Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Веб-застосунок для управління послугами в мережі барбершопів

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-1
Михайло НАУМОВ
(власне ім'я, прізвище)

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

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

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

Керівник: ст. викл. Антон ПОРОШЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

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

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

Освітня програма Комп’ютерні системи та мережі
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Наумову Михайлу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Веб-застосунок для управління послугами в мережі барбершопів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи

1) мова програмування C#
2) документація фреймворку React
3) середа розробки Visual Studio та Visual Studio Code
4) документація бази даних MySQL

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз предметної області
2. Аналіз використаних технологій
3. Програмна реалізація
4. Інструкція користувача
5. Висновки

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

Слайд-презентація – 13 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	09.06.2025-13.06.25	
2	Формування переліку вимог до програми	14.06.2025-19.06.25	
3	Вибір технологій розробки та	20.06.25-25.06.25	
	інструментальних засобів		
4	Розробка алгоритмічного забезпечення	26.06.25-01.07.25	
5	Розробка та тестування веб-застосунку	02.07.25-6.07.25	
6	Відлагодження програмних модулів	07.07.25-08.07.25	
7	Оформлення матеріалів кваліфікаційної роботи	09.07.25-13.07.25	

Дата видачі завдання " 9 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Антон ПОРОШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 65 с., 21 рис., 1 дод., 12 джерел.

ВЕБ-ЗАСТОСУНОК, ІНТЕРНЕТ, ВЕБ-СЕРВІС, СЕРВЕР, КЛІЄНТ, РОЗПОДІЛ, КЛІЄНТ, ПРАЦІВНИК, C#, MYSQL, REACT, TYPESCRIPT.

Метою кваліфікаційної роботи є розробка веб-застосунку, який допоможе автоматизувати процес запису клієнтів на послуги в барбершопі. Майстри можуть переглядати записи, взаємодіяти з клієнтами та керувати власним розкладом, а клієнти можуть легко вибирати майстра, тип послуги та відповідний час.

У ході виконання кваліфікаційної роботи було проведено аналіз поточних рішень у цій галузі, щоб визначити основні обмеження, які були враховані під час розробки застосунку.

Фреймворк React і мова TypeScript використовуються для створення клієнтської частини, що дозволяє писати код, який можна типізувати. Серверна частина виконується за допомогою ASP.NET Core (C#), а дані зберігаються та обробляються в реляційній даних MySQL.

ABSTRACT

Bachelor’s thesis: 65 pages, 21 figures, 1 appendices, 12 sources.

WEB-APPLICATION, INTERNET, WEB-SERVICE, SERVER, CLIENT, DISTRIBUTION, CLIENT, EMPLOYEE, C#, MYSQL, REACT, TYPESCRIPT.

The major goal of this thesis is s to develop a web application that will help automate the process of booking clients for services in a barbershop. Masters can view records, interact with clients and manage their own schedule, and clients can easily select a master, type of service and a suitable time.

During the qualification work, an analysis of current solutions in this area was conducted to identify the main limitations that were taken into account during the development of the application.

The React framework and the TypeScript language are used to create the client part, which allows writing typed code. The server part is implemented using ASP.NET Core (C#), and data is stored and processed in the MySQL relational database.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

СУБД – система управління базою даних

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

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

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

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

HTTP – протокол передавання гіпертексту (англ. HyperText Transfer Protocol)

JIT – компіляція під час виконання (англ. Just-In-Time compilation)

LINQ – мова запитів у C# (англ. Language Integrated Query)

MVC – архітектурний шаблон (англ. Model–View–Controller)

ORM – об’єктно-реляційне відображення (англ. Object-Relational Mapping)

POS – точка продажу (англ. Point of Sale)

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

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

SVG – масштабована векторна графіка (англ. Scalable Vector Graphics)

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

ВСТУП

Більшість персональних комп'ютерів, портативних комп'ютерів і мобільних пристроїв сьогодні залишаються підключеними до Інтернету, який стає основним каналом для взаємодії між користувачами та цифровими сервісами. Вони дають змогу стандартизувати робочі процеси, зменшити кількість ручних операцій та забезпечити централізоване управління ресурсами. З кожним роком усе більше бізнесів переходять на веб-платформи, щоб відповідати очікуванням клієнтів і залишатися конкурентоспроможними.

Веб-застосунок є однією з найбільш поширених форм взаємодії з онлайн-сервісами, дозволяючи використовувати їх без встановлення додаткового програмного забезпечення. Це робить веб-застосунки дуже зручними не лише для пошуку інформації, але й для розв'язання бізнес-завдань на практиці, особливо в сфері обслуговування.

Інтернет став не лише джерелом статичного контенту, але й повноцінною платформою для дистанційної роботи через інтерактивні системи. В минулому програми встановлювалися локально, тоді як зараз лише їх логіка працює, і доступ до них здійснюється через браузер. Поточна робоча модель стала можливою завдяки використанню веб-фреймворків і стандартизованих протоколів обміну даними, таких як HTTP та REST.

Для малих бізнесів, таких як барбершопи, існує потреба автоматизації планування запису клієнтів, управління персоналом і надання послуг. Персонал може перебувати під великим тиском через зростання кількості відвідувачів, збільшення команди майстрів та негнучке адміністрування. У таких обставинах стає критично важливим використовувати цифровий інструмент, який дозволяє легко здійснювати онлайн-запис для користувача опираючись на відгуки щодо конкретного майстра.

Веб-застосунок є одним з найефективніших способів реалізації подібної системи. З його допомоги доступ до сервісу може бути здійснений з будь-якої платформи, СУБД стає централізованою, а система авторизації між клієнтами та співробітниками може легко управлятися та розширюватися на кілька філій або навіть на всю мережу закладів. Головною перевагою є можливість адаптації до змін у бізнес-процесах без повної перебудови системи. Це робить веб-застосунок не лише зручним, а й стратегічно доцільним рішенням для зростаючого бізнесу.

Сучасні фреймворки значно спростили створення таких веб-систем. Вони надають набір готових до використання компонентів (інструментів і систем), наприклад для керування БД, доступу до дозволів або обробки запитів. Наприклад, для клієнтської сторони – React з TypeScript. Вони є інструментами які дозволяють створювати адаптивні, швидкі і надійні інтерфейси. Серверна сторона побудована на ASP.NET Core – високопродуктивному, безпечному і кросплатформенному фреймворку. Дані записуються у реляційну БД MySQL, яка забезпечує цілісність даних та здатність працювати з великою кількістю записів.

Розробка веб-застосунк дає можливість не лише завершити завдання автоматизації запису, але й врахувати всі характеристики конкретного барбершопу: розклад персоналу, типи послуг та нестандартні побажання клієнтів. Це забезпечує гнучкість, і вирізняє заклад серед конкурентів завдяки чудовому цифровому обслуговуванню, яке забезпечується завдяки цьому веб-застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема організації запису та управління розкладом у барбершопах

У сучасному світі сфера послуг вимагає швидкої реакції на клієнта та добре організованих внутрішніх процесів. Однією з ключових особливостей бізнесу барбершопів є наявність легкої та прозорої системи для клієнтів, щоб записатися до барбера. Від того, наскільки ефективно організовано цей процес, залежить рівень задоволеності клієнтів, робочого навантаження на персонал та якості обслуговування загалом.

Записи традиційно управлялися вручну, через телефонні дзвінки, месенджерами або відвідуванням салону, що призводило до помилок, дублювання та втрачання записів і в кінцевому підсумку розчаровувало клієнтів бізнесу.

Найскладніше завдання – призначити клієнта барберу, враховуючи розклад, навантаження, спеціалізацію та побажань відвідувачів. Клієнти не завжди можуть передбачити, коли конкретний барбер доступний, а барбери не мають системи, яка б працювала як застосунок з їх розкладом. Це зменшує ефективність роботи і дає більше роботи для адміністратора барбершопу, який мусить оновлювати та координувати зміни вручну.

Необхідність цифрової автоматизації цих процесів стає очевидною. Вони потребують створення системи, де клієнт буде бачити простий інтерфейс, щоб вибрати майстра, дату, послугу, а потім отримати вільний час для запису до обраного барбера.

Система також повинна дозволяти барберу встановлювати особисті розклади, переглядати всі заброньовані записи та управляти вільними слотами для запису тощо. Важливо, щоб система не створювала конфліктів в розкладі, накладок чи дублювання.

Впровадження подібного веб-застосунку дає змогу централізувати управління процесами запису, зменшити кількість ручних дій з боку персоналу, підвищити точність у роботі з графіками та значно покращити взаємодію між клієнтами й майстрами. У результаті виграють усі учасники процесу: клієнт отримує комфортний і зрозумілий сервіс, барбер – зручний інструмент планування, а адміністратор – засіб ефективного управління навантаженням.

1.2 Аналіз існуючих рішень

На сучасному ринку існує багато цифрових продуктів, які автоматизують запис клієнтів у сфері послуг, в основному для перукарень і салонів краси. Більшість таких рішень реалізовані у вигляді «програмне забезпечення як послуга» (SaaS), і доступні як веб-сервіс через браузер або мобільний застосунок. В цілому такі системи можна розділити на три типи: CRM-системи для сервісного бізнесу, вузькоспеціалізовані сервіси, орієнтовані на салони, а також індивідуальні або відкриті рішення, які реалізуються як окремі проєкти.

До категорії універсальних сервісів можна віднести такі платформи, як Fresha, SimplyBook.me, Square Appointments і Vagaro. Вони підтримують функціонал запису клієнтів, управління розкладом, обліку послуг, історії відвідувань і навіть онлайн-оплати. Деякі з них можуть надавати навіть інструменти аналітики, генеруючи звіти та взаємодію з клієнтами через нагадування або маркетингові розсилки. Характерною рисою цих джерел є легкість і швидкість, з якими вони можуть бути впроваджені та мають стабільну підтримку. Але їх основними недоліками є закритість коду, обмежена функціональність і залежність від умов підписки чи тарифних планів. Системи такої природи зазвичай не можуть бути адаптовані до бізнес-логіки, пов'язаної з внутрішніми процедурами конкретної установи або системи.

Існують спеціалізовані платформи, такі як Booksy, GlossGenius, Timely та Boulevard, які задовольняють потреби перукарень і салонів, наприклад. Їхня панель управління більш зручна і вони концентруються виключно на взаємодії між майстрами та клієнтами. Деякі з них включають мобільні застосунки для клієнтів, перегляд доступного часу в реальному часі, моніторинг якості перукарів та інтеграцію календаря. Проте такі рішення не дозволяють гнучко розвивати функціональність, додавати нові бізнес-процеси або інтеграцію внутрішніх облікових і адміністративних систем.

Інша категорія – це відкриті або індивідуальні проекти, які компанії або незалежні розробники розробляють як власні продукти на основі відкритих джерел. Такі рішення призначені для вирішення специфічних проблем щодо розподілу клієнтів між майстрами, точного планування, обліку робочих годин і контролю над кількома компаніями в мережі. З правильною стратегією один єдиний веб-застосунок може забезпечити будь-яке узгодження з бізнес-процесами, можливістю підтримувати будь-які зміни функціональності, здатністю інтегруватися з сервісами, які працюють локально, та повним контролем над даними. Однак великі проекти є дорожчими, вимагають вищої навички дизайну, реальної команди розробки та підтримки протягом усього циклу проєкту.

Часом також використовуються різні загальні інструменти, такі як Google Calendar, Excel, телеграм-боти або Trello, які лише частково покривають потреби у плануванні чи нагадуваннях. Але ці інструменти не орієнтовані на бізнес і не забезпечують достатнього рівня безпеки даних, авторизації користувачів, статистики та гнучкого контролю доступу. Вони не підходять для організації роботи на етапах за межами першого рівня.

У світлі наведеної вище інформації слід зазначити, що доступні нині продукти є або занадто стандартними, або не легко піддаються адаптації під будь-який індивідуальний заклад. Тому створення власного веб-застосунку з відкритою архітектурою, власною функціональністю і масштабованістю в межах мережі перукарень – це цілком розумне і виправдане рішення.

1.2.1 Аналіз платформи Booksy

Booksy [1] – це відома платформа для автоматизації запису клієнтів, яка зосереджена здебільшого на сфері краси та догляду, зокрема, на барбершопах. Пропозиція включає два інструменти: один для клієнтів, інший для працівників салону. Це дозволяє користувачам обирати професіоналів та послуги, переглядати доступні години та записуватися онлайн, а для професіоналів це допомагає керувати своїм розкладом, забезпечувати видимість майбутніх візитів і управляти базою клієнтів з особистої панелі приладів.

Booksy зарекомендував себе як чудова альтернатива іншим програмам для складання розкладів завдяки простому та легкому в користуванні інтерфейсу, розробленому для малих і середніх обслуговуючих підприємств. Крім звичайного бронювання, сервіс надає клієнтам автоматичні нагадування про майбутні прийоми, аналітичні дані та зберігання інформації, можливість залишити зворотний зв'язок, а навіть може забезпечити зручний інтерфейс як на Android, так і на iPhone. Одна з головних переваг платформи – це її ринок: каталог послуг, де клієнти можуть знайти професіоналів у своїй місцевості, що підвищує можливості бізнесу для залучення нових клієнтів.

Проте у Booksy є один недолік. Платформа не підтримує кодинг та не дозволяє користувачам змінювати бізнес-логіку, що може бути проблемою для закладів з нетрадиційним розкладом або робочими процесами, встановленими урядом. До того ж, все складне, як-то аналітика чи певні нагадування, доступні лише для професійних користувачів, що створює труднощі для малого бізнесу. Крім того, вона менш гнучка для інтеграції зі сторонніми сервісами чи індивідуальними сайтами, і багато чого з того, що вона робить, вимагає дещо незграбної взаємодії з інтерфейсом самої системи.

Тому, хоча Booksy може бути ефективним, готовим рішенням «з коробки» для перукарень, які не можуть або не хочуть вкладати кошти у створення власного програмного забезпечення, вона недостатньо

масштабована для компаній, яким потрібен глибокий контроль над своєю цифровою платформою. Для них є сенс розробити індивідуальне веб-орієнтоване програмне забезпечення, яке ідеально впишеться в їх бізнес-операції та принесе бажані результати.

1.2.2 Аналіз платформи Boulevard

Boulevard [2] – це сучасна платформа управління бізнесом у сфері краси, яка позиціонує себе як «преміум-рішення» для салонів, барбершопів і медичних естетичних офісів. Boulevard не призначена для будь-якого бізнесу загального характеру. На відміну від більшості базових сервісів, Boulevard призначена для підприємств, які зосереджені на наданні відмінного обслуговування.

Платформа також включає різноманітні налаштування для персоналізації, від розкладу, онлайн-бронювання та управління персоналом до CRM, інтеграції POS-систем або аналітики.

Boulevard також має потужну систему бронювання, що враховує тривалість послуги та тип спеціаліста, на додаток до історії відвідувань минулими клієнтами та навіть кількість часу, відведеного на прибирання між клієнтами. Це забезпечує максимальне використання робочого часу та мінімальну непродуктивну зайнятість.

Сервіс стійкий до слабких годин бронювання та збільшує середні чеки завдяки рекомендаціям щодо часу бронювання, автоматично згенерованим нагадуванням, повторним відвідуванням і інструментам управління лояльністю для клієнтів: бонусні програми, подарункові сертифікати та знижки.

Boulevard також має красивий та інтуїтивно зрозумілий інтерфейс і надійну систему звітів, яку власники бізнесу можуть використовувати для відстеження фінансових показників, відвідуваності та продуктивності персоналу.

Але, незважаючи на свою потужність, платформа має свої недоліки. Найбільшою з них є те, що Boulevard орієнтована на англомовний сегмент, що може бути не надто добре для регіонального використання в конкретних територіях, зокрема в Україні.

Крім того, для використання всіх функцій платформи потрібен преміум-план, який може бути занадто дорогим для невеликих барбершопів або для підприємств на початковій стадії розвитку. Платформа також закрита для модифікацій – внутрішня логіка може бути змінена, а функціональність може бути розширена тільки в тому, що дозволяють розробники – доступ до вихідного коду або повного API не надається. Також необхідно вирішити питання з інтеграцією локальних платіжних систем, бухгалтерського обліку або обміну повідомленнями.

Підсумовуючи, можна сказати, що Boulevard є міцним і багатофункціональним програмним забезпеченням, яке добре підходить для середніх та великих підприємств, які працюють у сфері послуг і мають високий попит на автоматизацію та контроль якості. Однак для бізнесів, які хочуть налаштувати систему під свої внутрішні бізнес-процеси та уникнути прив'язки до закритих комерційних сервісів, може бути доцільнішою власна вебпрограма. Такий підхід дозволяє врахувати особливі вимоги певного місця, уникнути ліцензійних обмежень, забезпечити зростання системи без додаткових витрат тощо.

1.2.3 Аналіз платформи Vagaro

Одне з найпопулярніших SaaS рішень для компаній у галузі краси, здоров'я та фітнесу сьогодні – Vagaro [3]. Платформа використовується багатьма підприємствами в США, Канаді та різних інших англомовних країнах. Сервіс пропонує повний набір рішень для управління записами клієнтів, головними розкладами, онлайн-сервісами, фінансовим обліком, точкою продажу товарів і послуг та маркетинговими акціями. Vagaro

користуються малі та середні підприємства сфери обслуговування, включаючи барбершопи, завдяки широкому функціоналу та стабільній роботі.

Мультифункціональність Vagaro – безперечно, його сила: він розроблений як інструмент «все в одному», але є можливість відключити функції, які не потрібні бізнесу. Наприклад, окрім звичайного бронювання клієнтів, він дозволяє продавати товари через онлайн-магазин, генерувати подарункові сертифікати, керувати членствами клієнтів і відстежувати відвідуваність і навантаження майстра в реальному часі. Також система має розширену функціональність POS, що дозволяє обробляти продажі, друкувати чеки, приймати платежі кредитними картками та керувати податками. Іншою великою перевагою є наявність хмарного календаря та спеціального мобільного застосунка для Android і iOS, що гарантує, що клієнти та персонал можуть взаємодіяти з системою на ходу.

Хоча система багата на функції, у Vagaro є певні недоліки, які особливо критичні для деяких типів бізнесу. По-перше, цей сайт насамперед англійською мовою – ще немає повної локалізації для українських користувачів. Це додає рівень складності у використанні сервісу в районах, де персонал не говорить англійською достатньо добре. По-друге, плата за підписку залежить від кількості працівників, функціональних модулів і активних функцій, що може стати фінансовим тягарем для маленької перукарні або стартапу. Крім того, існує дуже мало можливостей для налаштування інтерфейсу та бізнес-логіки в межах платформи – користувачі обмежені функціональністю, наданою розробниками, і не можуть самостійно налаштувати або інтегрувати з іншими системами.

У підсумку, Vagaro є універсальним рішенням у сфері краси для всіх форм управління закладом, що включає планування та управління клієнтами, управління фінансами, статистикою. Тим не менш, якщо бізнес потребує можливості впровадження навіть нестандартних процесів, інтеграції з локальною мовою та гнучкішої інтеграції – слід обрати індивідуальний веб-

застосунок. Це дозволяє розробити систему, яка відповідає унікальним потребам конкретного барбершопу або мережі, і при цьому дає можливість масштабування в майбутньому.

1.3 Постановка задачі

Управління часом, розподіл робочого навантаження між співробітниками, ведення записів і підтримка певної історії взаємодії – це все частина обслуговування клієнтів, яке здійснюється в барбершопі. У невеликих приватних підприємствах ці операції іноді виконуються вручну, іноді за допомогою простих інструментів, що не забезпечують необхідної безпеки, надійності і гнучкості. Усе це можна автоматизувати за допомогою веб-застосунку, що згладжує взаємодію між клієнтами, адміністрацією та співпрацівниками, роблячи її більш організованою, ніж стандартна бізнес-структура.

Значна частина втрат часу в таких закладах відбувається через неузгодженість у записах, зміни в розкладі, дублювання запитів або відсутність сповіщень. Програмна система дозволяє синхронізувати всі дії між учасниками процесу та мінімізувати ризик людського фактору.

Веб-застосунок, на відміну від десктопного застосунку, не потребує встановлення на комп'ютер клієнта – користувач відкриває веб-браузер і запускає застосунок через посилання. Це дозволяє охопити ширший ринок, і обслуговування стає набагато простішим, особливо при роботі з мережею закладів.

Крім того, така система полегшує адміністрування в реальному часі – зміни в послугах або графіках стають миттєво доступними всім учасникам процесу без оновлення програмного забезпечення. Упровадження веб-рішення також дає змогу легко масштабувати систему з одного салону на мережу філій без потреби в розробці нової інфраструктури.

На основі виявлених недоліків існуючих рішень розробляються функціональні вимоги до майбутнього продукту. Запропонований веб-застосунок має бути здатним автоматизувати всі основні процеси, починаючи від створення записів клієнтів до управління персоналом.

Він також має передбачати ведення журналу подій, що дозволить адміністрації аналізувати історію дій користувачів і покращувати якість обслуговування. Кожен користувач матиме доступ до індивідуальної панелі управління з відповідними правами доступу залежно від його ролі в системі.

Ролі користувачів:

- клієнт – може бронювати послугу, переглядати вільний час для запису, обирати професіонала;

- барбер – може переглядати список своїх клієнтів, налаштовувати свою доступність, підтверджувати або скасовувати записи;

- адміністратор – має доступ до розкладів, послуг, керує списком записів та переглядає активність по всій системі.

Крім функціональних можливостей, система повинна бути оптимізована для мобільних пристроїв, оскільки значна частина клієнтів взаємодіє із сервісами саме зі смартфонів.

Також сайт повинен бути розроблений з урахуванням безпеки, збереження особистих даних, адаптивного дизайну та розширеності. Можливість подальшого інтегрування із сторонніми системами, такими як SMS- або email-сповіщення, також є бажаним функціоналом.

1.4 Порівняння існуючих рішень із розробленим застосунком

У процесі вивчення предметної області було проаналізовано певні доступні інструменти для автоматизації записів у салонах краси та перукарнях в таких платформах, як Fresha, Booksy, Boulevard та Vagaro. Вони надають широкий спектр функціональності – онлайн-запис, ведення клієнтської бази, адміністрування послуг тощо. Проте, незважаючи на свою

розвиненість, ці сервіси мають обмеження, що знижують їхню гнучкість у випадках, коли потрібна адаптація під специфіку окремого закладу.

Одним із найпоширеніших недоліків готових платформ є закритий код, який не дозволяє модифікувати внутрішню логіку та змінювати структуру інтерфейсу. Крім того, більшість цих систем, вимагають щомісячної підписки, яка може бути занадто дорогою для малих бізнесів. Багато сервісів, особливо міжнародних, не мають українських локалізацій, що ускладнює їх використання в нашій середі.

На відміну від комерційних SaaS-продуктів, розроблений у рамках цієї роботи веб-застосунок є відкритим, гнучким та адаптованим рішенням для однієї або мережі барбершопів. Він розроблений з використанням сучасного технологічного стека (React та TypeScript, ASP.NET Core, MySQL), щоб забезпечити високу продуктивність, універсальний інтерфейс та повну можливість для розширення функціональності в майбутньому без залежності від зовнішніх API або інфраструктури.

Ще однією перевагою є можливість дуже точно визначати ролі користувачів: клієнтів, перукарів та адміністраторів, які можуть мати доступ до різних підкатегорій функцій та використовувати спеціальну панель управління. Вони мають розроблену систему управління розкладом, яка дозволяє перукарю встановлювати свою доступність, дозволяє адміністратору бачити загальну картину записів та контролювати активність системи

Таким чином, розроблений застосунок вигідно вирізняється на тлі існуючих аналогів завдяки поєднанню гнучкості, відкритості, дешевизні, локалізації, простоти в обслуговуванні та можливості розширення, що робить його конкурентоздатною альтернативою для малого та середнього бізнесу в сфері надання послуг.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Огляд технологій розробки веб-застосунків

Сучасні веб-застосунки базуються на архітектурі клієнт-сервер і ділять логіку на дві основні частини – на сторону клієнта та серверну частину, також або frontend і backend [4]. Клієнт виконує роль інтерфейсу, що використовується користувачем, а сервер обробляє запити, обробляє дані і займається збереженням даних. Зазвичай такі можливості реалізуються через два програмні компоненти, які взаємодіють один з одним віддалено через мережу, використовуючи протоколи HTTP або HTTPS.

Клієнтська частина веб-застосунку складається здебільшого з трьох компонентів: HTML для вмісту, CSS для стилю та JavaScript для інтерактивності [5]. Для створення складних і масштабованих інтерфейсів, частіше використовуються фреймворки на основі JavaScript, такі як React, Vue.js або Angular, які зараз використовуються все частіше. У цьому проєкті React використовується разом з TypeScript – типізований, стабільний та зручний у підтримці код. React спрощує створення модулів з архітектурою на основі компонентів, що робить їх ідеальними для повторно використовуваних елементів інтерфейсу користувача, таких як реєстраційні форми, списки сервісів та календарі розкладів [6].

Серверна частина застосунку розроблена на основі ASP.NET Core – швидкого, сучасного та продуктивному фреймворку від Microsoft для створення кросплатформенних веб-застосунків. Його відмінними особливостями є це підтримка SQL і REST, безпроблемна інтеграція з БД та функції безпеки API. Логіка обробки даних реалізується за допомогою мови C#, яка пропонує високу типізовану безпеку принципи об'єктно-орієнтованого дизайну [7].

Для збереження даних використовується реляційна СУБД MySQL через її стабільність і простоту в адмініструванні. Застосунок взаємодіє з

базою даних через Entity Framework Core яка дозволяє працювати з таблицями БД як із класами C# замість написання запитів, а також надає міграції для плавних змін у схемі БД.

Під час розробки веб-застосунків значну роль відіграє вибір середовища розробки. Одними з найпоширеніших IDE для відповідних технологій є: Visual Studio (ASP.NET Core) і Visual Studio Code (фронтенд) [8]. Ці середовища мають такі функції, як підсвічування синтаксису, відлагодження, автодоповнення коду, інтеграція з системами контролю версій та інші утиліти, які допомагають і пришвидшують розробку.

Таким чином, веб-застосунок, розроблений з використанням новітніх технологій для фронтенду та бекенду, може бути адаптований до потреб малих і середніх бізнесів у сфері послуг.

2.1.1 Мова програмування C#

C# – це потужна та гнучка мова програмування, яку можна використовувати для написання надійних та адаптивних програм для вашого бізнесу або організації на .NET framework. Вона використовує об'єктно-орієнтовану парадигму, схожу на Java, C++ та інші мови високого рівня, і синтаксис мови легкий для розуміння як для початківців, так і для досвідчених користувачів.

Ця мова використовується при створенні веб-застосунків для керування послугами в барбершопі, які ми будемо розвивати впродовж цього підручника, щоб застосувати серверну логіку, використовуючи ASP.NET Core framework. C# добре спроектований і підтримує всі сучасні функції мови, що дозволяє писати ефективний, підтримуваний і безпечний код. Автоматичне очищення пам'яті через систему збору сміття дозволяє уникнути витоку пам'яті та помилок пам'яті через неправильні операції з управління об'єктами. Мова має низку функціональних особливостей, включаючи делегати, події і лямбда-вирази та технологію LINQ, яка

забезпечує можливість запиту та маніпуляцій зі структурованими даними, використовуючи синтаксис запитів до бази даних, та дозволяє розробникам працювати з колекціями інтуїтивно та декларативно. [9]

C# додає підтримку для різних моделей компіляції. Стандартна JIT-компіляція дозволяє оптимізувати виконання програми під час її запуску. Також можна використовувати попередню компіляцію, що дуже корисно для налаштування продуктивності при обмежених ресурсах або для створення автономних сервісів.

Це тому, що мова є дуже гнучкою і легко вбудовується в .NET Core. C# широко використовується для написання веб-сервісів, REST API та веб-застосунків, але також застосовується в хмарних, мобільних та ігрових розробках.

В проекті веб-застосунку барбершопу серверна частина розроблена мовою C#, серверна частина забезпечує обробку користувацьких запитів, контроль БД, авторизацію користувачів, а також підтримку стійкої взаємодії між клієнтом та сервером. Він також користується широким використанням і підтримкою спільноти, а також подальшим розвитком, що є надійною основою для реалізації комерційних та прикладних програмних рішень.

2.1.2 React

React – це відома бібліотека JavaScript з відкритим вихідним кодом, створена компанією Meta для створення інтерфейсів користувача в Інтернеті. Найбільшою особливістю React є його компонентний дизайн, що означає, що можна розділити інтерфейс користувача на невеликі багаторазові частини з логікою і станом. Це спрощує масштабування, обслуговування та повторне використання коду в одному проєкті або в кількох частинах системи.

React застосовується до клієнтської частини при створенні веб-застосунка для управління послугами барбершопу. За допомогою React створюється динамічний інтерфейс, який змінюється у відповідь на дії

користувача без перезавантаження всієї сторінки. Це забезпечує високу швидкість роботи, зручність та плавність у роботі. Користувач може в реальному часі бачити, які часи доступні для запису, переходити від одного барбера до іншого, додавати або відміняти записи, і все це відбувається миттєво завдяки системі віртуального DOM, що є основою React.

Бібліотека чудово поєднується з мовою TypeScript, забезпечуючи статичну типізацію та зручність у роботі з кодом. Це зменшує класи помилок, покращує автозавершення у середовищах розробки (IDE) та робить структуру компонентів зрозумілішою. Крім того, React має багате середовище бібліотек, які відповідають за маршрутизацію, керування станом, валідацію форм, API-комунікацію [10].

React також легко інтегрується з RESTful API, що дозволяє ефективно працювати з серверною частиною, реалізованою на ASP.NET Core. Взаємодії клієнт/сервер – наприклад, відправка запиту на створення зустрічі або перевірка запропонованих послуг, здійснюються через HTTP-запити в JSON-пакетах, що відповідає сучасним практикам веб-розробки.

Завдяки своїй універсальності, легкості розробки та стабільному розвитку, React є чудовою платформою для створення сучасних односторінкових застосунків, особливо тих, які мають високу інтерактивність та адаптивний дизайн.

2.1.3 ASP.NET Core

ASP.NET Core – це сучасний веб-фреймворк для створення високопродуктивних, кросплатформених веб-застосунків і хмарних сервісів. Він є наступником ASP.NET і побудований на Core.NET, тому застосунки можуть працювати не тільки на Windows, але й на Linux та macOS, що чудово підходить для розгортання систем у хмарному або іншому серверному середовищі.

У рамках кваліфікаційної роботи ASP.NET Core використовується для розробки серверної частини (бекенду) веб-застосунків, яка управляє взаємодією між клієнтом і БД. Саме через цей фреймворк обробляються HTTP-запити, забезпечується авторизація та автентифікація користувачів, виконується обробка даних та взаємодія з БД MySQL. Його архітектура враховує модульність – багато з функціональностей розроблено як сервіси, що взаємодіють з іншими сервісами, щоб уникнути надмірних залежностей і мати максимальний контроль над внутрішніми процесами.

Однією з ключових переваг є вбудована підтримка ASP.Core для шаблону MVC, який розділяє можливості між презентаційною, логічною та інформаційною моделями. Це значно спрощує структуру застосунку, його підтримку та можливість паралельної роботи над його частинами. Крім того, фреймворк підтримує шаблон проектування проміжного програмного забезпечення, що дозволяє налаштувати обробку HTTP-запитів завдяки іншим відповідним етапам.

ASP.NET Core має функції для реалізації автентифікації JWT (JSON Web Token), обмеження доступу на основі ролей, запобігання міжсайтовим підробкам запиту (CSRF-атаки), спільному використанню ресурсів з різних джерел і SQL-ін'єкцій. Таким чином, можна реалізувати надійну систему безпеки, що є критично важливим для веб-застосунка, який містить персональні дані, такі як контактні дані, реєстрації послуг, історії взаємодій.

Ще однією перевагою є тісний зв'язок між ASP.NET Core та ORM Entity Framework Core, який можна використовувати для взаємодії з БД на рівні об'єктів. Це значно спрощує роботу зі створення запитів та з самими даними, без написання SQL з нуля.

Отже, ASP.NET Core – це надійний вибір для створення серверної частини веб-застосунку, що мають усі можливості для того, щоб забезпечити гнучкість, безпеку, масштабованість та високу продуктивність. Це також є причиною, чому ця технологія є найкращою для розробки функціоналу у проекті автоматизації запису клієнта на прийом у мережі барбершопів.

2.1.4 MySQL

MySQL є однією з найбільш широко використовуваних реляційних систем управління даними [11]. Будучи відкритим вихідним кодом, це найкращий вибір для веб-проєктів. Він відповідає стандарту SQL для визначення даних, зміни даних і, в певній мірі, доступу до даних. MySQL є доброю опцією для серверних застосунків, яким потрібно зберігати та витягувати великі обсяги даних зі складними зв'язками між сутностями.

При розробці застосунка для ведення бази записів барбершопу, що використовується для вирішення базових завдань, було використано MySQL для збереження всіх основних даних: дані користувачів, майстрів, дані клієнтів, послуги, розклади, дані для клієнтів, історію відвідувань та інші. На основі реляційної моделі даних користувач може легко створювати логічні зв'язки між таблицями (наприклад, пов'язати записи клієнта з певним майстром і типом послуги), що важливо для функціонування сервісу.

Однією з ключових характеристик MySQL є те, що вона має добру продуктивність для «класичних» CRUD-операцій (створення, читання, оновлення, видалення), таким чином, добре підходить для веб-застосунків з високою конкуренцією. Підтримка індексів, зовнішніх ключів, унікальних обмежень і транзакцій, які забезпечують цілісність і узгодженість даних. Ці характеристики дозволяють не лише зберігати дані, але й виконувати складні операції в БД без переривання логіки.

MySQL добре інтегрується з платформою ASP.NET Core через використання Entity Framework Core, що робить можливим роботу безпосередньо з БД на об'єктному рівні, не вдаючись до сирих SQL-запитів. Доступ до даних може бути розроблений у вигляді класів C#, використовуючи ORM (Object-Relational Mapping), що підвищує продуктивність розробки, безпеку і зберігає систему здатною до підтримки в майбутньому.

Ще один плюс на користь MySQL – це зручність використання: як популярна БД, MySQL підтримується широкою спільнотою користувачів, великою кількістю документації і графічними інструментами (наприклад, MySQL Workbench або phpMyAdmin), що означає, можна швидко перевірити структуру таблиць, редагувати записи тощо, коли це необхідно.

Таким чином, вибір MySQL як основної СУБД проєкту дає можливість реалізувати стабільну обробку даних, створення будь-якої конфігурації БД, надійне забезпечення зберігання інформації та швидке масштабування при навантаженні або зростанні проєкту.

2.1.5 Entity Framework Core

Entity Framework Core (EF Core) – це написане з нуля ORM компанії Microsoft, яке має на меті забезпечити відкриту платформу .NET. Це дозволяє розробникам працювати з БД, використовуючи об'єкти .NET, без необхідності писати громіздкі SQL-запити для маніпуляції даними. Це значно спрощує роботу з даними, допомагає зберігати код чистим і мінімізує ризик помилок при прямому доступі до БД [12].

У середовищі веб-застосунку для мережі барбершопів, EF Core відокремлює бізнес-логіку, реалізовану за допомогою ASP.NET Core, від БД MySQL. Усі системні сутності: користувачі, майстри, послуги та розклади змодельовані як класи C#, які EF Core відображає на таблиці БД. Цей метод уникає надмірності структури даних у коді й БД і цим самим знижує вартість обслуговування проєкту.

EF Core надає сильну підтримку для міграції БД, що допомагає змінити структуру БД, не втрачаючи дані. Все це можна синхронізувати з реальними таблицями, використовуючи деякі спеціальні команди, що дуже корисно при роботі в команді або оновленні системи. ORM також забезпечує автоматичне створення БД під час першого запуску проєкту, що зводить до мінімуму час початкового розгортання.

Однією з інших великих переваг EF Core є підтримка LINQ (інтегрований в мову запит), що дозволяє писати запити до БД у стилі C# з легким для читання, написання та розуміння форматом. Це дозволяє швидко отримати доступ до потрібних даних, сортувати, фільтрувати та групувати їх без необхідності створення складних SQL-запитів. Крім того, EF Core забезпечує, що запити виконуються з хорошою продуктивністю, навіть з мільйонами записів у системі.

Він також може працювати з шаблонами проєктування, такими як Repository (сховище) або Unit Of Work (одиниця роботи), де останній призначений для логічного групування доступу до даних, підвищення тестованості компонентів і підтримуваності великого коду. Це означає, що реальну БД можна замінити на InMemory (у пам'яті) тільки для тестів без запуску всього серверного оточення.

Отже, застосування EF Core в проєкті гарантує комфортний і безпечний доступ до БД, повний контроль над тим, що було додано або прочитано, скорочення часу виходу на ринок, усунення потенційних помилок з некоректним SQL, усталену платформу для серверної частини веб-застосунків.

EF Core має вбудовану підтримку відстеження змін і за замовчуванням визначає, які властивості об'єкта фактично змінилися, і лише ці оновлення застосовує назад до бази даних. Це також підвищує продуктивність і дозволяє уникнути конфліктів, коли кілька користувачів одночасно отримують записи. Завдяки гнучкій системі конфігурації та можливості керувати відносинами один-до-багатьох, багато-до-багатьох та іншими, програміст може створити складну систему відносин між сутностями без необхідності глибоко занурюватися в SQL-представлення Також EF Core дозволяє застосовувати Fluent API або анотації для точного контролю над схемою БД, включно з обмеженнями, індексами та правилами зв’язності, що особливо актуально в масштабованих проєктах.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

3.1 Огляд структури проєкту

Функціональні аспекти реалізації застосунку побудовані за принципом модульності та розділення інтересів, що є типовим для сучасних веб та серверних застосунків. Проєкт поділений на дві основні частини: серверну частину, або бекенд, і клієнтську частину – фронтенд. Вони знаходяться у відповідних папках. На рисунку 3.1 представлено структуру серверної частини, а на рисунку 3.2 – клієнтської частини.

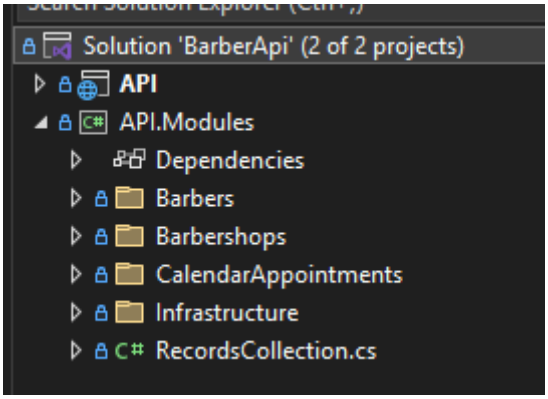

Рисунок 3.1 – Структура серверної частини проекту

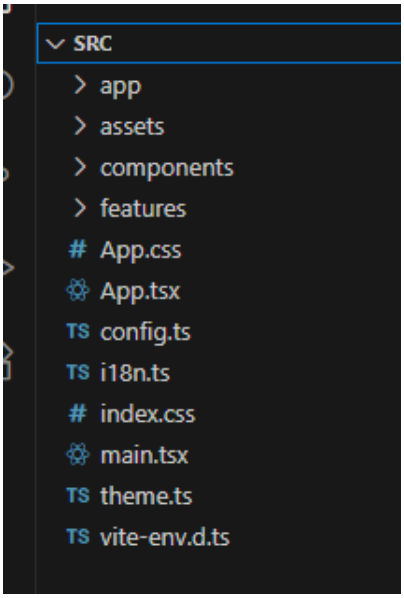

Рисунок 3.2 – Структура клієнтської частини проекту

Серверна частина розміщена в папці api/ і має архітектурну структуру з різними модулями для кожної доменної області (наприклад, Barbers, Barbershops, CalendarAppointments). Кожен модуль містить власні контролери, моделі, сервіси, репозиторії та обробники виключень, що дає можливість розділити бізнес-логіку і зберегти підтримку проєкту завдяки його масштабованості. Операції з БД, зберігання та отримання сутностей з контекстами і міграціями відбуваються у своїх власних шарах.

Фронтенд знаходиться в папці web і розроблений за компонентним підходом. Усі функціональні блоки (наприклад, робота з майстрами, календарем, аутентифікацією) розміщені в окремих підкаталогах з компонентами, сторінками, API-сервісами і допоміжними функціями. Це має перевагу чистої ієрархії і ненасиченої навігації по коду, що також полегшує додавати більше функціональності.

Загалом, така організація структури проєкту сприяє підтримці чистоти коду, забезпечує незалежність окремих компонентів та дозволяє ефективно розвивати застосунок у майбутньому без ризику виникнення критичних помилок при внесенні змін.

3.2 Каталог API.Modules

Каталог API.Modules центральним елементом серверної частини застосунку, тут зібрані основні функції модулів. У цьому каталозі організовані підкаталоги, де кожен відповідає за окрему доменну область, наприклад, барбери, барбершопи, календарні записи тощо. Така структура логічно розділяє код за напрямками операцій, що полегшує його перегляд, тестування і подальший розвиток проєкту.

Окрім доменних модулів, у каталозі є спільний інфраструктурний код, який забезпечує взаємодію з БД, прив'язки залежностей та інших загальні сервіси. Завдяки цьому, каталог API.Modules виступає своєрідним каркасом, на якому будується уся бізнес-логіка серверної частини застосунку.

Підкаталог Barbers містить усі компоненти, необхідні для роботи з інформацією про барберів. Він організований так, щоб забезпечити чітке розділення операційних відповідальностей між різними частинами модуля. Контролери і контракти розміщені в підкаталозі API, вони обробляють прийом, обробку та передачу даних між клієнтом і сервером. Моделі, що описують структуру об'єктів барбера можуть бути знайдені в підкаталозі з такою назвою, а також репозиторії для доступу до даних і взаємодії з БД.

Окремо виділені класи для обробки виняткових ситуацій, що дозволяє ефективно управляти помилками, пов’язаними з цим модулем. Така структура каталогу також робить зручною підтримку, розширення функцій та покращення надійності всієї системи.

Підкаталог Barbershops спрямований на організацію всіх компонентів, що стосуються управління інформацією про барбершопи. Його структура побудована за аналогією з іншими доменними модулями для забезпечення єдності правопису та підтримки коду.

У підпапці API розміщені контролери, які обробляють запити, що стосуються барбершопів, та контракти, які визначають дані, що повинні передаватися між клієнтом і сервером. Моделі описують основні властивості барбершопів, а репозиторії реалізують логіку доступу до даних та взаємодію з БД.

Для обробки специфічних помилок, які можуть виникати під час роботи з барбершопами, передбачені окремі класи-винятки. Ця організація каталогу дає архітектурну чистоту, спрощує розширення функціональності та забезпечує зручність у супроводі й тестуванні модуля.

Усе, що стосується записів у календарі та управління зустрічами, знаходиться в підкаталогі CalendarAppointments. Її архітектура розроблена таким чином, щоб розділити обов'язки між компонентами модуля.

У директорії API розташовані контролери для створення, редагування, скасування, перегляду та отримання всіх подій календаря. Тут також знаходяться контракти, які визначають формат даних, що обмінюються між

клієнтом і сервером. Моделі представляють основні характеристики запису в календарі, а репозиторії містять логіку запитів і доступ до БД.

Для обробки виняткових ситуацій в управлінні подіями календаря (такими як зміна скасованої зустрічі, додавання неіснуючого учасника), у рішенні передбачені класи винятків. Така специфічна організація директорії полегшує додавання нових функцій, добре підтримує чистоту коду і робить модуль підтримуваним і тестованим.

Підкаталог Infrastructure включає загальні елементи інфраструктури, які використовуються для запуску основних модулів програми. Тут знаходиться набір класів, відповідальних за налаштування залежностей, реєстрацію сервісів і організацію доступу до сховища даних. Ця директорія, серед іншого, містить код підключення до БД, а також допоміжні сервіси, які використовуються різними модулями проєкту.

Розміщення інфраструктурних елементів в окремому каталозі означає, що налаштування централізовані, а спільними ресурсами легше керувати. Це додає гнучкості архітектурі, полегшує масштабування, а також спрощує обслуговування та розширення програми.

3.3 Каталог web

Каталог web є клієнтською частиною застосунку, яка обробляє інтерфейс користувача та взаємодіє з серверною частиною за допомогою API. Структура цього каталогу побудована за принципами сучасної фронтенд-розробки з використанням компонентного підходу.

В основі каталогу знаходяться основні конфігураційні файли, налаштування збірки, залежності та налаштування запуску проекту. Логіка, яка керує застосунком, розташована в папці src, і кожен функціональний блок розміщений у власній папці тут. Це робить код зрозумілим і легким у орієнтуванні, полегшує додавання нових функцій та підтримує чистоту архітектурного коду.

Окремо виділені папки для стилів, ресурсів, утиліт та мовних файлів, що дозволяє гнучко змінювати зовнішній вигляд, локалізувати та повторно використовувати код. Завдяки цій організації каталог web сприяє належному розвитку клієнтської частини застосунку, оскільки він представляє зручний та інтуїтивно зрозумілий інтерфейс для користувача.

3.4 Каталог features

Каталог features у складі клієнтської частини проєкту призначений для організації основних функціональних можливостей застосунку. Кожна підпапка в цьому каталозі представляє окремий напрямок або частину функціоналу (автентифікація, робота з майстрами, барбершоп, календар, переклад тощо), перелік усіх підкаталогів зображений на зображені 3.3.

У межах кожного модуля зберігаються відповідні компоненти, сервіси для взаємодії з API, типи даних, утиліти та файли стану. Це дозволяє розділити логіку кожної функції функціональним чином і полегшити її обслуговування, тестування та розширення.

Дизайн каталогу функцій дозволяє модульність коду, повторне використання та легке масштабування клієнтської частини застосунку. Це сприяє підтримці чистоти архітектури та забезпечує зручність у роботі над проектом.

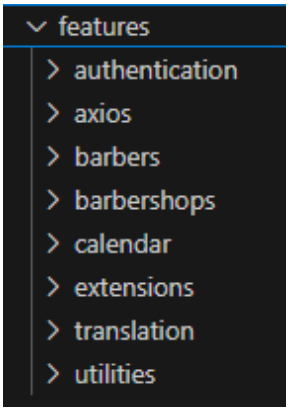

Рисунок 3.3 – Структура підкаталогів features

Першим варто описати підкаталог utilities, він містить допоміжні функції та сервіси, які використовуються в різних частинах клієнтської частини застосунку. Він містить загальні утиліти, які не пов'язані з конкретним функціональним модулем, але допомагають виконувати стандартні операції, такі як збереження даних у локальному сховищі, обробка рядків, форматування даних тощо.

Наявність папки утиліт запобігає дублюванню коду між функціями, дозволяє легше отримувати доступ до допоміжного коду та заохочує до хорошої модульності коду. Таким чином, будь-яка частина або модуль застосунку може легко імпортувати необхідну функцію з каталогу утиліт і легко розробляти або підтримувати клієнтську частину застосунку.

Підкаталог translation призначений для організації багатомовної підтримки клієнтської частини застосунку. Ця папка містить файли з перекладами на конкретні мови, а також частини та логіку для зміни мови інтерфейсу та динамічного завантаження текстів.

Тут розташовані окремі папки з текстовими ресурсами на наданих мовах, що полегшує додавання нових перекладів або редагування існуючих. Каталог також може включати елементи, які забезпечують вибір мови користувачем та механізми управління станом локалізації.

Завдяки такій конструкції каталог перекладів дозволяє зручно вбудовувати концепцію багатомовності в програмне забезпечення та робить підтримку та розширення списку мов застосунку зручним і зрозумілим для кінцевих користувачів, які говорять різними мовами.

У підкаталогі extensions містяться розширення для стандартних типів або функцій, які використовуються в клієнтській частині застосунку. ут зберігаються додаткові методи або утиліти, що розширюють можливості базових об’єктів TypeScript, наприклад, роботу з рядками, масивами або іншими структурами даних.

Розширення дозволяють додавати функціональність до стандартних нових типів без дублювання цього коду по всьому проєкту. Це означає, що

розробка стає легшою, повторне використання коду збільшується, а архітектура стає чистішою.

Каталог extensions робить код більш читабельним і гнучким, оскільки вся додаткова функціональність для вбудованих типів зібрана в одному місці, і кожну з них можна просто використовувати в будь-якому модулі застосунку.

Підкаталог calendar містить усі компоненти, типи та логіку, пов’язані з відображенням і керуванням календарем та календарними подіями у клієнтській частині застосунку. У цьому каталозі зберігаються компоненти для відображення календаря, діалогові вікна для створення та редагування подій, а також допоміжні файли для роботи з датами та статусами подій.

Крім візуальних компонентів, є також типи даних, допоміжні методи для форматування дати, API-сервіс для взаємодії з записами календаря та все, що може знадобитися для синхронізації з клієнтським станом календаря. Така архітектура дозволяє зберігати всю логіку, пов'язану з календарем, разом в одній папці, що значно полегшує розробку, тестування та розширення функціональності.

Завдяки цьому каталог calendar забезпечує зручну організацію коду, підвищує модульність та сприяє підтримці чистої архітектури клієнтської частини застосунку.

Підкаталог barbershops призначений для накопичення всіх сутностей, типів, методів для роботи з барбершопами на стороні клієнта застосунку. Він містить сервіси для роботи з API для отримання, створення, оновлення та видалення інформації про барбершоп. Це також може включати типи даних, утиліти та допоміжні функції, які можуть бути корисними для роботи з інформацією про барбершоп. Така структура дозволяє відокремити функціональність барбершопів, це зручно для тестування, розробки та подальшого додавання нових функцій до застосунку. Завдяки виділенню окремого каталогу для роботи з барбершопами, досягається модульність

коду, підвищується його читабельність і підтримується чиста архітектура клієнтської частини проєкту.

Підкаталог barbers містить все, що застосунок повинен знати про майстрів у фронтенд-частині застосунку: від компонентів, типів і логіки до відображення або управління даними, пов'язаними з майстрами. Цей каталог призначений для зберігання списку майстрів та їх відповідних рейтингів і відгуків, а також діалогів для бронювання послуг.

Окрім візуальних компонентів, реалізація також надає сервіси для взаємодії з API, типи даних, утиліти та управління станом, щоб гарантувати отримання, оновлення та збереження даних про майстрів. Така організація дозволяє ізолювати функціонал, пов’язаний із барберами, що спрощує розробку, тестування та подальше розширення можливостей застосунку.

Завдяки структурі каталогу barbers досягається модульність, підвищується читабельність коду та забезпечується чиста архітектура клієнтської сторони проєкту.

Підкаталог Axios містить налаштування та утиліти для запитів API на клієнтській частині застосунку. Основним елементом цього каталогу є конфігурація екземпляра бібліотеки Axios, яка використовується для взаємодії з серверним API. У цьому каталозі зберігаються файли, що визначають базові параметри підключення, обробку помилок, додавання заголовків авторизації та інші аспекти мережевої взаємодії. Централізоване налаштування HTTP-клієнта дозволяє забезпечити єдиний підхід до виконання запитів у всіх модулях застосунку, спростити обробку типових ситуацій та підвищити безпеку передачі даних. Завдяки виділенню окремого каталогу для роботи з Axios, досягається повторне використання коду, та забезпечується чистота архітектури клієнтської частини проєкту.

Підкаталог authentication містить все, що стосується компонентів, типів і логіки для користувача на стороні клієнта. Тут зберігаються сервіси для взаємодії з API автентифікаці, механізми керування станом користувача, а також типи даних і контракти, що визначають структуру запитів та

відповідей для реєстрації, входу, виходу та інших операцій з обліковими записами.

Цей каталог також може містити утиліти для збереження токенів, компоненти для обробки сесій користувачів і логіку для відображення помилок аутентифікації. Така організація дозволяє повністю відокремити все, що стосується аутентифікації, що полегшує розробку та тестування та розширення застосунку. Завдяки структурі каталогу authentication він є модульним, безпечним і підтримує архітектуру проекту на стороні клієнта в належному стані.

3.5 Каталог components

Каталог components включає багаторазові інтерфейсні компоненти, які складають інтерфейс користувача клієнтської частини застосунку. Кожен підкаталог або файл у каталозі є незалежним функціональним блоком або елементом інтерфейсу (сторінки, що показують майстрів, календарі, панелі навігації, форми для введення даних, діалогові вікна тощо). Структура підкаталогів components зображена на зображені 3.4.

Компоненти легко зрозуміти, оскільки вони розділяють завдання – кожен виконує конкретну роботу і може бути використаний у різних частинах застосунку. Це усуває дублювання коду, полегшує розробку та тестування і робить легким підтримку інтерфейсу та додавання нових функцій до нього.

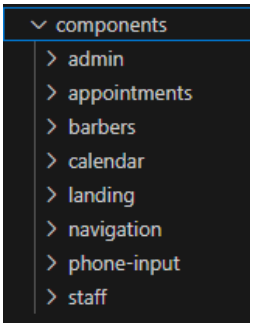

Рисунок 3.3 – Структура підкаталогів components

Інтерфейсні компоненти для роботи персоналом барбершопу включені в каталог staff у клієнтській частині застосунку. Він містить функції для відображення списку співробітників, показу детальної інформації про кожного члена команди та всі операції з управління персоналом.

Цей каталог має інтеграцію з відповідними сервісами для отримання даних про персонал та їх модифікації, а також містить форми для введення даних і діалогові вікна для підтвердження дій, таких як записи клієнтів. Така організація дозволяє відокремити функції, пов'язані з персоналом, і призводить до краще структурованого коду, а також полегшує розширення можливостей інтерфейсу.

Підкаталог phone-input містить компоненти для введення та обробки номерів телефонів у клієнтській частині застосунку. Основний компонент цього каталогу забезпечує зручний інтерфейс для введення номеру телефону, включаючи валідацію формату, автоматичне додавання коду країни та підказки для користувача.

Використання окремого каталогу для цього функціоналу дозволяє централізовано зберігати цю функціональність у спеціальному каталозі, де можна легко налаштувати логіку номерів телефонів. Це також полегшує підтримку та вдосконалення функціональності цього компонента, наприклад, шляхом додавання додаткових форматів або локалізацій.

Підкаталог navigation включає елементи для організації навігації в клієнтській частині застосунку. Це охоплює будь-які елементи інтерфейсу, що використовуються для переміщення між сторінками або розділами, такі як: заголовок сайту, меню, навігаційна панель тощо. Компоненти цього каталогу виконують логіку відображення сторінок в активному стані, здійснюють перехід і можуть мати логотипи, кнопки входу/виходу, перемикачі мов тощо. Модульність дозволяє легко налаштовувати та розширювати інтерфейс, не розбираючись у заплутаній файловій структурі, щоб додати одне маленьке посилання. Колекція навігації сайту значно покращує зручність використання, роблячи інтерфейс дружнім до

користувача та доступним, враховуючи легкість навігації між основними функціями застосунку.

Підкаталог landing складається з компонентів, що впливають на вміст головної (привітальної) сторінки застосунку. Тут розміщуються елементи інтерфейсу, які формують перше враження користувача: інформаційні блоки, банери, заклики до дії, короткі описи про сервіс та його переваги. Частини цього опису можуть включати адаптивні дизайнерські особливості для користувача, динамічний контент та швидкі інструменти навчання, щоб ознайомити користувачів з функціями застосунку. Розміщуючи landing-сторінки сторінку в іншому каталозі, ми маємо можливість розробляти, тестувати та підтримувати вміст головної сторінки незалежно від решти сайту. Каталог landing-сторінок служить для створення привабливого та інформативного початкового екрану, щоб залучити користувачів і переконати їх скористатися сервісом.

Підкаталог calendar включає елементи для представлення та керуванням календарем у клієнтській частині. Він містить елементи дизайну для попереднього перегляду календаря, створення подій, редагування та перегляду інтерфейсу, діалогові вікна для кожного запису пристрою тощо.

Елементи цього каталогу охоплюють логіку того, як представлені дати, як вибирається день, як модель пов'язана з іншими модулями (наприклад, бронювання), а також можуть включати стиль для загального представлення календаря. Ця структура дозволяє інкапсулювати всю функціональність календаря, що полегшує розробку та тестування, а також додавання нових функцій інтерфейсу в майбутньому.

Завдяки структурі каталогу календаря можлива модульність, повторне використання коду, а клієнтська частина проекту підтримується для чистої архітектури.

Підкаталог barbers включає колекцію інтерфейсних компонентів для відображення інформації, пов'язаної з майстрами, та взаємодії з нею у клієнтському застосунку. Він містить списки майстрів, екран деталей з

бічними меню, форми відгуків, діалогові вікна для бронювання, та відображення рейтингу майстра. З можливістю оглянути написані відгуки, які описані в іншому казалозі.

Кожна частина каталогу має окрему функцію: наприклад, для перегляду повної інформації про майстра, замовлення послуги, написання відгуку. Це допомагає повторно використовувати компоненти в різних частинах інтерфейсу.

Завдяки такій структурі забезпечується модульність і легкість супроводу коду. Компоненти легко масштабуються та адаптуються під нові вимоги проєкту. Це сприяє підтримці чистої архітектури та підвищує зручність розробки клієнтської частини застосунку.

Підкаталог appointments у складі components включає інтерфейсні компоненти для управління записами з клієнтської частини застосунку. Він містить компоненти для відображення списку записів, деталізацію окремих прийомів, а також взаємодію користувача з функціоналом створення, редагування чи скасування запису.

Елементи в цього каталогу реалізують логіку, яка дозволяє переглядати майбутні та минулі призначення, інтегруватися з календарем і можуть містити діалоги, такі як підтвердження дій або зміна чогось. Така структура дозволяє інкапсулювати функціональності, пов'язані з призначеннями, повторне використання коду та легкість розширення функціональностей інтерфейсу.

Підкаталог адміністратора (admin) складається з частин, призначених для побудови адміністративної частини застосунку. Це включає функції, які дозволяють адміністраторам мати доступ до розширених функцій управління системою, таких як: управління користувачами, перегляд статистики, налаштування сервісів, модерація контенту тощо.

Компоненти цього каталогу може також включати панелі управління, таблиці даних, форми для зміни налаштувань та інші утиліти, корисні для контролю системи. Винесення адміністративних аспектів функціональності в

окремий каталог дозволить зберегти код чистим, покращити безпеку та полегшити роботу зі створення та тестування функціональності, унікальної для адміністраторів.

3.6 Каталог assets

Каталог assets включає статичні файли, що використовуються у фронтенд-частині застосунку для забезпечення візуального оформлення, а також покращення користувацького досвіду. До цього входять зображення, іконки, логотипи, SVG, а також інші медіа, які можна використовувати в компонентах та інтерфейсах.

У межах проєкту для мережі барбершопів у цьому каталозі розміщуються, зокрема, стилізовані іконки для кнопок запису, фотографії салонів, аватари майстрів та графічні елементи інтерфейсу. Завдяки виділенню окремого каталогу для статичних файлів проєкт краще організований, легко шукати та замінювати ресурси, а також це дозволяє мати медіа-елементи в різних частинах застосунку. Це допомагає підтримувати чисту архітектуру та дозволяє оновлювати візуальний контент без необхідності змінювати основний код інтерфейсу. У випадку редизайну достатньо лише замінити відповідні файли в assets, не змінюючи логіку компонентів.

Крім того, централізоване зберігання медіа-файлів спрощує командну роботу над проєктом. Візуальна узгодженість між різними сторінками застосунку досягається за рахунок повторного використання єдиної графіки з каталогу. Додавання нових ресурсів не впливає на логіку застосунку, що підвищує стабільність розробки. Такий підхід також сприяє швидшому впровадженню змін у дизайні та підвищує гнучкість у роботі з візуальними елементами.

3.7 Каталог app

Каталог app містить основні файли, що відповідають за ініціалізацію, конфігурацію та глобальний стан клієнтської частини застосунку. Він містить такі частини, як налаштування зберігання глобального стану (наприклад, сховище Redux), хуки для роботи з цим станом та інші службові файли, які гарантують правильну роботу всього застосунка.

У рамках реалізації проєкту для мережі барбершопів у цьому каталозі також розташовано конфігурацію тем оформлення інтерфейсу, глобальні сервіси для сповіщень та обробки помилок.

Матеріали в цій директорії зазвичай підключаються на верхньому рівні ієрархії застосунка (наприклад, забезпечують зв'язки між модулями, обробляють глобальні події, зберігають налаштування в одному місці). Така структура сприяє централізації основних функціональних аспектів застосунка, полегшуючи масштабування та обслуговування коду.

Завдяки єдиному джерелу глобальних параметрів спрощується передача конфігурацій, таких як токени автентифікації, мова інтерфейсу та налаштування розкладу. Це особливо корисно для підтримки різних ролей користувачів – клієнтів, барберів та адміністраторів.

Оскільки ми домовилися, що код розміщується в директорії застосунка, ми досягаємо більш щільної архітектури, зручнішої розробки та того, що клієнтська частина проєкту буде працювати. Це також дозволяє швидко впроваджувати зміни у глобальній логіці без необхідності модифікувати окремі модулі.

Завдяки чітко організованій структурі app команда може швидше орієнтуватися в проєкті та уникати дублювання коду. Централізоване управління станом підвищує надійність і передбачуваність роботи застосунку. Такий підхід спрощує командну розробку та забезпечує легкість інтеграції нових функцій у майбутньому.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Застосунок складається з наступних функціональних сторінок, розбитих
на блоки:

- головна сторінка (рисунок 4.1);
- сторінка для працівників;
- сторінка для адміністратора.

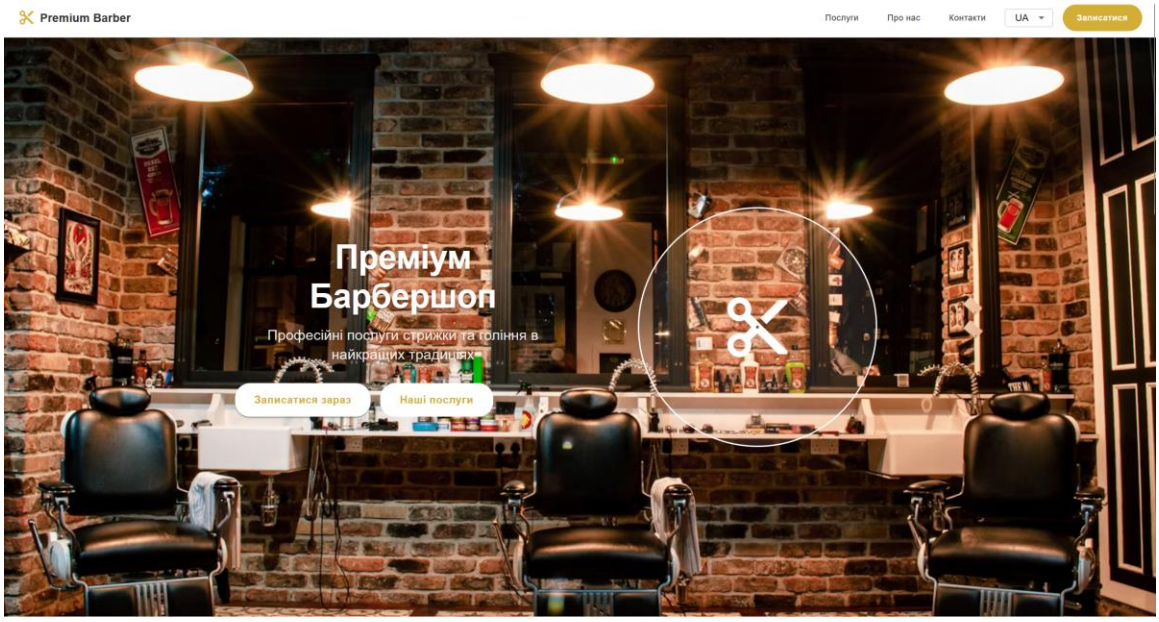

Рисунок 4.1 – Головна сторінка

4.1 Навігація головною сторінкою

З головної сторінки застосунку барбершопу користувачі одразу потрапляють на сучасну цільову сторінку з красивим дизайном і зрозумілою навігацією. У верхній частині сторінки знаходиться заголовок, що містить логотип «Premium Barber», меню навігації та кнопку «Забронювати зараз» для швидкого бронювання послуг.

Основна частина сторінки починається з секції яка містить заголовок «Преміум Барбершоп» і підзаголовок «Професійні послуги стрижки та

гоління в найкращих традиціях». Тут користувачу надаються два основні варіанти: «Забронювати зараз» для прямого бронювання послуги та «Наші послуги» для ознайомлення з тим, що пропонує барбершоп.

Сторінка виглядає наступним чином: «Наші послуги» (Рисунок 4.2) з описом і цінами на кожну послугу, «Про наш барбершоп» (Рисунок 4.3), де є інформація про заклад і те, що його відрізняє, та «Відвідайте нас сьогодні» з адресою, номером телефону та годинами роботи.

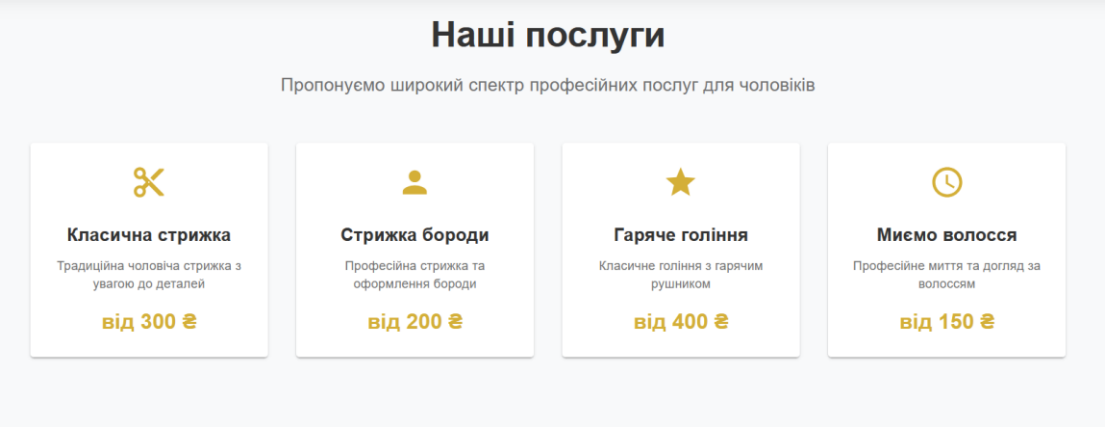

Рисунок 4.2 – Секція «Наші послуги»

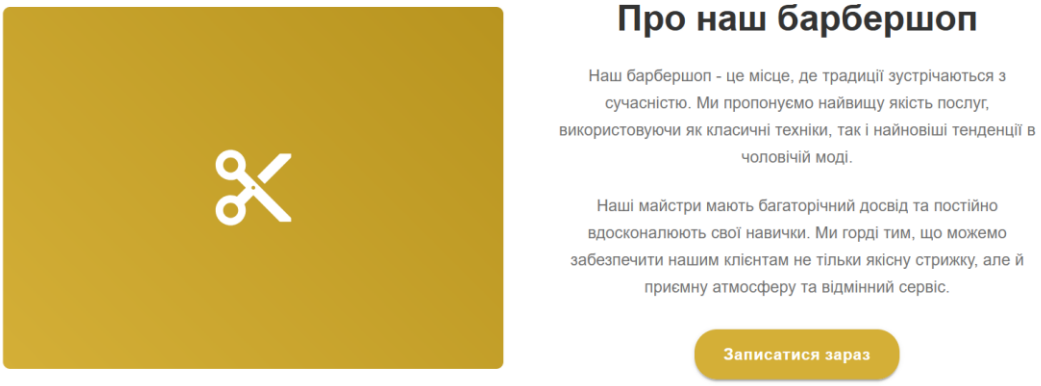

Рисунок 4.3 – Секція «Про наш барбершоп»

Однією з особливостей, що відрізняє цей застосунок, є те, що вам не потрібно входити в систему, щоб переглянути інформацію, як це робиться в багатьох інших сервісах. Користувач може переглядати всі пропозиції та

послуги барбершопу без входу в систему. Коли користувач натискає кнопку «Забронювати зараз», з'являється бічна панель, яка дозволяє вибрати барбера, філію, процедури та час візиту, а також заповнити контактні дані, необхідні для бронювання.

Завдяки цій концепції застосунок пропонує миттєвий і зручний доступ до всіх основних функцій без надмірностей, що в кінцевому підсумку підвищує конверсію відвідувачів у клієнтів і покращує загальний користувацький досвід.

4.2 Запис на послуги

Щоб записатися на прийом до майстра, користувачу достатньо натиснути кнопку «Записатися зараз» на веб-сайті. Після цього на правій стороні екрана відкриється бокове меню, де користувач можете вибрати майстра та послуги.

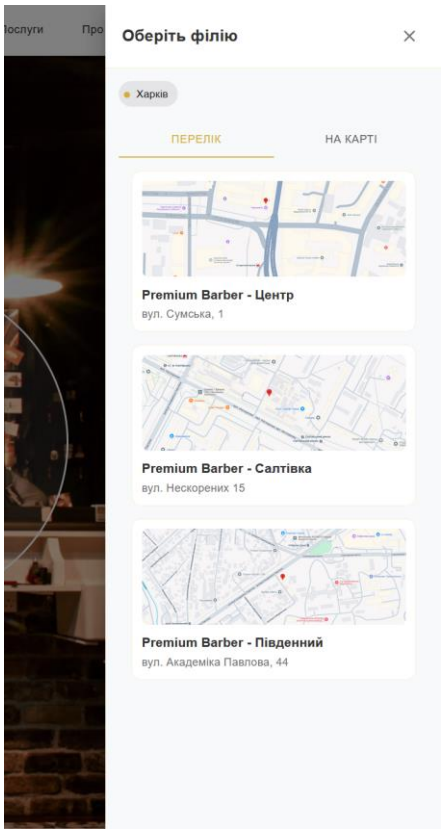

Рисунок 4.4 – Бокове меню для запису

Першим кроком буде вибір філії барбершопу (рисунок 4.4) – користувач бачить перед собою перелік усіх філій, з їх адресами. Користувачу достатньо обрати будь-яку зручну для нього філію, і натиснути по ній щоб перейти у наступне меню.

Після вибору філії користувачу буде запропоновано вибрати майстра. Система показує доступних майстрів, відображаючи їхні фотографії, імена та відгуки (рисунок 4.5). також тут можна почитати відгуки інших клієнтів на конкретного майстра (рисунок 4.6), та залишити свій відгук (рисунок 4.7).

Якщо це користувача це не влаштовує, але для нього немає різниці, хто саме його обслуговує – користувач має змогу вибрати варіант «Будь-який доступний співробітник»

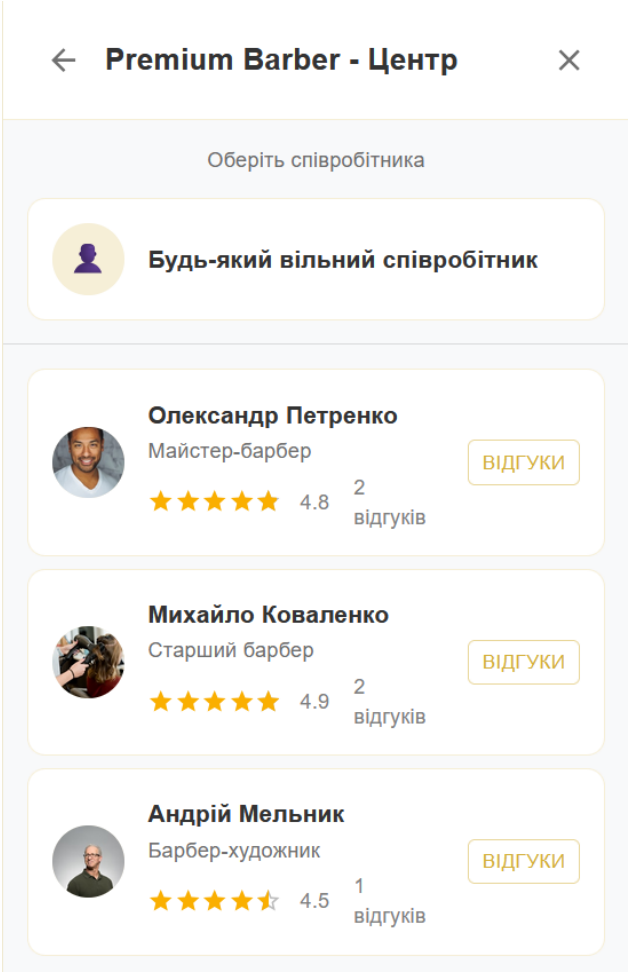

Рисунок 4.5 – Бокове меню для вибору майстра

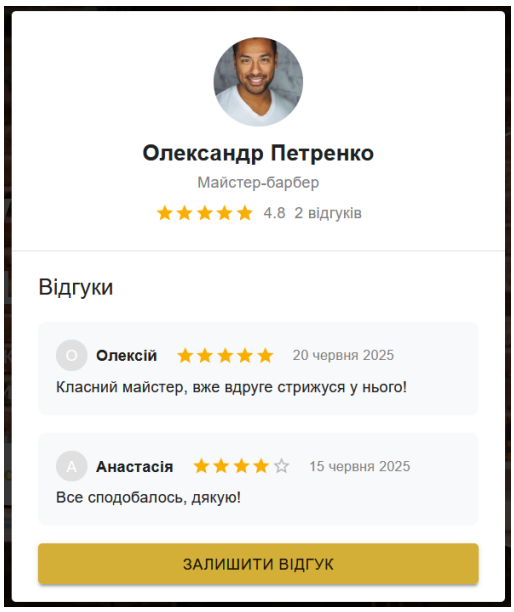

Рисунок 4.6 – Вікно з відгуками

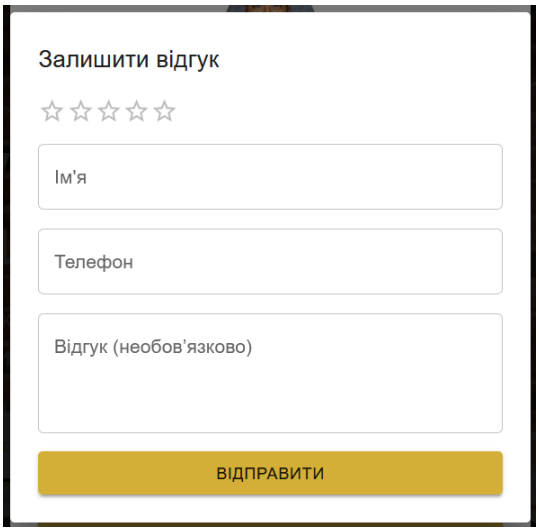

Рисунок 4.7 – Вікно для надання відгуку

Після вибору майстра користувачу потрібно буде вибрати час для візиту. Усі доступні часи для запису у майстра будуть відображені як часові слоти на поточний день, і на наступний з шагом у 30 хвилин (рисунок 4.8). Користувачу достатньо натиснути на буд-який слот зі зручною для нього датою та часом.

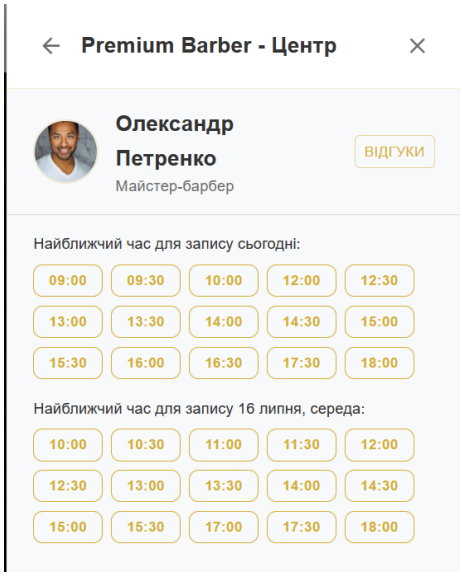

Рисунок 4.8 – Меню з вибором часу запису

Після вибору часу користувачу буде запропоновано вибрати послуги. Про кожну послугу надається інформація про її тривалість, вартість, та короткий опис самої послуги (рисунок 4.9). У користувача є можливість обрати одну або одразу декілька послуг. Загальний час і вартість вибраних послуг з’являться внизу екрана (рисунок 4.10), після чого користувачу треба буде натиснути «Далі» щоб продовжити.

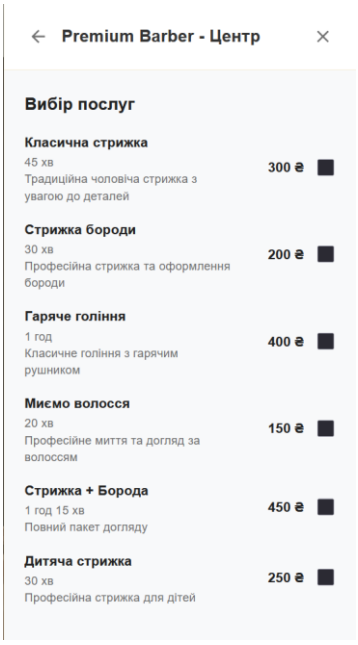

Рисунок 4.9 – Меню з вибором часу запису

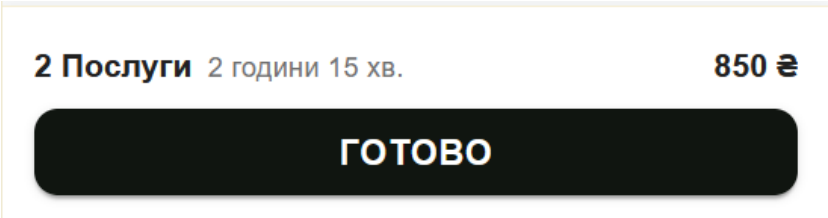

Рисунок 4.10 – Загальний час та ціна обраних послуг

У наступному меню користувачу надається вікно з чотирма полями – ім’я, email, номер телефону та поле для додаткової інформації (рисунок 4.11). В цих полях користувачу треба зазначити персональні дані корректно, щоб у користувача не було проблем з записом. Слід зазначити, що поля, позначені зірочкою (*), є обов'язковими, а також те, що поле «Email» та «Телефон» підлягають автоматичній валідації, щоб мати точні дані про користувача. Після заповнення необхідних полів користувачу треба натиснути кнопку «Записатися».

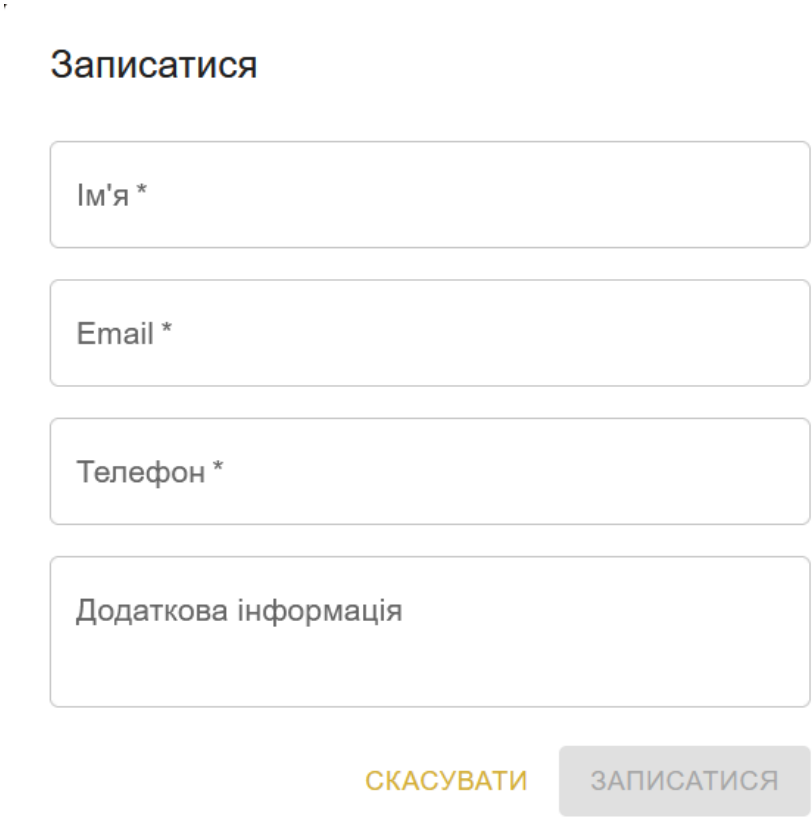

Рисунок 4.11 – Вікно запису

Також користувач може в будь який момент натиснути кнопку «Назад» у верхній частині меню, щоб повернутися до попереднього кроку. А щоб скасувати меню запису та не створювати запис, користувачу достатньо натиснути кнопку «Х» у верхньому правому куті. Якщо користувач захоче змінити вже вибрану філію, майстра або час, йому треба повернутися за допомогою кнопки «Назад», доки він не дійде до потрібного кроку.

4.3 Меню майстрів

Сторінка управління персоналом використовується для перегляду та управління записами клієнтів до конкретних майстрів. Ця сторінка доступна лише для авторизованих користувачів з відповідним рівнем доступу.

На верхній частині сторінки є панель вибору співробітника (рисунок 4.12). Користувачу треба натиснути на фото або картку співробітника, для того щоб переглянути записи відповідного майстра. Вибраний співробітник буде візуально виділений.

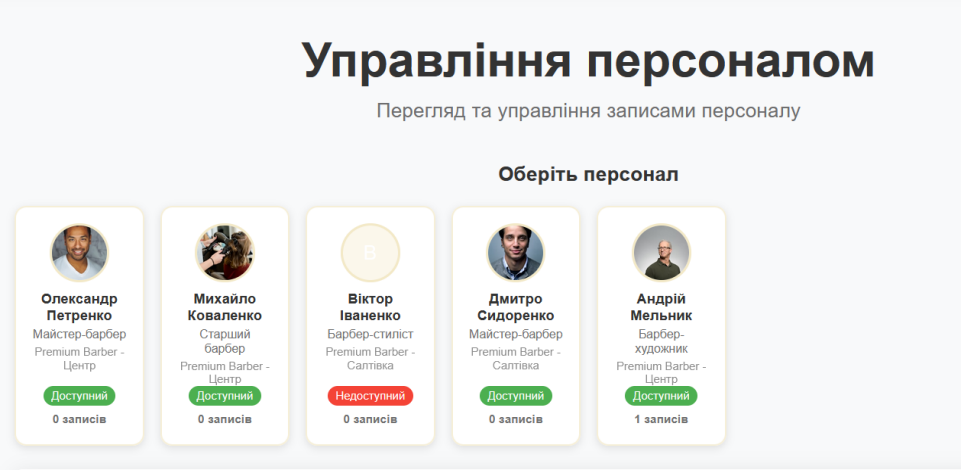

Рисунок 4.12 – Панель вибору співробітника

Після вибору співробітника користувач може побачити огляд усіх записів у цього майстра. Якщо записів немає, з'явиться відповідне повідомлення «У вибраного співробітника наразі немає записів.»

Інформація про запис містить в собі таку інформацію, як: ім'я клієнта, статус запису (підтверджено, очікує, скасовано), дата і час візиту, список обраних послуг, контактний номер телефону клієнта, загальна вартість і дні обслуговування та за наявності коментарі клієнта.

Також тут відображається поточний стан запису (прийнятий, скасований чи очікується). Якщо запис ще не був оброблений користувачем, то знизу буде дві кнопки для того щоб підтвердити запис або скасувати (рисунок 4.13).

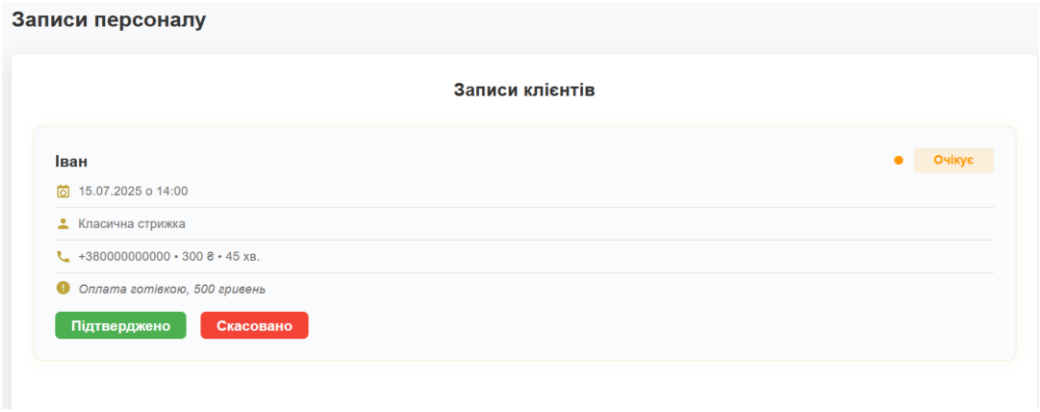

Рисунок 4.13 – Вигляд неприйнятого запису клієнта

Якщо клієнт записався до «Будь-якого доступного співробітника», замість конкретного майстра, то до таких записів надається кнопка "Взяти роботу" (рисунок 4.14), після натискання цієї кнопки запис буде додано до відповідного співробітника і запис змінить статус на «Підтверджено».

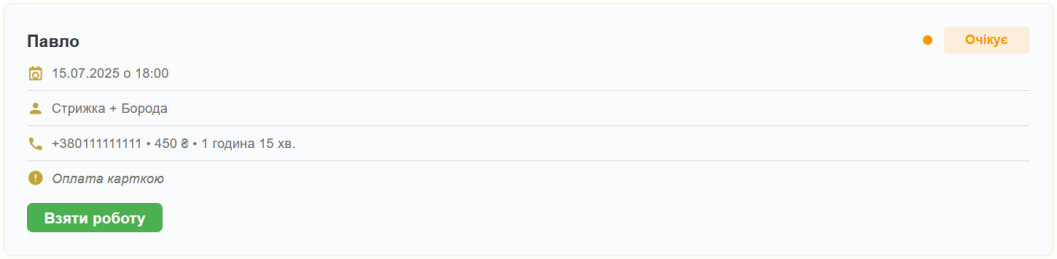

Рисунок 4.14 – Вигляд неприйнятого запису клієнта, зробленим через опцію «Будь-який доступний співробітник»

Слід зауважити, що якщо користувач хочете повернутися до основного списку співробітників, ви також маєте можливість натиснути на одного з інших співробітників з верхньої панелі. Також статуси записів оновлюються і зберігаються в режимі реального часу.

4.4 Панель адміністратора

Адміністративна панель призначена для управління основними елементами барбершопу: філіями, персоналом та записами клієнтів. Доступ до цієї сторінки мають лише авторизовані користувачі з відповідними правами.

Для доступу до адміністративної панелі користувачу необхідно бути авторизованим у системі. Якщо користувач не авторизовані, система покаже вікно входу в адмін панель (рисунок 4.15).

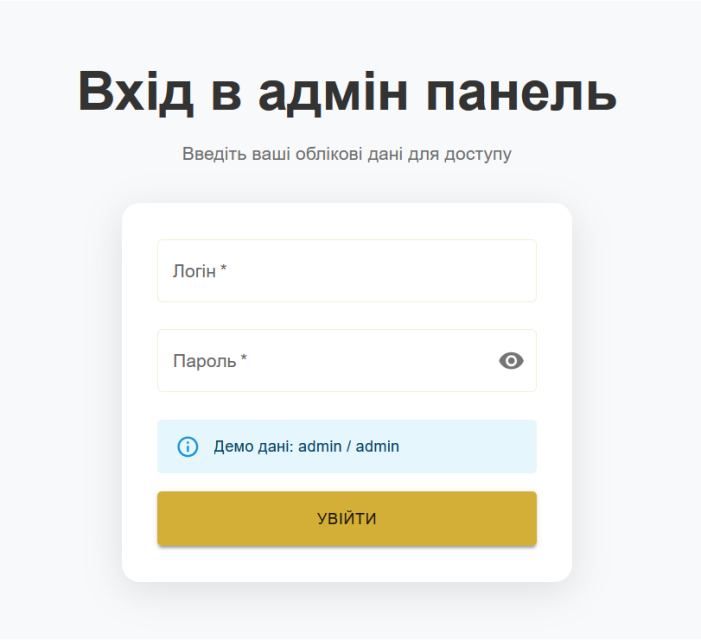

Рисунок 4.15 – Вікно входу в адмін панель

Адміністративна панель має три основні розділи, доступ до яких здійснюється через вкладки у верхній частині сторінки: філії, персонал та записи.

У вкладці «Філії» (рисунок 4.16) користувач має можливість переглядати список всіх філій барбершопу, додавати нові філії, натиснувши кнопку «Додати філію», редагувати існуючі філії, змінюючи їх назву, адресу, контактну інформацію, години роботи тощо або видаляти філії (операція проводиться з підтвердженням дії користувача)

При додаванні або редагуванні філії потрібно заповнити наступні поля (рисунок 4.17): назва філії, адреса, телефон, email, координати (широта і довгота) для відображення на карті, години роботи, опис та URL зображення.

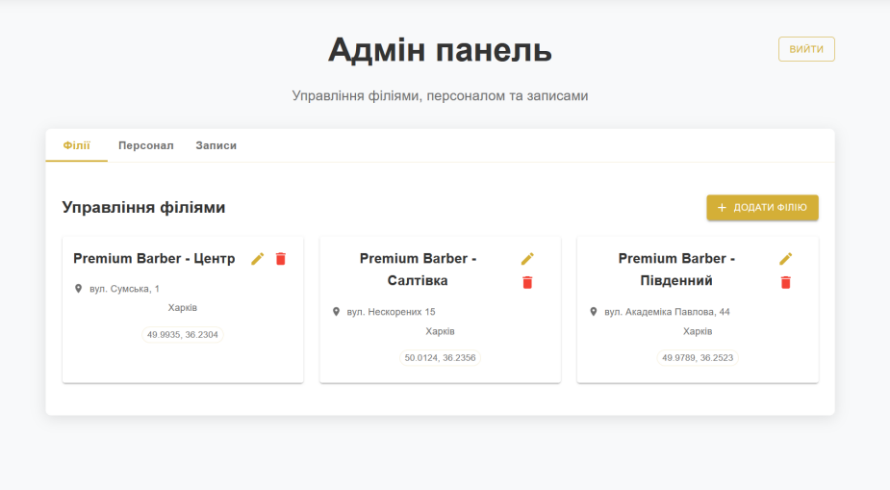

Рисунок 4.16 – Вигляд меню управління філіями

Рисунок 4.17 – Вікно додавання філії

У вкладці «Персонал» користувач має можливість переглядати список всіх співробітників, додавати нових співробітників, натиснувши кнопку «Додати співробітника», редагувати інформацію про існуючих співробітників та видаляти співробітників (операція проводиться з підтвердженням дії користувача) (рисунок 4.18).

При додаванні або редагуванні співробітника потрібно заповнити такі поля, як: ім'я співробітника, спеціалізація, досвід роботи, рейтинг, філія, до якої прикріплений співробітник, опис, URL зображення та статус доступності для записів (рисунок 4.19).

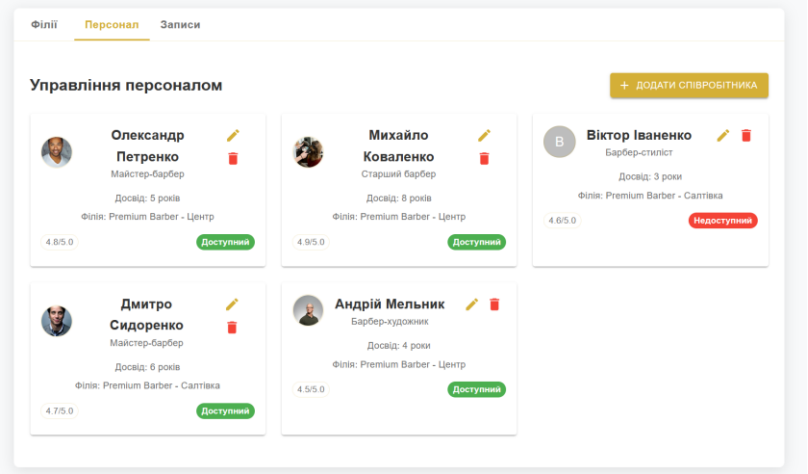

Рисунок 4.18 – Вигляд меню управління персоналом

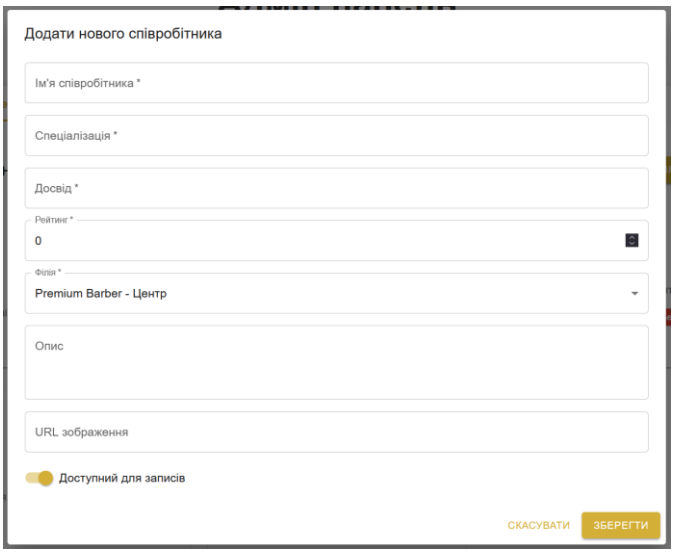

Рисунок 4.19 – Вікно додавання нового співробітника

У вкладці «Записи» (рисунок 4.20) користувач має можливість переглядати всі записи клієнтів, фільтрувати записи за барбером та статусом, редагувати існуючі записи, змінювати статус записів (підтверджено, очікує, скасовано), видаляти окремі записи, очищати всі записи (з підтвердженням дії). Кожен запис містить наступну інформацію: ім'я клієнта, email та телефон клієнта, дата та час візиту, обрані послуги. тривалість та вартість послуги, статус запису та за наявності додаткову інформацію від клієнта.

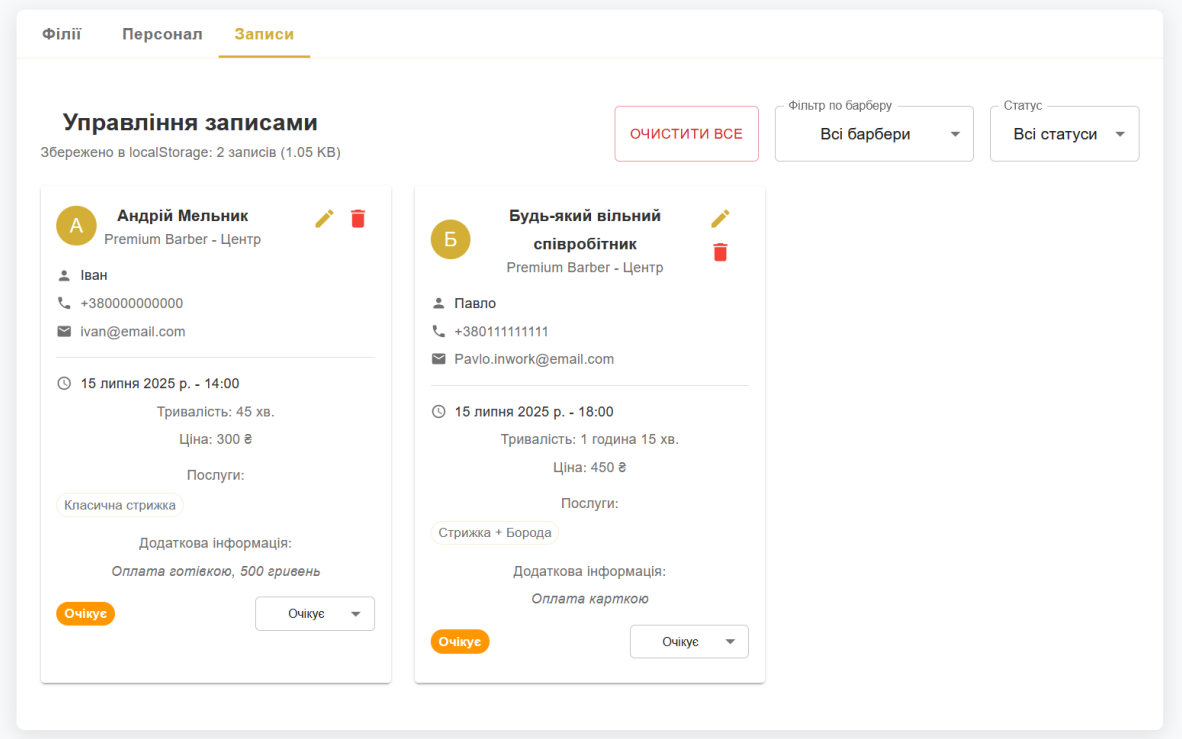

Рисунок 4.20 – Вигляд меню управління персоналом

Всі зміни в адміністративній панелі зберігаються автоматично. Для повернення до попередньої вкладки достатньо натиснути на відповідну вкладку у верхній частині сторінки. Система відображає повідомлення про успішне виконання операцій або про помилки, що виникли під час роботи. Деякі операції (наприклад, видалення філії або співробітника) вимагають підтвердження для запобігання випадковим діям.

ВИСНОВКИ

В рамках кваліфікаційної роботи було створено веб-застосунок для автоматизації процесів обслуговування клієнтів у мережі барбершопів, зокрема в плануванні та управлінні доступністю майстрів, а також у комунікації між адміністрацією, майстрами та відвідувачами. Мета полягає в тому, щоб надати рішення, яке значно спростить операції для співробітників і підвищить зручність використання для клієнтів.

Під час аналізу існуючих сервісів було відзначено, що готові платформи не завжди забезпечують гнучку адаптацію логіки робочого процесу до особливостей конкретного закладу, звужують обсяг функціональності або не надають доступу до внутрішніх налаштувань.

Був розроблений веб-застосунок, де основні точки взаємодії представлені кінцевим користувачам. Клієнти можуть самостійно ознайомитися зі списком послуг, познайомитися з майстрами, вибрати найбільш зручний час для візиту та записатися онлайн. Майстри мають свій власний розклад, можуть керувати своїм робочим часом і бачити записи клієнтів у міру їх надходження. Адміністративний розділ контролює всі розклади майстрів.

Для серверної частини проєкту використовувалася мова програмування C# у поєднанні з ASP.NET Core та реляційною СУБД MySQL у парі з Entity Framework Core. Клієнтська частина написана з використанням React та TypeScript, що дозволило створити простий і швидкий інтерфейс, який підтримує сучасні інтерактивні елементи.

Усі функціональні задачі, поставлені на початковому етапі розробки БУЛИ успішно реалізованІ. Подальший розвиток системи передбачає розширення особистого кабінету клієнта, інтеграцію платіжної системи для попередньої оплати послуг, реалізацію функції нагадувань про запис, а також впровадження багатомовності для роботи з клієнтами з різних регіонів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Booksy – платформа для бронювання зустрічей. URL: https://booksy.com (дата звернення: 12.06.2025).

2. Boulevard – платформа для клієнтського сервісу. URL: https://www.joinblvd.com (дата звернення: 12.06.2025).

3. Vagaro – платформа для управління бізнесом у сфері краси. URL: https://www.vagaro.com/ (дата звернення: 12.06.2025).

4. Мартін Р. С. Чистий код: створення, аналіз та рефакторинг. Книга для програмістів. Прентіс Голл, 2008. 464 с.

5. W3C: Web Application Architecture. URL: https://www.w3.org/TR/webarch/ (дата звернення: 27.06.2025).

6. React – офіційний сайт. URL: https://reactjs.org (дата звернення: 15.06.2025).

7. ASP.NET Core – офіційна документація. URL: https://learn.microsoft.com/en-us/aspnet/core (дата звернення: 17.06.2025).

8. Visual Studio Code – офіційний сайт. URL: https://code.visualstudio.com (дата звернення: 19.06.2025).

9. SOLID-принципи в .NET. URL: https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles (дата звернення: 17.06.2025).

10. TypeScript – офіційний сайт. URL: https://www.typescriptlang.org (дата звернення: 15.06.2025).

11. MySQL – офіційний сайт. URL: https://www.mysql.com (дата звернення: 17.06.2025).

12. Entity Framework Core – офіційний сайт. URL: https://learn.microsoft.com/en-us/ef/core (дата звернення: 18.06.2025).