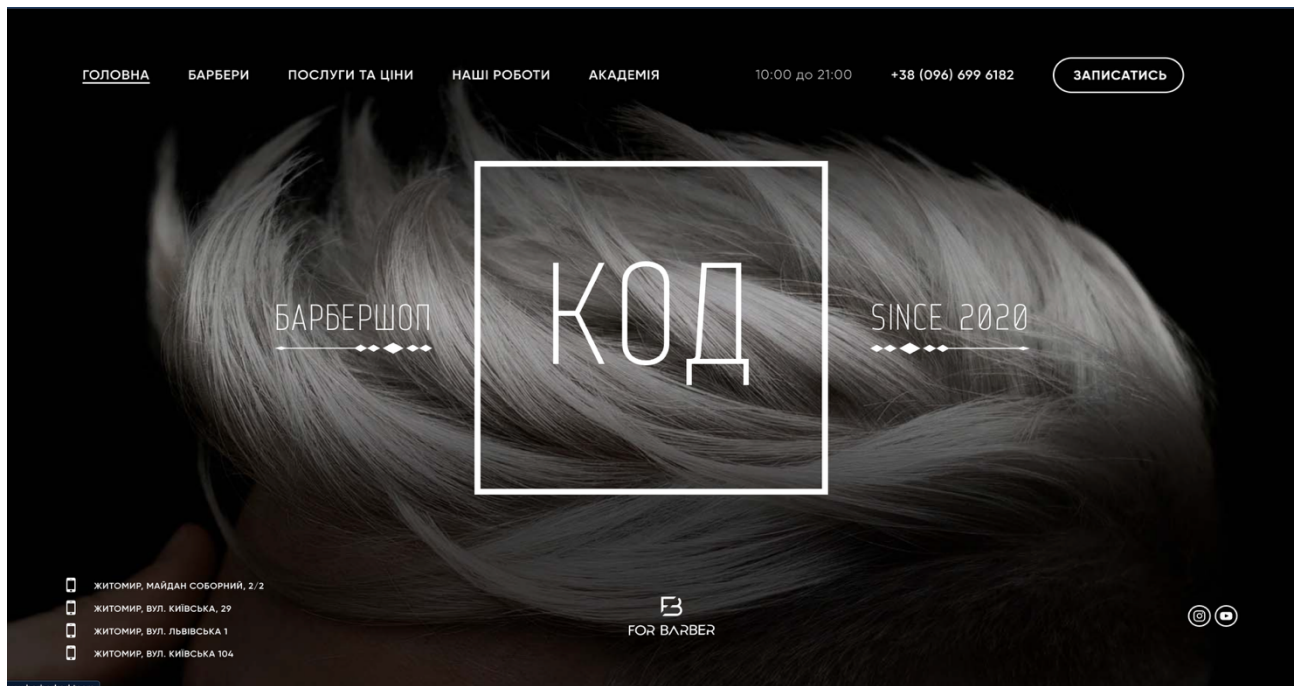


Рис. 1.1. Головна сторінка сайту «КОД»

Оскільки мова йде про мережу барбершопів, на сайті передбачено відображення інформації про всі доступні локації, зокрема в нижній частині сторінки розміщено адреси кожного закладу. Це дозволяє користувачу швидко знайти найближчу точку та отримати базову інформацію для відвідування.

Окрім цього, сайт містить окремі сторінки, де можна детальніше ознайомитися з майстрами, переглянути перелік послуг і ціни, а також оцінити галерею виконаних робіт. Така структура є логічною та зрозумілою для користувача, оскільки охоплює основні запити потенційного клієнта перед записом.

Загалом сайт виглядає доволі не складним у реалізації, що особливо помітно через перенесення функціоналу онлайн-запису на сторонній сервіс EasyWeek, який відкривається на іншому домені. Це рішення спрощує технічну реалізацію, однак водночас розриває цілісність користувацького досвіду, оскільки клієнт змушений переходити між різними ресурсами для завершення дії.

Серед переваг варто відзначити сучасний зовнішній вигляд сайту, який відповідає актуальним тенденціям сучасного дизайну, а також загальну зрозумілість і простоту інтерфейсу. Навігація є інтуїтивною, а структура сторінок

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє швидко знайти потрібну інформацію без зайвих дій, що позитивно впливає на перше враження користувача (рис.1.2).

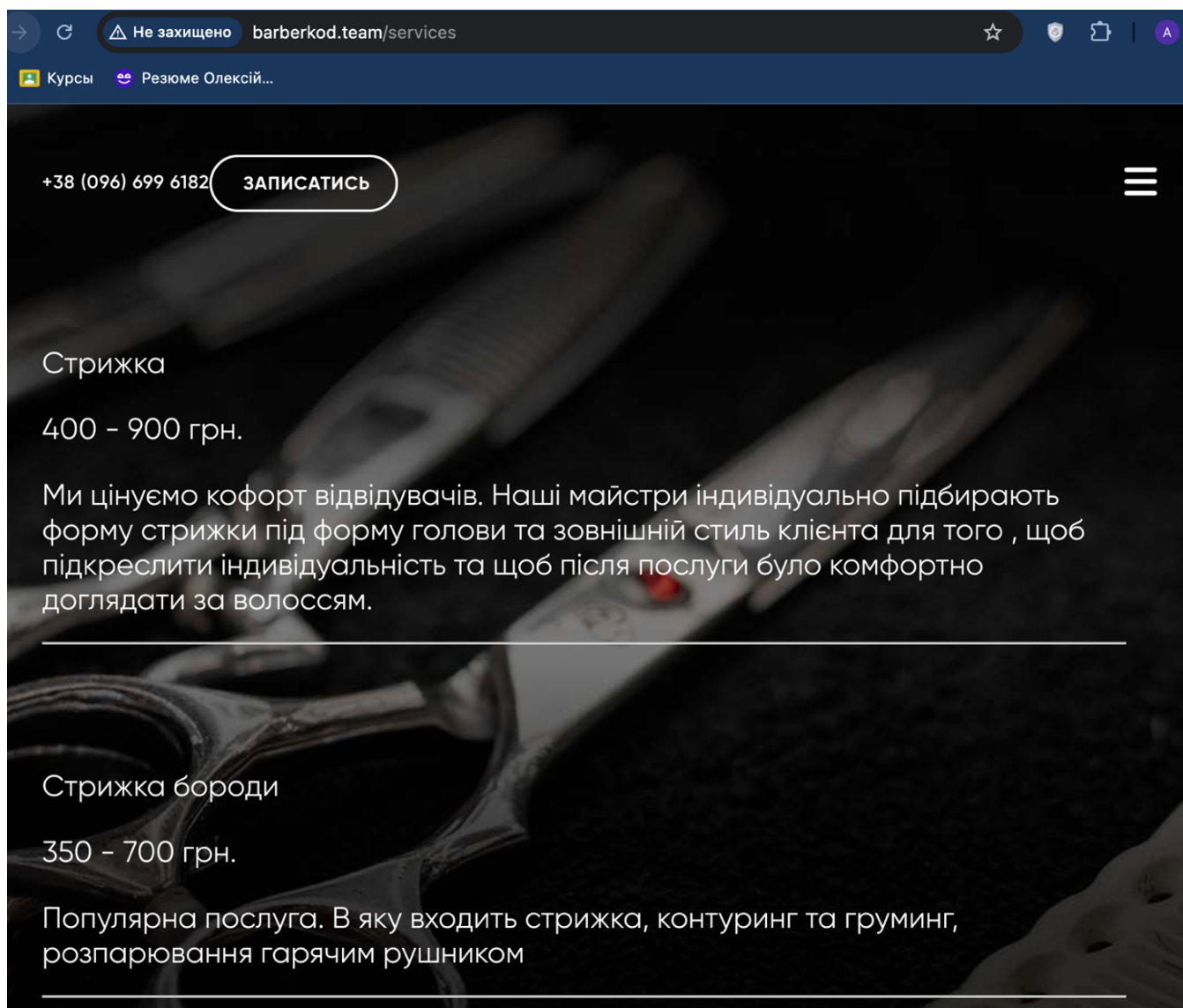


Рис. 1.2. Сторінка з цінами сайту “КОД”

Разом із тим, сайт має ряд недоліків, які впливають як на зручність використання, так і на його ефективність у залученні нових клієнтів. Зокрема, він не найкращим чином адаптований під пошукову видачу, що може негативно позначатися на його видимості у результатах пошуку. Відсутність HTTPS-протоколу також є суттєвим недоліком, оскільки це впливає на рівень довіри користувачів і може викликати застереження при взаємодії з сайтом. Крім того, контентна частина ресурсу є досить обмеженою та не повністю задовольняє інформаційні потреби нових відвідувачів.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Окремо варто зазначити, що сайт не має повноцінної адміністративної частини, що значно ускладнює процес його підтримки та оновлення контенту. Уся інформація, представлена на сайті, редагується безпосередньо на сервері, що вимагає технічних навичок і додаткового часу. Такий підхід є незручним для повсякденного використання, особливо в умовах, коли дані можуть часто змінюватися, наприклад, ціни, графік роботи майстрів або перелік послуг.

Узагальнюючи, можна сказати, що сайт барбершопу виконує роль своєрідної онлайн-візитівки або портфоліо, яке містить базову інформацію про заклад і перенаправляє користувача на сторонній ресурс для здійснення запису. Такий підхід має свої обмеження, зокрема змушує працівників дублювати інформацію про майстрів, послуги та ціни на двох різних платформах. Це не лише ускладнює процес адміністрування, але й створює ризик виникнення неузгодженостей у даних, що вказує на необхідність подальшої автоматизації та інтеграції функціоналу в межах єдиної системи.

Наявні обмеження підтверджують доцільність переходу до комплексної веб-системи, яка об'єднує інформаційний сайт, модуль онлайн-запису та інструменти адміністрування в межах єдиного середовища. Це дозволяє забезпечити узгодженість даних, зменшити кількість ручних операцій та підвищити ефективність управління бізнес-процесами, що є критично важливим для сучасних закладів сфери послуг.

Наступним етапом доцільно більш детально розглянути безпосередньо сервіс EasyWeek, оскільки він виступає одним із ключових і найбільш поширених аналогів для реалізації онлайн-запису на послуги. Даний сервіс належить до універсальних рішень для бронювання та управління записами клієнтів у сфері послуг і активно використовується малим бізнесом. Він надає базовий набір інструментів для створення розкладу, керування доступністю майстрів. Саме цей інструмент широко використовується барбершопами як готове рішення для організації бронювання, що дозволяє швидко та відносно дешево впровадити базовий функціонал без необхідності розробки власної системи (рис.1.3).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

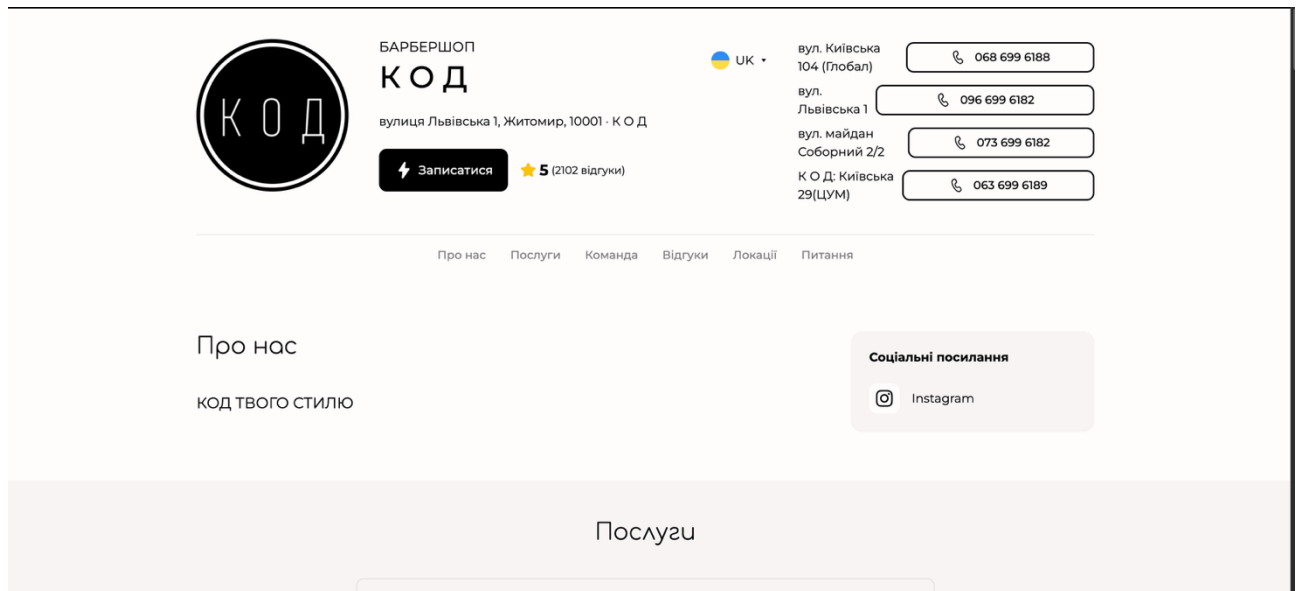


Рис. 1.3. Сторінка сайту “КОД” на сервісі easyweek

Даний сервіс побудований за принципом універсальності та надає можливість розміщення всієї необхідної інформації про заклад в межах однієї сторінки. Користувач отримує доступ до базових даних, включаючи перелік послуг, доступні часові слоти, інформацію про майстрів, а також відгуки, оцінки та коментарі інших клієнтів. Проте, незважаючи на наявність цих елементів, їх представлення не завжди є зручним для сприйняття, що може ускладнювати процес ознайомлення з інформацією.

Інтерфейс системи є стандартним і уніфікованим для всіх типів бізнесів, які користуються даним сервісом. Саме через це він не виглядає достатньо сучасним або стилістично адаптованим під тематику барбершопу, що може впливати на загальне враження користувача. Відсутність індивідуального дизайну та брендингу обмежує можливості створення унікального користувацького досвіду та знижує рівень впізнаваності конкретного закладу.

Форма запису на послуги реалізована у вигляді стандартного, але логічно побудованого процесу (рис.1.4). Користувачу спочатку пропонується обрати зручний час або конкретного майстра, після чого відбувається вибір послуги. Така послідовність є зрозумілою та не викликає труднощів у використанні навіть для нових користувачів. Додатково система підтримує механізм сповіщень через SMS,

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

що дозволяє інформувати клієнтів про підтвердження запису або нагадування про візит.

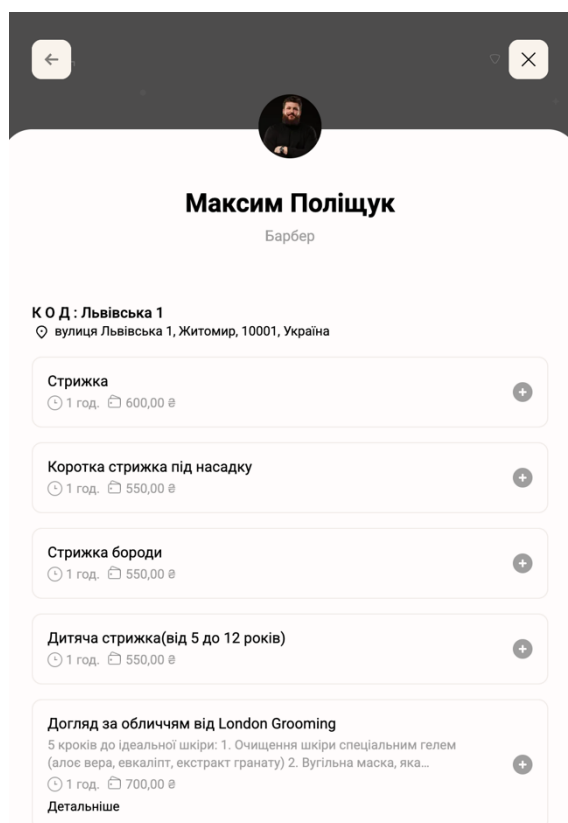


Рис. 1.4. Форма запису на платформі easyWeek

Водночас важливо відзначити і переваги даного рішення. Однією з ключових є відносно невисока вартість використання, що робить сервіс доступним для малого бізнесу. Завдяки своїй універсальності він може бути швидко впроваджений без значних витрат часу та ресурсів, що є особливо доцільним для закладів, які вже мають сформовану клієнтську базу і потребують лише базового інструменту для організації запису.

Проте в умовах високої конкуренції та зростаючих вимог до якості цифрового сервісу подібні універсальні рішення можуть поступатися більш гнучким і кастомним системам. Обмежені можливості кастомізації, відсутність унікального користувацького досвіду та складність інтеграції з іншими бізнес-процесами знижують їхню ефективність у довгостроковій перспективі.

У результаті компанії, що використовують індивідуально розроблені рішення, отримують перевагу за рахунок кращої адаптації під власні потреби,

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

покращеної взаємодії з клієнтами та більш глибокої автоматизації внутрішніх процесів.

Наступним об'єктом для порівняння я обрав барбершоп Gentlmen's club. Це велика всеукраїнська мережа, всього по Україні в них налічується 34 заклади, і з технічної точки зору, централізовано контролювати таку кількість філій, майстрів та клієнтських запитів за допомогою єдиної платформи – це досить нетривіальна задача. Саме тому для мого дослідження буде особливо цікаво проаналізувати архітектуру та функціонал цього сайту.

Варто відмітити візуальну складову: на головній сторінці активно використовуються фонові відеоматеріали. Це вдале маркетингове рішення, яке з перших секунд ефективно передає автентичну атмосферу закладу та занурює клієнта у відповідний настрій. Проте такий підхід має один суттєвий технічний недолік – він робить завантаження сторінки значно довшим, що може негативно вплинути на користувацький досвід, особливо на мобільних пристроях (рис. 1.5).

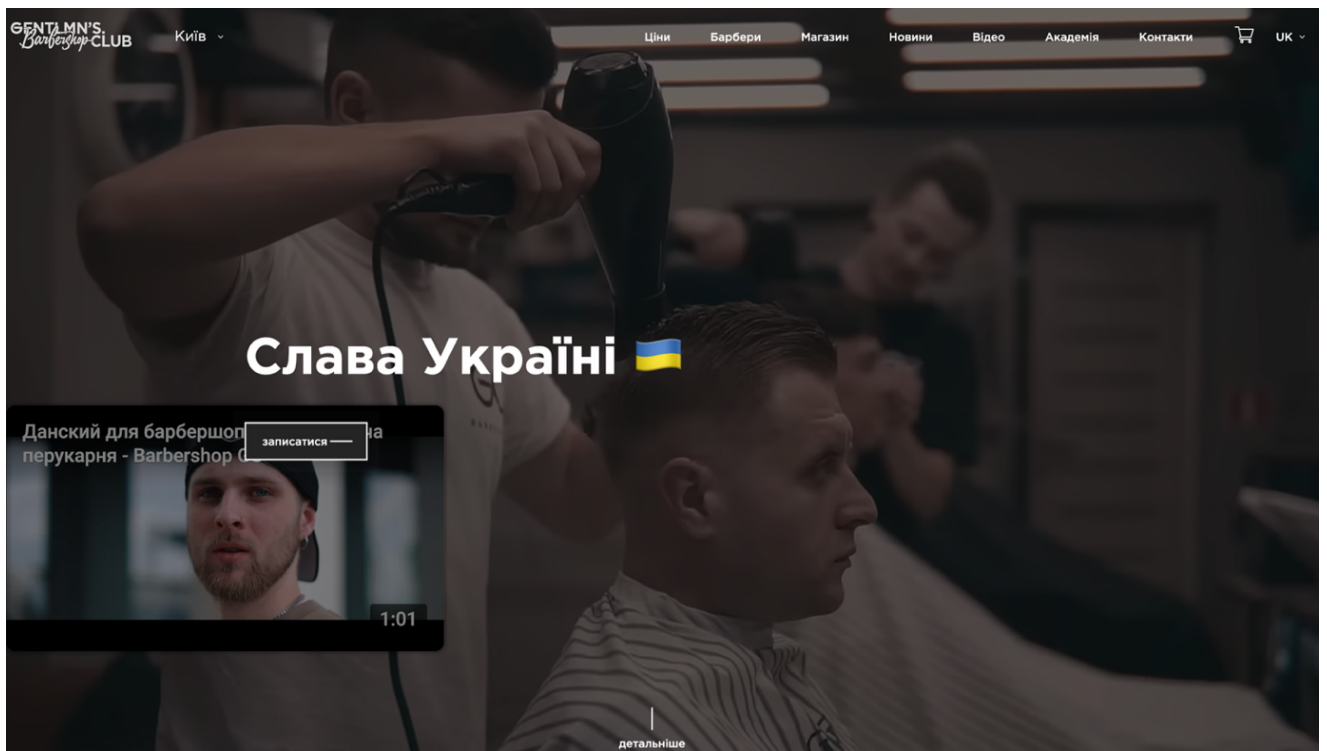
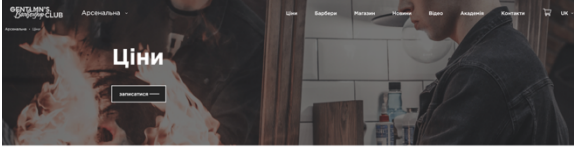


Рис. 1.5. Головна сторінка сайту Gentlmen's club

На сторінці ціни за послуги розбиті по рівню досвіду кожного барбера (рис. 1.6). Це дає клієнту краще розуміння цінової політики барбершопу, але не дає

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

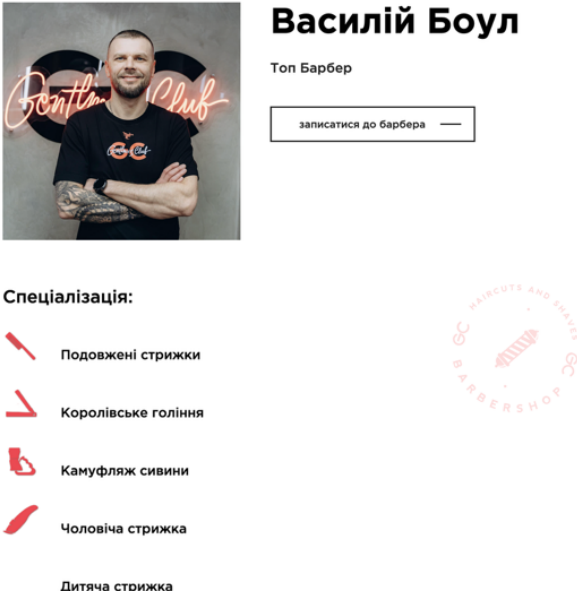
гнучкості самим майстрам самотійно визначати вартість своїх послуг. Сама таблиця реалізована невдалим чином, вона має обмежені рамки, що не дозволяє бачити всю інформацію повністю, і доводиться використовувати скролл.



	Барбер	Старший Барбер	Топ Барбер	Шеф Барбер
а	650	800	900	1000
і нус (без	350	500	450	600
і нус (з	450	500	600	700
шугу (до 3 мм)	350	450	550	600
шугу (фейд)	450	550	700	800
а і стрижка	1100	1300	1500	1700
шуга	450	500	550	600
(5-12 років)	600	750	850	950
машин	600	650	700	750

Рис. 1.6. Сторінка з цінами

Окрім загальної інформації, на сайті реалізовано окремий розділ, присвячений команді майстрів, з можливістю переходу на персональну сторінку кожного барбера (рис. 1.7). Такий підхід є цікавим рішенням з точки зору формування довіри до бренду, клієнти мають змогу заздалегідь познайомитися з персоналом, дізнатися про їхній стаж, спеціалізацію та відчутти атмосферу закладу. Це робить сервіс більш відкритим та персоналізованим.



Василь Боул
Топ Барбер

записатися до барбера

Спеціалізація:

- Подовжені стрижки
- Королівське гоління
- Камуфляж сивини
- Чоловіча стрижка
- Дитяча стрижка

Рис. 1.7. Сторінка з цінами

Але відсутність галереї виконаних стрижок чи прикладів оформлення бороди є, на мою думку, слабким місцем у маркетинговій стратегії. Це змушує потенційного клієнта залишати сайт і шукати цю інформацію на сторонніх ресурсах.

На основі проведеного аналізу можна створити порівняльну таблицю, щоб об'єктивно оцінити всі переваги та недоліки існуючих сайтів (табл. 1.1).

Таблиця 1.1.

Порівняння аналогів

Критерій оцінки	КОД	Gentlmens club	easyWeek
Власний сайт	так	так	ні
Онлайн запис	ні	ні	так
Інформація про барберів	так	так	так
Окремі сторінки майстрів	ні	так	ні
Панель адміністратора	ні	так	так
Можливість керування контентом	ні	так	ні
Галерея робіт	так	ні	ні
Seo оптимізація	висока	середня	низька
Єдина система адміністрування	ні	ні	так
Гнучке керування цінами	ні	ні	так

Детальний аналіз вебсайтів ключових конкурентів на ринку (як локальних лідерів, зокрема барбершопу «КОД», так і великих мережових представників) дозволив визначити основні технічні та інтерфейсні вимоги до сучасного вебпродукту у сфері послуг. Дослідження показало, що більшість сайтів виконують роль виключно інформаційних візиток, де ключова функція це онлайн-запис реалізована через сторонні скрипти та віджети. Це призводить до зниження швидкості завантаження сторінок, порушення цілісності дизайну та залежності від працездатності сторонніх сервісів.

Крім того, суттєвою проблемою існуючих рішень є відсутність зручної системи керування контентом. Будь-які зміни у вартості посліг, списку персоналу чи графіку роботи потребують прямого втручання безпосередньо в код сайту, що ускладнює адміністрування для кінцевого власника бізнесу.

Таким чином, розробка вебсайту на базі власної спеціалізованої CMS є повністю обґрунтованою. Це дозволить створити єдину, технічно оптимізовану екосистему, де інтерфейс користувача та панель адміністратора працюють як одне ціле. Продукт забезпечить високу швидкість відгуку сторінок, безшовну інтеграцію модуля бронювання без сторонніх сервісів, а також надасть власнику закладу зручний інструментарій для автономного керування всіма бізнес-процесами та контентом сайту.

1.3. Визначення архітектури програмного продукту

Під час проєктування CMS для перукарні з модулями онлайн-бронювання та аналітики ключовим етапом є вибір архітектури програмної системи. Від обраного архітектурного підходу залежать масштабованість, продуктивність, зручність розробки, а також можливість подальшого розвитку програмного продукту без кардинальних змін в його структурі [18].

З огляду на поставлені завдання та специфіку предметної області, для реалізації системи обрано монолітну архітектуру. Такий підхід передбачає побудову єдиного програмного застосунку, в якому всі основні функціональні модулі (клієнтська частина, адміністративна панель, бізнес-логіка та робота з базою даних) інтегровані в одному середовищі. Монолітна архітектура є доцільним вибором для малого та середнього бізнесу, оскільки забезпечує простоту розгортання, зменшує складність підтримки та не потребує значних ресурсів на інфраструктуру. Додатковою перевагою такого підходу є централізоване керування кодовою базою, що спрощує тестування, відлагодження та подальше розширення функціоналу системи.

Особливістю реалізації системи є розмежування підходів до клієнтської та адміністративної частин при збереженні єдиної монолітної структури [10].

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнтська сторона сайту (публічна частина) реалізується у вигляді статичних сторінок, що формуються на сервері. Такий підхід забезпечує високу швидкість завантаження, коректну індексацію пошуковими системами та ефективну SEO-оптимізацію, що є критично важливим для залучення нових клієнтів у конкурентному середовищі сфери послуг. У контексті сучасних веб-технологій подібна реалізація може бути побудована на основі підходу server-side rendering або гібридного рендерингу, де сервер формує початковий HTML, а подальша взаємодія відбувається через JavaScript.

Водночас адміністративна частина системи орієнтована на внутрішнє використання персоналом та власником бізнесу, тому для її реалізації передбачено використання JavaScript-фреймворків [28]. Це дозволяє створити динамічний інтерфейс із миттєвим оновленням даних, зручними формами керування розкладом, контентом та аналітичними показниками без перезавантаження сторінок. У практичній реалізації подібний підхід часто базується на використанні Vue.js як реактивного фронтенд-фреймворку, який забезпечує компонентну структуру інтерфейсу та зручну роботу зі станом застосунку.

Додатково у зв'язці з Laravel може використовуватися Inertia.js, що дозволяє будувати SPA-подібний інтерфейс без необхідності створення окремого API. У такій архітектурі Laravel виконує роль серверної частини в рамках MVC-патерну, обробляючи запити, керуючи бізнес-логікою та повертаючи сторінки через Inertia, тоді як Vue.js відповідає за відображення та інтерактивність інтерфейсу. Це забезпечує більш тісну інтеграцію фронтенду і бекенду та спрощує розробку порівняно з повністю розділеними системами.

Для зберігання та обробки даних у системі використовуватиметься SQL-база даних, яка забезпечує цілісність, надійність та структурованість інформації [1].

Реляційна модель даних є оптимальною для зберігання інформації про клієнтів, записи на послуги, розклад майстрів, фінансові операції та аналітичні показники. Використання SQL дозволяє реалізувати складні запити для формування звітів і бізнес-аналітики, що є важливою складовою функціональності

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

системи [5]. Також застосування ORM-рівня в межах серверної частини дозволяє абстрагувати роботу з базою даних та підвищити зручність розробки, зменшуючи кількість ручних SQL-запитів.

Архітектура взаємодії обраного технологічного стеку наведена нижче (рис. 1.8). Вона відображає, як рухаються дані між серверною та клієнтською частинами системи, а також розподіл відповідальності між ними. Однією з головних переваг такого підходу є те, що немає потреби створювати та підтримувати окремий REST API.

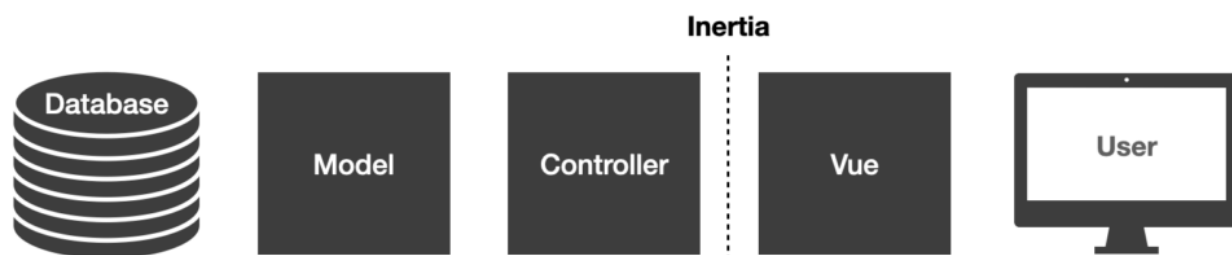


Рис. 1.8. Архітектура взаємодії системи

Основним рівнем зберігання даних виступає база даних. Робота з нею здійснюється через моделі фреймворку Laravel, які забезпечують зручний доступ до даних і приховують складність взаємодії з БД. За обробку бізнес-логіки відповідають контролери. Коли надходить запит, контролер звертається до відповідних моделей, виконує необхідні операції, наприклад фільтрацію даних або обчислення показників, і формує результат для подальшої передачі на клієнтську частину.

Окрему роль у цій архітектурі відіграє прошарок Inertia, який на схемі позначений як розділювальна пунктирна лінія. Замість класичного підходу, де сервер або генерує html, або повертає JSON для окремого фронтенду, тут контролери Laravel через Inertia передають назву потрібного фронтенд-компонента разом із даними, які йому необхідні.

На стороні клієнта ці дані приймає Vue і передає їх у вигляді props у компоненти. Фронтенд у цьому випадку займається переважно відображенням інтерфейсу та обробкою взаємодій користувача, не беручи на себе складної бізнес-

логіки. Це дозволяє чітко розділити відповідальності між частинами системи та спростити її подальший розвиток.

Користувач працює з системою як із SPA-додатком, перехід по сторінкам відбувається швидко, без повного перезавантаження, а відчуття взаємодії залишається плавним і приємним. При цьому маршрутизація залишається на боці Laravel, що дозволяє зберегти монолітну структуру застосунку, централізовано контролювати доступи та логіку переходів, а також спрощує супровід і тестування системи.

Таким чином, обрана архітектура поєднує переваги монолітного підходу та сучасного JavaScript-фронтенду. Вона дозволяє мати єдину серверну основу з повним контролем над даними, водночас забезпечуючи динамічний та інтерактивний інтерфейс для користувача. Окремо варто відзначити, що такий підхід добре підходить для SEO-орієнтованих сторінок і водночас дозволяє будувати складні адміністративні інтерфейси на основі Vue. У результаті система отримує баланс між продуктивністю, зручністю використання та можливістю масштабування відповідно до зростання функціональних вимог.

1.4. Обґрунтування вибору засобів реалізації та вимоги до апаратного забезпечення

Наступним етапом розробки інформаційної системи є обґрунтований вибір програмних засобів, які забезпечать стабільну роботу, високу продуктивність, можливість масштабування та зручність супроводу в подальшому. Обраний технологічний стек має відповідати сучасним стандартам веб-розробки, а також бути раціональним з точки зору впровадження у бізнесах малого та середнього рівня.

Для реалізації серверної частини системи було обрано фреймворк Laravel версії 12, який є одним із провідних інструментів у середовищі PHP-розробки. Даний фреймворк надає структурований підхід до створення застосунків, базується на архітектурному шаблоні MVC та містить широкий набір вбудованих можливостей для роботи з базами даних, управління користувачами, а також

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізації механізмів автентифікації та авторизації. Крім того, Laravel забезпечує належний рівень безпеки та спрощує реалізацію складної бізнес-логіки, що є важливим при створенні системи управління записами, клієнтами та аналітичними даними [14].

Для організації взаємодії між серверною логікою та клієнтською частиною адміністративного інтерфейсу використовується Inertia.js. Такий підхід дозволяє поєднати класичну модель серверного рендерингу з можливостями сучасних фронтенд-технологій, не створюючи при цьому окремого API-рівня. Завдяки цьому забезпечується більш проста архітектура застосунку та зменшується складність розробки. Inertia.js дозволяє реалізувати плавну навігацію між сторінками та створити інтерфейс, який за своєю поведінкою наближений до односторінкових застосунків [12].

Для побудови динамічної адміністративної частини обрано фреймворк Vue.js, який добре підходить для створення інтерактивних інтерфейсів. Він забезпечує реактивність даних, що дозволяє миттєво відображати зміни без перезавантаження сторінок. Це особливо важливо для адміністративної панелі, де користувачі активно працюють із формами, таблицями, розкладом та аналітичними показниками. Використання Vue.js підвищує зручність роботи персоналу та сприяє більш ефективному управлінню системою [29].

Для оформлення зовнішнього вигляду інтерфейсу застосовується Tailwind CSS – сучасний CSS-фреймворк, який дозволяє швидко створювати адаптивні та естетично привабливі інтерфейси. Його використання зменшує потребу у написанні великої кількості власних стилів, забезпечує єдність дизайну та спрощує підтримку візуальної частини системи. Крім того, Tailwind CSS добре підходить для створення інтерфейсів, які коректно відображаються на різних типах пристроїв включаючи смартфони та планшети [24].

Публічна частина вебсайту, орієнтована на клієнтів, реалізується за допомогою шаблонізатора Blade. Такий підхід дозволяє генерувати сторінки на сервері, що позитивно впливає на швидкість їх завантаження та забезпечує

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

коректну індексацію пошуковими системами. Відмова від використання важких JavaScript-рішень у клієнтській частині сприяє покращенню продуктивності та загального користувацького досвіду [17].

Для зберігання даних використовується реляційна система керування базами даних PostgreSQL, яка відзначається високою надійністю, підтримкою складних запитів та транзакційністю. Вона дозволяє ефективно працювати зі структурованими даними, такими як інформація про клієнтів, записи на послуги, графіки роботи майстрів, фінансові операції та результати аналітики. Це робить її оптимальним вибором для реалізації подібної системи [13].

За результатами порівняльного аналізу було визначено та обрано технологічний стек, який найкраще відповідає вимогам щодо продуктивності, простоти розробки та подальшого супроводу системи. Обрані рішення забезпечують узгоджену роботу всіх компонентів системи та спрощують її масштабування в майбутньому, що надзвичайно важливо в сучасних умовах ринку. Короткий опис кожної використаної технології та її роль у проєкті подано узагальнено в таблиці 1.2.

Таблиця 1.2.

Стек програми

Технологія	Призначення
Laravel 12	Серверний фреймворк
Php 8.4	Серверна мова програмування.
Inertia.js	Організація взаємодії між серверною та клієнтською частинами
Vue.js	Фреймворк для побудови динамічної адміністративної панелі
Tailwind CSS	CSS-фреймворк для оформлення зовнішнього вигляду інтерфейсу
Blade	Шаблонізатор для публічної частини вебсайту (клієнтської)
PostgreSQL	Реляційна СУБД
TypeScript	Мова програмування
Git	Система контролю версій
Vite	Інструмент збірки
Docker	Платформа контейнеризації для ізольованого запуску СУБД

Отже, поєднання технологій Laravel, Inertia.js, Vue.js, Tailwind CSS та PostgreSQL формує збалансований стек, який дозволяє створити сучасну, продуктивну та зручну в експлуатації систему. Такий підхід забезпечує не лише ефективну реалізацію поточного функціоналу, але й створює основу для подальшого розвитку та масштабування програмного продукту відповідно до потреб бізнесу.

На основі обраного технологічного стеку можна сформулювати вимоги до апаратного забезпечення, які забезпечать стабільну та продуктивну роботу системи. Оскільки основна обробка даних виконується на серверній частині, саме сервер є ключовим елементом, що визначає загальну продуктивність рішення.

Сервер повинен працювати під керуванням Linux-дистрибутива Ubuntu Server версії не нижче 22.04 LTS. Використання саме LTS-версії є обов'язковим, оскільки вона забезпечує довгострокову підтримку, стабільність оновлень та сумісність із серверним програмним забезпеченням, що використовується у системі. На сервері має бути попередньо встановлене та налаштоване середовище виконання PHP, вебсервер Nginx та система керування базою даних.

Додатковою вимогою є наявність Composer останньої стабільної версії. Composer використовується як основний інструмент керування залежностями PHP-застосунку, забезпечує встановлення бібліотек, їх оновлення та контроль версій. Його відсутність унеможлиблює коректне розгортання та підтримку серверної частини системи, тому він повинен бути встановлений на етапі підготовки серверного середовища.

Для серверної частини доцільно використовувати багатоядерний процесор рівня не менше 2-4 фізичних ядер із частотою від 2.4 ГГц. Така конфігурація дозволяє ефективно обробляти одночасні HTTP-запити, виконання бізнес-логіки та операції з базою даних без суттєвих затримок.

У випадку зростання навантаження система може масштабуватися шляхом збільшення кількості ядер або переходу на більш продуктивну серверну інфраструктуру.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

Оперативна пам'ять є критичним ресурсом для стабільної роботи вебсервера, PHP-FPM процесів та системи керування базою даних. Рекомендований мінімальний обсяг становить 8 ГБ, однак для більш комфортної роботи та можливості паралельної обробки запитів доцільно використовувати конфігурацію на рівні 16 ГБ. Це забезпечує відсутність перевантаження пам'яті при пікових навантаженнях і стабільну роботу кешування.

Дискова система повинна базуватися на накопичувачі типу SSD, оскільки саме швидкість читання та запису безпосередньо впливає на продуктивність роботи бази даних і файлової системи застосунку. Рекомендований мінімальний обсяг становить 100-200 GB із використанням, що забезпечують значно нижчі затримки порівняно з іншими рішеннями.

Рекомендована пропускна здатність мережевого каналу становить не менше 100 Мбіт/с із низьким рівнем затримок, що надасть швидку взаємодію користувачів із системою та стабільну роботу http запитів. Клієнтська частина системи не висуває високих вимог до апаратного забезпечення, оскільки виконується у сучасному веббраузері. Достатньо пристрою з 2–4 GB оперативної пам'яті та будь-якого сучасного процесора, здатного коректно відображати JavaScript-інтерфейси, а також стабільного підключення до мережі Інтернет.

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз предметної області та обґрунтовано основні підходи до розробки програмної системи для управління барбершопом. Зокрема, розглянуто актуальність створення подібного рішення в умовах сучасного ринку послуг, де важливими є автоматизація бізнес-процесів, зручність взаємодії з клієнтами та ефективне управління ресурсами закладу.

У процесі дослідження було проаналізовано існуючі аналоги, зокрема локальні вебсайти барбершопів та використання сторонніх сервісів онлайн-запису. Встановлено, що більшість рішень обмежуються функціональністю сайту-візитівки у поєднанні з універсальними платформами бронювання, що не забезпечують достатнього рівня гнучкості, інтеграції та автоматизації. Виявлені

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

недоліки, такі як відсутність повноцінної адміністративної частини, дублювання даних, обмежені можливості аналітики та слабка адаптація під SEO, підтверджують доцільність розробки власної комплексної системи.

Також у розділі було визначено ключові вимоги до майбутнього програмного продукту, включаючи реалізацію зручного онлайн-запису, управління розкладом, ведення клієнтської бази, фінансового обліку та аналітики. Окрему увагу приділено підвищенню якості користувацького досвіду, забезпеченню сучасного інтерфейсу та оптимізації системи для пошукових систем.

На основі проведеного аналізу було обрано монолітну архітектуру як найбільш доцільну для реалізації системи з урахуванням її масштабів та цільової аудиторії. Обґрунтовано використання підходу з розділенням публічної та адміністративної частин, де клієнтська частина реалізується з урахуванням SEO-вимог, а адміністративна – як динамічний інтерфейс для зручного управління.

Крім того, було визначено технологічний стек, який включає використання сучасних інструментів веб-розробки, що забезпечують ефективну реалізацію системи, її продуктивність, надійність та можливість подальшого масштабування. Обрані технології дозволяють створити цілісне рішення, яке поєднує зручність використання, функціональність та відповідність сучасним стандартам.

Таким чином, у межах розділу було сформовано теоретичну та технологічну основу для подальшої розробки системи, визначено її ключові особливості, переваги над існуючими аналогами та сформовано напрямки практичної реалізації.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1. Визначення варіантів використання та вимог системи

Під час проєктування важливим кроком є розуміння варіантів використання системи, та основних сценаріїв взаємодії користувацьких ролей з програмним продуктом.

Визначення та аналіз варіантів використання дозволить детально розглянути логіку роботи майбутнього продукту, чітко визначити функціональні вимоги, та розібратись в розумінні процесів які будуть реалізовані в системі (рис. 2.1). Крім того, такий підхід допомагає виявити потенційні прогалини в архітектурі та уникнути помилок ще на етапі планування. Це в свою чергу сприятиме ефективній реалізації всіх функціональних можливостей сайту [7].

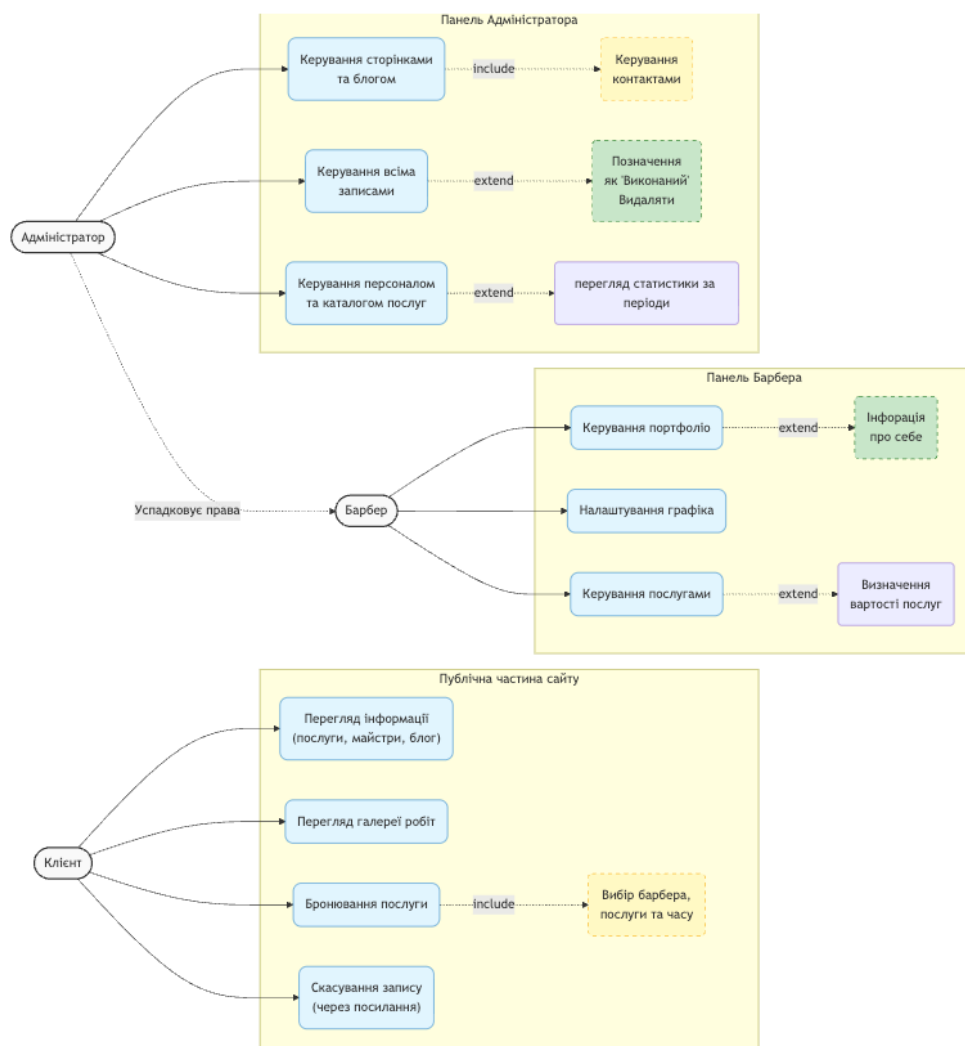


Рис. 2.1. Варіанти використання

Змн.	Арк.	№ докум.	Підпис	Дата

В нашій системі адміністратор виконує роль центрального елемента керування всіма основними видами бізнес-процесів сайту. Його функціонал значно ширший у порівнянні з іншими користувачами, оскільки саме адміністратор відповідає за стабільність роботи барбершопу, актуальність інформації та координацію взаємодії між клієнтами та перукарями. Через адміністративну панель здійснюється контроль над контентом сайту, керування персоналом, записами клієнтів та відображення загальних статистичних метрик для здійснення аналізу та прийняття управлінських рішень.

Адміністратор відповідає за формування публічного образу барбершопу та інформаційне наповнення сайту. Через систему керування сторінками та блогом він може редагувати головну сторінку, змінювати контактну інформацію, публікувати статті для блогу. Це дозволяє підтримувати актуальність контенту та використовувати сайт не лише як систему бронювання, а і як маркетинговий інструмент. Ведення блогу позитивно впливає на SEO-оптимізацію сайту та допомагає залучати нових клієнтів через пошукові системи, як основний елемент залучення нових клієнтів.

Однією з ключових функцій адміністратора є керування каталогом послуг. Саме адміністратор формує перелік процедур, які доступні для бронювання клієнтами. У системі він може створювати нові послуги, вказувати їхню назву, тривалість виконання, вартість та додатковий опис. Наприклад, адміністратор може додати окрему послугу для стрижки бороди, укладки волосся або комплексного догляду. Усі ці дані використовуються у процесі онлайн-бронювання та відображаються клієнтам у публічній частині сайту. Таким чином, саме адміністратор визначає структуру всіх послуг, які надає барбершоп.

Важливою складовою адміністративного функціоналу є керування персоналом. Адміністратор створює облікові записи для нових барберів, редагує їхні дані та контролює доступ до системи. Без створення відповідного профілю майстер не може авторизуватися у власній панелі керування, налаштовувати робочий графік або з'являтися у списку доступних спеціалістів для клієнтів. Це

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

означає, що адміністратор фактично відповідає за підключення нових працівників до системи та підтримання актуального складу персоналу.

Окреме значення має функція керування записами клієнтів. Адміністратор отримує доступ до повного розкладу всіх барберів та може переглядати інформацію про кожне бронювання. У випадку виникнення форс-мажорних ситуацій, наприклад відсутності майстра або технічних проблем, адміністратор має право скасувати будь-який запис. Це дозволяє швидко реагувати на зміни та підтримувати коректну роботу системи бронювання.

Крім скасування записів, адміністратор також завершує цикл обслуговування клієнта через функцію позначення запису як виконаного. Після завершення процедури відповідний запис отримує статус «Виконаний», що означає успішне завершення обслуговування. У реальній системі така дія є важливою складовою фінансового обліку та формування статистики роботи барбершопу. Саме на основі завершених записів можна аналізувати кількість наданих послуг, оцінювати навантаження на працівників та формувати звітність.

Окрему увагу приділено статистичному дашборду барбера, який дозволяє майстру аналізувати загальну ефективність та контролювати робочі показники. Після авторизації барбер отримує доступ до персональної панелі, де відображається інформація лише про його записи, доходи та виконані послуги. Такий підхід забезпечує персоналізовану аналітику та допомагає оцінювати результати роботи за різні періоди часу.

У статистичному блоці адміністратор може переглядати кількість виконаних записів, загальний дохід від наданих послуг та середній чек клієнта. Система автоматично аналізує завершені бронювання та формує основні показники ефективності, що дозволяє майстру бачити власне навантаження та фінансові результати. Додатково можуть відображатися найпопулярніші послуги, які найчастіше замовляють клієнти.

Для зручності аналізу передбачено графічне представлення статистики. Адміністратор може переглядати динаміку доходів та кількість записів у вигляді

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

графіків за день, тиждень, місяць або рік. Це дозволяє оцінювати сезонні зміни попиту, визначати найбільш завантажені дні та аналізувати тенденції у роботі. Подібна аналітика допомагає майстру ефективніше планувати власний графік та оцінювати рівень продуктивності.

Система також підтримує фільтрацію статистики за часовими періодами та конкретними датами. Завдяки цьому адміністратор може швидко отримувати інформацію про власну діяльність за обраний проміжок часу. Такий функціонал є корисним для контролю командних результатів, планування робочого навантаження та мотивації для підвищення якості обслуговування клієнтів.

Дуже важливою частиною цілісної системи є панель барбера. Завдяки ній більшість частини управління операційною діяльністю бізнесу покладається на самих перукарів. Вони працюють як фрілансери, які самі визначають час роботи та ціни за свої послуги, а барбершоп виступає платформою для роботи.

Панель барбера реалізує концепцію управління персональним брендом та операційною діяльністю майстра. Вона є окремим функціональним модулем, який забезпечує кожному барберу повний контроль над тим, як він представлений у системі, як організований його робочий час та як відбувається взаємодія з клієнтами. У межах діаграми варіантів використання цей блок можна розглядати як сукупність прецедентів, що формують професійне середовище роботи майстра.

Однією з ключових частин є управління особистим брендом, яке безпосередньо впливає на популярності барбера серед клієнтів. Барбер має можливість редагувати свій публічний профіль, змінюючи аватар, опис професійного досвіду та контактну інформацію. Це дозволяє підтримувати актуальність даних і формувати довіру з боку клієнтів. Окрему роль відіграє цифрове портфоліо, яке є візуальною демонстрацією робіт майстра. Барбер може завантажувати фотографії виконаних зачісок, укладок або інших процедур, тим самим формуючи власну професійну презентацію на сайті.

Важливою функцією портфоліо є можливість сортування робіт. Барбер самостійно визначає порядок відображення фотографій, розміщуючи найкращі

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

роботи на початку списку. Це дозволяє підкреслити рівень майстерності та створити позитивне перше враження у потенційних клієнтів. Також передбачено можливість видалення застарілих або неактуальних робіт, що дозволяє підтримувати актуальність портфоліо. У сукупності ці функції формують сукупність маркетингового представлення майстра, який безпосередньо впливає на конверсію переглядів у реальні записи.

Наступним важливим компонентом є гнучке управління робочим часом, яке реалізоване у вигляді динамічної системи розкладу. Барбер не працює зі статичним графіком, а формує власну робочу сітку на основі доступних часових слотів. Він може визначати робочі дні, години прийому клієнтів та планувати свій графік на тиждень або місяць вперед. Такий підхід дозволяє адаптувати робочий час під індивідуальні потреби майстра.

Окрему роль відіграє управління часовими слотами. Барбер може скасовувати конкретні інтервали часу без необхідності видалення всього робочого дня. Наприклад, якщо виникає потреба зробити перерву або тимчасово залишити робоче місце, він може заблокувати лише відповідний проміжок часу. Це забезпечує гнучкість у плануванні та дозволяє уникати перевантаження або накладення записів. Такий механізм є критично важливим для запобігання overbooking та підтримання коректної роботи системи бронювання.

В системі враховується залежність доступного часу від кількості робочих місць у барбершопі та їх фактичної зайнятості. Це означає, що формування доступних часових слотів відбувається не лише на рівні графіка окремого майстра, а й з урахуванням фізичної інфраструктури закладу. Кожне крісло розглядається як окрема одиниця ресурсів, яка може бути зайнята або вільна у конкретний проміжок часу.

У процесі запису барбер не лише обирає час, але й закріплює його за конкретним робочим місцем, фактично резервуючи крісло на період виконання послуги. Такий підхід дозволяє уникнути конфліктів між майстрами, оптимізувати завантаженість простору та забезпечити коректний розподіл клієнтів між

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

доступними робочими зонами. В результаті система стає більш точною в плануванні та ефективнішою в управлінні ресурсами барбершопу.

Ще одним функціональним блоком є формування власного прайс-листа, який визначає перелік послуг, що фактично надаються барберу. Хоча загальний каталог послуг створюється адміністратором, саме барбер обирає, які з них він готовий виконувати. Це дозволяє формувати індивідуальну спеціалізацію майстра. Наприклад, один барбер може працювати лише зі стрижками, тоді як інший поєднує стрижки з голінням або іншими процедурами.

Додатково передбачена можливість активування та деактивування окремих послуг у будь-який момент. Це дає змогу оперативно реагувати на зміни робочих умов, наприклад у разі відсутності інструментів або зміни професійного фокусу. Окрім цього, така функціональність мінімізує ризик конфліктів під час онлайн-запису, оскільки користувачі бачитимуть для вибору лише актуальні пропозиції. Таким чином, цей прецедент відображає концепцію професійної спеціалізації та гнучкого налаштування сервісів.

Операційна частина панелі барбера представлена у вигляді моніторингу записів та щоденного навантаження. Майстер має доступ до актуального списку клієнтів на день у режимі реального часу. Це дозволяє йому ефективно планувати робочий процес та контролювати чергу. Крім того, система надає детальну інформацію про кожен запис, включаючи складність замовлення та перелік обраних послуг.

Завдяки цьому барбер може заздалегідь підготувати робоче місце та необхідні інструменти для виконання процедур. Такий функціонал формує чітке планування робочого дня, що підвищує ефективність обслуговування клієнтів та зменшує час очікування.

Окрему увагу приділено питанням безпеки та персоналізації інтерфейсу. Барбер має можливість керувати налаштуваннями безпеки, включаючи зміну пароля та використання двофакторної автентифікації. Оскільки майстер працює з персональними даними клієнтів, такими як імена та контактні номери, захист

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

інформації є критично важливим елементом системи.

Додатково реалізовано функцію персоналізації інтерфейсу, яка дозволяє обирати тему оформлення системи, зокрема світлий або темний режим. Це забезпечує комфортну роботу в різних умовах освітлення, що особливо важливо для щоденного використання системи в робочому середовищі барбершопу.

Отже, панель барбера є комплексним модулем, який поєднує інструменти управління брендом, робочим графіком, послугами та операційною діяльністю. Вона забезпечує баланс між гнучкістю роботи майстра та структурованістю бізнес-процесів системи.

Третім компонентом системи є клієнтська панель та логіка взаємодії користувача з барбершопом. Вона реалізує концепцію зручного, інтуїтивного та повністю цифрового процесу запису без необхідності створення облікового запису, що знижує поріг входу та підвищує конверсію відвідувачів у реальних клієнтів. У межах діаграми варіантів використання цей блок можна розглядати як послідовність прецедентів, що відображають шлях клієнта від ознайомлення з послугами до завершення запису та його подальшого управління.

Однією з ключових частин є етап дослідження та вибору, який формує основу прийняття рішення клієнтом. На цьому рівні користувач має доступ до публічної інформації про барбершоп, включаючи перелік майстрів, їх портфоліо та опис професійної діяльності. Особисті сторінки барберів формують індивідуальне представлення кожного майстра, де відображаються його роботи, відсортовані відповідно до налаштувань самого барбера. Додатково клієнт може ознайомитися з переліком послуг та їх вартістю, яка може варіюватися залежно від обраного майстра, що дозволяє сформувати більш точне уявлення про кінцеву вартість візиту.

Наступним етапом є процес запису, який реалізований як багатокроковий сценарій взаємодії клієнта із системою. Спочатку користувач обирає майстра, після чого система автоматично фільтрує доступні послуги, залишаючи лише ті, які реально надаються обраним барбером. Далі відбувається пошук доступного часу,

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

під час якого система аналізує графік майстра та тривалість обраної послуги, формуючи перелік вільних часових слотів. Це виключає можливість накладання записів та забезпечує коректне планування. У випадках, коли користувач не знаходить зручного часу, система додатково аналізує наступні періоди та пропонує найближчі доступні дати. Завершальним кроком є оформлення запису, під час якого клієнт вводить базові контактні дані для підтвердження бронювання.

Окрему роль відіграє механізм керування записом після його створення. Оскільки система не передбачає обов'язкової реєстрації клієнтів, доступ до керування візитом реалізовано через безпечні унікальні посилання. Після створення запису клієнт отримує підтвердження з криптографічно підписаним URL, який дозволяє переглянути або скасувати бронювання без додаткової авторизації. При переході за цим посиланням користувач може виконати скасування візиту, а система додатково перевіряє стан запису, запобігаючи повторним або некоректним діям. Такий підхід забезпечує баланс між зручністю користувача та безпекою операцій.

Отже, у цьому розділі було визначено ключові ролі користувачів в системі: Адміністратора, Барбера та Клієнта, і побудовано діаграму варіантів використання, яка наочно демонструє їхні сценарії взаємодії з платформою. Аналіз цих прецедентів дозволив чітко структурувати логіку роботи системи, розмежувавши функціонал між панеллю адміністратора, робочим кабінетом барбера та публічною частиною сайту. Це створює надійне підґрунтя для формування остаточних функціональних вимог та подальшої ефективної розробки вебзастосунку.

2.2. Розробка бази даних системи

Для зберігання інформації в системі автоматизації роботи барбершопу була реалізована реляційна база даних на основі PostgreSQL. Вона використовується як основний механізм збереження всіх даних платформи та забезпечує стабільну й швидку роботу при постійній взаємодії великої кількості користувачів і записів.

Під час проєктування структури даних основну увагу приділено логічному розділенню інформації на окремі сутності та зменшенню дублювання даних. Це

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє підтримувати цілісність інформації та спрощує подальше розширення системи без необхідності суттєвих змін у вже існуючій архітектурі.

У межах бази даних зосереджено всі ключові процеси роботи барбершопу: облік користувачів різних ролей (адміністраторів, барберів і клієнтів), профілі майстрів із їхніми роботами, перелік доступних послуг і індивідуальні налаштування цін, а також розклад роботи та система бронювання. Окремо зберігаються дані про записи клієнтів, їхній статус і історія відвідувань, що дозволяє відстежувати повний цикл взаємодії з сервісом.

Нижче подано таблицю (табл. 2.1) з описом та призначенням основних таблиць бази даних.

Таблиця 2.1.

Структура бази даних

Назва таблиці	Призначення
users	Головна таблиця системи. Зберігає дані Адміністраторів та Барберів (роль, ім'я, email, пароль, аватар, біографія)
schedules	Зберігає робочий графік Барберів.
bookings	Зберігає всі записи клієнтів.
services	Глобальний довідник усіх послуг барбершопу
service_user	Зв'язує Барберів та Послуги.
booking_service	Зв'язує конкретний запис клієнта (booking) з обраними послугами
portfolio_works	Зберігає фотографії для портфоліо Барберів.
pages	Зберігає контент статичних сторінок (наприклад, тексти та налаштування для Головної сторінки).
posts	Зберігає статті для Блогу. Містить заголовок, текст, зображення та статус публікації (is_published).
password_reset_tokens	Зберігає тимчасові токени для безпечного відновлення забутого пароля користувачами.
sessions	Зберігає інформацію про активні сесії авторизованих користувачів (для безпеки та відстеження пристроїв).

Тепер детально розглянемо приклади таблиць і полів, які відображають важливі частини сайту барбершопу.

Спершу розглянемо таблицю users (табл. 2.2). Вона призначена для зберігання інформації про користувачів системи, включаючи адміністраторів та барберів. Вона забезпечує автентифікацію та авторизацію користувачів, зберігання

персональних даних, налаштувань профілю, інформації для двофакторної автентифікації, а також підтримує функціонал персональних сторінок барберів і безпечного доступу до системи.

Таблиця 2.2.

Опис таблиці users

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор користувача
name	varchar	Повне ім'я користувача
email	varchar	Електронна пошта користувача
role	varchar	Роль користувача в системі
email_verified_at	timestamp, nullable	Дата та час підтвердження електронної пошти
password	varchar	Хешований пароль користувача
avatar	varchar, nullable	Шлях до аватара користувача
position	varchar, nullable	Посада або спеціалізація барбера
slug	varchar	Унікальний URL-ідентифікатор профілю
bio	text, nullable	Біографія або опис користувача
remember_token	varchar	Токен для функції «Запам'ятати мене»
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення
deleted_at	timestamp, nullable	Час м'якого видалення запису
two_factor_secret	text, nullable	Секретний ключ для двофакторної автентифікації
two_factor_recovery_codes	text, nullable	Коди відновлення доступу для 2FA
two_factor_confirmed_at	timestamp, nullable	Час підтвердження двофакторної автентифікації

Таблиця bookings (табл. 2.3) призначена для зберігання інформації про бронювання клієнтів у системі. Вона містить дані про запис до барбера, контактну інформацію клієнта, дату та час візиту, статус бронювання, вартість послуг, а також дозволяє реалізувати механізм технічного блокування часових слотів у графіку роботи. Додатково таблиця забезпечує можливість подальшого аналізу завантаженості майстрів і формування статистики відвідувань, що є важливим для оптимізації роботи закладу.

Таблиця 2.3.

Опис таблиці bookings

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор бронювання
user_id	bigint	Ідентифікатор барбера, до якого здійснено запис
client_first_name	varchar	Ім'я клієнта
client_last_name	varchar	Прізвище клієнта
client_email	varchar	Електронна пошта клієнта
client_phone	varchar	Контактний номер телефону клієнта
start_at	datetime	Дата та час початку візиту
end_at	datetime	Дата та час завершення візиту
total_price	decimal	Загальна вартість послуг
status	varchar	Поточний статус бронювання
is_blocked	boolean	технічне блокування слота
notes	text	Додаткові коментарі або примітки
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення
deleted_at	timestamp	Час м'якого видалення бронювання
status	varchar	Поточний статус бронювання

Таблиця services (табл. 2.4) призначена для зберігання інформації про послуги, які надаються у барбершопі. Вона містить основні характеристики послуг, зокрема їх назву, опис та тривалість виконання, що дозволяє використовувати ці дані під час формування записів клієнтів і планування графіка роботи барберів.

Сервіси створює адміністратор на своїй панелі, після чого вони стають доступними до використання, така структура дозволяє централізовано керувати переліком послуг та оперативно оновлювати їх параметри відповідно до потреб бізнесу.

Таблиця 2.4.

Опис таблиці services

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор послуги
name	varchar	Назва послуги
description	text	Опис або додаткова інформація про послугу
duration_minutes	integer	Тривалість виконання послуги у хвилинах
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця service_user (табл. 2.5) призначена для реалізації зв'язку між барберами та послугами, які вони надають. Вона дозволяє визначити перелік доступних послуг для кожного барбера, зберігати індивідуальні ціни, а також контролювати активність певної послуги у конкретного спеціаліста.

Таблиця 2.5.

Опис таблиці service_user

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор запису
user_id	bigint	Ідентифікатор барбера
service_id	bigint	Ідентифікатор послуги
is_active	boolean	Ознака активності послуги у барбера
price	decimal	Індивідуальна вартість послуги у конкретного барбера
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця booking_service (табл. 2.6) призначена для зберігання інформації про послуги, що входять до конкретного бронювання. Вона реалізує зв'язок між записами клієнтів та обраними послугами, а також дозволяє фіксувати актуальну вартість послуги на момент створення бронювання.

Таблиця 2.6.

Опис таблиці booking_service

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор запису
booking_id	bigint	Ідентифікатор бронювання
service_id	bigint	Ідентифікатор послуги
price	decimal	Вартість послуги на момент оформлення запису
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця pages призначена для зберігання контенту статичних або динамічних сторінок системи. Вона дозволяє керувати інформаційними сторінками (наприклад, «Про нас», «Контакти», головна сторінка) через структуровані дані у форматі JSON, що забезпечує гнучкість у відображенні та редагуванні контенту без зміни коду (табл. 2.7). Завдяки такому підходу контент можна доволі швидко адаптувати під зміни вимог або оновлення структури сайту без втручання в програмну логіку системи.

Використання JSON-структур дає можливість зберігати різні типи контенту в межах одного запису, включаючи текстові блоки, зображення, списки, банери та інші елементи інтерфейсу. Це значно спрощує процес адміністрування сайту та забезпечує можливість централізованого керування вмістом через адміністративну панель.

Таблиця 2.7.

Опис таблиці pages

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор сторінки
slug	varchar	Унікальний системний ідентифікатор (URL-адреса сторінки)
content	Json	Структурований контент сторінки
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця schedules (табл. 2.8) призначена для зберігання інформації про графік роботи барберів. Вона дозволяє визначати робочі зміни кожного спеціаліста, включаючи дату, час початку та завершення зміни, а також робоче місце.

Таблиця 2.8.

Опис таблиці schedules

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор графіка
user_id	bigint	Ідентифікатор барбера
chair_number	integer	Номер робочого місця (крісла)
date	date	Дата робочої зміни
start_time	time	Час початку зміни
end_time	time	Час завершення зміни
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця portfolio_works (табл. 2.9) призначена для зберігання робіт із портфоліо барберів. Вона дозволяє формувати персональні галереї майстрів, демонструвати їхні навички та стиль виконання робіт, а також керувати порядком відображення елементів у профілі. Це важлива частина системи персонального брендингу барбера, яка впливає на привабливість профілю та допомагає клієнтам оцінити якість послуг ще до запису. Найголовніше що барбер сам може вести цю сторінку, і не залежати ні від кого.

Таблиця 2.9.

Опис таблиці portfolio_works

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор запису портфоліо
user_id	bigint	Ідентифікатор барбера, якому належить робота
image_path	varchar	Шлях до зображення виконаної роботи
description	varchar	Опис роботи або додаткові коментарі
order	integer	Порядок відображення роботи в портфоліо
created_at	timestamp	Дата та час створення запису
updated_at	timestamp	Дата та час останнього оновлення запису

Таблиця posts (табл. 2.10) використовується для реалізації блогу системи та є основним інструментом для публікації інформаційного контенту. Вона дозволяє створювати та керувати статтями, які можуть містити як маркетингові матеріали (новини барбершопу, акції, оновлення сервісу), так і корисний контент для клієнтів (поради з догляду, тренди стрижок, рекомендації стилю). Завдяки підтримці SEO-структури та статусу публікації система забезпечує гнучке керування видимістю контенту та його відображенням на сайті.

Окрему роль відіграє поле `content`, яке дозволяє зберігати повноцінний текст статті у форматі HTML або Markdown, що дає можливість реалізувати багатий форматований контент із заголовками, зображеннями та структурованими блоками. Поля `excerpt` та `image_path` використовуються для формування прев'ю поста у списках або на головній сторінці, що покращує користувацький досвід і навігацію.

Також таблиця підтримує базову авторську інформацію через поле `author`, а поле `is_published` дозволяє розділяти чернетки та опубліковані матеріали, що є важливим для процесу контент-менеджменту. Завдяки цьому таблиця `posts` є центральним інструментом для побудови блогу всередині платформи.

Таблиця 2.10.

Опис таблиці `posts`

Поле	Тип	Призначення
<code>id</code>	<code>bigint</code>	Унікальний ідентифікатор поста
<code>title</code>	<code>varchar</code>	Заголовок публікації
<code>slug</code>	<code>varchar</code>	Унікальний URL-ідентифікатор поста
<code>excerpt</code>	<code>text</code>	Короткий анонс поста
<code>content</code>	<code>longtext</code>	Основний текст поста (HTML або Markdown)
<code>image_path</code>	<code>varchar</code>	Головне зображення для прев'ю
<code>author</code>	<code>varchar</code>	Автор публікації
<code>is_published</code>	<code>boolean</code>	Статус публікації (опубліковано/чернетка)
<code>created_at</code>	<code>timestamp</code>	Дата та час створення поста
<code>updated_at</code>	<code>timestamp</code>	Дата та час останнього оновлення поста

Системні таблиці Laravel відіграють також важливу роль у забезпеченні стабільної та ефективної роботи всього застосунку, оскільки вони відповідають за внутрішню інфраструктуру фреймворку. Вони забезпечують обробку сесій

користувачів, кешування даних для пришвидшення доступу до інформації, а також керування чергами задач, що виконуються у фоновому режимі. Завдяки цьому система працює швидше, краще масштабується та не перевантажує основну базу даних зайвими операціями.

Зокрема, таблиця sessions зберігає інформацію про активні сесії користувачів, що дозволяє підтримувати авторизацію, відстежувати активність та реалізовувати механізми безперервної взаємодії з системою. Таблиця cache використовується для тимчасового збереження даних, які часто запитуються, що суттєво зменшує кількість звернень до основної бази даних і покращує продуктивність застосунку. У свою чергу, таблиця jobs є частиною системи черг Laravel і забезпечує виконання відкладених або ресурсоємних задач у фоновому режимі, таких як надсилання повідомлень або обробка даних.

Запропонована структура бази даних забезпечує узгодженість інформації за рахунок використання зовнішніх ключів між таблицями, а також дотримання принципів нормалізації, що мінімізує дублювання даних і підвищує їхню цілісність. Наприклад, зв'язки між барберами та послугами, а також між бронюваннями та обраними сервісами дозволяють гнучко керувати даними без необхідності їх дублювання в різних таблицях.

Окремо варто відзначити можливість масштабування системи, оскільки основні сутності (користувачі, бронювання, послуги, графіки та портфоліо) винесені в окремі таблиці з чітко визначеними зв'язками. Це дозволяє без ускладнень розширювати функціональність платформи, наприклад додаючи нові типи послуг, додаткові атрибути профілів або розширену аналітику роботи барберів.

Діаграма бази даних (рис. 2.2) відображає взаємозв'язки між ключовими таблицями, які формують повний цикл роботи платформи: від користувачів системи та їх ролей до управління послугами, записами клієнтів, графіками роботи та контентною частиною (портфоліо, сторінки, блог). У сукупності ці сутності створюють цілісну екосистему для організації роботи барбершопу, де кожен етап

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

від створення послуги до бронювання та виконання логічно пов'язаний із наступним.

Крім того, така модель даних спрощує подальшу обробку інформації та формування звітності, оскільки дані зберігаються у структурованому вигляді, включно з підтримкою JSON-полів для гнучкого контенту сторінок. Чітко визначені зв'язки між таблицями забезпечують швидкий доступ до потрібної інформації, наприклад до розкладу барбера або історії бронювань клієнта.

Також важливим аспектом є безпека та контроль доступу до даних. Розмежування ролей користувачів (адміністратор, барбер тощо) дозволяє обмежувати доступ до певних функцій і даних залежно від рівня прав, що підвищує захищеність системи та зменшує ризик несанкціонованих змін.



Рисунок 2.2 - Діаграма бази даних системи

У підсумку така архітектура бази даних забезпечує стабільну роботу платформи, її гнучкість до змін бізнес-вимог та можливість подальшого масштабування. Вона також сприяє високій продуктивності системи навіть при збільшенні обсягу даних і кількості користувачів, що є важливим для довгострокового розвитку сервісу.

2.3. Проектування та реалізація алгоритмів роботи системи

Оскільки ключовим елементом нашої системи є функція запису клієнтів на стрижку, то першим варто розглянути алгоритм контролю доступності часових інтервалів, який забезпечує коректне планування записів та виключає можливість накладання подій. Його основне призначення полягає у синхронізації реального робочого графіка барбера з новими запитами на бронювання, а також у запобіганні конфліктам між уже існуючими записами та адміністративними обмеженнями. Саме цей механізм формує базовий рівень надійності всієї системи розкладу.

На першому етапі алгоритм отримує вхідні параметри, необхідні для обробки запиту. До них належать ідентифікатор барбера, дата запланованого візиту, бажаний час початку, а також перелік обраних клієнтом послуг. Ці дані формують початковий контекст операції та використовуються для подальших перевірок у різних модулях системи.

Далі виконується перевірка відповідності запиту робочому графіку барбера. Для цього здійснюється звернення до таблиці `schedules`, у якій зберігається інформація про зміну майстра. Алгоритм перевіряє, чи існує активний робочий інтервал на обрану дату та чи входить запитаний час початку у межі робочої зміни, визначеної параметрами `start_time` та `end_time`. Якщо умова не виконується, подальша обробка запиту припиняється.

Наступним кроком є визначення очікуваної тривалості візиту. На основі масиву обраних послуг система обчислює загальний час виконання, після чого формується розрахунковий час завершення для запису. Цей етап є критичним, оскільки саме від нього залежить точність подальшої перевірки перетинів у розкладі.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Після цього виконується основний етап алгоритму – перевірка на наявність колізій із вже існуючими записами. Система формує запит до таблиці bookings, у межах якого аналізуються всі записи, що належать відповідному барберу. Перевірка здійснюється за класичними правилами перетину часових інтервалів: новий запис не може починатися або завершуватися всередині вже зайнятого проміжку, а також не може повністю охоплювати існуючий запис. Таким чином виключаються всі можливі сценарії накладання.

Окремо в межах цього етапу враховуються технічні та службові блокування часу. Записи, позначені прапорцем `is_blocked = true`, трактуються системою як недоступні інтервали, незалежно від їхнього походження. Вони обробляються аналогічно до звичайних бронювань, що дозволяє барберу резервувати час для перерв, підготовки або інших робочих потреб.

На завершальному етапі алгоритм приймає рішення щодо можливості створення запису. Якщо в результаті перевірки виявлено хоча б один конфліктний інтервал, система перериває процес і повертає помилку доступності із рекомендацією вибору альтернативного часу. У випадку відсутності перетинів усі перевірки вважаються успішними, і транзакція переходить до етапу збереження нового запису в базі даних.

У процесі роботи бронювання важливе місце займає алгоритм динамічного розрахунку тривалості сеансу та формування фінальної вартості послуг. Його роль полягає у забезпеченні точного планування часу візиту клієнта та коректного формування чека з урахуванням індивідуальних налаштувань барбера. Цей механізм дозволяє системі одночасно вирішувати дві задачі: запобігати конфліктам у розкладі та забезпечувати гнучку цінову політику для кожного майстра.

На початковому етапі алгоритм ініціалізує службові змінні, які використовуються для накопичення результатів обчислень. Зокрема, встановлюється `total_duration`, що дорівнює нулю і вимірюється у хвилинах, а також `total_price`, який ініціалізується як 0.00. Ці змінні надалі використовуються як акумулятори для сумарної тривалості та загальної вартості всіх обраних послуг.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі система переходить до обробки масиву послуг, які були вибрані користувачем. Кожна послуга передається у вигляді унікального ідентифікатора `service_id`, що дозволяє однозначно визначити її характеристики у базі даних. Алгоритм послідовно ітерується по цьому масиву, забезпечуючи обробку кожного елемента без втрати даних.

На наступному етапі відбувається обчислення загальної тривалості сеансу. Для кожного `service_id` система звертається до таблиці `services` та отримує значення `duration_minutes`, яке відповідає тривалості виконання конкретної процедури. Отримані значення додаються до `total_duration`, у результаті чого формується сумарний час, необхідний для виконання всього набору послуг.

Після завершення обчислення тривалості система визначає очікуваний час завершення візиту. Для цього значення `total_duration` додається до початкового часу запису `start_at`, у результаті чого формується точне значення `end_at` у форматі Y-m-d H:i:s. Отриманий результат є критично важливим, оскільки саме він використовується у алгоритмі перевірки доступності часових слотів та запобігання колізіям.

Окремим етапом реалізовано формування фінальної вартості послуг, що базується на індивідуальному прайс-листі кожного барбера. Для цього система звертається до зв'язувальної таблиці `service_user`, у якій зберігаються налаштування доступності та цін для конкретних майстрів. Пошук здійснюється за парою параметрів `user_id` та `service_id`, що дозволяє точно визначити ціну для кожної послуги.

У процесі розрахунку додатково перевіряється поле `is_active`. Якщо послуга вимкнена для конкретного барбера, виконання алгоритму переривається, а транзакція скасовується, що запобігає створенню некоректних записів. У випадку, коли індивідуальна ціна задана у таблиці `service_user`, саме вона додається до `total_price`. Якщо ж індивідуальна ціна відсутня, система може використовувати базову вартість із таблиці `services` як резервний варіант, забезпечуючи безперервність розрахунку.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

На завершальному етапі відбувається фіксація результатів у базі даних. Загальна вартість запису зберігається в таблиці bookings у полі total_price, тоді як детальна інформація про кожну послугу, включаючи її фактичну ціну на момент бронювання, записується у таблицю booking_service. Такий підхід дозволяє зберегти історичну точність даних і гарантує, що подальші зміни у прайсі не впливатимуть на вже створені записи та сформовані чеки.

У межах системи управління барбершопом важливим аспектом є не лише координація часу роботи майстрів, але й ефективний розподіл фізичних ресурсів, зокрема робочих крісел. Для цього реалізовано окремий алгоритм розподілу фізичних ресурсів, який забезпечує коректне планування завантаження робочого простору та запобігає конфліктам між барберами. Його основна задача полягає у раціональному використанні доступних крісел та недопущенні ситуацій, коли два майстри одночасно претендують на один і той самий фізичний ресурс.

На першому етапі алгоритму ініціюється процес створення нового запису розкладу. Це може відбуватися як через адміністративну панель, так і автоматично в межах системи планування. Для цього формується запис у таблиці schedules, який містить інформацію про конкретного барбера, дату роботи та закріплений номер крісла. Саме ці параметри є базовими для подальших перевірок.

Далі виконується перевірка зайнятості обраного крісла на вказану дату. Система здійснює пошук у таблиці schedules, використовуючи комбінацію двох ключових полів – date та chair_number. Такий підхід дозволяє швидко визначити, чи вже використовується це робоче місце іншим майстром у межах заданого дня.

Наступним кроком алгоритм переходить до аналізу часових інтервалів. У випадку, якщо крісло вже закріплене за іншим барбером на цю дату, система додатково порівнює часові рамки змін. Якщо часові проміжки не перетинаються, система дозволяє закріплення одного й того самого крісла за різними барберами, зокрема, аналізуються значення start_time та end_time як нового запису, так і вже існуючих записів для цього ж chair_number. Це дозволяє визначити, чи існує фактичне накладання робочих інтервалів у межах одного фізичного ресурсу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

У разі виявлення конфлікту система переходить до етапу вирішення колізії. Збереження нового запису розкладу блокується, а користувачу (адміністратору) повертається повідомлення про неможливість використання обраного крісла в заданий проміжок часу. Додатково алгоритм може сформувати рекомендації щодо альтернативних вільних крісел, які доступні у відповідний часовий інтервал, що спрощує процес ручного коригування розкладу.

Важливою оптимізаційною складовою є використання складеного індексу за полями `chair_number` та `date` у базі даних. Завдяки цьому пошук конфліктів виконується максимально ефективно навіть при значному обсязі даних, оскільки система звужує вибірку до конкретного дня та конкретного фізичного ресурсу. У результаті алгоритм забезпечує високу продуктивність планування та стабільну роботу системи навіть при високому навантаженні.

У сучасних вебсистемах, особливо в проєктах, що поєднують сервісну та маркетингову складову, важливу роль відіграє механізм керування контентом. У даній системі для барбершопу реалізовано окремий алгоритм створення та публікації блогових матеріалів, який забезпечує адміністраторам можливість формувати інформаційні статті, новини та акційні пропозиції. Окрему увагу в цьому процесі приділено автоматичній генерації SEO-дружніх URL-адрес, що підвищує видимість сайту в пошукових системах.

На першому етапі алгоритм здійснює прийом і первинну обробку вхідних даних, які надходять від адміністратора через інтерфейс керування контентом. До цих даних належать заголовки статті, короткий опис, та основний текст, який може бути представлений у форматі HTML або Markdown, а також медіафайл зображення, що використовується як прев'ю матеріалу. Усі ці параметри формують базову структуру майбутнього запису в блозі.

Наступним кроком виконується генерація унікального ідентифікатора сторінки – `slug`, який використовується у формуванні людино-зрозумілого URL. Для цього заголовки статті транслітерується з кирилиці в латиницю, після чого пробіли замінюються на дефіси. Наприклад, заголовок «Як доглядати за бородою»

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

перетворюється на якорну URL-структуру виду yak-doglyadati-za-borodoyu. Такий підхід забезпечує читабельність та SEO-оптимізацію адреси сторінки.

Після первинної генерації slug система виконує перевірку його унікальності. Для цього здійснюється запит до таблиці posts, у якій зберігаються всі існуючі публікації. Якщо виявляється, що згенерований slug уже використовується, алгоритм автоматично модифікує його шляхом додавання числового суфікса, наприклад yak-doglyadati-za-borodoyu-1. Такий механізм гарантує абсолютну унікальність URL навіть при повторюваних або схожих заголовках.

Далі виконується обробка завантаженого медіаконтенту. Зображення проходить валідацію на відповідність допустимим форматам і обмеженням за розміром, після чого зберігається у файловому сховищі системи, наприклад у директорії storage/app/public/posts. У базі даних при цьому фіксується лише відносний шлях до файлу, що дозволяє ефективно управляти ресурсами та спрощує міграцію або зміну файлової структури в майбутньому.

Завершальним етапом є визначення статусу публікації матеріалу. Адміністратор може обрати, чи буде стаття одразу доступна користувачам, чи збережеться у вигляді чернетки. У першому випадку запис створюється зі значенням is_published = true і стає доступним у публічній частині сайту. У другому випадку встановлюється значення is_published = false, що дозволяє зберегти матеріал для подальшого редагування або відкладеної публікації. Таким чином забезпечується гнучке керування контентом і контроль над інформаційним наповненням платформи.

У процесі побудови системи онлайн-бронювання важливим є забезпечення можливості керування записом без обов'язкової реєстрації користувача. Це особливо актуально для клієнтів, які залишають лише мінімальний набір контактних даних. Це дозволяє користувачу швидше оформити запис без введення зайвих даних для реєстрації.

Для вирішення цієї задачі реалізовано алгоритм формування безпечного посилання для керування бронюванням, який дозволяє клієнту скасовувати свій

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

візит без авторизації, водночас повністю виключаючи можливість доступу до чужих записів.

На першому етапі алгоритму відбувається фіксація створеного бронювання. Після проходження всіх попередніх перевірок (доступність часу, розрахунок тривалості та вартості) система створює новий запис у таблиці bookings. У результаті цієї операції формується унікальний ідентифікатор бронювання id, який стає базовим елементом для подальшої генерації захищеного посилання.

Далі запускається механізм генерації криптографічно підписаного URL. Для цього використовується вбудований функціонал фреймворку (наприклад, Laravel Signed Routes), який формує захищене посилання на основі секретного ключа сервера та параметрів запиту. Базовий маршрут сторінки скасування, наприклад /booking/cancel/{id}, доповнюється підписом, що гарантує цілісність і автентичність запиту.

Окремо до згенерованого посилання додається параметр часу дії, який обмежує період його валідності. У типовій реалізації посилання є дійсним лише до моменту start_at бронювання мінус певний резервний інтервал (наприклад, 2 години). Це дозволяє уникнути пізніх або потенційно небажаних скасувань у момент, коли послуга вже має бути надана. Приклад сформованого URL може виглядати як /booking/cancel/145?expires=1684758000&signature=a3f9b2c8e

Наступним етапом є відправка сформованого посилання клієнту. Воно інтегрується в повідомлення підтвердження бронювання та надсилається на електронну пошту або номер телефону, вказані під час запису. Таким чином клієнт отримує єдиний інструмент для подальшого керування своїм візитом без необхідності створення облікового запису.

Фінальним етапом алгоритму є валідація запиту при переході за посиланням. Перед відображенням сторінки скасування система перевіряє криптографічний підпис, порівнюючи його з очікуваним значенням, сформованим на основі тих самих параметрів та секретного ключа сервера. Додатково перевіряється термін дії посилання через параметр expires. У випадку невідповідності підпису або

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

закінчення терміну дії доступ до операції скасування блокується, що гарантує захист від підробки запитів та несанкціонованого доступу до бронювань інших користувачів. Також система фіксує спроби невалідних запитів у логах для подальшого моніторингу та аналізу безпеки. Це дозволяє своєчасно виявляти потенційні атаки та підвищує загальний рівень надійності.

2.4. Реалізація системи

Серверна частина системи реалізована на основі фреймворку Laravel, який забезпечує структуровану організацію коду та підтримує класичний архітектурний шаблон MVC. У даному випадку роль представлення частково заміщена фронтенд-фреймворком Vue.js, а зв'язок між ними реалізовано через Inertia.js, що дозволяє будувати SPA-подібний додаток без необхідності створення окремого REST API.

Основним елементом бекенд-логіки є директорія `app/Models/`, у якій знаходяться моделі даних, такі як `User`, `Booking` та `Service`. Кожна модель відповідає за взаємодію з конкретною таблицею бази даних та описує логічні зв'язки між сутностями. Наприклад, модель користувача визначає зв'язок «один-до-багатьох» між барбером і його послугами або записами клієнтів. Таким чином, моделі виступають центральною частиною бізнес-логіки системи та забезпечують цілісність даних.

Контролери розміщені в директорії `app/Http/Controllers/` і виконують роль посередників між запитам користувача та бізнес-логікою моделей. Вони приймають HTTP-запити, обробляють їх, взаємодіють із базою даних через моделі та повертають відповідь на фронтенд. Для кращої структуризації проєкту контролери поділені на логічні групи: `Admin`, `Barber` та `Clients`. Такий поділ дозволяє ізолювати функціонал різних ролей і спрощує подальшу підтримку системи.

Окремим шаром безпеки виступають `middleware`-компоненти, які знаходяться в директорії `app/Http/Middleware/`. Вони виконують функцію фільтрів запитів і перевіряють права доступу користувачів до певних ресурсів. Наприклад, `middleware CheckRole` обмежує доступ до адміністративної панелі лише

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачам із відповідною роллю. Це забезпечує захист системи від несанкціонованого доступу та підвищує рівень безпеки.

Маршрутизація системи реалізована через директорію routes/, де файли web.php, admin.php та settings.php визначають відповідність між URL-адресами та діями контролерів. Така структура дозволяє логічно розділити публічну частину сайту, адміністративну панель та налаштування системи, що значно покращує читабельність та масштабованість коду.

Окрему роль відіграє директорія database/migrations/, яка відповідає за керування структурою бази даних. Міграції дозволяють версіонувати зміни в таблицях, забезпечуючи можливість відтворення структури БД на будь-якому сервері без ручного налаштування. Це є важливим елементом DevOps-підходу та спрощує процес розгортання системи.

Клієнтська частина системи реалізована за допомогою Vue.js із використанням TypeScript, що забезпечує строгий контроль типів і зменшує кількість помилок під час розробки. Весь фронтенд-код знаходиться в директорії resources/js/, яка має чітко структуровану організацію для зручності масштабування проєкту.

У папці resources/js/pages/ розташовані основні сторінки застосунку. Кожен файл у цій директорії відповідає окремому екрану інтерфейсу, наприклад профілю барбера, сторінці послуг або системі бронювання. Коли бекенд-контролер обробляє запит, він передає керування відповідному Vue-компоненту, який відображає користувацький інтерфейс.

Папка resources/js/components/ містить повторно використовувані UI-елементи, які застосовуються в різних частинах системи. Це можуть бути компоненти інтерфейсу, такі як AppHeader, ImageUploader або AlertError. Використання компонентного підходу дозволяє забезпечити єдину стилістику інтерфейсу та спрощує внесення змін, оскільки редагування одного компонента автоматично впливає на всі сторінки, де він використовується. Повторне використання компонентів дозволяє скоротити обсяг дубльованого коду, спрощує

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

процес тестування та пришвидшує подальшу розробку нових функціональних можливостей. Крім того, централізоване зберігання компонентів полегшує підтримку проєкту та забезпечує більш зручну організацію клієнтської частини застосунку.

У директорії `resources/js/layouts/` зберігаються шаблони сторінок, які визначають загальну структуру інтерфейсу вебзастосунку. Наприклад, `AdminLayout` використовується для адміністративної панелі системи, а `ClientLayout` для публічної частини сайту. Вони формують базову структуру сторінок, включаючи навігацію, сайдбар та футер, що забезпечує єдину логіку в процесі відображення інтерфейсу сайту.

Окремо виділено папку `resources/js/composables/`, яка містить повторно використовувану бізнес-логіку у вигляді функцій, що застосовуються у різних частинах інтерфейсу. Наприклад, `useAppearance` відповідає за перемикання теми інтерфейсу (світла/темна), а `useInitials` використовується для генерації аватарів на основі імені користувача та забезпечує уніфіковане представлення профілів. Такий підхід дозволяє уникати дублювання коду, спрощує його супровід та підвищує загальну модульність проєкту.

Ключовим елементом архітектури є `Inertia.js`, який виступає проміжним шаром між бекендом та фронтом і забезпечує узгоджену взаємодію між `Laravel` і `Vue` без необхідності створення окремого REST API. Його використання дозволяє передавати дані напряму з серверної частини у клієнтські компоненти, мінімізуючи додаткові рівні абстракції та спрощуючи логіку обміну інформацією.

Завдяки `Inertia.js` система працює як SPA, при цьому вся бізнес-логіка залишається на стороні `Laravel`, що дозволяє централізовано керувати даними та процесами. Дані, які формуються у контролерах, передаються безпосередньо у `Vue`-компоненти у вигляді `props`, що забезпечує їх швидке відображення та зменшує затримки між запитами і рендерингом інтерфейсу. Це значно спрощує розробку, скорочує кількість проміжних шарів і робить систему більш продуктивною та легкою в підтримці [9].

Висновки до другого розділу

У даному розділі було розглянуто процес реалізації системи автоматизації роботи барбершопу, який охоплює як серверну, так і клієнтську частини застосунку, а також основні принципи організації взаємодії між ними. Детально описано архітектурні рішення, що базуються на використанні фреймворку Laravel для реалізації бекенд-логіки та Vue.js з TypeScript для побудови інтерактивного користувацького інтерфейсу, що забезпечує сучасний рівень зручності та гнучкості. Особливу увагу приділено структурі проєкту, яка включає моделі, контролери, middleware, маршрути та міграції, що в сукупності забезпечує чіткий розподіл відповідальностей, логічну організацію коду та підтримує масштабованість системи у разі подальшого розвитку.

Також було проаналізовано принцип взаємодії між бекендом і фронтом за допомогою Inertia.js, який дозволяє реалізувати SPA-подібну поведінку без необхідності створення окремого REST API, що суттєво зменшує складність архітектури. Це значно спрощує розробку та подальшу підтримку системи, забезпечуючи швидкий і передбачуваний обмін даними між серверною та клієнтською частинами. Додатково розглянуто структуру фронту, яка побудована на принципах модульності, повторного використання компонентів та розділення логіки, що підвищує зручність розвитку інтерфейсу та зменшує кількість дубльованого коду, а також робить систему більш структурованою та зрозумілою для подальшої роботи розробників.

У результаті запропонована архітектура реалізації системи забезпечує стабільну та надійну основу для подальшого розширення функціоналу, ефективну організацію коду та високу продуктивність роботи застосунку навіть при збільшенні навантаження або кількості користувачів. Чіткий поділ на логічні модулі дозволяє легко супроводжувати систему, впроваджувати нові можливості та підтримувати її в довгостроковій перспективі без втрати структурної цілісності, що є важливим фактором для реальних комерційних продуктів.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1. Порядок встановлення та налаштування системи

Процес встановлення та налаштування інформаційної системи керування барбершопом полягає у підготовці робочого середовища, завантаженні проєкту з репозиторію та налаштуванні основних параметрів його роботи. Система реалізована як монолітний вебзастосунок на базі Laravel, що дозволяє централізовано керувати всією бізнес-логікою. Для створення динамічного інтерфейсу без перезавантаження сторінок використовується Inertia.js, який поєднує серверну частину на PHP із клієнтською частиною, розробленою на Vue.js та TypeScript.

Першим етапом встановлення є підготовка середовища виконання. Для коректної роботи системи необхідно встановити PHP версії 8.4 або новішої, Composer для керування PHP-залежностями, Node.js версії 20 або новішої, а також npm для встановлення JavaScript-бібліотек. Додатково використовується система керування версіями Git, яка забезпечує можливість клонування репозиторію та подальшої роботи з кодом. Також необхідно встановити одну з підтримуваних систем баз даних, таких як MySQL, PostgreSQL або SQLite.

Після підготовки середовища виконується клонування репозиторію проєкту за допомогою Git-команди. Далі користувач переходить у директорію проєкту та встановлює всі необхідні залежності серверної частини за допомогою Composer. Після цього встановлюються клієнтські залежності через npm, які відповідають за роботу фронтенд-частини застосунку.

Наступним етапом є конфігурація середовища через файл змінних оточення .env, у якому задаються ключові параметри системи: підключення до бази даних, налаштування поштового сервера для надсилання повідомлень, а також службові ключі безпеки застосунку.

Коректне налаштування цих параметрів є критично важливим для стабільної роботи системи, оскільки вони визначають взаємодію між основними компонентами платформи.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Після налаштування середовища виконується генерація унікального ключа застосунку, який використовується для шифрування даних та забезпечення безпеки. Далі запускаються міграції бази даних, які автоматично створюють необхідну структуру таблиць, включаючи користувачів, послуги, графіки роботи та записи клієнтів. За потреби також можуть бути виконані сідери для наповнення бази тестовими даними.

Після завершення налаштування серверної частини запускається локальний сервер Laravel та процес компіляції фронтенду через Vite. У результаті система стає доступною у веббраузері за локальною адресою. Клієнтська частина автоматично оновлюється під час розробки, що дозволяє оперативно перевіряти зміни в інтерфейсі.

Діаграма розгортання відображає загальну структуру взаємодії компонентів системи після завершення етапу із налаштування середовища та запуску застосунку (рис. 3.1).

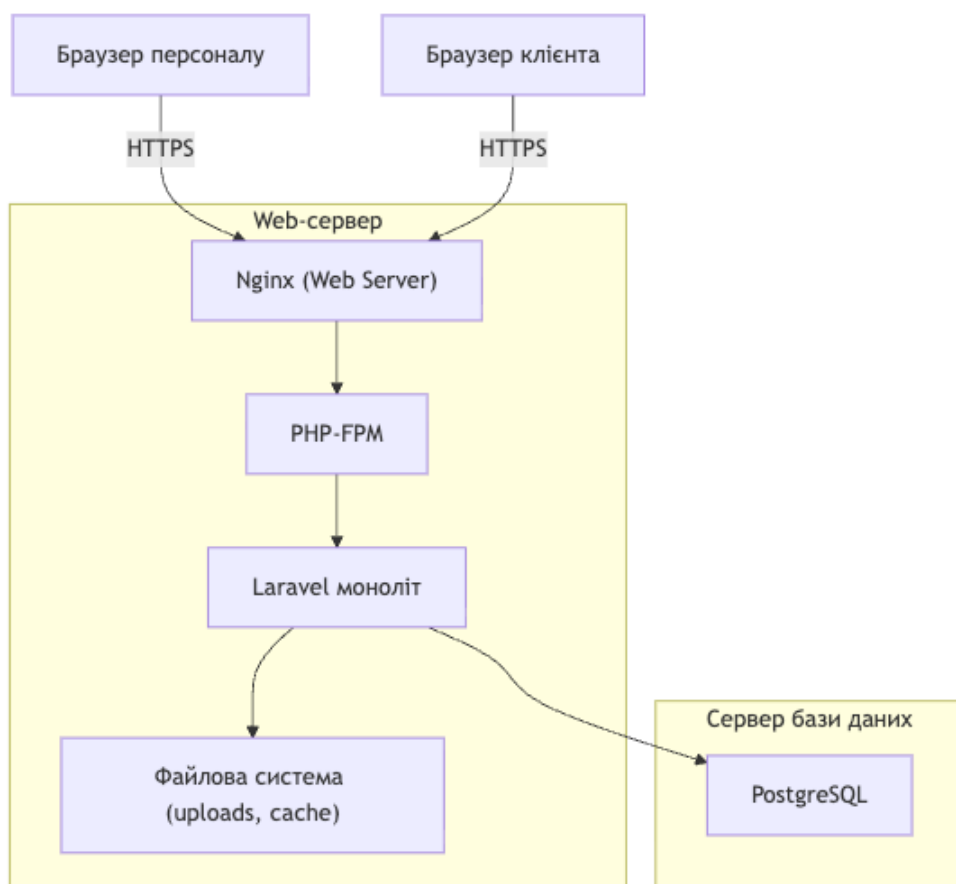


Рис. 3.1. Діаграма розгортання

Всі користувачі взаємодіють із платформою через веббраузери, надсилаючи http запити до вебсервера. Уся взаємодія відбувається через захищений протокол HTTPS, що забезпечує безпечну передачу даних між клієнтською та серверною частинами системи.

На стороні серверного середовища використовується вебсервер Nginx, який приймає вхідні запити та передає їх до інтерпретатора PHP-FPM. Далі запити обробляються монолітним застосунком на базі Laravel, який реалізує архітектурний шаблон MVC. Саме на цьому рівні виконується вся бізнес-логіка системи, включаючи обробку запитів користувачів, керування записами на послуги та взаємодію з базою даних.

Додатково в межах серверного середовища функціонує файлова система, яка використовується для зберігання завантажених користувачами файлів, кешу та інших службових даних застосунку. Це дозволяє оптимізувати роботу системи та зменшити навантаження на базу даних.

Окремим компонентом інфраструктури виділено сервер бази даних PostgreSQL, який забезпечує надійне збереження всієї структурованої інформації системи, включаючи дані користувачів, записи клієнтів, послуги та розклади. Взаємодія Laravel із базою даних здійснюється через ORM Eloquent, що забезпечує зручну та безпечну роботу з даними.

Критично важливо в процесі розробки використовувати CI/CD підхід, реалізований на базі GitLab. Після кожного внесення змін у репозиторій автоматично запускається pipeline, який включає етапи перевірки коду, встановлення залежностей та виконання тестів. Це дозволяє своєчасно виявляти помилки та забезпечує стабільність основної гілки проєкту. Головним етапом CI/CD-процесу є автоматизоване розгортання застосунку на сервері після успішного проходження всіх перевірок. Виконується деплой через підключення до серверного середовища, оновлюючи код застосунку, виконуючи міграції бази даних та перезапускаючи необхідні сервіси. Такий підхід значно скорочує час доставки змін у продакшн та мінімізує ризики, пов'язані з ручним розгортанням.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Структура інтерфейсу

Інтерфейс розробленої системи управління барбершопом створений з акцентом на інтуїтивну зрозумілість та зручність, що гарантує зручну взаємодію з платформою як на комп'ютерах, так і на мобільних пристроях клієнтів і всіх співробітників. Усі візуальні компоненти об'єднані в єдиний сучасний веб-застосунок на базі Vue.js та відповідних фронтенд-бібліотеки tailwind, що забезпечує безперебійну роботу та гнучкий адаптивний дизайн для коректної та безперебійної роботи системи.

Після переходу на сайт користувач потрапляє на головну сторінку сайту, де може ознайомитись з закладом (рис. 3.2). Головна клієнтська сторінка зустрічає користувача сучасним преміальним дизайном, який орієнтований на зручність взаємодії та візуальну привабливість, виконаним у глибокій темній кольоровій гамі з використанням контрастних помаранчевих акцентів. У верхній частині екрана розташована навігаційна панель, яка забезпечує швидкий доступ до ключових розділів платформи: переліку послуг, профілів майстрів, галереї робіт, блогу та контактної інформації. Для максимального комфорту клієнтів у правому куті хедера закріплено номер телефону та контрастну кнопку, що дозволяє миттєво ініціювати процес запису з будь-якої точки сайту.

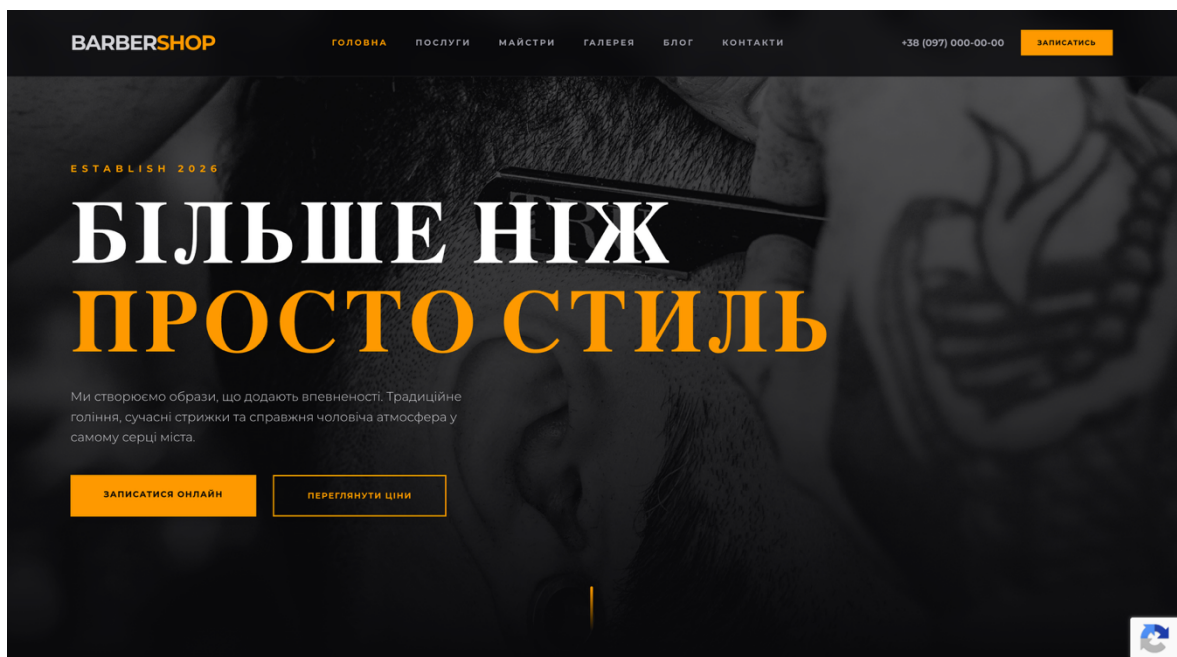


Рис. 3.2. Головна сторінка

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Логіка відображення ціни базується на агрегації даних з таблиці майстрів. Оскільки різні співробітники мають різну кваліфікаційну категорію, система не виводить фіксовану суму. Замість цього виконується запит із групуванням, який визначає мінімальну та максимальну вартість конкретної послуги серед усіх активних спеціалістів закладу.

Сторінка послуг (рис. 3.3) реалізована у вигляді структурованого інформаційного блоку, дані до якого динамічно завантажуються з бази даних через програмні інтерфейси. Навігаційне меню у верхній панелі автоматично синхронізується з поточною позицією користувача, візуально виділяючи активний розділ. Головним функціональним елементом сторінки є модульна сітка з переліком доступних процедур. Кожна картка містить унікальну назву послуги, короткий текстовий опис із деталізацією процедури та динамічно сформоване поле з вартістю.

Інтерфейс виводить цей результат у вигляді цінового діапазону. Це дозволяє клієнту одразу зорієнтуватися в ціновій політиці до моменту переходу до модуля вибору конкретного фахівця.

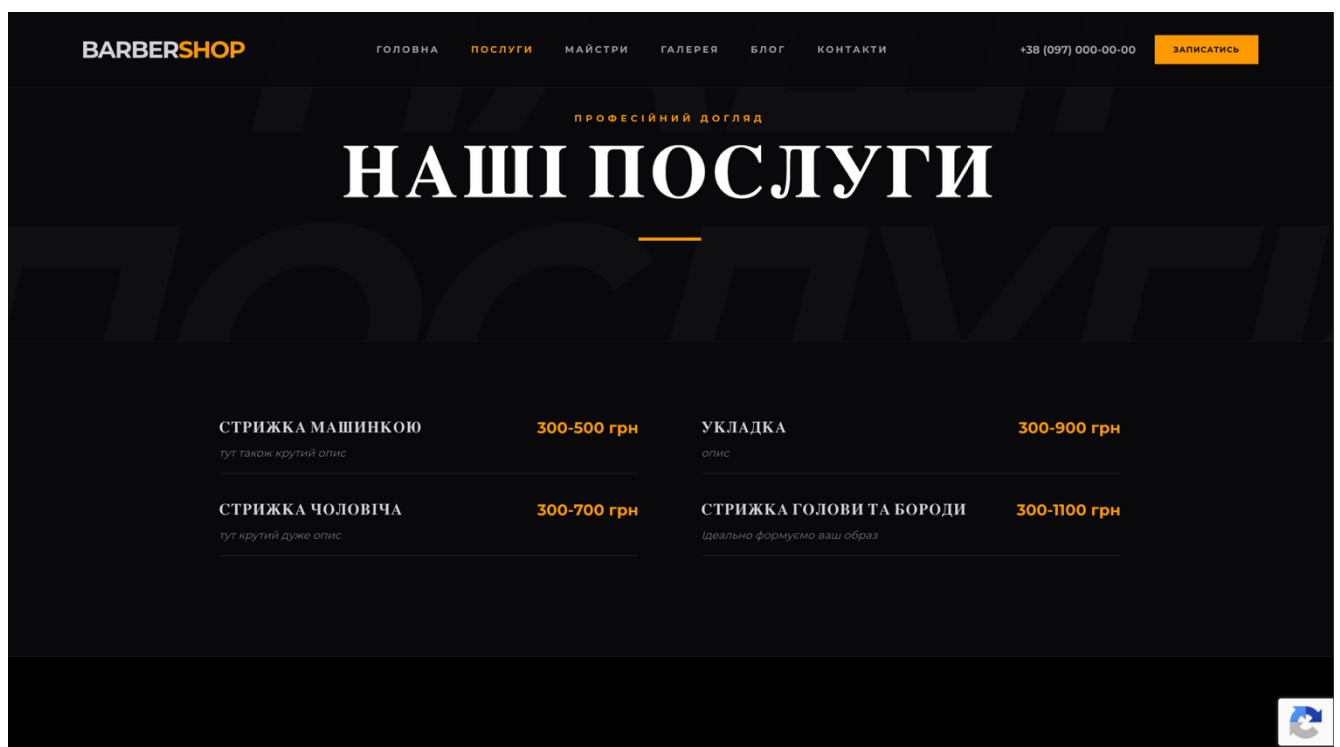


Рис. 3.3. Сторінка вакансії

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці майстрів (рис. 3.4) зображені картки всіх барберів. Тут можна коротко познайомитись з кожним перукарем, а також кожна картка є посиланням на персональне портфоліо майстра.

Для пришвидшення завантаження використовується підхід lazy loading, що дозволяє швидше завантажити сторінку, особливо при нестабільному підключенні до інтернету.

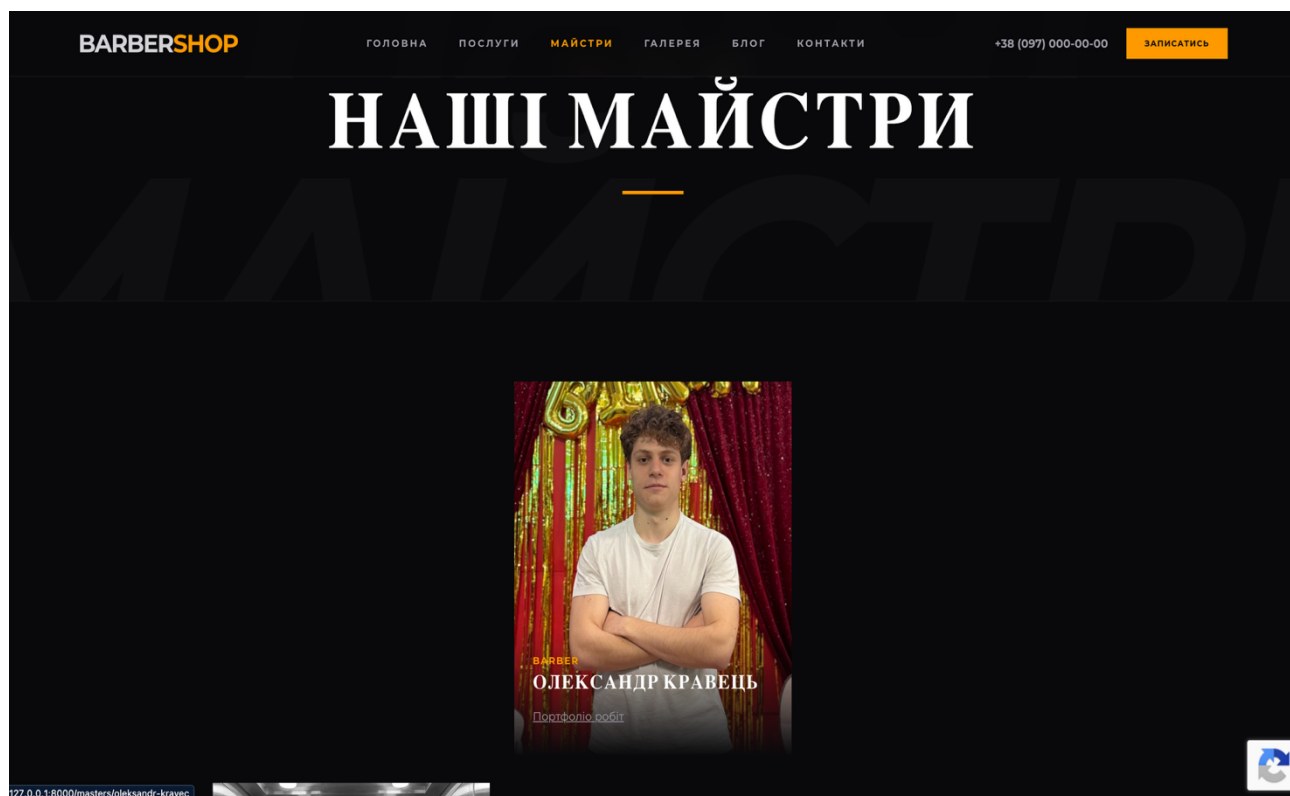


Рис. 3.4. Сторінка барберів

Сторінка профілю майстра розроблена для детального ознайомлення клієнта з кваліфікацією обраного фахівця. Робоча область логічно розділена на два функціональні сегменти: ліворуч розміщено компонент персональної інформації співробітника з його фотографією, посадою, ім'ям та кнопкою запису, яка передає дані до модуля бронювання для оформлення сеансу саме до цього спеціаліста. Права частина інтерфейсу реалізована як асиметрична модульна сітка портфоліо, у яку з бази даних веб-застосунку динамічно підвантажуються зображення робіт майстра. Це забезпечує користувачеві повноцінне уявлення про рівень кваліфікації фахівця та допомагає прийняти обґрунтоване рішення перед записом на послугу (рис. 3.5).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

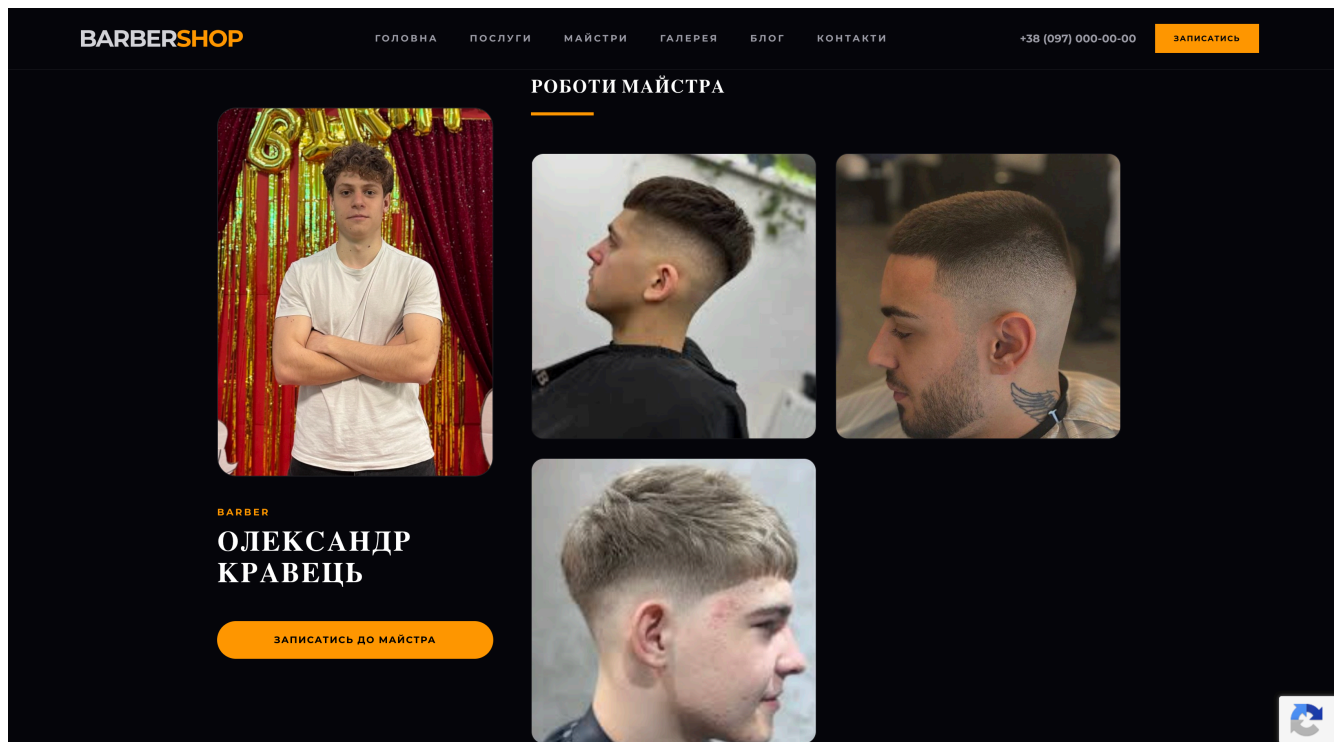


Рис. 3.5. Особиста сторінка барбера

Найголовніший елемент інтерфейсу користувача це безпосередньо форма запису на стрижку. Вона являє собою модальне вікно-степпер в правій частині екрану, яке складається з 4 кроків, що послідовно проводять користувача через увесь процес бронювання, завдяки чому взаємодія стає більш зрозумілою та логічно структурованою, а також зменшується навантаження під час вибору параметрів запису і заповнення необхідних даних, адже потрібно зробити тільки маленький крок. Такий підхід дозволяє поступово вводити користувача в процес оформлення запису, не перевантажуючи його великою кількістю інформації одночасно.

На першому кроці (рис. 3.6) користувач обирає майстра, до якого бажає записатись, при цьому відображається список доступних спеціалістів з фотографією з профілю, короткою інформацією про рівень професійності барбера та можливістю швидкого перегляду базових даних про кожного з них. Це дає змогу клієнту одразу сконцентруватись лише на виборі фахівця та перейти до наступного етапу лише тільки визначення з вибором.

Після того як клієнт зробив свій вибір, необхідно натиснути кнопку продовжити, щоб перейти до другого кроку.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 3.8. Третій крок форми запису

Четвертий крок призначений для збору персональних контактних даних клієнта та фінальної валідації замовлення перед його збереженням у базі даних. Візуальна структура форми включає обов'язкові поля для введення імені, прізвища та номера телефону з інтегрованою маскою форматування. Також передбачено опціональне поле для введення електронної адреси та текстовий блок для залишення спеціальних коментарів для майстра (рис. 3.9).

Рис. 3.9. Четвертий крок форми запису

Далі розглянемо інтерфейс модуля адміністратора (рис. 3.10). Він відіграє важливу роль організації та контролю всіх процесів діяльності барбершопу.

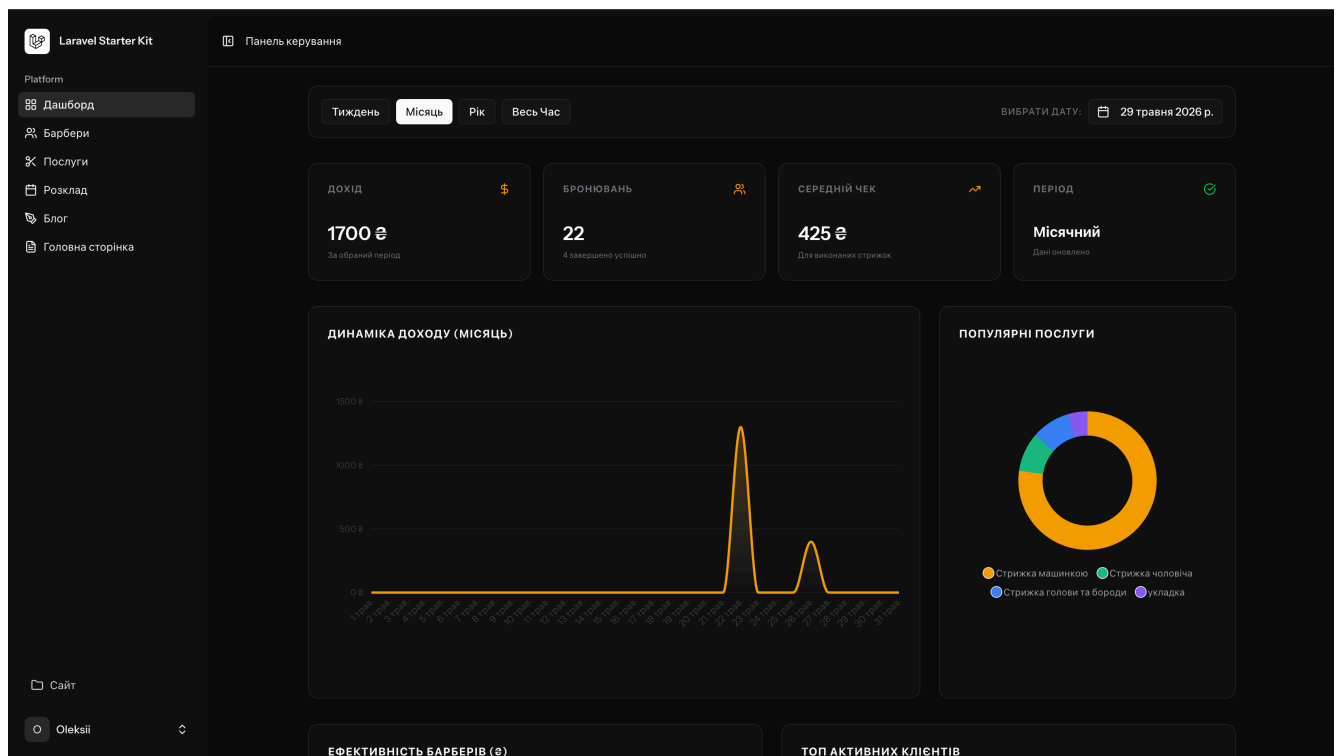


Рис. 3.10. Головна сторінка панелі адміністратора

На рисунку зображено головну сторінку панелі керування, яка забезпечує адміністратора інструментами для моніторингу та управління бізнес-процесами барбершопу. Ліва бічна панель виконує функцію головного навігаційного меню, надаючи прямий доступ до модулів управління персоналом, переліком послуг, робочим розкладом, контентом блогу та налаштуваннями клієнтської частини сайту. У нижній частині меню реалізовано блок керування обліковим записом поточного авторизованого адміністратора.

Верхня панель робочої області містить інструменти для фільтрації аналітичних даних. Користувач може застосувати швидкі фільтри для вибору звітного періоду, наприклад тижня чи місяця, або скористатися календарем для вказівки конкретної дати. Під панеллю фільтрації розташований блок ключових показників ефективності, який автоматично обчислює та виводить загальний дохід, кількість успішно завершених бронювань та розмір середнього чека за встановлений проміжок часу.

Основна частина екрана відведена під інтерактивні аналітичні інструменти. Графік динаміки доходу відображає фінансові надходження у розрізі окремих днів, що дозволяє відстежувати зміни прибутку та визначати найбільш завантажені дні. Поруч розміщено діаграму популярних послуг, яка збирає статистику замовлень та розраховує частку кожної виконаної послуги у загальному обсязі. У нижній частині сторінки передбачені модулі для збору статистики щодо продуктивності окремих майстрів.

Сторінка розкладу забезпечує централізований контроль за завантаженістю персоналу та розподілом фізичних робочих місць. Перехід до цього розділу здійснюється через активний пункт бічного навігаційного меню. У верхній частині робочої області інтегровано компонент вибору дати, який дозволяє адміністратору формувати розклад та переглядати базу записів на будь-який конкретний календарний день (рис. 3.11).

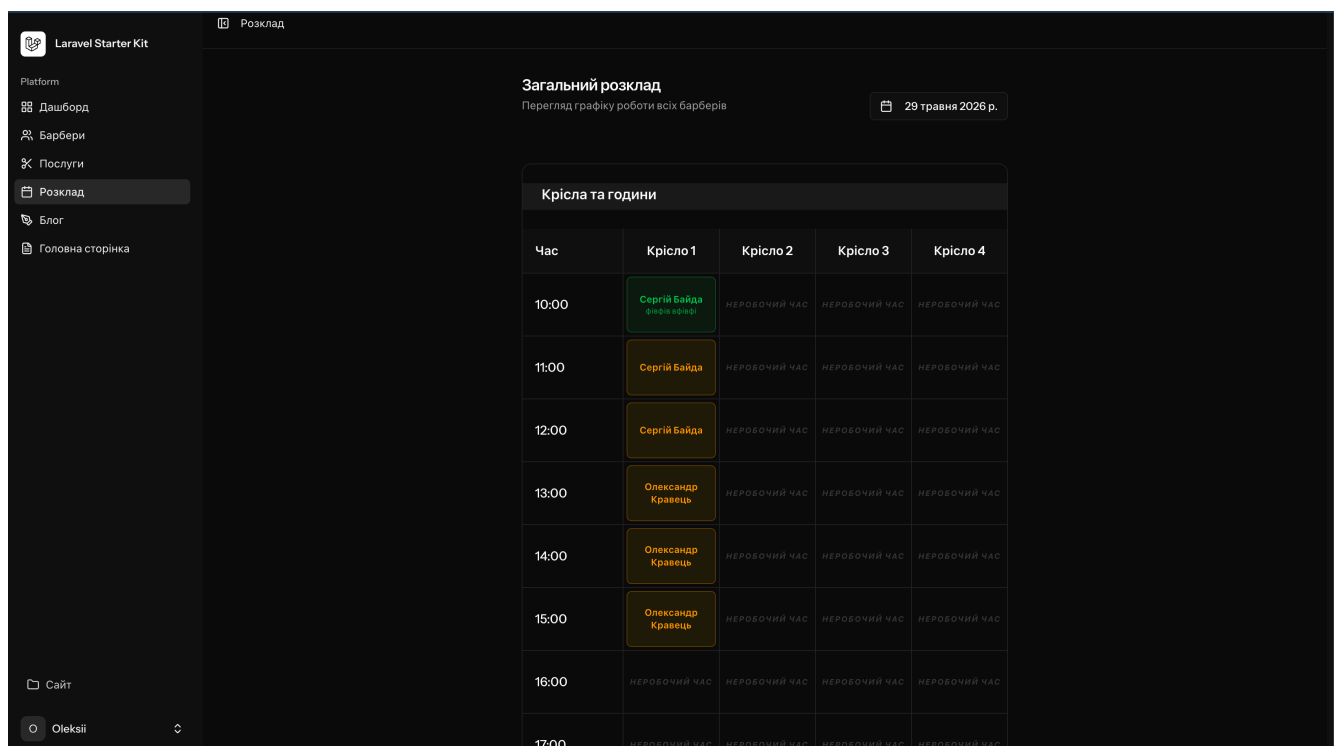


Рис. 3.11. Сторінка розкладу в адміністратора

Основним функціональним елементом сторінки є таблиця загального розкладу. Вертикальна вісь розбита на стандартизовані часові інтервали тривалістю в одну годину, що охоплюють робочий день закладу. Горизонтальна вісь

репрезентує доступні робочі зони, поділені за номерами крісел. Перетин цих параметрів утворює сітку комірок, які відображають фактичний статус кожного робочого місця у визначений проміжок часу.

Інформація в комірках генерується динамічно на основі підтверджених транзакцій бронювання з бази даних. Зайняті слоти містять дані про майстра, за яким закріплено відповідне крісло на вказаний час. Недоступні для бронювання проміжки автоматично блокуються системою та отримують статус неробочого часу. Такий підхід до візуалізації даних гарантує уникнення конфліктів у графіку, унеможливорює подвійний запис на одне робоче місце та надає адміністратору інструмент для оперативного керування потоком клієнтів.

Однією з найважливіших функцій доступних адміністратору є можливість керування обліковими записами барберів. Для цього виділена окрема сторінка на робочій панелі (рис. 3.12). На ній відображаються активні працівники, при наведенні курсору миші з'являється можливість побачити детальнішу інформацію.

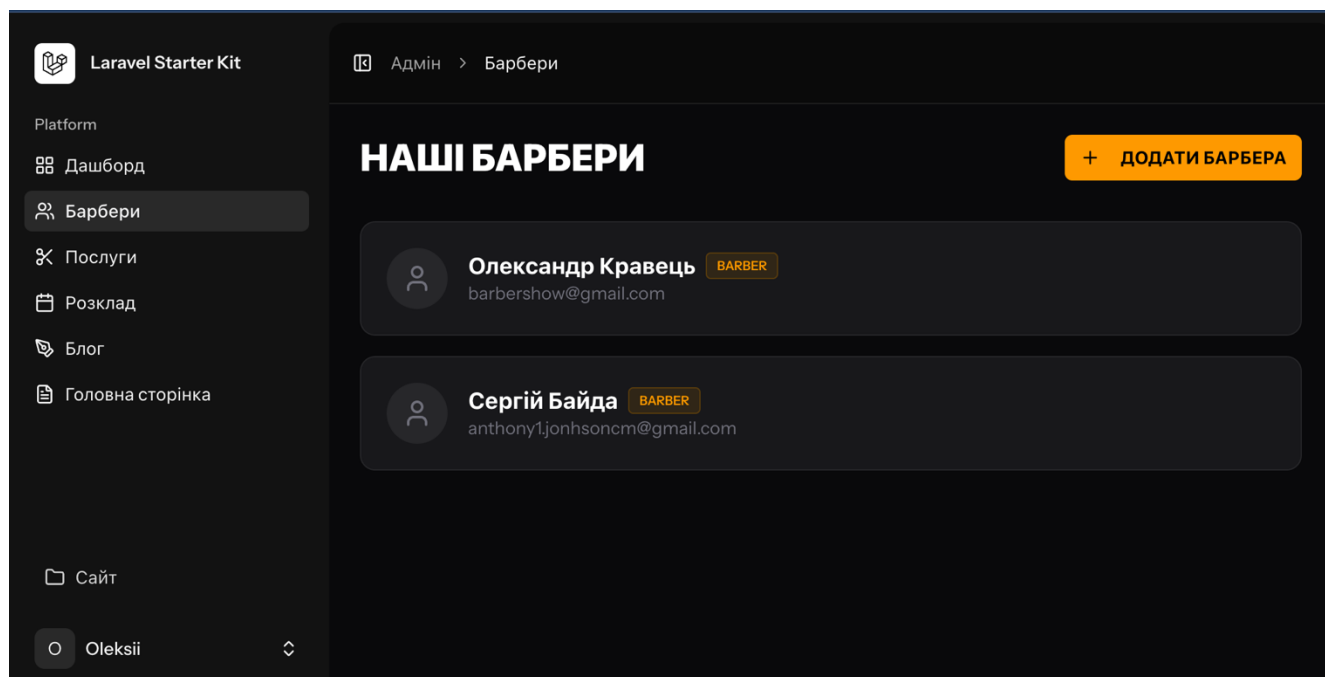


Рис. 3.12. Сторінка керування персоналом

На цій сторінці в панелі управління адміністратор системи отримує повноцінний інструмент для управління персоналом, маючи змогу не лише додавати нових спеціалістів до робочого графіку, але й вилучати за потреби.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

Для того щоб розширити штат та внести інформацію про нового фахівця, необхідно почати процес створення облікового запису. Відразу після натискання на кнопку Додати барбера, у робочій області екрана миттєво відкривається форма для реєстрації нового члена команди (рис. 3.13).

Рис. 3.13. Форма реєстрації нового барбера

Після реєстрації сторінка з усіма фахівцями оновиться, і новий член команди буде доданий в систему. Для входу за цією електронною адресою буде надано пароль, який адміністратор повинен передати новому барберу для входу (рис. 3.14)

Рис. 3.14. Успішна реєстрація барбера

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

Ще однією з функцій адміністратора є можливість керувати всіма видами послуг які надає барбершоп (рис.3.15). Ця функція дуже важлива, тому що залишає за адміністрацією закладу контроль за процесами які відбуваються в середині, і одночасно дає можливість майстрам з великого списку послуг обрати які їм підходять, та встановити за них власну ціну.

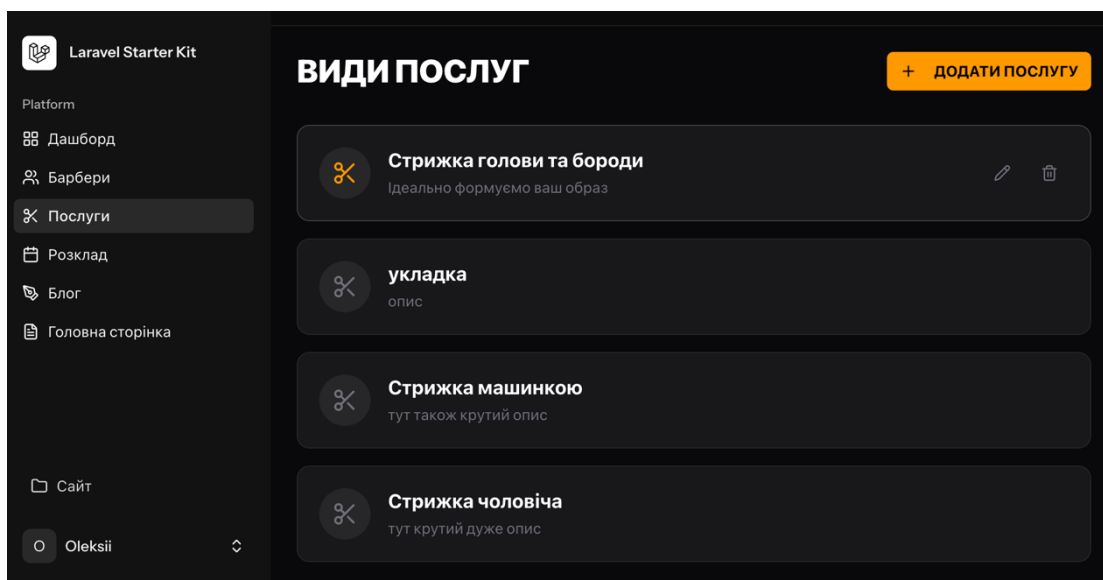


Рис. 3.15. Сторінка керування послугами

На сторінці додавання послуги (рис. 3.16) адміністратор вводить назву, тривалість (заданий діапазон від 30 хв до 2 годин) та опис. Діапазон часу сформований за класичною нормою для таких послуг, що допомагає зробити процес універсальнішим.

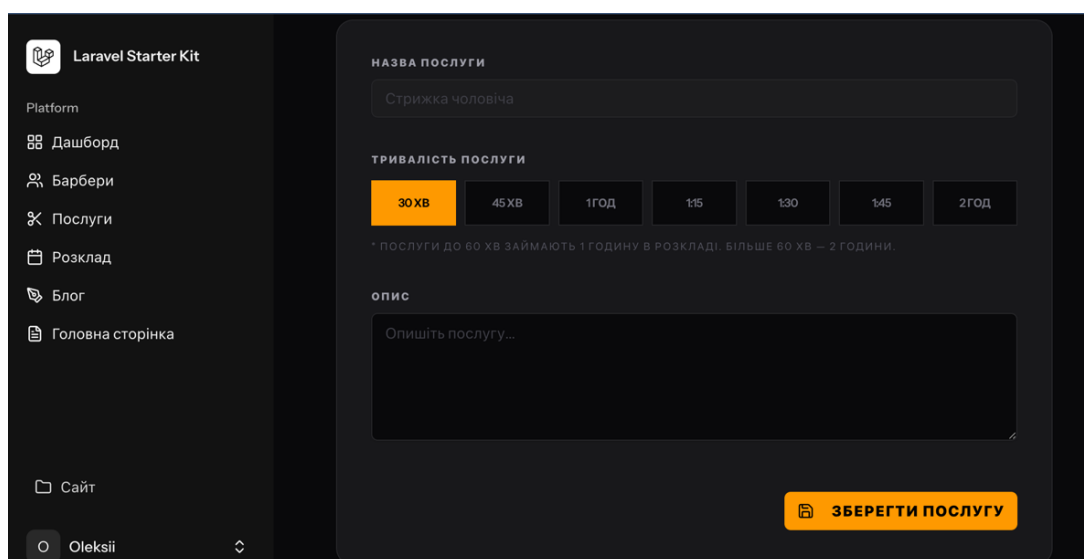


Рис. 3.16. Сторінка додавання послуги

Тепер варто детальніше розглянути третій, але не менш важливий модуль нашої системи, який безпосередньо стосується робочого кабінету самого барбера. Завдяки зручному та продуманому інтерфейсу, кожен майстер отримує достатній рівень автономії та може самостійно, без необхідності зайвих звернень до адміністратора, гнучко налаштовувати свій особистий профіль.

Перейшовши до розділу налаштувань профілю (рис. 3.17), майстер має змогу персоналізувати свою сторінку, задаючи власне фото. Вбудований інструментарій системи дозволяє не просто завантажити зображення, а й зручно відкадрувати його безпосередньо у вікні браузера, щоб портрет виглядав максимально професійно та відповідав стандартам платформи. Окрім візуальної складової, барбер заповнює ключову текстову інформацію: своє ім'я та прізвище, а також вказує поточну посаду чи статус у закладі (наприклад Top Barber). Для зручної ідентифікації майстра в мережі реалізована можливість задати унікальну персональну URL-адресу, за якою клієнти зможуть швидко знаходити його сторінку. Також передбачено спеціальне поле Про себе, де барбер може детально описати свій професійний досвід, підхід до стрижок чи унікальні навички, що допоможе створити привабливий імідж для потенційних клієнтів та підвищити рівень довіри до майстра.

Рис. 3.17. Сторінка налаштування профілю

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

Розділ Портфоліо, відіграє ключову роль у формуванні професійного іміджу та залученні нових клієнтів. Барбер отримує повний контроль над власною візуальною візитівкою. Основна мета цього функціоналу полягає у наданні майстру інструменту зручно керувати персональною галереєю виконаних робіт, а також гнучко налаштовувати порядок їх відображення для відвідувачів платформи, створюючи максимально привабливу презентацію своїх професійних навичок (рис. 3.18).

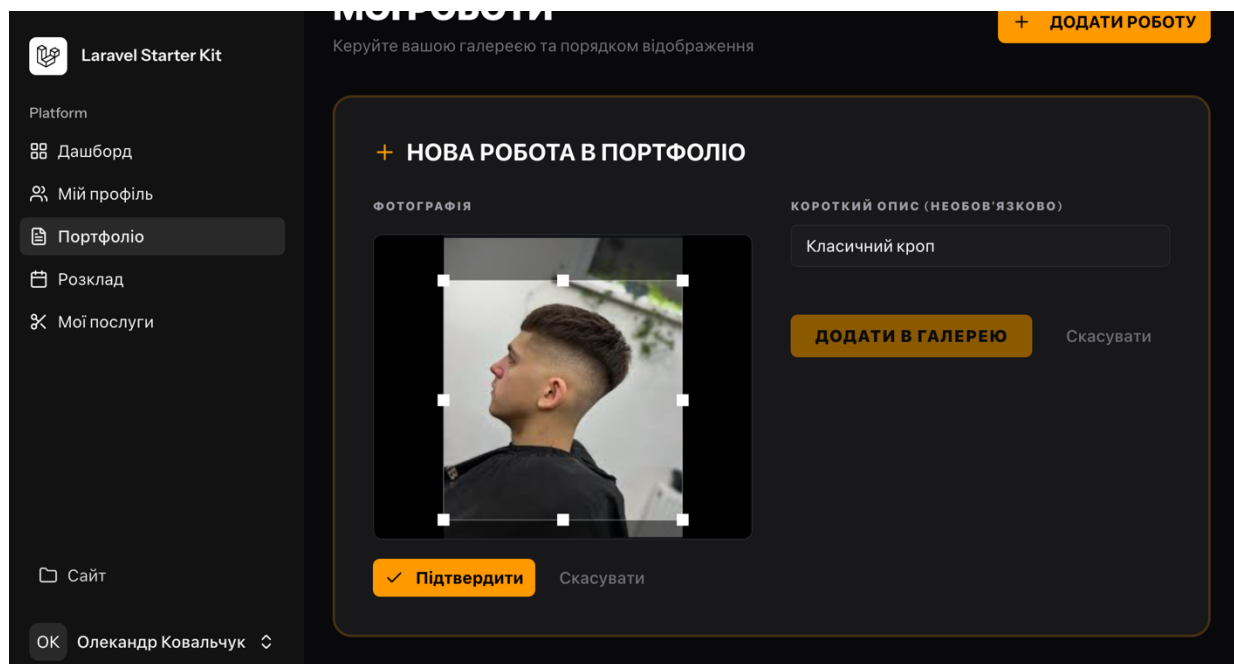


Рис. 3.18. Сторінка налаштування профілю

Найбільш інноваційним рішенням розробленої системи є інтерактивний процес планування робочого розкладу, який кардинально відрізняється від звичних табличних підходів. Перейшовши до розділу Розклад через навігаційне меню, користувач отримує доступ до візуально орієнтованого модуля бронювання робочих змін. Центральним елементом цього функціоналу є графічна інтерактивна схема-план самого приміщення барбершопу (рис. 3.19). На деталізованому кресленні головним об'єктом взаємодії виступають пронумеровані робочі крісла. Такий просторовий підхід дозволяє майстру не лише забронювати свої години, а й заздалегідь закріпити за собою конкретне фізичне робоче місце на зміну. Цей підхід повністю виключає можливість конфліктів та накладок, коли два барбери могли б випадково претендувати на одне й те саме крісло в один і той же час, а

найголовніше що такий варіант організації роботи є інтуїтивно зрозумілий всім учасниками процесу.

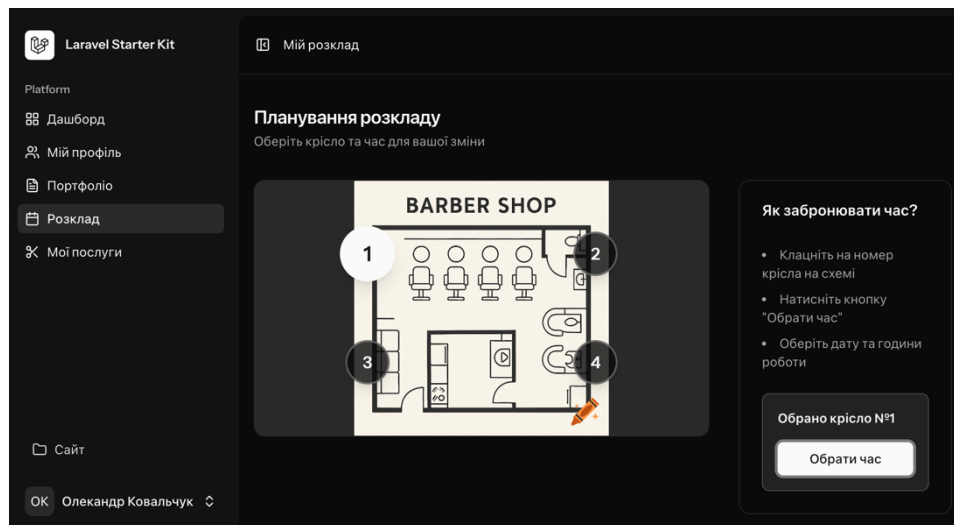


Рис. 3.19. Сторінка планування розкладу

Після закріплення за собою крісла відкривається модальне вікно (рис. 3.20) для вибору дня та робочих годин. Барбер просто клікає на бажані часові слоти, які зручно підсвічуються як обрані. Система автоматично запобігає накладкам: уже зайнятий іншими майстрами час стає неактивним і недоступним для вибору. Після виділення своєї зміни достатньо натиснути кнопку Підтвердити. Цей графік залишається гнучким - заброньовані години можна буде скасувати згодом, але виключно за умови, що на них ще не встиг записатися клієнт.

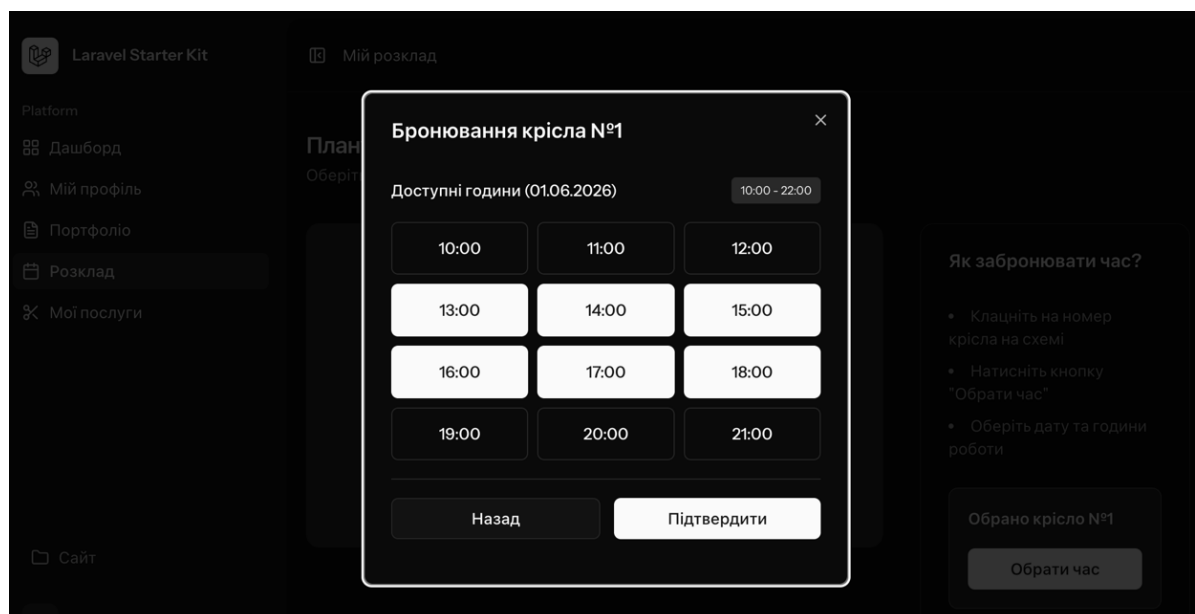


Рис. 3.20. Вікно вибору часу

На сторінці розкладу майстер отримує зручний та наочний огляд свого розкладу на обраний день (рис. 3.21).

Основну частину екрана займає таблиця погодинного графіку, де чітко відображаються як заплановані візити із деталями запису (ім'я клієнта, замовлена послуга, вартість), так і вільні часові слоти. Праворуч розташована корисна інформаційна панель, яка автоматично генерує статистику поточної зміни: вона відображає загальну кількість записів, встановлені робочі години (наприклад, з 10:00 до 22:00) та прогнозований фінансовий дохід. Також для зручності передбачена кнопка Вибрати дату, що дозволяє легко перемикатися між днями та планувати свою роботу наперед.

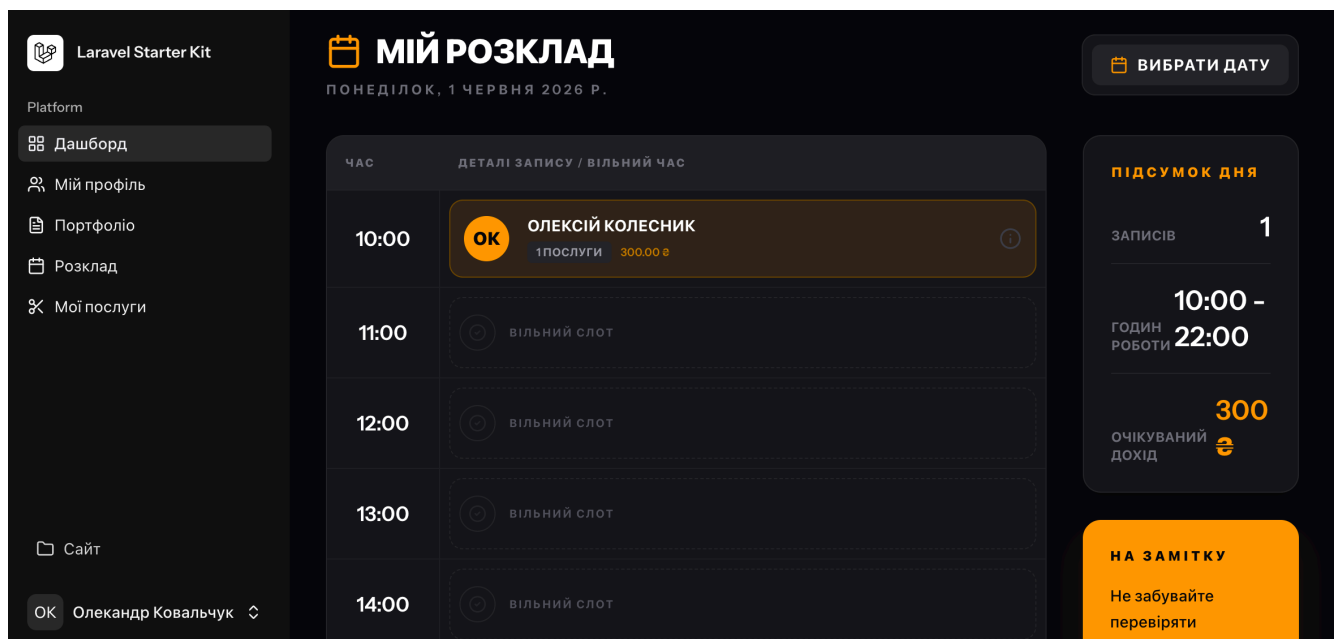


Рис. 3.21. Сторінка розкладу барбера

3.3. Тестування роботи програмного комплексу

Тестування веборієнтованої системи керування барбершопом є критично важливим етапом процесу розробки програмного забезпечення. Воно дозволяє перевірити правильність роботи всіх функціональних модулів, забезпечити стабільне виконання основних бізнес-процесів, пов'язаних із записом клієнтів, управлінням послугами та графіком роботи майстрів, а також оцінити зручність використання вебплатформи для всіх користувачів [2].

Тестування основних сценаріїв показані в табл. 3.1. Основну увагу приділено ручному функціональному тестуванню створення запису клієнтом, та управління ним зі сторони персоналу.

Таблиця. 3.1.

Тестування сценаріїв використання

Роль	Кроки виконання	Очікуваний результат
Клієнт	1. Користувач натискає кнопку «Записатися» 2. В модальному вікні обирає барбера 3. Обирає послугу 4. Користувач обирає день та час доступних слотів 5. Вводить ім'я, прізвище, номер телефону та електронну пошту, натискає підтвердити запис 6. З'являється повідомлення про успішне створення запису 7. Користувач отримує лист на пошту, з посиланням на сторінку керування записом	Створено запис на стрижку з усіма деталями
Адміністратор	1. Адміністратор переходить на вкладку Розклад 2. Обирає потрібний день 3. В таблиці позначаються робочі години та клієнти 4. При натисканні на клієнта з'являється модальне вікно з деталями запису 5. При натисканні на Скасувати запис скасовується, а слот стає доступним для користувачів	Адміністратор правильно бачить всі записи та може керувати ними

Барбер	1. На головній сторінці в панелі барбера обирати день 2. В таблиці відображаються вільні слоти та вже створені записи 3. При натисканні на створений запис з'являється модальне вікно з деталями запису 4. В правій частині в розділі підсумок дня відображається інформація з годинами зміни, кількістю запланованих клієнтів та очікуваним доходом	Барбер коректно бачить інформацію про свої робочі години, записи та клієнтів.
--------	---	---

Для перевірки працездатності системи керування барбершопом було застосовано комплексний підхід до тестування, який охоплював перевірку функціональних можливостей, інтерфейсу користувача та коректності взаємодії між окремими модулями системи. Основною метою тестування було підтвердження відповідності реалізованого функціоналу встановленим вимогам, а також виявлення можливих помилок і недоліків.

У процесі тестування перевірено основні функціональні модулі системи, а саме управління користувачами, управління послугами барбершопу, формування графіка роботи майстрів, бронювання записів клієнтами, ведення портфоліо барберів, обробку відгуків та адміністративну панель керування. Для кожного модуля було розроблено набір тестових сценаріїв, які охоплювали як стандартні варіанти використання, так і альтернативні та виняткові ситуації. Це дало можливість перевірити правильність обробки введених даних, коректність бізнес-логіки та стійкість системи до неочікуваних дій користувачів.

Окрему увагу було приділено тестуванню інтерфейсу користувача. Перевірялися коректність відображення сторінок у різних браузерях, зручність

навігації між розділами системи, адаптивність інтерфейсу для екранів різних розмірів, а також правильність роботи форм для введення даних, кнопок керування та елементів взаємодії. Проведене тестування дозволило переконатися, що інтерфейс є зрозумілим та зручним для адміністраторів, барберів і клієнтів [6].

Під час перевірки якості програмного продукту використовувалося дослідницьке тестування, яке належить до технік тестування, що ґрунтуються на досвіді тестувальника. На відміну від формалізованих тестових сценаріїв, цей підхід передбачає одночасне вивчення функціоналу системи, проєктування тест кейсів та їх перевірку, що дозволяє гнучко адаптувати процес тестування під поточну поведінку застосунку [4]. Такий метод дозволив виявити низку потенційних проблем, пов'язаних із нестандартними діями користувачів, обробкою неочікуваних комбінацій даних та взаємодією між окремими компонентами системи.

Техніки тестування, що базуються на досвіді, використовувалися як основний інструмент тестування, та дали змогу оцінити поведінку системи в умовах реального використання, імітуючи різні сценарії взаємодії кінцевих користувачів із продуктом.

Завдяки поєднанню структурованих тестових сценаріїв і дослідницького підходу було досягнуто достатнього рівня покриття функціональних можливостей системи, а також виявлено та усунуто низку дрібних недоліків інтерфейсу і логіки роботи. Це в цілому підвищило стабільність і надійність системи перед впровадженням у робоче середовище.

Висновки до третього розділу

У межах розділу було розглянуто порядок встановлення та налаштування програмного забезпечення, що демонструє можливість розгортання системи в робочому середовищі. Також розглянуто особливості реалізації інтерфейсу користувача та механізми взаємодії з даними, які забезпечують зручне керування послугами, записами клієнтів, графіком роботи майстрів та іншими складовими діяльності барбершопу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

Було розглянуто інтерфейс системи та порядок роботи з ним, включаючи основні елементи навігації, структуру сторінок та логіку взаємодії користувача з різними функціональними модулями. Особливу увагу приділено зручності використання інтерфейсу та логічній послідовності виконання основних дій у програм.

Для підтвердження якості розробленого програмного продукту було проведено комплексне тестування системи. Перевірено функціональність окремих модулів за допомогою тестових сценаріїв, протестовано користувацький інтерфейс, а також застосовано дослідницьке тестування для виявлення можливих помилок під час виконання дій користувачів. Результати тестування засвідчили коректність роботи реалізованого функціоналу, стійкість системи до помилкових вхідних даних та відповідність програмного продукту поставленим вимогам.

Отримані результати підтверджують, що розроблена інформаційна система забезпечує ефективну автоматизацію основних процесів діяльності барбершопу, спрощує управління записами клієнтів, послугами та персоналом, а також створює зручні умови для взаємодії всіх категорій користувачів із системою.

У результаті проведених робіт система продемонструвала готовність до використання в умовах реального бізнесу, забезпечуючи ефективне управління записами клієнтів, розкладом майстрів та обліком наданих послуг. Тестування підтвердило коректну роботу всіх основних функцій відповідно до заданих вимог, що дає підстави рекомендувати систему до впровадження в практичну діяльність реального закладу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було здійснено комплексне дослідження предметної області та розроблено веб-систему для управління барбершопом, орієнтовану на автоматизацію ключових бізнес-процесів закладу сфери послуг. У межах роботи проаналізовано сучасні підходи до організації подібних систем, визначено їхні основні недоліки та сформульовано вимоги до функціоналу, необхідного для ефективного управління діяльністю барбершопу в умовах цифрової трансформації ринку. Робота охоплює повний цикл створення програмного продукту, від аналізу предметної області до реалізації та тестування готової системи.

У першому розділі було розглянуто предметну область дослідження та виявлено основну проблематику, пов'язану з недостатнім рівнем автоматизації процесів у барбершопках. Було обґрунтовано необхідність створення єдиної інформаційної системи, яка забезпечує взаємодію між клієнтами, майстрами та адміністрацією. Окрему увагу приділено актуальності цифровізації сфери послуг, підвищенню рівня сервісу та зменшенню впливу людського фактору на організаційні процеси.

Також в цьому розділі було проведено аналіз існуючих аналогів програмних продуктів. У результаті дослідження встановлено, що більшість рішень мають обмежену функціональність і не забезпечують комплексного підходу до управління діяльністю барбершопу. Це дозволило сформулювати вимоги до майбутньої системи та визначити напрямки її функціонального розширення.

Окремо було розглянуто питання архітектури програмного забезпечення. Обґрунтовано вибір багаторівневої архітектури з поділом на клієнтську та серверну частини, що забезпечує гнучкість, масштабованість і зручність подальшого розвитку системи. Крім того, було визначено основні принципи взаємодії між модулями та обробки даних.

У другому розділі було розглянуто процес проектування та реалізації програмного забезпечення. Зокрема, було визначено варіанти використання

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

системи та сформовано функціональні вимоги для різних категорій користувачів. Було проаналізовано основні сценарії взаємодії з системою, включаючи процес онлайн-запису клієнтів, управління розкладом майстрів, адміністрування послуг та роботу з інформаційним наповненням сайту.

Ще в цьому розділі було спроектовано структуру бази даних системи. Визначено основні сутності та зв'язки між ними, що дозволило забезпечити цілісність та логічну організацію даних. База даних охоплює інформацію про користувачів, записи, послуги, графіки роботи, фінансові операції та інші ключові елементи системи.

У третьому розділі було розглянуто інтерфейс користувача, порядок встановлення та налаштування програмного продукту, а також проведено тестування системи. Особливу увагу приділено зручності інтерфейсу та відповідності принципам сучасного дизайну. У межах тестування перевірено функціональність основних модулів, стабільність роботи системи та коректність взаємодії між компонентами, що підтвердило готовність програмного продукту до використання.

У результаті виконання роботи було розроблено повноцінну веб-систему для управління барбершопом, яка забезпечує автоматизацію ключових бізнес-процесів, підвищує ефективність управління та сприяє покращенню якості обслуговування клієнтів. Розроблена система є масштабованою, зручною у використанні та може бути впроваджена в реальних умовах діяльності.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бази даних: проектування та реалізація/ Г. С. Погромська, Н.А. Махровська.
– Місто: Видавництво, 2019. – 183 с. URL:
http://dspace.mdu.edu.ua/jspui/bitstream/123456789/482/1/%D0%9F%D0%BE%D0%B3%D1%80%D0%BE%D0%BC%D1%81%D1%8C%D0%BA%D0%B0%2C%20%D0%9C%D0%B0%D1%85%D1%80%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B0_%D0%91%D0%B0%D0%B7%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%82%D0%B0%20%D1%80%D0%B5%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F.pdf
2. Жухур В. Основні підходи до ефективного тестування ПЗ. DOU. 2024. URL:
<https://dou.ua/forums/topic/48845/> (дата звернення: 01.06.2026).
3. Кормщикова О. О. Вплив цифровізації на ефективність управління персоналом в сучасних умовах. Моделювання та прогнозування економічних процесів. 2024. Т. 1, № 1. URL:
<https://mreproc.fmm.kpi.ua/article/view/298316/291085> (дата звернення: 01.06.2026).
4. Свірідова Т., Марікуца У. Дослідницьке тестування як засіб підвищення якості аплікації. Комп'ютерні науки та інформаційні технології. 2013. № 751. С. 365–368. URL:
<https://science.lpnu.ua/sites/default/files/journal-paper/2024/feb/33861/vis751komp-nauky-365-368.pdf> (дата звернення: 01.06.2026).
5. Сухий О. Л. Алгоритми пошуку в інформаційних системах: методичні рекомендації, К., 2015. 72 с. URL:
https://lib.iitta.gov.ua/704951/1/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4.%20%D1%80%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%86%D1%96%D1%97_%D0%A1%D1%83%D1%85%D0%B8%D0%B9%202015.pdf (дата звернення: 15.05.2026).
6. Тестування інтерфейсу користувача: стандарти, чек-лист та кроки. URL:
<https://wezom.com.ua/ua/blog/testing-ui-user-interface> (дата звернення: 01.06.2026).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

7. Diagram Tutorial URL: <https://www.lucidchart.com/pages/uml-component-diagram>. (дата звернення: 23.03.2026).
8. Docker Documentation. URL: <https://docs.docker.com/> (дата звернення: 26.03.2026).
9. Exploring Advanced Design Patterns in Vue.js. LogRocket. URL: <https://blog.logrocket.com/exploring-advanced-design-patterns-vue-js/> (дата звернення: 01.06.2026).
10. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.
11. Fowler, M.. Refactoring: Improving the Design of Existing Code. Boston: Addison-Wesley, 2018, 448 p. URL: <https://dl.ebooksworld.ir/motoman/Refactoring.Improving.the.Design.of.Existing.Code.2nd.edition.www.EBooksWorld.ir.pdf>
12. Inertia.js Documentation. URL: <https://inertiajs.com/docs/v3/getting-started> (дата звернення: 01.06.2026).
13. Jelvix. Choosing between MySQL vs PostgreSQL vs SQL Server. URL: <https://jelvix.com/blog/mysql-postgresql-sql-server> (дата звернення: 25.05.2026).
14. Laravel Documentation URL: <https://laravel.com/docs/12.x/installation> (дата звернення: 01.06.2026).
15. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval Cambridge University Press, 2008. URL: <https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf> (дата звернення: 25.05.2025).
16. Martin R. C. Clean Agile: Back to Basics. Boston: Prentice Hall, 2019. 240 p.
17. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Boston: Prentice Hall, 2008. 464 p.
18. Martin, R. C.. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Upper Saddle River: Prentice Hall, 2017, 432 p.
19. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 01.06.2026).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

20. PostgreSQL in Docker: A Step-by-Step Guide for Beginners. DataCamp. URL: <https://www.datacamp.com/tutorial/postgresql-docker> (дата звернення: 01.06.2026).
21. reCAPTCHA URL: <https://developers.google.com/recaptcha> (дата звернення: 01.06.2026).
22. Ries, E.. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Business, 2011, 336 p. URL: <https://ia800509.us.archive.org/7/items/TheLeanStartupErickRies/The%20Lean%20Startup%20-%20Erick%20Ries.pdf>
23. Shadcn-vue Documentation URL: <https://shadcn-vue.com/> (дата звернення: 20.04.2026).
24. Tailwind CSS Documentation URL: <https://tailwindcss.com/docs> (дата звернення: 20.04.2026).
25. The Ultimate Barbershop Marketing Strategy: A 2026 Guide to Client Acquisition. URL: <https://biz.booksy.com/en-us/blog/barbershop-marketing> (дата звернення: 01.06.2026).
26. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 01.06.2026).
27. Understanding Laravel Authentication: Best Practices and Tips. URL: <https://medium.com/@aiman.asfia/understanding-laravel-authentication-best-practices-and-tips-cf00bcb894a4> (дата звернення: 01.06.2026).
28. Vite. Next Generation Frontend Tooling. URL: <https://vitejs.dev/guide/>. (дата звернення: 20.03.2026).
29. Vue.js Documentation. URL: <https://vuejs.org/guide/introduction.html> (дата звернення: 01.06.2026).
30. What is Monolithic Architecture? IBM. URL: <https://www.ibm.com/think/topics/monolithic-architecture> (дата звернення: 01.06.2026).

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг роутингу:

<?php

```

use App\Http\Controllers\Clients\BlogController;
use App\Http\Controllers\Clients\ContactsController;
use App\Http\Controllers\Clients\GalleryController;
use App\Http\Controllers\Clients\HomeController;
use App\Http\Controllers\Clients\MastersController;
use App\Http\Controllers\Clients\ServicesController;
use App\Http\Controllers\TestController;
use Illuminate\Support\Facades\Route;
use Laravel\Fortify\Features;

```

```
Route::get('/test', [TestController::class, 'index'])->name('test');
```

```

Route::prefix('api/booking')->name('api.booking.')->group(function () {
    Route::get('/barbers', [\App\Http\Controllers\Api\BookingController::class, 'getBarbers'])->name('barbers');
    Route::get('/services', [\App\Http\Controllers\Api\BookingController::class, 'getServices'])->name('services');
    Route::get('/slots', [\App\Http\Controllers\Api\BookingController::class, 'getAvailableSlots'])->name('slots');
    Route::get('/nearest-slot', [\App\Http\Controllers\Api\BookingController::class, 'getNearestSlot'])->
>name('nearest-slot');
    Route::post('/store', [\App\Http\Controllers\Api\BookingController::class, 'store'])->name('store');
});

```

```

Route::middleware(['auth'])->group(function () {
    Route::post('/upload/image', [\App\Http\Controllers\Admin\UploadController::class, 'image'])->
>name('upload.image');
});

```

```

Route::name('client.')->group(function () {
    Route::get('/', [HomeController::class, 'index'])->name('home');
    Route::get('/services', [ServicesController::class, 'index'])->name('services');
    Route::get('/masters', [MastersController::class, 'index'])->name('masters');
    Route::get('/masters/{slug}', [MastersController::class, 'show'])->name('masters.show');
    Route::get('/gallery', [GalleryController::class, 'index'])->name('gallery');
    Route::get('/contacts', [ContactsController::class, 'index'])->name('contacts');
    Route::get('/blog', [BlogController::class, 'index'])->name('blog');
    Route::get('/blog/{slug}', [BlogController::class, 'show'])->name('blog.show');
});

```

```

Route::get('/bookings/{booking}/cancel', [\App\Http\Controllers\Clients\BookingCancellationController::class,
'show'])
    ->name('bookings.cancel.show')
    ->middleware('signed');

```

```

Route::post('/bookings/{booking}/cancel', [\App\Http\Controllers\Clients\BookingCancellationController::class,
'cancel'])
    ->name('bookings.cancel.confirm')
    ->middleware('signed');

```

```

Route::middleware(['auth', 'role:barber'])->prefix('barber')->name('barber.')->group(function () {
    Route::get('/', [\App\Http\Controllers\Barber\DashboardController::class, 'index'])->name('dashboard');

```

```

Route::get('/profile', [\App\Http\Controllers\Barber\ProfileController::class, 'edit'])->name('profile.edit');
Route::put('/profile', [\App\Http\Controllers\Barber\ProfileController::class, 'update'])->name('profile.update');

Route::get('/portfolio', [\App\Http\Controllers\Barber\PortfolioController::class, 'index'])->
>name('portfolio.index');
Route::post('/portfolio', [\App\Http\Controllers\Barber\PortfolioController::class, 'store'])->
>name('portfolio.store');
Route::patch('/portfolio/order', [\App\Http\Controllers\Barber\PortfolioController::class, 'updateOrder'])->
>name('portfolio.order');
Route::delete('/portfolio/{portfolio}', [\App\Http\Controllers\Barber\PortfolioController::class, 'destroy'])->
>name('portfolio.destroy');

Route::get('/schedule', [\App\Http\Controllers\Barber\ScheduleController::class, 'index'])->
>name('schedule.index');
Route::post('/schedule', [\App\Http\Controllers\Barber\ScheduleController::class, 'store'])->
>name('schedule.store');
Route::delete('/schedule/slot', [\App\Http\Controllers\Barber\ScheduleController::class, 'cancelSlot'])->
>name('schedule.cancel-slot');
Route::delete('/schedule/{schedule}', [\App\Http\Controllers\Barber\ScheduleController::class, 'destroy'])->
>name('schedule.destroy');

Route::get('/my-services', [\App\Http\Controllers\Barber\ServiceController::class, 'index'])->
>name('services.index');
Route::post('/my-services', [\App\Http\Controllers\Barber\ServiceController::class, 'update'])->
>name('services.update');
});

```

<?php

```

use App\Http\Controllers\Admin\HomeController;
use App\Http\Controllers\Admin\PageController;
use App\Http\Controllers\Admin\UploadController;
use App\Http\Controllers\Admin\PostController;
use App\Http\Controllers\Admin\BarberController;
use App\Http\Controllers\Admin\ScheduleController;
use App\Http\Controllers\Admin\ServiceController;
use Illuminate\Support\Facades\Route;

```

```

Route::get('/', [HomeController::class, 'index'])->name('home');
Route::post('/upload/image', [UploadController::class, 'image'])->name('upload.image');

```

```

Route::get('/schedule', [ScheduleController::class, 'index'])->name('schedule.index');
Route::post('/bookings/{booking}/cancel', [ScheduleController::class, 'cancel'])->name('bookings.cancel');
Route::post('/bookings/{booking}/complete', [ScheduleController::class, 'complete'])->
>name('bookings.complete');

```

```

Route::resource('posts', PostController::class)->names('posts');
Route::patch('posts/{post}/toggle', [PostController::class, 'toggle'])->name('posts.toggle');

```

```

Route::resource('barbers', BarberController::class)->names('barbers')->except(['show', 'edit', 'update']);

```

```

Route::resource('services', ServiceController::class)->names('services');

```

```

Route::prefix('pages')->name('pages.')->group(function () {
    Route::get('/home', [PageController::class, 'editHome'])->name('home');
});

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 90 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

Route::put('/home', [PageController::class, 'updateHome'])->name('home.update');
});

<?php

use App\Http\Controllers\Settings\PasswordController;
use App\Http\Controllers\Settings\ProfileController;
use App\Http\Controllers\Settings\TwoFactorAuthenticationController;
use Illuminate\Support\Facades\Route;

Route::middleware(['auth'])->group(function () {
    Route::redirect('settings', '/settings/profile');

    Route::get('settings/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('settings/profile', [ProfileController::class, 'update'])->name('profile.update');
});

Route::middleware(['auth', 'verified'])->group(function () {
    Route::delete('settings/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');

    Route::get('settings/password', [PasswordController::class, 'edit'])->name('user-password.edit');

    Route::put('settings/password', [PasswordController::class, 'update'])
        ->middleware('throttle:6,1')
        ->name('user-password.update');

    Route::inertia('settings/appearance', 'settings/Appearance')->name('appearance.edit');

    Route::get('settings/two-factor', [TwoFactorAuthenticationController::class, 'show'])
        ->name('two-factor.show');
});

```

Лістинг контролерів:

```

<?php

namespace App\Http\Controllers\Admin;

use App\Enums\UserRole;
use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
use Inertia\Inertia;

class BarberController extends Controller
{
    public function index()
    {
        $barbers = User::where('role', UserRole::BARBER)
            ->withCount(['bookings as future_bookings_count' => function ($query) {
                $query->where('status', '!=', 'cancelled')
                    ->where('start_at', '>=', now());
            }])
            ->latest()
            ->get();
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 91 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

return Inertia::render('Admin/Barbers/Index', [
    'barbers' => $barbers,
]);
}

public function create()
{
    return Inertia::render('Admin/Barbers/Create');
}

public function store(Request $request)
{
    $validated = $request->validate([
        'first_name' => 'required|string|max:255',
        'last_name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
    ]);

    $password = Str::random(12);

    User::create([
        'first_name' => $validated['first_name'],
        'last_name' => $validated['last_name'],
        'name' => $validated['first_name'] . ' ' . $validated['last_name'],
        'email' => $validated['email'],
        'password' => Hash::make($password),
        'role' => UserRole::BARBER,
    ]);

    return redirect()->route('admin.barbers.index')->with('password', $password);
}

public function destroy(User $barber)
{
    if ($barber->role !== UserRole::BARBER) {
        abort(403);
    }

    $futureBookingsCount = \App\Models\Booking::where('user_id', $barber->id)
        ->where('status', '!=', 'cancelled')
        ->where('start_at', '>=', now())
        ->count();

    if ($futureBookingsCount > 0) {
        return back()->with('delete_error', [
            'message' => "Неможливо видалити барбера, оскільки у нього є {$futureBookingsCount} активних записів у майбутньому.",
            'barber_id' => $barber->id
        ]);
    }

    $barber->schedules()->where('date', '>=', now()->toDateString())->delete();
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 92 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

$barber->delete();

return back()->with('success', 'Барбера видалено та майбутній розклад очищено');
}
}

```

<?php

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use App\Models\Booking;
use App\Models\Service;
use App\Models\User;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Inertia\Inertia;

```

```

class HomeController extends Controller
{
    public function index(Request $request)
    {
        $period = $request->input('period', 'month');
        $dateParam = $request->input('date', now()->toDateString());
        $date = Carbon::parse($dateParam);

        $startAt = now()->startOfMonth();
        $endAt = now()->endOfMonth();
        $groupBy = "start_at::date";

        switch ($period) {
            case 'week':
                $startAt = $date->copy()->startOfWeek();
                $endAt = $date->copy()->endOfWeek();
                $groupBy = "start_at::date";
                break;
            case 'month':
                $startAt = $date->copy()->startOfMonth();
                $endAt = $date->copy()->endOfMonth();
                $groupBy = "start_at::date";
                break;
            case 'year':
                $startAt = $date->copy()->startOfYear();
                $endAt = $date->copy()->endOfYear();
                $groupBy = "TO_CHAR(start_at, 'YYYY-MM')";
                break;
            case 'all':
                $startAt = Carbon::parse('2020-01-01');
                $endAt = now()->addYears(10);
                $groupBy = "TO_CHAR(start_at, 'YYYY-MM')";
                break;
        }
    }
}

```

```
$totalRevenue = Booking::where('status', 'completed')
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 93 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

->whereBetween('start_at', [$startAt, $endAt])
->sum('total_price');

$totalBookings = Booking::whereBetween('start_at', [$startAt, $endAt])->count();
$completedBookings = Booking::where('status', 'completed')
->whereBetween('start_at', [$startAt, $endAt])
->count();

$averageCheck = $completedBookings > 0 ? $totalRevenue / $completedBookings : 0;

$rawRevenue = Booking::where('status', 'completed')
->whereBetween('start_at', [$startAt, $endAt])
->select(
    DB::raw("$groupBy as label"),
    DB::raw('SUM(total_price) as total')
)
->groupBy(DB::raw("$groupBy"))
->orderBy('label')
->get()
->pluck('total', 'label');

$revenueByPeriod = [];
$current = $startAt->copy();

while ($current <= $endAt) {
    $label = "";
    if ($period === 'week' || $period === 'month') {
        $label = $current->toDateString();
        $current->addDay();
    } else {
        $label = $current->format('Y-m');
        $current->addMonth();
    }

    $revenueByPeriod[] = [
        'label' => $label,
        'total' => $rawRevenue[$label] ?? 0,
    ];

    if ($period === 'all' && count($revenueByPeriod) > 100) break;
}

$popularServices = DB::table('booking_service')
->join('bookings', 'booking_service.booking_id', '=', 'bookings.id')
->join('services', 'booking_service.service_id', '=', 'services.id')
->whereBetween('bookings.start_at', [$startAt, $endAt])
->select('services.name', DB::raw('count(*) as total'))
->groupBy('services.id', 'services.name')
->orderByDesc('total')
->limit(5)
->get();

```

```

$barbersPerformance = Booking::where('status', 'completed')
->whereBetween('start_at', [$startAt, $endAt])
->join('users', 'bookings.user_id', '=', 'users.id')
->select('users.name', DB::raw('SUM(total_price) as revenue'), DB::raw('count(*) as bookings_count'))
->groupBy('users.id', 'users.name')
->orderByDesc('revenue')
->get();

return Inertia::render('Admin/Home', [
    'stats' => [
        'kpi' => [
            'total_revenue' => round($totalRevenue, 2),
            'total_bookings' => $totalBookings,
            'completed_bookings' => $completedBookings,
            'average_check' => round($averageCheck, 2),
        ],
        'revenue_chart' => $revenueByPeriod,
        'services_chart' => $popularServices,
        'barbers_chart' => $barbersPerformance,
    ],
    'filters' => [
        'period' => $period,
        'date' => $dateParam,
    ]
]);
}
}

```

<?php

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use App\Models\Page;
use Illuminate\Http\Request;
use Inertia\Inertia;

```

```

class PageController extends Controller
{
    public function editHome()
    {
        $page = Page::firstOrCreate(
            ['slug' => 'home'],
            ['content' => $this->getDefaultContent()]
        );

        return Inertia::render('Admin/Pages/HomeEdit', [
            'content' => $page->content
        ]);
    }

    public function updateHome(Request $request)
    {
        $validated = $request->validate([
            'hero' => 'required|array',
            'features' => 'required|array',

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 95 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        'about' => 'required|array',
        'cta' => 'required|array',
    ]);

    $page = Page::where('slug', 'home')->first();
    $page->update([
        'content' => $validated
    ]);

    return back()->with('status', 'Контент успішно оновлено!');
}

private function getDefaultContent()
{
    return [
        'hero' => [
            'uptitle' => 'Establish 2026',
            'title' => 'Більше ніж <br><span class="text-amber-500">просто стрижка</span>',
            'description' => 'Ми створюємо образи, що додають впевненості. Традиційне гоління, сучасні стрижки та справжня чоловіча атмосфера у самому серці міста.',
            'button_text' => 'Записатися онлайн',
        ],
        'features' => [
            ['title' => 'Професіоналізм', 'description' => 'Наші майстри володіють класичними та сучасними техніками на найвищому рівні.'],
            ['title' => 'Швидкість', 'description' => 'Ми цінуємо ваш час, тому працюємо оперативно без втрати якості результату.'],
            ['title' => 'Атмосфера', 'description' => 'Насолоджуйтесь якісними напоями та приємним спілкуванням під час стрижки.'],
        ],
        'about' => [
            'uptitle' => 'Наша історія',
            'title' => 'Мистецтво деталі у <br><span class="text-amber-500">кожному русі</span>',
            'experience_years' => '10',
            'quote' => 'Барбершоп це не просто місце для стрижки. Це чоловічий клуб, де традиції зустрічаються з сучасним стилем.',
            'description' => 'Кожен наш майстер це фанат своєї справи. Ми постійно вдосконалюємося, відвідуємо майстер-класи світового рівня, щоб ви отримували сервіс, на який заслуговуєте.',
        ],
        'cta' => [
            'title' => 'Готовий <br><span class="text-amber-500">змінити свій стиль?</span>',
            'button_text' => 'Забронювати час',
        ]
    ];
}
}

```

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Models\Post;

use Illuminate\Http\Request;

use Illuminate\Support\Str;

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 96 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

use Inertia\Inertia;

class PostController extends Controller
{
    public function index()
    {
        return Inertia::render('Admin/Posts/Index', [
            'posts' => Post::latest()->get()
        ]);
    }

    public function create()
    {
        return Inertia::render('Admin/Posts/Create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'title' => 'required|string|max:255',
            'excerpt' => 'nullable|string',
            'content' => 'required|string',
            'image_path' => 'nullable|string',
            'author' => 'nullable|string',
            'is_published' => 'boolean',
        ]);

        $validated['slug'] = Str::slug($validated['title']);

        Post::create($validated);

        return redirect()->route('admin.posts.index')->with('success', 'Пост успішно створено!');
    }

    public function edit(Post $post)
    {
        return Inertia::render('Admin/Posts/Edit', [
            'post' => $post
        ]);
    }

    public function update(Request $request, Post $post)
    {
        $validated = $request->validate([
            'title' => 'required|string|max:255',
            'excerpt' => 'nullable|string',
            'content' => 'required|string',
            'image_path' => 'nullable|string',
            'author' => 'nullable|string',
            'is_published' => 'boolean',
        ]);

        if ($post->title !== $validated['title']) {
            $validated['slug'] = Str::slug($validated['title']);
        }
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 97 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

$post->update($validated);

return redirect()->route('admin.posts.index')->with('success', 'Пост оновлено!');
}

public function destroy(Post $post)
{
    $post->delete();
    return back()->with('success', 'Пост видалено!');
}

public function toggle(Post $post)
{
    $post->update([
        'is_published' => !$post->is_published
    ]);

    return back()->with('success', 'Статус змінено!');
}
}

```

<?php

```

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Booking;
use App\Models\Schedule;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Inertia\Inertia;

class ScheduleController extends Controller
{
    public function index(Request $request)
    {
        $dateStr = $request->input('date', now()->toDateString());
        $date = Carbon::parse($dateStr);

        $schedules = Schedule::with(['user' => function ($query) {
            $query->withTrashed()->select('id', 'name', 'first_name', 'last_name');
        }])
            ->whereDate('date', $date)
            ->get();

        $bookings = Booking::with(['barber', 'services'])
            ->where('status', '!=', 'cancelled')
            ->whereDate('start_at', $date)
            ->get();

        return Inertia::render('Admin/Schedule/Index', [
            'schedules' => $schedules,
            'bookings' => $bookings,
            'selectedDate' => $dateStr,
        ]);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 98 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

}

public function cancel(Request $request, Booking $booking)
{
    $request->validate([
        'remove_schedule' => 'required|boolean',
    ]);

    $booking->update([
        'status' => 'cancelled',
    ]);

    if ($request->remove_schedule) {
        $date = $booking->start_at->toDateString();
        $bookingStart = $booking->start_at->format('H:i:s');
        $bookingEnd = $booking->end_at->format('H:i:s');

        $schedules = Schedule::where('user_id', $booking->user_id)
            ->where('date', $date)
            ->where('start_time', '<', $bookingEnd)
            ->where('end_time', '>', $bookingStart)
            ->get();

        foreach ($schedules as $schedule) {
            $oldStart = $schedule->start_time;
            $oldEnd = $schedule->end_time;

            if ($oldStart >= $bookingStart && $oldEnd <= $bookingEnd) {
                $schedule->delete();
            } elseif ($oldStart < $bookingStart && $oldEnd > $bookingEnd) {
                $schedule->update(['end_time' => $bookingStart]);

                Schedule::create([
                    'user_id' => $schedule->user_id,
                    'chair_number' => $schedule->chair_number,
                    'date' => $schedule->date,
                    'start_time' => $bookingEnd,
                    'end_time' => $oldEnd,
                ]);
            } elseif ($oldStart < $bookingStart) {
                $schedule->update(['end_time' => $bookingStart]);
            } elseif ($oldEnd > $bookingEnd) {
                $schedule->update(['start_time' => $bookingEnd]);
            }
        }
    }

    return back();
}

public function complete(Booking $booking)
{
    if (now()->lt($booking->end_at)) {
        return back()->withErrors(['message' => 'Ви не можете підтвердити стрижку до її завершення.']);
    }
}

```

```

        $booking->update([
            'status' => 'completed',
        ]);

        return back();
    }
}

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Service;
use Illuminate\Http\Request;
use Inertia\Inertia;

class ServiceController extends Controller
{
    public function index()
    {
        return Inertia::render('Admin/Services/Index', [
            'services' => Service::latest()->get(),
        ]);
    }

    public function create()
    {
        return Inertia::render('Admin/Services/Create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
            'duration_minutes' => 'required|integer|min:30|max:120',
        ]);

        Service::create($validated);

        return redirect()->route('admin.services.index')->with('success', 'Послугу створено');
    }

    public function edit(Service $service)
    {
        return Inertia::render('Admin/Services/Edit', [
            'service' => $service,
        ]);
    }

    public function update(Request $request, Service $service)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
        ]);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 100 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        'duration_minutes' => 'required|integer|min:30|max:120',
    ]);

    $service->update($validated);

    return redirect()->route('admin.services.index')->with('success', 'Послугу оновлено');
}

public function destroy(Service $service)
{
    $service->delete();

    return back()->with('success', 'Послугу видалено');
}
}

```

<?php

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;

```

```

class UploadController extends Controller
{
    public function image(Request $request)
    {
        $request->validate([
            'image' => 'required|image|max:5120',
        ]);

        $file = $request->file('image');
        $filename = Str::random(40) . '.' . $file->getClientOriginalExtension();
        $path = $file->storeAs('uploads', $filename, 'public');

        $url = Storage::disk('public')->url($path);

        return response()->json([
            'path' => $path,
            'url' => $url,
        ]);
    }
}

```

API контроллер:

<?php

```
namespace App\Http\Controllers\Api;
```

```

use App\Http\Controllers\Controller;
use App\Models\Booking;
use App\Models\Service;
use App\Models\User;
use App\Enums\UserRole;

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 101 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

use Carbon\Carbon;
use Carbon\CarbonPeriod;
use App\Mail\BookingConfirmed;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Mail;

class BookingController extends Controller
{
    public function getBarbers()
    {
        $barbers = User::where('role', UserRole::BARBER)
            ->select('id', 'first_name', 'last_name', 'avatar', 'position')
            ->get();

        return response()->json($barbers);
    }

    public function getServices(Request $request)
    {
        $barberId = $request->query('barber_id');

        $query = Service::query();

        if ($barberId) {
            $query->whereHas('barbers', function($q) use ($barberId) {
                $q->where('users.id', $barberId)->where('is_active', true);
            });
        }

        $services = $query->get()->map(function($service) use ($barberId) {
            $price = $barberId
                ? DB::table('service_user')
                    ->where('service_id', $service->id)
                    ->where('user_id', $barberId)
                    ->value('price')
                : null;

            return [
                'id' => $service->id,
                'name' => $service->name,
                'description' => $service->description,
                'duration' => $service->duration_minutes,
                'price' => $price ?? 0,
            ];
        });

        return response()->json($services);
    }

    public function getAvailableSlots(Request $request)
    {
        $request->validate([
            'barber_id' => 'required|exists:users,id',
            'date' => 'required|date_format:Y-m-d',
        ]);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 102 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

'service_ids' => 'required|array',
'service_ids.*' => 'exists:services,id',
]);

$barberId = $request->barber_id;
$date = Carbon::parse($request->date);

// Use the first service since it's now single-choice
$service = Service::find($request->service_ids[0]);
if (!$service) return response()->json([]);

// Effective duration: <= 60 min -> 60 min, > 60 min -> 120 min
$effectiveDuration = $service->duration_minutes <= 60 ? 60 : 120;

// Get working hours for the barber on this day
$schedule = DB::table('schedules')
->where('user_id', $barberId)
->whereDate('date', $date)
->get();

if ($schedule->isEmpty()) {
    return response()->json([]);
}

$availableSlots = [];

foreach ($schedule as $shift) {
    $start = Carbon::parse($date->format('Y-m-d') . ' ' . $shift->start_time);
    $end = Carbon::parse($date->format('Y-m-d') . ' ' . $shift->end_time);

    // Get existing bookings for this barber on this day
    $bookings = Booking::where('user_id', $barberId)
->where('status', '!=', 'cancelled')
->where(function($q) use ($start, $end) {
    $q->where('start_at', '<', $end)
->where('end_at', '>', $start);
    })
->get();

    $currentSlot = $start->copy()->ceilHour();

    while ($currentSlot->copy()->addMinutes($effectiveDuration)->lte($end)) {
        $slotEnd = $currentSlot->copy()->addMinutes($effectiveDuration);

        // Check if slot overlaps with any booking
        $isAvailable = true;
        foreach ($bookings as $booking) {
            if ($currentSlot->lt($booking->end_at) && $slotEnd->gt($booking->start_at)) {
                $isAvailable = false;
                break;
            }
        }

        if ($isAvailable && $currentSlot->gt(now())) {
            $availableSlots[] = $currentSlot->format('H:i');
        }
    }
}

```

```

        $currentSlot->addMinutes(60); // Step 60 mins (hourly)
    }
}

return response()->json(array_unique($availableSlots));
}

public function getNearestSlot(Request $request)
{
    $request->validate([
        'barber_id' => 'required|exists:users,id',
        'service_ids' => 'required|array',
        'service_ids.*' => 'exists:services,id',
    ]);

    $barberId = $request->barber_id;
    $service = Service::find($request->service_ids[0]);
    if (!$service) return response()->json(['message' => 'Service not found'], 422);

    $effectiveDuration = $service->duration_minutes <= 60 ? 60 : 120;
    $currentDate = Carbon::today();

    for ($i = 0; $i < 30; $i++) {
        $date = $currentDate->copy()->addDays($i);

        $schedule = DB::table('schedules')
            ->where('user_id', $barberId)
            ->whereDate('date', $date)
            ->get();

        if ($schedule->isEmpty()) continue;

        $bookings = Booking::where('user_id', $barberId)
            ->where('status', '!=', 'cancelled')
            ->whereDate('start_at', $date)
            ->get();

        foreach ($schedule as $shift) {
            $start = Carbon::parse($date->format('Y-m-d') . ' ' . $shift->start_time);
            $end = Carbon::parse($date->format('Y-m-d') . ' ' . $shift->end_time);

            $currentSlot = $start->copy()->ceilHour();
            if ($date->isToday() && $currentSlot->lt(now())) {
                $currentSlot = now()->ceilHour();
            }

            while ($currentSlot->copy()->addMinutes($effectiveDuration)->lte($end)) {
                $slotEnd = $currentSlot->copy()->addMinutes($effectiveDuration);

                $isAvailable = true;
                foreach ($bookings as $booking) {
                    if ($currentSlot->lt($booking->end_at) && $slotEnd->gt($booking->start_at)) {
                        $isAvailable = false;
                        break;
                    }
                }
            }
        }
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 104 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    }

    if ($isAvailable) {
        return response()->json([
            'date' => $date->format('Y-m-d'),
        ]);
    }
    $currentSlot->addMinutes(60);
}
}

return response()->json(['message' => 'На найближчий місяць вільних слотів не знайдено'], 404);
}

public function store(Request $request)
{
    $request->validate([
        'barber_id' => 'required|exists:users,id',
        'service_ids' => 'required|array',
        'service_ids.*' => 'exists:services,id',
        'date' => 'required|date_format:Y-m-d',
        'time' => 'required|date_format:H:i',
        'first_name' => 'required|string|max:255',
        'last_name' => 'required|string|max:255',
        'email' => 'nullable|email|max:255',
        'phone' => ['required', 'string', 'regex:/^\+380\s(\d{2})\s\d{3}-\d{2}-\d{2}$/',
        'notes' => 'nullable|string',
        'captcha_token' => 'required|string',
    ]);

    // Verify reCAPTCHA
    $response = Http::asForm()->post('https://www.google.com/recaptcha/api/siteverify', [
        'secret' => config('services.recaptcha.secret_key'),
        'response' => $request->captcha_token,
        'remoteip' => $request->ip(),
    ]);

    if (!$response->json('success') || $response->json('score') < 0.5) {
        return response()->json(['message' => 'Помилка перевірки на робота. Будь ласка, спробуйте ще раз або оновіть сторінку.'], 422);
    }

    $startAt = Carbon::parse($request->date . ' ' . $request->time);
    $service = Service::find($request->service_ids[0]);
    if (!$service) {
        return response()->json(['message' => 'Послугу не знайдено.'], 422);
    }

    $effectiveDuration = $service->duration_minutes <= 60 ? 60 : 120;
    $endAt = $startAt->copy()->addMinutes($effectiveDuration);

    // Double check availability (overlap check)
    $exists = Booking::where('user_id', $request->barber_id)
        ->where('status', '!=', 'cancelled')
        ->where('start_at', '<', $endAt)

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 105 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

->where('end_at', '>', $startAt)
->exists();

if ($exists) {
    return response()->json(['message' => 'На жаль, цей час вже зайнятий.'], 422);
}

$totalPrice = DB::table('service_user')
    ->where('service_id', $service->id)
    ->where('user_id', $request->barber_id)
    ->value('price') ?? 0;

try {
    DB::beginTransaction();

    $booking = Booking::create([
        'user_id' => $request->barber_id,
        'client_first_name' => $request->first_name,
        'client_last_name' => $request->last_name,
        'client_email' => $request->email,
        'client_phone' => $request->phone,
        'start_at' => $startAt,
        'end_at' => $endAt,
        'total_price' => $totalPrice,
        'notes' => $request->notes,
        'status' => 'pending',
    ]);

    $booking->services()->attach($service->id, ['price' => $totalPrice]);

    DB::commit();

    if ($booking->client_email) {
        $booking->load(['barber', 'services']);
        Mail::to($booking->client_email)->send(new BookingConfirmed($booking));
    }

    return response()->json([
        'message' => 'Запис успішно створено!',
        'booking' => $booking
    ]);

} catch (\Exception $e) {
    DB::rollBack();
    return response()->json(['message' => 'Помилка при створенні запису: ' . $e->getMessage()], 500);
}
}
}

```

Контроллери барбера:

<?php

namespace App\Http\Controllers\Barber;

use App\Http\Controllers\Controller;

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 106 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

use App\Models\Booking;
use App\Models\Schedule;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Inertia\Inertia;

class DashboardController extends Controller
{
    public function index(Request $request)
    {
        $date = $request->query('date', Carbon::today()->format('Y-m-d'));
        $barberId = Auth::id();

        $bookings = Booking::where('user_id', $barberId)
            ->whereDate('start_at', $date)
            ->with('services')
            ->orderBy('start_at')
            ->get();

        $schedules = Schedule::where('user_id', $barberId)
            ->whereDate('date', $date)
            ->get();

        $dayStart = $schedules->min('start_time') ?? '09:00:00';
        $dayEnd = $schedules->max('end_time') ?? '21:00:00';

        if ($bookings->isNotEmpty()) {
            $bookingStart = $bookings->min('start_at')->format('H:i:s');
            $bookingEnd = $bookings->max('end_at')->format('H:i:s');

            if ($bookingStart < $dayStart) {
                $dayStart = $bookingStart;
            }
            if ($bookingEnd > $dayEnd) {
                $dayEnd = $bookingEnd;
            }
        }

        return Inertia::render('Barber/Dashboard', [
            'bookings' => $bookings,
            'schedules' => $schedules,
            'dayRange' => [
                'start' => $dayStart,
                'end' => $dayEnd,
            ],
            'selectedDate' => $date,
        ]);
    }
}
<?php

namespace App\Http\Controllers\Barber;

use App\Http\Controllers\Controller;

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 107 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

use App\Models\PortfolioWork;
use Illuminate\Http\Request;
use Inertia\Inertia;

class PortfolioController extends Controller
{
    public function index(Request $request)
    {
        return Inertia::render('Barber/Portfolio/Index', [
            'works' => $request->user()->portfolioWorks
        ]);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'image_path' => 'required|string',
            'description' => 'nullable|string|max:255',
        ]);

        $user = $request->user();
        $maxOrder = $user->portfolioWorks()->max('order') ?? 0;

        $user->portfolioWorks()->create([
            'image_path' => $validated['image_path'],
            'description' => $validated['description'],
            'order' => $maxOrder + 1,
        ]);

        return back()->with('success', 'Роботу додано до портфоліо');
    }

    public function updateOrder(Request $request)
    {
        $request->validate([
            'ids' => 'required|array',
            'ids.*' => 'exists:portfolio_works,id'
        ]);

        foreach ($request->ids as $index => $id) {
            PortfolioWork::where('id', $id)
                ->where('user_id', $request->user()->id)
                ->update(['order' => $index]);
        }

        return back()->with('success', 'Порядок оновлено');
    }

    public function destroy(PortfolioWork $portfolio)
    {
        if ($portfolio->user_id !== auth()->id()) {
            abort(403);
        }

        $portfolio->delete();
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 108 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return back()->with('success', 'Роботу видалено');
    }
}

<?php

namespace App\Http\Controllers\Barber;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Str;
use Inertia\Inertia;

class ProfileController extends Controller
{
    public function edit(Request $request)
    {
        return Inertia::render('Barber/Profile/Edit', [
            'user' => $request->user()
        ]);
    }

    public function update(Request $request)
    {
        $user = $request->user();

        $validated = $request->validate([
            'first_name' => 'required|string|max:255',
            'last_name' => 'required|string|max:255',
            'position' => 'nullable|string|max:255',
            'bio' => 'nullable|string',
            'avatar' => 'nullable|string',
            'slug' => 'required|string|max:255|unique:users,slug,' . $user->id,
        ]);

        $validated['name'] = $validated['first_name'] . ' ' . $validated['last_name'];
        $validated['slug'] = Str::slug($validated['slug']);

        $user->update($validated);

        return back()->with('success', 'Профіль оновлено');
    }
}

```

```

<?php

namespace App\Http\Controllers\Barber;

use App\Http\Controllers\Controller;
use App\Models\Schedule;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Inertia\Inertia;

class ScheduleController extends Controller

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 109 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

{
    public function index(Request $request)
    {
        $schedules = Schedule::with(['user' => function ($query) {
            $query->withTrashed()->select('id', 'name');
        }])
            ->where('date', '>=', now()->toDateString())
            ->get();

        $userId = $request->user()->id;

        $schedules->each(function ($schedule) {
            $schedule->has_bookings = \App\Models\Booking::where('user_id', $schedule->user_id)
                ->where('status', '!=', 'cancelled')
                ->where(function ($query) use ($schedule) {
                    $dateStr = $schedule->date->toDateString();
                    $start = $dateStr.' '.$schedule->start_time;
                    $end = $dateStr.' '.$schedule->end_time;

                    $query->where('start_at', '<', $end)
                        ->where('end_at', '>', $start);
                })
                ->exists();
        });

        $myBookings = \App\Models\Booking::where('user_id', $userId)
            ->where('status', '!=', 'cancelled')
            ->where('start_at', '>=', now()->toDateString())
            ->get(['start_at', 'end_at']);

        return Inertia::render('Barber/Schedule', [
            'schedules' => $schedules,
            'myBookings' => $myBookings,
        ]);
    }

    public function cancelSlot(Request $request)
    {
        $validated = $request->validate([
            'date' => 'required|date',
            'chair_number' => 'required|integer',
            'hour' => 'required|integer|min:0|max:23',
        ]);

        $dateStr = Carbon::parse($validated['date'])->toDateString();
        $hourStart = sprintf('%02d:00:00', $validated['hour']);
        $hourEnd = sprintf('%02d:00:00', $validated['hour'] + 1);

        $schedule = Schedule::where('user_id', $request->user()->id)
            ->where('chair_number', $validated['chair_number'])
            ->where('date', $dateStr)
            ->where('start_time', '<=', $hourStart)
            ->where('end_time', '>=', $hourEnd)
            ->first();

        if (!$schedule) {

```

```

        return back()->withErrors(['error' => 'Запис не знайдено.']);
    }

    $hasBookingInSlot = \App\Models\Booking::where('user_id', $request->user()->id)
        ->where('status', '!=', 'cancelled')
        ->where(function ($query) use ($dateStr, $hourStart, $hourEnd) {
            $start = $dateStr.' '.$hourStart;
            $end = $dateStr.' '.$hourEnd;
            $query->where('start_at', '<', $end)
                ->where('end_at', '>', $start);
        })
        ->exists();

    if ($hasBookingInSlot) {
        return back()->withErrors(['error' => 'Неможливо відмінити годину, на яку є запис клієнта.']);
    }

    $oldStart = $schedule->start_time;
    $oldEnd = $schedule->end_time;

    if ($oldStart === $hourStart && $oldEnd === $hourEnd) {

        $schedule->delete();
    } elseif ($oldStart === $hourStart) {

        $schedule->update(['start_time' => $hourEnd]);
    } elseif ($oldEnd === $hourEnd) {

        $schedule->update(['end_time' => $hourStart]);
    } else {

        $schedule->update(['end_time' => $hourStart]);

        Schedule::create([
            'user_id' => $request->user()->id,
            'chair_number' => $schedule->chair_number,
            'date' => $schedule->date,
            'start_time' => $hourEnd,
            'end_time' => $oldEnd,
        ]);
    }

    return back()->with('success', 'Годину успішно видалено з розкладу');
}

public function store(Request $request)
{
    $validated = $request->validate([
        'chair_number' => 'required|integer|min:1|max:4',
        'date' => 'required|date|after_or_equal:today',
        'start_time' => 'required|string',
        'end_time' => 'required|string|after:start_time',
    ]);

    $chairOverlap = Schedule::where('chair_number', $validated['chair_number'])
        ->where('date', $validated['date'])

```

```

->where(function ($query) use ($validated) {
    $query->where('start_time', '<', $validated['end_time'])
    ->where('end_time', '>', $validated['start_time']);
})
->exists();

if ($chairOverlap) {
    return back()->withErrors(['start_time' => 'Цей час уже зайнятий на обраному кріслі.']);
}

$barberOverlap = Schedule::where('user_id', $request->user()->id)
->where('date', $validated['date'])
->where(function ($query) use ($validated) {
    $query->where('start_time', '<', $validated['end_time'])
    ->where('end_time', '>', $validated['start_time']);
})
->exists();

if ($barberOverlap) {
    return back()->withErrors(['start_time' => 'Ви вже працюєте в цей час на іншому кріслі.']);
}

$request->user()->schedules()->create($validated);

return back()->with('success', 'Розклад успішно збережено');
}

public function destroy(Schedule $schedule)
{
    if ($schedule->user_id !== auth()->id()) {
        abort(403);
    }

    $hasBookings = \App\Models\Booking::where('user_id', $schedule->user_id)
    ->where('status', '!=', 'cancelled')
    ->where(function ($query) use ($schedule) {
        $dateStr = $schedule->date->toDateString();
        $start = $dateStr.'.'.$schedule->start_time;
        $end = $dateStr.'.'.$schedule->end_time;

        $query->where('start_at', '<', $end)
        ->where('end_at', '>', $start);
    })
    ->exists();

    if ($hasBookings) {
        return back()->withErrors(['error' => 'Неможливо видалити розклад, на який уже є записи клієнтів.']);
    }

    $schedule->delete();

    return back()->with('success', 'Запис видалено');
}
}

```

```

<?php

namespace App\Http\Controllers\Barber;

use App\Http\Controllers\Controller;
use App\Models\Service;
use Illuminate\Http\Request;
use Inertia\Inertia;

class ServiceController extends Controller
{
    public function index(Request $request)
    {
        $user = $request->user();
        $user->load('services');

        $services = Service::all()->map(function ($service) use ($user) {
            $userPivot = $user->services->where('id', $service->id)->first();

            return [
                'id' => $service->id,
                'name' => $service->name,
                'description' => $service->description,
                'is_active' => $userPivot ? (bool)$userPivot->pivot->is_active : false,
                'price' => $userPivot ? $userPivot->pivot->price : null,
            ];
        });

        return Inertia::render('Barber/Services/Index', [
            'services' => $services,
        ]);
    }

    public function update(Request $request)
    {
        $request->validate([
            'services' => 'array',
            'services.*.id' => 'required|exists:services,id',
            'services.*.is_active' => 'required|boolean',
            'services.*.price' => 'nullable|numeric|min:0',
        ]);

        $user = $request->user();

        $syncData = [];
        foreach ($request->services as $serviceData) {
            if ($serviceData['is_active']) {
                $syncData[$serviceData['id']] = [
                    'price' => $serviceData['price'] ? : 300,
                    'is_active' => true,
                ];
            }
        }

        $user->services()->sync($syncData);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 113 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return back()->with('success', 'Послуги оновлено');
    }
}

```

Контролери блоку клієнта:

```
<?php
```

```
namespace App\Http\Controllers\Clients;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Post;
```

```
class BlogController extends Controller
```

```

{
    public function index()
    {
        $posts = Post::where('is_published', true)->latest()->get();

        return view('clients.blog', compact('posts'));
    }

```

```

    public function show($slug)
    {
        $post = Post::where('slug', $slug)
            ->where('is_published', true)
            ->firstOrFail();

        return view('clients.blog-post', compact('post'));
    }
}

```

```
<?php
```

```
namespace App\Http\Controllers\Clients;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Booking;
```

```
use Illuminate\Http\Request;
```

```
use Illuminate\Support\Facades\URL;
```

```
use Inertia\Inertia;
```

```
class BookingCancellationController extends Controller
```

```

{
    public function show(Request $request, Booking $booking)
    {
        if (!$request->hasValidSignature()) {
            abort(403, 'Invalid or expired signature.');
```

```

        }

        if ($booking->status === 'cancelled') {
            return Inertia::render('bookings/Cancelled', [
                'booking' => $booking->load('barber', 'services'),
                'already_cancelled' => true,
            ]);
        }
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 114 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    }

    return Inertia::render('bookings/Cancel', [
        'booking' => $booking->load('barber', 'services'),
        'cancel_url' => URL::signedRoute('bookings.cancel.confirm', ['booking' => $booking->id]),
    ]);
}

public function cancel(Request $request, Booking $booking)
{
    if (!$request->hasValidSignature()) {
        abort(403, 'Invalid or expired signature.');
```

```

    }
}

<?php
```

```

namespace App\Http\Controllers\Clients;
```

```

use App\Enums\UserRole;
use App\Http\Controllers\Controller;
use App\Models\User;
```

```

class MastersController extends Controller
{
    public function index()
    {
        $masters = User::where('role', UserRole::BARBER)
            ->whereNotNull('slug')
            ->latest()
            ->get();

        return view('clients.masters', compact('masters'));
    }

    public function show($slug)
    {
        $master = User::where('role', UserRole::BARBER)
            ->where('slug', $slug)
            ->with(['portfolioWorks' => function($query) {
                $query->orderBy('order');
            }])
            ->firstOrFail();

        return view('clients.master-portfolio', compact('master'));
    }
}
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 115 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

<?php

```
namespace App\Http\Controllers\Clients;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\Service;
```

```
class ServicesController extends Controller
```

```
{
```

```
    public function index()
```

```
    {
```

```
        $services = Service::query()
```

```
            ->join('service_user', 'services.id', '=', 'service_user.service_id')
```

```
            ->where('service_user.is_active', true)
```

```
            ->select('services.*')
```

```
            ->selectRaw('MIN(service_user.price) as min_price, MAX(service_user.price) as max_price')
```

```
            ->groupBy('services.id', 'services.name', 'services.description', 'services.created_at', 'services.updated_at')
```

```
            ->get();
```

```
        return view('clients.services', compact('services'));
```

```
    }
```

```
}
```

КЛІЄНТСЬКІ СТОРІНКИ:

```
<script setup lang="ts">
```

```
import { Head, router } from '@inertiajs/vue3';
```

```
import { DatePicker } from 'v-calendar';
```

```
import { ref, computed } from 'vue';
```

```
import Heading from '@components/Heading.vue';
```

```
import { Button } from '@components/ui/button';
```

```
import { Card, CardContent, CardHeader, CardTitle } from '@components/ui/card';
```

```
import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogDescription, DialogFooter } from '@components/ui/dialog';
```

```
import { Badge } from '@components/ui/badge';
```

```
import AppLayout from '@layouts/AppLayout.vue';
```

```
import { index as scheduleIndex } from '@routes/admin/schedule';
```

```
import 'v-calendar/dist/style.css';
```

```
interface User {
```

```
    id: number;
```

```
    name: string;
```

```
    first_name: string;
```

```
    last_name: string;
```

```
}
```

```
interface Service {
```

```
    id: number;
```

```
    name: string;
```

```
    price: number;
```

```
}
```

```
interface Booking {
```

```
    id: number;
```

```
    user_id: number;
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 116 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    client_first_name: string;
    client_last_name: string;
    client_email: string;
    client_phone: string;
    start_at: string;
    end_at: string;
    total_price: string;
    status: string;
    notes: string;
    barber?: User;
    services?: Service[];
  }

```

```

interface Schedule {
  id: number;
  user_id: number;
  chair_number: number;
  date: string;
  start_time: string;
  end_time: string;
  user: User;
}

```

```

const props = defineProps<{
  schedules: Schedule[];
  bookings: Booking[];
  selectedDate: string;
}>();

```

```

const breadcrumbs = [
  {
    title: 'Розклад',
    href: '/admin/schedule',
  },
];

```

```

const date = ref(new Date(props.selectedDate));
const selectedBooking = ref<Booking | null>(null);
const isDialogOpen = ref(false);
const isRemovingSchedule = ref(false);

```

```

const isPastBooking = computed(() => {
  if (!selectedBooking.value) return false;
  const endTime = new Date(selectedBooking.value.end_at);
  return new Date() >= endTime;
});

```

```

const updateDate = (newDate: Date) => {
  const dateStr = newDate.toISOString().split('T')[0];
  router.get(scheduleIndex.url({ query: { date: dateStr } })), {}, { preserveState: true });
};

```

```

const hours = Array.from({ length: 13 }, (_, i) => i + 10);
const chairs = [1, 2, 3, 4];

```

```

const getScheduleFor = (chair: number, hour: number) => {

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 117 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

const timeStr = `${hour.toString().padStart(2, '0')}:00:00`;
return props.schedules.find(s =>
  s.chair_number === chair &&
  timeStr >= s.start_time &&
  timeStr < s.end_time
);
};

const getBookingFor = (barberId: number, hour: number) => {
  return props.bookings.find(b => {
    const startTime = b.start_at.split(/[sT]/)[1];
    const endTime = b.end_at.split(/[sT]/)[1];

    if (!startTime || !endTime) return false;

    const startHour = parseInt(startTime.split(':')[0]);
    const endHour = parseInt(endTime.split(':')[0]);
    const endMinutes = parseInt(endTime.split(':')[1]);

    const actualEndHour = endMinutes > 0 ? endHour : endHour - 1;

    return b.user_id === barberId && hour >= startHour && hour <= actualEndHour;
  });
};

const showBookingDetails = (booking: Booking) => {
  selectedBooking.value = booking;
  isRemovingSchedule.value = false;
  isDialogOpen.value = true;
};

const cancelBooking = () => {
  if (!selectedBooking.value) return;

  if (confirm('Ви впевнені, що хочете скасувати цей запис?')) {
    router.post(`/admin/bookings/${selectedBooking.value.id}/cancel`, {
      remove_schedule: isRemovingSchedule.value
    }, {
      onSuccess: () => {
        isDialogOpen.value = false;
      }
    });
  }
};

const completeBooking = () => {
  if (!selectedBooking.value) return;

  router.post(`/admin/bookings/${selectedBooking.value.id}/complete`, {}, {
    onSuccess: () => {
      isDialogOpen.value = false;
    }
  });
};

const formatTime = (hour: number) => `${hour}:00`;

```

```

const formatDateTime = (dateStr: string) => {
  if (!dateStr) return "";
  const parts = dateStr.split(/[sT]/);
  const datePart = parts[0];
  const timePart = parts[1] ? parts[1].substring(0, 5) : '00:00';

  const [y, m, d] = datePart.split('-');
  return `${d}.${m}.${y}, ${timePart}`;
};

</script>

<template>
  <Head title="Розклад" />
  <AppLayout :breadcrumbs="breadcrumbs">
    <div class="max-w-7xl mx-auto py-8 px-4">
      <div class="flex flex-col md:flex-row md:items-center justify-between gap-4 mb-8">
        <Heading title="Загальний розклад" description="Перегляд графіку роботи всіх барберів" />

        <div class="flex items-center gap-4">
          <DatePicker
            v-model="date"
            mode="date"
            @update:model-value="updateDate"
            color="gray"
            locale="uk"
            :popover="{ visibility: 'click' }"
          >
            <template #default="{ togglePopover }">
              <Button variant="outline" @click="togglePopover">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24"
                  fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="mr-2">
                  <rect width="18" height="18" x="3" y="4" rx="2" ry="2"/>
                  <path d="M16 2v4"/>
                  <path d="M8 2v4"/>
                  <path d="M3 10h18"/>
                </svg>
                {{ date.toLocaleDateString('uk-UA', { day: 'numeric', month: 'long', year: 'numeric' }) }}
              </Button>
            </template>
          </DatePicker>
        </div>
      </div>

      <Card class="overflow-hidden">
        <CardHeader class="bg-muted/50">
          <CardTitle class="text-lg">Крісла та години</CardTitle>
        </CardHeader>
        <CardContent class="p-0 overflow-x-auto">
          <table class="w-full border-collapse">
            <thead>
              <tr class="border-b bg-muted/30">
                <th class="p-4 text-left font-semibold border-r w-32">Час</th>
                <th v-for="chair in chairs" :key="chair" class="p-4 text-center font-semibold border-r
last:border-r-0">
                  Крісло {{ chair }}
                </th>
              </tr>
            </thead>
          </table>
        </CardContent>
      </Card>
    </div>
  </AppLayout>
</template>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 119 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        </tr>
    </thead>
    <tbody>
        <tr v-for="hour in hours" :key="hour" class="border-b last:border-b-0 hover:bg-muted/5
transition-colors">
            <td class="p-4 font-medium border-r bg-muted/10">
                {{ formatTime(hour) }}
            </td>
            <td v-for="chair in chairs" :key="chair" class="p-2 border-r last:border-r-0 h-20 relative">
                <template v-for="slot in [getScheduleFor(chair, hour)]" :key="slot?.id || `${chair}-${
hour}`">
                    <template v-if="slot">
                        <div
                            v-for="booking in [getBookingFor(slot.user_id, hour)]"
                            :key="booking?.id || 'no-booking'"
                            :class="[
                                'absolute inset-1 flex flex-col items-center justify-center rounded-md border text-
xs font-semibold p-1 text-center transition-colors',
                                booking
                                    ? booking.status === 'completed'
                                        ? 'bg-blue-500/10 border-blue-500/30 text-blue-600 dark:text-blue-400
cursor-pointer hover:bg-blue-500/20'
                                        : 'bg-green-500/10 border-green-500/30 text-green-600 dark:text-green-500
cursor-pointer hover:bg-green-500/20'
                                    : 'bg-amber-500/10 border-amber-500/30 text-amber-600 dark:text-amber-500
hover:bg-amber-500/20'
                            ]"
                            @click="booking ? showBookingDetails(booking) : null"
                        >
                            <span>{{ slot.user?.name || 'Видалений' }}</span>
                            <span v-if="booking" class="text-[10px] opacity-80 font-normal">
                                {{ booking.client_first_name }} {{ booking.client_last_name }}
                            <span v-if="booking.status === 'completed'" class="block text-[8px] uppercase
font-bold text-blue-500">Виконано</span>
                            </span>
                        </div>
                    </template>
                    <div v-else class="w-full h-full flex items-center justify-center text-[10px] text-muted-
foreground/30 uppercase tracking-widest font-bold italic text-center leading-tight">
                        неробочий час
                    </div>
                </template>
            </td>
        </tr>
    </tbody>
</table>
</CardContent>
</Card>

<Dialog v-model:open="isDialogOpen">
    <DialogContent class="sm:max-w-[425px]">
        <DialogHeader>
            <DialogTitle>Деталі запису</DialogTitle>
            <DialogDescription>
                Інформація про бронювання та клієнта
            </DialogDescription>
        </DialogHeader>
    </DialogContent>
</Dialog>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 120 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

</DialogHeader>

<div v-if="selectedBooking" class="grid gap-4 py-4">
  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Клієнт:</span>
    <span class="col-span-3 text-sm font-semibold">
      {{ selectedBooking.client_first_name }} {{ selectedBooking.client_last_name }}
    </span>
  </div>
  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Телефон:</span>
    <span class="col-span-3 text-sm">{{ selectedBooking.client_phone }}</span>
  </div>
  <div v-if="selectedBooking.client_email" class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Email:</span>
    <span class="col-span-3 text-sm">{{ selectedBooking.client_email }}</span>
  </div>
  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Час:</span>
    <span class="col-span-3 text-sm">
      {{ formatDateTime(selectedBooking.start_at) }} - {{
formatDateTime(selectedBooking.end_at).split(',')[1] }}
    </span>
  </div>
  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Барбер:</span>
    <span class="col-span-3 text-sm">{{ selectedBooking.barber?.name }}</span>
  </div>
  <div class="grid grid-cols-4 items-start gap-4">
    <span class="text-sm font-medium text-muted-foreground">Послуги:</span>
    <div class="col-span-3 flex flex-wrap gap-1">
      <Badge v-for="service in selectedBooking.services" :key="service.id" variant="secondary"
class="text-[10px]">
        {{ service.name }} ({{ service.price }} грн)
      </Badge>
    </div>
  </div>
  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Сума:</span>
    <span class="col-span-3 text-sm font-bold text-green-600">{{ selectedBooking.total_price }}
грн</span>
  </div>
  <div v-if="selectedBooking.notes" class="grid grid-cols-4 items-start gap-4">
    <span class="text-sm font-medium text-muted-foreground">Нотатки:</span>
    <span class="col-span-3 text-sm italic text-muted-foreground">{{ selectedBooking.notes
}}</span>
  </div>

  <div class="grid grid-cols-4 items-center gap-4">
    <span class="text-sm font-medium text-muted-foreground">Статус:</span>
    <div class="col-span-3">
      <Badge v-if="selectedBooking.status === 'completed'" variant="outline" class="bg-blue-50
text-blue-700 border-blue-200">Виконано</Badge>
      <Badge v-else variant="outline" class="bg-green-50 text-green-700 border-green-
200">Очікується</Badge>
    </div>
  </div>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
| | | | | | | 121 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

</div>

<div v-if="selectedBooking.status !== 'completed'" class="border-t pt-4 mt-2">
  <label class="flex items-center gap-2 cursor-pointer group">
    <input type="checkbox" v-model="isRemovingSchedule" class="w-4 h-4 rounded border-
gray-300 text-red-600 focus:ring-red-600">
    <span class="text-sm font-medium group-hover:text-red-600 transition-colors">Звільнити
крісло (видалити робочі години барбера)</span>
  </label>
</div>
</div>

<DialogFooter class="flex flex-col sm:flex-row justify-between items-center gap-2 sm:gap-0">
  <div class="flex flex-col sm:flex-row gap-2 w-full sm:w-auto">
    <template v-if="selectedBooking?.status !== 'completed'">
      <div v-if="!isPastBooking" class="text-[10px] text-amber-600 font-medium flex items-
center bg-amber-50 p-2 rounded border border-amber-100 mb-2 sm:mb-0">
        <svg xmlns="http://www.w3.org/2000/svg" width="12" height="12" viewBox="0 0 24
24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"
class="mr-1"><circle cx="12" cy="12" r="10"/><polyline points="12 6 12 12 16 14"/></svg>
        Підтвердження доступне після закінчення сеансу
      </div>
      <Button
        v-if="isPastBooking"
        type="button"
        variant="default"
        class="bg-blue-600 hover:bg-blue-700 w-full sm:w-auto"
        @click="completeBooking"
      >
        Підтвердити виконання
      </Button>
      <Button type="button" variant="destructive" class="w-full sm:w-auto"
@click="cancelBooking">
        Скасувати
      </Button>
    </template>
  </div>
  <Button type="button" variant="outline" class="w-full sm:w-auto" @click="isDialogOpen =
false">
    Закрити
  </Button>
</DialogFooter>
</DialogContent>
</Dialog>
</div>
</AppLayout>
</template>

<style scoped>
:deep(.vc-container) {
  border: none;
  font-family: inherit;
}
</style>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 122 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

<script setup lang="ts">
import { Head, useForm } from '@inertiajs/vue3';
import ImageUploader from '@components/ImageUploader.vue';
import RichTextEditor from '@components/RichTextEditor.vue';
import { Button } from '@components/ui/button';
import { Checkbox } from '@components/ui/checkbox';
import { Input } from '@components/ui/input';
import { Label } from '@components/ui/label';
import { Textarea } from '@components/ui/textarea';
import AppLayout from '@layouts/AppLayout.vue';
import type { BreadcrumbItem } from '@types';

const breadcrumbs: BreadcrumbItem[] = [
  { title: 'Адмін', href: '/admin' },
  { title: 'Блог', href: '/admin/posts' },
  { title: 'Нова стаття', href: '#' },
];

const form = useForm({
  title: "",
  excerpt: "",
  content: "",
  image_path: "",
  author: "",
  is_published: true,
});

const submit = () => {
  form.post('/admin/posts');
};
</script>

<template>
  <Head title="Нова стаття" />

  <AppLayout :breadcrumbs="breadcrumbs">
    <div class="max-w-4xl mx-auto p-6 bg-zinc-900 text-white pb-20">
      <h1 class="text-3xl font-black uppercase mb-8">Нова стаття</h1>

      <form @submit.prevent="submit" class="space-y-8">
        <div class="flex items-center space-x-2 bg-zinc-900 p-4 rounded-xl border border-zinc-800">
          <Checkbox id="is_published" :checked="form.is_published" @update:checked="form.is_published
= $event" />
          <Label for="is_published" class="text-sm font-medium leading-none cursor-
pointer">Опублікувати статтю (якщо вимкнено - буде збережено як чернетку)</Label>
        </div>

        <div class="space-y-2">
          <Label class="text-zinc-400 uppercase tracking-widest text-xs font-bold">Заголовок</Label>
          <Input v-model="form.title" class="bg-zinc-900 border-zinc-800 text-white py-6 text-lg"
placeholder="Введіть назву статті..." />
        </div>

        <div class="space-y-2">
          <Label class="text-zinc-400 uppercase tracking-widest text-xs font-bold">Головне
зображення</Label>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 – ПЗ | Арк. |
| | | | | | | 123 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    <ImageUploader v-model="form.image_path" :aspectRatio="16/9" />
  </div>

  <div class="space-y-2">
    <Label class="text-zinc-400 uppercase tracking-widest text-xs font-bold">Короткий опис (для
картки)</Label>
    <Textarea v-model="form.excerpt" rows="3" class="bg-zinc-900 border-zinc-800 text-white"
placeholder="Коротко про що стаття..." />
  </div>

  <div class="space-y-2">
    <Label class="text-zinc-400 uppercase tracking-widest text-xs font-bold">Контент статті</Label>
    <RichTextEditor v-model="form.content" />
  </div>

  <div class="grid md:grid-cols-2 gap-6">
    <div class="space-y-2">
      <Label class="text-zinc-400 uppercase tracking-widest text-xs font-bold">Автор</Label>
      <Input v-model="form.author" class="bg-zinc-900 border-zinc-800 text-white" placeholder="Ім'я
автора..." />
    </div>
  </div>

  <div class="pt-6 border-t border-zinc-900">
    <Button type="submit" class="bg-amber-500 hover:bg-amber-600 text-black w-full py-8 text-xl
font-black uppercase tracking-widest transition-all" :disabled="form.processing">
      Опублікувати статтю
    </Button>
  </div>
</form>
</div>
</AppLayout>
</template>

```

```

<script setup lang="ts">
import { Head, useForm } from '@inertiajs/vue3';
import { Save, Loader2 } from 'lucide-vue-next';
import ImageUploader from '@components/ImageUploader.vue';
import { Button } from '@components/ui/button';
import { Input } from '@components/ui/input';
import { Label } from '@components/ui/label';
import { Textarea } from '@components/ui/textarea';
import AppLayout from '@layouts/AppLayout.vue';
import type { BreadcrumbItem, User } from '@types';

```

```

const props = defineProps<{
  user: User;
}>();

```

```

const breadcrumbs: BreadcrumbItem[] = [
  { title: 'Кабінет барбера', href: '/barber' },
  { title: 'Мій профіль', href: '#' },
];

```

```

const form = useForm({
  first_name: props.user.first_name || "",

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 124 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

last_name: props.user.last_name || "",
position: props.user.position || "",
bio: props.user.bio || "",
avatar: props.user.avatar || "",
slug: props.user.slug || "",
});

const submit = () => {
  form.put('/barber/profile');
};
</script>

<template>
  <Head title="Редагувати профіль" />

  <AppLayout :breadcrumbs="breadcrumbs">
    <div class="p-6 bg-zinc-950 min-h-screen text-white">
      <div class="max-w-4xl mx-auto">
        <h1 class="text-3xl font-black uppercase mb-8 tracking-tight">Налаштування профілю</h1>

        <form @submit.prevent="submit" class="grid grid-cols-1 md:grid-cols-3 gap-8">
          <!-- Left: Avatar & Personal -->
          <div class="md:col-span-1 space-y-6">
            <div class="bg-zinc-900 border border-zinc-800 p-6 rounded-2xl">
              <Label class="text-zinc-400 uppercase text-[10px] font-black tracking-widest mb-4 block">Фото майстра</Label>
              <ImageUploader v-model="form.avatar" :aspectRatio="3/4" />
              <p class="text-zinc-600 text-[10px] mt-2 italic text-center">Рекомендовано: 3:4 (портрет)</p>
            </div>
          </div>

          <!-- Right: Info -->
          <div class="md:col-span-2 space-y-6">
            <div class="bg-zinc-900 border border-zinc-800 p-8 rounded-2xl space-y-6">
              <div class="grid md:grid-cols-2 gap-6">
                <div class="space-y-2">
                  <Label for="first_name" class="text-zinc-400 uppercase text-[10px] font-black tracking-widest">Ім'я</Label>
                  <Input id="first_name" v-model="form.first_name" required class="bg-zinc-950 border-zinc-800 focus:border-amber-500" />
                </div>
                <div class="space-y-2">
                  <Label for="last_name" class="text-zinc-400 uppercase text-[10px] font-black tracking-widest">Прізвище</Label>
                  <Input id="last_name" v-model="form.last_name" required class="bg-zinc-950 border-zinc-800 focus:border-amber-500" />
                </div>
              </div>

              <div class="space-y-2">
                <Label for="position" class="text-zinc-400 uppercase text-[10px] font-black tracking-widest">Посада</Label>
                <Input id="position" v-model="form.position" placeholder="Top Barber" class="bg-zinc-950 border-zinc-800 focus:border-amber-500" />
              </div>
            </div>
          </div>
        </form>
      </div>
    </div>
  </AppLayout>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 125 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        <div class="space-y-2">
            <Label for="slug" class="text-zinc-400 uppercase text-[10px] font-black tracking-
widest">URL-адреса (Slug)</Label>
            <div class="flex items-center gap-2">
                <span class="text-zinc-600 text-sm">/masters/</span>
                <Input id="slug" v-model="form.slug" required class="bg-zinc-950 border-zinc-800
focus:border-amber-500" />
            </div>
            <p v-if="form.errors.slug" class="text-red-500 text-xs">{{ form.errors.slug }}</p>
        </div>

        <div class="space-y-2">
            <Label for="bio" class="text-zinc-400 uppercase text-[10px] font-black tracking-
widest">Про себе (для сторінки портфоліо)</Label>
            <Textarea id="bio" v-model="form.bio" rows="5" class="bg-zinc-950 border-zinc-800
focus:border-amber-500 min-h-[150px]" />
        </div>

        <div class="pt-4 flex justify-end">
            <Button type="submit" :disabled="form.processing" class="bg-amber-500 hover:bg-amber-
600 text-black font-black uppercase tracking-wider px-8">
                <Loader2 v-if="form.processing" class="w-4 h-4 mr-2 animate-spin" />
                <Save v-else class="w-4 h-4 mr-2" />
                Зберегти зміни
            </Button>
        </div>
    </div>
</div>
</form>
</div>
</div>
</AppLayout>
</template>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 126 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |