Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем управління
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи для пошуку та бронювання послуг»

Виконав: магістр 2 курсу, групи 2АКІТР-23м
спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва напряму підготовки, спеціальності)

Богдан ПОЛЬГУЛЬ
(ім'я ПРІЗВИЩЕ)

Керівник: к.т.н., професор каф. КСУ
Микола БИКОВ
(ім'я ПРІЗВИЩЕ)
«4» 12 2024 р.

Опонент:
Сергій КРИВОГУБЧЕНКО
(ім'я ПРІЗВИЩЕ)
«4» 12 2024 р.

Допущено до захисту
Завідувач кафедри КСУ
Вячеслав КОВТУН
(прізвище та ініціали)
«9» грудня 2024 року

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Зав. кафедри КСУ
В'ячеслав КОВТУН
"14" жовтня 2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Польгулю Богдану Васильовичу.
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи для пошуку та бронювання послуг
керівник роботи Биков Микола Максимович
затверджені наказом ВНТУ від "17" вересня 2024 року №310
2. Термін подання студентом роботи "1" грудня 2024 року
3. Вихідні дані до роботи: підтримка Google API; розширений пошук для бронювання послуг; економія часу для користувача; середовище моделювання – Visual Studio Code; перелік матеріалів для розробки:

1. "Optimizing Sorting Algorithms for Large Data Sets," [Електронний ресурс] – Режим доступу: https://blog.algorithmexamples.com/sorting-algorithm/top-15-sorting-algorithms-for-large-datasets/
2. "Sorting and Searching Algorithms Performance," [Електронний ресурс] – https://www.doabledanny.com/searching-and-sorting-algorithms-in-javascript
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. "Database System Concepts," *McGraw-Hill*, 2010

4. Зміст текстової частини: Вступ; Обґрунтування доцільності створення автоматизованої системи для пошуку та бронювання послуг; Проектування автоматизованої системи для пошуку та бронювання послуг; Розробка програмного забезпечення; Економічний розділ.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) Діаграма бізнес-процесу запису клієнта; Діаграма бізнес-процесу замовлення витратних матеріалів; Схема алгоритму реалізації автоматизованої системи для пошуку та бронювання послуг; UML-діаграма послідовності; Плакати демонстраційні.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Кавецький В.В. – професор кафедри економіки підприємства і виробничого	13.10.24	23.11.24

2. Дата видачі завдання "14" жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Обґрунтування доцільності створення автоматизованої системи для пошуку та бронювання послуг. Постановка задач	03.09.24	07.09.24	
2	Розробка технічного завдання	08.09.24	10.09.24	
3	Розробка архітектури системи	11.09.24	19.09.24	
4	Вибір технологій розробки системи	20.09.24	01.11.24	
5	Розробка програмного забезпечення системи	02.11.24	14.11.24	
6	Тестування програмного забезпечення	15.11.24	18.11.24	
7	Оформлення пояснювальної записки	19.10.24	23.11.24	
8	Апробація і публікації результатів	24.11.24	30.11.24	
9	Графічні матеріали Плакати демонстраційні Діаграма бізнес-процесу запису клієнта Діаграма бізнес-процесу замовлення витратних матеріалів Схема алгоритму реалізації автоматизованої системи для пошуку та бронювання послуг. UML-діаграма використання програмного забезпечення.	 21.11.24 22.11.24 24.11.24 26.11.24 28.11.24 29.11.24	 21.11.24 23.11.24 25.11.24 27.11.24 29.11.24 30.11.24	

Студент _____ Богдан ПОЛЬГУЛЬ
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ Микола БИКОВ
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 621.374.415

Польгуль Б. В. Автоматизована система для пошуку та бронювання послуг. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп'ютерно- інтегровані технології та робототехніка, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2024. 155 с.

На укр. мові. Бібліогр.: 72 назв; рис.: 23; табл. 11.

Метою магістерської кваліфікаційної роботи є розробка автоматизованої системи для пошуку та бронювання послуг, що забезпечить покращення процесу бронювання завдяки використанню сучасних алгоритмів обробки даних. У роботі проведено огляд основних проблем автоматизації пошуку та бронювання послуг, а також аналіз існуючих рішень. На основі цього обґрунтовано необхідність створення нової системи. У теоретично-методичній частині розроблено моделі бізнес-процесів та алгоритми для оптимізації цих процесів, а також проведено їх оцінку за ефективністю та впливом на продуктивність системи. У практичній частині описано вибір інструментів і технологій для розробки програмного забезпечення, архітектуру системи, а також розроблено й протестовано користувацький інтерфейс. Описано результати тестування системи, зокрема покращення швидкості обробки запитів та зручності для користувачів.

Економічна частина роботи містить аналіз витрат на впровадження системи та розрахунок її собівартості.

Ілюстративна частина містить 4 плакатів і креслень з ілюстрацією результатів роботи.

Ключові слова: автоматизована система, бронювання послуг, алгоритм, програма.

ABSTRACT

UDC 621.374.415

Polhul B.V. Automated system for searching and booking services. Master's qualification thesis on specialty 174 - Automation, computer-integrated technologies and robotics, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2024. 155 p.

In Ukrainian language. Bibliographer: 72 titles; fig.: 23; tabl. 11.

The goal of the master's thesis is to develop an automated system for searching and booking services, which will improve the booking process through the use of modern data processing algorithms. The work includes an overview of the main problems related to the automation of service search and booking, as well as an analysis of existing solutions. Based on this, the need for creating a new system is justified. In the theoretical-methodological part, business process models and algorithms for optimizing these processes are developed, and their effectiveness and impact on system performance are evaluated. The practical part describes the selection of tools and technologies for software development, the system architecture, and the design and testing of the user interface. The results of system testing are presented, including improvements in query processing speed and user convenience.

The economic part of the work includes an analysis of the costs of implementing the system and a calculation of its cost-effectiveness.

The illustrative section consists 4 of materials that demonstrate the testing results and the efficiency of the developed system.

Keywords: automated system, reservation of services, algorithm, program.

ЗМІСТ

ВСТУП

Актуальність теми. На поточному етапі розвитку інформаційних технологій зростає попит на автоматизовані системи, які забезпечують належний пошук і резервування різних послуг. В умовах великого обсягу даних і широкого спектру пропозицій оптимізація обробки та аналізу інформації є важливим завданням, користувачі можуть приймати швидкі і точні рішення. Сучасна система забезпечує не тільки зручний інтерфейс, але і механізм ефективної обробки великих обсягів даних для надання точних рекомендацій і швидкого бронювання.

У зв'язку з цим виникає потреба в розробці автоматизованої платформи, яка використовує сучасні алгоритми обробки інформації та технології штучного інтелекту для пошуку та бронювання різних послуг. Ця система повинна мати можливість аналізувати дані про доступність, вартість і якість послуг, надаючи індивідуальний підхід до кожного користувача.

Таким чином, розробка ефективної автоматизованої платформи для пошуку та бронювання послуг є актуальним завданням, що сприятиме поліпшенню взаємодії користувачів з різними пропозиціями, підвищить швидкість обробки запитів і рівень задоволеності клієнтів.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконана згідно плану науково-дослідної роботи кафедри КСУ.

Мета і задачі дослідження. Метою магістерської кваліфікаційної роботи є підвищення ефективності процесу пошуку та бронювання послуг за рахунок розробки системи для автоматизації даного процесу.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- Розробити структуру автоматизованої системи для пошуку та бронювання послуг;
- Впровадити алгоритми аналізу та обробки даних про послуги;
- Оптимізувати процес бронювання для зменшення часу обробки запитів;

- Дослідити механізми рекомендацій на основі вподобань користувачів;

- Розробити користувацький інтерфейс для зручної взаємодії з системою;

- Провести тестування та аналіз ефективності системи.

Об’єктом дослідження є автоматизовані системи пошуку та бронювання послуг, їх структура, алгоритми обробки даних та взаємодія з користувачами.

Предметом дослідження є методи та моделі автоматизації процесів пошуку та бронювання, алгоритми аналізу даних та рекомендацій.

Методи дослідження. У процесі дослідження застосовувалися: теорія інформації, методи машинного навчання, теорія множин, алгоритми пошуку та рекомендацій, комп’ютерне моделювання для аналізу та перевірки теоретичних положень.

Наукова новизна одержаних результатів.

- Удосконалено методи пошуку та обробки даних для автоматизованих систем бронювання послуг;

- Запропоновано алгоритми рекомендацій, що дозволяють персоналізувати пропозиції для користувачів на основі їхніх попередніх виборів та вподобань;

- Розроблено та протестовано нові підходи до підвищення ефективності процесу бронювання за рахунок оптимізації алгоритмів обробки запитів.

Практична цінність. Результати дослідження дозволили розробити систему, яка забезпечує ефективний пошук та бронювання послуг, покращує швидкість обробки даних та підвищує рівень задоволеності користувачів.

Достовірність теоретичних положень підтверджується застосуванням сучасних методів обробки даних та аналізу результатів тестування системи на реальних даних.

Особистий внесок здобувача. Більшість результатів отримано автором самостійно.

Апробація та публікації результатів роботи. Основні положення й результати досліджень були представлені та обговорені на науково-технічній конференції ФІІТА (2024). За результатами досліджень було опубліковано тези доповіді в збірнику матеріалів цієї конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДЛЯ ПОШУКУ ТА БРОНЮВАННЯ ПОСЛУГ

1.1 Аналіз проблем в галузі автоматизованої системи пошуку та бронювання послуг

Розвиток нових технологій та широке впровадження Інтернету призвели до значних змін у способах взаємодії компаній з клієнтами. Мережа Інтернет (www) надала кожному доступ до величезної кількості інформації незалежно від місця проживання. Проте, для ефективної організації цього потоку інформації необхідна правильна структура, особливо у контексті надання послуг [2].

Автоматизовані системи пошуку та бронювання послуг стали важливим елементом бізнесу в багатьох областях, таких як готельний бізнес, медичні установи, особливо салони краси та салони краси Дека. Основна мета такої системи-спростити процес взаємодії з клієнтами на підприємстві та підвищити ефективність управління бізнес-процесами [3]. Однак на практиці при розробці та впровадженні автоматизованої системи існують різні проблеми, які необхідно ретельно проаналізувати перед початком проекту.

Однією з головних проблем, яку повинна вирішити автоматизована система, є правильне зберігання та обробка інформації про клієнтів та їх замовлення. Без належного програмного забезпечення існує ризик втрати даних через людську помилку або технічний збій. Наприклад, якщо косметолог вирішить взяти відпустку, а менеджер неправильно зареєструє клієнта в день відсутності, це може призвести до втрати або дискомфорту клієнта, що негативно позначиться на репутації закладу [4].

Автоматизовані системи суттєво знижують ризики, пов’язані з організацією та управлінням розкладом співробітників, а також відіграють важливу роль у покращенні якості обслуговування клієнтів. Завдяки

впровадженню таких систем процес керування клієнтськими обліковими записами стає набагато ефективнішим і прозорішим. Зокрема, автоматичне оновлення інформації про графік роботи фахівців дозволяє уникнути розбіжностей і помилок у розкладі. Ця функція особливо актуальна для спеціалістів, таких як перукарі та стилісти, для яких швидка адаптація до змін у розкладі є важливим чинником стабільної роботи.

Автоматизовані системи надають клієнтам можливість переглядати актуальну інформацію про доступні часи роботи майстрів у реальному часі. Це допомагає уникати перебронювання та збоїв у розкладі, які можуть негативно вплинути на репутацію закладу і призвести до втрати клієнтів.

Однією з ключових переваг таких систем є можливість інтеграції кількох процесів на одній платформі. Наприклад, система автоматично оновлює реєстраційні дані клієнтів, забезпечуючи своєчасну синхронізацію між різними працівниками та адміністраторами. Це значно скорочує потребу у ручному введенні даних і мінімізує ризик людських помилок.

Аналіз бізнес-процесів клієнта, відображений на діаграмі (Рис. 1.1), дозволяє детально розглянути всі етапи обслуговування. На початковому етапі клієнт обирає потрібну послугу зі списку доступних варіантів, ознайомлюється із вільними часовими слотами та вибирає майстра, який надає обрану послугу. Після цього система перевіряє доступність обраного часу. Якщо вибраний слот вільний, клієнт підтверджує бронювання, що активує подальші дії системи.

Система автоматично формує підтвердження та надсилає його клієнту через електронну пошту або інший зручний канал зв’язку, забезпечуючи точність і оперативність сповіщень. Завдяки цьому між клієнтом і закладом встановлюється надійний інформаційний зв’язок, який мінімізує ризик непорозумінь. Такий підхід сприяє підвищенню зручності користування послугами, адже клієнт має змогу в будь-який момент отримати доступ до інформації про своє бронювання.

Більше того, система враховує всі аспекти взаємодії, що дозволяє створити безперервний цикл обслуговування. Клієнт отримує впевненість у

тому, що його потреби враховані, а заклад — інструменти для оптимізації роботи персоналу. Це підвищує загальну ефективність процесів, сприяючи покращенню клієнтського досвіду та формуванню довіри до закладу.

Автоматизація реєстраційних процесів має широкий спектр переваг, які позитивно впливають на всі аспекти діяльності закладу. Вона спрощує процес комунікації з клієнтами, дозволяючи їм легко отримувати інформацію про доступні послуги та спеціальні пропозиції. Завдяки інтеграції з CRM-системами автоматизація допомагає зберігати історію записів клієнтів, що сприяє створенню персоналізованих пропозицій. Такі системи також забезпечують автоматичне надсилання нагадувань про заплановані візити, що знижує ризик пропусків і сприяє більш ефективному використанню часу. Крім того, автоматизація дає можливість інтеграції платіжних систем, що спрощує процес розрахунків і покращує клієнтський досвід. Для співробітників це зручний інструмент, який зменшує адміністративне навантаження, дозволяючи більше часу приділяти безпосередньо роботі з клієнтами. Аналіз даних, зібраних за допомогою таких систем, допомагає керівництву виявляти ключові тренди та оперативно реагувати на зміни в попиті. Це забезпечує конкурентну перевагу на ринку, адже заклади можуть швидше адаптуватися до нових викликів. У сучасному світі автоматизація стає не просто перевагою, а необхідністю для розвитку бізнесу. Впровадження таких систем сприяє підвищенню загальної продуктивності організації, допомагаючи їй досягати довгострокових цілей. У підсумку, автоматизація стає основою для створення інноваційного середовища, орієнтованого на якість і результат.

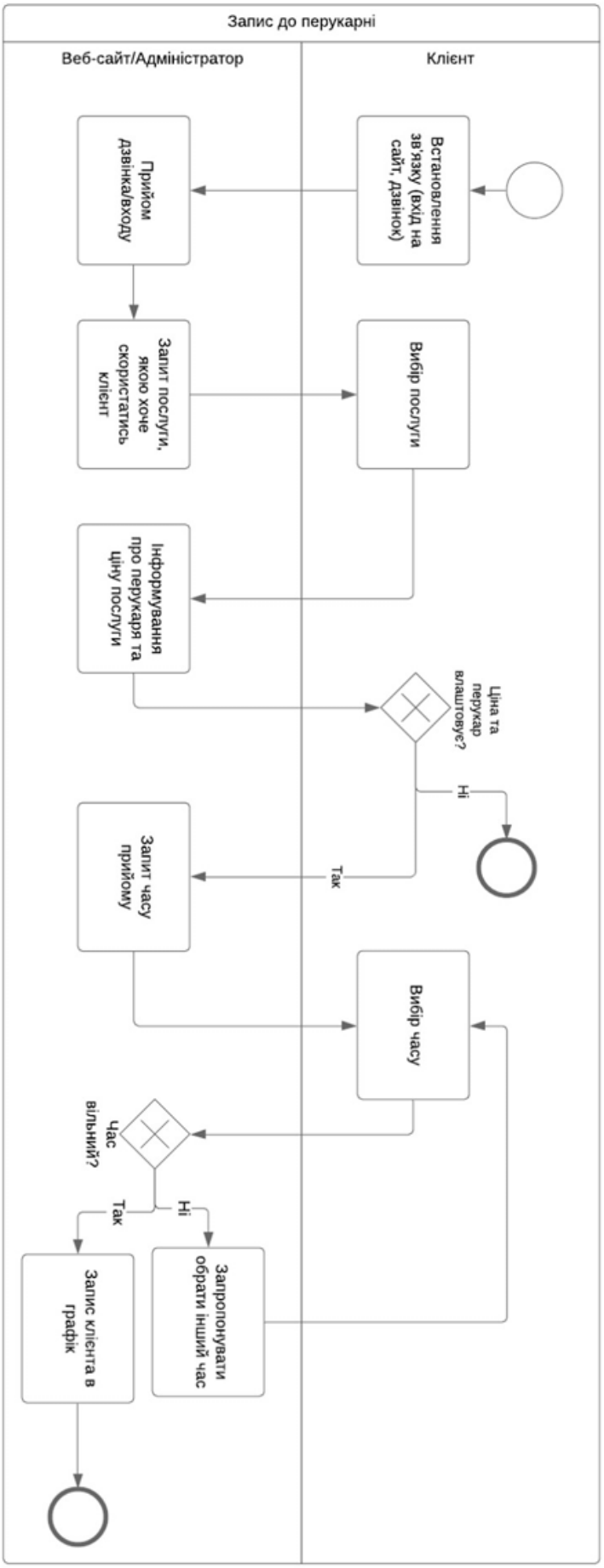


Рисунок 1.1 – Діаграма бізнес-процесу запису клієнта

Автоматизація цих процесів значно знижує навантаження на працівників і в той же час підвищує якість обслуговування, оскільки дозволяє системі швидко та точно взаємодіяти з клієнтами. Окрім того, одним із ключових аспектів ефективної діяльності бізнесу, що надає послуги, є правильне управління ресурсами. Для перукарень це включає не лише забезпечення робочих місць необхідними засобами (шампуні, одноразові рукавички, хімічні препарати для волосся тощо), а й організацію ефективного планування графіків майстрів та управління бронюваннями клієнтів.

Автоматизація управлінських процесів, таких як замовлення витратних матеріалів, є важливим кроком для підвищення продуктивності та оптимізації роботи підприємства. Використання автоматизованих систем значно полегшує організацію постачання та управління ресурсами. Наприклад, така система здатна автоматично контролювати наявність витратних матеріалів у перукарнях чи салонах краси і оперативно надсилати запити до постачальників, коли запаси досягатимуть критично низького рівня. Це дозволяє зменшити ймовірність виникнення ситуацій, коли брак необхідних матеріалів чи засобів може призвести до порушення безперервності надання послуг. Система, що здійснює автоматичний моніторинг наявності матеріалів, є надзвичайно корисною, оскільки вона працює в режимі реального часу, забезпечуючи постійне оновлення інформації про стан запасів. Завдяки такій автоматизації персонал може зосередитись на основних задачах, таких як обслуговування клієнтів, не витрачаючи часу на ручну перевірку ресурсів. Окрім того, автоматизація мінімізує ймовірність помилок, які можуть виникнути під час ручного контролю залишків матеріалів або замовлення. Використання автоматизованих систем дозволяє уникати надлишкових замовлень, що сприяє зниженню витрат на зберігання та забезпечує ефективну взаємодію з постачальниками.

Один з важливих аспектів автоматизації такого процесу – це можливість інтеграції системи замовлення витратних матеріалів із постачальниками. Це дає змогу здійснювати закупівлі з мінімальним втручанням з боку співробітників салону. Наприклад, система може не лише автоматично надсилати запити на

постачання матеріалів, а й вибирати найвигідніші пропозиції від постачальників на основі попередніх умов співпраці чи договорів. Це дозволяє заощаджувати кошти та забезпечувати стабільність постачань.

Як показано на Рис. 1.2, процес замовлення матеріалів складається з кількох етапів і включає в себе важливі кроки. Спочатку система відстежує наявність запасів і автоматично визначає рівень матеріалів для кожної позиції. Коли кількість конкретного матеріалу досягає мінімального рівня, генерується автоматичний запит на постачання, щоб уникнути дефіциту. Далі система надсилає сформований запит постачальнику та вибирає найкраще збережене замовлення з бази даних.

Постачальник підтверджує замовлення після отримання запиту і підтвердження необхідних матеріалів. На цьому етапі оновлюються дані про очікувану доставку, такі як дата доставки і кількість товару. Ця інформація інтегрована в загальний план роботи організації, що дає змогу керівництву планувати надання послуг із урахуванням термінів доставки матеріалів. Такий підхід забезпечує безперервність процесів, знижує ризик затримок та оптимізує управління ресурсами.

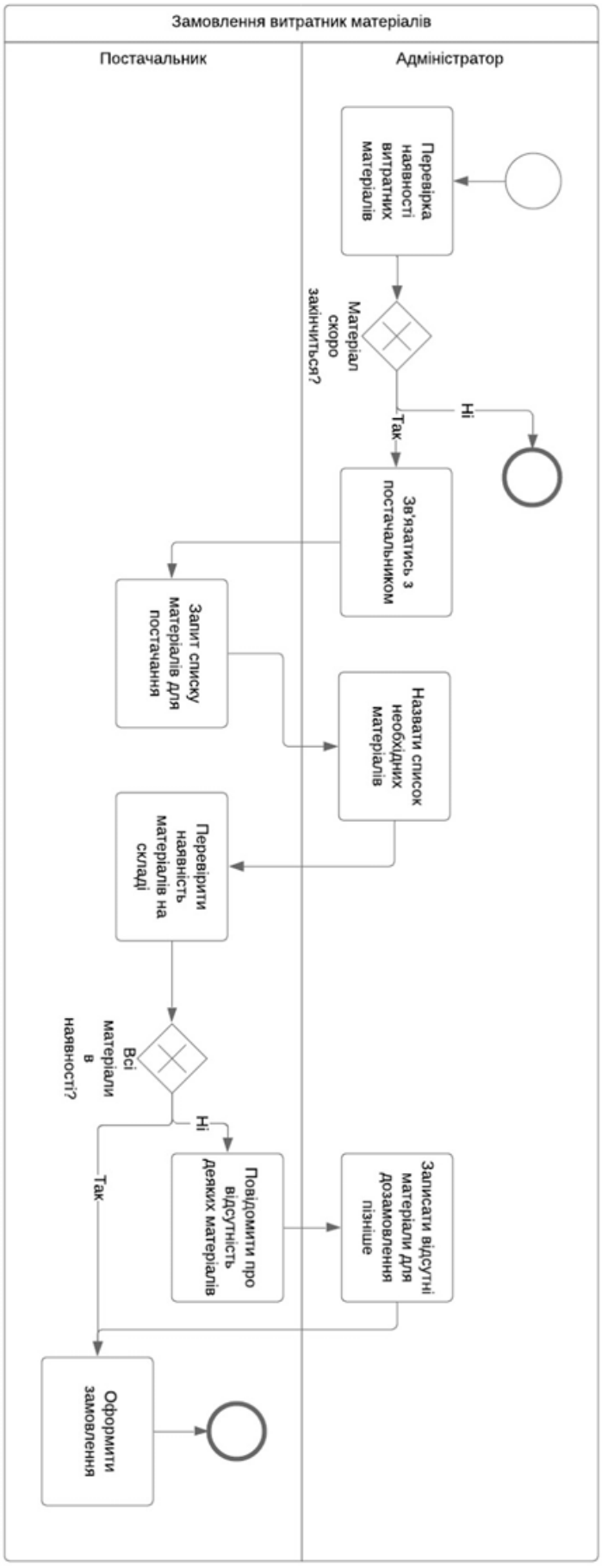


Рисунок 1.2 – Діаграма бізнес-процесу замовлення витратних матеріалів

Завдяки застосуванню автоматизованих систем для моніторингу запасів і їх інтеграції з постачальниками, сучасні компанії мають можливість ефективніше управляти своїми ресурсами, що сприяє значному зниженню витрат і поліпшенню рівня обслуговування клієнтів. Автоматичний контроль запасів допомагає підприємствам уникнути проблем, пов'язаних із надлишком або дефіцитом необхідних товарів. Завдяки інтеграції з постачальниками система може оперативно реагувати на зміни в попиті і забезпечувати безперервне постачання витратних матеріалів для перукарень та салонів краси. Крім того, можливість прогнозування потреб на основі історичних даних дозволяє уникати зайвих витрат на зберігання та закупівлю матеріалів, що оптимізує фінансові витрати компанії [6].

Зростання популярності перукарень і салонів краси призводить до збільшення їхньої кількості, що може бути складно управляти без централізованої системи. У таких випадках важливою є автоматизація всіх операційних процесів, що відбуваються в різних точках обслуговування. Без автоматизованої системи кожен заклад змушений постійно контактувати з керівником або власником для передачі даних щодо фінансів, робочого навантаження та інших важливих аспектів бізнесу. Автоматизовані управлінські системи дозволяють здійснювати моніторинг і контроль за всіма процесами в реальному часі, надаючи власникам бізнесу можливість отримувати актуальну інформацію та оперативно приймати рішення. Наприклад, керівники можуть відстежувати фінансові показники кожного закладу, управляти розподілом навантаження між працівниками і швидко реагувати на нестандартні ситуації, наприклад, на нестачу матеріалів або збільшення кількості клієнтів.

Централізоване управління значно спрощує доступ до важливої інформації та знижує ризик втрати даних або затримок в прийнятті рішень. Уявімо, що власнику великої мережі перукарень потрібно постійно зв'язуватися з адміністраторами кожного закладу для оновлення даних. Це створює додаткові труднощі, адже будь-яка втрата чи неточність інформації може

призвести до серйозних проблем, таких як недостатня кількість матеріалів чи погане планування графіків працівників. Автоматизована система дозволяє уникнути таких ризиків завдяки зберіганню всіх даних в єдиній базі та наданню доступу до них із будь-якого пристрою в зручному форматі для керівника [7].

Також важливою є інформаційна безпека, яка відіграє вирішальну роль при впровадженні автоматизованих систем. Сучасні системи повинні гарантувати захист конфіденційної інформації про клієнтів, їхні персональні дані та записи. Оскільки дедалі більше клієнтів записуються на послуги через Інтернет, питання безпеки набуває особливої важливості. Використання технологій шифрування, багатофакторної аутентифікації та інших засобів захисту необхідне для запобігання несанкціонованому доступу до даних. Забезпечення надійної безпеки даних є важливим для формування довіри клієнтів, оскільки у сучасних умовах конкуренції споживачі обирають ті сервіси, які можуть гарантувати захист їх особистих даних [8]. Автоматизовані системи бронювання та пошуку послуг мають важливу роль у розвитку бізнесу. Вони дозволяють спростити внутрішні процеси і підвищити задоволеність клієнтів завдяки зручному доступу до послуг. Клієнти можуть записатися на прийом у будь-який час, не звертаючись до адміністратора або не відвідуючи заклад особисто. За допомогою автоматизованих систем вони можуть обирати зручний час для візиту, отримувати нагадування про прийом і навіть скасовувати чи змінювати запис онлайн [9].

Автоматизація процесів бронювання і управління ресурсами приносить значні переваги для бізнесу. По-перше, вона знижує операційні витрати завдяки зменшенню потреби в персоналі для обробки записів та управління запасами. По-друге, автоматизація покращує ефективність роботи компанії завдяки точному плануванню графіків співробітників, оптимізації запасів та можливості швидко реагувати на зміни в попиті клієнтів [9]. Окрім того, автоматизовані системи дають змогу збирати й аналізувати дані про клієнтів і їхні вподобання, що сприяє поліпшенню обслуговування та збільшенню лояльності споживачів.

Отже, впровадження автоматизованої системи в перукарнях і салонах

краси є необхідним кроком для підвищення ефективності бізнесу. Така система дозволить не лише спростити управління ресурсами, але й забезпечить надійний захист даних і покращить якість обслуговування клієнтів. Як результат, компанії зможуть знизити операційні витрати і підвищити свою конкурентоспроможність.

1.2 Аналіз відомих рішень в галузі автоматизованої системи пошуку та бронювання послуг

Сучасні автоматизовані системи пошуку та бронювання послуг є невід'ємною частиною цифрової трансформації у багатьох галузях, таких як туризм, медицина, краса, розваги, а також у сфері громадського харчування та транспортних послуг. Ці системи дозволяють користувачам швидко і зручно знаходити потрібні послуги, здійснювати бронювання або запис на прийом у режимі реального часу. Ключовою перевагою таких систем є можливість забезпечення користувачів швидким доступом до необхідної інформації без необхідності додаткових телефонних дзвінків, особистих візитів чи інших трудомістких процедур. Розвиток інформаційних технологій сприяє постійному вдосконаленню таких систем, що робить їх більш доступними, зручними та функціональними для користувачів [10].

Автоматизовані системи бронювання значно полегшують процес управління ресурсами для підприємств. Для прикладу, у сфері туризму такі системи допомагають клієнтам знайти та забронювати готелі, авіаквитки, екскурсії або автомобілі, не виходячи з дому. У медицині системи електронного запису пацієнтів дозволяють швидко та зручно організувати прийом до лікаря, вибираючи зручний час і лікаря. У сфері краси, зокрема в перукарнях, автоматизовані системи допомагають клієнтам знайти майстрів, переглянути доступні послуги, ознайомитися з відгуками інших клієнтів та забронювати послуги на зручний час.

Значну роль у розвитку таких систем відіграє розвиток Інтернету, який

забезпечує миттєвий доступ до різноманітних платформ і рішень у сфері автоматизації бронювання. Інтернет є важливим ресурсом не лише для користувачів, але й для бізнесу, який за допомогою таких систем може керувати своїми ресурсами, відстежувати записи клієнтів, оптимізувати графіки роботи співробітників та аналізувати статистику бронювань для подальшого вдосконалення бізнес-процесів [11].

Для всебічного аналізу існуючих рішень було визначено низку ключових критеріїв оцінки:

- Функціонал онлайн-бронювання: здатність системи обробляти запити на бронювання послуг через інтернет у реальному часі.
- Наявність профілів користувачів: можливість зберігання та управління персональними даними клієнтів, історією бронювань і уподобаннями.
- Модулі управління персоналом: інструменти для планування графіків роботи, моніторингу зайнятості співробітників та оцінки їхньої ефективності.
- Функціонал інвентаризації: автоматизація процесів обліку матеріалів та ресурсів, необхідних для надання послуг.
- Інтеграція зі сторонніми API: підтримка зовнішніх інтерфейсів для обміну даними з іншими системами, такими як платіжні сервіси або CRM.
- Відкритість вихідного коду: можливість доступу до коду для модифікації та адаптації під конкретні вимоги бізнесу.

Цей логічний ланцюг оцінювання допоможе виявити ключові аспекти, які потрібно врахувати при розробці нової автоматизованої системи, здатної задовольнити потреби користувачів і підвищити ефективність бізнес-процесів [9].

Однією з ключових інновацій у галузі автоматизованих систем бронювання є розвиток глобальних дистрибуційних систем (GDS), таких як Amadeus, Sabre та Galileo, які спочатку були розроблені для авіакомпаній, готелів і туристичних агентств. Ці системи дозволяють автоматизувати процеси

пошуку та бронювання послуг, забезпечуючи масштабованість, швидкість обробки запитів і інтеграцію з широким колом постачальників послуг. Основними перевагами таких систем є широке охоплення та можливість отримання актуальної інформації про наявність послуг [12]. Однак їх використання потребує значних фінансових інвестицій у розробку та підтримку.

Переваги GDS:

- Широке охоплення різних видів послуг.
- Масштабованість і здатність обробляти великі обсяги транзакцій.
- Актуальність інформації та доступ до численних постачальників.

Недоліки GDS:

- Висока вартість інтеграції та підтримки.
- Складність використання для малого бізнесу через потребу в значних фінансових ресурсах.
- Можливість перевантаження сервісу через обробку величезної кількості запитів.

Іншою популярною платформою в індустрії краси є Fresha. Вона також забезпечує можливість бронювання послуг у перукарнях та салонах краси, а також ведення бази клієнтів, контролю за відвідуваністю і нарахування лояльності. Fresha особливо популярна серед малого бізнесу, оскільки не стягує плату за підписку, а лише комісію за транзакції. Це робить її привабливим рішенням для перукарень, які хочуть мінімізувати витрати на управління бронюванням. Крім того, Fresha інтегрує функцію автоматичних нагадувань клієнтам, що допомагає знизити кількість пропущених візитів [13].

Переваги Fresha:

- Безкоштовне використання для малого бізнесу (без абонентської плати, лише комісії за транзакції).
- Інтуїтивно зрозумілий інтерфейс, що спрощує керування бронюванням, базами клієнтів та фінансовими операціями.
- Автоматичні нагадування для клієнтів, що допомагає зменшити

кількість пропущених візитів.

- Інтеграція з іншими бізнес-інструментами, такими як бухгалтерія та управління запасами.

Недоліки Fresha:

- Залежність від транзакційної комісії: хоча платформа безкоштовна, комісії за транзакції можуть стати значним фінансовим навантаженням для бізнесів із великим обігом клієнтів.
- Недостатньо гнучка для великих мереж: Fresha здебільшого підходить для малого та середнього бізнесу; великі підприємства можуть потребувати більше налаштувань та кастомізації.
- Залежність від інтернет-з'єднання: платформа є повністю онлайн-сервісом, тому можливі проблеми з доступом через відсутність стабільного інтернету.
- Обмежені можливості інтеграції з зовнішніми CRM-системами для більш комплексного управління клієнтами та маркетингом.

Серед технологічних інновацій варто відзначити платформи Schedulicity і Square Appointments, які інтегрують платіжні системи, управління запасами та автоматизацію обслуговування клієнтів. Square Appointments дозволяє перукарням приймати онлайн-оплату, вести бухгалтерський облік і контролювати фінансові потоки в одному інтерфейсі, що спрощує роботу бізнесу [14].

Переваги Schedulicity і Square Appointments:

- Комплексність рішень: можливість виконання декількох завдань (бронювання, облік, фінансові операції) в одній системі.
- Управління персоналом: дозволяє відслідковувати робочий графік і продуктивність майстрів.

Спеціалізовані платформи: StyleSeat спрямована на професіоналів у сфері краси, надаючи інструменти для просування послуг через соціальні мережі. Це дозволяє майстрам залучати нових клієнтів безпосередньо через свої профілі, що сприяє зростанню популярності послуг [15].

Переваги StyleSeat:

- Оптимізація процесу бронювання: зменшує накладки у графіку та підвищує загальний рівень обслуговування клієнтів.
- Аналітичні інструменти: допомагають відстежувати найзатребуваніші послуги та адаптувати бізнес під потреби клієнтів.

Недоліки StyleSeat:

- Залежність від технічної інфраструктури: можливі проблеми з доступом до платформи через недостатній інтернет.
- Вартість підключення: платформи можуть стягувати високі комісії за транзакції, що вплине на фінансові результати бізнесу.

Переваги таких систем для перукарень очевидні: вони оптимізують процес бронювання, дозволяють уникати накладок у графіку, знижують адміністративне навантаження та підвищують загальний рівень обслуговування клієнтів [16]. Завдяки автоматизованим нагадуванням та можливостям управління клієнтськими базами, майстри можуть зосередитися на наданні послуг, а не на вирішенні організаційних питань. Однак, недоліком може бути залежність від технічної інфраструктури, зокрема інтернету, а також вартість підключення до таких платформ для деяких користувачів. Деякі з платформ можуть також стягувати високі комісії за транзакції, що може впливати на фінансовий результат бізнесу [17].

Загалом, автоматизовані системи пошуку та бронювання послуг у сфері перукарень демонструють високий рівень ефективності та адаптації до сучасних потреб клієнтів і бізнесу. Вони сприяють підвищенню конкурентоспроможності на ринку, забезпечують персоналізований підхід до кожного клієнта та впроваджують інноваційні рішення для покращення якості обслуговування. У майбутньому можна очікувати подальшого розвитку таких систем, зокрема завдяки використанню штучного інтелекту для персоналізованих рекомендацій, а також технологій віртуальної та доповненої реальності для надання клієнтам нових вражень при виборі послуг.

1.3 Формулювання вимог та специфікація до створюваної системи

Розробка автоматизованої системи для пошуку та бронювання послуг є важливою частиною створення сучасних онлайн-платформ для надання послуг [16]. Така система повинна не тільки відповідати основним вимогам функціональності, але й забезпечувати високий рівень продуктивності, безпеки, зручності використання та персоналізації для задоволення потреб користувачів різного віку та з різними технічними навичками [18].

Продуктивність системи є ключовим показником її успішності на ринку. Високий рівень продуктивності передбачає швидке та ефективне виконання операцій, що є важливим для збереження користувачів на платформі. Критичні фактори продуктивності включають такі аспекти [19]:

- **Час відгуку.** Це час, який потрібен системі для виконання запиту користувача. Чим швидше система відповідає на запит, тим кращий досвід отримують користувачі. Для досягнення оптимального часу відгуку можуть використовуватись технології кешування даних та оптимізації SQL-запитів.
- **Оптимізація використання ресурсів.** Розробка системи, яка раціонально використовує серверні ресурси, дозволяє зменшити витрати на інфраструктуру та збільшити кількість одночасних користувачів. Використання мікросервісної архітектури може допомогти у вирішенні питання масштабованості.
- **Навантаження на сервери.** Система повинна бути здатна витримувати великі навантаження під час пікових періодів. Наприклад, якщо велика кількість користувачів одночасно намагається забронювати послуги в популярних перукарнях, система повинна залишатися стабільною.

Для ілюстрації, формула, яка відображає продуктивність системи в залежності від часу відгуку та навантаження на сервери, може виглядати так:

$$P = \frac{T}{L} \tag{1.1}$$

де:

- P — продуктивність;
- T — час відгуку;
- L — навантаження на систему (кількість одночасних запитів).

Безпека є одним із найважливіших аспектів розробки будь-якого веб-додатку. Користувачі повинні бути впевнені, що їхні персональні дані, зокрема інформація про бронювання, особисті дані, та дані про платіжні карти, будуть захищеними. Основними принципами безпеки є [20]:

- Конфіденційність - це значає, що доступ до персональних даних має бути обмежений лише до тих користувачів і перукарів, які мають на це право. Для цього можуть використовуватися механізми аутентифікації та авторизації.
- Усі дані, які зберігаються або передаються через систему, мають бути захищеними від несанкціонованих змін. Використання протоколу HTTPS для шифрування даних під час їх передачі є стандартом у сучасних веб-додатках.
- Система повинна бути захищена від атак типу DoS (Denial of Service), які можуть зробити її недоступною для користувачів. Розробники мають передбачити механізми для запобігання таких атак.

Для підвищення рівня безпеки можуть використовуватись такі технології, як двофакторна аутентифікація (2FA), яка вимагає від користувачів надання другого етапу перевірки (наприклад, через SMS або мобільний додаток) перед доступом до системи.

Дизайн є ще однією важливою складовою успішної автоматизованої системи. Користувачі, незалежно від рівня їхньої технічної підготовки, повинні мати можливість легко орієнтуватися в інтерфейсі. Основні принципи дизайну для цієї системи включають [21]:

- Важливо, щоб інтерфейс був простим і зрозумілим. Наприклад, кнопки для бронювання послуг повинні бути великими та помітними, а всі основні функції – легкодоступними.

- Система має підтримувати різні розміри екранів і бути адаптованою для мобільних пристроїв. Це особливо важливо, оскільки більшість користувачів здійснюють бронювання через смартфони.
- Користувачі повинні швидко знаходити потрібні розділи сайту. Це досягається за допомогою грамотного розташування елементів та чіткої структури меню.

Сучасні веб-сервіси потребують персоналізованого підходу до кожного користувача. Це означає, що система повинна запам'ятовувати вподобання користувачів і пропонувати їм відповідний контент. Основні методи персоналізації включають [22]:

- Система повинна фіксувати дії користувачів, наприклад, які послуги вони бронюють найчастіше, і на основі цього пропонувати релевантні варіанти.
- Використання алгоритмів машинного навчання дозволить системі пропонувати користувачам нові послуги або знижки, виходячи з їхніх попередніх бронювань.
- Система може надсилати користувачам нагадування про майбутні бронювання або пропонувати спеціальні акції.

Для цього веб-сервісу існують певні функціональні вимоги, які він повинен реалізовувати. Основні функції для клієнтів включають:

- Аутентифікація: можливість входу до системи.
- Запис до перукарні: функція бронювання послуг.
- Редагування профілю: можливість змінювати особисті дані.

Для перукарів передбачені такі функції:

- Аутентифікація: вхід до системи.
- Перегляд записів: можливість переглядати заплановані бронювання.
- Внесення інформації: можливість редагувати деталі про послуги.

Для адміністраторів перукарні основні функції включають:

- Аутентифікація: вхід до системи.
- Перегляд записів: можливість переглядати всі бронювання та

редагувати їх.

Крім того, веб-сервіс повинен відповідати певним нефункціональним вимогам, таким як:

- Мова інтерфейсу: українська.
- Доступ до Інтернету: обов'язкова умова для роботи сервісу.
- Сумісність з браузерами: підтримка Google Chrome версії 57 і вище, Mozilla Firefox версії 52 і вище, Safari версії 11 і вище, Microsoft Edge версії 16 і вище.
- Технології бази даних: необхідність використання MongoDB версії 3.6 і вище.
- Операційні системи для розгортання: Windows (7, 8, 10) або Linux (Ubuntu 20.04 LTS, Fedora 32, CentOS 8).
- HTTP запити: всі взаємодії між додатками повинні відбуватися за допомогою HTTP запитів.

Розробка автоматизованої системи для пошуку та бронювання послуг є важливим кроком у модернізації та спрощенні процесу надання послуг у сфері краси. Сучасний ринок вимагає від таких платформ не лише надання базових функцій для користувачів, але й забезпечення високого рівня безпеки, продуктивності, зручності використання та персоналізації. У процесі створення системи було враховано численні вимоги, що спрямовані на забезпечення якості роботи платформи для різних груп користувачів — клієнтів, перукарів і адміністраторів. Це дозволить досягти головної мети проєкту — задоволення потреб користувачів та підвищення ефективності роботи перукарень.

Основним завданням системи є забезпечення швидкого та зручного пошуку послуг, доступних для бронювання, що дозволить клієнтам обирати перукарські послуги відповідно до своїх потреб і переваг. Для цього було розроблено інтуїтивний інтерфейс, який надає можливість швидко знаходити необхідну інформацію про перукарні, їхні послуги та доступні дати. Завдяки впровадженню функції фільтрації послуг за різними критеріями (розташування, ціна, рейтинг тощо), користувачі зможуть швидко вибирати оптимальні

варіанти для бронювання.

Однією з головних переваг системи є її висока продуктивність. Застосування методів оптимізації бази даних, включаючи використання кешування та оптимізації SQL-запитів, дозволяє значно скоротити час відповіді системи навіть при великій кількості одночасних запитів. Ця система буде здатна обробляти значну кількість користувачів, що дозволить їй залишатися стабільною під час пікових навантажень, коли попит на перукарські послуги зростає.

Забезпечення безпеки є ще одним важливим аспектом роботи системи. У сучасному світі, де кіберзагрози зростають, особливо важливою є захищеність особистих даних користувачів та їхніх фінансових операцій. Впроваджені механізми шифрування даних, а також багатофакторна аутентифікація надають гарантії конфіденційності інформації користувачів і захищають її від несанкціонованого доступу. Крім того, система передбачає заходи захисту від атак типу "brute force" та DDoS, що допоможе підтримувати стабільність і доступність сервісу [23].

Система також передбачає функцію персоналізації, яка дозволяє адаптувати роботу платформи під індивідуальні потреби користувачів. Використання алгоритмів машинного навчання дозволить системі аналізувати поведінку користувачів, запам'ятовувати їхні вподобання та пропонувати їм відповідні послуги. Це сприятиме збільшенню кількості повторних бронювань, підвищенню лояльності користувачів і, як результат, сприятиме зростанню популярності платформи серед клієнтів та перукарів [25].

Особливу увагу приділено зручності використання системи, що включає в себе продуманий і зрозумілий дизайн. Система розроблена таким чином, щоб користувачі з різними технічними навичками могли легко орієнтуватися в ній та здійснювати всі необхідні дії — від реєстрації до бронювання. Також було враховано важливість адаптивності системи, що дозволяє користувачам отримувати доступ до платформи з будь-якого пристрою — як настільного комп'ютера, так і мобільного телефону [26]. Адаптивний дизайн забезпечує

коректне відображення інтерфейсу на різних розмірах екранів, що дозволяє розширити аудиторію користувачів.

Окрім цього, розробка автоматизованої системи пошуку та бронювання послуг відповідає сучасним вимогам до взаємодії з користувачем. Впровадження системи рекомендацій на основі попередніх бронювань користувача дозволить підвищити персоналізованість пропозицій. Ця система допоможе клієнтам швидше знаходити необхідні послуги, заощаджуючи їх час і забезпечуючи зручний користувацький досвід.

Важливим елементом для успішної реалізації проєкту стало забезпечення чіткої взаємодії між різними компонентами системи. Вибір архітектури на основі мікросервісів дозволяє легко масштабувати систему та інтегрувати нові функції в майбутньому без значних витрат на переробку існуючої інфраструктури. Це дозволить платформі зростати разом із розвитком бізнесу та розширенням кількості користувачів, забезпечуючи стабільну роботу та гнучкість у впровадженні нових можливостей.

На завершення, розроблена система повністю відповідає вимогам сучасного ринку, забезпечуючи високий рівень безпеки, продуктивності, персоналізації та зручності використання. Вона є потужним інструментом для перукарів, що дозволяє їм оптимізувати процес роботи з клієнтами, а для клієнтів — забезпечує комфортне та швидке бронювання послуг [27]. Система готова до впровадження та подальшого масштабування, а також до інтеграції нових функцій і можливостей, що зробить її ще більш привабливою для різних категорій користувачів.

Успішна реалізація цього проєкту має потенціал значно змінити спосіб надання перукарських послуг, спростивши процес взаємодії між клієнтами та майстрами. Впровадження автоматизованої системи дозволить підвищити ефективність роботи перукарень, зменшити кількість незручностей для клієнтів і сприятиме подальшому розвитку індустрії краси в умовах цифрової трансформації.

1.4 Висновки

У цьому розділі проведено аналіз існуючих рішень у сфері автоматизованих систем пошуку та бронювання послуг. Розглянуто ключові функціональні можливості таких платформ, а також їхні недоліки, що дозволяє виявити потребу у вдосконаленні цих систем.

Аналіз показав, що більшість популярних рішень, представлених на ринку, мають певні обмеження, зокрема щодо функціоналу, гнучкості налаштувань та зручності використання. Це може створювати складнощі для кінцевих користувачів та адміністраторів. Наприклад, деякі системи не забезпечують повного управління розкладом майстрів або не підтримують інтеграцію з іншими необхідними сервісами.

Отже, результати аналізу свідчать про доцільність розробки власного інноваційного рішення, яке усуватиме недоліки існуючих платформ і пропонуватиме більш гнучкі та функціональні можливості для автоматизації процесів пошуку та бронювання послуг.

2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДЛЯ ПОШУКУ ТА БРОНЮВАННЯ ПОСЛУГ

2.1 Моделювання процесів пошуку та бронювання послуг

Моделювання процесу пошуку послуг є важливим етапом розробки системи, оскільки воно дозволяє створити чітке уявлення про те, як користувачі взаємодіють з платформою і які функції вони можуть використовувати для знаходження найбільш відповідних послуг. Для досягнення цього необхідно ретельно спроектувати послідовність дій, починаючи з введення запиту користувачем до отримання результатів пошуку. На першому етапі важливо визначити основні функціональні вимоги, серед яких ключовими є можливість пошуку послуг за різними критеріями, таких як тип послуги, місцезнаходження, ціна тощо. Крім того, слід передбачити функціональність фільтрації та сортування результатів, що дозволяє користувачам зручно відсортовувати послуги за ціною, рейтингом або іншими характеристиками.

Однією з важливих складових є механізм порівняння послуг, який дає змогу користувачеві оцінити кілька варіантів і вибрати найкращий. Для досягнення цієї мети потрібно створити чітку та зрозумілу інтерфейсну структуру, яка надасть користувачам усі необхідні інструменти для зручного перегляду і порівняння послуг. Процес пошуку та фільтрації повинен бути максимально інтуїтивно зрозумілим, щоб користувачі могли швидко знайти те, що їм потрібно, без необхідності глибоких технічних знань. Для досягнення цієї мети важливо також оптимізувати пошукові алгоритми для швидкої обробки запитів, що дасть змогу зберігати високу продуктивність системи навіть при великій кількості користувачів та послуг. Використання таких методів, як індексація даних та застосування алгоритмів пошуку, дозволить зменшити час на обробку запитів та підвищить ефективність системи. Для цього можна використати такі методи:

- Блок-схема процесу пошуку послуг: на схемі слід відобразити основні

етапи процесу — від введення користувачем пошукового запиту до отримання списку відповідних послуг [28].

Приклад блок-схеми:

- Користувач вводить запит (наприклад, "бронювання стрижки у перукарні").
- Система запитує надати доступ до геолокації, або у разі відхилення просить ввести місто
- Відбувається перевіряє наявність послуг у базі даних.
- Виконується фільтрація за ключовими параметрами (місцезнаходження, ціна, рейтинг).
- Повертається список доступних варіантів.

- Для математичного моделювання пошуку можемо використовувати класичний алгоритм пошуку по індексу або алгоритм фільтрації з використанням SQL-запитів. Якщо система використовує NoSQL-базу даних, то можна застосувати пошук через ключі або індекси в NoSQL базах [29].

Модель пошуку: Для ефективного пошуку послуг у системі можуть бути використані різні математичні моделі, що забезпечують точність і швидкість обробки запитів. Класичні алгоритми пошуку, такі як пошук по індексу, дозволяють системі швидко знаходити потрібні дані [30]. У системах, що використовують SQL-бази даних, запити можуть бути реалізовані через стандартні SQL-запити, наприклад:

```
SELECT * FROM services
WHERE location = 'Київ' AND price <= 1000
ORDER BY rating DESC;
```

У випадку використання NoSQL бази даних, таких як MongoDB, для здійснення пошуку можна застосувати запити через ключі або індекси, що забезпечує швидкий доступ до даних [31].

Математична модель пошуку може бути представлена як:

$$R(q) = \{s_1, s_2, \dots, s_n\} \quad (2.1)$$

де:

- R(q) — результат пошуку,
- q — запит користувача,
- s1, s2,…, sn — набір знайдених послуг, що відповідають запиту.

Алгоритм фільтрації: Після отримання списку послуг необхідно застосувати алгоритм фільтрації, щоб користувач міг вибрати найбільш відповідний варіант. Один із найпопулярніших підходів — це алгоритм ранжування на основі векторної моделі, який може бути реалізований через TF-IDF (Term Frequency-Inverse Document Frequency) або косинусну схожість [32].

Кожна послуга в системі може бути представлена у вигляді вектора характеристик, де кожен елемент відповідає певному критерію, таким як ціна, локація, рейтинг та інші атрибути [33]. Наприклад, вектор для послуги може виглядати наступним чином:

s = [price, location_score, rating_score]

s = [price, location_score, rating_score]

s = [price, location_score, rating_score]

Формула для косинусної схожості між вектором запиту користувача та вектором послуги може бути представлена так:

$$\frac{\sum_{i=1}^{n} q_i s_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \cdot \sqrt{\sum_{i=1}^{n} s_i^2}} \tag{2.2}$$

де:

- q — вектор запиту користувача,
- s — вектор послуги,
- n — кількість параметрів для кожної послуги.

Використання косинусної схожості дозволяє виміряти ступінь схожості між запитом користувача та доступними послугами на основі їхніх векторних представлень. Цей метод оцінює, наскільки близькі за змістом два набори даних, що дозволяє точно визначити, які послуги найбільше відповідають запиту. Косинусна схожість ефективно враховує відмінності в термінах, використовуваних у запитах і описах послуг, тим самим підвищуючи

релевантність пошукових результатів. Завдяки цьому, система може більш точно і швидко видавати найбільш підходящі пропозиції, що значно покращує користувацький досвід.

2.2 Розробка алгоритмів для автоматизації пошуку та бронювання

Цей підрозділ розглядає деталі реалізації алгоритмів, що відповідають за автоматизацію процесів пошуку та бронювання послуг. Ми зосередимося на принципах роботи алгоритмів, структурі даних, які вони використовують, та математичних моделях, які забезпечують їх ефективність [34].

Алгоритм бронювання складається з кількох етапів, які забезпечують перевірку доступності послуги, створення бронювання та відправлення підтвердження користувачу.

1. Вибір послуги користувачем

Після завершення процесу пошуку, користувач отримує список доступних послуг. При виборі конкретної послуги система отримує унікальний ідентифікатор цієї послуги (ID_service) [35]. На цьому етапі важливо зберегти інформацію про вибрані параметри, оскільки вони впливають на подальшу обробку запиту.

2. Перевірка доступності послуги

На цьому етапі система перевіряє, чи є обрана послуга доступною на вказані дати. Процес перевірки включає:

- Збір даних: Система отримує дані про вже існуючі бронювання для даної послуги. Це може включати інформацію про дати, на які вже виконані бронювання, та їх статус.
- Перевірка конфлікту: Для перевірки наявності конфлікту між запитом користувача та вже існуючими бронюваннями, використовується логічна формула:

$$\left((D_{\text{start}} < B_{\text{end}}) \wedge (D_{\text{end}} > B_{\text{start}})\right) \tag{2.3}$$

Де:

- Dstart та Dend — дати, які вводить користувач.
- Bstart та Bend — дати, на які вже заброньовано послугу.

Ця формула перевіряє, чи накладаються дати, і повертає істину, якщо послуга доступна, або хибу, якщо вже є активне бронювання [36].

3. Створення бронювання

Якщо послуга доступна, система переходить до створення нового бронювання. На цьому етапі:

- Формування об'єкту бронювання: Бронювання представляється у формі структури даних, що включає інформацію про користувача, обрану послугу та дати [30]. Наприклад, структура бронювання може виглядати наступним чином:

$$\text{Booking} = \{\text{Uid}, \text{ID}_{\text{service}}, \text{D}_{\text{start}}, \text{D}_{\text{end}}, \text{Status}\} \quad (2.4)$$

Де:

- Uid — це ідентифікатор користувача
- Status — статус бронювання (наприклад, "підтверджено").

- Збереження даних: Після формування об'єкту бронювання, система зберігає дані в базі даних. Це може бути реалізовано за допомогою SQL-запиту для реляційних баз даних або документного запису для NoSQL баз.

4. Надсилання підтвердження користувачу

Після успішного створення бронювання, система генерує повідомлення підтвердження, яке включає деталі бронювання, такі як дата, час, обрана послуга та статус. Це повідомлення може бути надіслано через електронну пошту, SMS або через мобільний додаток [37].

Алгоритм пошуку послуг включає кілька основних етапів, що забезпечують ефективний пошук відповідних послуг.

1. Введення пошукового запиту

Користувач вводить запит (наприклад, "готель у Києві"), який система обробляє. Це запит представляється у формі рядка, який потрібно аналізувати для отримання ключових слів і параметрів.

2. Формування списку послуг

Система виконує запит до бази даних для отримання всіх доступних послуг. Використовуючи SQL або NoSQL [38], система може отримати всі записи з таблиці/колекції послуг:

- Для SQL: SELECT * FROM services WHERE location LIKE '%Київ%'
- Для NoSQL: db.services.find({ location: /Київ/ })

3. Фільтрація послуг

Після отримання загального списку послуг, система застосовує фільтри на основі параметрів запиту [33]. Це включає:

- Використання метрик: Розрахунок рейтингу кожної послуги за формулою TF-IDF (Term Frequency-Inverse Document Frequency), яка дозволяє оцінити релевантність послуг до запиту користувача.

$$R(s_i, q) = \frac{TF(s_i, q) \cdot \text{IDF}(q)}{\sum_{j=1}^{n} TF(s_j, q) \cdot \text{IDF}(q)} \quad (2.5)$$

Де:

- TF(si,q) — частота терміна у послузі sis_isi.
- IDF(q) — зворотна частота документа для терміна qqq.
- R(si,q) — рейтинг послуги sis_isi у відповідності до запиту qqq.

4. Сортування результатів

Після фільтрації послуг, система сортує результати [39]. Це може бути реалізовано через алгоритми сортування, наприклад:

- QuickSort: Для ефективного сортування, особливо якщо список послуг великий. QuickSort працює за принципом "розділяй і володарюй", що забезпечує високу продуктивність при великих обсягах даних.

5. Повернення результатів

Після завершення всіх етапів обробки система формує для користувача результат у вигляді відфільтрованого та відсортованого списку послуг. Цей список містить лише ті послуги, які відповідають запитам та критеріям користувача. Інформація надається у зручному форматі, що забезпечує легкість використання та інтеграції. Найчастіше дані представлені у вигляді JSON-відповіді, яка є стандартом для передачі структурованої інформації. Такий

підхід дозволяє швидко обробляти отримані дані на стороні клієнта. У JSON міститься повна інформація про всі релевантні послуги, включаючи їхні характеристики та доступність. Цей формат також забезпечує сумісність із різними платформами та пристроями. Завдяки цьому користувач отримує точний і корисний результат, готовий для подальшого використання [40].

Для підвищення ефективності алгоритмів пошуку та бронювання важливо застосовувати оптимізації. Однією з основних оптимізацій є кешування.

Алгоритм кешування [41]

- Перевірка кешу: Перед виконанням запиту до бази даних система перевіряє, чи існує результат запиту у кеші. Якщо результат є, він негайно повертається користувачу.
- Виконання пошуку: Якщо результату немає у кеші, система виконує запит до бази даних, після чого зберігає результат у кеші для майбутніх запитів.
- Видалення застарілих даних: Для підтримання актуальності кешу важливо реалізувати механізм видалення старих або неактивних записів. Це може бути виконано за допомогою таймерів або тригерів, які автоматично видаляють дані після певного часу.

Проаналізувавши подану вище інформацію, доцільно буде створити схему алгоритму для пошуку послуг та бронювання послуг. Отже, на Рис. 2.1 наведено дану схему.

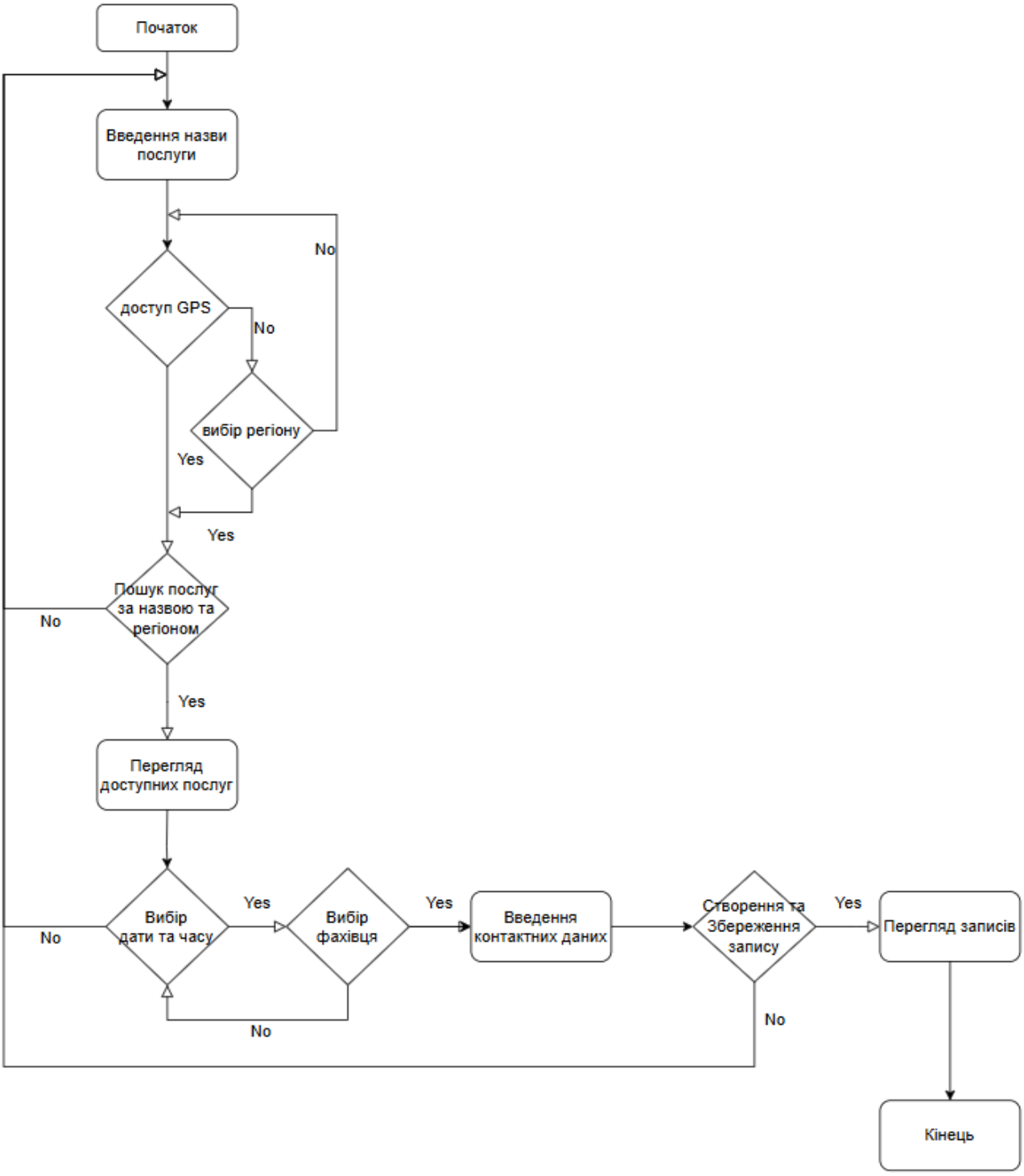


Рисунок 2.1 – Схема алгоритму реалізації автоматизованої системи для пошуку та бронювання послуг

Таким чином, алгоритми, розроблені для автоматизації процесів пошуку та бронювання, використовують складні математичні моделі, структуровані дані та оптимізаційні стратегії, які забезпечують ефективну і зручну роботу системи [42]. Це дозволяє забезпечити швидкий доступ до послуг, підвищити

їхню доступність та задоволеність користувачів.

2.3 Аналіз ефективності розроблених моделей та алгоритмів

Ефективність розроблених алгоритмів є ключовим аспектом, оскільки вона впливає на загальну продуктивність системи автоматизації пошуку та бронювання послуг. В цьому розділі ми розглянемо різні метрики [41], що використовуються для оцінки ефективності алгоритмів, включаючи обчислювальну складність, час виконання, використання пам'яті та точність результатів.

Обчислювальна складність алгоритму вимірює, як змінюється час виконання та використання ресурсів алгоритму в залежності від розміру вхідних даних [42]. Для аналізу розроблених алгоритмів важливо визначити найгірший сценарій, середній сценарій і найкращий сценарій їх виконання.

1. Пошуковий алгоритм

Для пошукового алгоритму, який базується на сортуванні, обчислювальна складність є:

$$O(n\log n) \tag{2.6}$$

де n — кількість елементів (послуг) у базі даних. Це пов'язано з тим, що алгоритми сортування, такі як QuickSort або MergeSort [43], мають логарифмічну складність через розділення елементів на менші підмножини. В результаті, чим більше число послуг у базі даних, тим більше часу знадобиться для їх обробки [42].

2. Алгоритм бронювання

Алгоритм бронювання виконує перевірку доступності послуги на основі запитів користувачів. Основна частина алгоритму перевіряє, чи накладаються дати запитів на вже існуючі бронювання [44]. У найгіршому випадку, коли всі бронювання потрібно перевірити, складність цього алгоритму може становити:

$$O(m) \tag{2.7}$$

де m — кількість уже існуючих бронювань для конкретної послуги.

Оскільки алгоритм повинен перевірити кожен елемент, його складність лінійна.

Час виконання алгоритму є ще однією важливою метрикою. Він визначає, скільки часу потрібно для виконання алгоритму при обробці певної кількості елементів [45]. Вимірювання часу виконання може здійснюватися за допомогою різних методів, таких як:

- Вимірювання в реальному часі: Проведення тестів в реальних умовах, щоб виміряти, скільки часу витрачається на виконання запиту. Це може включати запис часу до та після виконання алгоритму.
- Симуляції навантаження: Використання тестів для симуляції умов, коли декілька користувачів одночасно виконують запити, що дозволяє оцінити, як система справляється з навантаженням.

Використання пам'яті також є важливим аспектом, особливо для систем, які повинні обробляти великі обсяги даних [46]. Метрики використання пам'яті можуть включати:

- Динамічне виділення пам'яті: Оцінка пам'яті, яка потрібна для зберігання тимчасових змінних, структур даних та кешу.
- Статичне виділення пам'яті: Аналіз загального обсягу пам'яті, який використовується для зберігання інформації про послуги та бронювання у базі даних.

Важливо оптимізувати використання пам'яті для покращення загальної продуктивності системи [47], оскільки перевантаження пам'яті може призвести до зниження швидкості виконання алгоритмів.

Коректність бронювання — це критично важливий аспект, оскільки система повинна забезпечити, що одночасні запити від різних користувачів не призводять до конфліктів. Для забезпечення коректності бронювання використовують транзакції на рівні бази даних, які дотримуються принципів ACID (Atomicity, Consistency, Isolation, Durability) [48]:

- Атомарність (Atomicity): Забезпечує, що всі операції в транзакції виконуються успішно або жодна з них не виконується. Це запобігає частковому виконанню, яке може призвести до неконсистентності даних

[49].

- Узгодженість (Consistency): Гарантує, що транзакція переводить базу даних з одного узгодженого стану в інший, підтримуючи цілісність даних [50].
- Ізоляція (Isolation): Гарантує, що одночасні транзакції не впливають одна на одну, таким чином зберігаючи правильність операцій.
- Стійкість (Durability): Гарантує, що після успішного завершення транзакції її результати зберігаються навіть у разі збоїв системи [50].

Завдяки цим принципам, система може обробляти одночасні запити на бронювання, зберігаючи при цьому цілісність і коректність даних.

Точність результатів алгоритмів є ще одним важливим критерієм для оцінки їх ефективності [51]. Алгоритм має бути здатним правильно ідентифікувати доступні послуги, точно перевіряти їх наявність та коректно обробляти запити на бронювання [52].

Для перевірки точності можна використовувати:

- Тестування на контрольних даних: Введення відомих даних у систему для перевірки, чи алгоритм повертає очікувані результати.
- Аналіз помилок: Вивчення випадків, коли алгоритм дав неправильні результати, і проведення корекцій для покращення його точності.

Аналіз ефективності розроблених моделей і алгоритмів є важливою частиною процесу автоматизації пошуку та бронювання послуг, оскільки він дозволяє оцінити їх здатність до виконання поставлених завдань за мінімальних витрат ресурсів. Оцінка таких параметрів, як обчислювальна складність, час виконання, використання пам'яті і точність результатів, є необхідною для виявлення потенційних слабких місць у роботі алгоритмів. Цей аналіз дозволяє вчасно визначити можливості для оптимізації, що сприяє підвищенню загальної продуктивності системи та забезпечує кращий досвід для користувачів. Крім того, застосування принципів ACID (атомарності, узгодженості, ізоляції та стійкості) є важливим аспектом для забезпечення коректності операцій

бронювання, що забезпечує надійність та стабільність системи в цілому.

2.4 Висновок

У даному розділі було розглянуто моделювання процесів пошуку та бронювання послуг, розробку алгоритмів автоматизації цих процесів, а також проведено аналіз ефективності розроблених моделей та алгоритмів. Впровадження даних алгоритмів дозволить оптимізувати процес пошуку і бронювання, зменшити час обробки запитів і підвищити точність результатів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір інструментів та технологій для розробки системи

При розробці складної системи, зокрема автоматизованої платформи для пошуку та бронювання послуг, вибір технологій є одним із найважливіших етапів. Це пояснюється тим, що правильно підібрані інструменти дозволяють досягти оптимального балансу між продуктивністю, зручністю розробки, масштабованістю та безпекою. Основними критеріями вибору технологій у цьому проекті стали відповідність сучасним стандартам, забезпечення високої швидкості роботи системи та можливість її адаптації до зростання кількості користувачів і запитів. Крім того, важливим було створення рішення, яке є зрозумілим для розробників і може бути легко підтримано та вдосконалено в майбутньому.

Для реалізації фронтенд-частини проекту було обрано фреймворк Nuxt.js, який базується на Vue.js і надає широкі можливості для побудови універсальних додатків із серверним рендерингом (SSR). Однією з головних переваг Nuxt.js є його здатність забезпечувати швидке завантаження сторінок, що позитивно впливає на досвід користувачів, особливо у системах, які активно взаємодіють із великою кількістю клієнтів. Серверний рендеринг покращує індексацію сторінок пошуковими системами, що робить цей фреймворк ідеальним для проектів, які потребують високого рівня SEO-оптимізації.

Nuxt.js також спрощує організацію маршрутизації завдяки автоматичному створенню маршрутів на основі структури файлів, що зменшує обсяг рутинного коду. Ще однією важливою перевагою є інтеграція зі станом додатка через Vuex, що дозволяє ефективно керувати даними у великих проектах. Крім того, підтримка модульності в Nuxt.js забезпечує можливість легко додавати нові функції або інтегрувати сторонні бібліотеки без значних змін у кодовій базі. Однак фреймворк має й деякі недоліки, наприклад, складність налаштувань для специфічних випадків і більший обсяг стартового коду в порівнянні з

простішими підходами. Незважаючи на це, переваги Nuxt.js значно перевершують його недоліки, що робить його оптимальним вибором для цього проекту.

Бекенд-частина системи реалізована на основі NestJS – сучасного фреймворку для Node.js, який поєднує в собі переваги модульного підходу та використання TypeScript. Однією з головних причин вибору NestJS є його структурований підхід до побудови додатків, що відповідає принципам SOLID. Завдяки цьому код стає більш читабельним, а його підтримка – простішою. NestJS надає розробникам можливість використовувати вбудовані модулі, такі як WebSocket, GraphQL або REST API, що значно скорочує час на створення необхідної функціональності.

Ще однією перевагою NestJS є його інтеграція з іншими бібліотеками та фреймворками. Наприклад, у цьому проекті було використано бібліотеку TypeORM для роботи з базою даних. TypeORM забезпечує зручну взаємодію з реляційними базами даних, дозволяючи автоматично генерувати SQL-запити на основі моделей даних. Це значно зменшує ризик помилок і прискорює розробку.

Крім цього, NestJS підтримує розширюваність і масштабованість системи. Наприклад, для інтеграції з іншими сервісами використовувалися середовища Docker та Kubernetes, що спрощує розгортання і забезпечує стабільність роботи додатка навіть за високих навантажень. До недоліків NestJS можна віднести більш круту криву навчання для розробників, які раніше не працювали з TypeScript або не використовували модульний підхід. Проте в довгостроковій перспективі переваги цього фреймворку значно переважають його мінуси.

Для зберігання даних у проекті було обрано реляційну базу даних PostgreSQL. Цей вибір був зумовлений її стабільністю, продуктивністю та підтримкою складних запитів, що є важливим для систем, які працюють із великими обсягами даних. PostgreSQL надає широкі можливості для обробки структурованої інформації, а також підтримує зберігання нереляційних даних у форматі JSON. Це дозволяє комбінувати традиційний реляційний підхід із

сучасними методами роботи з даними, що робить систему більш гнучкою.

Ще однією перевагою PostgreSQL є її розширюваність. Вона дозволяє використовувати додаткові модулі, наприклад, для геопросторового аналізу або роботи з великими масивами даних. Завдяки цьому база даних здатна адаптуватися до специфічних вимог проекту. Окрім того, PostgreSQL забезпечує високий рівень безпеки, включаючи підтримку шифрування даних, що є критично важливим для збереження конфіденційності інформації в системах пошуку та бронювання послуг.

Однак використання PostgreSQL може вимагати додаткових ресурсів для налаштування та оптимізації запитів, особливо у великих проектах. Для цього у системі були впроваджені інструменти моніторингу та оптимізації, які дозволяють відстежувати продуктивність бази даних у реальному часі та своєчасно усувати потенційні вузькі місця.

Усі обрані технології були проаналізовані з точки зору їхніх переваг і недоліків, що дозволило зробити зважений вибір і забезпечити ефективність системи на всіх рівнях її роботи. Далі в тексті підрозділу будуть детально описані основні характеристики технологій, їхній вплив на продуктивність проекту, а також особливості їхньої інтеграції в загальну архітектуру системи.

Backend (Серверна частина)

1. Node js є сучасною платформою, яка дозволяє виконувати JavaScript-код на стороні сервера. Її популярність серед розробників пояснюється низкою значних переваг, що роблять її оптимальним вибором для створення продуктивних і масштабованих серверних додатків. У нашому випадку використання Node.js для серверної частини системи було зумовлено її потужними технічними можливостями, зокрема:

- Однією з ключових особливостей Node.js є подієво-орієнтована модель, яка використовує неблокуюче введення/виведення. Завдяки цьому платформа може обробляти велику кількість одночасних запитів, не створюючи надмірного навантаження на сервер. Наприклад, у традиційних багатопоточних моделях кожен запит

займає окремий потік, що може призвести до значного використання ресурсів системи. Натомість Node.js дозволяє обслуговувати тисячі запитів одночасно через подієвий цикл (event loop). Це особливо важливо для нашої системи, оскільки вона потребує швидкої обробки запитів від багатьох користувачів і забезпечення оперативного доступу до актуальної інформації.

- Node.js має одну з найбагатших екосистем серед платформ для розробки серверних додатків. Завдяки вбудованому менеджеру пакетів npm (Node Package Manager) розробники отримують доступ до мільйонів готових бібліотек і модулів. Ці бібліотеки дозволяють суттєво скоротити час розробки, адже більшість базових і навіть специфічних завдань уже вирішені. Наприклад, для обробки запитів користувачів, взаємодії з базою даних або реалізації автентифікації існують популярні пакети, які легко інтегрувати у проект. Для нашої системи використання бібліотек із npm дозволило оптимізувати розробку, зосередивши увагу команди на реалізації унікальної бізнес-логіки, замість створення базової функціональності з нуля.
- Node.js побудований на базі V8 – рушія JavaScript, розробленого компанією Google для браузера Chrome. Цей рушій забезпечує високу швидкість виконання JavaScript-коду завдяки компіляції коду у машинний код безпосередньо перед виконанням. Як результат, серверні додатки на Node.js працюють швидше та ефективніше, ніж ті, що побудовані на деяких інших платформах. У нашій системі, яка вимагає високої швидкодії для обробки великого обсягу запитів, ця особливість стала однією з вирішальних при виборі технології.
- Node.js є ідеальним рішенням для додатків, які потребують обміну даними в реальному часі. Завдяки підтримці WebSocket-з'єднань платформа забезпечує двосторонній зв'язок між сервером і клієнтом. Це означає, що інформація може оновлюватися миттєво без необхідності повторних запитів з боку клієнта. Для нашої системи, де

необхідно забезпечити актуальність інформації в режимі реального часу, ця можливість є критично важливою. Наприклад, у випадках, коли користувачі повинні отримувати оновлення даних миттєво після внесення змін, Node.js допомагає реалізувати таку функціональність ефективно та з мінімальними витратами ресурсів.

Node.js є оптимальним вибором для розробки серверної частини нашої системи завдяки своїй продуктивності, можливостям роботи з великими навантаженнями та підтримці реального часу. Асинхронна модель обробки запитів, доступ до багатої екосистеми npm і використання високопродуктивного рушія V8 дозволяють нам створювати надійні, масштабовані й швидкі рішення. У результаті це забезпечує високу якість роботи системи та задоволення потреб користувачів [56].

2. Для побудови серверної архітектури було обрано фреймворк NestJS, який є одним із найпотужніших інструментів для розробки серверних додатків у JavaScript та TypeScript. Цей фреймворк базується на архітектурних принципах, що використовуються у популярних корпоративних рішеннях, зокрема Angular, і забезпечує масштабованість, підтримку сучасних технологій та зручність розробки. Рішення обрати саме NestJS було обумовлено його численними перевагами:

- Модульна структура. NestJS дозволяє організовувати проект у вигляді модулів, що спрощує підтримку та розвиток системи. Це важливо для масштабованих проектів, де можна легко додавати нові модулі без значних змін у наявній архітектурі.
- Підтримка TypeScript. NestJS з самого початку розроблявся з урахуванням використання TypeScript, що дозволяє писати більш безпечний та підтримуваний код завдяки статичній типізації.
- Інтеграція з сучасними технологіями. Фреймворк дозволяє легко інтегрувати GraphQL, WebSockets, мікросервіси, що робить його універсальним інструментом для розробки різних типів додатків.

Серед інших популярних фреймворків, таких як Express.js та Koa.js,

NestJS виділяється своєю архітектурною перевагою – модульністю. Express.js, наприклад, є більш мінімалістичним і вимагає додаткового налаштування для організації проекту у вигляді модулів. У складних додатках це може призвести до заплутаної структури коду, яка ускладнює його підтримку. Koa.js також пропонує гнучкість, але не має готових рішень для інтеграції сучасних технологій, таких як GraphQL або WebSockets, що доступні в NestJS.
NestJS забезпечує готову основу для побудови складних додатків із мінімальними зусиллями на початковому етапі, надаючи розробникам можливість зосередитися на бізнес-логіці

NestJS є потужним і сучасним фреймворком, який ідеально підходить для розробки серверної архітектури нашої системи. Його модульна структура, підтримка TypeScript та інтеграція з передовими технологіями дозволяють створювати масштабовані, продуктивні й зручні у підтримці рішення. У порівнянні з іншими фреймворками, NestJS забезпечує оптимальний баланс між гнучкістю, функціональністю та зручністю використання, що робить його відмінним вибором для реалізації сучасних серверних додатків [57].

3. Для взаємодії з базою даних у нашій системі було обрано TypeORM — об'єктно-реляційний маппер (ORM), який надає зручні інструменти для роботи з даними без необхідності писати складні SQL-запити вручну. Цей вибір базувався на його широких можливостях і перевагах, які роблять розробку ефективнішою та зручнішою. TypeORM був обраний через такі переваги:

- Підтримка TypeScript. Як і NestJS, TypeORM підтримує TypeScript, що дозволяє писати типізовані запити до бази даних та знижує ймовірність помилок під час компіляції.
- Міграції бази даних. TypeORM надає зручний механізм для створення та керування міграціями бази даних, що полегшує її оновлення при внесенні змін у структуру таблиць.
- Підтримка складних запитів. Одна з сильних сторін TypeORM полягає в можливості виконання складних запитів, що дозволяє ефективно працювати з великими обсягами даних.

На ринку ORM існує кілька альтернатив, серед яких можна виділити Sequelize і Prisma. Sequelize — це популярний ORM, який підтримує широкий спектр баз даних і функцій, однак його архітектура менш орієнтована на використання TypeScript. Prisma, у свою чергу, забезпечує сучасний підхід до роботи з базами даних, але має специфічний синтаксис і більше орієнтований на створення нового проекту, ніж інтеграцію з існуючими рішеннями. TypeORM був обраний через кілька причин:

- Він краще інтегрується з NestJS, що дозволяє зменшити складність налаштування та покращує підтримку проекту.
- TypeORM забезпечує більш традиційний підхід до роботи з базами даних, що полегшує адаптацію розробників, які знайомі з іншими ORM.
- Він пропонує широкий набір інструментів для роботи з реляційними базами даних, що робить його ідеальним вибором для проектів зі складною архітектурою.

TypeORM став основним інструментом для роботи з базою даних у нашому проекті завдяки своїм численним перевагам, таким як інтеграція з TypeScript, підтримка міграцій і можливість виконання складних запитів. У поєднанні з NestJS, TypeORM забезпечує високий рівень продуктивності, зручності та стабільності, що дозволяє ефективно реалізовувати бізнес-логіку й масштабувати систему в майбутньому [58].

Frontend (Клієнтська частина)

1. Для розробки клієнтської частини нашої системи було обрано фреймворк Nuxt.js. Це потужний інструмент, побудований на основі Vue.js, який надає зручні механізми для створення сучасних веб-додатків з акцентом на продуктивність, SEO-оптимізацію та підтримку масштабованості. Рішення про використання Nuxt.js ґрунтувалося на його численних перевагах, які сприяють швидкій і ефективній розробці, а також забезпечують високу якість готового продукту. Основні причини вибору Nuxt.js:

- Компонентний підхід. Як і Vue.js, Nuxt.js дозволяє організувати

інтерфейс у вигляді окремих компонентів, що сприяє повторному використанню, зниженню дублювання коду та полегшує підтримку коду.

- Серверний рендеринг та SEO. Nuxt.js підтримує серверний рендеринг з коробки, що значно покращує SEO та швидкість завантаження сторінок, роблячи його чудовим вибором для створення SEO-оптимізованих додатків.
- Автоматична маршрутизація та структура проекту. Nuxt.js пропонує зручну автоматичну маршрутизацію, побудовану на основі структури файлів проекту, що спрощує налаштування та підтримку великих додатків.
- Широка підтримка спільноти та розвинена екосистема. Nuxt.js має активну спільноту та багату екосистему плагінів і модулів, що дозволяє швидко інтегрувати додаткові функції та отримувати підтримку від спільноти розробників

Компонентна архітектура дозволяє не тільки спростити процес розробки, але й зробити код більш структурованим і зрозумілим для інших членів команди.

Альтернативами могли бути такі фреймворки, як React або Angular, але з огляду на поставлені вимоги до клієнтської частини системи, Nuxt.js є оптимальним вибором, оскільки він:

- Забезпечує високу продуктивність завдяки серверному рендерингу.
- Підтримує створення модульної архітектури з компонентним підходом.
- Пропонує простоту налаштування маршрутизації та організації проекту.
- Має активну спільноту та багату екосистему модулів, що сприяє швидкому впровадженню нових функцій.

Таким чином, використання Nuxt.js дозволяє реалізувати всі функціональні вимоги до клієнтської частини, забезпечуючи її

масштабованість, підтримку SEO та зручність розробки. [59].

2. Розробка складних веб-додатків передбачає необхідність ефективного управління станом. Стан відображає поточний контекст програми, включаючи інформацію про користувача, налаштування інтерфейсу, завантажені дані та інші параметри. Для управління станом у клієнтській частині було обрано бібліотеку Vuex, яка є офіційним рішенням для екосистеми Vue.js. У цьому розділі розглянуто основні переваги, функціональні можливості Vuex та його вплив на структуру проєкту [60].

Vuex є бібліотекою для централізованого управління станом, що дозволяє зберігати весь стан програми у спеціальному сховищі (store) та забезпечує доступ до нього з будь-якого компонента. Використання Vuex надає низку переваг:

1. Централізоване управління станом. Усі дані програми зберігаються в одному місці, що робить стан додатку передбачуваним і спрощує його обробку. Це особливо важливо для великих додатків із розгалуженою архітектурою, де взаємодія між компонентами може бути складною. Наприклад, при зміні стану в одному компоненті Vuex автоматично оновлює інші компоненти, які залежні від цих даних.
2. Розділення логіки управління станом. Vuex пропонує чітку структуру для організації логіки управління станом:
    - Mutations (мутації). Використовуються для синхронного внесення змін у стан. Це забезпечує прозорість у відстеженні змін через DevTools.
    - Actions (дії). Дозволяють виконувати асинхронні операції, такі як завантаження даних із сервера, перед внесенням змін у стан.
    - Getters (гетери). Слугують для обчислення похідних даних на основі стану.
3. Масштабованість та модульність. Vuex дозволяє розділяти стан на окремі модулі, кожен із яких може мати свій набір стану, мутацій, дій та гетерів. Це полегшує підтримку та розширення великих додатків,

оскільки розробники можуть працювати з окремими модулями, не впливаючи на інші частини програми.

4. Тісна інтеграція з Vue.js. Vuex повністю підтримує реактивність Vue.js, що забезпечує автоматичне оновлення інтерфейсу при зміні стану. Це спрощує роботу розробників, оскільки їм не потрібно вручну синхронізувати стан із UI-компонентами.

При виборі інструменту для управління станом було враховано низку критеріїв, таких як інтеграція з Vue.js, простота налаштування, підтримка модульної архітектури та можливість масштабування. Vuex був обраний через його відповідність цим критеріям, а також через такі переваги:

- Стандартизація процесів управління станом. Використання Vuex забезпечує єдиний підхід до роботи зі станом, що полегшує навчання нових членів команди та спрощує обмін знаннями між розробниками.
- Прозорість та налагоджуваність. Завдяки інтеграції з Vue DevTools Vuex надає можливість відстежувати всі зміни стану у реальному часі, що суттєво полегшує процес налагодження.
- Гнучкість у роботі з асинхронними операціями. Actions дозволяють виконувати асинхронні операції, такі як API-запити, без порушення основних принципів управління станом.

У процесі вибору інструменту для управління станом було розглянуто альтернативні рішення, такі як Redux та MobX.

- Redux. Redux є популярною бібліотекою для управління станом, яка широко використовується в екосистемі React. Основною перевагою Redux є його універсальність, що дозволяє використовувати його з будь-якими фреймворками. Однак, порівняно з Vuex, Redux вимагає більше налаштувань та додаткового коду для інтеграції.
- MobX. MobX є ще одним рішенням для управління станом, яке надає більшу гнучкість і спрощує написання коду завдяки автоматичній реактивності. Проте MobX не забезпечує централізованого управління станом і може ускладнювати відстеження змін у великих додатках.

Обидва рішення мають свої переваги, але Vuex був обраний через його тісну інтеграцію з Vue.js, стандартизований підхід до управління станом і вбудовану підтримку модульної архітектури.

Використання Vuex для управління станом дозволяє ефективно вирішувати завдання, пов’язані з централізованим зберіганням даних, забезпечуючи масштабованість та модульність додатку. Основними перевагами є:

- Єдиний підхід до управління станом, що полегшує підтримку та розвиток проекту.
- Зручність у роботі з асинхронними операціями завдяки розподілу відповідальності між actions та mutations.
- Повна інтеграція з екосистемою Vue.js, що спрощує використання та підвищує продуктивність розробки.

Таким чином, Vuex є оптимальним вибором для управління станом у клієнтській частині, забезпечуючи високу продуктивність, передбачуваність стану та зручність розробки.

База даних

У рамках розробки даного проєкту було обрано PostgreSQL як основну реляційну базу даних. Це рішення було ухвалене на основі низки переваг цієї СУБД, зокрема її продуктивності, надійності та здатності ефективно працювати з великими обсягами даних. PostgreSQL є однією з найпопулярніших та найбільш стабільних систем управління базами даних, яка підтримує розширені можливості для роботи з даними, що забезпечує високу гнучкість в організації зберігання та обробки даних. [61]:

- Висока продуктивність. Однією з основних переваг PostgreSQL є її здатність ефективно обробляти великі обсяги даних та виконувати складні запити. Завдяки індексації, оптимізації запитів та підтримці масштабованості, PostgreSQL може впоратися з великими навантаженнями, що робить її підходящою для складних систем, які потребують швидкої обробки великих обсягів інформації. Це важливо

для забезпечення високої продуктивності в реальному часі та для підтримки швидкості роботи додатка, навіть при значних навантаженнях.

- Розширена підтримка SQL. PostgreSQL підтримує стандарт SQL, що дозволяє писати ефективні запити для реляційних баз даних. Крім цього, вона надає додаткові можливості, які підвищують гнучкість роботи з даними. Однією з таких можливостей є підтримка CTE (Common Table Expressions), яка дозволяє покращити читабельність та підтримуваність складних запитів. Інша важлива особливість — тригери та збережені процедури, які дозволяють автоматизувати обробку даних без необхідності вручну виконувати складні операції. Це значно полегшує управління даними та забезпечує надійність операцій, що виконуються на сервері.
- Надійність та безпека. PostgreSQL забезпечує високий рівень надійності завдяки підтримці транзакцій відповідно до стандарту ACID, що гарантує цілісність даних під час будь-яких операцій з ними. Крім того, система має вбудовану підтримку для бекапів і відновлення даних, що дозволяє забезпечити безпеку в разі збоїв або помилок у роботі з базою. Завдяки Write-Ahead Logging PostgreSQL може відновити дані до стану на момент останнього успішного запису, що робить систему стійкою до втрат даних при несподіваних відключеннях.
- Гнучка система прав доступу. Для забезпечення безпеки даних у PostgreSQL передбачена гнучка система прав доступу, яка дозволяє контролювати доступ до даних на різних рівнях. Можна налаштувати доступ для окремих користувачів або груп користувачів на рівні таблиць, рядків і навіть окремих стовпців. Ця можливість є особливо важливою для систем, де необхідно обмежити доступ до чутливих даних або забезпечити їх конфіденційність.

Вибір PostgreSQL як основної бази даних для цього проєкту був

обумовлений рядом важливих факторів. Серед них необхідність обробки великих обсягів даних, складність виконуваних запитів і вимоги до високої продуктивності та надійності системи. PostgreSQL надає потужні можливості для роботи з великими наборами даних, що є важливим для даного проєкту. Вона також забезпечує ефективне виконання складних SQL-запитів і пропонує широкі можливості для оптимізації запитів. Окрім цього, PostgreSQL має вбудовані інструменти для забезпечення цілісності та безпеки даних, що є критично важливим для збереження конфіденційності і надійності інформації в системі.

Порівнюючи PostgreSQL з іншими реляційними базами даних, такими як MySQL або MariaDB, можна відзначити, що хоча ці системи також є популярними і мають свої переваги, PostgreSQL має кілька важливих аспектів, які зробили її найбільш підходящим вибором для цього проєкту. Наприклад, MySQL, хоч і є швидким у простих операціях, не підтримує деякі складніші функції, які є в PostgreSQL, такі як CTE, рекурсивні запити або підтримка JSONB, що є важливими для розробки складних аналітичних додатків.

У порівнянні з MariaDB, яка є форком MySQL, PostgreSQL надає більше можливостей для гнучкої роботи з даними, зокрема завдяки підтримці більш складних запитів і оптимізованих механізмів для обробки транзакцій. [62].

Правильний вибір інструментів та технологій для розробки системи є критично важливим етапом проектування. У цьому підрозділі було проаналізовано та обґрунтовано рішення щодо використання конкретних технологій, таких як Node.js і NestJS для серверної частини, React для клієнтської частини, PostgreSQL для зберігання даних. Вибрані технології забезпечують високу продуктивність, масштабованість і гнучкість системи, що дозволяє задовольняти поточні потреби проекту та легко адаптуватися до майбутніх змін. Усі ці рішення також сприяють підтримці чистої архітектури та застосуванню найкращих практик програмування, що є важливим фактором для довгострокової підтримки та розвитку системи [63].

3.2 Розробка структури програмного забезпечення

Розробка автоматизованої системи для пошуку та бронювання послуг передбачає створення багаторівневої архітектури, що забезпечує гнучкість, надійність і масштабованість. Дана система реалізована на основі Nest.js для серверної частини та Nuxt.js для клієнтського інтерфейсу. Використання цих технологій обумовлене їхньою здатністю забезпечити швидку обробку запитів, інтеграцію з базами даних, а також оптимізацію інтерфейсу користувача, що підвищує зручність використання і покращує досвід користувачів.

В процесі розробки програмного забезпечення було обрано архітектуру на основі шаблону MVC (Model-View-Controller), який дозволяє чітко відокремити логіку бізнесу від представлення та управління даними. Цей підхід забезпечує легкість в підтримці та розширенні системи, що особливо важливо для автоматизованих систем бронювання, де часті зміни в бізнес-логіці та вимоги до інтерфейсу користувача є звичним явищем [64].

Для забезпечення зручної та безпечної взаємодії клієнтської і серверної частини застосовано також патерн "Посередник" (Mediator), що дозволяє мінімізувати зв’язність між компонентами і забезпечує централізоване керування обміном даними. Це є важливим елементом для оптимізації процесу бронювання, оскільки система отримує доступ до багатьох ресурсів одночасно (наприклад, перевірка наявності вільних місць або послуг у реальному часі) та обробляє великі обсяги запитів [65].

Система побудована з декількох функціональних модулів, кожен із яких відповідає за виконання конкретних завдань та має чітко визначену область відповідальності. Кожен модуль працює автономно, але водночас взаємодіє з іншими через визначені інтерфейси. Це забезпечує модульність, масштабованість та спрощує підтримку і розвиток системи:

1. Модуль користувача (User Module). Відповідає за реєстрацію, авторизацію та управління обліковими записами користувачів. В рамках цього модуля реалізовано механізми автентифікації та

авторизації, які побудовані на основі OAuth 2.0, що забезпечує безпечний доступ до особистих даних користувачів та їхнього профілю. Також впроваджено шаблон "Одинак" (Singleton) для управління сесіями користувачів, що дозволяє уникнути дублювання сесій і забезпечує єдиний контроль за доступом.

2. Модуль бронювання (Booking Module). Цей модуль забезпечує основний функціонал системи - бронювання послуг. Реалізація модулю побудована з використанням патерну "Фасад" (Facade), що дозволяє приховати складність операцій бронювання та обробки даних. Він взаємодіє з базою даних для перевірки доступності ресурсів у режимі реального часу, проводить синхронізацію з календарями та зовнішніми сервісами, такими як платіжні системи. Використання фасаду дозволяє зменшити кількість залежностей між модулями, роблячи їх більш незалежними один від одного, що спрощує їх тестування і підтримку.
3. Модуль пошуку (Search Module). Пошуковий модуль побудований з використанням шаблону "Декоратор" (Decorator), що дає можливість додавати або видаляти функції пошуку без зміни основної логіки. Це дозволяє підлаштовувати функціонал пошуку під вимоги користувачів, надаючи можливість фільтрації результатів за різними критеріями, такими як місцезнаходження, ціна, рейтинг та доступність.
4. Адміністративний модуль (Admin Module). Цей модуль створено для управління системою, моніторингу її стану, створення власного бізнесу та проведення аналітики. Використовуючи шаблон "Стратегія" (Strategy), адміністративний модуль надає гнучкі можливості для налаштування алгоритмів обробки даних, управління користувачами та генерації звітів.

Для взаємодії між різними модулями та компонентами було використано RESTful API, що дозволяє стандартизувати обмін даними і забезпечує сумісність з іншими сервісами та додатками. Основною особливістю є

застосування патерну "Адаптер" (Adapter), який забезпечує коректне перетворення даних між модулями. Це особливо корисно при роботі з різними форматами даних, що зберігаються в базі даних, а також при інтеграції з зовнішніми сервісами [66].

На рис. 3.1 зображено UML-діаграму послідовностей, яка ілюструє процес пошуку та бронювання послуг користувачем у системі. Ця діаграма детально описує основні етапи взаємодії: введення назви послуги, доступ до GPS або вибір регіону, пошук послуг за назвою та регіоном, перегляд доступних варіантів, вибір дати й часу, а також вибір фахівця. Після цього користувач вводить свої контактні дані, створює та зберігає запис, а також отримує можливість переглядати вже створені записи.

Діаграма відображає логічну послідовність дій, що забезпечує узгодженість процесів та цілісність даних на всіх етапах взаємодії. Крім того, такий підхід сприяє зручності для користувачів, оскільки кожен крок є інтуїтивно зрозумілим і не потребує значних зусиль. Система забезпечує оперативний пошук послуг, оптимізуючи процес вибору за допомогою фільтрів, таких як геолокація та наявність фахівців. У результаті користувач отримує швидкий доступ до релевантної інформації, що допомагає уникнути можливих помилок у процесі бронювання.

Важливим елементом є збереження всіх створених записів у єдиній базі даних, що дозволяє уникнути дублювання інформації та спрощує її подальше використання. Це рішення забезпечує не лише зручність для кінцевого користувача, а й оптимізує роботу адміністративної частини системи, полегшуючи управління записами.

Рисунок 3.1 – UML-діаграма послідовності програмного забезпечення

Клієнтська частина, реалізована на Vue.js, побудована з дотриманням принципів доступності (WCAG), що забезпечує зручність для всіх користувачів. Для структурування інтерфейсу було застосовано компонентний підхід, який дозволяє повторно використовувати елементи інтерфейсу, що полегшує підтримку і розширення системи. Компоненти були створені з

урахуванням шаблону "Компонент" (Component), що робить їх гнучкими та легко налаштовуваними [67].

Інтерфейс системи реалізовано з підтримкою адаптивного дизайну, що забезпечує його коректне відображення на пристроях різних розмірів та типів. Використання шаблону "Композиція" (Composition) для управління станом додатку дозволяє легко адаптувати інтерфейс залежно від змін у стані модуля чи параметрах, заданих адміністраторами. Такий підхід робить систему гнучкою та зручною для налаштувань, сприяючи кращому користувацькому досвіду.

Клієнтська частина проекту побудована з використанням компонентного підходу, де кожен компонент відповідає за певну функціональність. Ці компоненти можуть багаторазово використовуватися в різних частинах інтерфейсу, що знижує дублювання коду та полегшує підтримку проекту [68]. На рис. 3.2 продемонстровано організацію файлів і папок, яка забезпечує логічну структуру та інтуїтивно зрозумілу навігацію для розробників.

Структура коду спроектована таким чином, щоб спростити його обслуговування, забезпечуючи можливість легкого розширення функціоналу. Завдяки цьому нові функції можна швидко інтегрувати без порушення існуючої логіки. Логічна організація папок також сприяє командній роботі, оскільки всі учасники проекту можуть легко орієнтуватися в кодовій базі.

Додатково, компонентна архітектура дозволяє ізолювати функціональні частини інтерфейсу, що підвищує стабільність та знижує ризик появи помилок при внесенні змін. Це робить систему надійною та масштабованою, забезпечуючи її відповідність сучасним вимогам до веб-додатків.

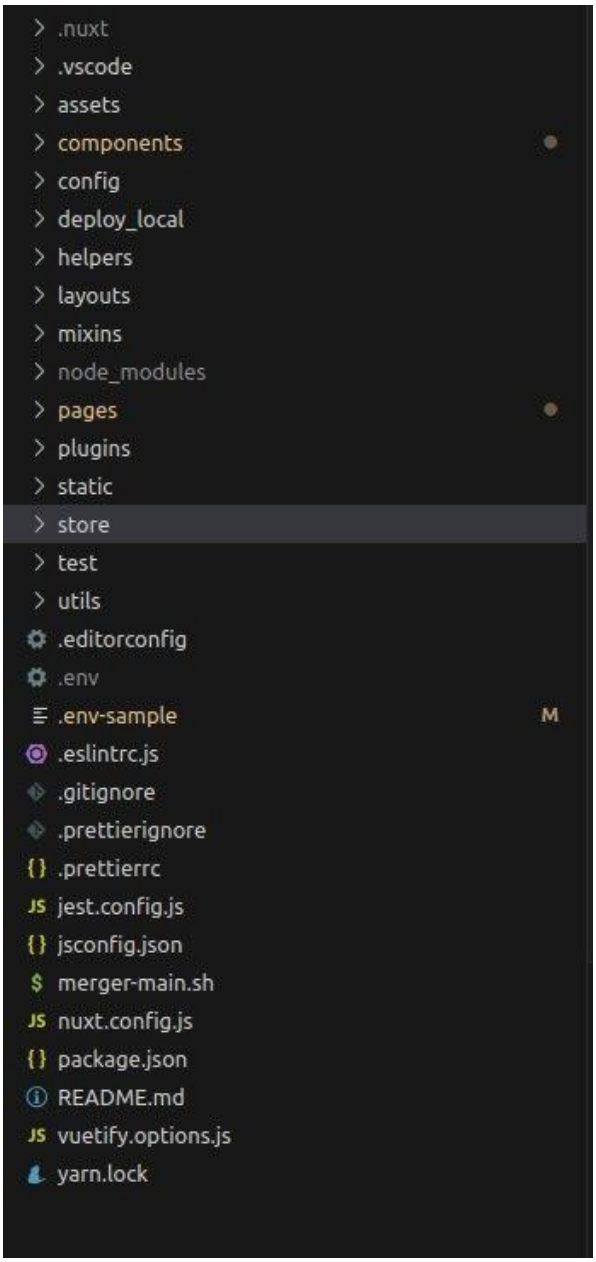


Рисунок 3.2 – Структура проекту

Рис. 3.2 ілюструє компонентну структуру клієнтської частини, яка забезпечує зручність підтримки і розширення системи. Компоненти побудовані з урахуванням принципів повторного використання та легкої модифікації, що дає можливість швидко адаптувати інтерфейс під нові вимоги або змінювати окремі елементи без суттєвого впливу на інші частини системи. Завдяки адаптивному дизайну інтерфейс системи може коректно відображатися на різних пристроях, що покращує досвід користувачів незалежно від розміру екрана.

Для зберігання даних було обрано базу даних PostgreSQL, яка добре підходить для обробки великої кількості записів і дозволяє легко керувати транзакціями. Використання ORM (Object-Relational Mapping) дозволяє знизити складність доступу до даних і уникнути SQL-ін’єкцій. Було впроваджено шаблон "Кешування" (Caching), який дозволяє зберігати дані в кеші, що значно зменшує навантаження на базу даних і прискорює виконання запитів [69].

Розроблена система автоматизованого пошуку та бронювання послуг відповідає сучасним вимогам до продуктивності, безпеки та зручності користування. Використання Nest.js і Vue.js дозволяє досягти високої продуктивності як на сервері, так і на клієнті, а застосовані патерни проектування роблять систему гнучкою для подальшого розширення та адаптації під нові вимоги ринку.

3.3 Реалізація інтерфейсу користувача

Реалізація інтерфейсу користувача є однією з ключових складових розробки сучасних веб-додатків, оскільки він безпосередньо впливає на зручність взаємодії користувача з системою та її ефективність у виконанні поставлених завдань. Від правильного проектування інтерфейсу залежить не лише зовнішній вигляд програми, але й зручність доступу до її функцій, що важливо для забезпечення позитивного користувацького досвіду. У цьому підрозділі детально розглянуто процес створення інтерфейсу для нашої системи, включаючи його основні архітектурні рішення та ключові компоненти, які формують взаємодію користувача з додатком.

Інтерфейс був поділений на декілька основних компонентів, кожен з яких відповідає за свою частину функціональності.

Наприклад:

1. Компоненти для введення даних: Форма для реєстрації користувача, форми для оновлення даних, а також компоненти для пошуку та фільтрації послуг, де користувач може вводити запити для обробки

системою та знаходити необхідні варіанти.

2. Компоненти для відображення даних: Таблиці, списки, картки, що динамічно відображають дані з сервера. Для відображення великих обсягів даних була використана пагінація та lazy loading для оптимізації продуктивності.
3. Компоненти для взаємодії з користувачем: Модальні вікна, спливаючі повідомлення про помилки та успіхи, підтвердження дій. Це забезпечує комфортну взаємодію з користувачем і надає йому інтуїтивно зрозумілу інформацію.

Для досягнення високої зручності користування інтерфейс був розроблений з урахуванням принципів адаптивного дизайну, що дозволяє забезпечити коректну роботу додатка на різних пристроях, від мобільних телефонів (Рис. 3.2) до десктопних комп'ютерів (Рис. 3.3).

1. Медіазапити (media queries) дозволяють змінювати макет і розміри елементів залежно від роздільної здатності екрана. Наприклад, для адаптації стилів під мобільні пристрої використовуються такі медіазапити:

```
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

2. Flexbox та CSS Grid використовуються для створення гнучких і зручних макетів, що автоматично підлаштовуються під різні екранні розміри. Ось приклад використання Flexbox для мобільної версії:

```
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
}
```

```
@media (max-width: 768px) {
  .container {
     flex-direction: column;
  }
}
```

Інтерфейс активно взаємодіє з серверною частиною через API-запити. Для цього були використані інструменти для роботи з асинхронними запитами, такі як Axios, що дозволяє безпечно і ефективно здійснювати HTTP-запити до серверу. Наприклад:

```
import axios from 'axios';
const fetchData = async () => {
   try {
      const response = await axios.get('http://localhost:3000/services');
      setData(response.data);
   } catch (error) {
       console.error('Error fetching data', error);
   }
};
```

Дані, отримані від бекенду, проходять через відповідні компоненти для відображення користувачу. Всі запити здійснюються в асинхронному режимі, що дозволяє не блокувати інтерфейс під час очікування відповіді від сервера.

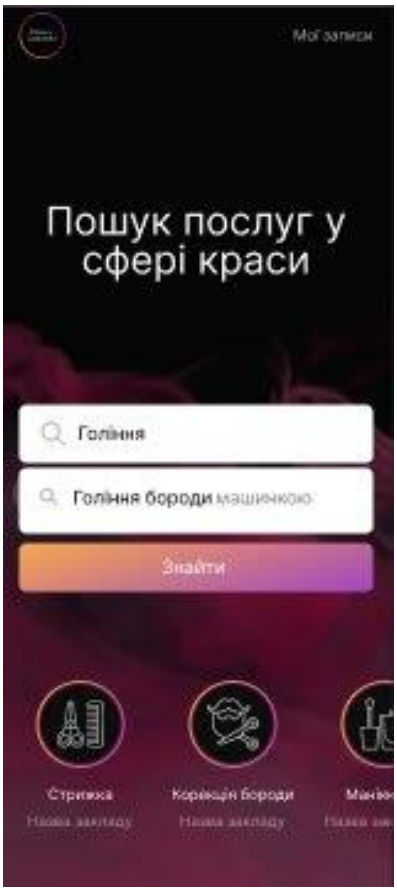


Рисунок 3.3 – Адаптивна верстка для мобільних пристроїв

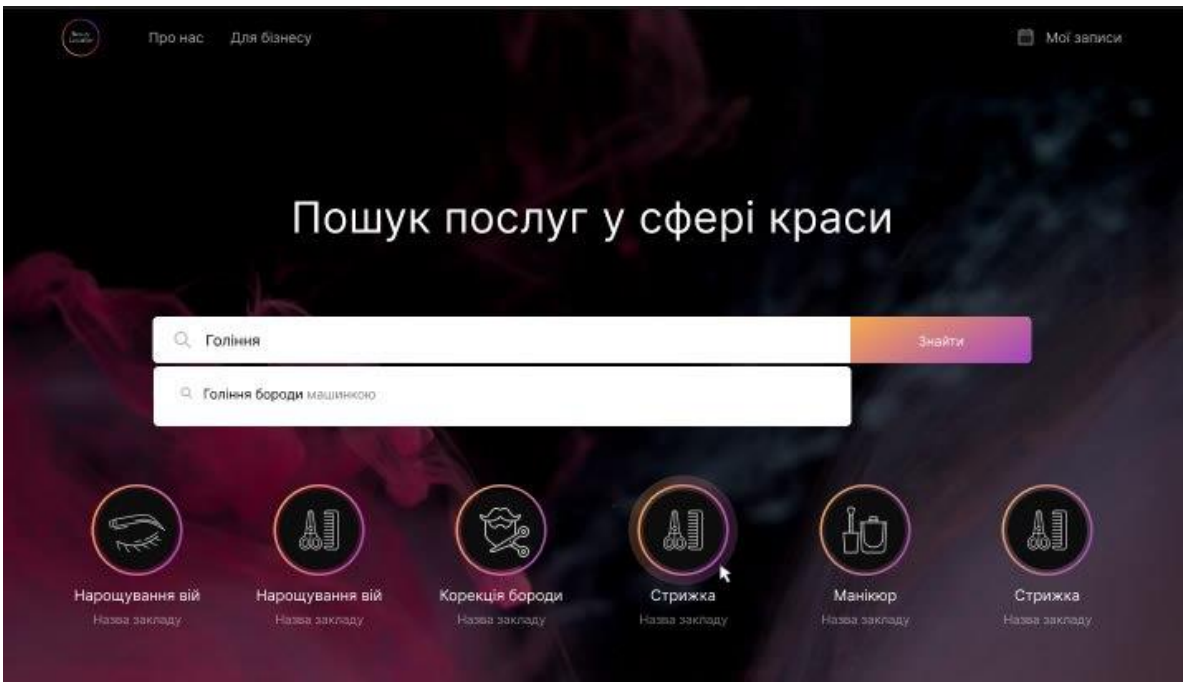


Рисунок 3.4 – Адаптивна верстка для десктопних пристроїв

Реалізація інтерфейсу користувача для даної системи базується на

використанні сучасних технологій, що дозволяють забезпечити високу продуктивність, зручність у використанні та адаптивність на різних пристроях. Основними інструментами, що використовуються для створення інтерфейсу, є Nuxt.js та Vuex. Nuxt.js, який побудований на основі Vue.js, дозволяє створювати універсальні додатки з підтримкою серверного рендерингу, що забезпечує швидке завантаження сторінок і покращує SEO. Це важливо для систем, які повинні бути зручними та доступними для широкої аудиторії користувачів.

Використання Vuex, в свою чергу, дозволяє ефективно управляти станом додатку. Vuex є централізованим сховищем для всього стану додатка, що робить управління даними зручнішим і передбачуванішим. З таким підходом, всі компоненти додатка мають доступ до загальних даних, що полегшує взаємодію між ними і спрощує масштабування системи в майбутньому.

Модульна архітектура, запропонована цими інструментами, сприяє гнучкості та зручності підтримки додатка. Кожен модуль відповідає за певну частину функціоналу, що дозволяє розвивати додаток без значних змін в основному коді. Це також полегшує тестування та оптимізацію.

Адаптивний дизайн забезпечує, щоб інтерфейс коректно відображався на різних типах пристроїв, будь то мобільні телефони, планшети або настільні комп’ютери. Це гарантує зручне та ефективне використання системи для всіх користувачів, незалежно від того, яке обладнання вони використовують.

Інтеграція з бекендом дозволяє створювати динамічний інтерфейс, що оновлюється в реальному часі, надаючи користувачам актуальну інформацію та можливість взаємодії з системою без необхідності перезавантаження сторінок. Це сприяє більш інтерактивному та зручному користувацькому досвіду..

3.4 Тестування додатку

Тестування додатка є критично важливою складовою частиною процесу розробки, оскільки від його якості залежить стабільність роботи програми, її

здатність виконувати всі визначені функціональні вимоги та забезпечувати задоволення потреб користувачів. У випадку додатка, орієнтованого на пошук і запис на послуги, тестування повинно бути комплексним і всебічним, покриваючи всі основні функціональні можливості системи, такі як пошук, вибір послуг, реєстрація користувачів, запис на послугу, а також обробка даних користувачів.

Одним з важливих аспектів тестування є перевірка системи на стійкість до можливих помилок і помилкових дій з боку користувачів. Це включає тестування на некоректний ввід даних, перевірку обробки помилок при неправильному форматі запитів, перевірку системи на наявність вразливостей, а також тестування на навантаження для визначення того, як система реагує на великий обсяг запитів і користувачів одночасно.

Тестування повинно здійснюватися на різних етапах розробки з метою виявлення можливих помилок на ранніх стадіях, що дозволить знизити витрати часу та ресурсів на виправлення проблем в майбутньому. На початкових етапах важливо виконувати юніт-тести, які перевіряють окремі функціональні блоки та методи, а також інтеграційні тести, що оцінюють взаємодію різних компонентів системи. На більш пізніх етапах важливими є енд-ту-енд тести, що забезпечують перевірку повного циклу взаємодії користувача з додатком, а також стрес-тести для визначення межі стійкості додатка при високих навантаженнях.

Завдяки таким тестам можна виявити та усунути можливі помилки ще до того, як продукт потрапить до кінцевих користувачів. Це дозволяє значно знизити ризик появи критичних помилок у продакшн-версії та забезпечити стабільну і безпечну роботу додатка в умовах реального використання.

Позитивні тестові кейси

Позитивні тестові кейси є основою для перевірки правильності роботи основних функцій додатка. Вони включають перевірку сценаріїв, у яких користувач взаємодіє з додатком правильно, і всі функціональні можливості виконуються відповідно до вимог.

1. Введення назви послуги
   - Тестовий кейс 1.1: Введення коректної назви послуги (Рис. 3.6)
     - Передумови: Користувач запустив додаток і знаходиться на екрані пошуку послуг.
     - Кроки:
       1. Ввести назву послуги, яка існує в базі даних.
       2. Натиснути кнопку пошуку.
     - Очікуваний результат: Пошук відображає послуги, що відповідають введеній назві.

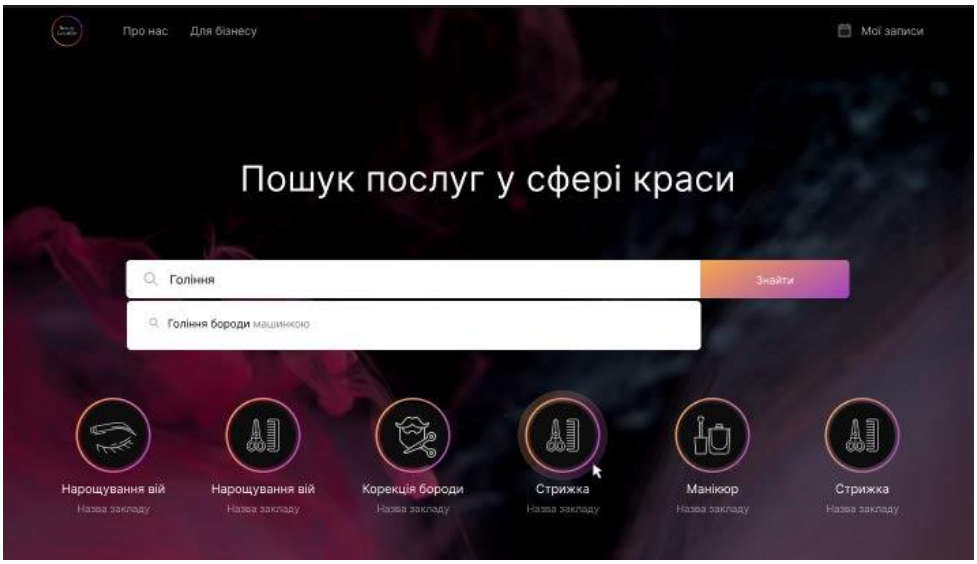


Рисунок 3.6 – Введення послуги у пошук

   - Тестовий кейс 1.2: Введення назви послуги з різними регістрами (Рис. 3.7)
     - Передумови: Користувач знаходиться на екрані пошуку послуг.
     - Кроки:
       1. Ввести назву послуги з різними регістрами літер (наприклад, "ГолІнНя" замість "Гоління").
       2. Натиснути кнопку пошуку.
     - Очікуваний результат: Пошук відображає послуги, що відповідають введеній назві.

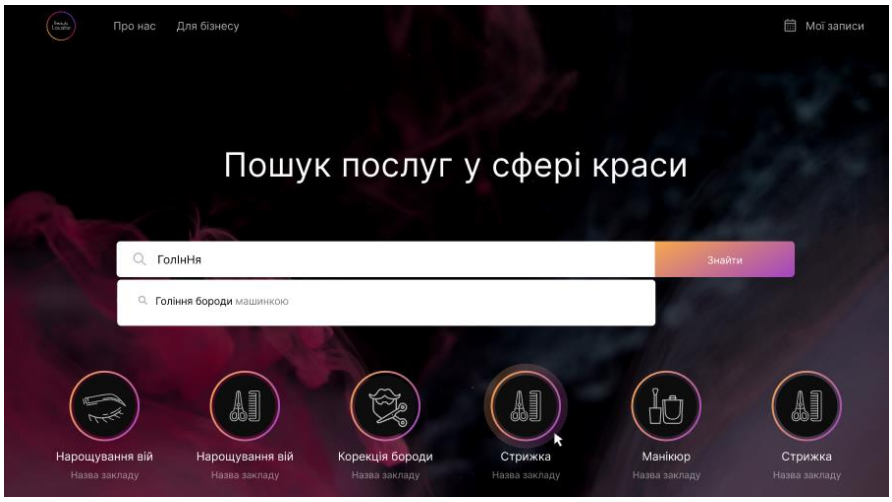


Рисунок 3.7 – Результати пошуку з різними регістрами

- Тестовий кейс 1.3: Введення назви послуги з пробілами на початку і в кінці (Рис. 3.8)
  - Передумови: Користувач знаходиться на екрані пошуку послуг.
  - Кроки:
    - Ввести назву послуги з різними регістрами літер (наприклад, " Геління" замість "Гоління").
    - Натиснути кнопку пошуку.
  - Очікуваний результат: Пошук правильно обробляє пробіли і відображає послугу.

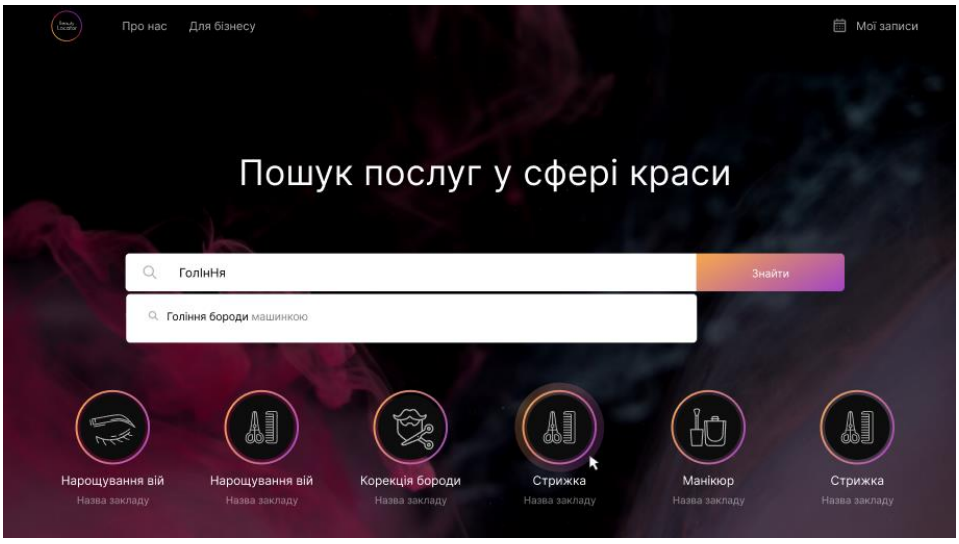


Рисунок 3.8 – Результати пошуку з пробілами

2. Геолокація
   - Тестовий кейс 2.1: Перевірка визначення геолокації користувача (Рис. 3.9)
     - Передумови: Користувач дозволив додатку використовувати геолокацію.
     - Кроки:
       1. Відкрити додаток і дозволити доступ до геолокації.
       2. Перевірити точність визначення геолокації користувача.
     - Очікуваний результат: Адреса підтягується коректна, що відповідає дійсному місцезнаходженню.

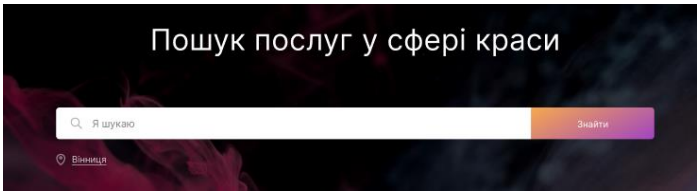


Рисунок 3.9 – Адреса отримана по GPS

   - Тестовий кейс 2.2: Перевірка відображення послуг у певному радіусі (Рис. 3.10)
     - Передумови: Геолокація активована.
     - Кроки:
       1. Відкрити додаток і дозволити доступ до геолокації.
       2. Встановити радіус пошуку послуг.
       3. Перевірити, чи відображаються послуги у заданому радіусі.
     - Очікуваний результат: Відображаються тільки ті послуги, які знаходяться в межах заданого радіусу.

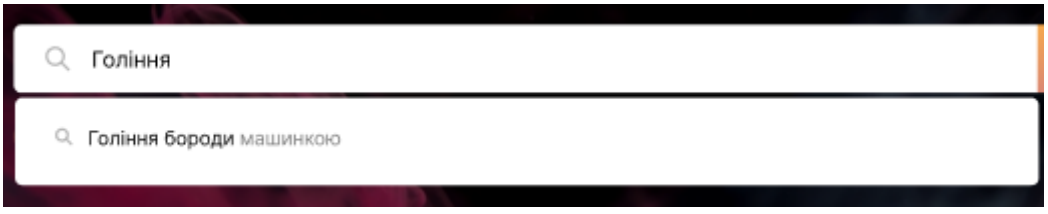


Рисунок 3.10 – Результат пошуку, що відповідає заданому радіусу GPS

3. Пошук послуг за категорією
   - Тестовий кейс 3.1: Пошук послуг за категорією (Рис. 3.11).
     - Передумови: Користувач знаходиться на екрані пошуку послуг.
     - Кроки:
       1. Вибрати категорію послуг (наприклад, "Гоління бороди").
       2. Натиснути кнопку пошуку.
     - Очікуваний результат: Виводяться послуги, які належать до вибраної категорії.

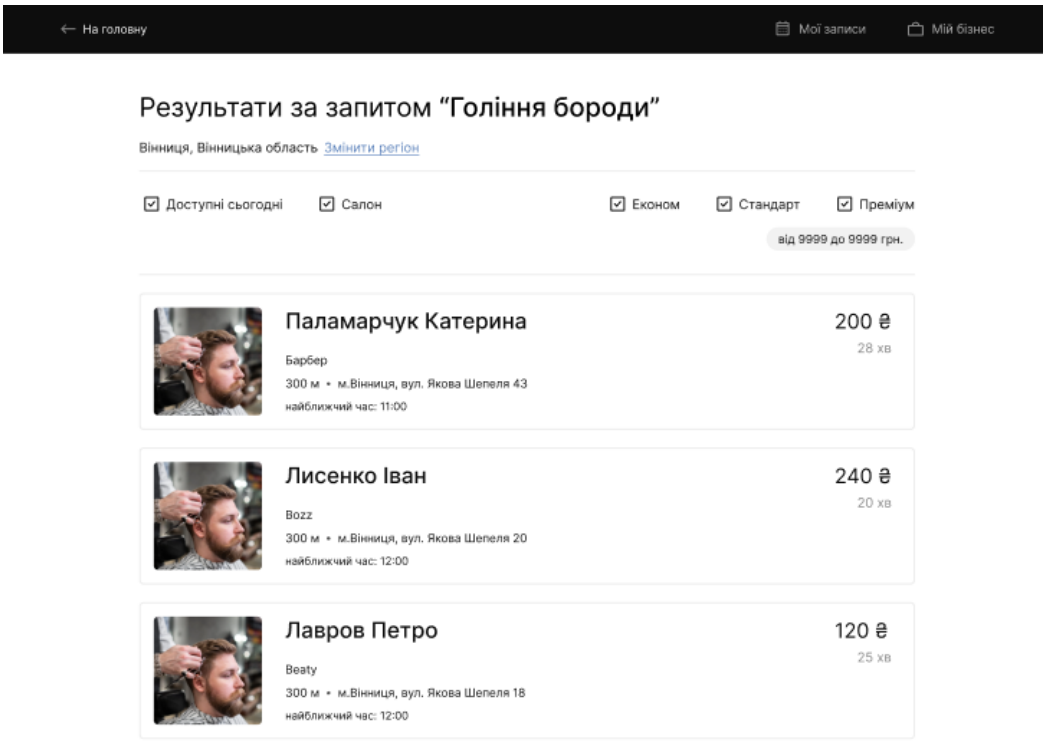


Рисунок 3.11 – Сторінку результатів запиту, після обрання послуги

4. Вибір послуги, дати та часу
   - Тестовий кейс 4.1: Вибір послуги (Рис. 3.13), дати та часу (Рис. 3.13).
     - Передумови: Користувач знаходиться на екрані запису.
     - Кроки:
       1. Вибрати послугу (наприклад, "Масаж").
       2. Вибрати дату (наприклад, 25 грудня).
       3. Вибрати час (наприклад, 15:00).

- Очікуваний результат: Відображається доступний час для обраної послуги на вибрану дату.

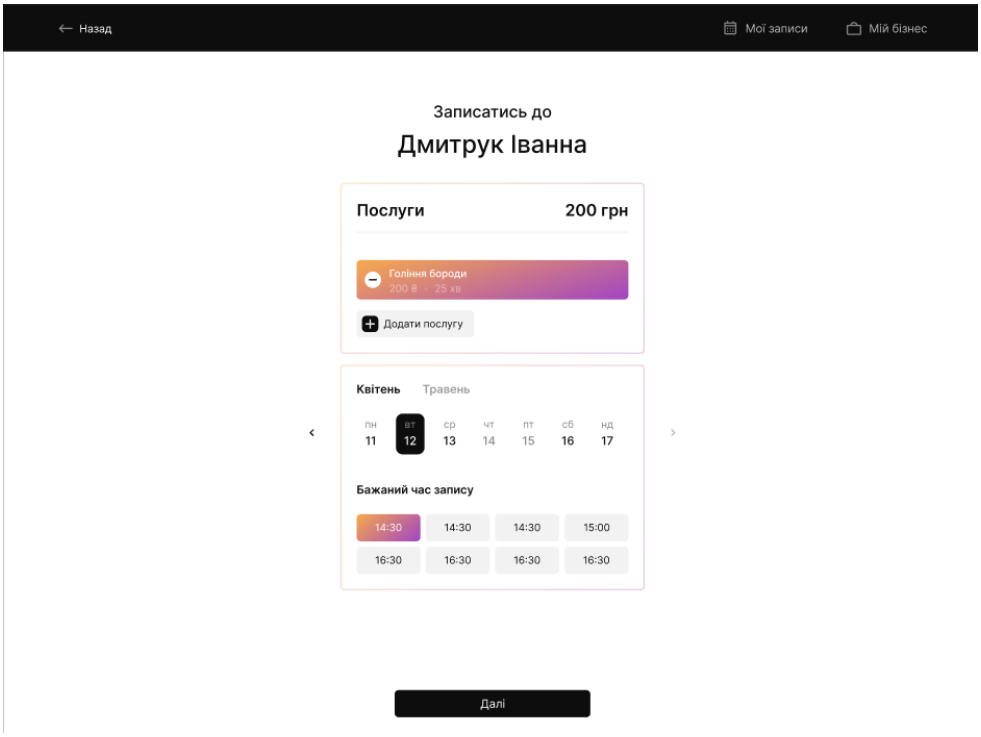


Рисунок 3.12 – Сторінка обраного запису

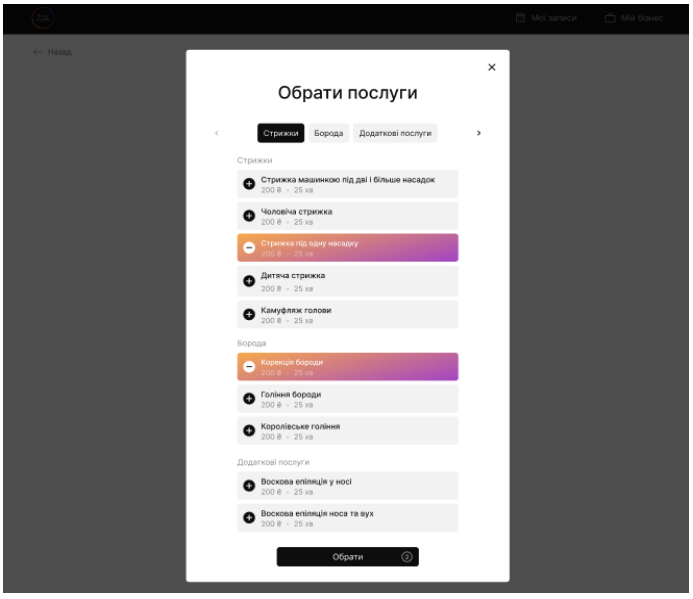


Рисунок 3.13 – Вікно доступних послуг даного фахівця

5. Вибір фахівця

- Тестовий кейс 5.1: Вибір фахівця (Рис. 3.14).
  - Передумови: Користувач знаходиться на екрані вибору фахівця.

- Кроки:
    1. Вибрати фахівця зі списку доступних (наприклад, "Іванов Іван").
- Очікуваний результат: Виводиться інформація про вибраного фахівця.

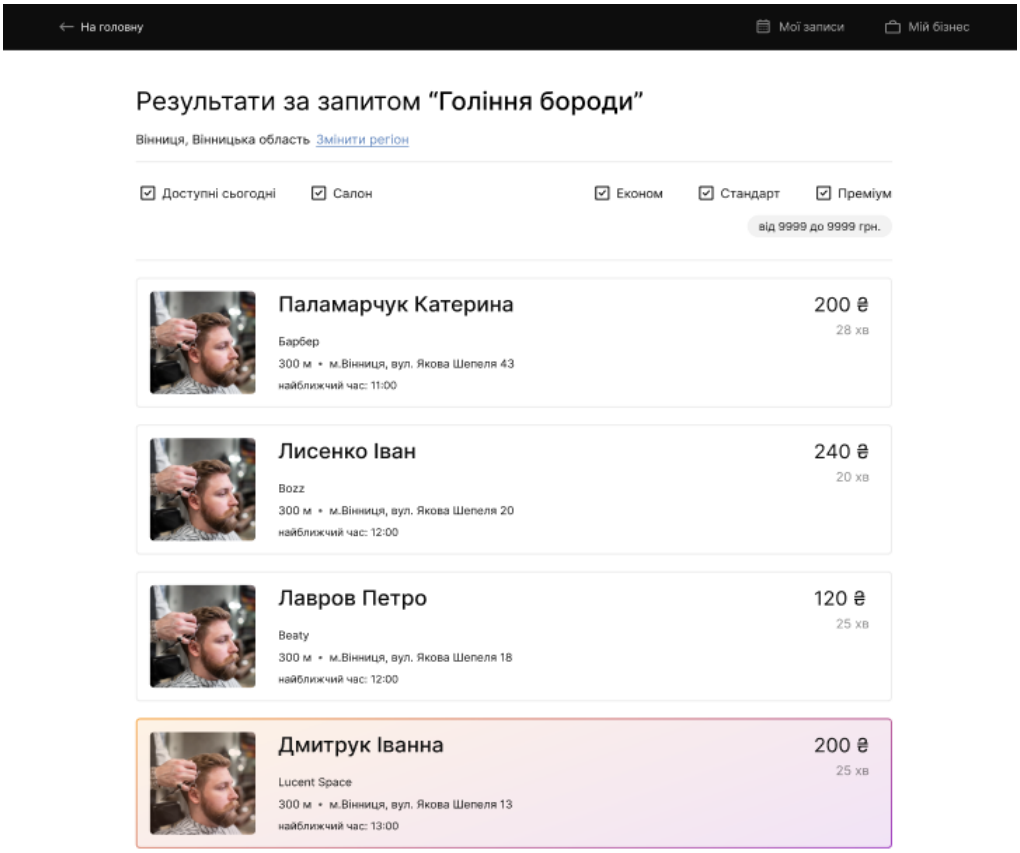


Рисунок 3.14 – Інформація про фахівців

6. Введення контактних даних
    - Тестовий кейс 6.1: Введення коректних контактних даних (Рис. 3.15) та результат створеного запису (Рис. 3.16) .
    - Передумови: Користувач знаходиться на екрані введення контактних даних.
    - Кроки:
        1. Ввести коректний номер телефону.
        2. Ввести коректну електронну адресу
    - Очікуваний результат: Дані зберігаються без помилок.

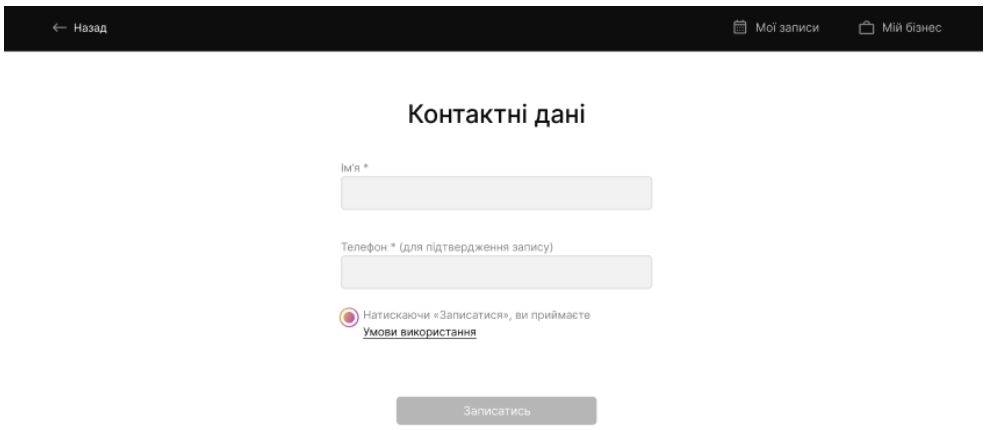


Рисунок 3.15 – Сторінка введення контактних даних

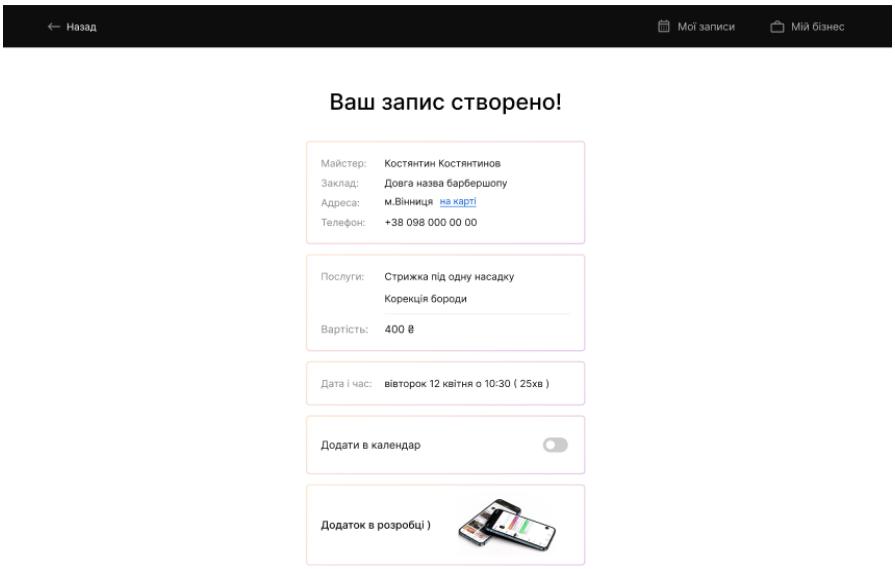


Рисунок 3.16 – Сторінка результату створеного запису

7. Підтвердження запису
   - Тестовий кейс 7.1: Підтвердження запису (Рис. 3.17).
     - Передумови: Користувач знаходиться на екрані підтвердження запису.
     - Кроки:

1. Підтвердити запис на вибрану послугу, фахівця, дату та час.

- Очікуваний результат: Запис підтверджено, і він відображається в особистому кабінеті користувача.

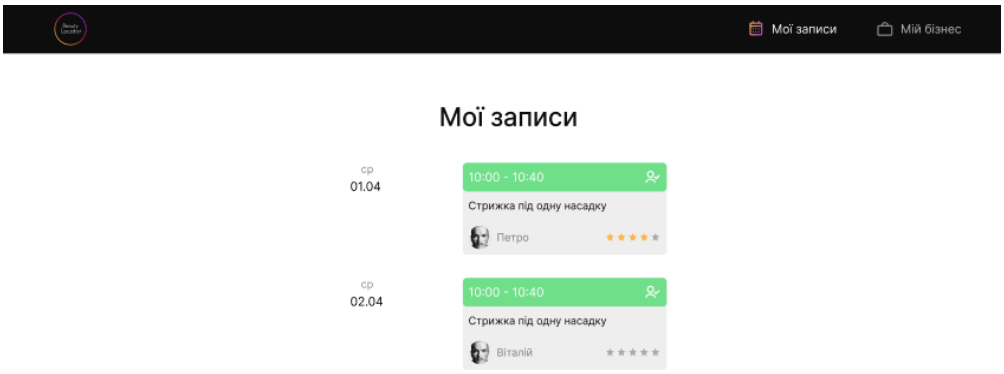


Рисунок 3.17 - Перелік записів користувача

Негативні тестові кейси

Негативні тестові кейси спрямовані на виявлення помилок, які можуть виникнути при некоректних або неочікуваних діях з боку користувача.

1. Введення назви послуги
   - Тестовий кейс 1.1: Введення порожнього поля (Рис. 3.18).
     - Передумови: Користувач знаходиться на екрані пошуку послуг.
     - Кроки:
       1. Залишити поле для введення назви послуги порожнім.
       2. Натиснути кнопку пошуку.
     - Очікуваний результат: Виводиться повідомлення про необхідність введення назви послуги.

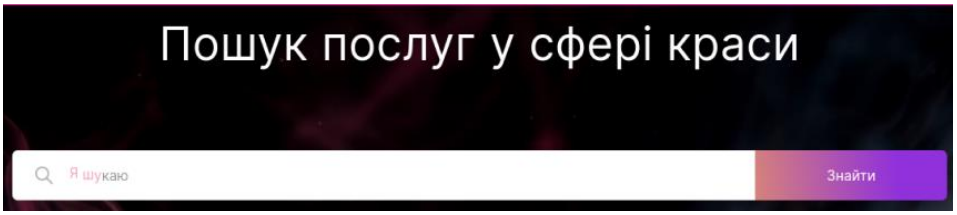


Рисунок 3.18 – Індикатор введення послуги

- Тестовий кейс 1.2: Введення неіснуючої назви послуги (Рис. 3.19).

- Передумови: Користувач знаходиться на екрані пошуку послуг.
- Кроки:
    1. Ввести неіснуючу назву послуги (наприклад, "Неіснуюча послуга").
    2. Натиснути кнопку пошуку.
- Очікуваний результат: Виводиться повідомлення про відсутність результатів пошуку.

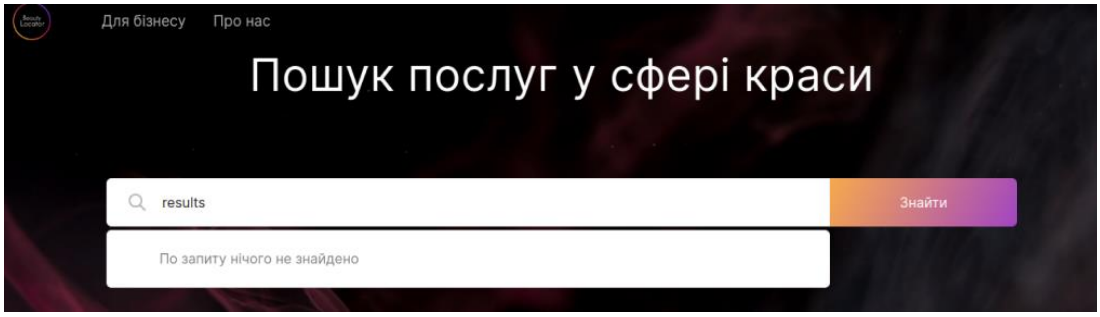


Рисунок 3.19 – Повідомлення, про відсутність результатів пошуку

2. Геолокація
    - Тестовий кейс 2.1: Відмова в доступі до геолокації (Рис. 3.20).
        - Передумови: Користувач відмовився від доступу до геолокації.
        - Кроки:
            1. Відкрити додаток і відмовитися від доступу до геолокації.
        - Очікуваний результат: Виводиться повідомлення про необхідність увімкнути геолокацію або ввести адрусу вручну для коректної роботи додатка.

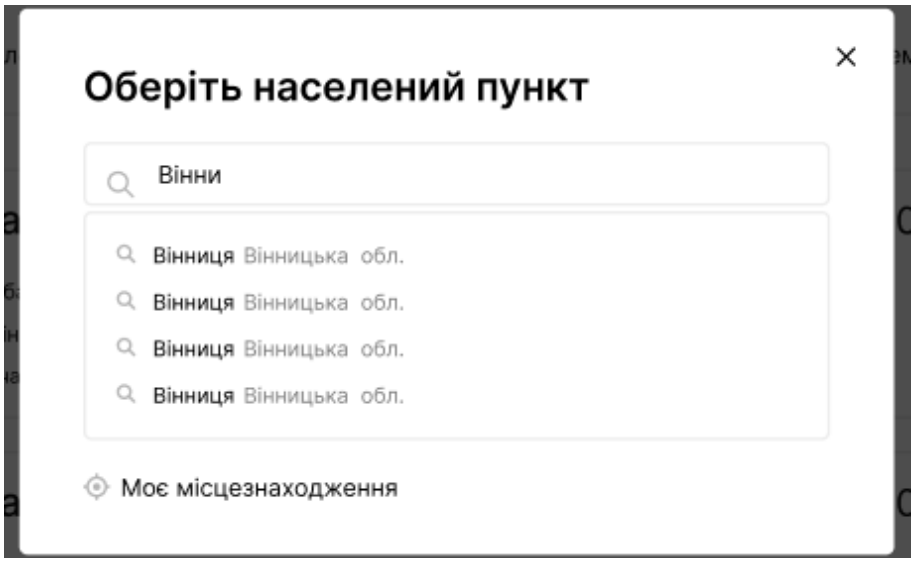


Рисунок 3.20 – Введення адреси в поле

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, оскільки воно забезпечує стабільність і надійність додатка, а також гарантує його відповідність вимогам користувачів. Процес тестування дозволяє виявити помилки на різних етапах розробки, що сприяє підвищенню якості продукту і знижує ризики виникнення критичних помилок після випуску. У рамках тестування використовуються різні типи тестових кейсів, що дозволяє комплексно оцінити функціональність додатка. Одним із ключових аспектів є включення позитивних тестів, які перевіряють правильність роботи основних функцій. Ці тести орієнтовані на забезпечення коректної роботи додатка в типових умовах, відповідно до вимог користувачів і сценаріїв використання.

Окрім цього, негативні тестові кейси мають велике значення, оскільки вони допомагають виявити потенційні вразливості системи. Вони фокусуються на перевірці того, як система реагує на некоректні дії з боку користувачів, такі як введення неправильної інформації, несанкціоновані запити або некоректні операції з даними. Негативні тести дозволяють оцінити стійкість і безпеку додатка, а також переконатися, що він здатний адекватно реагувати на непередбачувані ситуації без збоїв або втрати даних.

Процес тестування має проводитись на всіх етапах розробки, починаючи

з ранніх стадій, коли виявляються основні помилки в коді чи архітектурі, і закінчуючи етапами фінального тестування, коли перевіряється загальна працездатність і готовність додатка до використання. Таке поетапне тестування дозволяє максимально швидко виявляти і виправляти недоліки, знижуючи ризики виникнення серйозних проблем у майбутньому. Загалом, тестування додатка є важливим інструментом для забезпечення високої якості продукту та задоволення вимог кінцевих користувачів, що робить його незамінним етапом у розробці програмного забезпечення.

3.5 Інструкція користувачу програми

Програма для пошуку та бронювання послуг є зручним інструментом, який значно спрощує процес пошуку та організації запису на необхідні послуги. Користувачі можуть не лише знаходити потрібні послуги, але й отримувати детальну інформацію про доступних фахівців, переглядати їхні відгуки, спеціалізацію та доступні часові проміжки. Вибір зручного часу та дати для запису дає можливість планувати візит без необхідності дзвонити чи узгоджувати деталі з адміністрацією. Після вибору послуги і фахівця, користувач може ввести свої контактні дані, що дозволяє підтвердити бронювання і отримати нагадування про майбутній запис. Таким чином, програма не лише спрощує процес запису, але й робить його максимально зручним та ефективним, забезпечуючи високий рівень комфорту для користувача. Крім того, інтерфейс програми інтуїтивно зрозумілий, що дозволяє швидко освоїтись навіть тим, хто не має великого досвіду у користуванні подібними системами. Нижче наведено детальну інструкцію для ефективного використання програми:

1. Введення назви послуги
    - Крок 1: Введіть назву послуги, яку ви шукаєте, в поле для пошуку. Крок 2: Після введення оберіть одну із запропонованого списку та натисніть кнопку «Пошук», щоб перейти до

наступного етапу

2. Доступ до GPS
    - Крок 3: Програма запитає доступ до GPS, щоб визначити ваше місцезнаходження та автоматично вибрати ваш регіон.
        - Після введення назви послуги програма перевіряє, чи є доступ до GPS-даних на пристрої користувача. Якщо доступ є, користувач може скористатися GPS для автоматичного визначення його місцезнаходження та пошуку послуг у його регіоні.
        - Якщо користувач не має доступу до GPS або хоче вручну вибрати місцезнаходження, він може вибрати регіон із запропонованого списку. Це важливий крок, оскільки він дозволяє обмежити пошук послуг тільки тими, що доступні в конкретному регіоні.
3. Пошук послуг за назвою та регіоном.
    - Крок 5: Після вибору регіону або визначення місцезнаходження через GPS програма здійснює пошук доступних послуг, які відповідають введеній назві та обраному регіону. Пошук відображає лише ті послуги, які можна забронювати у вказаному місці.
4. Перегляд доступних послуг
    - Крок 6: Після успішного пошуку програма відображає всі доступні послуги, з якими можна здійснити запис. Користувач може переглядати список послуг, оцінювати їх і вибирати ту, яка відповідає його потребам.
5. Вибір дати та часу
    - Крок 7: Після вибору послуги користувач повинен вибрати зручний час для її надання. Програма відображає доступні варіанти дати та часу для вибраної послуги. Користувач має можливість вибрати один із запропонованих варіантів або

повернутися до попереднього етапу.

6. Вибір фахівця
    - Крок 8: Після вибору часу та дати виберіть фахівця, який надаватиме послугу. Програма покаже доступних фахівців відповідно до обраної послуги та часу.
    - Якщо фахівець доступний, ви зможете перейти до введення контактних даних.
7. Введення контактних даних
    - Крок 9: Введіть свої контактні дані, включаючи ім’я, телефон та електронну пошту.
    - Ці дані необхідні для створення запису та підтвердження вашої реєстрації.
8. Створення та збереження запису.
    - Крок 10: Після введення контактних даних натисніть кнопку "Зберегти" для створення запису. Програма автоматично збереже ваш запис у системі.
9. Перегляд записів
    - Крок 11: Ви можете переглянути всі ваші записи, натиснувши кнопку "Переглянути записи". Це дозволяє вам перевірити статус ваших бронювань.
    - Вибір цього варіанту перенесе вас до списку усіх активних записів.

Важливі зауваження:

- Доступ до GPS: Якщо ви відмовитеся від надання доступу до GPS, вам буде запропоновано вручну вказати ваш регіон, що може зайняти більше часу.
- Заповнення полів: Переконайтесь, що всі обов’язкові поля заповнені правильно перед підтвердженням запису.
- Перегляд записів: Ви можете переглядати та скасовувати ваші записи через відповідний розділ, якщо вам потрібно змінити дату, час або

фахівця.

- Зберігання записів: Усі записи зберігаються в системі та доступні для перегляду в будь-який час через розділ "Перегляд записів".

Програма для бронювання послуг розроблена таким чином, щоб забезпечити максимальний комфорт і зручність для користувачів, однак для її ефективного використання необхідно дотримуватися кількох простих правил. По-перше, користувачам слід уважно заповнювати всі обов’язкові поля, оскільки це безпосередньо впливає на успішне завершення процесу бронювання. Наприклад, якщо контактні дані не будуть введені коректно, це може призвести до помилки при підтвердженні запису, і програма не зможе зберегти цю інформацію. Крім того, програма має можливість зберігати дані та дозволяє редагувати існуючі записи, що дає користувачам можливість вносити зміни в процесі бронювання без необхідності починати все знову. Однак, якщо користувач відмовляється від надання доступу до GPS, програма запропонує вручну вибрати регіон, що може зайняти більше часу, оскільки доведеться шукати відповідне місце на карті.

Особливістю програми є її інтерактивність і висока адаптивність до потреб користувачів, що дозволяє швидко і без проблем змінювати обраний час, спеціаліста або навіть саму послугу. Така гнучкість є важливою перевагою для користувачів, оскільки вони можуть змінювати свої плани в будь-який момент, якщо виникає необхідність перенести або скасувати запис. Крім того, система має механізм відновлення даних, що дозволяє зберегти попередні записи в разі помилкового скасування або зміни, і користувачі можуть повернути їх за допомогою кількох кліків. Всі зміни відбуваються в реальному часі, що гарантує актуальність даних і дозволяє уникнути ситуацій, коли користувачі стикаються з застарілою або некоректною інформацією. Цей підхід забезпечує постійну синхронізацію даних, що робить систему більш надійною. Завдяки такій гнучкості програма здатна задовольняти різноманітні потреби користувачів, будь то зміна часу або вибір іншого спеціаліста, а також дозволяє безперешкодно адаптуватися до змін у планах або обставинах. В кінцевому

рахунку, це підвищує зручність і ефективність роботи користувачів, що є важливим аспектом при використанні подібних сервісів.

3.6 Висновок

У процесі розробки автоматизованої системи для пошуку та бронювання послуг були виконані важливі етапи, що включали обґрунтування вибору мов програмування, технологій і середовищ, створення програмних компонентів та проведення тестування. Для реалізації були обрані такі мови та технології, як JavaScript та фреймворк Nuxt.js, які забезпечують високу сумісність, можливість розробки динамічної та інтерактивної функціональності, багатство доступних ресурсів і зручний синтаксис.

У процесі розробки програмного забезпечення було детально розглянуто структуру системи, що дозволяє користувачам обирати потрібні послуги, спеціалістів, дату і час, а також вводити контактні дані для бронювання. Інтерфейс розроблений таким чином, щоб максимально спростити взаємодію користувача із системою, забезпечуючи інтуїтивно зрозумілі можливості для введення і підтвердження замовлення.

Проведене тестування дозволило впевнитися у стабільності та надійності функціональних можливостей системи, перевірити коректність виконання основних функцій та належне відображення інформації для користувача. У результаті тестування були виявлені та виправлені можливі помилки, що забезпечило високу якість роботи сервісу та безперебійне функціонування.

Завдяки виконаним етапам було створено зручну, функціональну та інтуїтивно зрозумілу автоматизовану систему для пошуку та бронювання послуг, яка повністю відповідає потребам користувачів.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Розробка автоматизованої системи для пошуку та бронювання послуг» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Розрахунок узагальненого коефіцієнта якості розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка автоматизованої системи для пошуку та бронювання послуг» є оцінювання науково-технічного рівня та рівня комерційного потенціалу

розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [70].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		

1. Технічна здійсненність концепції	5	5	4
2. Ринкові переваги (наявність аналогів)	4	3	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	4	5
5. Ринкові переваги (експлуатаційні витрати)	3	3	4
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	38	40	40
Середньоарифметична сума балів $СБ_с$	39,3		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [70].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно з результатами досліджень, рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи для пошуку та бронювання послуг» становить 39,3 бали. Це значення, відповідно до таблиці 4.3, вказує на високу комерційну значущість проведених досліджень. Такий рівень комерційного потенціалу перевищує середній, що свідчить про високий попит на подібні рішення в даній сфері. Це також підтверджує важливість розробки та впровадження системи для покращення ефективності бізнес-процесів. Таким чином, проект має значний потенціал для успішної реалізації і

комерціалізації.

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [71]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{ai}} \,, \tag{4.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *i*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}} \,; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість параметрів, що обробляються в запиті	од.	12	40	3,33	0,15
Швидкість обробки запиту ресурсом	с	0,5	0,06	8,33	0,2
Рівень інтуїтивної доступності інтерфейсу	бал	6	9	1,5	0,25
Обсяг баз даних, що підлягають обробці	Мб	100	400	4	0,25
Формування та оновлення баз даних	бал	4	8	2	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 3{,}33 \cdot 0{,}15 + 8{,}33 \cdot 0{,}2 + 1{,}5 \cdot 0{,}25 + 4 \cdot 0{,}25 + 2 \cdot 0{,}15 = 3{,}84.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,84 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему

«Розробка автоматизованої системи для пошуку та бронювання послуг», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [70]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17500{,}00 \cdot 21 / 21 = 17500{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17500,00	833,33	21	17500,00
Інженер-програміст	20200,00	961,90	8	7695,24
Інженер-проектувальник	16400,00	780,95	15	11714,29

автоматизованих систем				
Консультант-реєстратор (адміністратор)	15600,00	742,86	4	2971,43
Всього				39880,95

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Розробка автоматизованої системи для пошуку та бронювання послуг» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М \cdot К_i \cdot К_с}{Т_р \cdot t_{зм}} , \quad (4.6)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн;

$К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [70];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 8000,00 · 1,10 · 1,15 / (21 · 8) = 60,24 *грн.*
$З_{р1}$ = 60,24 · 8,00 = 481,90 *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка офісного обладнання	8,00	2	1,10	60,24	481,90
Підготовка робочого місця розробника автоматизованих систем	4,00	2	1,10	60,24	240,95
Інсталяція програмного забезпечення розробки системи	5,00	5	1,70	93,10	465,48
Формування бази даних	10,00	4	1,50	82,14	821,43
Тренування інформаційного ресурсу	3,50	3	1,35	73,93	258,75
Налагодження програмних блоків	4,50	5	1,70	93,10	418,93
Тестування автоматизованої системи	7,00	4	1,50	82,14	575,00
Всього					3262,44

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (39880,95 + 3262,44) · 11 / 100% = 4745,77 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (39880,95 + 3262,44 + 4745,77) · 22 / 100% = 10535,62 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Розробка автоматизованої системи для пошуку та бронювання послуг».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1$ = 3 · 196,00 · 1,03 - 0 · 0 = 605,64 грн.

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн

Папір канцелярський офісний	196,00	3	0	0	605,64
Папір креслярський	25,00	10	0	0	257,50
Начиння канцелярське	185,00	3	0	0	571,65
Органайзер офісний	201,00	3	0	0	621,09
Картридж для графічного принтера	658,00	2	0	0	1355,48
Картридж для принтера	1058,00	1	0	0	1089,74
Диск оптичний	23,00	4	0	0	94,76
FLASH-пам’ять	172,00	1	0	0	177,16
Інше	150,00	1,000	0	0	154,50
Всього					4927,52

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка автоматизованої системи для пошуку та бронювання послуг» відсутні.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спеціалізованого устаткування, яке є необхідним для проведення досліджень та експериментів. Ці витрати включають також витрати на проектування, виготовлення, транспортування, монтаж та встановлення спецустаткування, яке повинно відповідати вимогам конкретних наукових або виробничих завдань. Важливо, що ці витрати охоплюють не тільки придбання устаткування, а й забезпечення його належного функціонування в процесі досліджень, що включає

налаштування та тестування. На даний момент витрати за цією статтею відсутні, що може свідчити про відсутність необхідності в додатковому спецустаткуванні для виконання поточних наукових або експериментальних робіт.

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *7264,00 · 1 · 1,02 = 7409,28 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище математичного інженерного моделювання Matchcad 12	1	7264,00	7409,28
Абонплата доступу до мережі Internet	1	350,00	357,00

Всього	7766,28

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації, який передбачає рівномірне списання вартості активу протягом його корисного терміну служби. Це дозволяє спростити розрахунки та забезпечити стабільні витрати на амортизацію протягом усього періоду використання активу. Для цього застосовуємо наступну формулу:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (37899{,}00 \cdot 1) / (3 \cdot 12) = 1052{,}75$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника програмного забезпечення	37899,00	3	1	1052,75
Персональний комп'ютер інженера-	32599,00	3	1	905,53

розробника інформаційної системи				
Оргтехніка	7770,00	4	1	161,88
Приміщення дослідної лабораторії	450000,00	30	1	1250,00
ОС Windows 10	6820,00	3	1	189,44
Прикладний пакет Microsoft Office 2016	6630,00	3	1	184,17
Всього				3743,76

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 10,98$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,25 \cdot 160,0 \cdot 10,98 \cdot 0,95 / 0,97 = 439,20$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника програмного забезпечення	0,25	160,0	439,20
Персональний комп'ютер інженера-розробника інформаційної системи	0,08	160,0	140,54

Оргтехніка	0,25	2,5	6,86
Пристрої передачі даних	0,02	160,0	35,14
Всього			621,74

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Розробка автоматизованої системи для пошуку та бронювання послуг» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» відсутні.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.13)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (39880,95 + 3262,44) · 30 / 100% = 12943,02 грн.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не були враховані в інших

статтях витрат підприємства, але які все ж мають пряме відношення до процесу виробництва або надання послуг. Ці витрати можуть включати, наприклад, витрати на оренду спеціалізованого обладнання або витрати на специфічні матеріали, використані для виконання досліджень. Важливо, що ці витрати можуть бути безпосередньо віднесені на собівартість досліджень за прямими ознаками, якщо вони мають чіткий зв'язок з виробничим процесом або конкретними проектами. Тому підприємство повинно ретельно визначати та документувати ці витрати для коректного обліку та розподілу їх по собівартості.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (39880,95 + 3262,44) · 50 / 100% = 21571,70 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать витрати, пов’язані з управлінням організацією, які включають витрати на планування, координацію та контроль виробничих процесів. Сюди також входять витрати на винахідництво та раціоналізацію, що забезпечують удосконалення виробничих процесів та підвищення їх ефективності. Важливим елементом накладних витрат є витрати на підготовку та перепідготовку кадрів, а також їх навчання для забезпечення високої кваліфікації персоналу. Також до накладних витрат можна віднести витрати, пов’язані з набором робочої сили, включаючи рекламу вакансій, проведення співбесід та підготовку документів. Крім того, витрати на оплату послуг банків, необхідні для здійснення фінансових операцій та ведення рахунків, також вважаються накладними. Витрати на освоєння виробництва

продукції, науково-технічну інформацію та рекламу сприяють розвитку підприємства та його популяризації на ринку.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = (39880,95 + 3262,44) · 100 / 100% = 43143,39 грн.

Витрати на проведення науково-дослідної роботи на тему «Розробка автоматизованої системи для пошуку та бронювання послуг» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.16)$$

$B_{заг}$= 39880,95 +3262,44 +4745,77 +10535,62 +4927,52 +0,00 +0,00 + 7766,28 + 3743,76 +621,74 +0,00 +12943,02 +21571,70 +43143,39 = 153142,20 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.17)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 153142,20 / 0,95 = 161202,31 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів

тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Розробка автоматизованої системи для пошуку та бронювання послуг» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	6000	8000	8000	7000

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 250000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 50,00 грн;

$\pm\Delta Ц_o$ – зміна вартості послуги від впровадження результатів, приймемо 12,03 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [56]**:**

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.18)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а

коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (12,03 \cdot 250000,00 + 62,03 \cdot 6000) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 919735,62$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (12,03 \cdot 250000,00 + 62,03 \cdot 14000) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1054821,10$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (12,03 \cdot 250000,00 + 62,03 \cdot 22000) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1189906,59$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (12,03 \cdot 250000,00 + 62,03 \cdot 29000) \cdot 0,83 \cdot 0,4 \cdot (1-0,18/100\%) = 1308106,39$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.19)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,15$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових

чистих прибутків у цьому році.

$ПП$ =919735,62/(1+0,15)[1]+1054821,10/(1+0,15)[2]+1189906,59/(1+0,15)[3]+ +1308106,39/(1+0,15)[4]=799770,10+797596,30+782382,90+747914,08=3127663,37 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.20}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 161202,31 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 161202,31 = 322404,62 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.21}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 3127663,37 грн;

PV – теперішня вартість початкових інвестицій, 322404,62 грн.

$E_{абс} = ПП - PV$ = 3127663,37 - 322404,62 = 2805258,75 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.22}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 2805258,75 грн;

PV – теперішня вартість початкових інвестицій, 322404,62 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+2805258,75/322404,62)^{1/4} = 0,76.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.23}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,76 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка автоматизованої системи для пошуку та бронювання послуг» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.24}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,76 = 1,31 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи для пошуку та бронювання послуг» становить 39,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,84 рази.

Також термін окупності становить 1,31 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка автоматизованої системи для пошуку та бронювання послуг».

ВИСНОВКИ

У магістерській кваліфікаційній роботі була розроблена система для автоматизованого пошуку та бронювання послуг, яка відповідає сучасним вимогам щодо швидкості, зручності та точності обробки запитів користувачів. Задача розробки структури автоматизованої системи була успішно вирішена, оскільки система забезпечує ефективну взаємодію користувача з платформою та надає можливість автоматизованої обробки запитів.

У процесі роботи було розроблено алгоритми для аналізу та обробки даних про послуги, що дозволяє значно підвищити точність і швидкість обробки запитів. Це дозволяє користувачам отримувати найбільш актуальні та точні пропозиції відповідно до їхніх потреб. Алгоритми також забезпечують персоналізацію пропозицій на основі попередніх виборів користувачів, що значно покращує користувацький досвід і ефективність системи.

Однією з важливих задач була оптимізація процесу бронювання, щоб зменшити час обробки запитів. В результаті впровадження нових алгоритмів обробки даних було досягнуто значного скорочення часу на прийняття рішень, що підвищує ефективність роботи системи і зручність для користувачів.

Розробка механізмів рекомендацій на основі вподобань користувачів дозволила створити інтелектуальну систему, що забезпечує індивідуальний підхід і підвищує рівень задоволеності клієнтів, що підтверджується результатами тестування ефективності системи.

Користувацький інтерфейс був розроблений для забезпечення зручності використання, що важливо для впровадження системи на ринок. Тестування підтвердило високу надійність та функціональність програмного забезпечення, що свідчить про доцільність його комерційного використання.

Таким чином, задачі, сформульовані у вступі, були успішно вирішені, а розроблена система демонструє високий потенціал для покращення ефективності пошуку та бронювання послуг і є перспективною для подальшого розвитку та комерційного впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Польгуль Б. В. Розробка автоматизованого веб-сервісу з надання перукарських послуг. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20931
2. Мельниченко С. Автоматизовані системи бронювання в туризмі. [Електронний ресурс]. URL: https://tourlib.net/statti_ukr/melnychenko7.htm
3. Antonina Kamenchuk. Огляд програм для онлайн-бронювання. [Електронний ресурс]. URL: https://easyweek.com.ua/specifika-rinku-sistem-onlajn-zapisu.html
4. Гудзовата О. Автоматизовані системи управління готелями. [Електронний ресурс]. URL: https://tourlib.net/statti_ukr/gudzovata.htm
5. Ящук В. Аналіз сучасного стану інформаційних технологій та систем в індустрії туризму. [Електронний ресурс]. URL: https://tourlib.net/statti_ukr/yaschuk4.htm
6. Огляд систем онлайн-бронювання: виклики та перспективи. [Електронний ресурс]. URL: https://stfalcon.com/uk/blog/post/online-booking-system
7. Анастасія. Система онлайн-бронювання: чому вона необхідна для туристичного та готельного бізнесу. [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/pii/S0278431918304212
8. Гудзовата О. АВТОМАТИЗОВАНІ СИСТЕМИ УПРАВЛІННЯ ГОТЕЛЯМИ. [Електронний ресурс]. URL: http://journals-lute.lviv.ua/journal/15_2013/22.pdf
9. Гафіяк А. Проблеми створення автоматизованої інформаційної системи управління персоналом. [Електронний ресурс]. URL: https://economyandsociety.in.ua/journals/13_ukr/244.pdf
10. Кравчук М. Г. Інформаційні системи в менеджменті туризму: теорія та практика. М: Видавництво ЛНУ, 2021.
11. Singh Y. R. M. S. S. R. and Srivastava A. B. K. E-Business and E-Commerce Management. 4th ed. Pearson Education Limited, 2019.
12. Amie Parnaby. Online Booking Systems: The Pros and Cons from the Customer’s Perspective. [Електронний ресурс]. URL:

https://news.simplybook.me/online-booking-systems-the-pros-and-cons-from-the-customers-perspective

13. Бабій С. М. Впровадження автоматизованих систем бронювання в Україні. [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/abs/pii/S2212567119300534
14. Nishad Bhan. 7 Challenges of Online Booking System. [Електронний ресурс]. URL: https://webkul.com/blog/7-challenges-of-online-booking-system/
15. Шевчук А. Ю. “Еволюція систем бронювання: від традиційних до автоматизованих” in Інформаційні технології в туризмі: проблеми та рішення. Національний університет харчових технологій, 2020.
16. Сидоренко А. М. “Автоматизація системи бронювання в індустрії туризму,” in Індивідуальне проектування в інформаційних системах. Київський національний університет технологій та дизайну, 2019.
17. IEEE Guide to Software Requirements Specifications. [Електронний ресурс]. URL: https://chumpolm.wordpress.com/wp-content/uploads/2018/09/ieee-std-830-1984.pdf
18. Pressman R. S. Software Engineering: A Practitioner's Approach. 9th ed. McGraw-Hill Education, 2019.
19. Грицюк. Ю. І, Нємова. О. А. Методологія формулювання вимог до програмних систем. [Електронний ресурс]. URL: https://www.researchgate.net/publication/328142794_Osoblivosti_formuluvanna_vimog_do_programnogo_zabezpecenna
20. Sommerville I. Software Engineering. 10th ed. Pearson, 2016.
21. Smith. F. Agile Requirements: How to Write User Stories. [Електронний ресурс]. URL: https://www.agilealliance.org/agile101/user-stories/
22. Finkelstein A. and Finkelstein A. A. "Requirements Engineering: A Roadmap." In: Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 1-10.
23. Chisholm R. Service Design: From Insight to Implementation. New York: O'Reilly Media, 2016.

24. Magal S. R. and Word J. Integrated Business Processes with ERP Systems. New Jersey: Wiley, 2011.
25. Aguirre-Urreta M. I. and Hu J. Service Operations Management: The Importance of Service Process Modelling. International Journal of Operations & Production Management, vol. 37, no. 5, pp. 606-630, 2017.
26. McCulloch S. J., and B. J. G. A. "Modelling Service Processes: An Integrated Approach." Journal of Service Management, vol. 25, no. 4, pp. 490-511, 2014.
27. Моделювання процесів у системах бронювання. [Електронний ресурс]. URL: https://www.researchgate.net/publication/271706967_Modeling_Booking_Processes_in_Service_Industries
28. Пістунов І. М. Моделювання бізнес-процесів. [Електронний ресурс]. URL: http://pistunovi.inf.ua/MOD_BIZ_IIPOU.pdf
29. Крилик. О. П. Оптимізація на основі моделювання. [Електронний ресурс]. URL: https://www.visual-paradigm.com/guide/business-process-modeling/what-is-business-process-modeling/
30. Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C. Introduction to Algorithms. 3rd ed. Cambridge: MIT Press, 2009.
31. Sedgewick R. and Wayne K. Algorithms. 4th ed. Boston: Addison-Wesley, 2011.
32. Robert S. W. Data Structures and Algorithm Analysis in C++. 4th ed. Upper Saddle River: Pearson, 2011.
33. Winston W. L. Operations Research: Applications and Algorithms. 4th ed. Boston: Cengage Learning, 2004.
34. VisuAlgo. Algorithms Visualizer. [Електронний ресурс]. URL: https://visualgo.net/en/sorting
35. Fruage K. GeeksforGeeks: Sorting Algorithms. [Електронний ресурс]. URL: https://www.geeksforgeeks.org/sorting-algorithms/
36. Know Thy Complexities. Big O Cheat Sheet. [Електронний ресурс]. URL: https://www.bigocheatsheet.com
37. C Tutorial. Introduction to Sorting Algorithms. [Електронний ресурс]. URL:

https://www.tutorialspoint.com/data_structures_algorithms/sorting_algorithms.htm

38. Algorithm Articles. Optimizing Sorting Algorithms for Large Data Sets. [Електронний ресурс]. URL: https://blog.algorithmexamples.com/sorting-algorithm/why-are-certain-sorting-algorithms-ideal-for-large-datasets/
39. Maha Saadeh. Performance Evaluation of Parallel Sorting Algorithms on IMAN1 Supercomputer. [Електронний ресурс]. URL https://d1wqtxts1xzle7.cloudfront.net/71522987/ijast.2016.95-libre.pdf?1635320374=&response-content-disposition=inline%3B+filename%3DPerformance_Evaluation_of_Parallel_Sorti.pdf&Expires=1732755442&Signature=ZVO4GoaNL242xv63dpEH5ej7nMVLxpu0p0ZY3yB3plC8pohjKIGG8y3GohceQbY20SkGq9~UKCMOQu1Qs9eiz1O5FYgtSUnBwZzeCngjMZLiqWv7u93g7ZZwDzc~sR1K~TwoN76x2CfagbHe2whEkgkI7mS9nh4hsRtpyppSGi4BaPb8DPS~USGzU~cOmsAeEe-jOkVSwtvB2ChB4bUoi2oLEChGRzYmb~fB09yTMG4TbZl9o0OEx-mB1sA1wzV9d7SmAWNBWT4ZX5xx3dMeWKESyQtsrkX3Z4ECl68ij7byBGFB5VJYeyPrjGtbEfurVJER0K9eNf4H1rNvGzSEVw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
40. Christopher C. B. The Importance of Sorting Algorithms in Automation. [Електронний ресурс]. URL: https://www.intechopen.com/chapters/67932
41. Allain. A. Sorting Algorithm Comparison. [Електронний ресурс]. URL: https://www.cprogramming.com/tutorial/computersciencetheory/sortcomp.html
42. Bykov, M. M., Kuzmin, I. V., & Yakovenko, A. I. (2001). Data clustering using potential codes. Bulletin of Vinnytsia Polytechnic Institute, 6, 61-64.
43. Bykov, M. M., Konate, K., & Raimi, A. (2009). Theoretical foundations of information representation and processing using potential codes. Bulletin of VPI, 2, 88-98.
44. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. Cambridge: MIT Press.
45. Weiss, M. A. (2013). Data Structures and Algorithm Analysis in C++ (4th ed.).

Upper Saddle River: Pearson.

46. M. A. & A. J. P. (2014). Performance Evaluation of Sorting Algorithms: A Study. Journal of Computer Applications, 96(2), 30-35.
47. Vyankateshpareek V. Sorting & Searching Algorithms. [Електронний ресурс]. URL: https://medium.com/@vyankateshpareek733/sorting-algorithms-11a9057d688f
48. Brown M. Understanding Algorithm Complexity / M. Brown. – Cambridge: Academic Press, 2020. – 180 с.
49. Johnson T. Measuring Algorithm Performance / T. Johnson. – New York: Springer, 2019. – 220 с.
50. Smith R. Database Transactions and ACID Principles / R. Smith. – San Francisco: Oracle Press, 2018. – 300 с.
51. Patel A., Kumar V. Algorithm Analysis for Search and Booking Systems / A. Patel, V. Kumar. – Oxford: Elsevier, 2017. – 250 с.
52. Williams J. Evaluating Algorithm Performance in Service Booking Systems / J. Williams. – London: InTechOpen, 2019. – 310 с.
53. Taylor L., Roberts D. Automating Service Booking: A Study of Algorithm Efficiency / L. Taylor, D. Roberts. – Berlin: Springer, 2021. – 280 с.
54. Green P. Database Management Systems / P. Green. – New York: McGraw-Hill, 2020. – 350 с.
55. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts. McGraw-Hill.
56. Niode P. Security Best Practices – Node. [Електронний ресурс]. URL: https://nodejs.org/en/learn/getting-started/security-best-practices
57. Myśliwiec K. NestJS Documentation. [Електронний ресурс]. URL: https://docs.nestjs.com
58. Forta B. SQL (TypeORM). [Електронний ресурс]. URL: https://docs.nestjs.com/recipes/sql-typeorm
59. David B. What Does a Front-End Developer Do? Complete Guide to the Front-End Developer Profession. [Електронний ресурс]. URL:

https://frontendmasters.com/guides/front-end-handbook/2018/what-is-a-FD.html
60. Evan Y. Vuex. [Електронний ресурс]. URL: https://vuex.vuejs.org/guide
61. Stonebraker M. PostgreSQL: The World's Most Advanced Open Source Relational Database. [Електронний ресурс]. URL: https://www.postgresql.org
62. Solid IT. DB-Engines Ranking. [Електронний ресурс]. URL: https://db-engines.com/en/ranking
63. Handsontable. 9 Best JavaScript Frameworks to Use in 2024. [Електронний ресурс]. Retrieved from https://ninetailed.io/blog/best-javascript-frameworks
64. Reenskaug T. MVC. [Електронний ресурс]. URL: https://brander.ua/technologies/mvc
65. Helm R. Mediator. [Електронний ресурс]. URL: https://refactoring.guru/design-patterns/mediator
66. Erich G. Adapter. [Електронний ресурс]. URL: https://refactoring.guru/design-patterns/adapter
67. Judy B. Web Content Accessibility Guidelines (WCAG). [Електронний ресурс]. URL: https://www.w3.org/TR/WCAG21
68. Evan Y. Composition API FAQ. [Електронний ресурс]. URL: https://vuejs.org/guide/extras/composition-api-faq
69. Jeff F. SQL Injection. [Електронний ресурс]. URL: https://owasp.org/www-community/attacks/SQL_Injection
70. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
71. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.
72. Метод вказівки до МКР – 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» (кафедра комп’ютерних систем управління)

ДОДАТКИ

Додаток А
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: « Розробка автоматизованої системи для пошуку та бронювання послуг »

Тип роботи: магістерська кваліфікаційна робота –
(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра комп'ютерних систем управління
(кафедра, факультет (інститут), навчальна група)

Керівник Биков М.М., професор кафедри КСУ
(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	99%
Загальна схожість	1%

Аналіз звіту подібності (відмітити потрібне)

√ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботунаправити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або вній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Богдан ПОЛЬГУЛЬ
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток Б – Технічне завдання

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО

Зав. Кафедри КСУ ВНТУ,

д.т.н., процесор

_____В'ячеслав КОВТУН

«14» 10 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Розробка автоматизованої системи для пошуку та бронювання послуг»

08–31.МКР.009.00.000 ТЗ

Керівник магістерської кваліфікаційної

роботи: проф., к.т.н.

Микола БИКОВ

"___" _____ 2024 р.

Виконавець: ст. гр. 2АКІТ–23м

Богдан ПОЛЬГУЛЬ

"___" _____ 2024 р.

Вінниця 2024

1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи для пошуку та бронювання послуг.

1.2. Галузь застосування – автоматизація процесів бронювання та управління послугами у сферах гостинності, транспорту, охорони здоров'я, туризму та інших сервісних індустріях.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від “17” вересня 2024 року №310

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності процесу пошуку та бронювання послуг за рахунок розробки системи для автоматизації даного процесу.

4. Джерела розробки.

Магістерська дипломна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. "Optimizing Sorting Algorithms for Large Data Sets," [Електронний ресурс] – Режим доступу: https://blog.algorithmexamples.com/sorting-algorithm/top-15-sorting-algorithms-for-large-datasets/"
2. "Sorting and Searching Algorithms Performance," [Електронний ресурс] – https://www.doabledanny.com/searching-and-sorting-algorithms-in-javascript.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. "Database System Concepts," McGraw-Hill, 2010.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- пошук послуг
- бронювання послуг
- керування броюваннями
- аналіз даних
- користувацький інтерфейс

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 7\10;
- Microsoft Visual Studio 17;

5.2.2. Умови експлуатації системи:

- робота на стандартних ПК в приміщеннях зі стандартними умовами;
- можливість цілодобового функціонування системи;
- текст програмного забезпечення системи є цілком закритим.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1	Обґрунтування доцільності створення автоматизованої системи для пошуку та бронювання послуг. Постановка задач	03.09.2024 р.
2	Розробка технічного завдання	08.09.2024 р.
3	Розробка архітектури системи	11.09.2024 р.
4	Вибір технологій розробки системи	20.09.2024 р.
5	Розробка програмного забезпечення системи	02.11.2024 р.
6	Тестування програмного забезпечення	15.11.2024 р.
7	Оформлення пояснювальної записки	19.11.2024 р.
8	Апробація і публікації результатів роботи	24.11.2024 р.

6.2 Графічні матеріали:

1	Плакати демонстраційні	21.11. 2024 р.
2	Діаграма бізнес-процесу запису клієнта	22.11. 2024 р.
3	Діаграма бізнес-процесу замовлення витратних матеріалів	24.11. 2024 р.
4	Схема алгоритму реалізації автоматизованої системи для пошуку та бронювання послуг	26.11. 2024 р.
5	UM-діаграма послідовності	28.11. 2024 р.
6	Результати тестування	29.11. 2024 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «_20_» __листопада___ 2024 р.

7.2. Атестація проекту здійснюється на попередньому захисті. Попередній захист магістерської дипломної роботи провести до «01 » _грудня_____2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ДЕК. Захист магістерської дипломної роботи провести до «_10_» _грудня__ 2024 р.

Додаток В
(довідниковий)

Лістинги програм

Лістинг програми пошуку

```
<template lang="pug">
.autocomplete-results( :class="{'autocomplete-results_absolute': absolute, 'autocomplete-results_large': large}")
  ul.autocomplete-results__list
    li.autocomplete-results__item(
      v-for='(suggest, index) in checkItems()'
      :key='index'
      @click="onClickSuggestItem(suggest)"
      :class="setClassActivity(suggest)"
    )
      //- .autocomplete-results__item-prepend
        svg-icon(v-if="prependInner" :name="prependInner")
      .autocomplete-results__text.ml-10(v-html='attr ? suggest[attr] : suggest')
</template>

<script>
export default {
  name: 'AutocompleteResults',
  props: {
    items: {
      type: Array,
      default: () => []
    },
    value: {
      type: String,
      default: ''
    },
    prependInner: {
```

```
    type: String,
    default: ''
  },
  attr: {
    type: String,
    default: ''
  },
  absolute: {
    type: Boolean,
    default: true
  },
  large: {
    type: Boolean,
    default: false
  }
},
methods: {
  onClickSuggestItem(suggest) {
    this.$emit('select', suggest)
  },
  checkItems() {
    if (this.items.length) {
      return this.items
    } else return [{ name: 'По запиту нічого не знайдено' }]
  },
  isEmpty() {
    return !this.items.length
  },
  setClassActivity(item) {
    return {
      'autocomplete-results__item_selected':
        this.value === (this.attr ? item[this.attr] : item),
      'autocomplete-results__item_empty': !this.items.length
    }
  }
}
```

```
}
</script>

<style lang="scss">
.autocomplete-results {
  width: 100%;
  display: flex;
  flex-direction: column;
  margin: 4px 0;
  background: $main-bg-color;
  border-radius: 6px;
  border: 1px solid rgba(0, 0, 0, 0.07);
  &_absolute {
    position: absolute;
    top: 58px;
    left: 0;
  }
}

.autocomplete-results.autocomplete-results_large {
  margin: 0px 0;
}

.autocomplete-results__list {
  padding: 11px 0 !important;
}
.autocomplete-results__item {
  display: flex;
  align-items: center;
  padding: 4px 7px;
  &:hover {
    cursor: pointer;
    background: rgba(0, 0, 0, 0.07);
  }
  &_selected {
    background: rgba(0, 0, 0, 0.05);
```

```
  }
}
.autocomplete-results__item-prepend {
  margin-right: 4px;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  width: 31px;
  svg {
    width: 16px;
    height: 17px;
    opacity: 0.5;
  }
}
.autocomplete-results__text {
  text-overflow: ellipsis;
  white-space: nowrap;
  overflow: hidden;
  font-size: 15px;
  line-height: 18px;
}
.autocomplete-results__item_empty {
  opacity: 0.5;
  pointer-events: none;
}

@media only screen and (min-width: map-get($grid-breakpoints, 'md')) {
  .autocomplete-results {
    margin: 4px 0;
    &_absolute {
      top: 58px;
    }
  }

  .autocomplete-results__item-prepend {
```

```
    margin-right: 3px;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-shrink: 0;
    width: 35px;
    svg {
      width: 18px;
      height: 17px;
    }
  }
  .autocomplete-results__list {
    padding: 15px 0 !important;
  }
  .autocomplete-results__item {
    padding: 8px 13px;
  }

  .autocomplete-results.autocomplete-results_large {
    margin: 2px 0;
    .autocomplete-results__item {
      padding: 8px 21px;
    }
  }

  .autocomplete-results__text {
    font-size: 16px;
    line-height: 19px;
  }
}
</style>
```

Лістинг програми Місцезнаходження

```
<template lang="pug">
```

```
v-dialog(content-class="custom-dialog" v-model='value' width='656' persistent)
  .custom-dialog__content
    .custom-dialog__close
      svg-icon(name="close" @click="$emit('input', false)" color='#0d0d0d')
    .custom-dialog__title Оберіть населений пункт
    LocationAutocomplete(
      v-model="payload.address"
      :absolute="false"
      ref="search"
      :suggesters="suggesters"
      @autocomplete-handler="autocompleteHandler"
      @handler-select-location="handlerSelectLocation"
      @suggesters-clear="suggesters=[]"
      placeholder="Населений пункт"
      prependInner="search"
    )
  Loader(v-if="loading")
  .custom-dialog__footer
    .select-location__my-location( @click="getLocationAgain" )
      svg-icon(name="gps" color='#0d0d0d')
      | Моє місцезнаходження
</template>

<script>
import { mapActions } from 'vuex'
import { Geolocation } from '@capacitor/geolocation'
import { Preferences } from '@capacitor/preferences'
import locations from '@/mixins/locations'

export default {
  name: 'DialogLocation',
  components: {
    LocationAutocomplete: () =>
      import('~/components/UIcontrols/input/LocationAutocomplete')
  },
  mixins: [locations],
```

```
props: {
  value: {
    type: Boolean,
    default: false
  }
},
data() {
  return {
    selectedAddress: '',
    loading: false,
    payloadLoaded: false,
    payload: {
      address: '',
      latitude: '',
      longitude: ''
    }
  }
},
watch: {
  value(newValue) {
    if (!newValue) {
      this.$emit('input', false)
    }
  }
},
methods: {
  ...mapActions({
    getAddress: 'search/getAddress'
  }),
  async handlerSelectLocation() {
    const { latitude, longitude } = await this.geocoderHandler(
      this.payload.address
    )
    this.selectedAddress = this.payload.address
    if (latitude && longitude) {
      Object.assign(this.payload, { latitude, longitude })
```

```
    await Preferences.set({
      key: 'geolocation',
      value: JSON.stringify(this.payload)
    })
  }

  this.$emit('search', this.payload)
  this.$emit('input', false)
},
async getLocation() {
  const permission = await this.hasLocationPermission()
  if (navigator.geolocation && permission) {
    await this.getCurrentPosition()
  } else {
    this.payloadLoaded = true
  }
},
async getLocationAgain() {
  if (navigator.geolocation) {
    await Geolocation.checkPermissions()
      .then((res) => {
        if (res.location === 'denied') {
          this.requestLocation()
        } else {
          this.loading = true
          this.getCurrentPosition()
        }
      })
      .catch((error) => {
        console.log('checkPermissions error', error)
      })
  }
},
async hasLocationPermission() {
  return await Geolocation.checkPermissions()
    .then((res) => {
```

```
      if (res.location === 'granted') {
        return true
      } else {
        return false
      }
    })
    .catch((error) => {
      console.log('hasLocationPermission error', error)
      return false
    })
},
async requestLocation() {
  await Geolocation.requestPermissions()
    .then(() => {
      this.getCurrentPosition()
    })
    .catch((error) => {
      if (error.code === 'UNIMPLEMENTED') {
        alert('Спочатку увімкніть визначення геоданих в вашому браузері')
      }
    })
},
async getCurrentPosition() {
  await Geolocation.getCurrentPosition()
    .then((position) => {
      this.payload.latitude = position.coords.latitude
      this.payload.longitude = position.coords.longitude
      this.getAddress({
        latitude: this.payload.latitude,
        longitude: this.payload.longitude
      })
        .then((res) => {
          this.payload.address = this.selectedAddress = res.data.results
            .length
            ? res.data.results[0].formatted_address
            : ''
```

```
            Preferences.set({
              key: 'geolocation',
              value: JSON.stringify(this.payload)
            })

            this.$emit('search', this.payload)
            this.hideLocationLoader()
          })
          .catch((error) => {
            this.loading = false
            console.log('error', error)
          })
          .finally(() => {
            this.hideLocationLoader()
          })
        })
        .catch((error) => {
          this.loading = false
          if (error.code !== error.PERMISSION_DENIED) {
            console.log('getCurrentPosition error:', error)
            alert('Не вдалося визначити вашу геолокацію. Встановіть її вручну')
          }
        })
        .finally(() => {
          this.payloadLoaded = true
        })
    },
    hideLocationLoader() {
      this.$emit('input', false)
      this.loading = false
    }
  }
}
</script>
```

Лістинг програми результату пошуку

```
<template lang="pug">
.institution-item-wrapper(:class="{'institution-item-wrapper_active': isSelectedItem}"
@click="onGoToRecord")
  .institution-item__bg
  .institution-item__second-bg
    .institution-item__inner-second-bg
  .institution-item
    .institution-item__wrapp
      .institution-item__info
        .institution-item__logo-wrapper
          img.institution-item__logo(:src="avatarLink" alt="Master user")
        .institution-item__contact-data
          .institution-item__master {{masterName}}
          .institution-item__price-wrapper.d-md-none
            .institution-item__price {{item.tariff_service.price}} ₴
            .institution-item__duration {{item.tariff_service.duration}} хв
          .institution-item__title(v-if="item.staff.company && item.staff.company.company_type !==
'self_employed'") {{item.staff.company.name}}
          .institution-item__location-wrapper(v-if="item.staff.company")
            .institution-item__distance(v-if="typeof item.staff.company.distance === 'number'") {{distanceTitle}}
            .institution-item__location {{item.staff.company.address}}
          .institution-item__time-wrapper
            .institution-item__time-label найближчий час:
            .institution-item__time {{nearestFreeTime}}
      .institution-item__action.d-none.d-md-flex
        .institution-item__action-top
          .institution-item__price {{item.tariff_service.price}} ₴
          .institution-item__duration {{item.tariff_service.duration}} хв
</template>

<script>
import { DateTime, Duration } from 'luxon'
import { mapGetters, mapMutations } from 'vuex'
```

```
export default {
  name: 'InstitutionItem',
  props: {
    item: {
      type: Object,
      default: () => {}
    }
  },
  computed: {
    ...mapGetters({ selectedItem: 'search/selectedItem' }),
    nearestFreeTime() {
      if (this.item.nearest_free_time && this.item.nearest_free_time.date) {
        const nearestHours =
          this.item.nearest_free_time.time &&
          this.item.nearest_free_time.time.length
            ? this.item.nearest_free_time.time[0]
            : 1
        const date = DateTime.fromISO(this.item.nearest_free_time.date)
        const dateString = date.setLocale('uk-UA').toFormat('d MMMM')
        const hoursString = Duration.fromMillis(nearestHours * 60000).toFormat(
          'hh:mm'
        )
        return `${dateString} ${hoursString}`
      } else {
        return ''
      }
    },
    avatarLink() {
      return this.item.staff.avatar && this.item.staff.avatar.url
        ? this.item.staff.avatar.url
        : '/avatar.jpg'
    },
    masterName() {
      return `${this.item.staff.first_name} ${this.item.staff.last_name}`
    },
```

```
  isSelectedItem() {
    if (this.selectedItem && this.item) {
      return (
        this.item.tariff_service.id === this.selectedItem.tariff_service.id &&
        this.item.staff.id === this.selectedItem.staff.id
      )
    } else {
      return false
    }
  },
  distanceTitle() {
    const distance = this.item.staff.company.distance
    const km = 1000

    if (distance === 0) {
      return `5 м`
    }

    if (distance < 1 && distance > 0) {
      return `${distance * km} м`
    }

    return `${Math.round(distance)} км`
  }
},
methods: {
  ...mapMutations({ SET_SELECTED_ITEM: 'search/SET_SELECTED_ITEM' }),
  recordEvent() {
    this.SET_SELECTED_ITEM(this.item)
  },
  onGoToRecord() {
    const query = `service_id=${this.item.tariff_service.id}`

    location.href = `${process.env.APP_CLIENT_URL}/staff/${this.item.staff.id}/menu?${query}`
  }
}
```

```
}
</script>
<style lang="scss">
.institution-item-wrapper {
  position: relative;
  padding: 2px;
  &.institution-item-wrapper_active,
  &:hover {
    .institution-item__bg {
      background: linear-gradient(115.91deg, #ffb342 -9.26%, #9131dc 120.15%);
    }
    .institution-item__inner-second-bg {
      background: linear-gradient(
        115.91deg,
        rgba(#ffb342, 0.15) -9.26%,
        rgba(#9131dc, 0.15) 120.15%
      );
    }
  }
  &.institution-item-wrapper_active {
    .institution-item__inner-second-bg {
      background: linear-gradient(
        115.91deg,
        rgba(255, 179, 66, 0.15) -9.26%,
        rgba(145, 49, 220, 0.15) 120.15%
      );
    }
  }
}
.institution-item {
  position: relative;
  width: 100%;
  min-height: 182px;
  padding: 0 0 7px 0;
  border-radius: 6px;
  cursor: pointer;
```

```
  z-index: 2;
}
.institution-item__bg {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  border-radius: 6px;
  transition: all 0.25s;
  background: linear-gradient(
    0deg,
    rgba(13, 13, 13, 0.07) 0,
    rgba(13, 13, 13, 0.07) 100%
  );
  z-index: 0;
}
.institution-item__inner-second-bg {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  border-radius: 6px;
  transition: all 0.25s;
  background: white;
  opacity: 1;
  z-index: 1;
}
.institution-item__second-bg {
  position: absolute;
  top: 2px;
  left: 2px;
  width: calc(100% - 4px);
  height: calc(100% - 4px);
  border-radius: 6px;
```

```
  transition: all 0.25s;
  background: white;
  opacity: 1;
  z-index: 1;
}
.institution-item__wrapp {
  display: flex;
  justify-content: space-between;
}

.institution-item__info {
  display: flex;
  flex-direction: column;
  width: 100%;
}

.institution-item__logo-wrapper {
  margin: 0 auto;
  width: 220px;
  height: 220px;
  position: relative;
  flex-shrink: 0;
}

.institution-item__logo {
  width: 100%;
  height: 100%;
  object-fit: cover;
  border-radius: $regular-border-radius $regular-border-radius 0 0;
  background-color: $secondary-bg-color;
}

.institution-item__contact-data {
  padding: 17px 16px 16px 16px;
}
.institution-item__price-wrapper {
```

```
  display: flex;
  align-items: center;
  margin-bottom: 10px;
  & > *:not(:last-child) {
    position: relative;
    margin-right: 22px;
    &:after {
      content: '';
      position: absolute;
      background: rgba(13, 13, 13, 0.7);
      width: 5px;
      height: 5px;
      border-radius: 100%;
      right: -11px;
      top: calc(50% - 2px);
      opacity: 0.5;
    }
  }
}
.institution-item__master {
  @include medium-text;
  font-size: 18px;
  line-height: 24px;
  margin-bottom: 9px;
}

.institution-item__title,
.institution-item__distance,
.institution-item__location {
  font-size: 12px;
  position: relative;
  width: fit-content;
}
.institution-item__title {
  line-height: 16px;
}
```

```
.institution-item__distance,
.institution-item__location {
  line-height: 14px;
  display: inline;
}

.institution-item__distance {
  white-space: nowrap;
}

.institution-item__location-wrapper {
  display: inline-block;
  margin-bottom: 4px;
  line-height: 0;
  & > *:not(:last-child) {
    margin-right: 19px;
    &:after {
      content: '';
      position: absolute;
      background: rgba(13, 13, 13, 0.7);
      width: 4px;
      height: 4px;
      border-radius: 100%;
      right: -12px;
      top: calc(50% - 2px);
    }
  }
}

.institution-item__time-wrapper {
  display: flex;
  align-items: center;
  & > *:not(:last-child) {
    margin-right: 5px;
  }
}
```

```
.institution-item__time-label {
  &:first-letter {
    text-transform: uppercase;
  }
}
.institution-item__time-label,
.institution-item__time {
  font-size: 12px;
  line-height: 16px;
}

.institution-item__action {
  padding-top: 14px;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: flex-end;
}
.institution-item__action-top {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: flex-end;
}

.institution-item__price {
  font-size: 15px;
  line-height: 16px;
  white-space: nowrap;
}
.institution-item__duration {
  font-size: 15px;
  line-height: 16px;
  opacity: 0.5;
  white-space: nowrap;
}
```

```
@media only screen and (min-width: $extra-breakpoints) {
  .institution-item {
    box-shadow: unset;
    min-height: 148px;
    border-radius: 6px;
    padding: 18px 22px 18px 18px;
  }
  .institution-item__info {
    flex-direction: row;
  }
  .institution-item__logo-wrapper {
    margin: 0;
    width: 148px;
    height: 148px;
    margin-right: 20px;
  }
  .institution-item__action {
    padding-top: 4px;
  }
  .institution-item__logo {
    border-radius: $regular-border-radius;
  }

  .institution-item__info {
    flex-direction: row;
  }
  .institution-item__master {
    font-size: 24px;
    line-height: 30px;
    margin-bottom: 16px;
  }
  .institution-item__contact-data {
    padding: 0;
  }
  .institution-item__title {
```

```
  margin-bottom: 8px;
}
.institution-item__title,
.institution-item__distance,
.institution-item__location {
  font-size: 16px;
  line-height: 24px;
}
.institution-item__location-wrapper {
  margin-bottom: 8px;
  & > *:not(:last-child) {
    margin-right: 25px;
    &:after {
      width: 5px;
      height: 5px;
      right: -15px;
    }
  }
}
.institution-item__price {
  font-size: 18px;
  line-height: 22px;
}
.institution-item__duration {
  font-size: 18px;
  line-height: 32px;
  opacity: 0.5;
}
.institution-item__time-label {
  &:first-letter {
    text-transform: unset;
  }
}
.institution-item__time-label,
.institution-item__time {
  font-size: 15px;
```

```
    line-height: 24px;
  }
}

@media only screen and (min-width: map-get($grid-breakpoints, 'md')) {
  .institution-item__info {
    margin-right: 20px;
  }
  .institution-item__master {
    font-size: 30px;
    line-height: 36px;
    margin-bottom: 24px;
  }
  .institution-item__price {
    font-size: 28px;
    line-height: 32px;
    margin-bottom: 4px;
  }
  .institution-item__logo-wrapper {
    margin-right: 32px;
  }
}
</style>
```

Лістинг програми підключення до бази даних

```
import { Sequelize } from 'sequelize-typescript';
import { postgresModels } from './database.models';

export const databaseProviders = [
  {
    provide: 'SEQUELIZE',
    useFactory: async () => {
      const sequelize = new Sequelize({
        logging: false,
```

```
      dialect: 'postgres',
      host: process.env.DB_HOST,
      port: Number(process.env.DB_PORT),
      username: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
      database: process.env.DB_NAME,
    });
    sequelize.addModels([...postgresModels]);
    await sequelize.sync();
    return sequelize;
  },
 },
];
```

Додаток Г
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ З ПОШУКУ ТА БРОНЮВАННЯ ПОСЛУГ

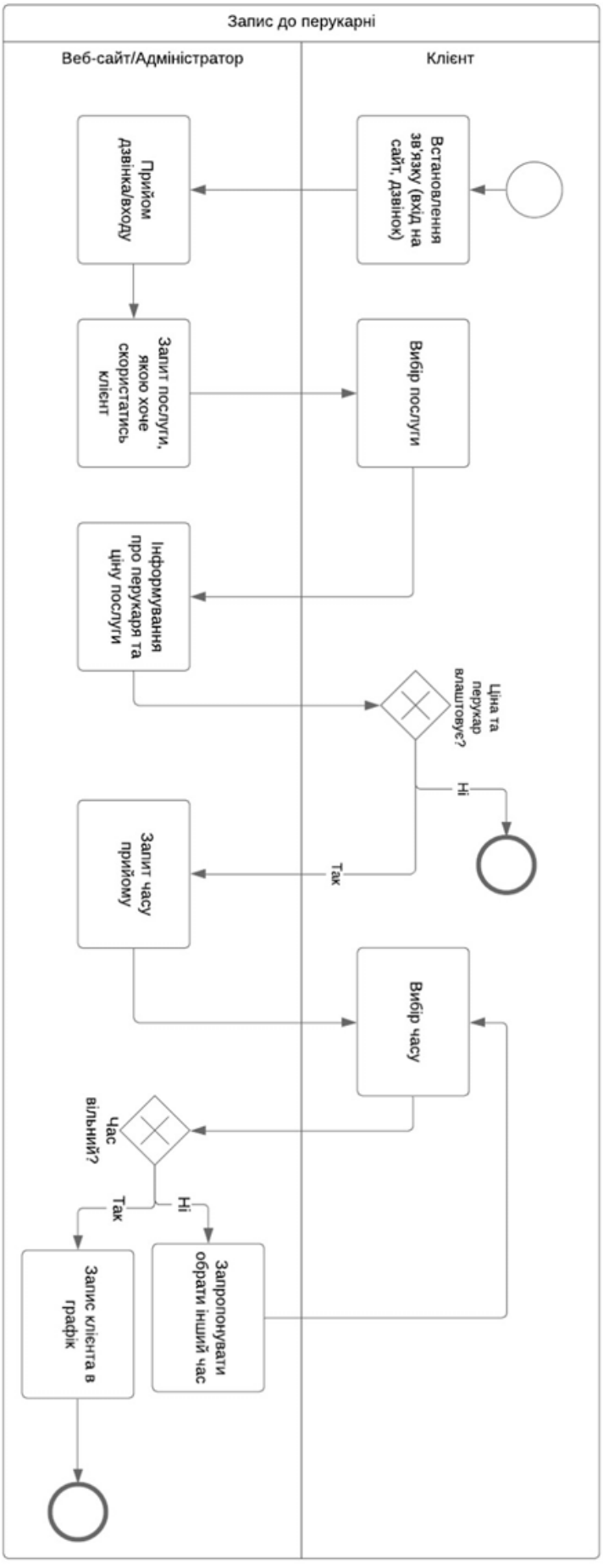
Студент групи 2АКІТ-23м

Підпис Богдан ПОЛЬГУЛЬ
Ім'я ПРІЗВИЩЕ

Керівник к.т.н., доцент, проф. кафедри КСУ

Підпис Микола БИКОВ
Ім'я ПРІЗВИЩЕ

ДІАГРАМА БІЗНЕС-ПРОЦЕСУ ЗАПИСУ КЛІЄНТА



ДІАГРАМА БІЗНЕС-ПРОЦЕСУ ЗАМОВЛЕННЯ ВИТРАТНИХ МАТЕРІАЛІВ

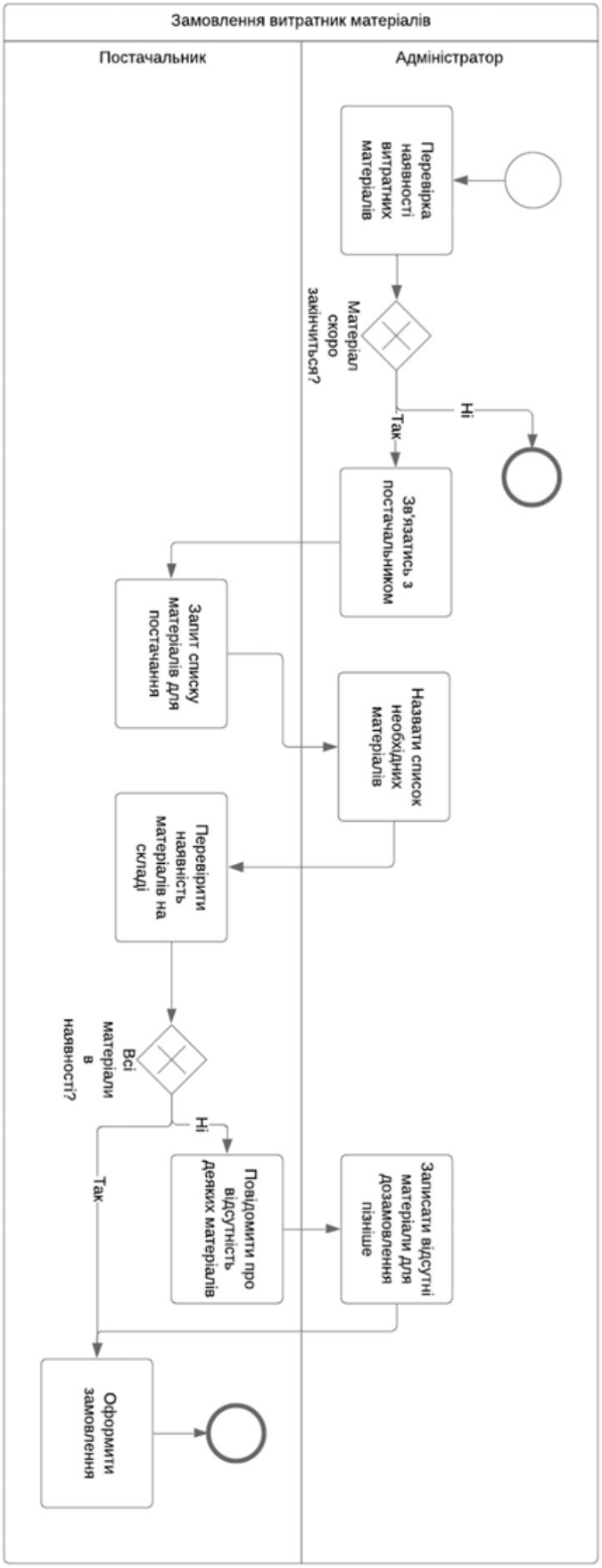
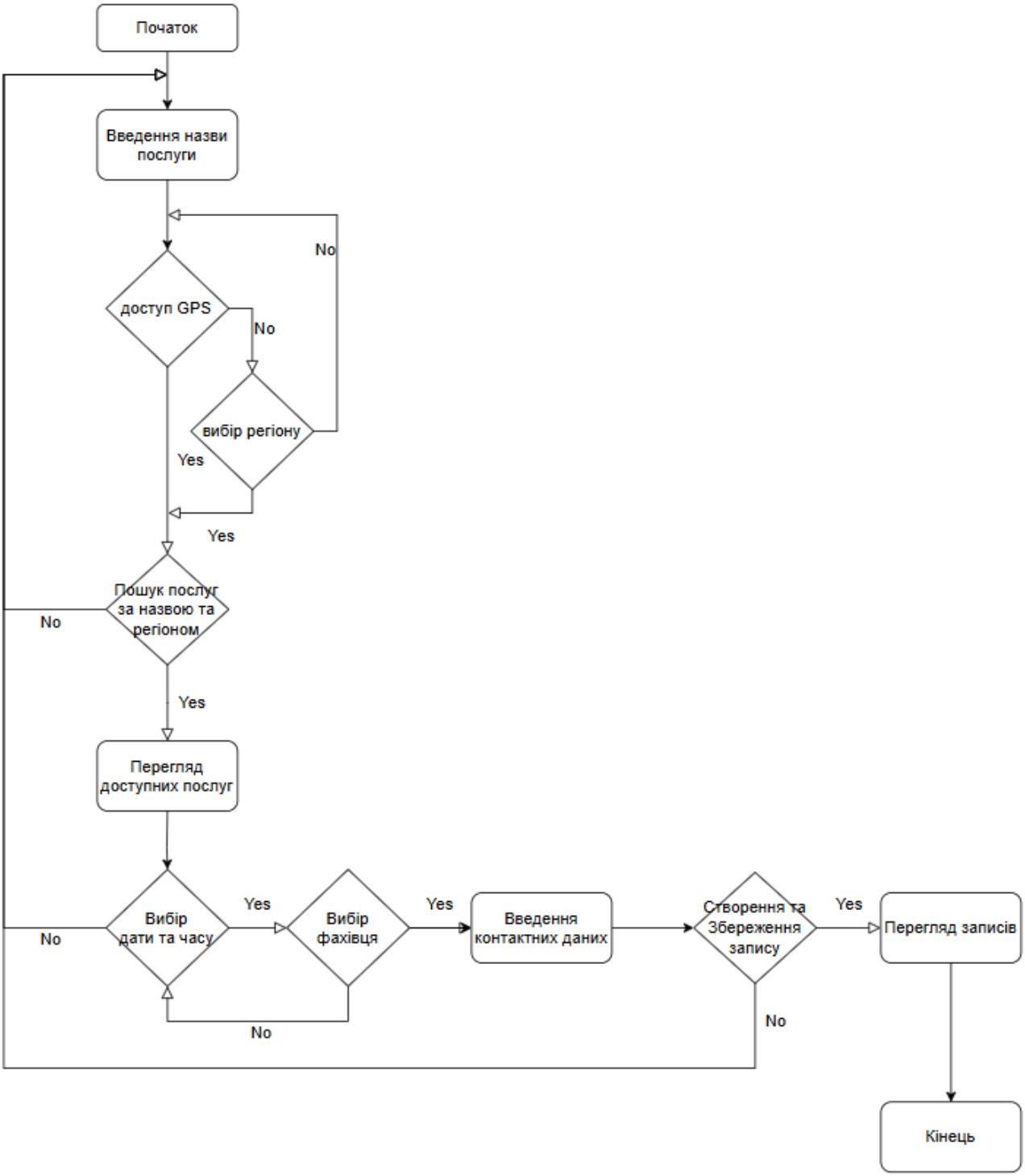
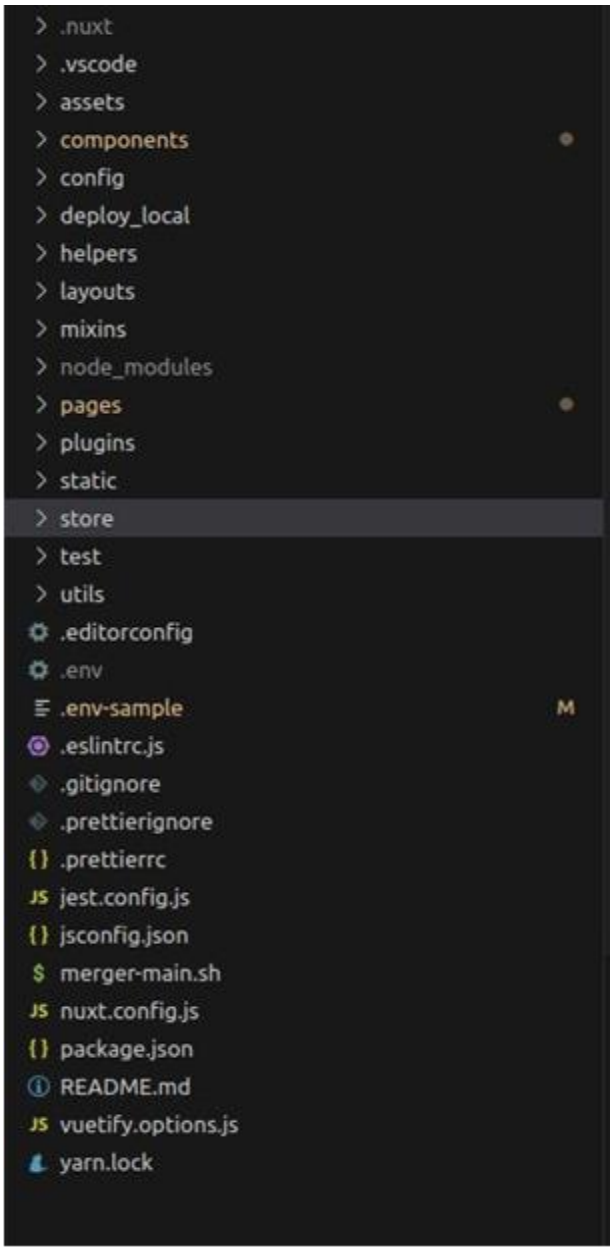


СХЕМА АЛГОРИТМУ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДЛЯ ПОШУКУ ТА БРОНЮВАННЯ ПОСЛУГ

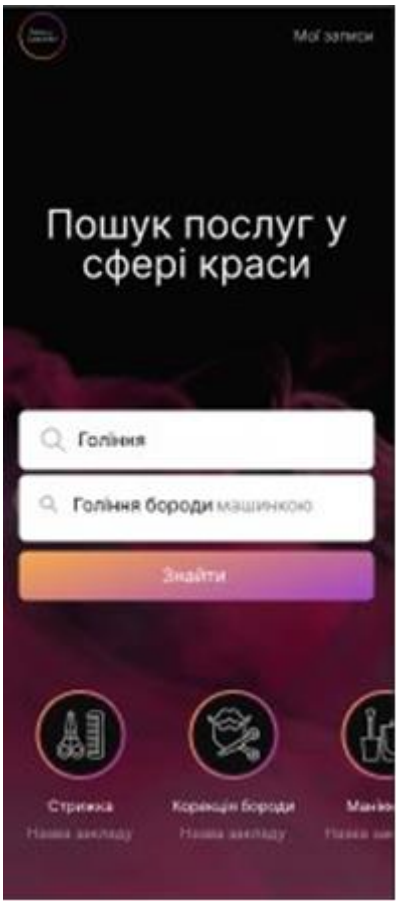


UML-ДІАГРАМА ПОСЛІДОВНОСТІ

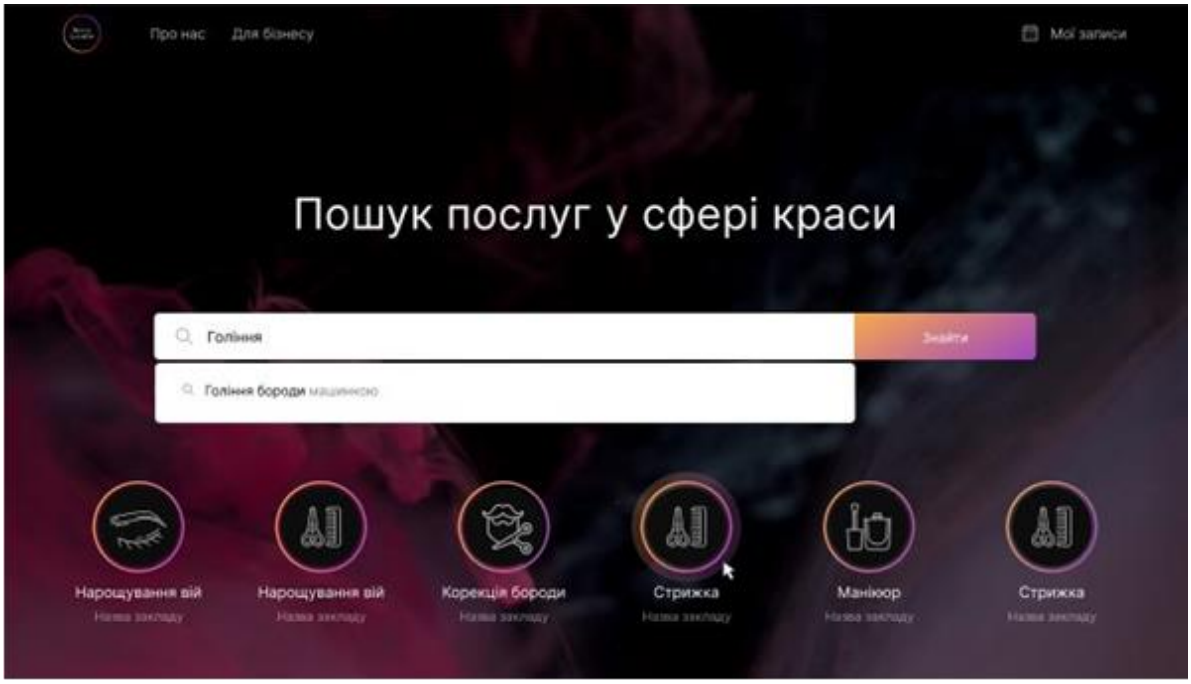
СТРУКТУРА ПРОЕКТУ



АДАПТИВНА ВЕРСТКА ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ



АДАПТИВНА ВЕРСТКА ДЛЯ ДЕСКТОПНИХ ПРИСТРОЇВ



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

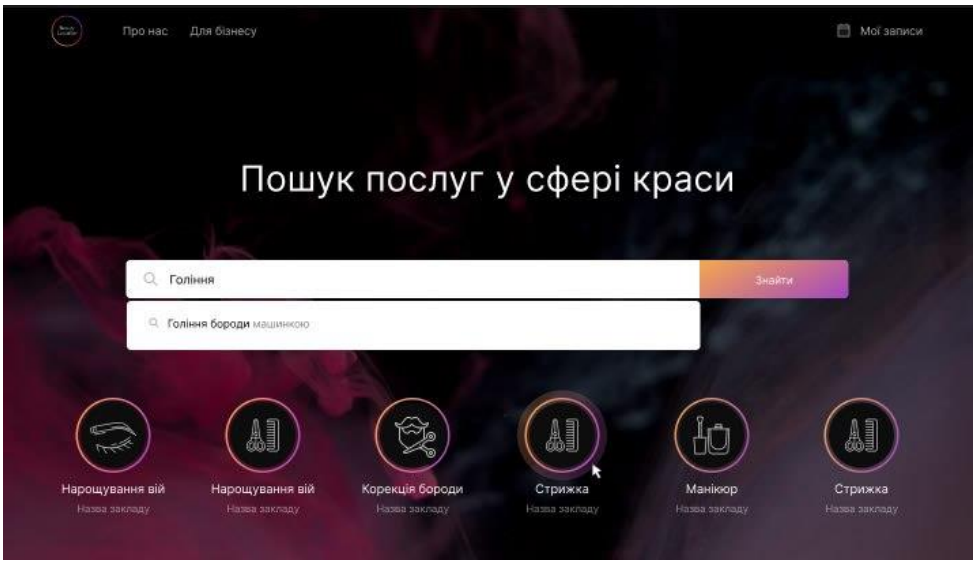


Рисунок 1 – Введення послуги у пошук

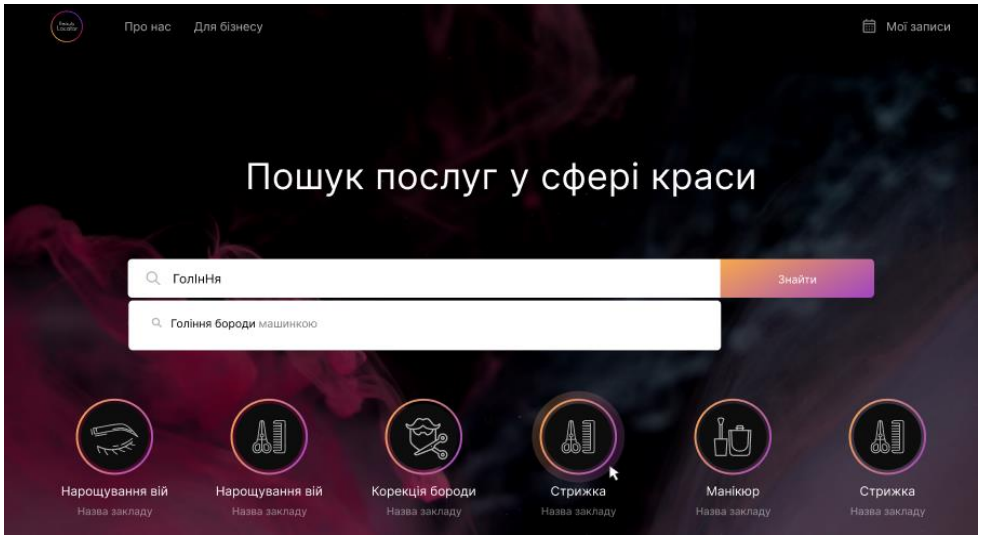


Рисунок 2 – Результати пошуку з різними регістрами

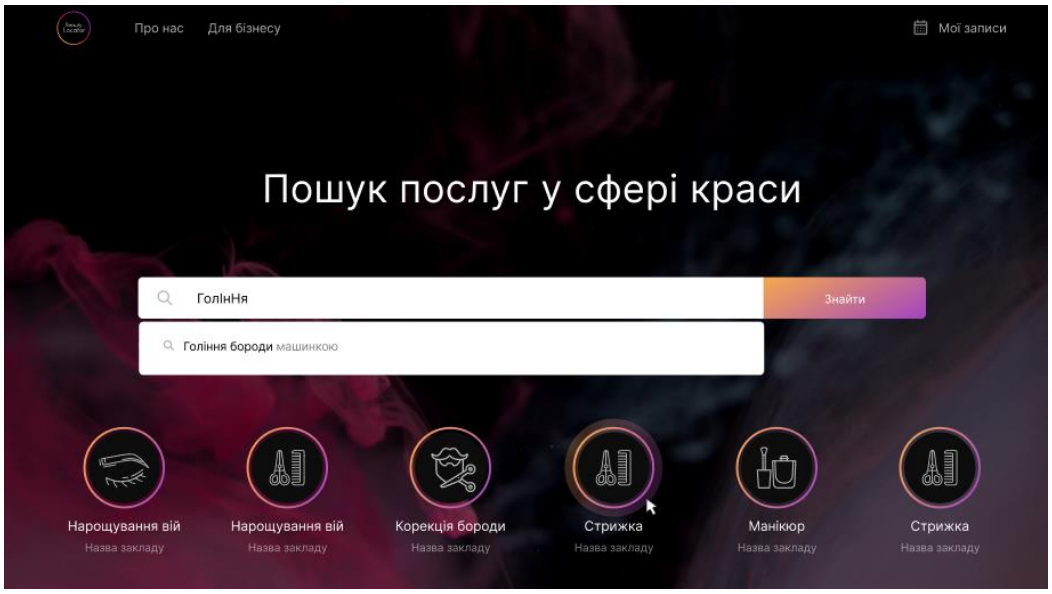


Рисунок 3 – Результати пошуку з пробілами

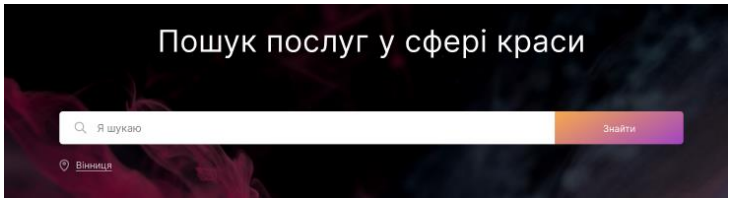


Рисунок 4 – Адреса отримана по GPS

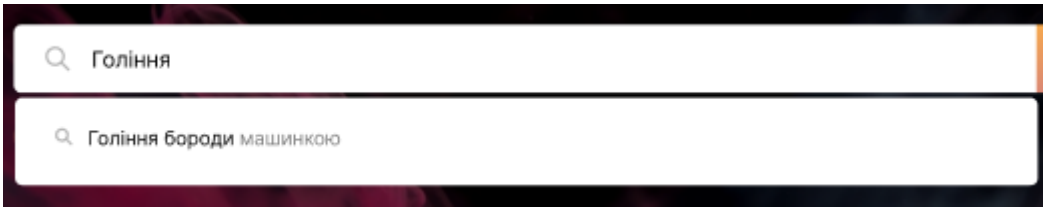


Рисунок 5 – Результат пошуку, що відповідає заданому радіусу GPS

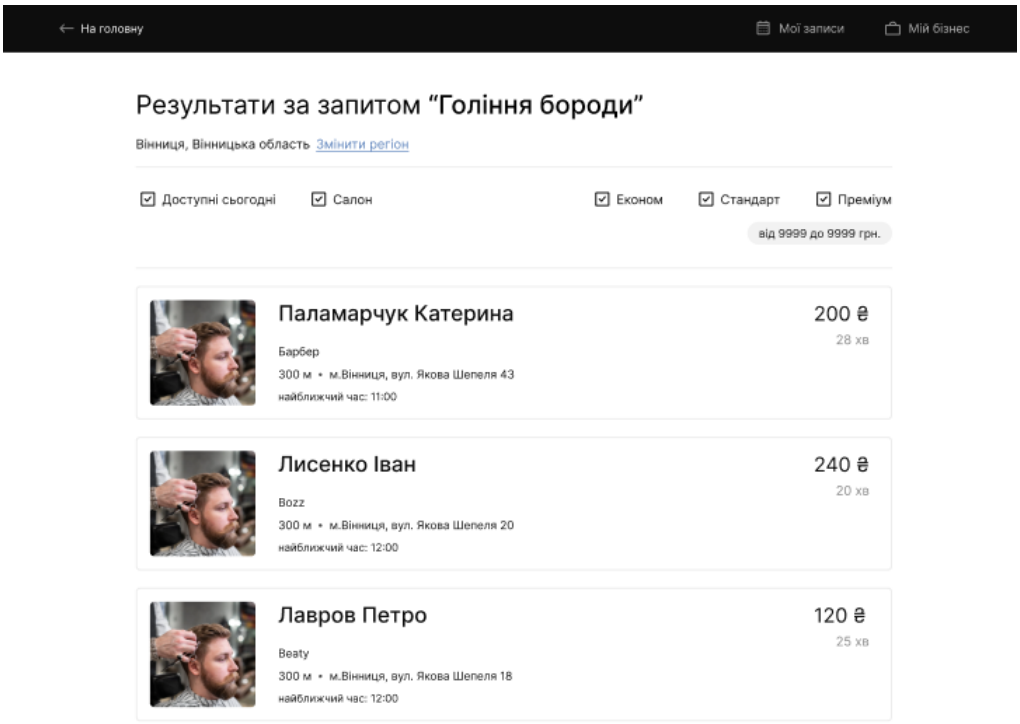


Рисунок 6 – Сторінку результатів запиту, після обрання послуги

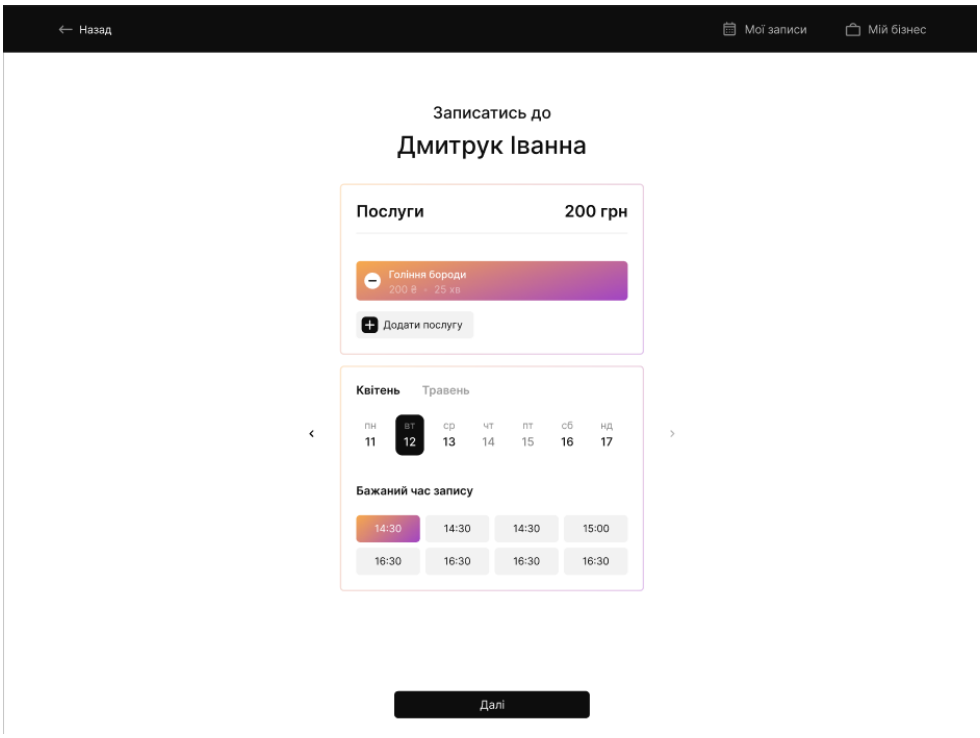


Рисунок 7 – Сторінка обраного запису

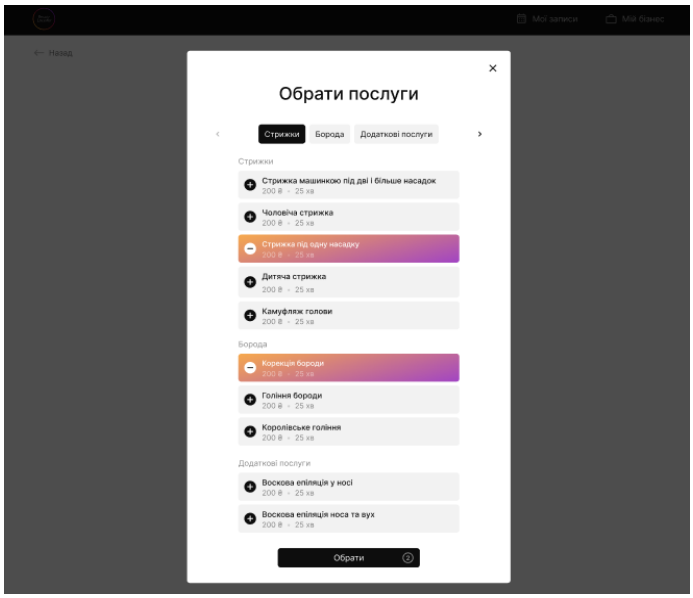


Рисунок 8 – Вікно доступних послуг даного фахівця

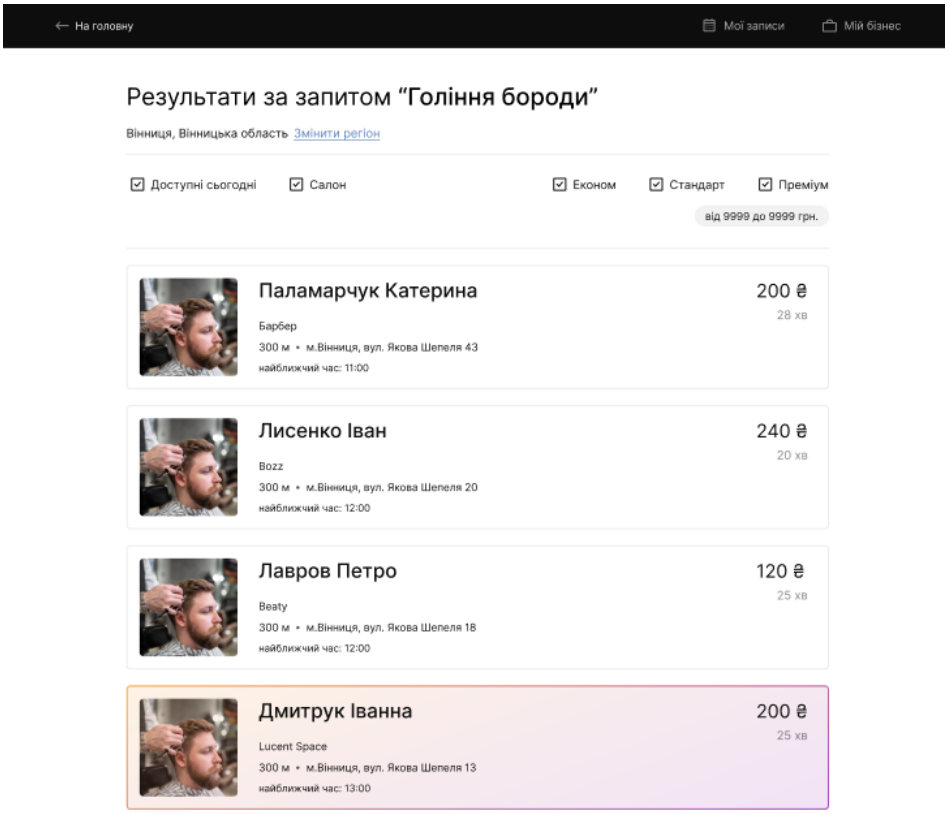


Рисунок 9 – Інформація про фахівців

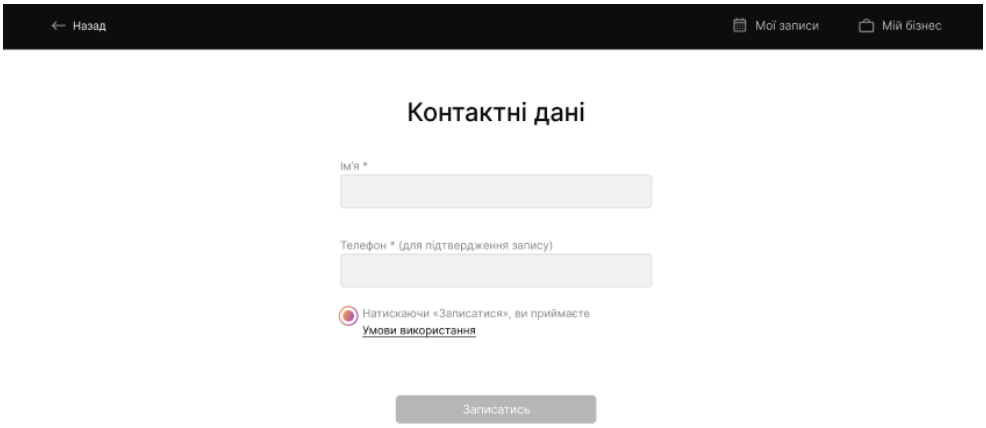


Рисунок 10 – Сторінка введення контактних даних

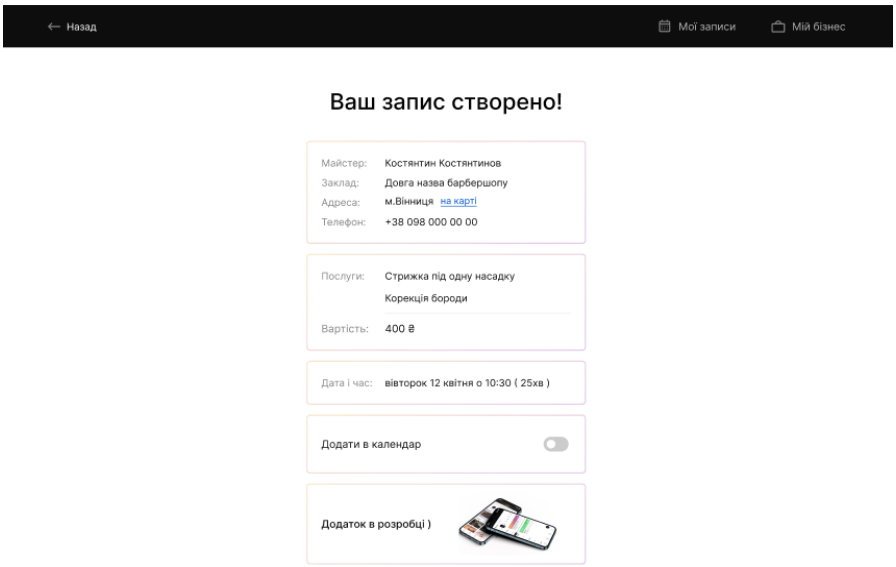


Рисунок 11 – Сторінка результату створеного запису

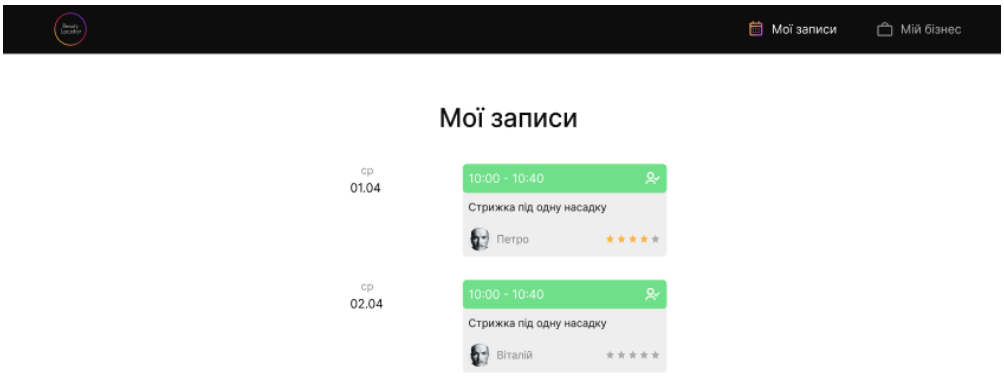


Рисунок 12 - Перелік записів користувача

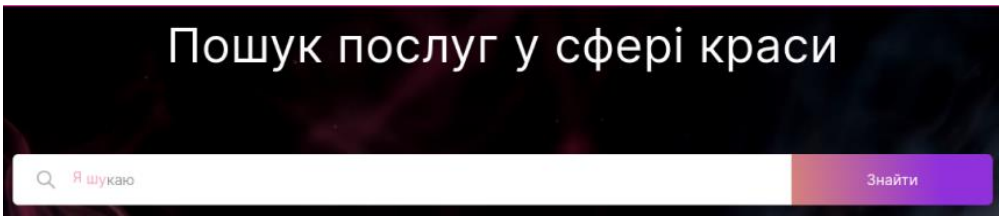


Рисунок 13 – Індикатор введення послуги

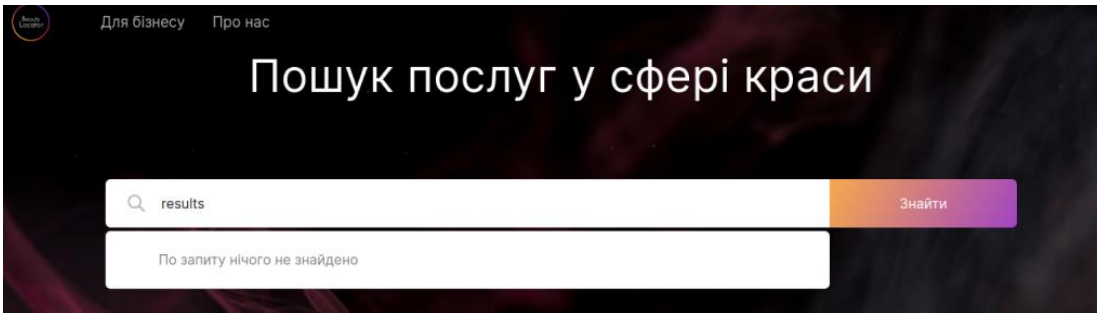


Рисунок 14 – Повідомлення, про відсутність результатів пошуку

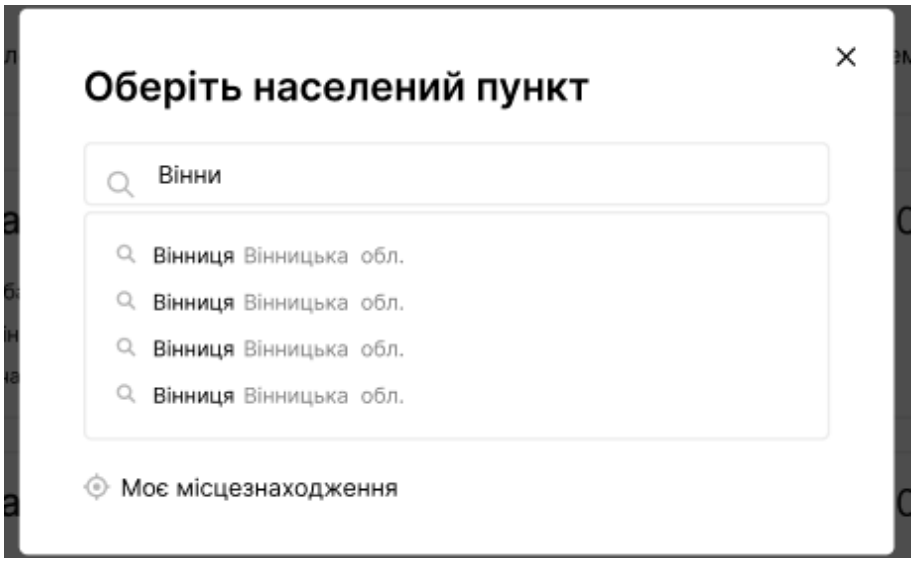


Рисунок 15 – Введення адреси в поле