

Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою

«Комп'ютерні науки»

зі спеціальності

122 – Комп'ютерні науки

тема роботи:

«Система автоматизації бронювання і організації робочих просторів коворкінгу на основі веб технологій»

Виконала ст. гр. КН-22-1 _____ Тімофєєва А. І.

Керівник _____ Савицький О. І.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: інформаційних технологій

Кафедра: автоматизації, комп'ютерних наук і технологій

Ступінь вищої освіти: Бакалавр

Спеціальність: 122 – Комп'ютерні науки

ЗАТВЕРДЖУЮ

Зав. кафедрою: к.т.н. Рубан С.А.

« 29 » січня 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-22-1 Тимофєєвій Анастасії Ігорівні

1. Тема кваліфікаційної роботи: «Система автоматизації бронювання і організації робочих просторів коворкінгу на основі веб-технологій»

затверджено наказом по університету № 173с від 30.03.2026 р.

2. Термін здачі кваліфікаційної роботи: 10.06.2026 р.

3. Склад кваліфікаційної роботи: Пояснювальна записка обсягом 8бс., презентація у Microsoft PowerPoint (20 слайдів) в електронному та друкованому вигляді

4. Консультанти кваліфікаційної роботи:

Розділ 1-2

доц. Савицький О. І.

Нормоконтроль

доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	<i>Вступ</i>	<i>01.04.26</i>
2	<i>Розділ 1</i>	<i>05.04.26</i>
3	<i>Розділ 2</i>	<i>01.05.26</i>
4	<i>Висновки</i>	<i>25.05.26</i>
5	<i>Оформлення кваліфікаційної роботи</i>	<i>28.05.26</i>
6	<i>Підготовка презентації та графічного матеріалу</i>	<i>20.05.26</i>
7	<i>Підготовка доповіді до захисту</i>	<i>10.06.26</i>

6. Дата видачі завдання: 28.01.2026р.

Керівник _____ / Савицький О. І./

7. **Запевнення:** Я, Тімофєєва Анастасія Ігорівна, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студентка _____ / Тімофєєва А. І./

АНОТАЦІЯ

Тімофєєва А. І. Система автоматизації бронювання і організації робочих просторів коворкінгу на основі веб технологій: кваліфікаційна робота бакалавра : 122 – Комп'ютерні науки. Кривий Ріг. Криворізький національний університет, 2026. 86 с.

Актуальність теми полягає в тому, що автоматизація управління бронюванням робочих просторів коворкінгу є важливим кроком у цифровій трансформації малого бізнесу, а розробка спеціалізованої веб-системи з інтерактивним інтерфейсом та транзакційним механізмом резервування є актуальним і практично значущим завданням.

Метою роботи є проєктування та розробка веб-застосунку автоматизації бронювання і організації робочих просторів коворкінгу на основі веб-технологій, що забезпечує цифрове самообслуговування клієнтів та інструменти управління для адміністратора.

У першому розділі було розглянуто загальну характеристику мережі коворкінг-центрів та її організаційну структуру, проведено аналіз поточного стану комп'ютеризації установи та виявлено системні недоліки існуючого процесу бронювання. Здійснено функціональне моделювання бізнес-процесів у нотації BPMN (AS IS та TO BE), проведено огляд та порівняльний аналіз існуючих рішень, а також сформовано технічне завдання на розробку системи.

У другому розділі розглядаються обґрунтування вибору програмно-технічних засобів розробки, проєктування реляційної бази даних. Описано реалізацію функцій системи з наведенням лістингів коду та скріншотів інтерфейсу. Наведено інструкцію користувача та результати функціонального тестування і тестування безпеки.

Ключові слова:

КОВОРКІНГ, АВТОМАТИЗАЦІЯ БРОНЮВАННЯ, ВЕБ-ЗАСТОСУНОК, NEXT.JS, NODE.JS, POSTGRESQL, REST API, БІЗНЕС-ПРОЦЕСИ.

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ.....	8
1.1 Загальна характеристика коворкінг-центру як об'єкта дослідження.....	8
1.2 Організаційна структура коворкінг-центру, роль і взаємодія підрозділів.....	11
1.2.1 Загальна схема організаційної структури коворкінг-центру...	11
1.2.2 Структура та функції адміністративного підрозділу	13
1.3 Аналіз нинішнього стану комп'ютеризації установи та обґрунтування розробки.....	15
1.4 Функціональне моделювання та аналіз бізнес-процесів.....	17
1.4.1 Функціональна модель AS-IS бізнес-процесів коворкінг-центру	17
1.4.2 Пропозиції щодо усунення наявних проблем та функціональна модель TO BE.....	19
1.5 Огляд існуючих рішень для автоматизації бронювання в коворкінгах.....	21
1.6 Обґрунтування доцільності проєктування та розроблення системи автоматизації бронювання і організації робочих просторів коворкінгу.....	25
1.7 Технічне завдання на проєктування.....	27
1.7.1 Мета та цілі створення системи та характеристика об'єкта автоматизації.....	27
1.7.2 Функції системи.....	30
1.7.3 Вимоги до інформаційного забезпечення.....	32
1.7.4 Вимоги до програмного забезпечення.....	34

1.7.5 Вимоги до інтерфейсу користувача, надійності та безпеки...	36
Висновки до розділу.....	38
РОЗДІЛ 2. ПРОЄКТУВАННЯ, СТВОРЕННЯ ТА АПРОБАЦІЯ	
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	39
2.1 Опис та обґрунтування вибору програмно-технічних засобів розроблення програмного продукту.....	39
2.2 Проєктування та створення бази даних.....	41
2.3 Реалізація функцій системи.....	43
2.3.1 Модуль автентифікації.....	43
2.3.2 Модуль перегляду локацій та робочих місць.....	46
2.3.3 Інтерактивна карта для Fixed Desk.....	49
2.3.4 Модуль бронювання.....	51
2.3.5 Особистий кабінет резидента.....	54
2.3.6 Адміністративна панель.....	56
2.4 Інструкція користувача.....	58
2.4.1 Загальна навігація та базові сторінки.....	58
2.4.2 Реєстрація, авторизація та процес бронювання.....	61
2.4.3 Робота адміністратора в панелі управління	64
2.5 Тестування програмного продукту	66
2.5.1 Функціональне тестування API.....	66
2.5.2 Функціональне тестування інтерфейсу.....	68
2.5.3 Тестування безпеки.....	72
2.5.4 Тестування продуктивності та адаптивності.....	75
Висновки до розділу.....	78
ВИСНОВКИ.....	80
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	82

ВСТУП

Ринок гнучких офісних просторів в Україні динамічно розвивається: коворкінги стають популярною альтернативою традиційним офісам для фрілансерів, стартапів та ІТ-компаній. Попри зростаючий попит, більшість коворкінг-центрів досі спирається на ручні методи управління бронюваннями – електронні таблиці, месенджери та телефонні дзвінки, – що призводить до операційних помилок і обмежує доступність сервісу.

Останнім часом активно використовуються сучасні веб-технології для автоматизації управління ресурсами підприємств, зокрема системи онлайн-бронювання з інтерактивними інтерфейсами, хмарними базами даних та REST API, що відкривають нові можливості для цифрової трансформації малого і середнього бізнесу.

Актуальність цієї роботи полягає у розробці спеціалізованої веб-системи автоматизації бронювання і організації робочих просторів коворкінгу, що усуває залежність операційного процесу від людського фактора та надає клієнтам зручний інструмент цифрового самообслуговування.

Метою роботи є проєктування та розробка веб-сайту автоматизації бронювання і організації робочих просторів коворкінгу на основі веб-технологій, що забезпечує цифрове самообслуговування клієнтів та інструменти управління для адміністратора.

Для досягнення цієї мети було вирішено такі завдання:

- аналіз предметної галузі, організаційної структури та бізнес-процесів коворкінг-центру;
- виявлення недоліків існуючої системи управління та обґрунтування необхідності автоматизації;
- огляд та порівняльний аналіз існуючих програмних рішень для автоматизації бронювання;
- проєктування бази даних та архітектури програмного забезпечення;

- реалізація функцій системи та розробка користувацького інтерфейсу;
- тестування розробленого програмного продукту.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальна характеристика коворкінг-центру як об'єкта дослідження

Коворкінг-центр є сучасним форматом організації робочого простору, що поєднує переваги традиційного офісу та гнучкість дистанційної зайнятості [1]. Концепція коворкінгу виникла у Сан-Франциско у 2005 році як реакція на зростаючий попит фрілансерів та незалежних фахівців на обладнані робочі місця поза домашнім середовищем. Від початкових ентузіастських просторів для ІТ-спеціалістів галузь трансформувалась у глобальну індустрію з мільйонами резидентів і десятками тисяч локацій по всьому світу. На відміну від класичної довгострокової оренди офісних приміщень, коворкінг пропонує резидентам повністю оснащені робочі місця без необхідності укладання тривалих орендних угод, придбання меблів чи організації власної ІТ-інфраструктури. Аудиторія таких просторів є надзвичайно різноманітною: від фрілансерів-одинаків і представників творчих професій до стартап-команд, ІТ-компаній та корпоративних резидентів, які шукають тимчасовий майданчик для проведення робочих зустрічей і нарад.

Ключовою характеристикою сучасного коворкінгу є синтез фізичного простору і сервісної складової. Резидент платить не лише за квадратні метри – він отримує доступ до екосистеми: швидкісного інтернету, переговорних кімнат, кухні, зони відпочинку, а нерідко і до спільноти однодумців та регулярних заходів. Саме ця сервісна надбудова відрізняє коворкінг від простої суборенди і обґрунтовує його позиціонування у рамках бізнес-моделі «простір як сервіс» (Space-as-a-Service) [2]. За аналітичними даними, гнучкий формат оренди офісних приміщень стає домінуючою тенденцією на ринку комерційної нерухомості, оскільки організації все частіше обирають гнучкі умови оренди замість довгострокових зобов'язань [3].

Об'єктом дослідження у цій кваліфікаційній роботі є мережа коворкінг-центрів, розташована у місті Київ – одному з найактивніших ринків гнучких

офісних рішень в Україні [4]. Незважаючи на складні зовнішньополітичні умови, ринок коворкінгу в Україні демонструє стійкість та адаптивність: попит на гнучкі формати оренди зростає як серед локальних фахівців, які переходять на дистанційний режим роботи, так і серед міжнародних організацій та НКО, що шукають тимчасові офісні рішення. Мережа охоплює три локації: Поділ, Печерськ та Осокорки, кожна з яких орієнтована на різні мікрорайони та типи резидентів. Такий географічно розподілений формат дозволяє охопити ширшу аудиторію і знизити залежність від завантаженості єдиної точки присутності.

Установа функціонує відповідно до бізнес-моделі «простір як сервіс» (Space-as-a-Service) [3] і орієнтована на широку аудиторію: фрілансерів, IT-фахівців, стартапи та невеликі компанії, які потребують гнучких умов оренди. Головна мета діяльності центру – надати резидентам повністю оснащений і технологічно забезпечений робочий простір з мінімальними адміністративними бар'єрами для доступу та максимальною зручністю користування. Центральним пріоритетом є зниження порогу входу для нових резидентів: відсутність довгострокових зобов'язань, мінімальний пакет документів та миттєва готовність робочого місця до використання.

Інфраструктура коворкінг-центру охоплює три функціональні зони, кожна з яких задовольняє специфічні потреби різних категорій резидентів. По-перше, відкритий простір (Open Space), призначений для індивідуальної роботи: він вміщує робочі місця, обладнані ергономічними меблями, підключенням до швидкісного інтернету та джерелами живлення. Open Space є найбільш демократичним форматом у ціновому відношенні і забезпечує максимальну гнучкість – резидент не прив'язаний до конкретного столу і може обирати місце щодня. По-друге, закріплені столи (Fixed Desk) – персональні робочі місця з постійно призначеним простором. Резидент, який орендує Fixed Desk, отримує гарантоване робоче місце з можливістю залишати на ньому особисті речі та обладнання. Цей формат є оптимальним для фахівців, які регулярно відвідують коворкінг і цінують стабільність та передбачуваність свого робочого середовища.

По-третє, переговорні кімнати (Meeting Room) – ізольовані приміщення, обладнані для проведення нарад, презентацій і відеоконференцій. Вони оснащені великим монітором або проєктором, відеокамерою, мікрофоном та системою відеозв'язку. Переговорні кімнати надаються в погодинну оренду і доступні як для резидентів, так і для зовнішніх клієнтів.

Тарифна система центру передбачає кілька форматів оренди, що охоплюють різні потреби та бюджети резидентів. Для Open Space і Fixed Desk передбачено три тарифних плани: денний, тижневий та місячний. Денна оренда надає доступ до простору протягом одного робочого дня і є оптимальним вибором для нерегулярних відвідувачів або фахівців, що перебувають у місті у відрядженні. Тижневий абонемент забезпечує доступ упродовж семи календарних днів і є вигідним для фахівців, зайнятих проєктною роботою з чіткими строками. Місячна підписка є найекономнішим варіантом із розрахунку на добу і орієнтована на постійних резидентів, для яких коворкінг є основним робочим місцем. Переговорні кімнати орендуються виключно погодинно, що відповідає їхньому призначенню і дозволяє ефективно використовувати обмежений ресурс без надмірних витрат з боку клієнта.

Сфера діяльності коворкінгу охоплює одночасно ринок комерційної нерухомості та ринок цифрових сервісів для бізнесу. Серед ключових особливостей установи слід виділити чітку технологічну орієнтованість: резиденти висувують підвищені вимоги до надійності мережевої інфраструктури, швидкості інтернет-з'єднання та безпеки локальної мережі. Технологічний профіль основної аудиторії (ІТ-фахівці, розробники, дизайнери) визначає специфічні вимоги і до інформаційної системи управління: вона має бути не лише функціональною, а й максимально зручною у використанні, доступною з будь-якого пристрою та здатною опрацьовувати запити в режимі реального часу без затримок.

1.2 Організаційна структура коворкінг-центру, роль і взаємодія підрозділів

1.2.1 Загальна схема організаційної структури коворкінг-центру

Організаційна структура коворкінг-центру побудована за лінійно-функціональним принципом управління [5], що поєднує чітку ієрархічну підпорядкованість із функціональною спеціалізацією кожного підрозділу. На вищому рівні управлінської вертикалі перебуває Директор, який відповідає за стратегічний розвиток установи, контроль фінансових потоків та представлення інтересів організації у відносинах із зовнішніми партнерами. Безпосередньо директору підпорядковані керівники трьох основних підрозділів: адміністративного, технічного забезпечення та господарського.

Адміністративний підрозділ виконує функції фронт-офісу і є центральним об'єктом автоматизації в межах цього проєкту – саме через нього проходять усі клієнтські запити: від первинної реєстрації до завершення бронювання. Відділ технічного забезпечення несе відповідальність за стабільність телекомунікаційної інфраструктури, налаштування та обслуговування обладнання, а також за функціонування системи контролю доступу до приміщень. Господарський відділ забезпечує фізичну готовність просторів: прибирання, підготовку меблів і підтримання загального стану зон у належному вигляді. Схема організаційної структури коворкінг-центру наведена на рис. 1.1.

Взаємодія між підрозділами здійснюється як вертикально – через розпорядження директора, – так і горизонтально – безпосередньо між керівниками відділів у поточному режимі. За відсутності єдиної інформаційної системи така комунікація нерідко супроводжується затримками та дублюванням даних. На схемі організаційної структури адміністративний підрозділ виділений окремо як об'єкт, що підлягає автоматизації. Лінійно-функціональна структура обрана з кількох причин. По-перше, вона відповідає масштабу установи: невелика чисельність персоналу не потребує складних матричних або проєктних структур, а чіткий розподіл ролей між трьома підрозділами охоплює всі ключові напрями операційної

діяльності. По-друге, ця структура забезпечує достатню гнучкість для сервісного бізнесу, де пріоритети і обсяги завдань можуть змінюватись залежно від сезонності та завантаженості. По-третє, вертикальна підпорядкованість дозволяє директору оперативно впроваджувати зміни в роботі будь-якого підрозділу без тривалих узгоджень.

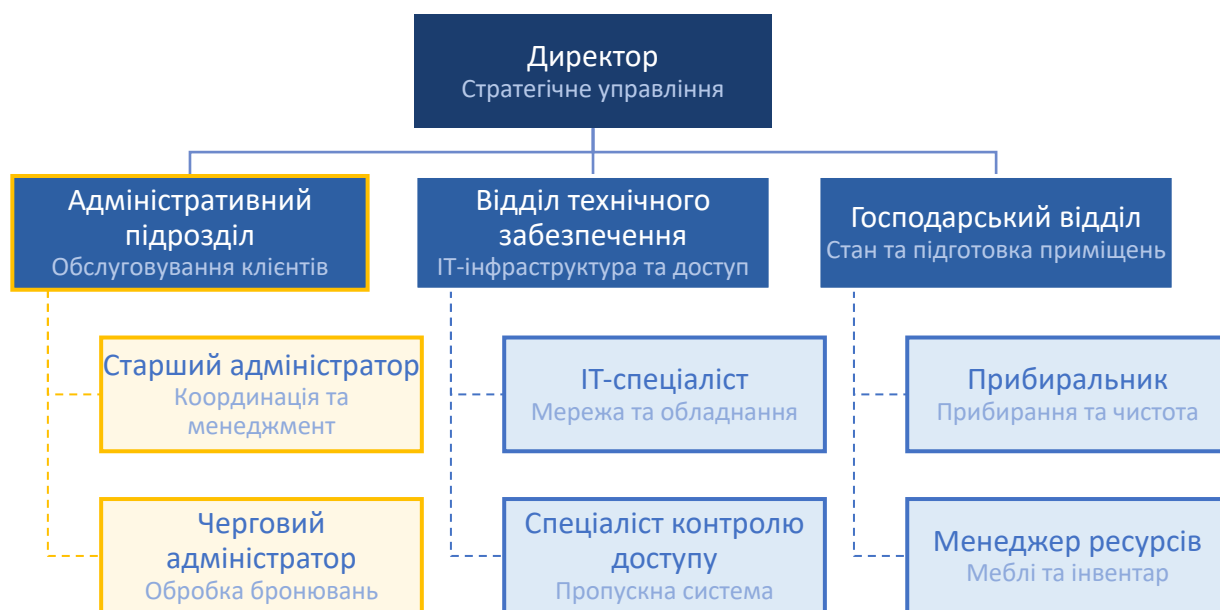


Рисунок 1.1 – Схема організаційної структури коворкінг-центру

Разом з тим поточна структура має характерну для малого бізнесу особливість: межі між підрозділами є рухомими, а навантаження нерівномірне. У піковий час – ранок та вечір буднів – адміністративний підрозділ одночасно опрацьовує кілька вхідних запитів, виконує касові операції і відповідає на телефонні дзвінки, тоді як технічний і господарський підрозділи переважно задіяні у підготовчій роботі. Відсутність єдиної інформаційної системи означає, що координація між підрозділами відбувається ситуативно, а не системно, що безпосередньо обмежує пропускну здатність всієї операційної ланцюжка. Така модель координації є прийнятною за невисокої завантаженості, проте у разі масштабування мережі або зростання кількості одночасних бронювань вона стає критичним вузьким місцем, що унеможливорює ефективне управління без додаткового персоналу чи автоматизованих інструментів.

1.2.2 Структура та функції адміністративного підрозділу

Адміністративний підрозділ коворкінг-центру є ключовою ланкою операційної діяльності установи та виконує роль фронт-офісу – першої та основної точки контакту між організацією та її клієнтами. Підрозділ забезпечує повний цикл обслуговування резидента: від первинного консультування і реєстрації облікового запису до завершення терміну оренди та виходу клієнта з приміщення.

Структурно адміністративний підрозділ складається з двох рівнів. На керівному рівні перебуває старший адміністратор (ком'юніті-менеджер), який здійснює загальну координацію підрозділу, формує розклад чергувань персоналу, взаємодіє з іншими відділами та відповідає перед директором за якість обслуговування і виконання планових показників. На виконавчому рівні працюють чергові адміністратори – фахівці, що безпосередньо обслуговують клієнтів протягом робочого дня: приймають і обробляють заявки на бронювання, здійснюють реєстрацію нових резидентів, контролюють доступ до приміщень і підтримують актуальність даних у системі обліку.

Специфіка роботи підрозділу визначається кількома ключовими чинниками. По-перше, адміністративний персонал одночасно опрацьовує заявки на три різних тарифних плани – денний, тижневий і місячний, – що вимагає чіткого обліку строків і запобігання перетину бронювань. По-друге, підрозділ обслуговує три типи фізичних зон: Open Space, закріплені столи (Fixed Desk) і переговорні кімнати, кожна з яких має власну цінову політику і правила бронювання. По-третє, відсутність наразі автоматизованої системи змушує персонал виконувати більшість операцій вручну – через електронні таблиці, телефон і месенджери, – що суттєво обмежує пропускну спроможність підрозділу і підвищує ризик помилок.

До функціональних обов'язків персоналу адміністративного підрозділу входить широкий спектр операційних завдань. Старший адміністратор відповідає за: стратегічне планування завантаженості ресурсного фонду; управління взаємовідносинами з постійними резидентами; підготовку зведеної звітності для директора щодо фінансових показників і рівня завантаженості; координацію між

технічним та господарським відділами при підготовці місць до заїзду нових клієнтів; вирішення конфліктних і нестандартних ситуацій, пов'язаних із бронюваннями або якістю сервісу.

Чергові адміністратори виконують такі оперативні функції: реєстрація нових клієнтів і внесення їхніх даних до системи обліку; обробка поточних заявок на бронювання (перевірка наявності, підтвердження, фіксація у таблиці); прийом оплати – готівкою або банківським переказом; моніторинг і оновлення статусів завантаженості зон у режимі реального часу; надання клієнтам консультативної підтримки щодо доступних місць, тарифів і умов оренди; контроль своєчасного завершення бронювань і звільнення місць; оповіщення технічного і господарського відділів про необхідність підготовки місця до нового заїзду.

Адміністративний підрозділ активно взаємодіє з усіма структурними підрозділами коворкінг-центру. Взаємодія з технічним відділом відбувається при виникненні несправностей в IT-інфраструктурі та при налаштуванні мережевого доступу для нових резидентів. З господарським відділом адміністратори координують підготовку приміщень до заїзду – прибирання, розстановку меблів, перевірку стану обладнання. З бухгалтерією здійснюється обмін фінансовою інформацією: адміністратор передає дані про виконані платежі та отримує підтвердження проведених транзакцій. Вся ця взаємодія наразі здійснюється у неформальному режимі – переважно через месенджери і телефонні дзвінки, без структурованого документообігу, що є одним із ключових недоліків поточної організації роботи.

Аналіз інформаційних потоків, відображених на рисунку 1.2, свідчить про значну інтенсивність і різноманітність операцій, що виконуються адміністративним підрозділом. Підрозділ виступає центральним вузлом обміну даними між усіма учасниками процесу – клієнтами, суміжними відділами і керівництвом. При цьому переважна більшість вхідних і вихідних потоків наразі не автоматизована: інформація передається усно, через месенджери або вручну вноситься до електронних таблиць. Відсутність стандартизованих форматів і каналів передачі

даних призводить до затримок, дублювання записів і ризику втрати критично важливої інформації. Саме ці недоліки документообігу є головним обґрунтуванням необхідності розробки єдиної інформаційної системи, яка перетворить неформалізовані потоки на структуровані транзакції у межах єдиної цифрової платформи.

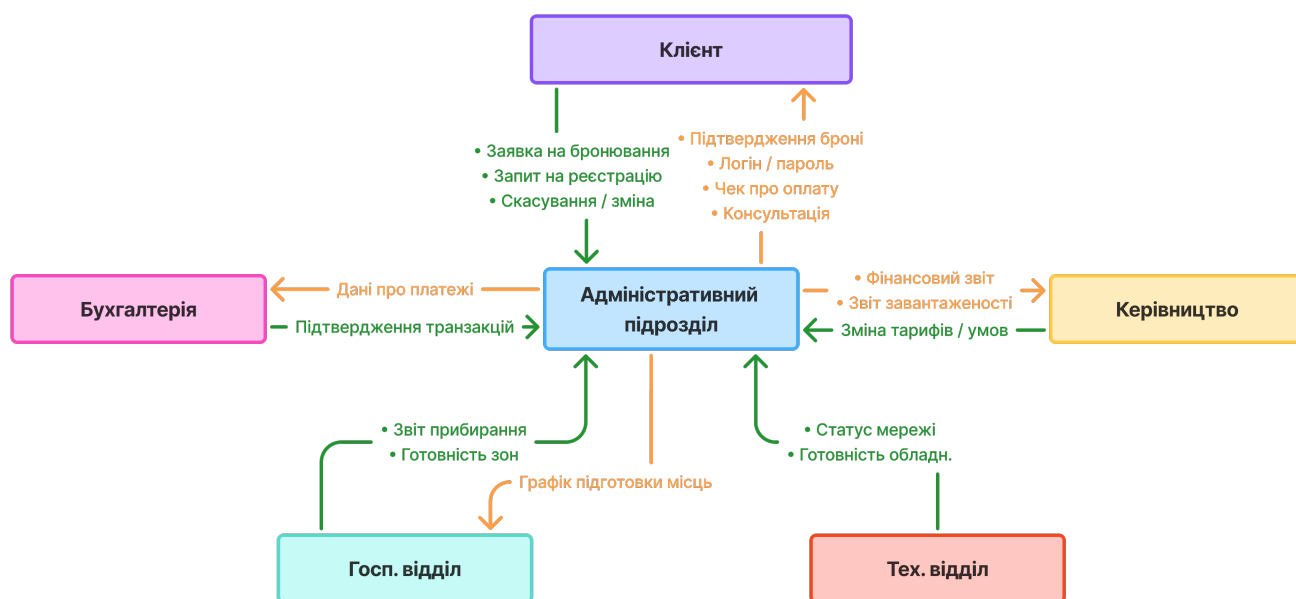


Рисунок 1.2 – Схема інформаційних потоків адміністративного підрозділу коворкінг-центру

1.3 Аналіз нинішнього стану комп'ютеризації установи та обґрунтування розробки

Поточний рівень комп'ютеризації коворкінг-центру характеризується використанням набору загальноцільових хмарних інструментів, що не утворюють єдиної інтегрованої інформаційної системи. Адміністративний підрозділ спирається переважно на сервіси Google Workspace [6]: розклад завантаженості робочих місць ведеться в електронних таблицях Google Sheets [7], збір первинних даних від клієнтів здійснюється через Google Forms [8], а комунікація між персоналом та клієнтами відбувається через корпоративну електронну пошту й месенджери. Оплата бронювань приймається через банківський переказ або готівкою безпосередньо в адміністратора.

Проведений аналіз дозволяє констатувати низку суттєвих недоліків наявного технологічного стеку. Використання електронних таблиць як основного інструменту обліку бронювань є принципово недостатнім: відсутні механізми забезпечення транзакційної цілісності даних, що породжує ризик «подвійного бронювання» – ситуації, коли через затримку ручного оновлення статусів одне й те саме місце опиняється зарезервованим за двома різними користувачами одночасно. Перевірка доступності місць на конкретну дату вимагає постійної участі адміністратора.

Окремою критичною проблемою є повна відсутність зовнішнього цифрового інтерфейсу для клієнтів. На сьогодні коворкінг не має жодного вебресурсу, який дозволяв би потенційному резиденту переглянути актуальну карту приміщень і самостійно здійснити бронювання без участі адміністратора. Усі звернення обробляються виключно персоналом, що суттєво обмежує доступність послуги у неробочий час та знижує конкурентоспроможність установи. Відсутність онлайн-оплати змушує клієнтів вдаватися до банківських переказів або відвідувати центр особисто, що є незручним за умов сучасного темпу ділового життя.

Фінансовий облік здійснюється у напівавтоматичному режимі: адміністратор вручну зіставляє банківські виписки із записами в таблицях, що не лише знижує продуктивність праці, а й систематично створює ризики фінансових розбіжностей. Нарешті, відсутність інтегрованої системи автоматичних сповіщень призводить до несвоєчасного інформування клієнтів про завершення терміну оренди, що негативно позначається на ефективності використання ресурсного фонду. Інформація про клієнтів, бронювання та фінансові операції зберігається в ізольованих інструментах, що унеможлиблює отримання консолідованої аналітики в режимі реального часу.

Таким чином, наявне програмне забезпечення не відповідає актуальним потребам коворкінг-центру з кількох причин: воно не забезпечує єдиного інформаційного середовища; не підтримує автоматичну перевірку конфліктів

бронювань; не надає клієнтам інструментів цифрового самообслуговування; не інтегрується з платіжними сервісами. Більшість критично важливих функцій залишаються нереалізованими, що підтверджує доцільність розробки спеціалізованої інформаційної системи на базі сучасних веб-технологій.

Описані проблеми не є унікальними для досліджуваного коворкінгу – вони характерні для широкого кола підприємств сфери послуг, що перебувають на етапі цифрової трансформації. Відповідно до сучасних підходів до побудови корпоративних інформаційних систем, ефективне управління сервісним бізнесом передбачає наявність єдиного інформаційного простору, в якому всі операційні дані – клієнтські профілі, бронювання, платежі та аналітика – зберігаються в централізованій базі даних і доступні авторизованим учасникам у режимі реального часу. Перехід від фрагментарних хмарних інструментів до спеціалізованої інтегрованої платформи є стандартним кроком у розвитку підприємств, які досягли межі масштабування в рамках ручних процесів. Саме цей перехід є центральним завданням розроблюваної системи.

1.4 Функціональне моделювання та аналіз бізнес-процесів

1.4.1 Функціональна модель AS-IS бізнес-процесів коворкінг-центру

Для формалізації поточного стану бізнес-процесів коворкінг-центру побудовано крос-функціональну діаграму у нотації BPMN (Business Process Model and Notation) [9], що відображає взаємодію між двома учасниками процесу – клієнтом та адміністратором. Діаграма наведена на рисунку 1.3.

Процес бронювання ініціюється клієнтом, у якого виникає потреба орендувати робоче місце. Клієнт самостійно зв'язується з адміністратором телефоном або електронною поштою, надсилаючи запит на бронювання із зазначенням бажаного типу місця та дати. Після цього клієнт переходить у стан очікування відповіді.

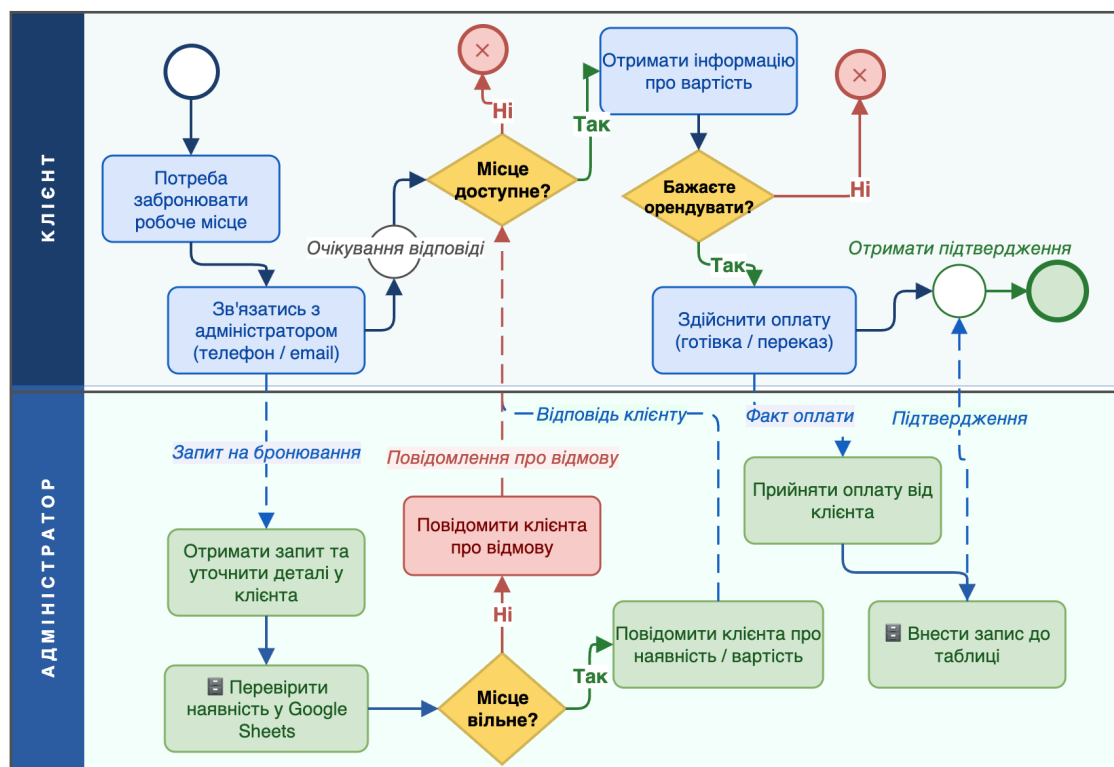


Рисунок 1.3 – Бізнес-процес бронювання робочого місця коворкінг-центру (AS-IS)

Отримавши запит, адміністратор уточнює деталі у клієнта та вручну перевіряє завантаженість у таблиці Google Sheets [7]. За результатом перевірки виникає перше розгалуження: якщо місце зайняте – адміністратор повідомляє клієнта про відмову і процес завершується; якщо місце вільне – адміністратор повідомляє клієнта про наявність і озвучує вартість оренди.

Отримавши відповідь, клієнт приймає рішення. Якщо він не бажає орендувати – процес завершується. Якщо погоджується – здійснює оплату готівкою або банківським переказом, після чого передає адміністратору факт оплати. Адміністратор приймає кошти, вручну вносить запис до таблиці та надсилає клієнту підтвердження бронювання.

Хронологічний аналіз процесу демонструє його залежність від людського фактора на кожному з етапів. У типовому сценарії час від отримання запиту до підтвердження бронювання становить від 15 хвилин до кількох годин – залежно від завантаженості адміністратора та оперативності клієнта щодо оплати. У неробочий

час і вихідні дні обробка запитів повністю зупиняється, що означає повну недоступність сервісу для клієнтів у цей період. Така модель є прийнятною лише за невисокого потоку звернень, однак у разі зростання кількості одночасних запитів вона критично знижує якість обслуговування.

Загалом побудована модель AS IS наочно підтверджує висновки, зроблені у підрозділі 1.3: поточний процес бронювання є повністю ручним, позбавленим механізмів автоматичної перевірки конфліктів і недоступним для клієнтів поза робочим часом адміністратора. Виявлені вразливості охоплюють одночасно три рівні – технологічний (відсутність транзакційного механізму), організаційний (надмірна залежність від одного виконавця) та клієнтський (відсутність самообслуговування), – що свідчить про системний характер проблем і обґрунтовує необхідність комплексного підходу до автоматизації.

1.4.2 Пропозиції щодо усунення наявних проблем та функціональна модель TO BE

Для усунення проблем, виявлених у підрозділах 1.3 та 1.4.1, пропонується розробка спеціалізованого веб-застосунку, що замінить фрагментарний набір ручних інструментів єдиною автоматизованою платформою. Оновлений бізнес-процес бронювання відображено у функціональній моделі TO BE, побудованій у нотації BPMN [9] та наведений на рисунку 1.4.

На відміну від моделі AS IS, де єдиним виконавцем усіх операцій є адміністратор, у моделі TO BE процес розподілено між двома учасниками: клієнтом та веб-застосунком. Адміністратор повністю виведений з операційного потоку бронювання.

Процес розпочинається з того, що клієнт самостійно реєструється або авторизується у системі, після чого переглядає інтерактивну карту поверху з актуальним статусом кожного місця. Обравши бажане місце, дату та тривалість оренди, клієнт ініціює перевірку доступності. Веб-застосунок автоматично

звертається до бази даних і повертає результат у режимі реального часу. При недоступності місця клієнт негайно отримує відповідне повідомлення і процес завершується – без жодної участі персоналу. При наявності місця клієнт переходить до підтвердження деталей бронювання та обраного тарифу.

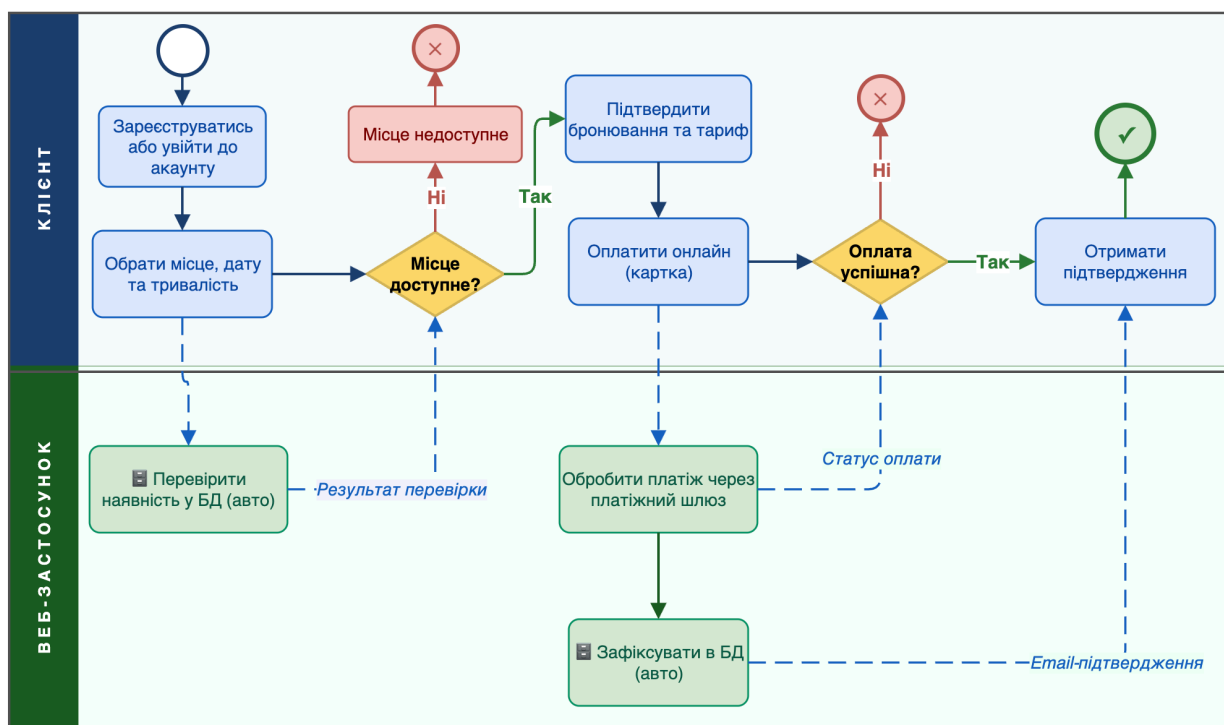


Рисунок 1.4 – Бізнес-процес бронювання робочого місця коворкінг-центру (TO BE)

Після підтвердження клієнт здійснює онлайн-оплату картою. Вебзастосунок передає платіжні дані до платіжного шлюзу та отримує статус транзакції. У разі невдалої оплати клієнт отримує повідомлення про помилку і процес завершується. У разі успішної оплати система транзакційно фіксує бронювання в базі даних – що унеможливорює подвійне резервування – і відображає клієнту детальний чек-підтвердження бронювання. На цьому процес завершується без будь-якого ручного втручання.

Порівняння моделей AS IS та TO BE виявляє принципові відмінності на кожному етапі процесу. У моделі AS IS перевірка доступності виконується адміністратором вручну через Google Sheets [7] і займає від кількох хвилин до кількох годин; у моделі TO BE вона виконується автоматично за долі секунди.

Прийом оплати, який раніше потребував фізичної присутності клієнта або очікування банківського переказу, замінюється миттєвою карткою транзакцією онлайн. Ручне внесення запису до таблиці, що було джерелом помилок і подвійних бронювань, замінюється транзакційним записом до реляційної бази даних. Нарешті, підтвердження, яке раніше адміністратор надсилав вручну, генерується і доставляється клієнту автоматично.

Впровадження розробленої системи відкриває також перспективи для подальшого розвитку. Накопичені дані про завантаженість і поведінку резидентів стануть основою для аналітичного модуля адмін-панелі. Наявність REST API [10] уможлиблює майбутню інтеграцію з корпоративними сервісами бронювання та мобільними застосунками. Модульна архітектура на базі Next.js [11] і Node.js [12] забезпечує горизонтальне масштабування системи у разі розширення мережі коворкінгів.

1.5 Огляд існуючих рішень для автоматизації бронювання в коворкінгах

З метою визначення функціональних вимог до розроблюваної системи та обґрунтування доцільності власної розробки проведено аналіз веб-сайтів трьох діючих коворкінг-центрів, що є найближчими аналогами проекту за форматом та цільовою аудиторією: NAT Coworking, PLATFORMA та Regus [13-15].

NAT Coworking [13] – львівський коворкінг, що позиціонує себе як простір для тих, «кому не байдуже де працювати». Сайт має сучасний мінімалістичний дизайн і повністю адаптований для мобільних пристроїв (рисунки 1.5). Навігація охоплює розділи з описом підписок, переговорних кімнат та офісів. Мова інтерфейсу – українська з перемиканням на англійську. Попри привабливий візуальний стиль, сайт позбавлений ключових функцій цифрового сервісу: відсутнє онлайнбронювання, карта поверху та особистий кабінет. Єдиний спосіб стати резидентом – заповнити контактну форму і чекати відповіді адміністратора. Така

модель взаємодії повністю відповідає описаній у підрозділі 1.3 моделі AS IS і демонструє характерну для малих коворкінгів відсутність цифрової автоматизації.

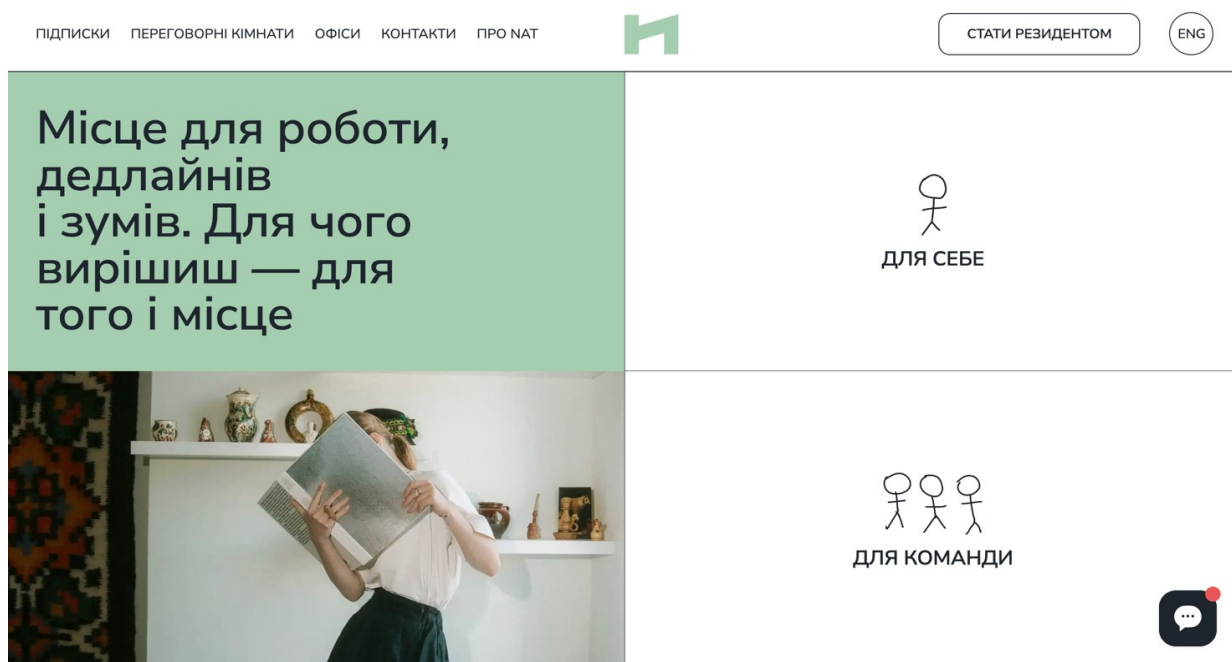


Рисунок 1.5 – Головна сторінка сайту NAT Coworking

PLATFORMA [14] – мережевий коворкінг у Києві, орієнтований на творчих фахівців та IT-команди. Сайт містить детальні описи робочих зон, галерею, розділи про події та партнерські програми. Дизайн є стилістично зрілим і відображає позиціонування бренду. Однак, як і у попередньому прикладі, функціональність обмежена інформаційним форматом: бронювання здійснюється виключно через звернення до менеджера по телефону або email, онлайн-оплата та особистий кабінет відсутні. Інтерфейс сайту PLATFORMA зображено на рисунку 1.6. Карта поверху для вибору конкретного місця не передбачена. Сайт фактично виконує роль цифрової брошури, а не повноцінного сервісного застосунку. Попри якісний дизайн і розвинений контент, відсутність будь-якого механізму самостійного бронювання означає, що клієнт залишається залежним від доступності менеджера так само, як і у випадку з повністю офлайн-сервісом. Це свідчить про те, що інвестиції у дизайн не компенсують відсутності функціональної складової, і підтверджує: для цільової аудиторії сучасного коворкінгу наявність онлайн-сервісу є не перевагою, а базовою очікуваною умовою.

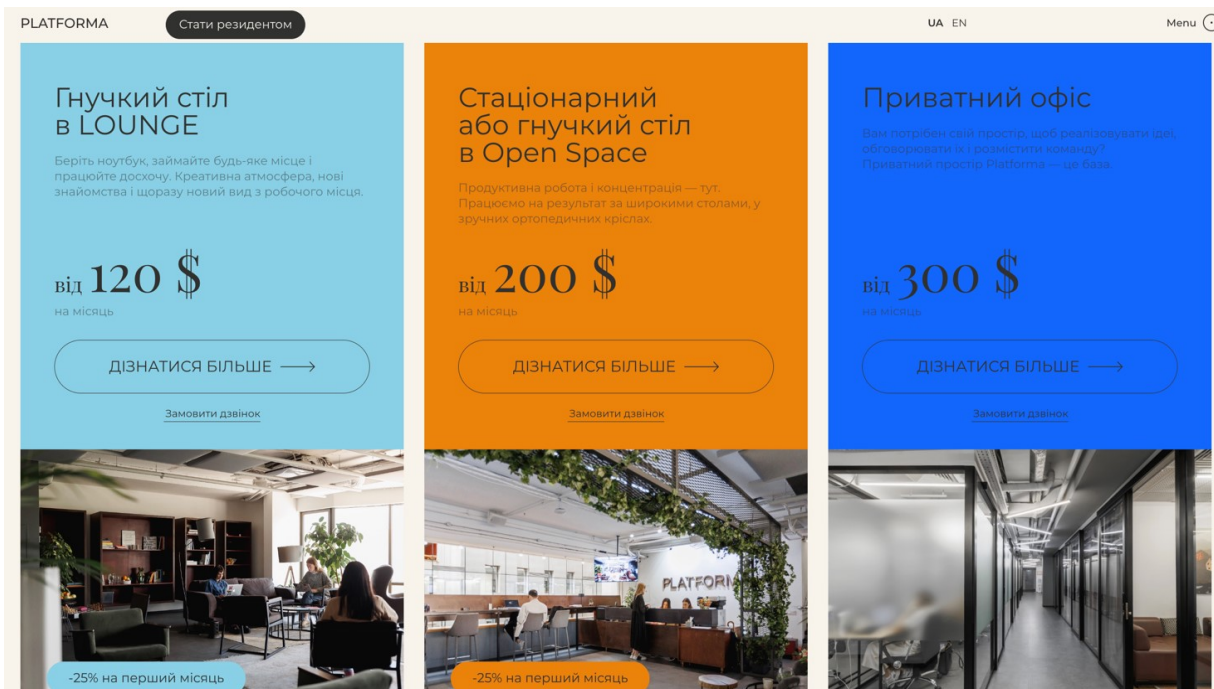


Рисунок 1.6 – Сторінка робочих просторів сайту PLATFORMA

Regus [15] – міжнародна мережа гнучких офісних просторів, що має локації в Києві та по всьому світу. На відміну від двох попередніх прикладів, Regus пропонує повноцінний цифровий сервіс, що наочно відображено на рисунку 1.7: онлайн-бронювання робочих місць і переговорних кімнат безпосередньо на сайті, особистий кабінет резидента, онлайн-оплату, мобільний застосунок. Сайт підтримує десятки мов і валют, має розвинену систему фільтрів за типом місця, датою та тривалістю оренди. Водночас Regus є корпоративною платформою глобального масштабу: інтерфейс перевантажений опціями, орієнтованими на великий бізнес, а сам сайт не передбачає інтерактивної карти конкретного поверху для вибору окремого робочого місця. Крім того, підключення до платформи Regus передбачає проходження корпоративної процедури верифікації та дотримання стандартів глобальної мережі, що є недоцільним і фінансово невиправданим для невеликого локального коворкінгу. Таким чином, попри розвинений функціонал, Regus не може бути використаний як готове рішення для досліджуваного об'єкта: він вирішує задачі іншого масштабу і не дозволяє адаптувати інтерфейс та логіку роботи під специфіку конкретної мережі.

Regus

Домашня сторінка
Команда
Швидкий доступ
Допомога

V h

Мій обліковий запис

VH

Додати компанію

Дані бронювання

Орієнтовне зображення

Lviv, Spaces Lviv Tech City

48 Stryjska Str., Lviv, 79021

пнд 11 трав 2026 р., 09:00 - 19:00

Додати повторення

+ Додати інше часове вікно

Додати їжу, напої та послуги

Необмежена кількість кави та чаю - цілий день

ДОДАТИ →

Необмежена кави та чай - півдня

ДОДАТИ →

Контакт для виставлення ...

Номер замовлення

Збори та платіжні дані

Робочий простір	19,00 USD
Комісія за обслуговування в неробочий час	2,38 USD

Вибраний тип приміщення:

Робоче місце в коворкінгу

Загальна сума за бронювання:

21,38 USD

Додати дані компанії

Рисунок 1.7 – Форма онлайн-бронювання сайту Regus

За результатами огляду трьох сайтів-аналогів складено порівняльну таблицю 1.1, у якій оцінено наявність ключових функцій за критеріями, що є обов'язковими для розроблюваної системи.

Таблиця 1.1 – Порівняльна характеристика сайтів коворкінг-центрів

Критерій	NAT Coworking	PLATFORMA	Regus
Онлайн-бронювання	×	×	✓
Карта поверху / вибір місця	×	×	×
Особистий кабінет	×	×	✓
Онлайн-оплата	×	×	✓
Мобільний застосунок	✓	×	✓
Адаптивний дизайн	✓	✓	✓

Аналіз таблиці 1.1 виявляє характерну закономірність: локальні українські коворкінги мають якісний дизайн, але позбавлені цифрового сервісу, тоді як міжнародні платформи пропонують розвинений функціонал, проте є надмірно складними і фінансово недоступними для невеликого коворкінгу. Жоден із

розглянутих аналогів не реалізує інтерактивну карту поверху для вибору конкретного місця – функцію, яка є однією з ключових у розроблюваній системі. Показово, що навіть Regus [15] – найбільш технологічно розвинена з аналізованих платформ – не надає клієнту можливості візуально обрати конкретний стіл на плані приміщення, обмежуючись лише вибором локації та типу місця. Це свідчить про те, що зазначена функція є не лише конкурентною перевагою розроблюваної системи, а й реальною прогалиною у пропозиції ринку загалом. Таким чином, результати порівняльного аналізу підтверджують наявність незайнятої ніші та доцільність власної розробки спеціалізованого веб-застосунку, адаптованого до потреб локального коворкінгу.

1.6 Обґрунтування доцільності проєктування та розроблення системи автоматизації бронювання і організації робочих просторів коворкінгу

Проведений аналіз дозволяє сформулювати обґрунтовані висновки щодо необхідності розробки спеціалізованого веб-застосунку для автоматизації управління коворкінг-центром.

Аналіз нинішнього стану комп'ютеризації установи засвідчив, що поточний інструментарій – електронні таблиці Google Sheets [7], контактні форми та месенджери – не утворює єдиної інформаційної системи і не відповідає базовим вимогам до сучасного цифрового сервісу. Відсутність централізованої бази даних унеможливорює транзакційну перевірку доступності місць, що систематично створює ризик подвійного бронювання. Прийом оплати здійснюється через банківський переказ, а клієнт позбавлений будь-якого інструменту самообслуговування поза робочим часом адміністратора.

Функціональне моделювання поточного стану бізнес-процесів [9] підтвердило, що адміністратор є єдиним «шлюзом» між клієнтом і системою на всіх етапах – від перевірки доступності до надсилання підтвердження. Це робить

операційну ефективність установи прямо залежною від людського фактора, обмежує пропускну здатність сервісу і унеможливлює його масштабування без залучення додаткового персоналу. Виявлений перелік проблем охоплює технологічний, організаційний та клієнтський рівні, що свідчить про системний, а не локальний характер недоліків.

Огляд сайтів-аналогів показав, що жодне з розглянутих рішень не задовольняє повному переліку вимог одночасно. Локальні українські коворкінги – NAT Coworking [13] та PLATFORMA [14] – мають якісний дизайн, однак функціонують виключно як інформаційні ресурси без можливості онлайн-бронювання, оплати чи перегляду карти поверху. Міжнародна платформа Regus [15] пропонує розвинений цифровий функціонал, але є корпоративним рішенням глобального масштабу, недоступним фінансово і надмірно складним для невеликого локального коворкінгу. Можливість візуального вибору робочого місця на інтерактивній карті поверху залишається нереалізованою в жодному з аналізованих рішень, що вказує на наявний попит, не задоволений існуючими продуктами.

Окремим аргументом на користь власної розробки є питання технологічної незалежності. Використання готових SaaS-платформ для управління коворкінгом, як-от Nexodus [16], як правило, передбачає щомісячну абонентську плату, обмежені можливості кастомізації та залежність від стороннього постачальника у питаннях підтримки та розвитку функціоналу. Власна система, розроблена на відкритих технологіях, позбавлена цих обмежень: вона може бути адаптована під будь-які специфічні вимоги бізнесу, розширена новими модулями без узгодження з постачальником і розгорнута на будь-якій хмарній інфраструктурі за вибором власника.

Не менш важливим є стратегічний аспект: накопичені власною системою дані про поведінку резидентів, завантаженість просторів і фінансові потоки є цінним активом для прийняття управлінських рішень. Можливість аналізувати ці дані в режимі реального часу через адміністративну панель дозволяє оперативно

коригувати тарифну політику, планувати розширення мережі та підвищувати якість сервісу на основі реальних даних, а не суб'єктивних оцінок персоналу.

Таким чином, результати аналізу, проведеного у підрозділах 1.3, 1.4.1 та 1.5, однозначно підтверджують доцільність проектування та розроблення власної системи автоматизації бронювання і організації робочих просторів коворкінгу на основі веб-технологій. Розробка на базі React/Next.js [11] та Node.js [12] дозволить реалізувати необхідний функціональний набір – онлайн-бронювання з інтерактивною картою поверху, особистий кабінет резидента та адміністративну панель – без надлишкових витрат і залежності від зовнішніх SaaS-постачальників. Це забезпечить підвищення операційної ефективності коворкінг-центру, покращення клієнтського досвіду та створить технологічну основу для подальшого масштабування мережі.

1.7 Технічне завдання на проектування

1.7.1 Мета, цілі створення системи та характеристика об'єкта автоматизації

Метою розробки є підвищення операційної ефективності коворкінг-центру та якості клієнтського досвіду шляхом впровадження єдиної цифрової платформи, що автоматизує процеси бронювання робочих місць, управління ресурсами та обробки платежів. Досягнення цієї мети передбачає перехід від фрагментарного набору ручних інструментів до інтегрованої інформаційної системи, яка охоплює весь операційний цикл – від першого звернення клієнта до формування аналітичної звітності для адміністратора.

Для досягнення поставленої мети визначено такі цілі створення системи:

- усунення ризику подвійного бронювання шляхом запровадження транзакційного механізму резервування місць у реляційній базі даних;

- надання клієнтам цифрового інструменту самообслуговування, що забезпечує можливість бронювання, оплати та управління резервуваннями цілодобово без участі адміністратора;
- реалізація інтерактивної карти поверху з відображенням актуального статусу кожного робочого місця в режимі реального часу;
- автоматизація процесу підтвердження бронювання та відображення клієнту детального чеку із зазначенням місця, дати та суми оренди;
- забезпечення адміністратора панеллю управління ресурсами, що дозволяє в режимі реального часу відстежувати завантаженість просторів і керувати каталогом робочих місць;
- формування цифрового профілю резидента з повною історією бронювань та платежів;
- скорочення операційного навантаження на адміністративний персонал шляхом переведення рутинних операцій у автоматизований режим.

Кожна з визначених цілей безпосередньо впливає з проблем, виявлених у попередніх підрозділах. Усунення ризику подвійного бронювання адресує ключову технологічну вразливість поточного процесу – відсутність механізму блокування запису на час його підтвердження. Цифровий інструмент самообслуговування усуває організаційну вразливість – залежність доступності сервісу від графіку роботи адміністратора. Інтерактивна карта поверху реалізує функцію, відсутню у всіх розглянутих аналогах, і формує ключову конкурентну перевагу системи. Панель адміністратора закриває потребу в консолідованій аналітиці, яка наразі недоступна через ізольованість даних у різних інструментах.

Об'єктом автоматизації є бізнес-процеси адміністративного підрозділу коворкінг-центру, зокрема: обробка заявок на бронювання, перевірка доступності робочих місць, прийом оплати та надання підтвердження клієнту. Вибір саме цього підрозділу як об'єкта автоматизації обумовлений результатами аналізу організаційної структури: адміністративний підрозділ є центральним вузлом усіх

інформаційних потоків і водночас найбільш навантаженою ланкою операційного процесу.

Коворкінг-центр управляє трьома типами фізичних просторів: Open Space з гнучкими робочими місцями, закріпленими столами (Fixed Desk) для постійних резидентів і переговорними кімнатами. Кожен тип простору має власну цінову політику і правила бронювання. Тарифна система передбачає три формати оренди: денний, тижневий та місячний, а для переговорних кімнат – погодинний. Цільова аудиторія є змішаною і включає фрілансерів, IT-фахівців, представників малого бізнесу та корпоративних клієнтів. Ця різноманітність вимагає від системи гнучкості: різні типи місць і тарифів мають опрацьовуватись єдиним уніфікованим механізмом без необхідності ручного налаштування для кожного випадку.

До складу системи, що підлягає автоматизації, входять такі підсистеми: реєстрація та автентифікація користувачів; перегляд та фільтрування доступних робочих місць; резервування обраного місця з перевіркою доступності; обробка онлайн-платежів; управління бронюваннями у особистому кабінеті резидента; адміністрування ресурсного фонду та перегляд аналітики завантаженості.

Важливо чітко визначити межі автоматизації. До сфери автоматизації належать усі цифрові операції між клієнтом і системою, а також між адміністратором і системою. Поза межами автоматизації залишаються: фізична підготовка приміщень до заїзду резидента, обслуговування обладнання, управління персоналом і формування фінансової звітності для зовнішніх органів.

Очікуваний ефект від впровадження системи охоплює три рівні. На технологічному рівні – усунення ризику подвійного бронювання і забезпечення цілісності даних. На організаційному рівні – звільнення адміністративного персоналу від рутинних операцій і перенаправлення його зусиль на роботу з клієнтами та розвиток спільноти. На клієнтському рівні – цілодобова доступність сервісу, прозорість процесу бронювання і зручний особистий.

1.7.2 Функції системи

Функціональні вимоги до системи сформовані на основі аналізу бізнес-процесів і систематизовані у вигляді діаграми варіантів використання (рисунок 1.8).

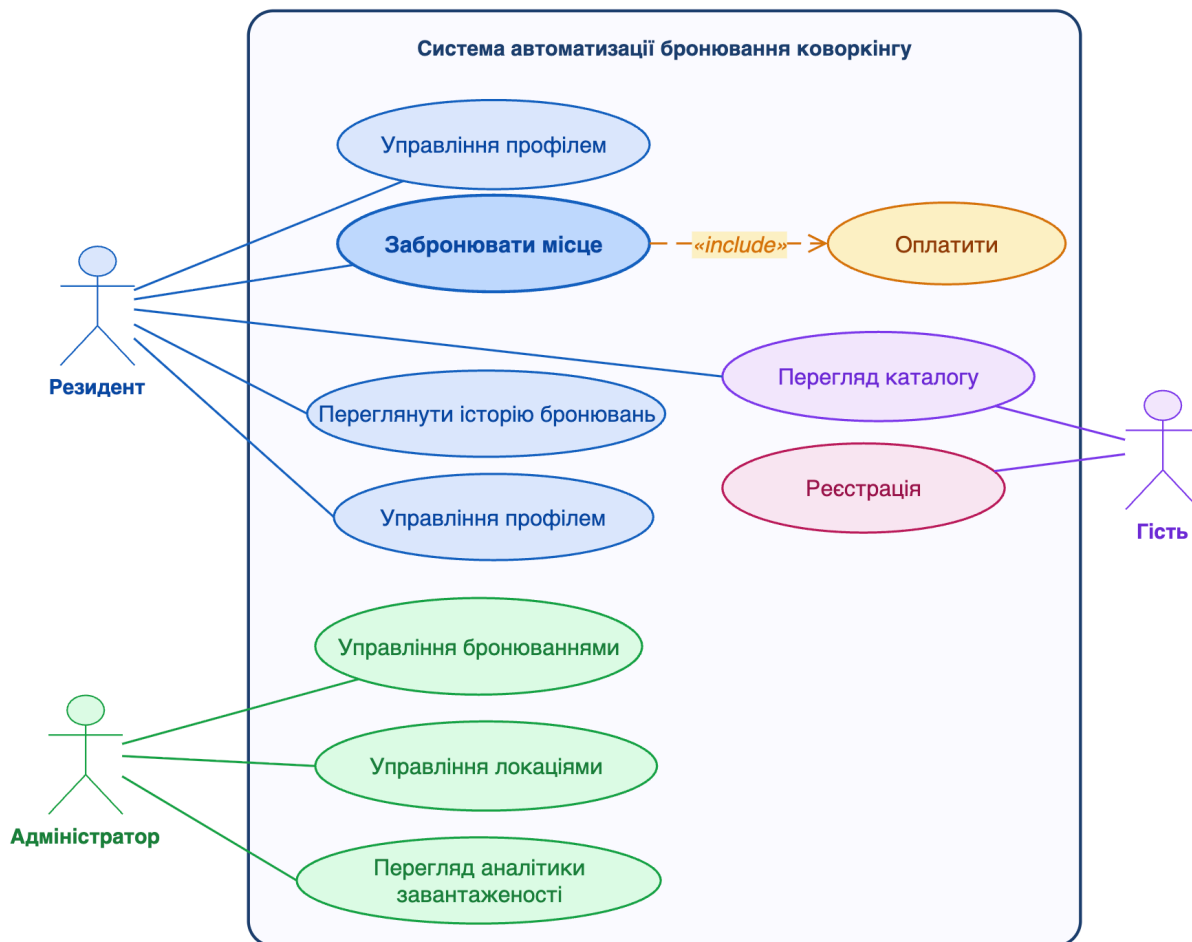


Рисунок 1.8 – Діаграма варіантів використання системи (Use Case Diagram)

Діаграма варіантів використання визначає три ролі в системі: Гість, Резидент та Адміністратор. Гість має доступ до базових функцій перегляду каталогу та реєстрації. Резидент, успадковуючи права гостя, отримує повний набір сервісних функцій. Адміністратор управляє ресурсами і бронюваннями через окремий захищений інтерфейс.

Система реалізує такі функціональні модулі:

- *Модуль автентифікації та авторизації* забезпечує реєстрацію нового користувача з валідацією введених даних, вхід до облікового запису з видачею JWT-

токену [17], відновлення паролю через email, а також розмежування прав доступу між роллю «Резидент» і роллю «Адміністратор». Несанкціонований доступ до захищених маршрутів блокується на рівні серверного middleware.

– *Модуль перегляду та вибору місця* реалізовано у форматі покрокового флоу. Для закріплених столів (Fixed Desk) реалізовано інтерактивну SVG-карту поверху з кольоровою індикацією статусу кожного місця: вільне, зайняте або вибране. Для Open Space та переговорних кімнат вибір здійснюється через картки з описом та характеристиками.

– *Модуль бронювання* забезпечує резервування обраного місця з миттєвою автоматичною перевіркою доступності у базі даних. Перевірка виконується транзакційно, що унеможливорює паралельне підтвердження одного місця двом різним користувачам. Після успішної перевірки бронювання фіксується в базі даних зі статусом «підтверджено».

– *Модуль оплати* реалізує інтеграцію з платіжним шлюзом для обробки карткових транзакцій. Після успішного платежу статус бронювання автоматично змінюється на «confirmed», система генерує електронний квиток і надсилає його на email резидента. У разі невдалої транзакції бронювання скасовується і місце знову стає доступним.

– *Особистий кабінет резидента* відображає список активних і завершених бронювань, дозволяє скасувати бронювання відповідно до встановлених правил, переглянути історію платежів та редагувати персональні дані профілю.

– *Адміністративна панель* забезпечує управління каталогом робочих місць: редагування та зміна статусу місця. Адміністратор має доступ до зведеного списку всіх бронювань з можливістю фільтрування за статусом та локацією. Реалізований модуль аналітики завантаженості відображає статистику використання ресурсів у вигляді агрегованих звітів.

1.7.3 Вимоги до інформаційного забезпечення

Інформаційна архітектура системи описана діаграмою потоків даних (DFD) (рисунок 1.9). Вона деталізує рух інформаційних масивів між чотирма основними процесами та трьома сховищами даних.

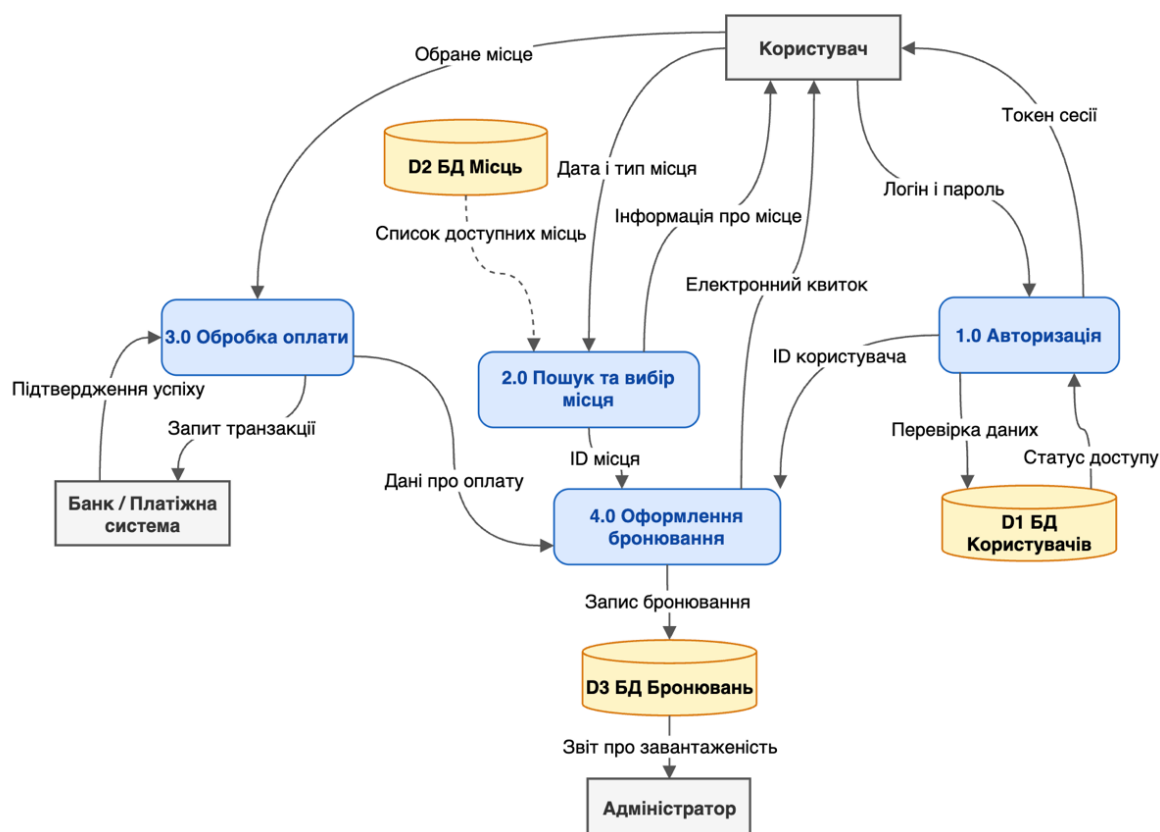


Рисунок 1.9 – Діаграма потоків даних системи (DFD)

Процес 1.0 («Авторизація») отримує облікові дані від користувача, перевіряє їх у сховищі D1 (БД Користувачів) і видає токен сесії для подальшої роботи із системою.

Процес 2.0 («Пошук та вибір місця») обробляє запит щодо дати і типу місця, звертається до сховища D2 (БД Місць) і повертає користувачу список доступних варіантів з актуальним статусом.

Взаємодія з банківською та платіжною системою відбувається у процесі 3.0 («Обробка оплати»), де формується транзакційний запит і отримується підтвердження успішного платежу.

Завершальний процес 4.0 («Оформлення бронювання») консолідує ідентифікатори місця і користувача та підтвердження оплати, транзакційно записує бронювання до сховища D3 (БД Бронювань) і генерує електронний квиток. Адміністратор отримує звіт про завантаженість безпосередньо зі сховища D3.

Таким чином, діаграма потоків даних наочно відображає розподіл відповідальності між компонентами системи та забезпечує повне охоплення всіх інформаційних взаємодій між зовнішніми сутностями, процесами і сховищами даних.

До складу інформаційного забезпечення системи входять такі структури даних: профіль користувача (ідентифікатор, ім'я, email, роль, хешований пароль); робоче місце (ідентифікатор, номер, тип, координати на карті, статус, тарифи); бронювання (ідентифікатор, посилання на користувача і місце, дата початку і завершення, тариф, статус, загальна сума).

Між структурами даних визначено такі зв'язки: один користувач може мати довільну кількість бронювань, тоді як кожне бронювання належить рівно одному користувачу і прив'язане рівно до одного робочого місця. Робоче місце може фігурувати у багатьох бронюваннях, але не може бути одночасно заброньоване двічі на той самий часовий інтервал – ця обмеження забезпечується на рівні транзакційної перевірки при записі до бази даних. Поле «статус» у сутності бронювання приймає одне з визначених значень: CONFIRMED, CANCELLED або COMPLETED, що дозволяє однозначно відстежувати стан кожного резервування протягом усього його життєвого циклу.

Вимоги до якості інформаційного забезпечення охоплюють три аспекти. По-перше, цілісність даних: система не повинна допускати стану, за якого бронювання існує без прив'язки до користувача або місця. По-друге, актуальність: статус доступності місця має відображатися в інтерфейсі в режимі реального часу – без необхідності ручного оновлення сторінки. По-третє, конфіденційність: паролі користувачів зберігаються виключно у вигляді хеш-значень і ніколи не передаються у відкритому вигляді ні між клієнтом і сервером, ні між сервером і базою даних.

1.7.4 Вимоги до програмного забезпечення

Система реалізується на основі трирівневої клієнт-серверної архітектури.

Клієнтська частина (Frontend) розробляється з використанням бібліотеки React.js [18] та фреймворку Next.js [11] версії 14 або вище. Next.js використовується у режимі статичного експорту (Static Export), що дозволяє генерувати готові HTML-файли на етапі збірки та розгортати їх без необхідності серверного рендерингу. Маршрутизація між сторінками відбувається без перезавантаження відповідно до SPA-підходу. Управління станом застосунку реалізується засобами бібліотеки Zustand [19]. Для стилізації компонентів застосовується комбінація Tailwind CSS [20] та SCSS-модулів [21]. Tailwind CSS забезпечує базові утиліти та адаптивну сітку, тоді як локальні SCSS-модулі використовуються для компонентно-специфічних стилів, що гарантує ізоляцію стилів і запобігає конфліктам між компонентами. Інтерактивна карта поверху реалізується засобами SVG з динамічним оновленням статусів через REST API [10].

Серверна частина (Backend) розробляється на платформі Node.js [12] версії 18 LTS або вище. Серверна частина реалізує REST API архітектуру: кожен ресурс системи (користувачі, місця, бронювання) має власний набір маршрутів з відповідними HTTP-методами (GET, POST, PUT, DELETE). Автентифікація і авторизація реалізуються на основі JSON Web Tokens (JWT) [17]: після успішного входу сервер видає підписаний токен з терміном дії, який клієнт передає у заголовок кожного захищеного запиту. Валідація вхідних даних здійснюється бібліотекою Zod [22].

База даних реалізується на основі PostgreSQL [23] версії 15 або вище. Взаємодія з базою даних відбувається через ORM Prisma [24], що забезпечує типізацію запитів, автоматичну міграцію схеми та захист від SQL-ін'єкцій. Транзакційний механізм PostgreSQL використовується при операціях бронювання для гарантування цілісності даних.

Модуль оплати реалізовано у демонстраційному режимі: після підтвердження бронювання система автоматично фіксує замовлення в базі даних зі статусом

«підтверджено» та розраховує загальну суму відповідно до обраного тарифного плану. Архітектура модуля спроектована з урахуванням подальшого підключення платіжного шлюзу Stripe [25] для обробки реальних карткових транзакцій. Для надсилання email-підтверджень передбачено інтеграцію з NodeMailer [26] через SMTP-сервер, яка може бути активована після налаштування поштового провайдера.

Середовище розгортання: клієнтська частина розгортається на платформі Vercel [27], серверна – на Railway [28]. Vercel забезпечує безкоштовний хостинг статичних файлів із підтримкою власного домену та SSL-сертифіката.

Наскрізною мовою програмування для обох частин системи обрано TypeScript [29]. Статична типізація дозволяє виявляти помилки на етапі компіляції, спрощує рефакторинг і забезпечує автодоповнення у середовищах розробки, що підвищує надійність та підтримуваність кодової бази. Спільна мова на фронтенді і бекенді дозволяє повторно використовувати типи даних, зокрема типи відповідей API, що мінімізує неузгодженість між клієнтом і сервером.

Вибір зазначеного технологічного стеку обумовлений кількома ключовими міркуваннями. По-перше, усі перелічені технології є відкритими і безкоштовними, що виключає витрати на ліцензування. По-друге, Next.js і Node.js мають широкую екосистему пакетів, активну спільноту та розвинену документацію, що знижує ризики розробки. По-третє, обраний стек дозволяє розгорнути систему на безкоштовних хмарних платформах без залежності від дорогої інфраструктури, що є критично важливим для малого бізнесу на початковому етапі цифрової трансформації. По-четверте, використання TypeScript як єдиної мови програмування для фронтенду і бекенду забезпечує уніфікованість кодової бази, спрощує командну розробку і дозволяє повторно використовувати типи даних між частинами системи без додаткових конвертацій. По-п'яте, усі обрані інструменти активно підтримуються та регулярно оновлюються, що гарантує довгострокову сумісність і захист від відомих вразливостей на рівні залежностей.

1.7.5 Вимоги до інтерфейсу користувача, надійності та безпеки

Веб-застосунок повинен коректно відображатися у сучасних браузерях – Google Chrome, Mozilla Firefox, Microsoft Edge та Safari – без встановлення додаткових розширень або плагінів. Мінімальна підтримувана версія браузера визначається підтримкою стандарту ES2020 [30].

Інтерфейс повинен відповідати принципам адаптивного дизайну [20] і забезпечувати зручну роботу на екранах з роздільною здатністю від 320 px (смартфони) до 2560 px (широкоформатні монітори). Компонентна сітка будується на базі двох контрольних точок: мобільний (до 768 px) і десктоп (понад 768 px). На мобільних пристроях навігаційне меню трансформується у компактний формат, картки локацій перебудовуються у вертикальний список, а форми займають всю ширину екрана – без горизонтального прокручування.

Навігація між розділами застосунку реалізується без повного перезавантаження сторінки відповідно до SPA-підходу, що забезпечує швидкий відгук інтерфейсу. Час відгуку на дії користувача не повинен перевищувати 200 мс для локальних операцій і 2 секунди для запитів до сервера. Під час виконання асинхронних операцій (запити до API, збереження даних) інтерфейс повинен відображати індикатори завантаження: кнопки переходять у стан «disabled» з текстом-індикатором, що унеможливорює повторне надсилання форми і інформує користувача про хід виконання операції.

Мова інтерфейсу – англійська. Усі повідомлення про помилки та підтвердження мають бути чіткими та інформативними: користувач повинен розуміти причину помилки і знати, які дії необхідно виконати для її усунення. Повідомлення про помилки валідації форм відображаються безпосередньо під відповідним полем – не у вигляді загального попередження. Інтерактивна карта поверху повинна відображати місця з кольоровою індикацією статусу (вільне, зайняте, вибране) та реагувати на взаємодію (наведення, клік) з відображенням детальної інформації.

Кольорова схема інтерфейсу та типографіка обираються з урахуванням достатнього рівня контрастності між текстом і фоном. Усі інтерактивні елементи – кнопки, посилання, поля введення – повинні мати чітко виражені стани: звичайний, при наведенні, активний та недоступний. Покроковий флоу бронювання супроводжується індикатором прогресу (breadcrumb або step bar), що дозволяє користувачу розуміти, на якому етапі він перебуває і скільки кроків залишилось.

Система повинна забезпечувати доступність сервісу на рівні не менше 99% часу упродовж місяця (uptime). Усі дані між клієнтом і сервером передаються виключно через захищений протокол HTTPS з використанням TLS версії 1.2 або вище.

Паролі користувачів зберігаються у базі даних виключно у хешованому вигляді з використанням алгоритму bcrypt із коефіцієнтом складності не менше 10. Вхідні дані на стороні сервера підлягають обов'язковій валідації та санітизації засобами бібліотеки Zod [22] для запобігання атакам типу SQL-ін'єкція та XSS (міжсайтовий скриптинг). Захист від CSRF-атак забезпечується використанням JWT-токенів [17] у заголовку запиту замість cookie-сесій.

Транзакційний механізм бронювання реалізується з використанням рівня ізоляції «Serializable» або «Repeatable Read» у PostgreSQL [23], що повністю виключає виникнення «фантомного читання» та паралельного подвійного бронювання. Для захисту від автоматизованих атак на форму входу реалізовується обмеження кількості невдалих спроб автентифікації (rate limiting) на рівні серверного middleware – не більше п'яти невдалих спроб упродовж десяти хвилин з однієї IP-адреси. Відповідно до рекомендацій OWASP [31], конфігурація HTTP-заголовків безпеки включає Content-Security-Policy, X-Frame-Options та Strict-Transport-Security для захисту від поширених векторів атак на веб-застосунки.

Доступ до маршрутів адміністративної панелі обмежений перевіркою ролі у JWT-токені на рівні серверного middleware. Будь-яка спроба звернення до захищених маршрутів без дійсного токена або з токеном іншої ролі повертає відповідь зі статусом 401 (Unauthorized) або 403 (Forbidden) відповідно.

Висновки до розділу:

У першому розділі проведено аналіз предметної галузі та сформовано основу для розробки системи автоматизації бронювання коворкінгу. Досліджено організаційну структуру мережі коворкінг-центрів та визначено адміністративний підрозділ як головний об'єкт автоматизації. Аналіз поточного стану комп'ютеризації виявив системні недоліки: відсутність єдиної інформаційної системи, ризик подвійного бронювання та відсутність цифрового інтерфейсу для клієнтів. Функціональне моделювання у нотації BPMN підтвердило надмірну залежність операційного процесу від людського фактора та обґрунтувало необхідність переходу до автоматизованої моделі TO BE. Огляд аналогів засвідчив відсутність на ринку готового рішення з повним набором необхідних функцій, що підтверджує доцільність власної розробки.

РОЗДІЛ 2

ПРОЄКТУВАННЯ, СТВОРЕННЯ ТА АПРОБАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис та обґрунтування вибору програмно-технічних засобів розроблення програмного продукту

Вибір програмно-технічних засобів здійснювався з урахуванням функціональних вимог до системи, визначених у технічному завданні, а також з огляду на продуктивність, масштабованість і простоту подальшої підтримки розробленого продукту. Система реалізована на основі трирівневої архітектури: клієнтська частина, серверна частина та база даних – кожна з яких використовує окремий технологічний стек.

Клієнтська частина:

Основою клієнтської частини є бібліотека React 19[18] у поєднанні з фреймворком Next.js 16 [11]. React забезпечує компонентний підхід до побудови інтерфейсу: кожен елемент сторінки реалізується як окремий повторно використовуваний компонент з власним станом і логікою відображення. Next.js розширює можливості React, додаючи файлову маршрутизацію на основі структури директорій, що спрощує організацію коду та навігацію між сторінками без перезавантаження. У даному проєкті Next.js використовується у режимі статичного експорту, що дозволяє згенерувати готові HTML-файли на етапі збірки та розгорнути їх на Vercel [27] без необхідності серверної інфраструктури для фронтенду.

TypeScript [29] використовується як на клієнтській, так і на серверній частині. Статична типізація дозволяє виявляти помилки на етапі розробки, а не під час виконання програми, що суттєво підвищує надійність коду та спрощує його підтримку. Особливо це відчутно при роботі з даними, що передаються між

клієнтом і сервером: типізація API-відповідей запобігає помилкам, пов'язаним із невідповідністю структур даних.

Для управління глобальним станом застосунку обрано бібліотеку Zustand [19]. На відміну від Redux, Zustand має мінімалістичний API і не вимагає написання великої кількості шаблонного коду. У проєкті Zustand використовується для зберігання даних авторизованого користувача (токен, роль, профіль) та синхронізації стану між компонентами.

Стилізація компонентів реалізована за допомогою SCSS-модулів [21] у поєднанні з Tailwind CSS 4 [20]. SCSS-модулі (CSS Modules) забезпечують локальну область видимості стилів: кожен компонент має власний файл `.module.scss`, що запобігає конфліктам класів між різними частинами застосунку. Tailwind CSS використовується для базових утиліт та адаптивної сітки, прискорюючи розробку типових елементів інтерфейсу.

Серверна частина:

Серверна частина розроблена на платформі Node.js [12] з використанням фреймворку Express.js [32]. Node.js забезпечує асинхронну обробку запитів на основі події, що робить його ефективним для застосунків з великою кількістю одночасних з'єднань. Express.js є мінімалістичним і гнучким фреймворком, що дозволяє будувати REST API з чіткою маршрутизацією та middleware-архітектурою. Серверна частина реалізує п'ять груп маршрутів: автентифікація, користувачі, робочі місця, бронювання та адміністрування.

Автентифікація реалізована на основі JSON Web Tokens (JWT) [17]. Після успішного входу сервер генерує підписаний токен, який клієнт зберігає локально і передає у заголовок кожного наступного запиту. Такий підхід є stateless – сервер не зберігає сесії, що спрощує масштабування. Паролі користувачів хешуються за допомогою алгоритму bcrypt з коефіцієнтом складності 10, що забезпечує захист даних навіть у разі витоку бази даних.

Валідація вхідних даних здійснюється бібліотекою Zod [22]. Zod дозволяє описувати схеми очікуваних даних і автоматично генерувати TypeScript-типи з цих

схем, що забезпечує узгодженість між валідацією та типізацією без дублювання коду.

База даних:

Як система управління базами даних обрано PostgreSQL [23] – реляційну СУБД з відкритим вихідним кодом, яка підтримує транзакції з різними рівнями ізоляції, що є критично важливим для коректної роботи механізму бронювання. Зокрема, транзакційний запис бронювання виключає можливість одночасного резервування одного місця двома різними користувачами.

Взаємодія з базою даних реалізована через ORM Prisma [24]. Prisma автоматично генерує типізований клієнт на основі схеми бази даних, що дозволяє працювати з даними у типобезпечний спосіб і отримувати підказки в редакторі коду. Міграції схеми керуються засобами Prisma Migrate, що забезпечує версіонування структури бази даних.

Середовище розгортання:

Клієнтська частина розгортається на Vercel [27]. Vercel є безкоштовним хостингом статичних файлів з підтримкою SSL-сертифіката, що відповідає вимогам безпеки до передачі даних. Серверна частина розгортається на хмарній платформі Railway [28], яка підтримує автоматичне розгортання при пуші до репозиторію та надає керовану базу даних PostgreSQL.

2.2 Проектування та створення бази даних

База даних є центральним компонентом системи, що забезпечує зберігання, цілісність та ефективний доступ до всіх операційних даних. Структура бази даних спроектована відповідно до предметної галузі та функціональних вимог, визначених у технічному завданні. Як СУБД використовується PostgreSQL [23], взаємодія з якою здійснюється через ORM Prisma [24]. Структурне представлення бази даних наведено на рисунку 2.1.

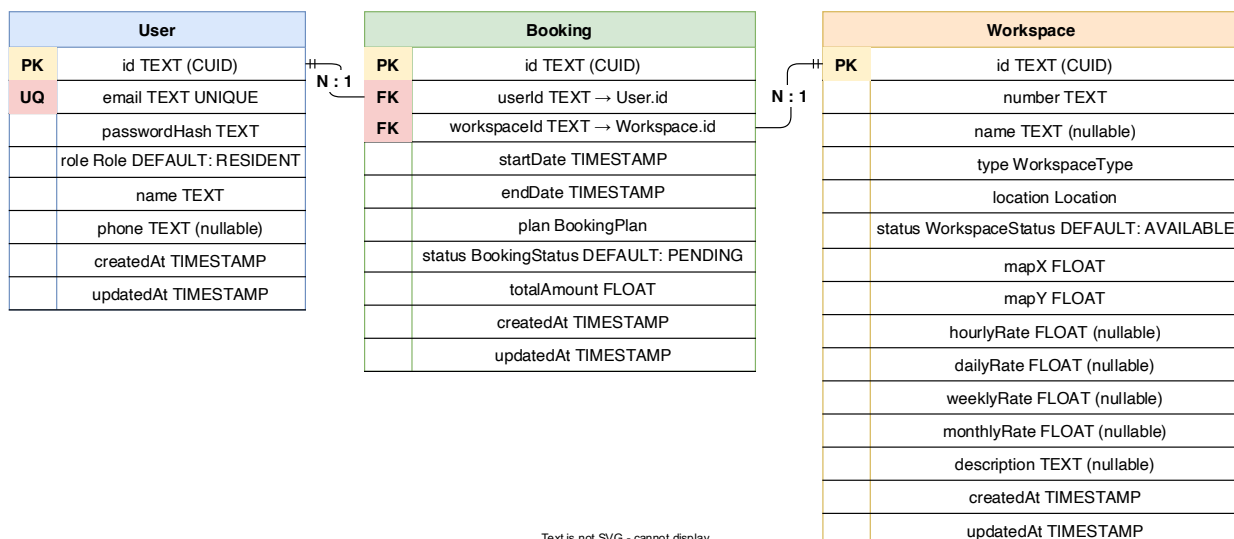


Рисунок 2.1 – ER-діаграма бази даних системи

База даних містить три основні таблиці: User, Workspace та Booking, а також шість перераховувальних типів (enum), що обмежують допустимі значення відповідних полів.

Перераховувальні типи: Role визначає роль користувача: GUEST – незареєстрований відвідувач, RESIDENT – зареєстрований клієнт (значення за замовчуванням), ADMIN – адміністратор. WorkspaceType визначає тип місця: OPEN_SPACE, FIXED_DESK, MEETING_ROOM. Location визначає локацію: PODIL, PECHERSK, OSOKORKY. WorkspaceStatus визначає стан місця: AVAILABLE або UNAVAILABLE. BookingStatus визначає стан бронювання: PENDING, CONFIRMED, CANCELLED. BookingPlan визначає тарифний план: HOUR – лише для переговорних кімнат, DAY, WEEK, MONTH.

Таблиця User: Таблиця User зберігає дані зареєстрованих користувачів. Первинний ключ id генерується за алгоритмом CUID. Поле email має унікальний індекс, що унеможливорює реєстрацію двох акаунтів з однаковою адресою. Пароль зберігається виключно у хешованому вигляді у полі passwordHash. Поле role має значення за замовчуванням RESIDENT. Поле phone є необов'язковим. Поля createdAt і updatedAt заповнюються автоматично.

Таблиця Workspace: Таблиця Workspace зберігає інформацію про всі робочі місця мережі. Поля type і location посиляються на відповідні enum-типи. Поля mapX і mapY зберігають координати місця на SVG-карті. Тарифні поля (hourlyRate, dailyRate, weeklyRate, monthlyRate) є необов’язковими, що дозволяє гнучко налаштовувати доступні плани для кожного типу місця: переговорні кімнати мають лише погодинний тариф, тоді як Open Space і Fixed Desk – денний, тижневий та місячний.

Таблиця Booking: Таблиця Booking є центральною і зберігає дані про всі бронювання. Поля userId і workspaceId є зовнішніми ключами, що посиляються на таблиці User і Workspace відповідно. Обидва зовнішні ключі налаштовані з правилом ON DELETE CASCADE – при видаленні користувача або місця всі пов’язані бронювання видаляються автоматично. Поле totalAmount зберігає розраховану суму оренди відповідно до обраного тарифного плану і тривалості.

Структура бази даних створювалась у два етапи. Перша міграція створила всі основні таблиці з підтримкою денного, тижневого і місячного тарифів. Друга міграція додала підтримку погодинного тарифу для переговорних кімнат: розширила enum BookingPlan значенням HOUR, додала поле hourlyRate до таблиці Workspace та зробила поля денного, тижневого і місячного тарифів необов’язковими.

2.3 Реалізація функцій системи

2.3.1 Модуль автентифікації

Модуль автентифікації забезпечує реєстрацію нових користувачів та вхід до системи на основі механізму JSON Web Tokens (JWT) [17]. На серверній стороні реалізовано два маршрути: POST /api/auth/register та POST /api/auth/login.

При реєстрації вхідні дані (ім’я, email, пароль, телефон) валідуються за схемою, визначеною бібліотекою Zod [22] – ім’я має містити не менше двох

символів, email повинен відповідати стандартному формату, пароль – не менше шести символів. У разі успішної валідації система перевіряє унікальність email у базі даних. Якщо такий email вже зареєстрований, повертається відповідь з кодом 409 (Conflict). За відсутності конфлікту пароль хешується алгоритмом bcrypt з коефіцієнтом складності 10, після чого запис користувача створюється в базі даних. У відповідь клієнт отримує JWT-токен та дані профілю. Лістинг коду маршруту реєстрації наведено на рисунку 2.2.

```
router.post('/register', async (req: Request, res: Response): Promise<void> => {
  const result = registerSchema.safeParse(req.body);
  if (!result.success) {
    res.status(400).json({ error: result.error.errors[0].message });
    return;
  }

  const { name, email, password, phone } = result.data;

  const existing = await prisma.user.findUnique({ where: { email } });
  if (existing) {
    res.status(409).json({ error: 'Email already in use' });
    return;
  }

  const passwordHash = await bcrypt.hash(password, 10);
  const user = await prisma.user.create({
    data: { name, email, passwordHash, phone },
    select: { id: true, name: true, email: true, role: true, phone: true },
  });

  const token = jwt.sign(
    { id: user.id, email: user.email, role: user.role },
    process.env.JWT_SECRET!,
    { expiresIn: '7d' }
  );

  res.status(201).json({ token, user });
});
```

Рисунок 2.2 – Лістинг коду маршруту реєстрації

При вході в систему сервер знаходить користувача за email, порівнює введений пароль з хешом через `bcrypt.compare()` та у разі збігу генерує JWT-токен з терміном дії 7 днів. Токен підписується секретним ключем із змінної середовища `JWT_SECRET` і містить у собі ідентифікатор, email та роль користувача. Захист маршрутів реалізований через middleware `authenticate`, який витягує токен з заголовка `Authorization: Bearer`, верифікує його підпис та додатково перевіряє існування користувача в базі даних. Це запобігає використанню токенів видалених акаунтів. Лістинг коду middleware автентифікації наведено на рисунку 2.3.

```

export async function authenticate(req: AuthRequest, res: Response, next: NextFunction): Promise<void> {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    res.status(401).json({ error: 'No token provided' });
    return;
  }

  const token = authHeader.slice(7);
  try {
    const payload = jwt.verify(token, process.env.JWT_SECRET!) as {
      id: string;
      role: string;
      email: string;
    };

    const dbUser = await prisma.user.findUnique({ where: { id: payload.id }, select: { id: true, role: true } });
    if (!dbUser) {
      res.status(401).json({ error: 'Session expired, please log in again' });
      return;
    }

    req.user = { ...payload, role: dbUser.role };
    next();
  } catch {
    res.status(401).json({ error: 'Invalid or expired token' });
  }
}

export function requireAdmin(req: AuthRequest, res: Response, next: NextFunction): void {
  if (req.user?.role !== 'ADMIN') {
    res.status(403).json({ error: 'Admin access required' });
    return;
  }
  next();
}

```

Рисунок 2.3 – Лістинг коду middleware автентифікації

Клієнтська частина зберігає отриманий токен у глобальному стані Zustand-сховища authStore. Сторінки реєстрації та входу наведено на рисунках 2.4 та 2.5 відповідно.

Sign Up

First Name

Phone Number

Email

Password

SIGN UP

By signing up, you agree to our [Terms & Privacy Policy](#).

Already have an account? [Log in](#)



Рисунок 2.4 – Форма реєстрації нового користувача

Log In

Email

Password

[Forgot password?](#)

LOG IN

[Don't have an account? Sign up](#)



Рисунок 2.5 – Форма авторизації користувача

2.3.2 Модуль перегляду локацій та робочих місць

Модуль перегляду забезпечує клієнту доступ до каталогу доступних локацій і робочих місць. Бронювання реалізоване у форматі покрокового флоу з чотирма послідовними кроками: спочатку користувач обирає локацію з трьох доступних відділень мережі, потім тип робочого місця (Open Space, Fixed Desk або Meeting Room), далі тарифний план з урахуванням обмежень для кожного типу місця, і нарешті переходить до кроку підтвердження з відображенням повного зведення бронювання та підсумкової вартості. Стан флоу зберігається у глобальному сховищі Zustand [19] між рендерами компонентів, що дозволяє користувачеві повертатись до попередніх кроків без втрати вже введених даних. Кнопка переходу до наступного кроку залишається недоступною доти, доки вибір на поточному кроці не зроблено, що виключає можливість переходу з неповними даними.

На серверній стороні маршрут GET /api/workspaces підтримує фільтрування за трьома параметрами: location, type та date. Параметри location і type передаються безпосередньо до Prisma-запиту як умови фільтрації в блоці where, що дозволяє

отримати лише ті записи, які відповідають обраній локації та типу місця. За наявності параметра `date` виконується додатковий запит до таблиці бронювань: для кожного робочого місця перевіряється наявність активного бронювання зі статусом `CONFIRMED`, дата початку якого збігається з переданою датою. На основі результату цієї перевірки до кожного запису про місце додається булеве поле `isBooked`, яке сигналізує про його зайнятість. Таким чином, клієнт отримує єдиний уніфікований масив об'єктів, де кожне місце вже містить актуальний статус доступності для обраної дати без необхідності виконувати додаткові запити з боку фронтенду. Лістинг коду маршруту отримання робочих місць з фільтрацією наведено на рисунку 2.6.

```
router.get('/', async (req: Request, res: Response): Promise<void> => {
  ··const { location, type, date } = req.query;

  ··const where: Record<string, unknown> = { status: 'AVAILABLE' };
  ··if (location) where.location = String(location).toUpperCase();
  ··if (type) where.type = String(type).toUpperCase();

  ··const workspaces = await prisma.workspace.findMany({
  ··  where,
  ··  include: {
  ··    bookings: date
  ··    ? {
  ··      where: {
  ··        status: { in: ['PENDING', 'CONFIRMED'] },
  ··        startDate: { lte: new Date(String(date)) },
  ··        endDate: { gte: new Date(String(date)) },
  ··      },
  ··    }
  ··    : false,
  ··  },
  ··  orderBy: [{ location: 'asc' }, { type: 'asc' }, { number: 'asc' }],
  ··});

  ··const result = workspaces.map((w) => ({
  ··  ...w,
  ··  isBooked: date ? (w.bookings as unknown[]).length > 0 : false,
  ··  bookings: undefined,
  ··}));

  ··res.json(result);
});
```

Рисунок 2.6 – Лістинг коду маршруту `GET /api/workspaces`

Клієнтська сторінка відображає доступні локації у вигляді карток. При виборі локації та типу місця система отримує відфільтрований список із сервера та відображає відповідні місця. Скріншот сторінки вибору локації наведено на рисунку 2.7, а сторінки перегляду доступних місць – на рисунку 2.8.

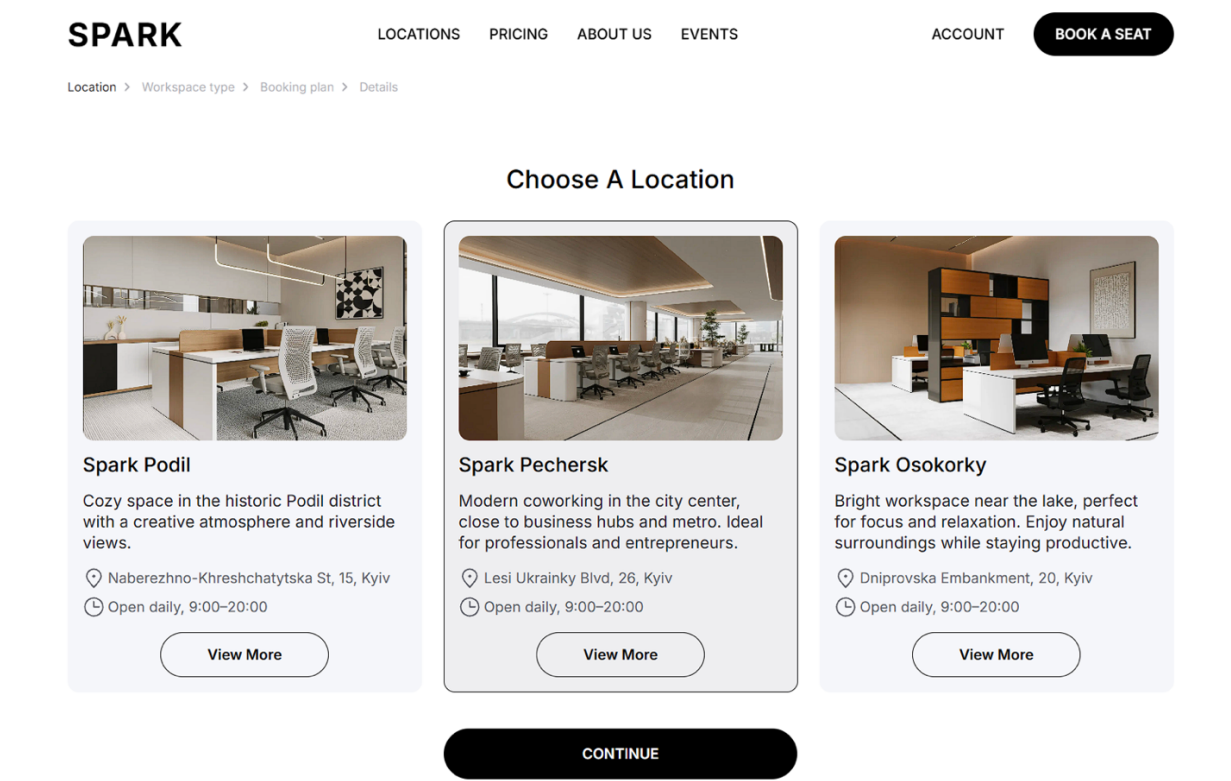


Рисунок 2.7 – Інтерфейс вибору локації коворкінгу

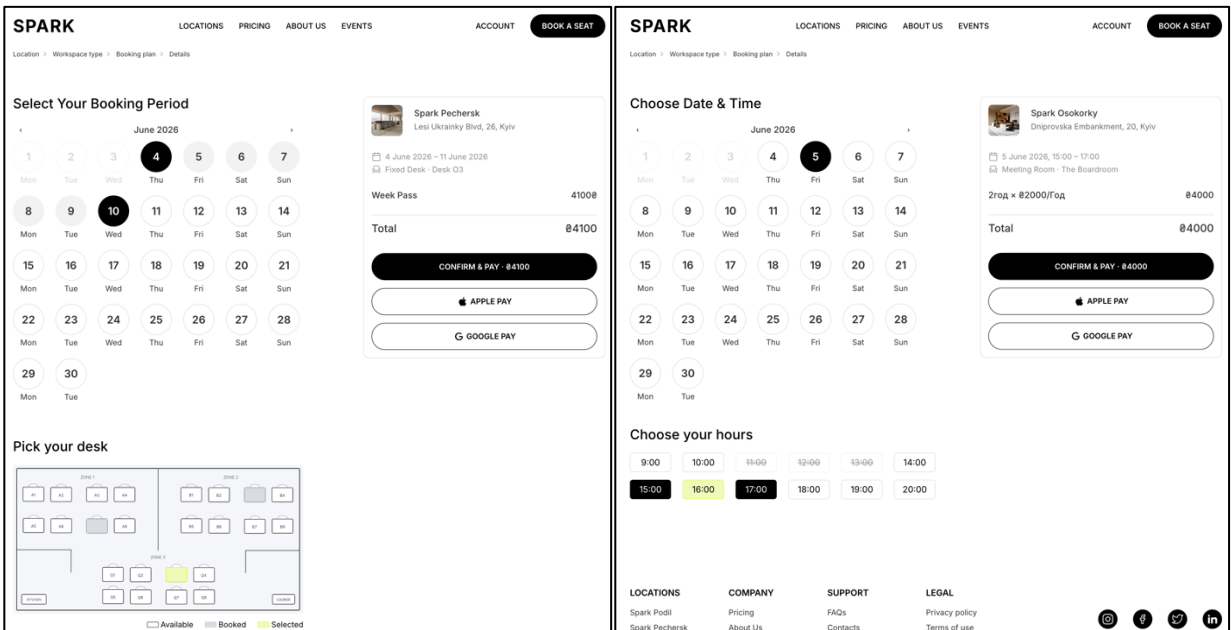


Рисунок 2.8 – Інтерфейс каталогу робочих місць

2.3.3 Інтерактивна карта для Fixed Desk

Для типу робочого місця Fixed Desk реалізовано інтерактивну карту поверху у форматі SVG, яка дозволяє клієнту візуально обрати конкретний стіл. Кожне місце на карті відповідає запису в базі даних і має координати $mapX$ та $mapY$, що визначають його положення на схемі.

Клієнтський компонент отримує список Fixed Desk з сервера разом з їхнім статусом зайнятості та відображає їх на SVG-полотні у вигляді прямокутників з кольоровою індикацією: вільне місце відображається прозорим, зайняте – сірим кольором, вибране користувачем – жовто-зеленим. При наведенні на вільне місце воно підсвічується напівпрозорим затемненням. При наведенні на місце відображається спливаюча підказка з номером. При кліку на вільне місце воно стає обраним і користувач переходить до наступного кроку флоу. Лістинг коду компонента SVG-карти наведено на рисунку 2.10, а результат його відображення – на рисунку 2.9.

Pick your desk

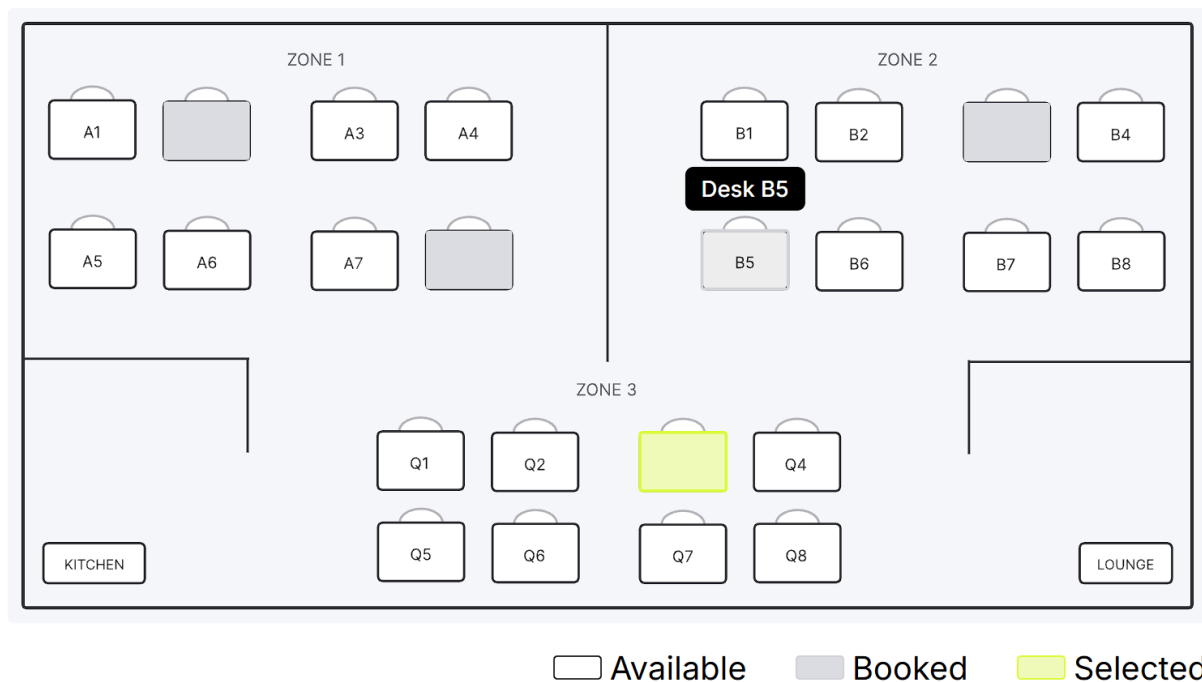


Рисунок 2.9 – Інтерактивна карта поверху для вибору Fixed Desk

```

<svg
  viewBox="0 0 600 308"
  className={styles.deskMapOverlay}
  xmlns="http://www.w3.org/2000/svg"
>
  {workspaces.map((w) => {
    const r = DESK_RECTS[w.number];
    if (!r) return null;
    const isBooked = !!w.isBooked;
    const isSelected = selected?.id === w.id;
    const isHov = hoveredId === w.id;
    return (
      <rect
        key={w.id}
        x={r.x}
        y={r.y}
        width={r.w}
        height={r.h}
        rx={1.3}
        fill={
          isSelected
            ? "#F0FAB8"
            : isBooked
            ? "#DBDBE2"
            : isHov
            ? "rgba(0,0,0,0.07)"
            : "transparent"
        }
        stroke={
          isSelected
            ? "#DAFA3E"
            : isHov && !isBooked
            ? "#D2D2D8"
            : "none"
        }
        strokeWidth={isSelected || (isHov && !isBooked) ? 1.5 : 0}
        style={{ cursor: isBooked ? "not-allowed" : "pointer" }}
        onClick={() => !isBooked && onSelect(w)}
        onMouseEnter={() => !isBooked && setHoveredId(w.id)}
        onMouseLeave={() => setHoveredId(null)}
      />
    );
  })}
</svg>

```

Рисунок 2.10 – Лістинг коду компонента інтерактивної SVG-карти

З технічної точки зору SVG-карта є векторним полотном із фіксованими розмірами 600×308 умовних одиниць, що відповідають пропорціям реального плану поверху. Координати кожного робочого місця зберігаються у полях `mapX` та `mapY` і визначають положення відповідного прямокутника (28×28 одиниць) на полотні. При завантаженні компонент надсилає запит до `GET /api/workspaces` з параметрами `location`, `type=FIXED_DESK` та обраною датою, отримує масив

об'єктів із координатами та полем `isBooked` і на їх основі програмно генерує SVG-елементи засобами `React`.

Управління вибором місця реалізовано через локальний `useState`-хук: при кліку на вільний прямокутник його ідентифікатор записується у стан `selectedId`, колір заливки змінюється на жовто-зелений, а сам ідентифікатор передається у глобальне сховище `Zustand` [19] для використання на кроці підтвердження. Повторний клік скасовує вибір. Зайняті місця повністю виключені з логіки взаємодії: обробник `onClick` для них не реєструється, а курсор набуває вигляду `not-allowed`.

Перевагою SVG-підходу є масштабованість без втрати чіткості та динамічна генерація елементів на основі актуальних даних із бази – будь-яка зміна статусу місця відображається коректно при наступному завантаженні компонента без оновлення графічних файлів.

2.3.4 Модуль бронювання

Модуль бронювання є ключовим з точки зору надійності системи, оскільки реалізує транзакційний механізм резервування, що унеможлиблює подвійне бронювання одного місця.

На серверній стороні маршрут `POST /api/bookings` вимагає автентифікації та приймає чотири параметри: ідентифікатор місця, дату початку, дату завершення та тарифний план. Після валідації вхідних даних виконується транзакція `prisma.$transaction()`, що охоплює кілька послідовних кроків. Спочатку система перевіряє існування та доступність робочого місця. Далі перевіряється відповідність тарифного плану типу місця: переговорні кімнати приймають лише погодинний план, усі інші типи – лише денний, тижневий або місячний. Для `Fixed Desk` і `Meeting Room` виконується перевірка конфліктів – пошук будь-якого існуючого бронювання зі статусами `PENDING` або `CONFIRMED`, що перетинається за часом з запитуваним інтервалом. Для `Open Space` перевірка конфліктів не

виконується, оскільки це простір спільного користування. За відсутності конфліктів система розраховує загальну вартість `totalAmount` залежно від тарифного плану та тривалості оренди. Нарешті, транзакційно створюється запис бронювання зі статусом `CONFIRMED`. Якщо будь-який з кроків завершується помилкою, уся транзакція відкочується. Лістинг коду транзакційного бронювання наведено на рисунку 2.11.

```
const booking = await prisma.$transaction(async (tx) => {
  const workspace = await tx.workspace.findUnique({
    where: { id: workspaceId },
  });
  if (!workspace || workspace.status === "UNAVAILABLE") {
    throw new Error("UNAVAILABLE");
  }

  if (workspace.type === "MEETING_ROOM" && plan !== "HOURLY") {
    throw new Error("MEETING_ROOM_HOURLY_ONLY");
  }
  if (workspace.type !== "MEETING_ROOM" && plan === "HOURLY") {
    throw new Error("HOURLY_MEETING_ROOMS_ONLY");
  }

  // Open space is a shared area – multiple passes can overlap
  if (workspace.type !== "OPEN_SPACE") {
    const conflict = await tx.booking.findFirst({
      where: {
        workspaceId,
        status: { in: ["PENDING", "CONFIRMED"] },
        AND: [{ startDate: { lt: end } }, { endDate: { gt: start } }],
      },
    });
    if (conflict) {
      throw new Error("CONFLICT");
    }
  }

  let totalAmount: number;
  if (plan === "HOURLY") {
    const hours = (end.getTime() - start.getTime()) / (1000 * 60 * 60);
    totalAmount = Math.round(workspace.hourlyRate! * hours * 100) / 100;
  } else if (plan === "DAY") {
    totalAmount = workspace.dailyRate!;
  } else if (plan === "WEEK") {
    totalAmount = workspace.weeklyRate!;
  } else {
    totalAmount = workspace.monthlyRate!;
  }

  return tx.booking.create({
    data: {
      userId: req.user!.id,
      workspaceId,
      startDate: start,
      endDate: end,
      plan,
      status: "CONFIRMED",
      totalAmount,
    },
    include: {
      workspace: { select: { number: true, type: true, location: true } },
      user: { select: { name: true, email: true } },
    },
  });
});
```

Рисунок 2.11 – Лістинг коду транзакції бронювання

На клієнтській стороні останній крок флоу відображає форму підтвердження з підсумком: місце, локація, тип, дата і тривалість, тарифний план та загальна вартість. Після натискання кнопки підтвердження клієнт надсилає запит до сервера і при успішній відповіді переходить на сторінку з деталями бронювання. Форма підтвердження бронювання наведено на рисунку 2.12, а сторінка успішного завершення – на рисунку 2.13.



Spark Osokorky

Dniprovska Embankment, 20, Kyiv



5 June 2026, 15:00 – 17:00



Meeting Room · The Boardroom

2год × ₪2000/Год

₪4000

Total

₪4000

CONFIRM & PAY · ₪4000

 APPLE PAY

 GOOGLE PAY

Рисунок 2.12 – Форма підтвердження бронювання

SPARK

LOCATIONS

PRICING

ABOUT US

EVENTS

ACCOUNT

BOOK A SEAT



BOOKING CONFIRMED.

SEE YOU AT SPARK.

We sent a confirmation to your email with a QR code for the front desk. You can find this booking any time in your account.

Location

Spark Podil

Workspace

Fixed Desk

Plan

Week Pass

Date

12 June 2026 – 19 June 2026

Desk

B2

Total paid

₪4100

GO TO MY BOOKINGS

DOWNLOAD INVOICE

Рисунок 2.13 – Сторінка успішного оформлення бронювання

2.3.5 Особистий кабінет резидента

Особистий кабінет надає авторизованому резиденту доступ до управління власними бронюваннями та перегляду профілю. Маршрут GET /api/bookings/my повертає всі бронювання поточного користувача у зворотному хронологічному порядку разом з деталями пов'язаного робочого місця. На серверній стороні запит виконується через Prisma [24] з використанням оператора include для підтягування пов'язаних даних про workspace в рамках одного запиту до бази даних без додаткових round-trip звернень. Ідентифікатор користувача витягується безпосередньо з JWT-токена [17] у middleware автентифікації, що унеможливорює отримання бронювань іншого резидента навіть при підробці параметрів запиту.

Клієнтська сторінка структурована у вигляді двох вкладок: «My Bookings» та «Profile». На вкладці бронювань реалізована система фільтрації за статусом із чотирма фільтрами: «Upcoming», «Past», «Cancelled» та «All». Поруч із кожним фільтром відображається лічильник відповідних записів, який обчислюється на клієнті на основі отриманого масиву даних без додаткових запитів до сервера. Додатково реалізовано рядок пошуку, що дозволяє фільтрувати список за назвою локації або номером кімнати в режимі реального часу. Активний фільтр і пошуковий рядок застосовуються одночасно, тобто користувач може шукати серед лише майбутніх або лише скасованих бронювань.

Кожна картка бронювання відображає повний набір деталей: дату початку з виділеним числом і місяцем для швидкого орієнтування, тарифний план, тип місця у вигляді тегу, повну адресу та назву локації, діапазон дат оренди та загальну вартість. Клієнтська сторінка відображає список бронювань у вигляді карток, де для кожного запису показано номер місця, локацію, тип, дати оренди, тарифний план, суму та поточний статус. Активні бронювання зі статусом CONFIRMED мають дві кнопки дії: «Details» для перегляду розгорнутої інформації та «Cancel» для ініціювання процедури скасування.

При натисканні кнопки «Cancel» відкривається модальне вікно підтвердження, яке відображає ключові деталі бронювання – локацію, тип місця,

дату та суму повернення – і пропонує два варіанти дії: підтвердити скасування або залишити бронювання активним. Такий підхід запобігає випадковому скасуванню і надає користувачеві можливість переглянути деталі перед прийняттям остаточного рішення.

На серверній стороні маршрут PATCH `/api/bookings/:id/cancel` спочатку перевіряє існування бронювання, потім підтверджує, що запит надходить від його власника або від адміністратора – в іншому випадку повертається відповідь з кодом 403 (Forbidden). Після успішної перевірки статус запису змінюється на CANCELLED через Prisma-транзакцію. Після успішного скасування клієнтська частина оновлює локальний стан без перезавантаження сторінки: лічильник вкладки «Upcoming» зменшується на 1, лічильник «Cancelled» збільшується на 1, а скасована картка переміщується до відповідної вкладки. Інтерфейс особистого кабінету наведено на рисунку 2.14.

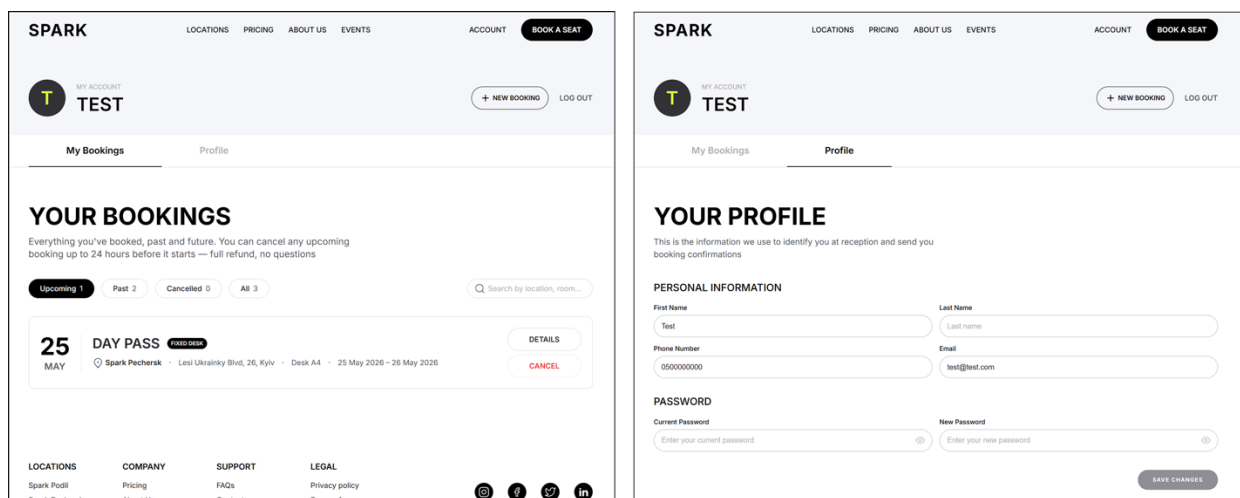


Рисунок 2.14 – Інтерфейс особистого кабінету резидента

Вкладка «Profile» відображає персональні дані резидента: ім'я, email та номер телефону. Дані завантажуються з маршруту GET `/api/users/me`, який також захищений middleware автентифікації і повертає лише дані поточного авторизованого користувача. Поле email відображається у режимі «тільки для читання», оскільки є унікальним ідентифікатором облікового запису і не підлягає зміні через інтерфейс. Поля імені та телефону доступні для редагування.

2.3.6 Адміністративна панель

Адміністративна панель надає користувачу з роллю ADMIN розширені можливості з управління системою та моніторингу завантаженості. Доступ до всіх адміністративних маршрутів обмежений middleware `requireAdmin`, яка перевіряє роль у JWT-токені [17] та повертає код 403 у разі спроби несанкціонованого доступу.

Дашборд адміністратора завантажує зведену. Сервер паралельно виконує шість запитів до бази даних за допомогою `Promise.all()` та повертає: загальний дохід від усіх підтверджених бронювань, дохід за поточний місяць, кількість активних бронювань, загальну кількість бронювань, кількість зареєстрованих резидентів, а також розподіл бронювань за типами місць і локаціями. Лістинг коду маршруту статистики наведено на рисунку 2.15, а скріншот дашборду – на рисунку 2.16.

```
router.get('/stats', authenticate, async (req: AuthRequest, res: Response): Promise<void> => {
  if (!guard(req, res)) return;

  const now = new Date();
  const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);

  const [totalRev, monthlyRev, activeBookings, totalBookings, totalMembers, confirmedWithWs] =
    await Promise.all([
      prisma.booking.aggregate({ where: { status: 'CONFIRMED' }, _sum: { totalAmount: true } }),
      prisma.booking.aggregate({
        where: { status: 'CONFIRMED', createdAt: { gte: startOfMonth } },
        _sum: { totalAmount: true },
      }),
      prisma.booking.count({ where: { status: 'CONFIRMED', endDate: { gte: now } } }),
      prisma.booking.count(),
      prisma.user.count(),
      prisma.booking.findMany({
        where: { status: 'CONFIRMED' },
        include: { workspace: { select: { type: true, location: true } } },
      }),
    ]);

  const bookingsByType: Record<string, number> = {};
  const bookingsByLocation: Record<string, number> = {};
  for (const b of confirmedWithWs) {
    bookingsByType[b.workspace.type] = (bookingsByType[b.workspace.type] ?? 0) + 1;
    bookingsByLocation[b.workspace.location] = (bookingsByLocation[b.workspace.location] ?? 0) + 1;
  }

  res.json({
    totalRevenue: totalRev._sum.totalAmount ?? 0,
    monthlyRevenue: monthlyRev._sum.totalAmount ?? 0,
    activeBookings,
    totalBookings,
    totalMembers,
    bookingsByType,
    bookingsByLocation,
  });
});
```

Рисунок 2.15 – Лістинг коду маршруту статистики

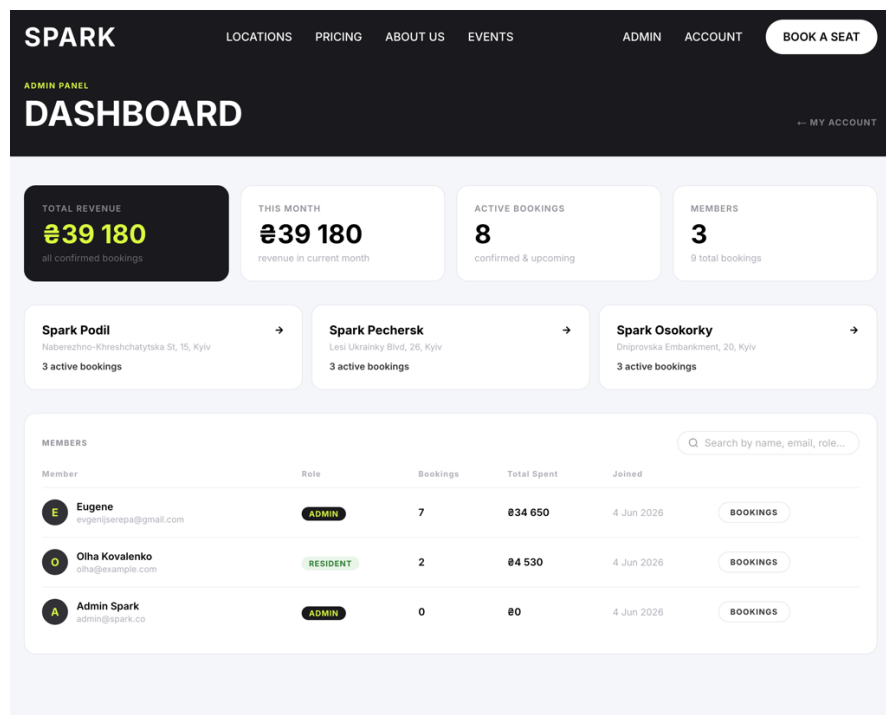


Рисунок 2.16 – Дашборд адміністративної панелі системи

Маршрут `GET /api/admin/users` повертає список усіх зареєстрованих користувачів з агрегованими даними: загальна кількість бронювань і сума витрат кожного резидента. Адміністратор може обрати будь-якого резидента і переглянути деталі всіх його. Скріншот таблиці резидентів наведено на рисунку 2.17.

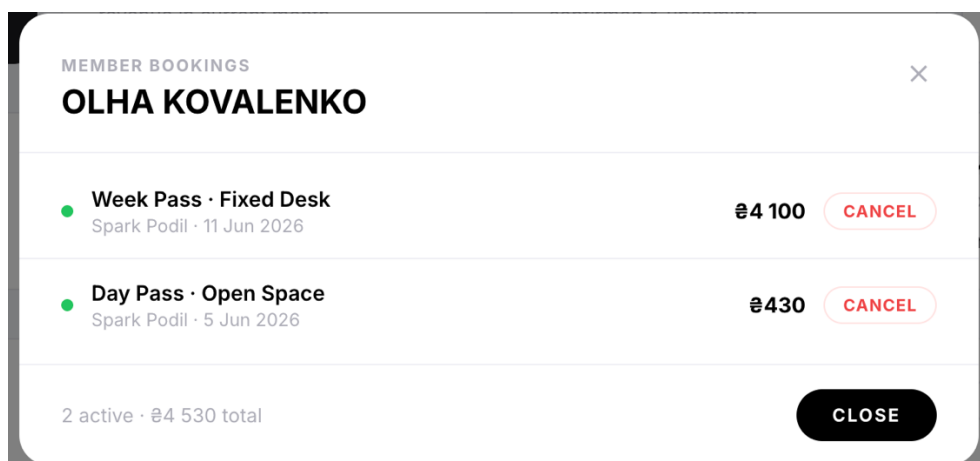


Рисунок 2.17 – Інтерфейс управління резидентами

Управління завантаженістю локацій реалізовано через, який повертає всі робочі місця обраної локації разом з активними бронюваннями та даними

резидентів. Адміністратор також може змінювати статус будь-якого місця (AVAILABLE / UNAVAILABLE). Інтерфейс сторінки управління локацією наведено на рисунку 2.18.

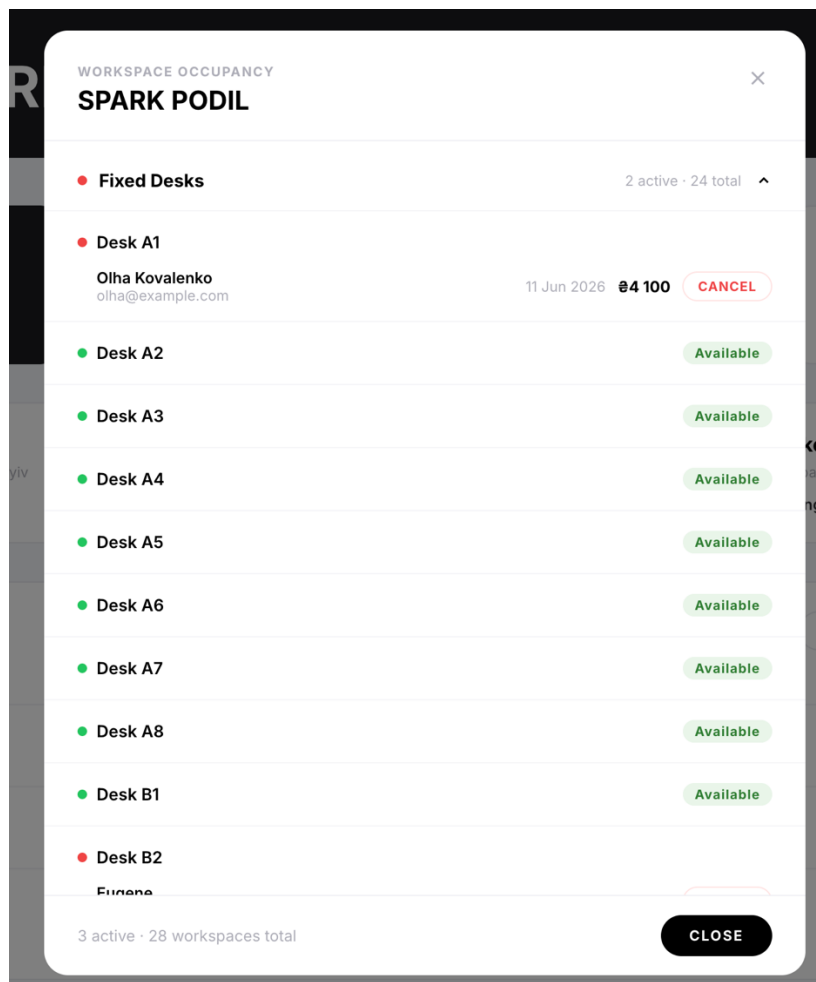


Рисунок 2.18 – Інтерфейс управління локацією коворкінгу

2.4. Інструкція користувача

2.4.1 Загальна навігація та базові сторінки

Веб-сайт має єдину навігаційну панель у верхній частині екрана, що забезпечує доступ до всіх розділів сайту. Для незареєстрованого відвідувача доступні кнопки входу та реєстрації. Після авторизації навігаційна панель відображає посилання на особистий кабінет та кнопку виходу з системи.

Головна сторінка є вхідною точкою застосунку і містить загальну інформацію про мережу коворкінгів: переваги, типи доступних просторів та заклик до дії з переходом до бронювання (рисунки 2.19). Сторінка адаптована для мобільних пристроїв та різних розмірів екранів.

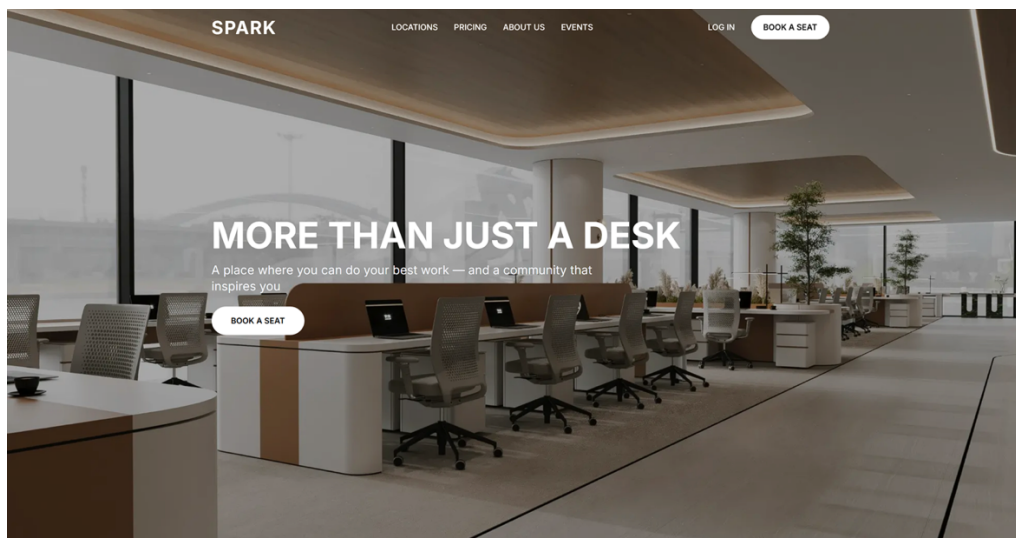


Рисунок 2.19 – Головна сторінка сайту

Сторінка «Локації» (рисунки 2.20) надає детальний опис кожної з трьох відділень мережі – Поділ, Печерськ та Осокорки. Для кожної локації відображається адреса, графік роботи, перелік зручностей у вигляді тегів та кнопка переходу до бронювання.

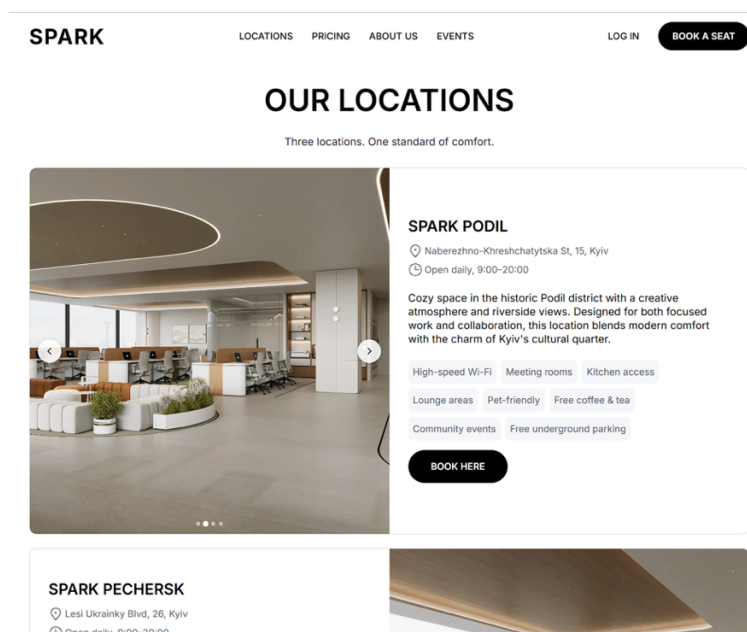


Рисунок 2.20 – Сторінка локацій коворкінгу

Сторінка «Ціни» відображає актуальні тарифні плани для кожного типу робочого місця у вигляді інтерактивних вкладок. Користувач може перемикатися між типами місць і порівнювати вартість денного, тижневого та місячного планів (рисунки 2.21).

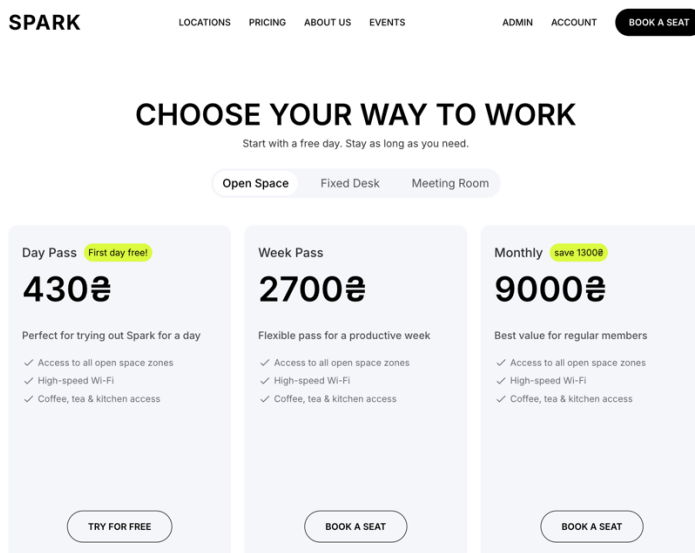


Рисунок 2.21 – Сторінка тарифних планів

Сторінка «Події» містить анонси заходів, що проводяться у просторах мережі: воркшопи, нетворкінг-вечори та тематичні зустрічі для резидентів (рисунки 2.22).

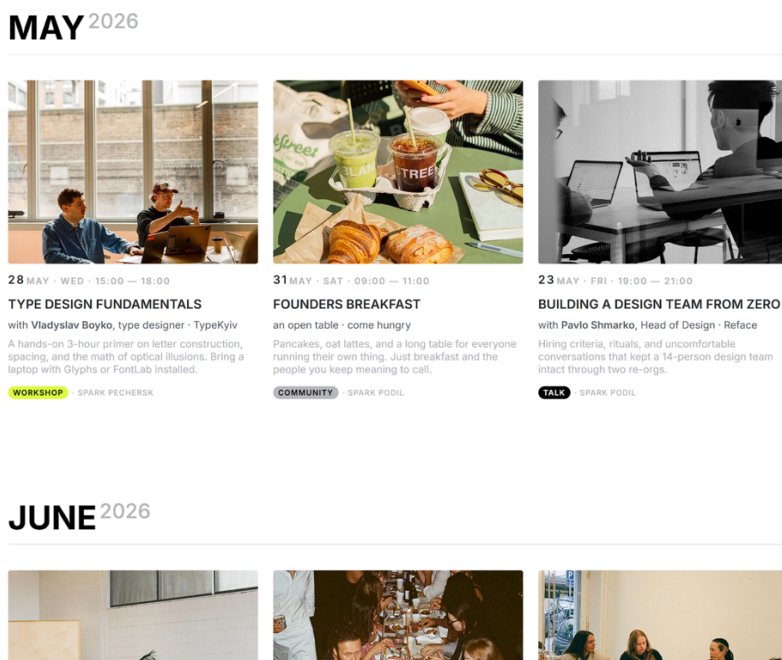


Рисунок 2.22 – Сторінка подій коворкінгу

Сторінка «Про нас» містить інформацію про концепцію мережі коворкінгів, її місію та цінності. Відвідувач може ознайомитися з історією заснування, командою та перевагами співпраці з мережею (рисунки 2.23).

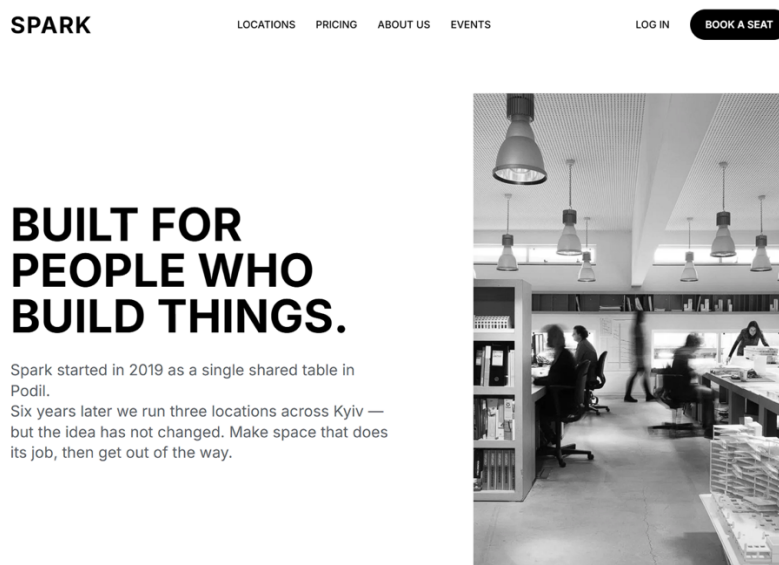


Рисунок 2.23 – Сторінка «Про нас»

2.4.2 Реєстрація, авторизація та процес бронювання

Для початку роботи із системою новий користувач має пройти реєстрацію. Для цього необхідно перейти на сторінку реєстрації, доступну за посиланням у навігаційному меню. Форма реєстрації (рисунки 2.4) містить поля для введення імені, адреси електронної пошти, паролю та номера телефону. Система виконує валідацію введених даних у режимі реального часу: ім'я має містити не менше двох символів, пароль – не менше шести. У разі спроби реєстрації з уже використаним email система відображає повідомлення про помилку. Після успішної реєстрації користувач автоматично авторизується і перенаправляється на головну сторінку.

Для повторного входу в систему зареєстрований користувач переходить на сторінку авторизації (рисунки 2.5), вводить email та пароль і натискає кнопку входу. У разі введення невірних даних система відображає відповідне повідомлення.

Процес бронювання робочого місця реалізований у форматі покрокового флоу і складається з чотирьох послідовних кроків.

Крок 1. Вибір локації. Користувач переходить до розділу бронювання та обирає одну з трьох доступних локацій мережі: Поділ, Печерськ або Осокорки. Кожна локація представлена карткою з назвою та адресою (рисунок 2.7). Після вибору локації користувач переходить до наступного кроку.

Крок 2. Вибір типу та конкретного місця. Система пропонує три типи робочих місць: Open Space, Fixed Desk та переговорна кімната. Після вибору типу відображається відповідний інтерфейс вибору місця. Для Open Space і переговорних кімнат показується список доступних варіантів (рисунок 2.8). Для Fixed Desk відображається інтерактивна карта поверху (рисунок 2.9), на якій сірим кольором позначено зайняті місця, а вільні доступні для вибору. Після кліку на вільний стіл він підсвічується жовто-зеленим кольором як обраний.

Крок 3. Вибір тарифного плану та дати. Залежно від типу обраного місця система відображає доступні тарифні плани (рисунок 2.24). Для переговорних кімнат доступне лише погодинне бронювання з вибором часового інтервалу. Для Open Space і Fixed Desk доступні денний, тижневий та місячний плани з вибором дати початку оренди. Система автоматично розраховує дату завершення і відображає загальну вартість оренди.

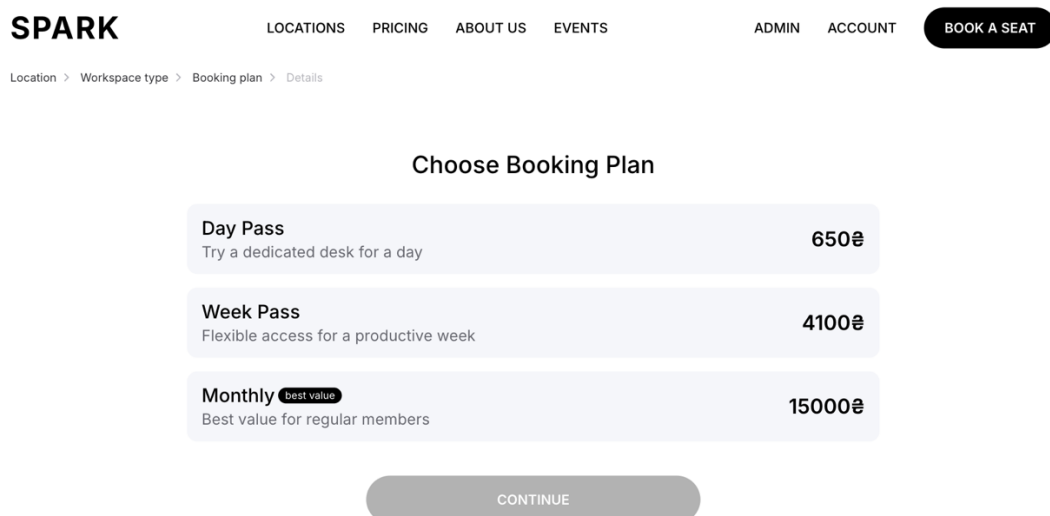


Рисунок 2.24 – Інтерфейс вибору тарифного плану

Крок 4. Підтвердження бронювання. На фінальному кроці користувач бачить зведену форму підтвердження (рисунк 2.12) з усіма деталями: локація, тип і номер місця, дати оренди, тарифний план та загальна сума. Після натискання кнопки підтвердження система надсилає запит до сервера, який транзакційно перевіряє доступність місця і фіксує бронювання. У разі успіху користувач переходить на сторінку підтвердження (рисунк 2.13) з деталями оформленого замовлення.

Для перегляду власних бронювань користувач переходить до особистого кабінету, доступного через навігаційне меню. Сторінка відображає повну історію бронювань у вигляді карток у зворотному хронологічному порядку (рисунк 2.14). Кожна картка містить інформацію про локацію, тип і номер місця, дати оренди, тарифний план, суму та поточний статус бронювання (рисунк 2.25).

Активне бронювання зі статусом «Підтверджено» може бути скасоване користувачем натисканням відповідної кнопки. Система запитує підтвердження скасування і після згоди змінює статус бронювання на «Скасовано». Скасовані бронювання залишаються у загальному списку з відповідним статусом.

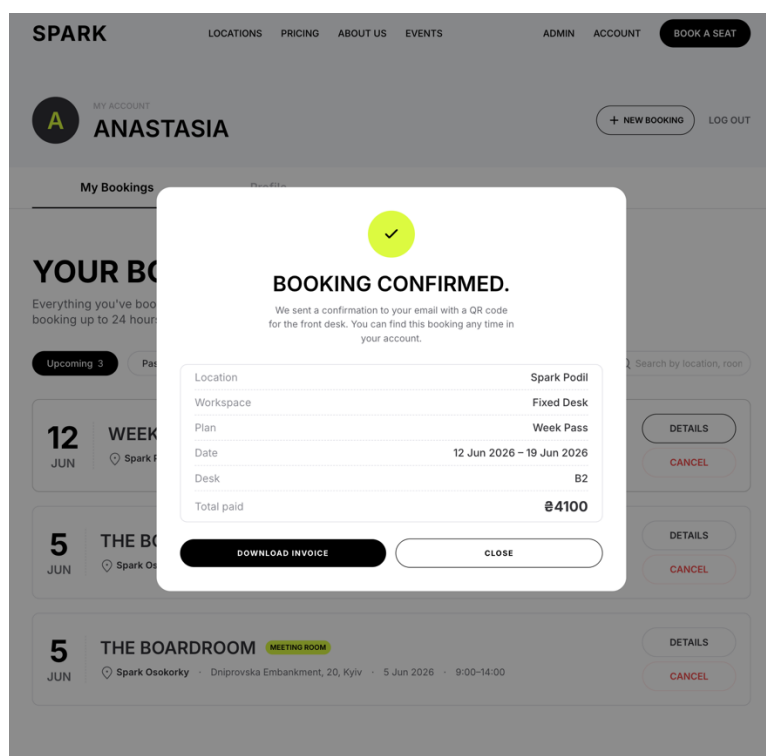


Рисунок 2.25 – Інтерфейс деталей бронювання в особистому кабінеті

2.4.3 Робота адміністратора в панелі управління

Адміністративна панель доступна лише користувачам з роллю ADMIN і відкривається автоматично після входу в систему з відповідними правами. Спроба перейти до адміністративного розділу з обліковим записом резидента призводить до автоматичного перенаправлення на головну сторінку – захист реалізовано на рівні клієнтського маршрутизатора Next.js [11] через перевірку ролі у збереженому JWT-токені [17]. Навігаційне меню адміністративної панелі містить три розділи: «Dashboard», «Members» та «Locations», між якими можна переходити без перезавантаження сторінки.

Головна сторінка панелі – дашборд (рисунок 2.16) – відображає зведену статистику: загальний дохід, дохід за поточний місяць, кількість активних бронювань, загальна кількість бронювань і резидентів, а також розподіл бронювань за типами місць і локаціями у вигляді наочних індикаторів. Усі показники обчислюються на серверній стороні через агрегаційні запити Prisma: загальний і місячний дохід визначаються як сума поля `totalAmount` усіх бронювань зі статусом `CONFIRMED`, активні бронювання – як кількість записів із датою початку, що не перевищила поточну дату. Це дозволяє адміністратору отримати актуальний фінансовий зріз у режимі реального часу без необхідності формування окремих звітів.

Розділ управління резидентами (рисунок 2.17) містить таблицю всіх зареєстрованих користувачів з агрегованими даними: кількість бронювань і загальна сума витрат кожного резидента. Таблиця підтримує пошук за ім'ям, адресою електронної пошти або роллю користувача – результати фільтруються в режимі реального часу без звернень до сервера, оскільки повний список завантажується одним запитом при відкритті розділу. Роль кожного користувача відображається у вигляді кольорового тегу: «RESIDENT» – зеленим, «ADMIN» – чорним, що дозволяє швидко розрізняти типи облікових записів у великому списку.

Адміністратор може обрати будь-якого резидента і переглянути детальну історію його бронювань у вигляді модального вікна. Вікно відображає

хронологічний список бронювань із зазначенням типу місця, локації, дати та суми для кожного запису. Активні бронювання мають кнопку «Cancel», що дозволяє адміністратору скасувати замовлення від свого імені – наприклад, у разі форс-мажорних обставин або вирішення конфліктних ситуацій. При скасуванні бронювання через адміністративну панель сервер виконує ту саму логіку, що й при скасуванні резидентом: статус змінюється на CANCELLED, місце стає доступним для нових бронювань. У нижній частині модального вікна відображається підсумок: кількість активних бронювань і загальна сума по резиденту, що оновлюється після кожного скасування.

Розділ управління локаціями (рисунок 2.18) дозволяє адміністратору переглянути поточну завантаженість кожної з трьох локацій мережі. Кожна локація відображається у вигляді окремої картки із зазначенням кількості активних бронювань. При переході до конкретної локації відображається детальний список її робочих місць з інформацією про активні бронювання – ім'я резидента, тип місця, дати оренди та контактні дані. Адміністратор може змінити статус будь-якого місця з AVAILABLE на UNAVAILABLE і навпаки. Зміна статусу виконується через маршрут PATCH /api/workspaces/:id/status і набирає чинності негайно: місце зі статусом UNAVAILABLE зникає з каталогу для клієнтів і не відображається при бронюванні, що дозволяє оперативно виводити місця з експлуатації для технічного обслуговування або ремонту без необхідності видалення записів із бази даних.

Таким чином, адміністративна панель забезпечує повний операційний контроль над усіма аспектами роботи коворкінгу в єдиному інтерфейсі: від моніторингу фінансових показників і управління клієнтською базою до оперативного регулювання доступності ресурсного фонду в режимі реального часу. На відміну від поточного стану, де аналогічна інформація розпорошена між таблицями та месенджерами, панель консолідує всі операційні дані з миттєвим доступом, що суттєво скорочує час на рутинні операції і дозволяє адміністратору зосередитись на розвитку спільноти та підвищенні якості сервісу.

2.5 Тестування програмного продукту

2.5.1 Функціональне тестування API

Функціональне тестування серверної частини охоплювало перевірку усіх REST API маршрутів системи. Для кожного тест-кейсу визначались вхідні дані, очікуваний результат і фактичний результат виконання. Повний перелік тест-кейсів та результати їх виконання наведено у таблиці 2.1.

Таблиця 2.1 – Результати функціонального тестування API

<i>№</i>	<i>Модуль</i>	<i>Опис тесту</i>	<i>Очікуваний результат</i>	<i>Фактичний результат</i>	<i>Статус</i>
ТС-01	Реєстрація	Коректні дані	HTTP 201, токен	HTTP 201, токен	✓
ТС-02	Реєстрація	Дублікат email	HTTP 409	HTTP 409	✓
ТС-03	Реєстрація	Некоректний email	HTTP 400	HTTP 400	✓
ТС-04	Реєстрація	Пароль < 6 символів	HTTP 400	HTTP 400	✓
ТС-05	Авторизація	Вірні дані	HTTP 200, токен	HTTP 200, токен	✓
ТС-06	Авторизація	Невірний пароль	HTTP 401	HTTP 401	✓
ТС-07	Робочі місця	Отримання всіх	HTTP 200, 84 записи	HTTP 200, 84 записи	✓
ТС-08	Робочі місця	Фільтр location=PODIL	HTTP 200, 28 записів	HTTP 200, 28 записів	✓
ТС-09	Робочі місця	Фільтр за датою	Поле isBooked присутнє	isBooked: 84 поля	✓
ТС-10	Бронювання	Fixed Desk, план DAY	HTTP 201, CONFIRMED	HTTP 201, CONFIRMED	✓
ТС-11	Бронювання	Конфлікт дат	HTTP 409	HTTP 409	✓
ТС-12	Бронювання	Meeting Room, план HOUR	HTTP 201, CONFIRMED	HTTP 201, CONFIRMED	✓
ТС-13	Бронювання	Meeting Room + план DAY	HTTP 400	HTTP 400	✓
ТС-14	Бронювання	Fixed Desk + план HOUR	HTTP 400	HTTP 400	✓

Продовження табл. 2.1

ТС-15	Бронювання	Скасування бронювання	HTTP 200, CANCELLED	HTTP 200, CANCELLED	✓
ТС-16	Бронювання	Мої бронювання	HTTP 200, список	HTTP 200, список	✓
ТС-17	Адмін	Статистика	HTTP 200, об'єкт	HTTP 200, об'єкт	✓
ТС-18	Адмін	Список резидентів	HTTP 200, список	HTTP 200, список	✓
ТС-19	Адмін	Зміна статусу місця	HTTP 200, оновлено	HTTP 200, оновлено	✓

Скріншоти виконання тестів блоків 1–2 (реєстрація та авторизація) наведено на рисунку 2.26. Результати виконання тестів блоків 3–5 (робочі місця, бронювання та адмін) – на рисунку 2.27. Усі 19 тест-кейсів пройшли успішно.

```

1 |===== BLOCK 1: REGISTRATION =====
2
3 --- TC-01: Valid registration ---
4 {"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNtcHRzbDdnMjAwMDAwMnJ0N3U5cnIyYWkiLCJlbWFn
5 HTTP 201
6
7 --- TC-02: Duplicate email ---
8 {"error":"Email already in use"}
9 HTTP 409
10
11 --- TC-03: Invalid email ---
12 {"error":"Invalid email address"}
13 HTTP 400
14
15 --- TC-04: Short password ---
16 {"error":"Password must be at least 6 characters"}
17 HTTP 400
18
19 |===== BLOCK 2: AUTH =====
20
21 --- TC-05: Login with correct credentials ---
22 {"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNtcHRzbDdnMjAwMDAwMnJ0N3U5cnIyYWkiLCJlbWFn
23
24 HTTP 200
25 TOKEN captured: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNtcHRzbDdnMjAwMDAwMnJ0N3U5cnIyYWkiLCJlbWFn
26
27 --- TC-06: Wrong password ---
28 {"error":"Invalid email or password"}
29 HTTP 401

```

Рисунок 2.26 – Результати API тестування: блоки реєстрації та авторизації (ТС-01 – ТС-06)


```

1 |===== BLOCK 3: WORKSPACES =====
2
3 --- TC-07: Get all workspaces ---
4 [{"id":"cmpfwg1d300026q1sq4hy5wzb","number":"OS","name":null,"type":"OPEN_SPACE","location":"PODIL"}
5 ...total workspaces: 84
6 HTTP 200
7
8 --- TC-08: Filter by location=PODIL ---
9 Items returned: 28
10 Locations present:      28 "location":"PODIL"
11 HTTP 200
12
13 --- TC-09: Filter by date (isBooked field) ---
14 [{"id":"cmpfwg1d300026q1sq4hy5wzb","number":"OS","name":null,"type":"OPEN_SPACE","location":"PODIL"}
15 isBooked fields present: 84
16 HTTP 200
17 Fixed Desk ID: cmpfwg1d300036q1s9v0z6n6l
18 Meeting Room ID: cmpfwg1d4000r6q1sh0kazgwx
19
20 ===== BLOCK 4: BOOKINGS =====
21
22 --- TC-10: Book Fixed Desk (DAY plan) ---
23 {"id":"cmptsm4410003p2rttrzxqe4v","userId":"cmpts17g20000p2rt7u9rr2ii","workspaceId":"cmpfwg1d30003sq
24 HTTP 201
25 Booking ID captured: cmptsm4410003p2rttrzxqe4v
26
27 --- TC-11: Conflict (same workspace + same dates) ---
28 {"error":"This workspace is already booked for the selected period"}
29 HTTP 409
30
31 --- TC-12: Meeting Room + HOUR plan ---
32 {"id":"cmptsm4ae0007p2rtjoy4vrjt","userId":"cmpts17g20000p2rt7u9rr2ii","workspaceId":"cmpfwg1d4000r6q1sh0kazgwx"}
33 HTTP 201
34
35 --- TC-13: Meeting Room + DAY plan (should fail) ---
36 {"error":"Meeting rooms can only be booked by the hour"}
37 HTTP 400
38
39 --- TC-14: Fixed Desk + HOUR plan (should fail) ---
40 {"error":"Hourly booking is only available for meeting rooms"}
41 HTTP 400
42
43 --- TC-15: Cancel booking from TC-10 ---
44 {"id":"cmptsm4410003p2rttrzxqe4v","userId":"cmpts17g20000p2rt7u9rr2ii","workspaceId":"cmpfwg1d30003sq
45 HTTP 200
46
47 --- TC-16: My bookings ---
48 [{"id":"cmptsm4ae0007p2rtjoy4vrjt","userId":"cmpts17g20000p2rt7u9rr2ii","workspaceId":"cmpfwg1d4000r6q1sh0kazgwx"}
49
50 Admin token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
51
52 ===== BLOCK 5: ADMIN =====
53
54 --- TC-17: Admin stats ---
55 {"totalRevenue":307,"monthlyRevenue":307,"activeBookings":2,"totalBookings":10,"totalMembers":8,"bookings":10}
56 HTTP 200
57
58 --- TC-18: Admin users list ---
59 [{"id":"cmpts17g20000p2rt7u9rr2ii","name":"*****","email":"test@example.com","password":"123456"}]
60
61 --- TC-19: Change workspace status ---
62 Using workspace ID: cmpfwg1d300026q1sq4hy5wzb
63 {"id":"cmpfwg1d300026q1sq4hy5wzb","number":"OS","name":null,"type":"OPEN_SPACE","location":"PODIL","status":"BOOKED"}
64 HTTP 200

```

Рисунок 2.27 – Результати API тестування: блоки робочих місць, бронювань та адмін-функцій (ТС-07 – ТС-19)

2.5.2 Функціональне тестування інтерфейсу

Тестування клієнтського інтерфейсу проводилось шляхом ручного проходження всіх сценаріїв взаємодії у браузері Chrome. Перевірялась коректність відображення повідомлень про помилки, відповідність поведінки інтерфейсу очікуваним сценаріям та коректність відображення результатів операцій.

Тестування форми реєстрації підтвердило коректну роботу клієнтської валідації: при введенні паролю коротше шести символів браузер відображає підказку ще до відправки форми на сервер. При введенні некоректного формату email система виводить повідомлення «Invalid email address», а при спробі реєстрації з вже використаним email – «Email already in use». Скріншоти валідації форми реєстрації наведено на рисунку 2.28. При спробі входу з невірним паролем форма авторизації відображає повідомлення «Invalid email or password» (рисунок 2.29).

The figure displays three variations of the 'Sign Up' form, each with a different validation error highlighted in a black banner at the top:

- Left Screenshot:** The error banner reads 'Invalid email address'. The form fields are: First Name (Natya), Phone Number (+380 000000000), Email (nastya.timofieieva@gmail.a), and Password (masked with dots). The 'CREATING ACCOUNT...' button is active.
- Middle Screenshot:** The error banner reads 'Email already in use'. The form fields are: First Name (Natya Second), Phone Number (+380 000000000), Email (nastya.timofieieva@gmail.com), and Password (masked with dots). The 'CREATING ACCOUNT...' button is active.
- Right Screenshot:** The error banner is not present. The form fields are: First Name (Natya), Phone Number (+380 000000000), Email (nastya.timofieieva@gmail.com), and Password (masked with dots). A tooltip message appears over the password field: 'Текст должен быть не короче 6 симв. Длина текста сейчас: 1 символ.' (Text must be at least 6 symbols. Current text length: 1 symbol). The 'SIGN UP' button is active.

Below each form, there is a link to 'Terms & Privacy Policy' and a 'Log in' link for users who already have an account.

Рисунок 2.28 – Клієнтська валідація форми реєстрації: короткий пароль, некоректний email, дублікат email

The figure shows the 'Log In' form with the following elements:

- Error Banner:** A black banner at the top reads 'Invalid email or password'.
- Form Fields:** Email (nastya.timofieieva@gmail.com) and Password (masked with dots).
- Buttons:** A 'LOGGING IN...' button is at the bottom.
- Links:** A 'Forgot password?' link is located below the password field, and a 'Sign up' link is at the very bottom for users without an account.

Рисунок 2.29 – Повідомлення про помилку авторизації при невірному паролі

Тестування повного циклу бронювання підтвердило коректну роботу покрокового флоу. Після проходження всіх кроків та підтвердження бронювання система відображає сторінку з повідомленням «BOOKING CONFIRMED. SEE YOU AT SPARK.» та детальним чеком: локація Spark Podil, тип Fixed Desk, план Week Pass, дата 12 June 2026 – 19 June 2026, місце B2, вартість 4400 грн (рисунок 2.30).

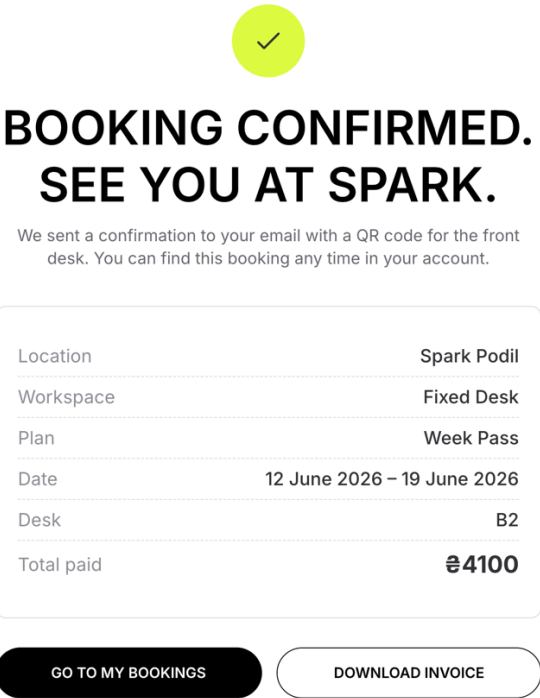


Рисунок 2.30 – Сторінка успішного підтвердження бронювання

В особистому кабінеті перевірено відображення списку бронювань з фільтрами «Upcoming», «Past», «Cancelled» та «All». Активне бронювання коректно відображається у вкладці «Upcoming 3» (рисунок 2.31).

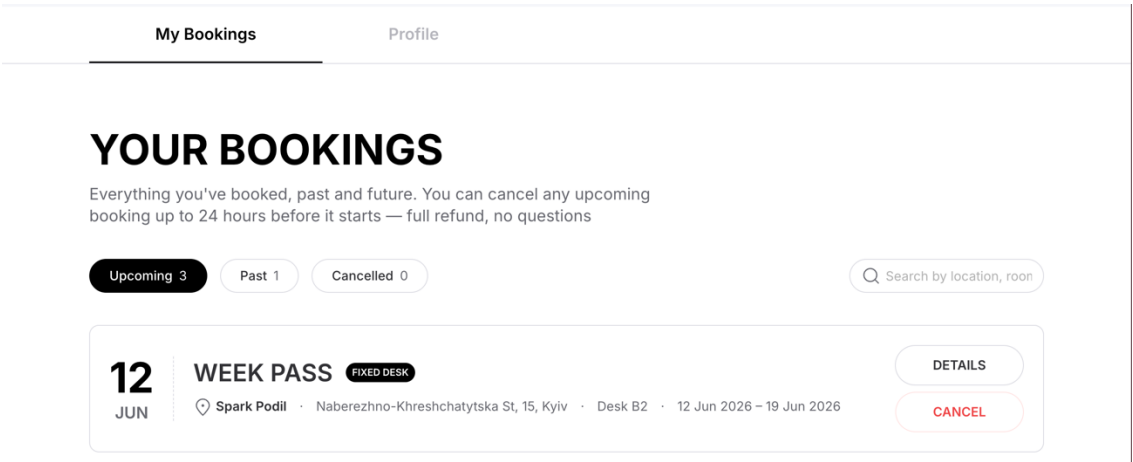


Рисунок 2.31 – Особистий кабінет резидента зі списком бронювань

При натисканні кнопки «Cancel» відкривається модальне вікно підтвердження з деталями бронювання та сумою повернення коштів (рисунок 2.32). Після скасування бронювання лічильник «Upcoming» змінився на 2, а «Cancelled» – на 1, що підтверджує коректне оновлення стану інтерфейсу без перезавантаження сторінки (рисунок 2.33).

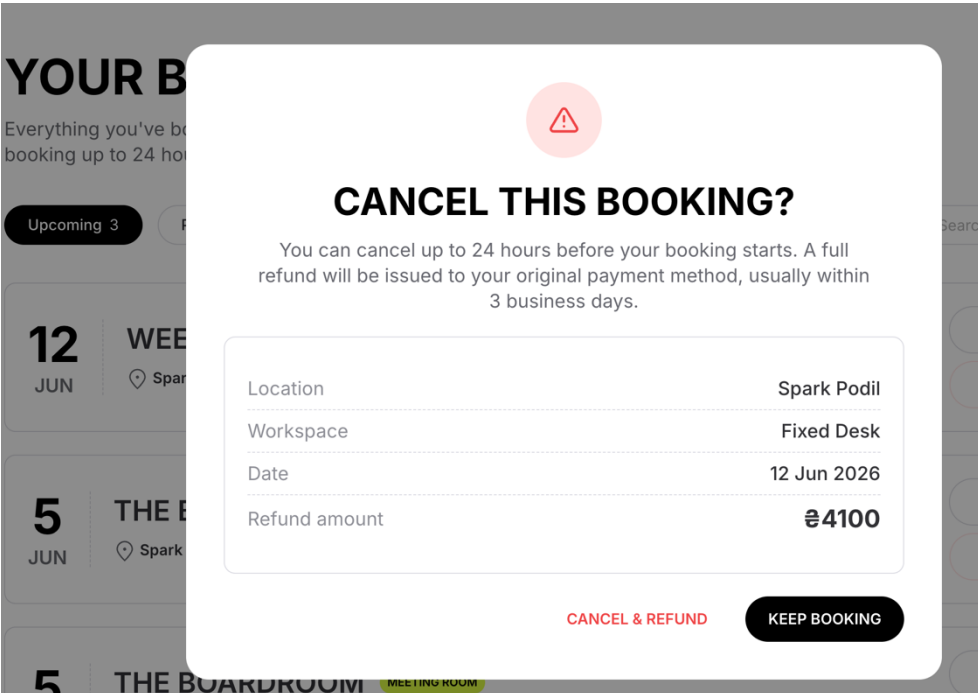


Рисунок 2.32 – Модальне вікно підтвердження скасування бронювання

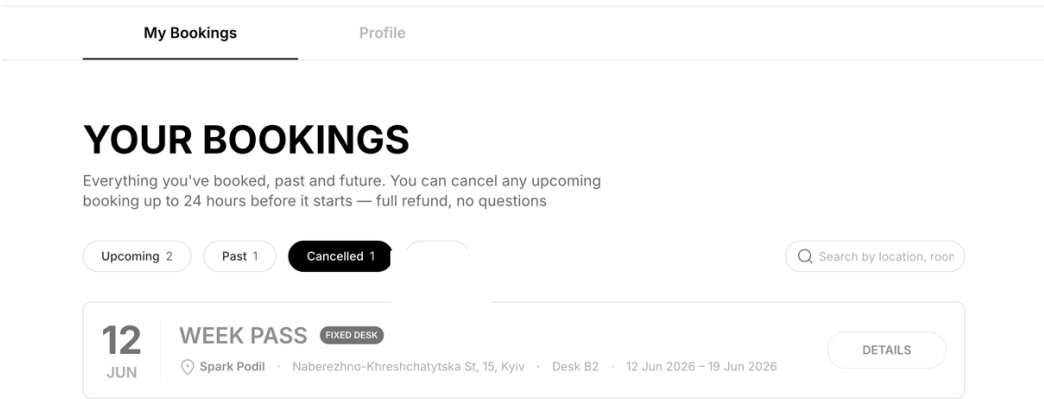


Рисунок 2.33 – Особистий кабінет резидента після скасування бронювання

Тестування адміністративної панелі підтвердило коректну роботу розділу управління резидентами. Таблиця відображає список учасників з роллю, кількістю бронювань та загальною сумою витрат. При натисканні кнопки «Bookings» для

конкретного резидента відкривається модальне вікно зі списком усіх його бронювань та статусами. На рисунку 2.34 показано стан до скасування бронювання – модальне вікно відображає 2 активних бронювання на суму 4100 та 430 грн, та стан після скасування адміністратором – кількість активних бронювань змінилась на 1.

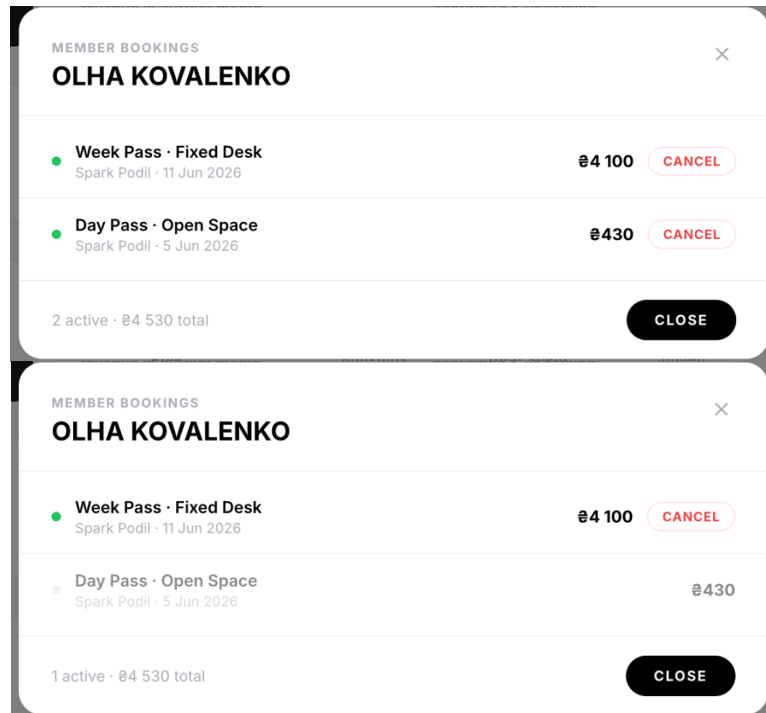


Рисунок 2.34 – Скасування бронювання резидента в адміністративній панелі: до та після

2.5.3 Тестування безпеки

Тестування безпеки спрямоване на перевірку механізмів автентифікації, авторизації та захисту даних. Результати наведено у таблиці 2.2 та на рисунку 2.35.

Таблиця 2.2 – Результати тестування безпеки системи

№	Опис тесту	Очікуваний результат	Фактичний результат	Статус
SEC-01	Запит без токена	HTTP 401, «No token provided»	HTTP 401	✓
SEC-02	Підроблений токен	HTTP 401, «Invalid or expired token»	HTTP 401	✓
SEC-03	Резидент → адмін-маршрут	HTTP 403, «Admin access required»	HTTP 403	✓

Продовження табл. 2.2

SEC-04	Скасування чужого бронювання	HTTP 403, «Not authorized»	HTTP 403	✓
SEC-05	Перевірка хешування паролів	Хеші bcrypt (\$2a\$10\$...) у БД	Хеші \$2a\$10\$ підтверджено	✓
SEC-06	SQL-ін'єкція через email	HTTP 400, запит відхилено	HTTP 400	✓
SEC-07	Доступ до /admin без ролі ADMIN	Перенаправлення на головну	Перенаправлення виконано	✓

```

1
2 SEC-01 – Request without token
3
4 {"error": "No token provided"}
5 HTTP STATUS: 401
6
7
8 SEC-02 – Fake / corrupted token
9
10 {"error": "Invalid or expired token"}
11 HTTP STATUS: 401
12
13
14 SEC-03 – Resident accessing admin route
15
16 Token owner: test@example.com (role: RESIDENT)
17 {"error": "Admin access required"}
18 HTTP STATUS: 403
19
20
21 SEC-04 – Cancel another user's booking
22
23 Booking owner: sec04user2@example.com
24 Attacker: test@example.com
25 Booking ID: cmptsw6rx000hp2rtjgsfseqn
26 {"error": "Not authorized"}
27 HTTP STATUS: 403
28
29
30 SEC-05 – Passwords stored as bcrypt hashes
31
32 email | passwordHash (prefix)
33 -----+-----
34 admin@spark.co | $2a$10$0mTmx7.0hB17H...
35 olha@example.com | $2a$10$8axwDmNbMwNk...
36 evgenijserepa@gmail.com | $2a$10$2bwYLoFGpvaAM...
37 test@test.com | $2a$10$qMmpX1qTkAdwd...
38 test2@test.com | $2a$10$69nIVCu9L8/o7...
39
40
41
42 SEC-06 – SQL injection via email field
43
44 {"error": "Invalid credentials format"}
45 HTTP STATUS: 400
46
47
48 SEC-07 – Admin page without admin role
49
50 Manual browser test – open /admin while logged in as RESIDENT.
51 Expected: immediate redirect to / (implemented in admin/page.tsx
52 via useEffect: if (user && user.role !== 'ADMIN') router.push('/'))

```

Рисунок 2.35 – Результати тестування безпеки: коди відповідей 401, 403 та хеші паролів у БД

SEC-01 перевіряє поведінку при зверненні до захищеного маршруту без токена. Запит надіслано до GET /api/bookings/my без заголовку Authorization –

сервер повернув статус 401 з повідомленням «No token provided», не допустивши звернення до бази даних.

SEC-02 перевіряє захист від підроблених JWT-токенів [17]. У заголовку передано довільний рядок замість валідного токена – бібліотека `jsonwebtoken` відхилила його при верифікації підпису і повернула статус 401, підтверджуючи неможливість підробки без знання серверного секретного ключа.

SEC-03 перевіряє розмежування прав між роллю `RESIDENT` і `ADMIN`. Запит до `GET /api/admin/stats` з токеном резидента повернув статус 403 з повідомленням «Admin access required» – `middleware requireAdmin` відхилив запит до виконання основної логіки маршруту.

SEC-04 перевіряє захист від скасування чужого бронювання. Другий тестовий користувач спробував скасувати бронювання першого – сервер зіставив ідентифікатор власника з токеном запитувача і повернув статус 403 «Not authorized».

SEC-05 перевіряє зберігання паролів у базі даних. Після реєстрації тестових облікових записів усі записи у полі `passwordHash` містять рядки з префіксом `$2a$10$`, що відповідає формату `bcrypt` з коефіцієнтом складності 10. Паролі у відкритому вигляді відсутні.

SEC-06 перевіряє стійкість до SQL-ін'єкцій. Передано рядок «' OR 1=1 --» як значення `email` – бібліотека `Zod` [22] відхилила запит на рівні валідації і повернула статус 400. Навіть при обході валідації `Prisma ORM` [24] використовує параметризовані запити, що унеможлиблює виконання довільного SQL-коду.

SEC-07 перевіряє захист маршруту `/admin` на клієнті. При спробі прямого переходу з обліковим записом резидента клієнтський компонент перевіряє роль зі сховища `Zustand` [19] і виконує перенаправлення на головну сторінку без відображення вмісту панелі.

Всі сім тест-кейсів пройшли успішно.

2.5.4 Тестування продуктивності та адаптивності

Тестування продуктивності проводилось за допомогою двох інструментів: вбудованого в Chrome DevTools Google Lighthouse [33] та онлайн-сервісу PageSpeed Insights [34]. Результати тестування зведено у таблицях 2.3 та 2.4.

Таблиця 2.3 – Результати тестування Google Lighthouse

Сторінка	Продуктивність	Доступність	Best Practices	SEO
/locations	94	94	100	100
/book	98	89	100	100

Таблиця 2.4 – Результати тестування PageSpeed Insights

Сторінка	Пристрій	Продуктивність	Доступність	Best Practices	SEO
/locations	Desktop	99	94	100	100
/locations	Mobile	84	94	100	100
/book	Desktop	98	89	100	100
/book	Mobile	95	89	100	100

Показники Best Practices та SEO становлять 100 балів на всіх перевірених сторінках. Продуктивність у десктопному режимі перевищує 98 балів, що свідчить про ефективність статичного експорту Next.js [11]. Дещо нижчий показник продуктивності сторінки /locations на мобільних пристроях (84 бали) пояснюється завантаженням зображень галереї з великою роздільною здатністю – цей показник може бути покращений шляхом застосування формату WebP та атрибуту lazy loading для зображень нижче лінії згортання. Загалом отримані результати відповідають сучасним стандартам якості веб-застосунків і підтверджують, що обраний технологічний стек забезпечує прийнятну швидкодію як для десктопних, так і для мобільних користувачів. Результати Lighthouse наведено на рисунку 2.36, PageSpeed Insights – на рисунку 2.37.

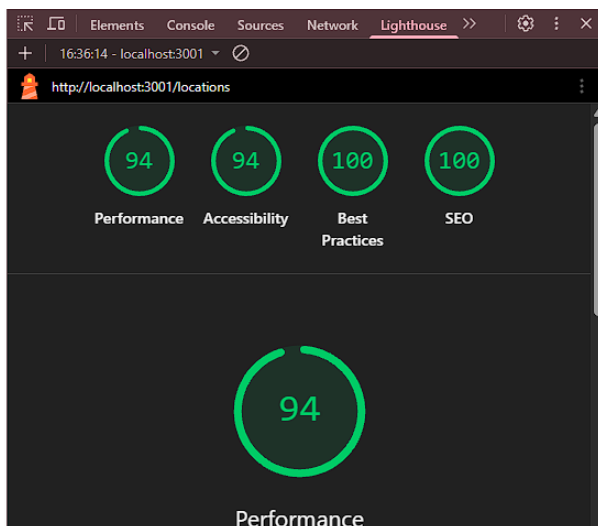


Рисунок 2.36 – Результати тестування продуктивності (Google Lighthouse, сторінка /locations)

PageSpeed Insights

Отчет от 31 мая 2026 г., 16:43:15

<https://coworking-fe.vercel.app/locations>

Анализировать

Мобильные устройства **Компьютер**

Определяйте фактическую производительность сайта

Нет данных

Выявляйте проблемы с производительностью

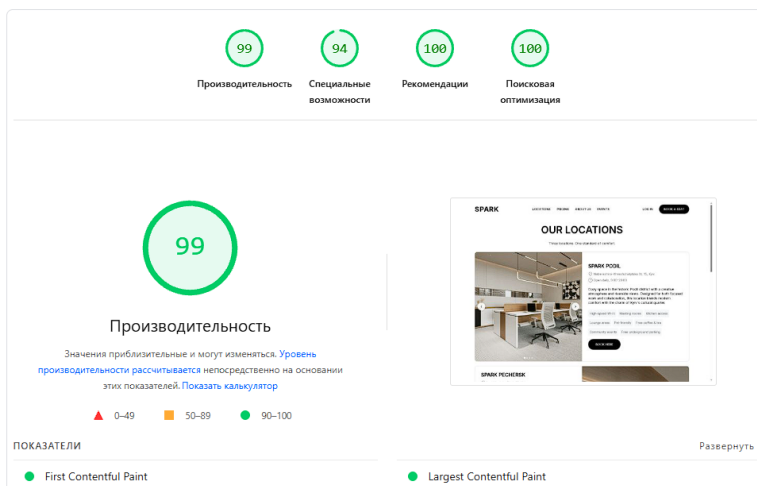


Рисунок 2.37 – Результати тестування PageSpeed Insights (сторінка /locations, Desktop)

Аналіз вкладки Network у Chrome DevTools [35] підтвердив прийнятний час завантаження головної сторінки: 33 запити, 25.8 КБ переданих даних, загальний час завершення – 2.70 секунди (рисунок 2.38).

Name	Status	Type	Initiator	Size	Time
book	200	fetch	requests.js:1	0.1 kB	40 ms
login	200	fetch	requests.js:1	0.2 kB	34 ms
events	200	fetch	requests.js:1	0.2 kB	62 ms
about	200	fetch	requests.js:1	0.1 kB	34 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	0.6 kB	34 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	0.7 kB	33 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	0.6 kB	34 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	0.6 kB	37 ms
pricing	200	fetch	requests.js:1	0.1 kB	34 ms
locations	200	fetch	requests.js:1	0.2 kB	35 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.1 kB	35 ms
_next_index.txt?_rsc=yIp22O3...	200	fetch	requests.js:1	1.0 kB	40 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	1.2 kB	33 ms
_next_tree.txt?_rsc=5CB68i4p...	200	fetch	requests.js:1	0.7 kB	33 ms
_next.book.txt?_rsc=arIoWpza...	200	fetch	requests.js:1	0.4 kB	33 ms
_next.book_PAGE_.txt?_rsc=...	200	fetch	requests.js:1	1.0 kB	34 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.1 kB	34 ms
_next.login.txt?_rsc=HQgJTOe...	200	fetch	requests.js:1	0.4 kB	34 ms
_next.login_PAGE_.txt?_rsc=I...	200	fetch	requests.js:1	1.1 kB	40 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.0 kB	39 ms
_next.events.txt?_rsc=LUOzXu...	200	fetch	requests.js:1	0.4 kB	43 ms
_next.events_PAGE_.txt?_rsc=...	200	fetch	requests.js:1	0.8 kB	34 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.1 kB	39 ms
_next.about.txt?_rsc=n4tDNspj...	200	fetch	requests.js:1	0.5 kB	35 ms
_next.about_PAGE_.txt?_rsc=...	200	fetch	requests.js:1	1.0 kB	34 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.0 kB	35 ms
_next.pricing.txt?_rsc=upn3sbf...	200	fetch	requests.js:1	0.4 kB	35 ms
_next.pricing_PAGE_.txt?_rsc=...	200	fetch	requests.js:1	1.6 kB	38 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.0 kB	35 ms
_next.locations.txt?_rsc=_F-Mx...	200	fetch	requests.js:1	0.4 kB	38 ms
_next.locations_PAGE_.txt?_r...	200	fetch	requests.js:1	1.0 kB	36 ms
_next_head.txt?_rsc=7h4NYy5...	200	fetch	requests.js:1	1.0 kB	34 ms
_next_PAGE_.txt?_rsc=8nTjlt...	200	fetch	requests.js:1	3.0 kB	38 ms

33 / 77 requests | 25.8 kB / 13,459 kB transferred | 34.2 kB / 18,257 kB resources | Finish: 2.70 s

Рисунок 2.38 – Аналіз завантаження сторінки у вкладці Network (Chrome DevTools)

Тестування адаптивності проводилось у режимі емуляції пристроїв Chrome DevTools для трьох типорозмірів екранів: iPhone SE (375 px), iPhone 14 Pro Max (430 px) та iPad Air (820 px). На всіх перевірених розмірах інтерфейс відображається коректно: навігаційне меню трансформується у мобільне «бургер»-меню, картки локацій перебудовуються у вертикальний список, форми зберігають читабельність (рисунок 2.39).

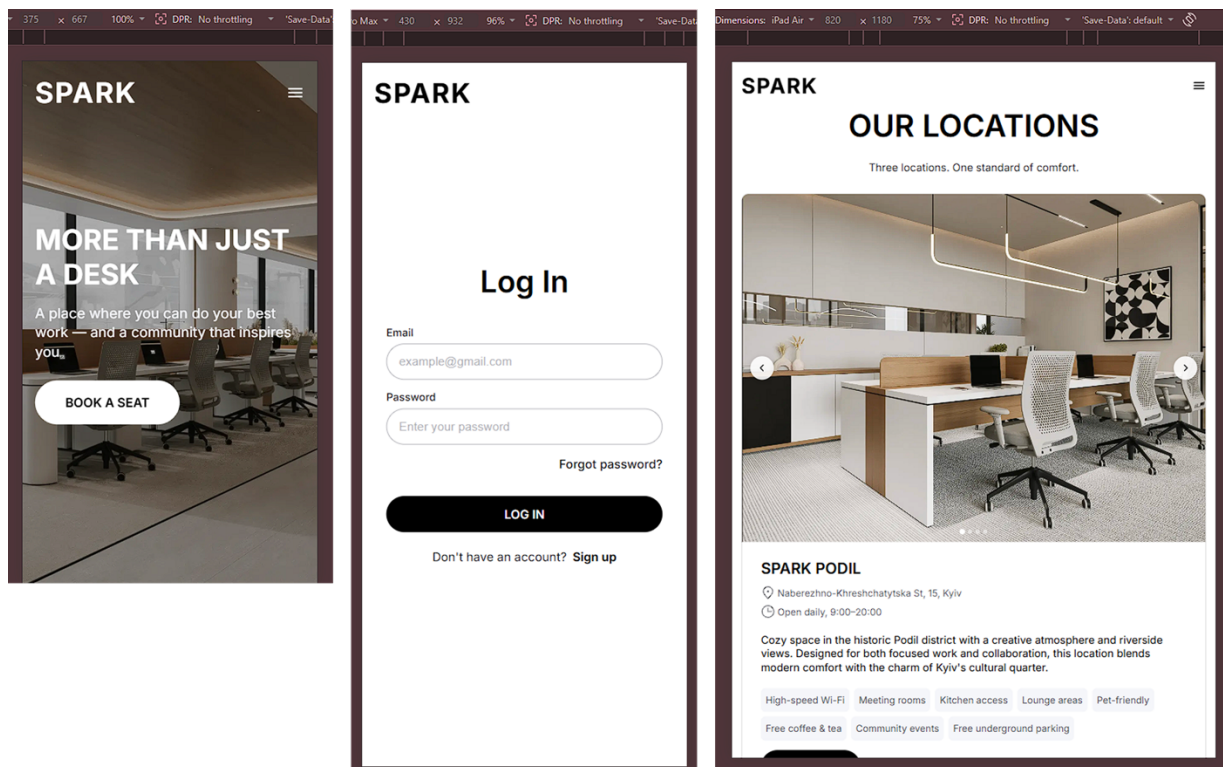


Рисунок 2.39 – Адаптивне відображення інтерфейсу: iPhone SE (375 px), iPhone 14 Pro Max (430 px), iPad Air (820 px)

Висновки до розділу:

У другому розділі виконано проєктування та програмну реалізацію системи автоматизації бронювання і організації робочих просторів коворкінгу. Обґрунтовано вибір технологічного стеку: Next.js 16 та React 19 для клієнтської частини, Node.js з Express для серверної, PostgreSQL та Prisma ORM для бази даних, JWT і bcrypt для автентифікації. Спроектовано реляційну базу даних, що охоплює три таблиці та шість enum-типів; структура реалізована через дві послідовні міграції Prisma Migrate. Реалізовано шість функціональних модулів системи: автентифікації, перегляду робочих місць з інтерактивною SVG-картою поверху, транзакційного бронювання, особистого кабінету резидента та адміністративної панелі зі зведеною статистикою. Складено інструкцію користувача, що охоплює повний цикл взаємодії – від базових сторінок до покрокового флоу бронювання та управління адміністративною панеллю. За результатами функціонального тестування 19 тест-кейсів API та 7 тест-кейсів безпеки всі перевірки пройшли

успішно. Тестування продуктивності засобами Google Lighthouse та PageSpeed Insights підтвердило високі показники якості: продуктивність у десктопному режимі становить 98–99 балів, SEO та Best Practices – 100 балів на всіх перевірених сторінках.

ВИСНОВКИ

У ході дослідження було проведено комплексний аналіз предметної галузі, спроектовано та розроблено веб-застосунок автоматизації бронювання і організації робочих просторів коворкінгу на основі сучасних веб-технологій. За результатами виконаної роботи сформульовано такі висновки:

1. Проведено аналіз предметної галузі та встановлено, що більшість коворкінг-центрів функціонують без спеціалізованого програмного забезпечення, спираючись на електронні таблиці, месенджери та ручні операції. Виявлено три системні недоліки поточного стану: відсутність транзакційного механізму перевірки доступності місць, відсутність цифрового інтерфейсу для клієнтів та надмірна залежність операційного процесу від адміністративного персоналу.

2. Засобами функціонального моделювання у нотації BPMN побудовано моделі поточного (AS IS) та вдосконаленого (TO BE) бізнес-процесів бронювання, що наочно демонструють перехід від ручного до повністю автоматизованого операційного циклу. Порівняльний аналіз трьох сайтів-аналогів підтвердив відсутність на ринку рішення, яке б поєднувало онлайн-бронювання, інтерактивну карту поверху та особистий кабінет резидента в єдиній доступній платформі.

3. Спроектовано та реалізовано реляційну базу даних на основі PostgreSQL та Prisma ORM. Транзакційний механізм перевірки конфліктів бронювань унеможливорює паралельне резервування одного місця двома користувачами.

4. Розроблено повнофункціональний веб-застосунок, що складається з клієнтської частини на Next.js 16 з React 19 та серверної частини на Node.js з Express. Реалізовано шість функціональних модулів: автентифікації на основі JWT та bcrypt, перегляду робочих місць, інтерактивної SVG-карти поверху для вибору Fixed Desk, транзакційного бронювання, особистого кабінету резидента та

адміністративної панелі зі зведеною статистикою. Застосунок розгорнуто на Vercel (клієнтська частина) та Railway (серверна частина).

5. Проведено комплексне тестування розробленого програмного продукту. Функціональне тестування охопило 19 API тест-кейсів та перевірку клієнтського інтерфейсу – всі сценарії пройшли успішно. Тестування безпеки підтвердило коректну роботу механізмів автентифікації та авторизації: несанкціоновані запити повертають коди 401 та 403, паролі зберігаються виключно у вигляді bcrypt-хешів, система захищена від SQL-ін'єкцій засобами Prisma ORM. Тестування продуктивності засобами Google Lighthouse показало 98–99 балів у десктопному режимі, 100 балів за критеріями Best Practices та SEO на всіх перевірених сторінках. Перевірено адаптивність інтерфейсу для екранів від 375 px (iPhone SE) до 820 px (iPad Air) – інтерфейс коректно відображається на всіх перевірених розмірах пристроїв, навігаційне меню трансформується у мобільний формат.

6. Практична цінність розробленої системи полягає у скороченні операційного навантаження на адміністративний персонал, усуненні ризику подвійного бронювання та забезпеченні клієнтам цілодобового доступу до сервісу без участі адміністратора.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- 1 Spinuzzi C. Working alone together: coworking as emergent collaborative activity. *Journal of Business and Technical Communication*, 2012. 441 с.
URL: https://www.researchgate.net/publication/258143922_Working_Alone_Together_Coworking_as_Emergent_Collaborative_Activity (дата звернення: 06.04.2026).
- 2 Flexible office space: now and next | Savills Impacts. *Savills Impacts*.
URL: <https://impacts.savills.com/social-change/flexible-office-space-now-and-next.html> (дата звернення: 06.04.2026).
- 3 Discover 2023's coworking space trends: demand, expectations & plans. *Deskmag* | *The Coworking Magazine*.
URL: <https://www.deskmag.com/en/coworking-news/2023-coworking-space-business-trends-demand-expectations-and-plans-market-survey-report> (дата звернення: 05.06.2026).
- 4 Аку С. 2024 coworking market research: UCA announces survey launch. *Коворкінг України*.
URL: <https://www.coworkingassociation.org.ua/en/post/2024-coworking-market-research-uca-announces-survey-launch> (дата звернення: 05.04.2026).
- 5 Осовська Г. В., Масловська Л. Ц., Осовський О. А. Менеджмент організацій : навч. посіб. Київ : Кондор, 2024. 366 с.
- 6 Google workspace |. *Google Workspace*.
URL: <https://workspace.google.com/intl/uk/> (дата звернення: 05.06.2026).
- 7 Google sheets | google workspace. *Google Workspace*.
URL: <https://workspace.google.com/products/sheets/> (дата звернення: 05.06.2026).
- 8 Google Formulare: | Google Workspace. *Google Workspace*.
URL: <https://workspace.google.com/products/forms/> (дата звернення: 05.06.2026).
- 9 BPMN – вікіпедія. *Вікіпедія*.
URL: <https://uk.wikipedia.org/wiki/BPMN> (дата звернення: 05.06.2026).

10 REST - glossary | MDN. *MDN Web Docs*.
URL: <https://developer.mozilla.org/en-US/docs/Glossary/REST> (дата звернення: 05.06.2026).

11 Vercel. Next.js docs | next.js. *Next.js by Vercel - The React Framework*.
URL: <https://nextjs.org/docs> (дата звернення: 05.06.2026).

12 Index of /docs/. *Node.js - Run JavaScript Everywhere*.
URL: <https://nodejs.org/docs/> (дата звернення: 05.06.2026).

13 Nat – коворкінг | робочий простір для тих, кому не байдуже де працювати. *Nat – коворкінг | робочий простір для тих, кому не байдуже де працювати*. URL: <https://www.natcoworking.com/> (дата звернення: 28.04.2026).

14 PLATFORMA - простір для роботи та розвитку | Коворкінг в Києві *Platforma. PLATFORMA - простір для роботи та розвитку | Коворкінг в Києві Platforma*. URL: <https://www.coworkingplatforma.com/> (дата звернення: 29.04.2026).

15 Regus. *Serviced Office Space, Coworking and Meeting Rooms* | Regus.
URL: <https://www.regus.com/en> (дата звернення: 29.04.2026).

16 Space management and bookings. *Nexodus*.
URL: <https://nexodus.com/space-management-and-bookings/> (дата звернення: 05.06.2026).

17 RFC 7519: JSON web token (JWT). *IETF Datatracker*.
URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 15.05.2026).

18 React. *React*. URL: <https://react.dev/> (дата звернення: 05.06.2026).

19 Introduction - zustand. *Poimandres documentation*.
URL: <https://zustand.docs.pmnd.rs/learn/getting-started/introduction> (дата звернення: 08.05.2026).

20 Tailwind CSS - rapidly build modern websites without ever leaving your HTML. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: <https://tailwindcss.com/> (дата звернення: 13.05.2026).

- 21 W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/REACT/react_css_modules.asp (дата звернення: 13.05.2026).
- 22 Intro | zod. *Zod*. URL: <https://zod.dev/> (дата звернення: 12.05.2026).
- 23 PostgreSQL: documentation. *PostgreSQL: The world's most advanced open source database*. URL: <https://www.postgresql.org/docs/> (дата звернення: 02.05.2026).
- 24 Get started with Prisma. *Prisma | Serverless Postgres, Type-Safe ORM, and Database Tools*. URL: <https://www.prisma.io/docs> (дата звернення: 02.05.2026).
- 25 Zahlungen. *Stripe-Dokumentation*. URL: <https://docs.stripe.com/payments> (дата звернення: 05.06.2026).
- 26 Nodemailer. *Nodemailer*. URL: <https://nodemailer.com/> (дата звернення: 05.06.2026).
- 27 Vercel: Build and deploy the best web experiences with the AI Cloud – Vercel. *Vercel*. URL: <https://vercel.com/> (дата звернення: 06.05.2026).
- 28 Railway docs. *Railway Docs*. URL: <https://docs.railway.com/> (дата звернення: 05.06.2026).
- 29 The starting point for learning TypeScript. *TypeScript: JavaScript With Syntax For Types*. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 05.06.2026).
- 30 ECMA-262 - ecma international. *Ecma International*. URL: <https://ecma-international.org/publications-and-standards/standards/ecma-262/> (дата звернення: 05.06.2026).
- 31 OWASP top ten web application security risks | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 05.06.2026).
- 32 Express.js · Node.js web application framework. *Express.js*. URL: <https://expressjs.com/en/> (дата звернення: 13.05.2026).

33 Introduction to lighthouse | chrome for developers. *Chrome for Developers*. URL: <https://developer.chrome.com/docs/lighthouse/overview> (дата звернення: 05.06.2026).

34 Get started with the pagespeed insights API | google for developers. *Google for Developers*. URL: <https://developers.google.com/speed/docs/insights/v5/get-started> (дата звернення: 05.06.2026).

35 Chrome devtools | chrome for developers. *Chrome for Developers*. URL: <https://developer.chrome.com/docs/devtools> (дата звернення: 05.06.2026).

36 ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с.(Інформація та документація).

37 ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с.(Інформація та документація).

38 ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с.(Інформація та документація).