

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебплатформа планування та контролю клінінгової компанії»

Виконала студентка групи ВТ-22-1
ШУЛЯК Яна Іванівна

Керівник роботи:
НЕРОДА Сергій Іванович

Рецензент:
ГРОМСЬКИЙ Олександр Олександрович

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ШУЛЯК Яна Іванівна**

Керівник роботи: **НЕРОДА Сергій Іванович**

Тема роботи: **«Вебплатформа планування та контролю клінінгової компанії»**,

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

Вихідні дані для роботи: **об'єктом дослідження** є організаційні та технічні процеси управління операційною діяльністю та клієнтським сервісом у сфері клінінгових послуг, а **предметом дослідження** є програмні засоби та підходи до розробки вебплатформи для автоматизації оформлення й обробки замовлень, планування роботи майстрів, контролю виконання клінінгових послуг, обробки оплати та взаємодії між користувачами системи

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Нерода С.І.	01.03.2026р.	01.03.2026р.
2	Нерода С.І.	01.03.2026р.	01.03.2026р.
3	Нерода С.І.	01.03.2026р.	01.03.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебплатформи планування та контролю клінінгової компанії та пояснювальної записки. Пояснювальна записка до випускної роботи містить 105 сторінок, 38 ілюстрацій, 20 таблиць, 30 використаних джерел та 27 сторінок додатків.

Метою випускної кваліфікаційної роботи є розробка та програмна реалізація вебплатформи планування та контролю діяльності клінінгової компанії, яка забезпечує автоматизацію процесів оформлення та обробки замовлень, управління персоналом, планування графіків і контролю надання послуг. Система покликана об'єднати основні бізнес-процеси компанії в єдиному середовищі, забезпечити зручну взаємодію між клієнтами, працівниками та адміністраторами, підвищити ефективність управління ресурсами та якість обслуговування клієнтів.

Вебплатформа побудована на основі клієнт-серверної архітектури з використанням технології SPA, а взаємодія між компонентами системи здійснюється за принципом REST API для забезпечення асинхронного обміну даними у форматі JSON. Обрана архітектура дає змогу відокремити бізнес-логіку сервера від представлення даних, забезпечуючи високу продуктивність та адаптивність інтерфейсу. База даних централізовано зберігає інформацію про послуги, а також про користувачів та їхні замовлення, що дозволить у подальшому швидко масштабувати систему за відповідної потреби.

Вебплатформа має адаптивний інтерфейс із послідовною навігацією, що дає змогу оформлювати замовлення на послуги, переглядати історію замовлень та керувати інформацією в профілі.

КЛЮЧОВІ СЛОВА: АВТОРИЗАЦІЯ, ВЕБПЛАТФОРМА, КЛІНІНГОВІ ПОСЛУГИ, СИСТЕМА ПЛАНУВАННЯ ТА УПРАВЛІННЯ, SPA.

					ІПЗ.КР.Б – 121 – 26 – ПЗ					
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа планування та контролю клінінгової компанії Пояснювальна записка			Літ.	Арк.	Аркушів
Розроб.		Шуляк Я.І.								
Керівник		Нерода С.І.							3	105
								Житомирська політехніка, група ВТ-22-1		
Рецензент		Громський О.О.								
Зав. каф.		Вакалюк Т.А.								

ABSTRACT

The bachelor's qualification thesis consists of a web platform for planning and managing a cleaning company and an explanatory note. The explanatory note of the bachelor's qualification thesis comprises 105 pages, 38 figures, 20 tables, 30 references, and 27 pages of appendices.

The purpose of the bachelor's qualification thesis is to develop and implement a web platform for planning and managing the activities of a cleaning company, which provides automation of order placement and processing, personnel management, schedule planning, and service delivery control. The system is designed to integrate the company's core business processes into a single environment, ensure convenient interaction between clients, employees, and administrators, improve resource management efficiency, and enhance customer service quality.

The web platform is based on a client-server architecture using Single Page Application (SPA) technology. Interaction between system components is implemented through a REST API to ensure asynchronous data exchange in JSON format. The chosen architecture enables the separation of server-side business logic from data presentation, ensuring high performance and interface responsiveness. The database centrally stores information about services, users, and their orders, which will facilitate the system's scalability to meet future business requirements.

The web platform features a logical, user-friendly, and intuitive interface that enables users to easily place service orders, view order history, and manage profile information.

KEYWORDS: AUTHORIZATION, WEB PLATFORM, CLEANING SERVICES, PLANNING AND MANAGEMENT SYSTEM, SPA.

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

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ.....	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури вебплатформи планування та контролю клінінгової компанії	19
Висновки до першого розділу	22
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ	24
2.1 Вимоги до системи.....	24
2.2 Розробка бази даних	35
2.3 Проєктування та реалізація алгоритмів роботи системи	38
2.4 Реалізація функціоналу вебплатформи.....	44
Висновки до другого розділу.....	48
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ.....	49
3.1 Порядок встановлення та налаштування параметрів системи	49
3.2 Структура інтерфейсу вебплатформи	52
3.3 Тестування роботи програмного продукту	67
Висновки до третього розділу	72
ВИСНОВКИ.....	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	76
ДОДАТКИ.....	78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SPA – Single Page Application

API – Application Programming Interface

JSON – JavaScript Object Notation

СУБД – система управління базами даних.

JWT – JSON Web Token

SQL – Structured Query Language

UML – Unified Modeling Language

HTTPS – HyperText Transfer Protocol Secure

HTTP – HyperText Transfer Protocol

URL – Uniform Resource Locator

REST – Representational State Transfer

ER – Entity Relationship

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

ВСТУП

Актуальність теми визначається потребою вдосконалення управління діяльністю клінінгових компаній в умовах зростання попиту на послуги та поширення цифрових форм взаємодії із замовниками. Сучасний споживач очікує швидкого оформлення заявки, прозорого розрахунку вартості, зручного вибору часу виконання робіт і можливості отримання актуальної інформації про стан замовлення, тоді як для компанії важливими стають координація роботи персоналу, планування графіків, облік оплат і контроль виконання послуг. Використання розрізнених засобів комунікації та обліку, зокрема електронних таблиць, телефонних узгоджень і повідомлень у месенджерах, ускладнює організацію цих процесів, знижує оперативність управління та підвищує ймовірність помилок під час розподілу замовлень, бронювання часу й оновлення даних. У зв'язку з цим розробка вебплатформи для планування та контролю діяльності клінінгової компанії є своєчасною та обґрунтованою, оскільки дає можливість об'єднати основні бізнес-процеси в межах єдиного інформаційного середовища, підвищити ефективність керування ресурсами та покращити якість обслуговування клієнтів.

Метою випускної кваліфікаційної роботи бакалавра є розробка вебплатформи для автоматизації основних процесів клінінгової компанії, зокрема оформлення та обробки замовлень, розподілу ролей користувачів, керування роботою майстрів, контролю якості виконаних послуг і формування аналітичної звітності для зменшення ризиків помилок під час ручного обліку.

Для досягнення поставленої мети потрібно детально проаналізувати предметну область та особливості надання послуг у цій сфері. Також потрібно спроектувати архітектуру вебплатформи з розмежуванням прав доступу для різних типів користувачів і розробити оптимізовану базу даних для зберігання інформації про замовлення, клієнтів, персонал та їхні доступи до системи. Окремо слід реалізувати основні модулі, створити адаптивний і простий у використанні інтерфейс з акцентом на мобільну версію та здійснити тестування системи для перевірки стабільності вебдодатку й точності аналітичних даних.

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

Об’єктом дослідження є процес управління операційною діяльністю клінінгової компанії, що охоплює планування виїздів персоналу, контроль якості прибирання на об’єктах, облік ресурсів та аналіз ефективності.

Предметом дослідження є програмні засоби та підходи до розробки вебплатформи для автоматизації процесів клінінгової компанії, зокрема оформлення замовлень, розрахунку вартості та тривалості робіт, планування графіка майстрів, бронювання вільного часу, контролю виконання послуг і обробки оплати. Особлива увага приділяється організації взаємодії між клієнтом, майстром та адміністратором, а також забезпеченню коректного зберігання й обробки даних у межах єдиного програмного комплексу.

Методи дослідження. У роботі застосовано комплексний підхід, який охоплює і теоретичні відомості, і практичні методи. Щоб вивчити бізнес-процеси клінінгових компаній використано метод системного аналізу, який дозволив розбити роботу на окремі модулі.

Інформаційна база досліджень складається з технічної документації до обраних технологій, важливе місце займають навчально-методичні матеріали зі спеціальності, які були фундаментом вибору архітектурних рішень. А також дані про внутрішні регламенти даної сфери та аналізу відкритих джерел.

Практична цінність полягає у можливості використання вебплатформи для автоматизації основних процесів роботи клінінгової компанії, а саме приймання замовлень, розрахунку вартості та тривалості послуг, вибору доступного часу, призначення майстрів, контролю виконання робіт і фіксації оплати. Розроблена система спрощує взаємодію між клієнтом, майстром та адміністратором, зменшує кількість ручних дій під час обробки заявок і допомагає зберігати всі дані про замовлення, графіки, послуги, оплату та відгуки в одному програмному комплексі.

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

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ

1.1 ПОСТАНОВКА ЗАДАЧІ

Зі зростанням кількості клієнтів клінінговим компаніям дедалі складніше обслуговувати внутрішні процеси вручну. Для клієнта процес оформлення замовлення зводиться до вибору послуги, дати та часу виконання робіт, а також заповнення заявки. Водночас система виконує низку внутрішніх операцій, зокрема перевірку доступності майстрів, розрахунок тривалості виконання замовлення, визначення необхідної кількості працівників, контроль завантаженості персоналу, обробку платежів, керування статусами замовлень та підтримку подальшого процесу їх виконання. У разі ведення обліку адміністратором вручну, наприклад за допомогою електронних таблиць або обміну повідомленнями, зростає ризик виникнення неузгодженостей. Наслідком цього може стати призначення одному майстру кількох замовлень на однаковий часовий проміжок або пропонування клієнту часу, який фактично вже зарезервований. Постає завдання розробити інформаційну систему, що автоматизує повний цикл обслуговування замовлення.

Першим завданням є реалізація механізму оформлення замовлень, де користувач повинен мати можливість обрати необхідні послуги, вказати адресу, дату та час їх виконання. На основі введених даних система автоматично розраховує вартість замовлення, визначає тривалість робіт, відображає доступні часові проміжки та враховує потребу в необхідній кількості працівників. Формування складу виконавців здійснюється з урахуванням обсягу замовлення та часу, необхідного для його виконання. Також доцільно передбачити можливість оформлення замовлення без попередньої реєстрації. У такому разі обліковий запис клієнта створюється автоматично на основі вказаного номера телефону, що забезпечує збереження інформації про замовлення та його прив'язку до відповідного користувача.

Другим завданням є організація роботи адміністратора та майстрів.

Адміністратор повинен мати окрему панель, де можна переглядати всі

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

замовлення, створювати заявки телефоном, змінювати статуси, контролювати оплату, працювати з клієнтами, майстрами, послугами, спеціальними пропозиціями та відгуками. Особливо важливо реалізувати керування графіком майстрів, оскільки саме він визначає доступні години. Система повинна підтримувати як налаштування типового тижневого графіка роботи, так і визначення індивідуального розкладу для окремих дат. Це дає змогу враховувати тимчасові зміни доступності працівників та забезпечувати актуальність графіка. Видалення часових слотів, зарезервованих у межах існуючих замовлень, не повинно допускатися з метою забезпечення цілісності даних. Для працівників компанії має бути передбачений спеціалізований кабінет, що забезпечує доступ до інформації про призначені замовлення, включаючи місце виконання робіт, запланований час, контактні дані клієнта, вартість послуг та поточний стан виконання замовлення. Функціональні можливості кабінету повинні також включати зміну статусу виконання робіт і завантаження фотографій результатів до та після надання послуги.

Третім завданням є забезпечення надійності системи під час бронювання, оплати та зміни статусів. Зокрема, необхідно запобігти ситуації, коли два клієнти одночасно намагаються забронювати один і той самий час. Для цього серверна частина повинна використовувати транзакції, блокування слотів та обов'язкову перевірку зайнятості майстрів. Так само система повинна зберігати стан оплати, незалежно від того, чи це готівкова, чи онлайн-оплата, сума, статус платежу та момент підтвердження. Якщо замовлення виконується майстром, його статус має змінюватися поступово, без довільних переходів, наприклад, замовлення не повинно одразу стати завершеним, якщо майстер ще не почав роботу або не додав потрібні фото.

На завершальному етапі слід визначити структуру даних, ролі користувачів і загальну керованість сервісу. У системі потрібно зберігати клієнтів, майстрів, послуги, категорії, замовлення, графіки, робочі слоти, відгуки, спеціальні пропозиції, оплату та фото виконаних робіт – усі ці дані мають бути пов'язані між собою, для забезпечення цілісного відображення

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

інформації. Також доцільно розмежувати доступ: клієнт працює зі своїм кабінетом і власними замовленнями, майстер бачить тільки призначені йому роботи, а адміністратор керує всією системою. Паролі повинні зберігатися в захищеному вигляді [1], а сервер має перевіряти права користувача перед виконанням важливих дій. Крім основної роботи із замовленнями, потрібно передбачити аналітику: кількість заявок, дохід, популярні послуги, проблемні замовлення, стан оплат і можливість експорту звітів.

Загалом, система повинна працювати як єдиний програмний комплекс, клієнт оформлює замовлення, майстрам призначаються роботи, адміністратор контролює процес, а всі дані залишаються впорядкованими в базі. Важливо, щоб кожна дія в системі одразу відображалася в потрібних частинах сервісу і зміна графіка впливала на доступні години, нове замовлення бронювало час майстра, а оновлення статусу було помітне і клієнту, і адміністратору. Такий підхід зменшує кількість ручних узгоджень і підвищує передбачуваність роботи сервісу.

1.2 Аналіз аналогів програмного продукту

З метою визначення ключових функціональних вимог, а також оптимальних архітектурних рішень для майбутньої платформи було проаналізовано існуючі аналоги подібних систем, що спеціалізуються саме на клінінгових послугах. У межах цієї кваліфікаційної роботи обрано найбільш популярні на вітчизняному ринку, а саме: «Кабанчик», «Чистий Кит» та «Клін Сервіс».

«Кабанчик» [2] – сервіс, який дає змогу знайти перевірених фахівців для вирішення необхідних питань, зокрема клінінгових. Ця платформа реалізує модель вільного ринку, де планування замовлень відбувається за принципом тендерної системи, де замовник публікує завдання, що потрібно виконати, обсяг роботи та загальний опис, а виконавці пропонують власні послуги. Головну сторінку зазначеного сервісу наведено на рис. 1.1.

Сайт дає змогу оформлювати замовлення онлайн (рис. 1.2), обирати потрібну послугу, час виконання та виконавця.

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

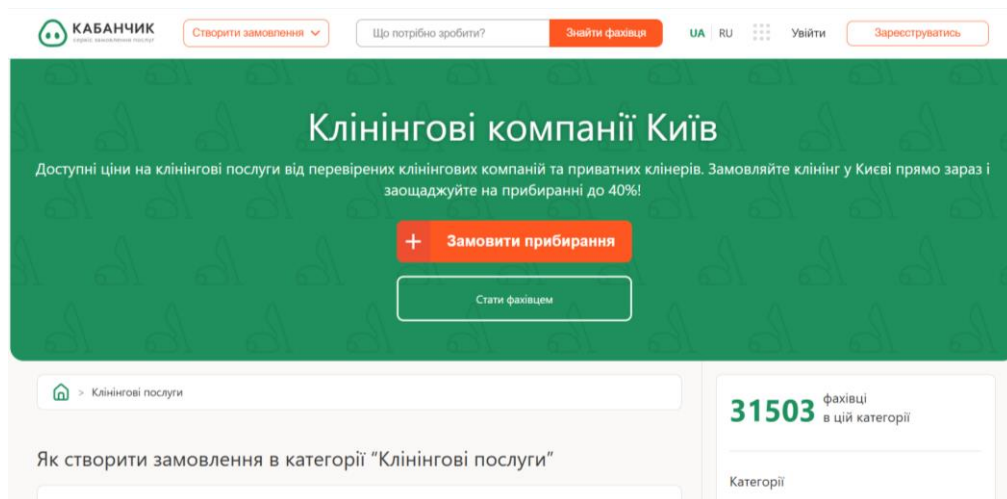


Рис. 1.1 – Сторінка клінінгових послуг

Адреса виконання замовлення

Місто:

Район:

Вулиця:

Будинок (конфіденційно):

Дата виконання замовлення

Березень 27 **Березень 28** Березень 29 Березень 30 Березень 31 Квітень 1

Коли необхідно виконати замовлення?

☒ В будь-який час ☐ З 8:00 до 12:00 ☐ З 12:00 до 16:00 ☐ З 16:00 до 22:00

Оплата роботи

Кількість меблів:

Орієнтовна вартість: грн

☒ Ціна обговорюється

☐ Крісло

☐ Стілець, пуф, табурет

☒ Диван звичайний або кулонний

☐ Диван кутовий

☐ Матрац з одного боку

☐ Матрац з двох сторін

Ця вартість вказана лише для попередньої оцінки бюджету вашого замовлення. В неї не включені матеріали та додаткові роботи. Точну вартість зможе сказати фахівець після обговорення деталей виконання робіт. Після виконання замовлення у вас буде можливість вказати фактичну суму оплати за роботу фахівця не враховуючи вартість матеріалу.

Рулсан О. Був(ла) на сайті 21 хвилину назад
134 відгуків | 98% позитивних
Запропонувати роботу

FIXO Service - Фіксимо ... Був(ла) на сайті 20 годин назад
71 відгуків | 98% позитивних
Запропонувати роботу

Дмитро Л. Був(ла) на сайті 2 години назад
349 відгуків | 98% позитивних
Запропонувати роботу

Дмитрий В. Був(ла) на сайті 3 години назад
202 відгуків | 98% позитивних
Запропонувати роботу

Вячеслав С. Був(ла) на сайті 15 годин назад
24 відгуків | 89% позитивних
Запропонувати роботу

Євген З. Був(ла) на сайті 3 години назад
52 відгуків | 89% позитивних
Запропонувати роботу

Рис. 1.2 – Сторінка замовлення послуги хімічистки меблів

Під час аналізу цього сервісу було виявлено низку недоліків, що роблять використання платформи незручним, як для кінцевого споживача, так і для бізнесу. Інтерфейс перевантажений і має застарілий дизайн. Внутрішня комунікація платформи погано структурована. Замовник самостійно фільтрує численні неперевірені пропозиції та вступає в тривале листування, що не гарантує очікуваної якості послуги. Окремою проблемою є повна відсутність

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

контролю за технічним процесом, оскільки платформа не контролює виконавців, тому не гарантує використання професійного обладнання та сертифікованих засобів. Як показано на рис. 1.3, профіль виконавця містить портфоліо та відгуки, на основі яких замовник самостійно оцінює майстра.

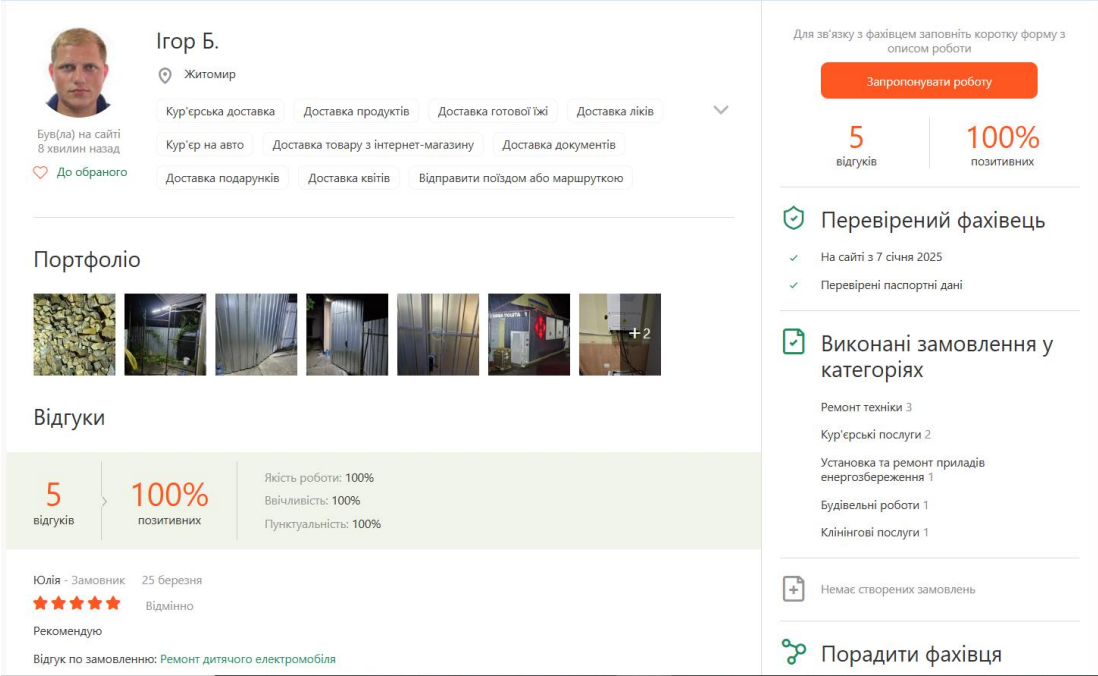


Рис. 1.3 – Модуль вибору виконавця та перегляду репутаційних показників

Процес формування вартості замовлення на платформі «Кабанчик» базується на моделі вільного ринку, де ціна, яку розраховує калькулятор, має лише рекомендаційний характер. Недолік такого підходу полягає в тому, що фінальна вартість визначається в результаті торгів між сторонами. Це призводить до фінансової невизначеності та демпінгу цін. Крім того, клієнт змушений щоразу вступати в повторні переговори з різними виконавцями.

На противагу такому підходу, у розроблюваній вебплатформі реалізується інша логіка ціноутворення. Система підтримуватиме гнучкий калькулятор, який не підсумовуватиме окремі позиції, а враховуватиме динамічний перехід до пакетних пропозицій, тобто під час досягнення певного набору послуг, ціна буде автоматично перерахована за вигіднішим тарифом та без участі адміністратора чи додаткових запитів з боку замовника.

«Чистий Кит» [3] – сучасна платформа для пошуку послуг для дому, яка є прикладом функціонально розвиненого сервісу, який спеціалізується на автоматизації замовлень клінінгових послуг.

Головний екран (рис. 1.4) дає змогу одразу почати оформлення через інтерактивний віджет з можливістю одразу вибрати параметри приміщення, що дає змогу отримати попередній розрахунок вартості за кілька натискань.

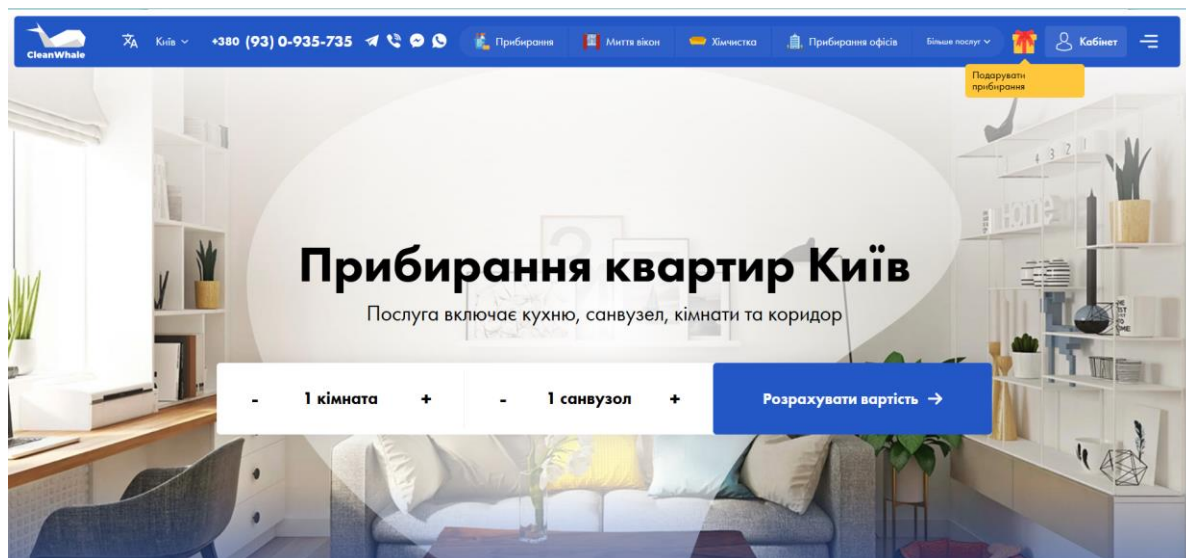


Рис. 1.4 – Головний екран платформи «Чистий Кит»

Процес оформлення замовлення побудований за допомогою покрокового алгоритму (рис. 1.5), де все необхідне поділене на логічні блоки, тому складні завдання вибору діляться на зрозумілі етапи. Такий підхід спрощує взаємодію користувача із системою, зменшує ймовірність помилок під час введення даних і забезпечує послідовне виконання всіх необхідних дій для формування замовлення. Крім того, така організація інтерфейсу полегшує перевірку введених даних перед остаточним підтвердженням заявки. Це особливо важливо для вебплатформи, у якій точність параметрів замовлення безпосередньо впливає на подальше планування робіт і коректність їх виконання.

1. Вибір послуг

2. Адреса та контактні дані

ВКАЖІТЬ ВАШУ АДРЕСУ

Оберіть своє місто: Київ ▾

Вулиця

Номер будинку

Номер квартири

Корпус

Номер під'їзду

Поверх

Код від домофона

КОНТАКТНІ ДАНІ

Ваше ім'я

Контактний телефон

+380

50 123 4567

E-mail адреса

Рис. 1.5 – Блоки у формі для замовлення

Критично важливою технічною особливістю є наявність реактивного «кошика» (рис. 1.6), який оновлює у реальному часі загальну вартість та орієнтовний час виконання роботи, після додавання клієнтом нової позиції, до прикладу. Це забезпечує фінансову прозорість та виключає необхідність додаткових переговорів з адміністратором.

Хімчистка 2900 UAH

Наші засоби для чищення гіпоалергенні, після хімчистки ми повністю змиваємо їх з усіх поверхонь. Хімчистка безпечна для дітей і домашніх тварин

Приблизний час роботи **1 година 20 хвилин**

Додаткові послуги

Хімчистка тримісного дивана (1)

Хімчистка килимового покриття (10)

Реферальна програма ▾ [Як це працює?](#)

До оплати: 2900 UAH

Замовити

Рис. 1.6 – Реактивний «кошик»

Перевагою можна вважати і те, що відгуки базуються на інтеграції з незалежними платформами, такими як Facebook та Google Maps. Використання

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

верифікованих зовнішніх коментарів замість статичного тексту значно підвищує рівень довіри користувачів до бренду.

Попри високу технологічну базу, аналіз виявив також певні недоліки, зокрема порушення звичних патернів поведінки користувача. Механізм перемикання між основними типами послуг винесений у хедер сайту, що є нетиповим рішенням для сервісних платформ. Таке розміщення негативно впливає на інтуїтивність інтерфейсу, оскільки користувач очікує бачити основні елементи керування замовленням безпосередньо в межах робочої області сторінки.

«Клін Сервіс» [4] – типовий приклад регіонального вузькоспеціалізованого ресурсу, що виконує роль цифрової вітрини.

Головний екран (рис. 1.7) має перевантажений візуальний стиль із використанням неактуальних графічних патернів, що погіршує сприйняття інтерфейсу.

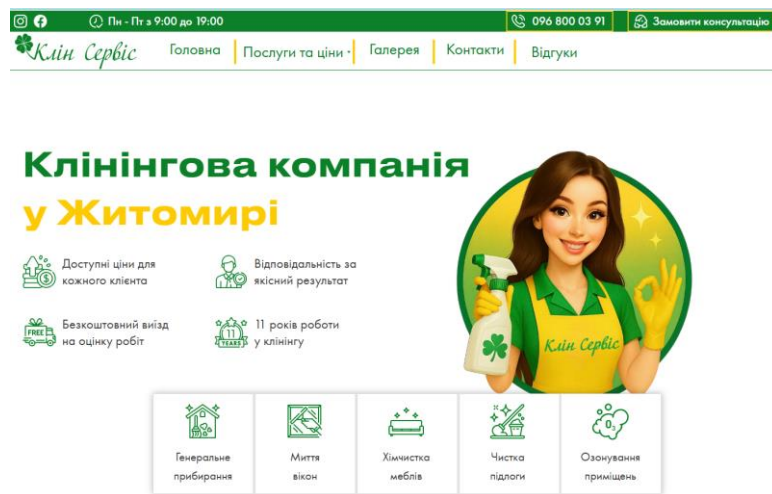


Рис. 1.7 – Головний екран «Клін Сервіс»

Вагомим або навіть критичним недоліком є відсутність повноцінної реалізації онлайн-замовлення. На рис. 1.8 продемонстровано форму зворотного зв'язку, яка реалізована як форма, у якій користувач може залишити контактний номер для зворотного дзвінка. Це створює комунікаційний бар'єр, оскільки клієнт не може самостійно обрати час і переглянути остаточну вартість. Такий підхід поступається автоматизованим системам за швидкістю обробки замовлень.

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

Рис. 1.8 – Форма зворотного зв'язку

Ціни на послуги (рис. 1.9) представлені за допомогою статичних блоків і не пов'язані з логікою калькуляції загальної вартості. За такого підходу до подання цін, користувач змушений самостійно розраховувати вартість. Це ускладнює оформлення об'ємних замовлень, а актуальність зазначених цін лишається невизначеною. Що стосується блоку відгуків, то він також представлений у статичному вигляді, тому користувачі можуть менше довіряти, оскільки такого роду відгуки піддаються модерації з боку власника сайту. Відсутні футер і швидкий доступ до юридичної інформації, що свідчить про незавершеність інтерфейсу.

Рис. 1.9 – Прайс-лист «Клін Сервіс»

Для підсумкового порівняння вищенаведених вебплатформ було створено порівняльну таблицю (табл. 1.1).

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

Порівняння аналогів вебплатформи

Критерій порівняння	Кабанчик	Чистий Кит	Клін Сервіс	Cleanly
Автоматичний розрахунок ціни	Частково, орієнтовна ціна	Так	Ні	Так, враховуючи знижки і пропозиції
Онлайн-бронювання	Через узгодження з виконавцем	Так	Ні	Так, з перевіркою доступних слотів
Особистий кабінет клієнта	Є базовий профіль	Частково	Ні	Так
Автоматичне призначення майстрів	Ні	Не виявлено	Ні	Так
Регулярні прибирання	Не є основною функцією	Так	Ні	Так
Онлайн-оплата	Залежить від виконавця	Так	Ні	Так
Підтверджені відгуки	Так, про виконавців	Так, через зовнішні джерела	Статичні	Так, після виконаного замовлення

Порівняння показало, що кожен із розглянутих аналогів вирішує лише частину завдань, пов'язаних із наданням клінінгових послуг. Сервіс «Кабанчик» орієнтований переважно на пошук окремих виконавців і не забезпечує централізованого контролю якості, графіка роботи та процесу виконання замовлення. «Чистий Кит» має зрозумілий і послідовний механізм оформлення, автоматичний розрахунок вартості та вибір часових слотів, однак у відкритому доступі не виявлено повноцінних інструментів для адміністрування діяльності компанії та роботи майстрів. «Клін Сервіс» здебільшого виконує роль інформаційного сайту, де користувач може переглянути послуги та залишити

заявку на зворотний дзвінок, але не може самостійно забронювати послугу або побачити фінальну вартість замовлення.

На відміну від наведених аналогів, розроблювана платформа забезпечує автоматичний розрахунок вартості, онлайн-бронювання, призначення майстрів відповідно до їхньої доступності, підтримку регулярних прибирань, онлайн-оплату, перевірені відгуки та аналітику для адміністратора.

Окремою перевагою є механізм формування спеціальних пропозицій на основі комбінацій послуг, адже, на відміну від статичних знижок, система аналізує вибрані клієнтом послуги та може автоматично запропонувати додаткову послугу або вигіднішу комбінацію. Наприклад, під час вибору певного типу прибирання клієнту може бути запропонована хімчистка або миття вікон зі знижкою, а самі пропозиції налаштовуються адміністратором, що дає змогу компанії гнучко керувати маркетинговими акціями без зміни програмного коду.

1.3 Вибір архітектури вебплатформи планування та контролю клінінгової компанії

Для реалізації зазначених вимог було обрано хмарну СУБД Neon на базі PostgreSQL, що забезпечить стабільну роботу, швидку обробку запитів та перспективу масштабування системи [5]. Загалом, технічна реалізація базуватиметься на побудові адаптивного вебінтерфейсу. Фронтенд-частина реалізовуватиметься як SPA на базі фреймворку Vue 3 з використанням Composition API, що забезпечує гнучкість та масштабованість коду [6]. Побудова інтерфейсу здійснюватиметься з використанням компонентної бібліотеки PrimeVue, яка дає змогу реалізувати складні елементи керування даними [7]. Додатково для маршрутизації між сторінками використовується Vue Router [8], для керування глобальним станом застосовується Pinia [9], а для стилізації інтерфейсу використовується Tailwind CSS [10].

Серверна частина функціонує на базі середовища виконання Node.js [11], а оскільки як клієнтська, так і серверна частини реалізуються мовою JavaScript, важливим теоретичним підґрунтям для розробки є принципи роботи з асинхронністю, об'єктами, функціями та модулями, розглянуті у праці М.

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

Haverbeke «Eloquent JavaScript» [12]. Безпосередня реалізація архітектури REST API та обробка HTTP-запитів здійснюється за допомогою фреймворку Express, який є оптимальним інструментом для швидкої та асинхронної обробки великих масивів даних у реальному часі [13]. Взаємодія з базою даних PostgreSQL, розміщеною на платформі Neon, виконується через бібліотеку node-postgres [14] із використанням пулу з'єднань, що дає змогу зменшити витрати ресурсів на встановлення нових підключень до бази даних і забезпечити стабільнішу роботу системи у разі збільшення кількості запитів.

Варто зауважити, що особливу увагу під час виконання вебплатформи приділено швидкодії та ергономічності взаємодії користувача із системою. Завдяки SPA-архітектурі основні ресурси застосунку завантажуються один раз, а взаємодія із сервером відбувається через асинхронні запити до API без повного перезавантаження сторінок. Це особливо важливо для модуля оформлення замовлення, оперативно оновлювати параметри візиту. Будь-яка зміна послуг, регулярності, дати або часу повинна відразу перераховувати підсумкову вартість і тривалість. На основі цих даних система має негайно коригувати список доступних часових слотів. Для візуалізації аналітичних даних в адміністративній панелі використовується Chart.js [15], що дає змогу відображати статистику замовлень, доходів і популярності послуг безпосередньо в браузері.

Серверна частина побудована за модульним принципом і передбачає розподіл обов'язків між маршрутами, middleware та допоміжними модулями бізнес-логіки. Маршрути відповідають за обробку запитів клієнта, middleware забезпечують перевірку автентифікації та ролей користувачів, а окремі утилітарні модулі містять логіку, пов'язану з розрахунком тривалості замовлення, кількості майстрів, статусів замовлень та інтеграцією з платіжними сервісами. Такий підхід спрощує підтримку проєкту та дає змогу вносити зміни в окремі частини системи без порушення роботи інших модулів.

Важливою складовою системи є механізм динамічного розрахунку замовлення. Вартість і тривалість формуються на основі вибраних клієнтом послуг, регулярності прибирання та спеціальних пропозицій. У базі даних

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

передбачені таблиці для зберігання інформації про послуги, категорії та вигідні пропозиції, що дає змогу адміністратору змінювати умови акцій через адміністративний інтерфейс без модифікації програмного коду. Окремо реалізовано механізм визначення необхідної кількості майстрів відповідно до загальної тривалості робіт і кількості активних працівників.

Безпека базується на розмежуванні прав доступу та використанні JWT-автентифікації [16]. Токен автентифікації передається через cookie, що дає змогу підтримувати сесію користувача між запитами. Для захисту паролів використовується алгоритм Bcrypt [17], який забезпечує одностороннє хешування паролів. У системі передбачено рольову модель доступу: клієнт, майстер та адміністратор мають різні маршрути, можливості й обмеження. Доступ до відповідних API-маршрутів обмежується серверними middleware, а на фронтенді додатково реалізовано маршрутні обмеження через Vue Router.

Для спрощення авторизації передбачено вхід за номером телефону. Користувач може авторизуватися за допомогою номера телефону та пароля, якщо пароль було створено раніше, або через підтвердження коду в Telegram-боті. Такий підхід відповідає специфіці сервісу, оскільки номер телефону є основним ідентифікатором клієнта під час оформлення замовлення та подальшій комунікації. Для адміністраторів і майстрів також передбачено вхід за обліковими даними, які надаються або створюються в межах системи.

Окремим елементом архітектури є інтеграція з платіжною системою LiqPay [18]. Вона використовується для реалізації онлайн-оплати замовлень після завершення роботи майстром. Система автоматично формує платіжний запит і перенаправляє користувача на сторінку оплати. Після завершення транзакції відбувається оновлення статусу платежу в базі даних. Для цього використовуються механізми зворотного запиту (callback) або додаткова синхронізація статусів. Це дає змогу підтримувати як оплату готівкою, так і онлайн-оплату залежно від вибраного способу розрахунку.

Для роботи із завантаженням файлів у системі використовується Multer [19]. Цей механізм застосовується для обробки зображень, зокрема фотографій

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

майстрів або інших медіафайлів, які можуть використовуватися в адміністративній частині платформи. Такий підхід дає змогу централізовано обробляти файли на сервері та зберігати їх у визначеній структурі проєкту.

Результатом вибору такого архітектурного рішення є повнофункціональна вебплатформа, що складається з клієнтської та серверної частин і забезпечує автоматизацію ключових процесів клінінгової компанії. Система дозволить клієнтам створювати замовлення, переглядати їхній статус, здійснювати оплату та залишати відгуки. Майстрам – працювати з призначеними замовленнями, а адміністраторам – керувати послугами, графіками, майстрами, спеціальними пропозиціями, оплатами, відгуками та аналітичними показниками. Завдяки цьому платформа створюватиме технологічну основу для стабільного розвитку клінінгового бізнесу.

Висновки до першого розділу

У першому розділі було обґрунтовано актуальність теми та проаналізовано передумови створення вебплатформи. Необхідність цього дослідження була обґрунтована, і підтверджена потреба в автоматизації процесу планування та контролю діяльності клінінгової компанії. Аналіз ринку показав зростання попиту на клінінгові послуги, що підвищило потребу в ефективному використанні ресурсів, зменшенні впливу людського фактора під час формування розкладу, забезпеченні прозорого ціноутворення та наданні якісного обслуговування клієнтів. Відповідно, розробка власної вебплатформи є доречним для вирішення цих викликів. На основі аналізу існуючих аналогів, включаючи такі сервіси, як «Кабанчик», «Чистий Кит» та «Клін Сервіс», було визначено основні переваги та недоліки існуючих рішень. Проблема полягає в тому, що більшість із них призначена для пошуку окремих виконавців або виступають інформаційними сервісами, а не для надання повної автоматизації процесів замовлення, планування, призначення майстрів та подальшої аналітики. У цьому випадку створення власної вебплатформи є доцільним рішенням, що дає змогу інтегрувати обслуговування клієнтів та адміністративне управління в єдину систему.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Для реалізації системи обрано архітектуру SPA на базі Vue 3, Composition API, PrimeVue, Pinia, Vue Router та Tailwind CSS. Обробка клієнтських запитів і виконання бізнес-логіки покладаються на серверну частину, розроблену на Node.js із використанням Express. Для зберігання даних застосовується PostgreSQL на платформі Neon, що підтримує централізоване керування даними та можливість масштабування вебплатформи. Для захисту даних у системі використовується аутентифікація на основі JWT-токенів, які передаються через cookie, а паролі користувачів зберігаються у вигляді хешів, створених за допомогою алгоритму bcrypt. Крім того, у системі реалізовано рольову модель доступу, яка розмежовує функціональні можливості клієнта, майстра та адміністратора. Користувачі можуть увійти за допомогою номера телефону та пароля або коду підтвердження в Telegram-боті. Також передбачено здійснення онлайн-платежів через LiqPay, що дає змогу здійснювати оплату після завершення замовлення. Таким чином, можна зробити висновок, що вебплатформа, яка інтегрує кабінет клієнта, кабінет майстра та адміністративну панель для керування послугами, замовленнями та користувачами, є ефективною та життєздатною на практиці. Обраний технологічний стек враховує недоліки, виявлені під час аналізу існуючих аналогів, та надає можливість реалізувати автоматизацію рутинних процесів, онлайн-бронювання послуг, автоматичне призначення майстрів, контроль виконання робіт, гнучке ціноутворення, систему акційних пропозицій, підтримку платежів та інструменти аналітики. Це формує стійку технологічну основу для розвитку клінінгового бізнесу та підвищення його конкурентоспроможності на ринку.

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

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ

2.1 Вимоги до системи

На етапі проєктування вебплатформи було визначено основні ролі користувачів системи, функціональні можливості кожної ролі та ключові об'єкти предметної області. Оскільки система призначена для автоматизації процесів клінінгової компанії, основними акторами виступають клієнт, майстер та адміністратор. Кожен із них взаємодіє з окремими модулями системи відповідно до власних прав доступу.

Основна ціль розробки вебплатформи полягає у створенні комплексного вебзастосунку для автоматизації процесів планування, оформлення та контролю виконання клінінгових послуг. Система має забезпечити зручну взаємодію між клієнтами, майстрами та адміністратором, зменшити кількість ручних операцій під час приймання замовлень, автоматизувати розподіл робочого часу персоналу та підвищити прозорість виконання робіт.

Функціональні вимоги до вебплатформи охоплюють основні дії, які мають бути доступними користувачам відповідно до їхніх ролей. Клієнт повинен мати можливість зареєструватися або увійти до системи, оформити замовлення, додати кілька послуг до кошика, обрати дату, час і адресу виконання, переглянути автоматично розраховану вартість, тривалість робіт і кількість необхідних майстрів. Також клієнту має бути доступний список власних замовлень, їхніх деталей, поточного статусу, інформації про оплату, призначених майстрів, фото до та після виконання роботи, а після завершення й підтвердження оплати – можливість залишити відгук. Для майстра передбачено окремий кабінет, у якому відображаються призначені замовлення, роботи на поточний день, майбутні замовлення та історія виконаних робіт. Він може змінювати етапи виконання замовлення, передавати геопозицію під час початку виконання робіт, завантажувати фотографії до та після надання послуги, а також фіксувати спосіб оплати після завершення робіт. Для адміністратора функціональні вимоги охоплюють керування замовленнями, користувачами,

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

майстрами, графіками, послугами, спеціальними пропозиціями, відгуками, оплатами та аналітичними даними. Окремо система має підтримувати автоматичне призначення майстрів, ручне коригування проблемних замовлень, формування регулярних замовлень і застосування знижок відповідно до умов спеціальних пропозицій.

Нефункціональні вимоги до вебплатформи описують якісні характеристики роботи системи. Інтерфейс має бути адаптивним і зручним для використання на різних розмірах екрана, оскільки клієнти, адміністратор і майстри можуть працювати із системою з різних пристроїв. Система повинна забезпечувати розмежування доступу за ролями, захист персональних даних користувачів, коректну роботу з одночасним бронюванням часових слотів і своєчасне оновлення інформації без повного перезавантаження сторінки. Оскільки вебплатформа працює з бронюванням часу майстрів, система має забезпечувати узгодженість даних і не допускати ситуації, коли один і той самий майстер або часовий слот одночасно призначається на кілька замовлень. Для цього під час створення замовлення, призначення майстрів і резервування часу повинні виконуватися перевірки доступності персоналу та коректності вибраного часу. Крім того, система має бути масштабованою, щоб у майбутньому можна було розширити перелік міст, кількість майстрів, набір послуг і додаткові сценарії роботи без повної перебудови архітектури.

Для збереження читабельності в основному тексті роботи подано укрупнені UML-діаграми варіантів використання для клієнта, майстра та адміністратора окремо. Деталізація окремих сценаріїв, зокрема створення замовлення, призначення майстрів, підтвердження оплати, фотофіксація виконаної роботи та залишення відгуку, подана в таблицях опису ключових прецедентів.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 2.1 – Діаграма варіантів використання клієнта

Як показано на рис. 2.1, клієнт взаємодіє із системою через клієнтський кабінет. Основними можливостями клієнта є реєстрація або авторизація, оформлення замовлення, керування власними замовленнями та їхніми деталями, оплата, перегляд фотофіксації та залишення відгуку. Під час оформлення замовлення клієнт формує кошик послуг, обирає дату та час, вводить адресу й отримує автоматично розраховану вартість. Після створення замовлення користувач може відстежувати його етапи виконання, переглядати призначених майстрів, погоджувати запропоновані зміни та, за потреби, редагувати або скасовувати замовлення в межах дозволеного часу.

Окремою функцією клієнтського кабінету є перегляд фотофіксації виконання роботи у форматі «до» та «після». Це дає змогу клієнту переконатися у факті виконання послуги та підвищує прозорість взаємодії між замовником і клінінговою компанією та запобігає виникненню спірних питань. Після завершення замовлення й підтвердження оплати клієнт отримує можливість залишити відгук, який надалі може бути використаний на головній сторінці платформи та в адміністративній панелі.



Рис. 2.2 – Діаграма варіантів використання майстра

На рис. 2.2 наведено варіанти використання для майстра. Майстер має доступ до окремого кабінету, де переглядає призначені йому замовлення, бачить роботи на поточний день, майбутні замовлення та історію виконаних робіт. Основний сценарій роботи майстра полягає у послідовній зміні етапів виконання замовлення: «В дорозі», «В роботі» та «Завершено».

Під час початку роботи система може вимагати передачу геопозиції, що дає змогу адміністратору перевірити, чи перебуває майстер за адресою виконання замовлення. Також майстер завантажує фото до початку роботи та після її завершення. Наявність фотофіксації є обов'язковою умовою завершення замовлення, оскільки вона підтверджує факт виконання робіт і може використовуватися для розгляду спірних ситуацій. Після завершення роботи майстер фіксує спосіб оплати, обраний клієнтом, наприклад готівкою або через онлайн-оплату.

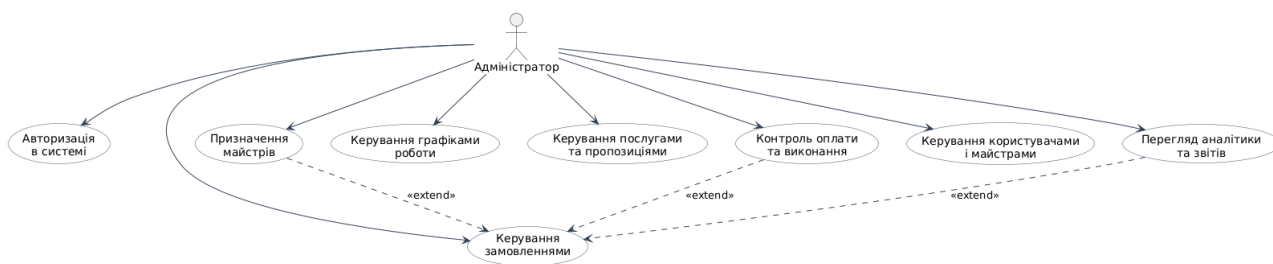


Рис. 2.3 – Діаграма варіантів використання адміністратора

Варіанти використання адміністратора подано на рис. 2.3. Адміністратор є користувачем із найширшими правами доступу та відповідає за керування основними бізнес-процесами системи. Він може переглядати всі замовлення, здійснювати пошук і фільтрацію, переглядати деталі кожного замовлення, контролювати статуси виконання та оплати. До важливих функцій адміністратора належить призначення майстрів. У стандартному сценарії система автоматично підбирає команду відповідно до тривалості замовлення та доступності майстрів. Якщо автоматичне призначення неможливе, адміністратор може вручну обрати майстрів або запропонувати клієнту інший час виконання. Також адміністратор керує графіками роботи майстрів, послугами, спеціальними

пропозиціями, користувачами, відгуками та переглядає аналітичні показники діяльності компанії.

Значну роль в адміністративній панелі відіграє контроль якості виконання, адже адміністратор має змогу переглядати фотографії до та після роботи, бачити, який майстер їх завантажив і коли саме це було зроблено. Це дає змогу контролювати фактичне виконання замовлень, швидше реагувати на скарги клієнтів та підвищувати якість сервісу.

Для уточнення функціональних вимог, поданих на UML-діаграмах варіантів використання, було сформовано описи ключових прецедентів системи, які представлені у табл. 2.1–2.9.

Таблиця 2.1

Опис прецеденту «Оформлення замовлення клієнтом»

Прецедент	Оформлення замовлення клієнтом
Актор	Клієнт
Передумови	У системі є активні послуги та доступні часові слоти
Основний сценарій	Клієнт обирає послуги, вводить адресу, дату й час. Система розраховує вартість, тривалість, кількість майстрів і створює замовлення
Результат	Замовлення створено, а час майстрів зарезервовано

У табл. 2.1 подано базовий прецедент, що відображає основний сценарій взаємодії клієнта з вебплатформою, а саме оформлення замовлення. Саме цей процес формує початкові дані для подальшої роботи інших модулів системи.

Таблиця 2.2

Опис прецеденту «Керування власними замовленнями»

Прецедент	Керування власними замовленнями
Актор	Клієнт
Передумови	Клієнт авторизований і має хоча б одне створене замовлення

Основний сценарій	Клієнт відкриває список замовлень, переглядає деталі, статус, дату, час, адресу, оплату та призначених майстрів
Результат	Клієнт отримує актуальну інформацію про замовлення та може виконати доступні дії

Після створення заявки важливим є доступ клієнта до вже сформованих замовлень, їх стану та детальної інформації. Відповідний прецедент керування власними замовленнями наведено в табл. 2.2. Наступним логічним етапом є організація виконання створеного замовлення, що передбачає підбір виконавців і резервування робочого часу. Опис прецеденту призначення майстрів подано в табл. 2.3.

Таблиця 2.3

Опис прецеденту «Призначення майстрів»

Прецедент	Призначення майстрів
Актор	Система, адміністратор
Передумови	Замовлення створено, у системі є активні майстри та робочі слоти
Основний сценарій	Система визначає тривалість замовлення, розраховує потрібну кількість майстрів, перевіряє їх доступність і автоматично резервує відповідні слоти
Результат	До замовлення призначено команду майстрів або сформовано запит на ручне опрацювання

Після призначення виконавців система переходить до етапу безпосереднього виконання робіт. У табл. 2.4 наведено прецедент, який описує послідовність дій майстра під час виконання замовлення.

Таблиця 2.4

Опис прецеденту «Виконання замовлення майстром»

Прецедент	Виконання замовлення майстром
Актор	Майстер
Передумови	Майстер авторизований, активний і призначений до замовлення

Основний сценарій	Майстер переглядає замовлення, переходить до статусу «В дорозі», починає роботу, передає геопозицію, виконує послугу та завершує замовлення
Результат	Замовлення отримує статус «Завершено» та переходить до етапу оплати

Окремим складником процесу виконання робіт є фотофіксація, оскільки вона забезпечує документальне підтвердження стану об'єкта до початку та після завершення послуги. Цей прецедент деталізовано в табл. 2.5.

Таблиця 2.5

Опис прецеденту «Фотофіксація виконання роботи»

Прецедент	Фотофіксація виконання роботи
Актор	Майстер
Передумови	Майстер призначений до замовлення, а замовлення перебуває в роботі
Основний сценарій	Майстер завантажує фото стану приміщення до початку роботи та фото результату після виконання послуги
Результат	Фото до та після роботи зберігаються в системі й доступні для перегляду клієнту та адміністратору

Після завершення робіт система переходить до етапу оплати, у межах якого враховується обраний спосіб розрахунку та оновлюється стан замовлення. Опис цього прецеденту наведено в табл. 2.6.

Таблиця 2.6

Опис прецеденту «Оплата замовлення»

Прецедент	Оплата замовлення
Актор	Клієнт, майстер, адміністратор
Передумови	Замовлення завершено, а спосіб оплати визначено майстром
Основний сценарій	Майстер обирає спосіб оплати. У разі онлайн-оплати клієнт переходить до LiqPay, а якщо готівкою, то майстер підтверджує отримання коштів
Результат	Статус оплати оновлено, а завершене й оплачене замовлення переноситься в архів

Завершальним елементом взаємодії клієнта із виконаним замовленням є можливість залишення відгуку, що дає змогу зафіксувати оцінку якості послуги. Відповідний сценарій подано в табл. 2.7.

Таблиця 2.7

Опис прецеденту «Залишення відгуку»

Прецедент	Залишення відгуку
Актор	Клієнт
Передумови	Замовлення завершено, оплату підтверджено, відгук для цього замовлення ще не залишено
Основний сценарій	Клієнт відкриває деталі завершеного замовлення, обирає оцінку та вводить текст відгуку
Результат	Відгук зберігається в системі та може відображатися в адміністративній панелі й на головній сторінці

Окрему групу становлять адміністративні прецеденти, пов'язані з наповненням системи даними про послуги, категорії та спеціальні пропозиції. Їх опис наведено в табл. 2.8, оскільки саме ці дані використовуються під час оформлення замовлень і розрахунку вартості.

Таблиця 2.8

Опис прецеденту «Керування послугами та спеціальними пропозиціями»

Прецедент	Керування послугами та спеціальними пропозиціями
Актор	Адміністратор
Передумови	Адміністратор авторизований у системі
Основний сценарій	Адміністратор створює, редагує або деактивує послуги, категорії та спеціальні пропозиції для автоматичного застосування знижок
Результат	Дані про послуги та пропозиції оновлюються й використовуються під час оформлення замовлення

Ще одним важливим адміністративним процесом є керування графіком майстрів, від якого залежить доступність часових слотів для нових заявок.

Відповідний прецедент, що описує формування шаблонного та денного графіка роботи виконавців, наведено в табл. 2.9.

Таблиця 2.9

Опис прецеденту «Керування графіком майстрів»

Прецедент	Керування графіком майстрів
Актор	Адміністратор
Передумови	Адміністратор авторизований, у системі створені профілі майстрів
Основний сценарій	Адміністратор переглядає майстрів, задає шаблон робочого графіка або редагує денні слоти для конкретної дати
Результат	Графік майстрів оновлюється та використовується системою під час пошуку доступного часу для замовлення

Після визначення основних варіантів використання доцільно перейти до формування об'єктно-орієнтованої структури вебплатформи. Вона відображає основні сутності предметної області, їхні властивості та взаємозв'язки між ними.

Окрім визначення ролей користувачів, під час проєктування системи було виділено основні сутності предметної області, до яких належать користувач, замовлення, послуга, майстер, графік роботи, часовий слот, оплата, відгук, спеціальна пропозиція та фотофіксація виконання робіт. Основною сутністю системи є користувач, який може мати одну з ролей: клієнт, майстер або адміністратор. Клієнт створює замовлення, переглядає їхній стан, здійснює оплату та залишає відгук. Майстер отримує призначені замовлення, змінює статуси виконання, передає геопозицію та додає фотофіксацію. Адміністратор керує даними системи, контролює виконання замовлень і відповідає за налаштування сервісу.

Сутність «Замовлення» є центральною в системі, оскільки пов'язує між собою багато об'єктів, зокрема клієнта, послуги, майстрів, дату, час, адресу, оплату, статус виконання та фотофіксацію. Одне замовлення може містити кілька послуг і бути призначене одному або кільком майстрам. Для реалізації

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

такого зв'язку використовується окрема сутність призначення майстрів до замовлення, що дає змогу коректно обробляти командні виїзди.

Сутність «Послуга» описує конкретний вид клінінгової роботи, її базову вартість, тривалість та категорію. На основі вибраних послуг система розраховує загальну вартість і тривалість замовлення. Додатково передбачено сутність «Спеціальна пропозиція», яка дає змогу автоматично застосовувати вигідні умови або знижки під час вибору певної комбінації послуг.

Сутності «Графік роботи» та «Часовий слот» використовуються для планування доступності майстрів. Графік визначає типову зайнятість майстра за днями тижня, а слоти часу дають змогу бронювати конкретні проміжки під замовлення. Це забезпечує контроль доступності персоналу та запобігає одночасному призначенню одного майстра на кілька замовлень.

Окремими об'єктами системи є «Оплата», «Відгук» та «Фотофіксація». Оплата містить інформацію про спосіб розрахунку, суму та статус оплати. Відгук може бути залишений лише після завершення замовлення й підтвердження оплати, що робить його перевіреним. Сутність «Фотофіксація» містить зображення до та після виконання роботи, тип фото, час завантаження та майстра, який його додав.

Для формального подання об'єктної структури системи використано UML-діаграму класів, оскільки UML є стандартизованою мовою візуального моделювання, яка застосовується для специфікації, візуалізації та документування структури програмних систем [20]. Використання саме діаграми класів дає змогу впорядковано відобразити основні сутності предметної області, їх атрибути, зв'язки між ними та загальну логіку організації даних у межах програмного комплексу. Такий підхід є доцільним на етапі проектування, оскільки дозволяє не лише формалізувати внутрішню будову системи, а й узгодити уявлення про неї між етапами аналітичного опрацювання, реалізації та подальшого супроводу. Побудова діаграми здійснювалася за допомогою PlantUML, який підтримує створення UML-діаграм, зокрема діаграм класів і варіантів використання, на основі текстового опису [21]. Це дає можливість

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

зберігати модель у формалізованому вигляді, швидко вносити зміни до структури та підтримувати узгодженість між проєктною документацією і фактичною реалізацією системи.

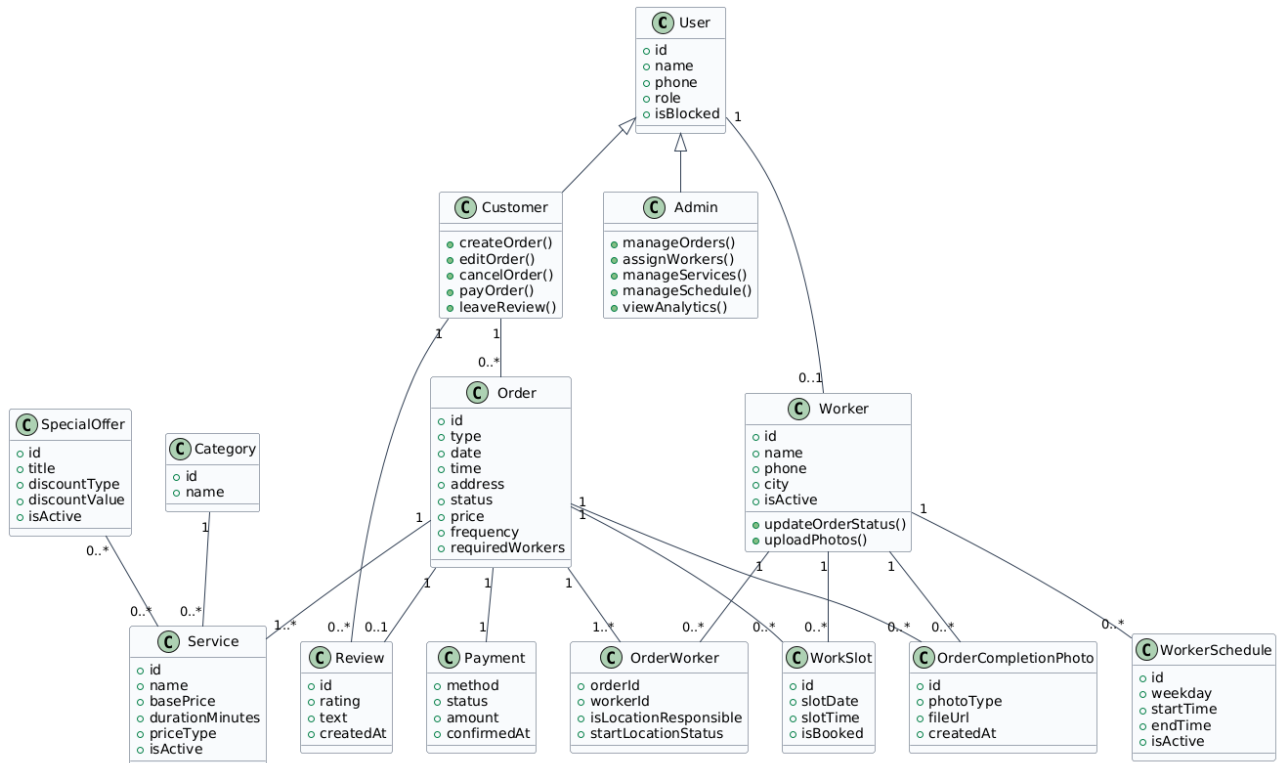


Рис. 2.4 – UML-діаграма класів вебплатформи

На рис. 2.4 подано основні класи системи, які відображають логічну структуру вебплатформи. Центральним класом є Order, оскільки саме замовлення поєднує клієнта, вибрані послуги, дату й час виконання, адресу, статус, оплату, майстрів та фотофіксацію виконаної роботи. Клас User є базовим для користувачів системи та містить загальні дані, зокрема ім'я, телефон, роль і стан облікового запису. Клас Customer описує клієнта, який створює замовлення, редагує або скасовує його, здійснює оплату та залишає відгук. Клас Admin описує адміністратора, який виконує адміністративні дії: керування замовленнями, майстрами, послугами, графіками та аналітикою. Клас Worker описує майстра, який виконує призначені замовлення, змінює їхні статуси та додає фотофіксацію.

Класи Service і Category використовуються для опису структури клінінгових послуг. Кожна послуга належить до певної категорії та містить дані про базову вартість, тривалість і тип розрахунку ціни. Клас SpecialOffer відображає спеціальні пропозиції, які можуть бути пов'язані з однією або кількома послугами та застосовуватися під час формування замовлення.

Для організації роботи майстрів передбачено класи WorkerSchedule і WorkSlot, де перший описує типовий графік роботи майстра, а другий – конкретні часові слоти, які можуть бути доступними або заброньованими. Зв'язок між Order, Worker і OrderWorker дає змогу призначати до одного замовлення кількох майстрів, що є важливим для обробки замовлень із різною тривалістю та складністю. Окремими класами є Payment, Review та OrderCompletionPhoto. Клас Payment зберігає інформацію про спосіб, суму та статус оплати. Клас Review використовується для перевірених відгуків, які можуть бути залишені лише після завершення й оплати замовлення. Клас OrderCompletionPhoto зберігає фото до та після виконання роботи, що забезпечує контроль якості та підтвердження факту надання послуги.

Отже, визначення варіантів використання та об'єктно-орієнтованої структури дало змогу сформувати логічну модель вебплатформи. Виділення основних акторів, прецедентів і класів системи створює основу для подальшого проєктування бази даних, серверної частини та користувацького інтерфейсу. Запропонована структура враховує основні бізнес-процеси клінінгової компанії, зокрема створення замовлень, планування роботи майстрів, контроль виконання, оплату та підтвердження якості наданих послуг.

2.2 Розробка бази даних

Для збереження даних вебплатформи використано реляційну базу даних PostgreSQL, розміщену на хмарній платформі Neon. Використання реляційної моделі є доцільним, оскільки основні об'єкти системи мають чітко визначені зв'язки між собою: користувачі створюють замовлення, замовлення містять послуги, до замовлень призначаються майстри, а графік роботи формується на основі часових слотів. PostgreSQL забезпечує підтримку зовнішніх ключів,

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

транзакцій, індексів та обмежень цілісності, що є важливим для коректного бронювання замовлень і недопущення конфліктів у розкладі [22].

Структуру бази даних наведено на рис. 2.5, який відображає основні таблиці бази даних та зв'язки між ними.

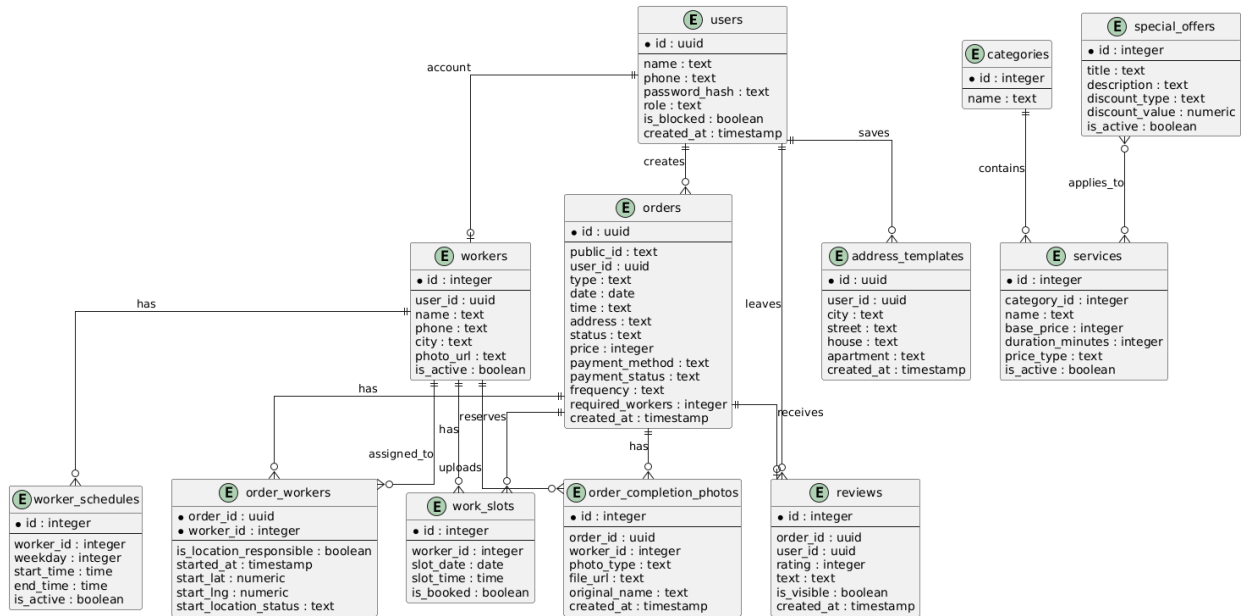


Рис. 2.5 – ER-діаграма бази даних

Центральною таблицею є **orders**, оскільки вона зберігає інформацію про створені замовлення: тип послуги, дату, час, адресу, статус, вартість, спосіб і стан оплати, кількість необхідних майстрів та інші параметри. Кожне замовлення пов'язане з користувачем через поле **user_id**, що дає змогу відображати замовлення у клієнтському кабінеті та обмежувати доступ до них лише власнику.

Таблиця **users** зберігає облікові записи користувачів системи. Для кожного користувача зберігаються ім'я, номер телефону, хеш пароля, роль і стан блокування. Поле **role** визначає рівень доступу користувача до системи, зокрема клієнтський, майстерський або адміністративний. Такий підхід дає змогу реалізувати розмежування прав доступу на рівні серверної логіки.

Для зберігання даних майстрів використовується таблиця **workers**. Вона містить ім'я, телефон, місто, фото, ознаку активності та зв'язок із користувачем через **user_id**. Окрема таблиця **order_workers** реалізує зв'язок між замовленнями та майстрами, оскільки одне замовлення може виконуватися кількома майстрами, а один майстер може бути призначений до багатьох замовлень. У цій

таблиці також зберігається інформація про відповідального за геопозицію майстра та дані перевірки місцезнаходження під час початку роботи.

Для планування роботи персоналу передбачено таблиці `worker_schedules` та `work_slots`. Таблиця `worker_schedules` описує типовий графік роботи майстра за днями тижня, а `work_slots` зберігає конкретні часові слоти на певну дату. Таблиця `work_slots` використовується для бронювання часу під замовлення та запобігання одночасному призначенню майстра на кілька робіт.

Послуги системи зберігаються в таблиці `services`, яка пов'язана з таблицею `categories`. Для кожної послуги зазначається назва, категорія, базова вартість, тривалість виконання, тип розрахунку ціни та стан активності. Ці дані використовуються під час формування кошика замовлення, автоматичного розрахунку вартості та визначення тривалості робіт.

Окрему роль у структурі бази даних відіграє таблиця `special_offers`, яка використовується для збереження спеціальних пропозицій і знижок. Вона дає змогу адміністратору задавати умови акцій, які можуть застосовуватися під час вибору певних комбінацій послуг. Це забезпечує гнучкість системи без необхідності змінювати програмний код під час оновлення маркетингових умов.

Таблиця `reviews` використовується для зберігання відгуків клієнтів. Вона пов'язана із замовленням і користувачем, що дає змогу гарантувати, що відгук залишено саме після реального виконання послуги. Такий підхід підвищує достовірність відгуків, які можуть відображатися на головній сторінці вебплатформи.

Для підтвердження якості виконання робіт створено таблицю `order_completion_photos`. Вона зберігає фото до та після виконання замовлення, посилання на файл, тип фото, час завантаження та майстра, який додав зображення. Ця таблиця пов'язана із замовленнями та майстрами, що дає змогу використовувати фотофіксацію як доказ виконаної роботи.

Таблиця `address_templates` зберігає адреси, які клієнт уже використовував під час оформлення замовлень. Це дає змогу повторно використовувати власні

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

адреси у формі створення замовлення, не відкриваючи доступ до адрес інших користувачів.

Для забезпечення цілісності даних у базі використовуються зовнішні ключі між основними таблицями, наприклад, замовлення пов'язується з користувачем, фотофіксація – із замовленням і майстром, а призначення майстрів – із таблицями замовлень та працівників. Це запобігає появі неузгоджених записів і дає змогу підтримувати коректну структуру даних. Також система працює з бронюванням часових слотів, і тому важливим є використання транзакцій під час створення замовлення та призначення майстрів. У межах однієї транзакції перевіряється доступність майстрів, резервуються відповідні часові слоти та створюються зв'язки між замовленням і майстрами. Такий підхід зменшує ризик ситуації, коли два клієнти одночасно бронюють один і той самий час. Для підвищення швидкодії запитів було передбачено використання індексів, зокрема для пошуку замовлень за користувачем, перевірки зайнятості слотів, отримання графіків майстрів і вибірки відгуків, це, в свою чергу, дає змогу швидше обробляти типові запити, які часто виконуються у клієнтському, адміністративному та майстерському кабінетах.

Отже, розроблена структура бази даних забезпечує збереження всіх основних даних вебплатформи, підтримує зв'язки між користувачами, замовленнями, майстрами, послугами, графіками, оплатами, відгуками та фотофіксацією. Використання зовнішніх ключів і окремих зв'язувальних таблиць дає змогу забезпечити цілісність даних, коректне планування роботи майстрів і подальше масштабування системи.

2.3 Проєктування та реалізація алгоритмів роботи системи

Для коректної роботи вебплатформи було спроектовано основні алгоритми, які відповідають за створення замовлення, бронювання часу, призначення майстрів, виконання робіт, оплату та застосування спеціальних пропозицій. Збереження послідовності виконання операцій і недопущення конфліктів під час бронювання є важливим, оскільки система працює з часовими слотами та декількома ролями користувачів.

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

Одним із ключових алгоритмів є створення замовлення та бронювання часових слотів відповідних майстрів, оскільки цей процес передбачає вибір послуг, автоматичний розрахунок параметрів замовлення, перевірку доступності майстрів і бронювання часу. Алгоритм процесу наведено на рис. 2.6.



Рис. 2.6 – Алгоритм створення замовлення та бронювання слотів

Як показано на рис. 2.6, після вибору послуг система формує кошик замовлення, розраховує загальну вартість, тривалість робіт і необхідну кількість майстрів, а вже після введення адреси та вибору дати й часу виконується перевірка доступності персоналу. Якщо необхідна кількість майстрів доступна, система створює замовлення, призначає майстрів і резервує відповідні часові слоти. Якщо їх недостатньо, клієнту пропонується обрати іншу дату або час.

Окремим етапом роботи системи є призначення майстрів до замовлення (рис. 2.7). Кількість необхідних майстрів залежить від загальної тривалості робіт і кількості активних майстрів у системі. Це дає змогу адаптувати фактичну тривалість візиту до обсягу замовлення та не перевищувати доступну кількість персоналу. Система спочатку отримує дані замовлення, визначає тривалість

робіт і розраховує потрібну кількість майстрів, після цього виконується перевірка активних майстрів і пошук майстрів, які мають вільні часові слоти на вибраний час. Якщо доступних майстрів достатньо, система автоматично призначає їх до замовлення й резервує слоти, але якщо автоматичне призначення неможливе, замовлення переходить до ручного опрацювання адміністратором.

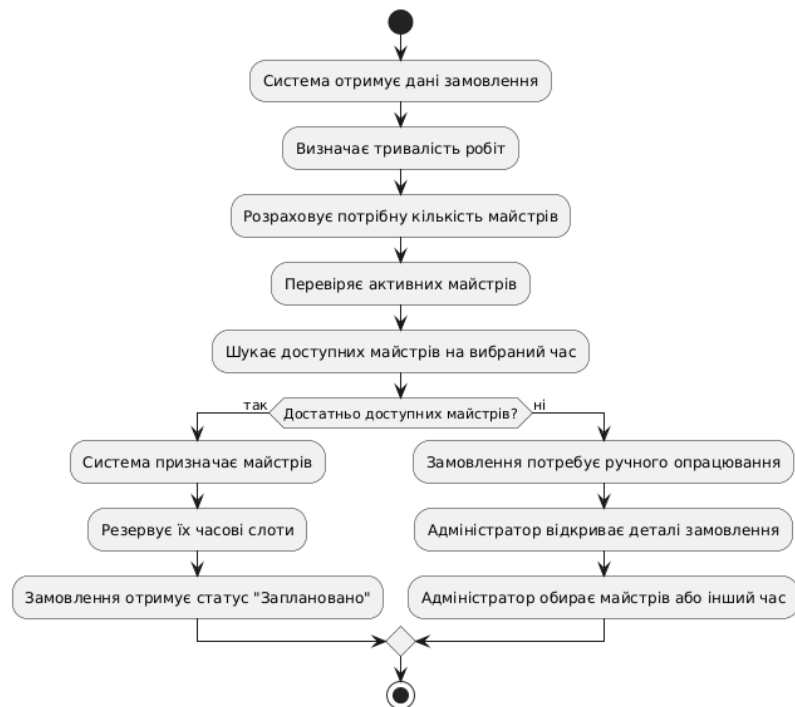


Рис. 2.7 – Алгоритм призначення майстрів до замовлення

Алгоритм виконання замовлення майстром (рис. 2.8) передбачає послідовну зміну етапів роботи та фіксацію факту виконання. Після входу в особистий кабінет майстер відкриває призначене йому замовлення, переглядає адресу, час виконання, контактні дані клієнта та перелік послуг. Далі він змінює статус замовлення на «В дорозі», що дає змогу системі зафіксувати початок виконання заявки та показати, що замовлення вже не перебуває лише на етапі очікування, а після прибуття за адресою, майстер натискає кнопку початку роботи та передає геопозицію. У разі, якщо під час виконання виникають проблеми, наприклад майстер не може знайти під'їзд, клієнт не відповідає або адреса потребує уточнення, у деталях замовлення зазначено номер телефону, за яким майстер зможе зв'язатися з клієнтом. Якщо майстер відповідає за передачу геопозиції, то система перевіряє збіг його місцезнаходження з адресою

замовлення. Зазначений механізм перевірки призначений не для обмеження дій майстра, а для підтвердження факту його прибуття до місця виконання робіт. Такий підхід надає адміністратору можливість здійснювати контроль за виконанням замовлень дистанційно та переконуватися, що роботи фактично розпочато за вказаною адресою. Після початку роботи майстер завантажує фото до прибирання, виконує послугу, а після завершення додає фото результату, і це дає змогу підтвердити виконання послуги й у разі потреби швидко перевірити якість прибирання. Саме тому завершення замовлення можливе лише за наявності обох типів фотографій. В алгоритмі передбачено контроль переходів між статусами. Замовлення не може одразу перейти в стан «Завершено», якщо перед цим не було розпочато роботу або не додано необхідні фото, це зменшує кількість випадкових помилок з боку майстра та допомагає зберігати правильну історію виконання замовлення. Для адміністратора це також спрощує контроль, оскільки кожне замовлення має зрозумілий поточний стан.

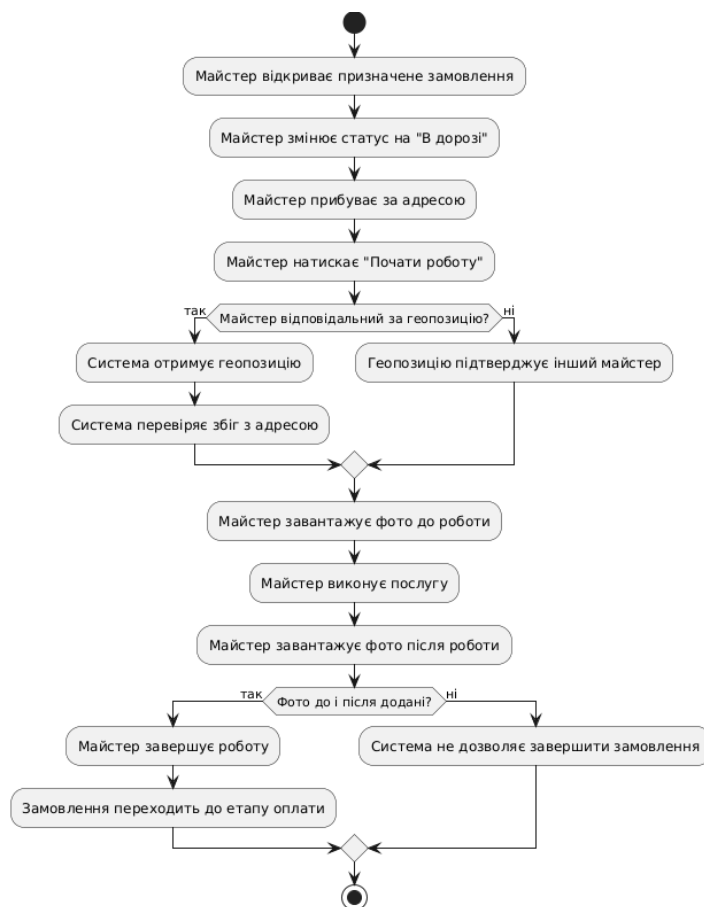


Рис. 2.8 – Алгоритм виконання замовлення майстром

Після завершення виконання послуги система переходить до обробки оплати готівкою або онлайн через LiqPay. На рис. 2.9 показано, що після завершення майстер обирає спосіб оплати, якщо клієнт оплачує онлайн, система формує платіж LiqPay, після чого клієнт переходить на сторінку оплати, після успішного платежу система отримує callback-запит і оновлює статус оплати, проте якщо клієнт оплачує готівкою, майстер підтверджує отримання коштів, а вже після підтвердження оплати завершене замовлення переноситься в архів.

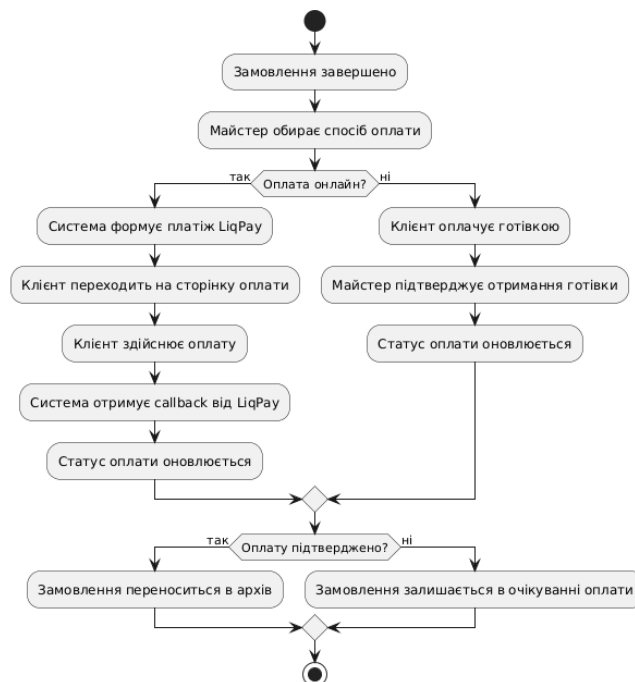


Рис. 2.9 – Алгоритм оплати замовлення

На рис. 2.10 показано, що після додавання послуг до кошика система перевіряє активні спеціальні пропозиції і, якщо умови пропозиції виконані, система визначає її тип. Для рекомендаційних пропозицій клієнту відображається додаткова послуга зі знижкою, яку він може додати до кошика, а для комбінаційних пропозицій знижка застосовується автоматично, якщо в кошику вже присутня необхідна комбінація послуг, також знижка може застосовуватися як до окремої рекомендованої послуги, так і до всього замовлення, залежно від типу пропозиції. Додатково в системі реалізовано алгоритм регулярних замовлень, клієнт обирає щомісячне, двотижнєве або щотижнєве прибирання, система формує набір майбутніх дат і створює відповідні замовлення. Для регулярних прибирань застосовується окрема знижка

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

залежно від періодичності. Якщо майбутня дата припадає на вихідний день або на неї немає доступної команди майстрів, то система не переносить замовлення автоматично без згоди клієнта, а позначає його як таке, що потребує узгодження і в цей процес втручається адміністратор, для узгодження нового варіанту.

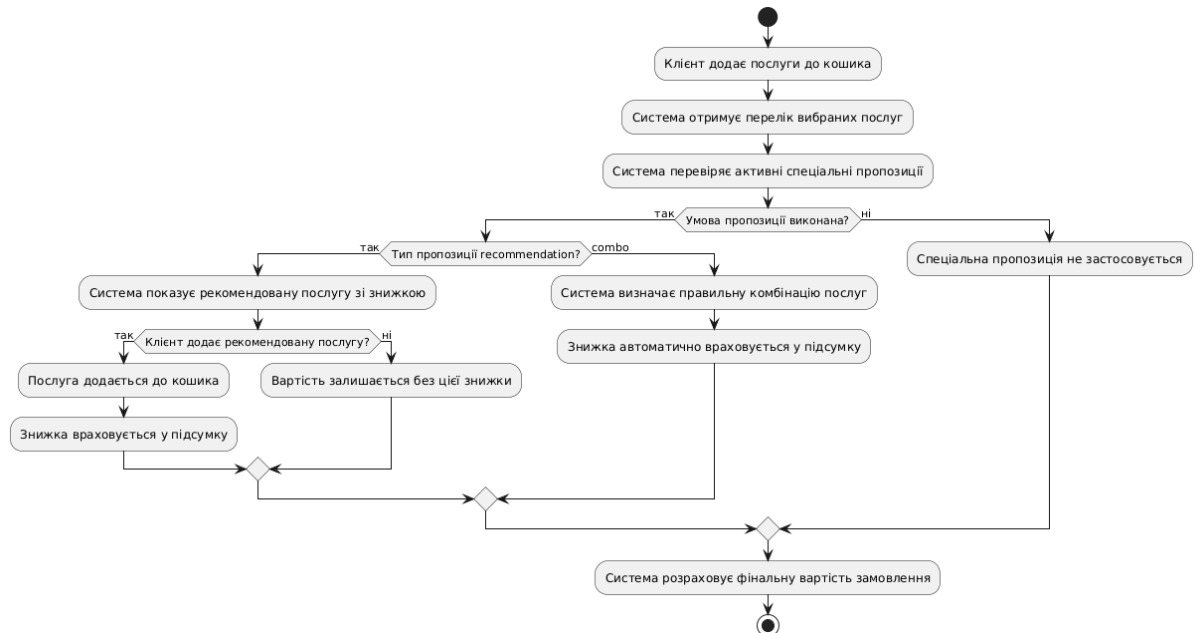


Рис. 2.10 – Алгоритм обробки спеціальних пропозицій

Клієнт може залишити відгук, але лише після завершення замовлення й підтвердження оплати, що дає змогу зберігати в системі лише відгуки, пов'язані з реально виконаними замовленнями. Такий підхід підвищує рівень довіри до відгуків, які можуть відображатися в адміністративній панелі та на головній сторінці платформи.

Щоб спростити повторне оформлення замовлення, реалізовано алгоритм роботи з шаблонами адрес, що дає змогу прискорити створення повторних замовлень. Оскільки адреса користувача належить до персональних даних, система зберігає лише ті адреси, які використовував конкретний клієнт, і не надає доступу до адрес інших користувачів. Такий підхід відповідає вимогам щодо обмеження доступу до персональної інформації та збереження її конфіденційності в цифровому середовищі [23].

Загалом спроектовані алгоритми забезпечують узгоджену роботу основних модулів вебплатформи, які охоплюють повний цикл обробки замовлення від

вибору послуг і бронювання часу до виконання робіт, оплати, підтвердження якості та подальшого збереження результатів у системі.

2.4 Реалізація функціоналу вебплатформи

Авторизація та розмежування ролей користувачів

Однією з частин реалізації вебплатформи є авторизація користувачів і розмежування доступу між різними ролями, оскільки система передбачає роботу не одного універсального користувача, а кількох типів учасників із різними правами та сценаріями взаємодії. Виділено три основні ролі: клієнт, адміністратор і майстер. Клієнт повинен мати доступ лише до власного кабінету та своїх замовлень, адміністратор – до панелі керування всією системою, а майстер тільки до тих заявок, які були призначені саме йому. Перевірки лише факту автентифікації недостатньо: необхідно також визначати, які дії дозволені користувачеві.

На серверній частині авторизація реалізована за допомогою JWT-токена, який формується після успішного входу користувача в систему і зберігається у cookie, завдяки чому браузер може автоматично передавати його під час наступних запитів до захищених маршрутів. Для такого способу зберігання важливо правильно налаштувати параметри cookie, зокрема HttpOnly та SameSite, оскільки вони зменшують ризик доступу до токена з клієнтського JavaScript і дозволяють контролювати його передачу між сайтами [24]. Після отримання запиту сервер перевіряє токен, дістає з нього ідентифікатор користувача, а потім додатково звертається до бази даних, щоб отримати актуальну роль, стан блокування та переконатися, що обліковий запис усе ще існує. Для захисту серверних маршрутів були створені окремі middleware, які дають змогу виконати попередню перевірку запиту ще до того, як він потрапить в основну логіку маршруту, і не дублювати однаковий код у кожному обробнику [25]. Загальний middleware перевіряє, чи користувач авторизований, а додаткові перевірки використовуються для окремих частин системи. Якщо роль користувача не відповідає потрібній, сервер не передає запит далі, а одразу

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

повертає помилку доступу. Це захищає службові операції навіть у випадку, якщо користувач спробує звернутися до API напряду.

На клієнтській частині реалізовано перевірку ролей через маршрутизатор, для чого використано navigation guards Vue Router, які дозволяють перехоплювати перехід між сторінками та за потреби перенаправляти користувача на інший маршрут. Ця перевірка не замінює серверний захист, але робить роботу з інтерфейсом зрозумілішою, а користувач не потрапляє на сторінки, які не відповідають його ролі. Окремо було враховано ситуацію, коли адміністратор блокує користувача, тому навіть за наявності дійсного токена сервер під час кожного захищеного запиту перевіряє актуальний стан облікового запису в базі даних і, якщо користувач заблокований, очищає cookie та забороняє подальший доступ. Фрагменти реалізації авторизації, middleware для перевірки ролей користувачів і клієнтського обмеження доступу через маршрутизатор наведено в додатку А.

Створення замовлення та розрахунок його параметрів

Створення замовлення реалізовано на серверній частині в модулі роботи із замовленнями. Для авторизованого клієнта використовується маршрут створення замовлення, а для користувача без реєстрації – маршрут гостьового замовлення. Сервер отримує список вибраних послуг, дату, час, адресу, коментар і параметри регулярності, після чого перевіряє послуги в таблиці services, бере їхню ціну та тривалість, формує загальну назву замовлення, розраховує суму й сумарний час виконання. Розрахунок кількості майстрів і фактичної тривалості візиту використовує допоміжний модуль, у якому загальна тривалість робіт переводиться в рекомендовану кількість майстрів. У реалізованій логіці один майстер орієнтовно покриває до 120 хвилин роботи, тому якщо замовлення займає більше часу, система збільшує кількість потрібних працівників, але додатково обмежує її фактичною кількістю активних майстрів, потім розраховується тривалість візиту, загальний час робіт ділиться між обраною кількістю майстрів і округлюється до годинного слота. Наприклад, замовлення на 240 хвилин може бути виконане двома майстрами за 120 хвилин, і саме ці 120

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

хвилин будуть використовуватися під час перевірки доступних слотів. Для гостьового замовлення додатково реалізовано створення або пошук користувача за номером телефону. Сервер приводить телефон до єдиного формату, після чого шукає в таблиці users запис із таким номером або службовою електронною адресою. Якщо користувача знайдено, замовлення прив'язується до нього. Якщо ні, створюється новий обліковий запис і тільки після цього система формує public_id, структуровану адресу, ціну, тривалість, кількість майстрів, дані про оплату й записує замовлення в таблицю orders, уже маючи реальний user_id. Відповідні фрагменти реалізації окремих модулів наведено в додатку Б.

Бронювання часових слотів майстрів

Ця логіка реалізована на серверній частині в модулі роботи із замовленнями, оскільки саме сервер повинен остаточно вирішувати, чи може замовлення бути створене на вибраний час чи ні. Система розраховує тривалості візиту та кількості майстрів і визначається, які часові слоти потрібні для виконання замовлення, після чого перевіряє таблицю work_slots. Якщо замовлення займає дві години, то перевіряється не тільки стартовий час, а й наступний слот, оскільки час має бути вільним протягом усього періоду виконання. Під час фактичного бронювання слотів використовується механізм блокування рядків через FOR UPDATE SKIP LOCKED. Це дає змогу уникнути ситуації, коли два клієнти майже одночасно намагаються створити замовлення на один і той самий час. Якщо один запит уже працює з певними слотами, інший не очікує їх звільнення та не використовує їх повторно, а автоматично пропускає заблоковані записи. Після отримання вільних слотів система групує їх за майстрами та перевіряє, чи кожен кандидат має всі потрібні часові проміжки. Сервер вибирає лише тих працівників, які вільні на весь час виконання, а не лише на першу годину. Якщо доступних майстрів достатньо, їхні слоти оновлюються в таблиці work_slots і позначаються як заброньовані, а якщо ні, то бронювання не виконується, а клієнт отримує повідомлення про недоступність вибраного часу. Реалізацію цього механізму наведено в додатку В.

Реалізація графіка майстрів

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

Графік майстрів є окремим адміністративним модулем, у якому передбачено два рівні роботи з розкладом: тижневий шаблон і окремий денний графік. Тижневий шаблон зберігається в таблиці worker_schedules і описує стандартні години роботи майстра на тиждень, а конкретні слоти на конкретну дату зберігаються в таблиці work_slots. Адміністратор обирає майстра та дату, після чого сервер перевіряє, чи є для цього майстра вже збережені слоти в таблиці work_slots, якщо такі записи є. Система повертає саме денний графік і додатково позначає, які години вже заброньовані замовленнями, а якщо окремого графіка на вибрану дату немає, сервер формує набір годин на основі тижневого шаблону майстра з таблиці worker_schedules. У клієнтській частині ці дані відображаються як список часових кнопок, де адміністратор може додати або прибрати доступні години. Під час збереження графіка система окремо перевіряє, які години вже зайняті замовленнями, і якщо певний слот уже заброньований, його не можна видалити, навіть якщо адміністратор змінює графік майстра на цей день, видалення можливе тільки для вільних годин. Фрагменти реалізації завантаження денного графіка, роботи з тижневим шаблоном і збереження слотів наведено в додатку Г.

Реалізація оплати замовлення

Оплату реалізовано з підтримкою двох варіантів: готівкою та онлайн через LiqPay. Дані про оплату зберігаються безпосередньо в таблиці orders, де для замовлення фіксується спосіб оплати, статус платежу, сума та час підтвердження. Для онлайн-оплати сервер формує платіж LiqPay на основі публічного ідентифікатора замовлення, суми, опису платежу та ключів платіжної системи, далі клієнт переходить на сторінку оплати LiqPay, а після завершення операції сервіс надсилає callback, сервер розшифровує отримані дані, перевіряє підпис, визначає статус платежу та оновлює відповідне замовлення в базі даних. Якщо платіж успішний, статус оплати змінюється на paid, а дата підтвердження зберігається в полі payment_confirmed_at. Для оплати готівкою окрема платіжна сторінка не формується, після завершення роботи майстер у своєму кабінеті обирає оплату готівкою та підтверджує отримання коштів, сервер оновлює

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

замовлення, встановлює спосіб оплати «cash», змінює статус платежу на «paid» та зберігає суму платежу. Фрагменти реалізації формування LiqPay-платежу, обробки callback і оновлення статусу оплати наведено в додатку Д.

Висновки до другого розділу

У другому розділі було спроектовано вебплатформу не як сторінку для приймання заявок, а як систему, де замовлення проходить повний цикл від вибору послуг клієнтом до виконання роботи майстром, підтвердження оплати та подальшого контролю з боку адміністратора. Для цього було визначено ролі користувачів, описано їхні можливості та обмеження доступу. На цьому етапі також сформовано вимоги до системи, які охоплюють оформлення замовлення, автоматичний розрахунок вартості й тривалості, призначення майстрів, роботу з графіками, фотофіксацію, оплату та перегляд історії виконаних робіт.

Для кращого розуміння роботи платформи було побудовано діаграми варіантів використання та описано основні алгоритми, які відповідають за найбільш важливі процеси. Моделювання дало змогу побачити, де система має виконувати перевірки, які дані повинні передаватися між етапами та в яких місцях потрібно захистити платформу від помилок.

Описано структуру бази даних, яка зберігає інформацію про користувачів, послуги, категорії, замовлення, адреси, майстрів, графіки, часові слоти, відгуки та оплату, і завдяки цьому система може не записати факт заявки, а підтримувати її подальший рух, показувати актуальний стан різним користувачам і зберігати потрібну історію для адміністратора.

Практична частина розділу показує, як спроектовані рішення перенесено в код. Було описано авторизацію та розмежування ролей, створення авторизованих і гостьових замовлень, розрахунок тривалості робіт і кількості майстрів, бронювання часових слотів, налаштування графіка майстрів та обробку оплати. Отже, у другому розділі було сформовано уявлення про структуру, логіку роботи та реалізацію основних модулів, що створює основу для подальшого створення програмного комплексу для планування, оформлення й контролю виконання клінінгових замовлень.

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

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ПЛАНУВАННЯ ТА КОНТРОЛЮ КЛІНІНГОВОЇ КОМПАНІЇ

3.1 Порядок встановлення та налаштування параметрів системи

Порядок встановлення та налаштування вебплатформи планування та контролю клінінгової компанії передбачає підготовку та конфігурацію комплексу програмних компонентів, які спільно забезпечують функціонування системи. Архітектура включає клієнтську частину, сервер застосунку, базу даних, сховище файлів і зовнішній платіжний сервіс. Кожен із зазначених компонентів виконує окремі завдання, водночас підтримуючи постійну взаємодію з іншими складовими системи через механізми обміну даними. Такий підхід забезпечує цілісність роботи вебплатформи та коректне виконання основних бізнес-процесів. Завдяки узгодженій роботі всіх компонентів реалізуються функції оформлення та обробки замовлень, планування графіків виконання робіт, координації діяльності працівників, моніторингу стану замовлень, зберігання супровідної інформації та здійснення платіжних операцій. Загальну схему розгортання вебплатформи наведено на рис. 3.1.

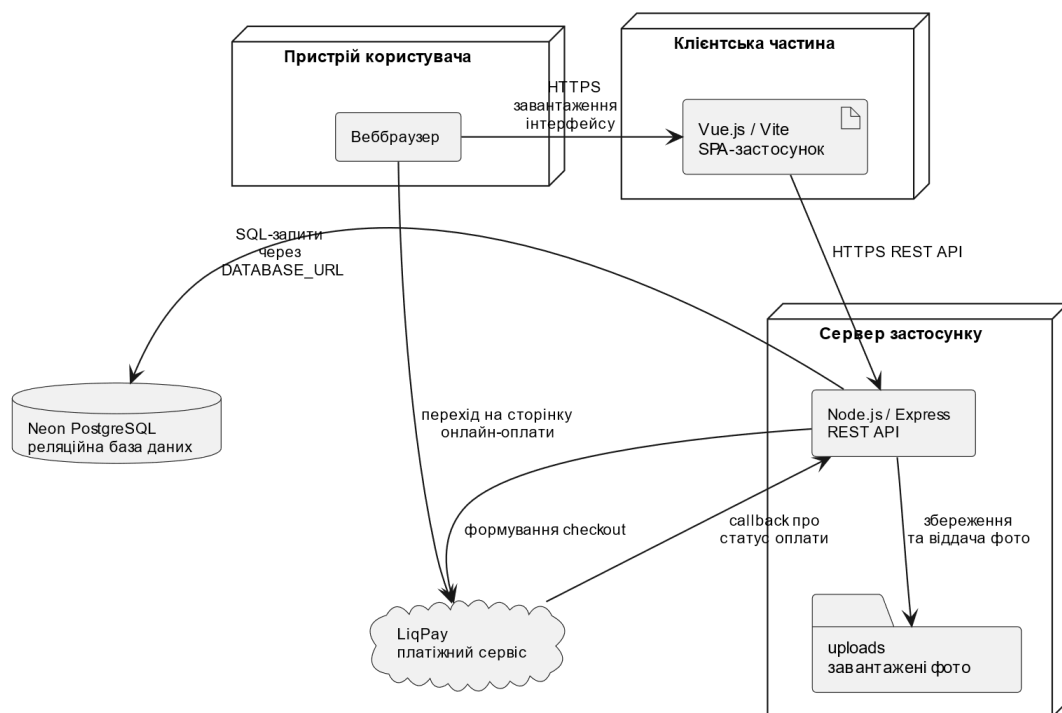


Рис. 3.1 – Діаграма розгортання вебплатформи

На діаграмі розгортання відображено взаємодію користувача з вебплатформою через веббраузер, встановлений на персональному пристрої. Після відкриття системи завантажується клієнтський застосунок, створений на основі технологій Vue 3 та Vite за архітектурою односторінкового вебзастосунку. Його призначення полягає у відображенні інтерфейсу, навігації між функціональними модулями, опрацюванні дій користувача та формуванні запитів до серверного рівня. Прямий доступ до бази даних із боку клієнта не передбачений. Усі операції з інформацією виконуються через сервер, де здійснюється перевірка вхідних даних, обробка бізнес-правил та виконання необхідних обчислень. Такий підхід сприяє чіткому розподілу функцій між рівнями архітектури, підвищує захищеність службових відомостей і відповідає базовим принципам безпечної веброзробки [26].

Серверний рівень побудовано на платформі Node.js із використанням фреймворку Express. Саме тут зосереджено виконання основних операцій програмного комплексу. Сервер приймає та опрацьовує запити від клієнтського застосунку, виконує автентифікацію користувачів, визначає права доступу відповідно до ролі, формує відповіді для інтерфейсу та координує роботу інших компонентів. Крім цього, на його стороні реалізовано створення і супровід замовлень, розрахунок вартості послуг, збереження адресної інформації, обробку платіжних операцій і контроль зміни статусів на різних етапах виконання робіт. Через сервер також функціонують адміністративна панель, особистий кабінет клієнта та робочий кабінет майстра. Обмін даними між клієнтською та серверною складовими здійснюється за допомогою HTTP-запитів до REST API, тоді як механізми захисту мережевої взаємодії визначаються особливостями середовища розгортання.

Окремою складовою вебплатформи є база даних Neon PostgreSQL, призначена для зберігання структурованої інформації, необхідної для функціонування сервісу. У ній розміщуються відомості про користувачів, ролі доступу, перелік послуг і їх категорії, замовлення, графіки роботи майстрів, доступні часові інтервали, відгуки клієнтів, адресні шаблони, персонал та інші

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

сутності предметної області. Підключення до бази даних налаштовується за допомогою змінної середовища DATABASE_URL, що дає змогу змінювати параметри доступу без внесення змін до вихідного коду. Такий спосіб конфігурації спрощує адміністрування програмного комплексу, підвищує безпеку конфіденційних параметрів і полегшує перенесення вебплатформи між різними середовищами розробки, тестування та експлуатації.

У структурі серверного рівня також передбачено файлове сховище, призначене для зберігання матеріалів, що завантажуються під час роботи із замовленнями. Насамперед це фотографії, які клієнт додає під час оформлення заявки, а також зображення, що використовуються для фіксації результатів виконаних робіт. Такі файли розміщуються в окремому каталозі на стороні сервера та обробляються через серверний застосунок. Подібна організація зберігання дає змогу централізовано керувати медіаданими, контролювати доступ до них і пов'язувати завантажені матеріали з відповідними замовленнями. Крім того, це унеможливлює прямий доступ користувачів до внутрішньої файлової структури та сприяє підвищенню рівня безпеки.

Важливою складовою конфігурації є інтеграція із зовнішнім платіжним сервісом LiqPay, який використовується для проведення онлайн-оплати. Під час виконання платіжної операції сервер формує необхідні параметри та передає їх до платіжної системи. Після завершення обробки платежу сервіс надсилає службове повідомлення з результатом транзакції, на підставі якого оновлюється відповідний запис у базі даних. Незважаючи на те, що фінансові розрахунки здійснюються поза межами вебплатформи, інформація про їх результат зберігається всередині програмного комплексу та використовується під час подальшої обробки замовлення. Такий механізм забезпечує коректну взаємодію між зовнішнім платіжним інструментом і внутрішньою системою обліку, а також виключає дублювання розрахункових операцій.

Перед початком експлуатації вебплатформи необхідно виконати підготовку клієнтської та серверної складових, оскільки їх функціонування здійснюється у вигляді двох незалежних процесів. На першому етапі для кожної

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

частини встановлюються необхідні програмні залежності, після чого налаштовуються параметри середовища. Для клієнтського застосунку ключовим параметром є змінна `VITE_API_ORIGIN`, яка визначає адресу сервера для надсилання запитів. Особливістю середовища Vite є використання префікса `VITE_` для всіх змінних, що мають бути доступними у клієнтському коді. За відсутності такого префікса значення не буде включене до зібраної версії застосунку [27]. З цієї причини адреса серверного рівня винесена до окремого конфігураційного параметра, що спрощує зміну налаштувань під час переходу між різними середовищами розробки та розгортання.

Для серверної складової задається набір параметрів, які визначають особливості її роботи та взаємодії із зовнішніми ресурсами. До них належать номер мережевого порту, рядок підключення до бази даних, секретний ключ для формування токенів автентифікації, адреса клієнтського застосунку для налаштування дозволених запитів, а також реквізити інтеграції з платіжним сервісом LiqPay. Після завершення конфігурації запускається процес Node.js, який забезпечує обробку API-запитів, взаємодію з базою даних, керування авторизацією користувачів, роботу із замовленнями, розкладом та платіжними операціями. Клієнтська складова запускається за допомогою середовища Vite та надає користувачам графічний інтерфейс для доступу до функціональних можливостей вебплатформи.

3.2 Структура інтерфейсу вебплатформи

Інтерфейс вебплатформи спроектовано з урахуванням особливостей роботи різних категорій користувачів, що зумовило поділ функціоналу на окремі логічні модулі. Структура застосунку включає публічну головну сторінку, форму оформлення замовлення, особистий кабінет клієнта, робочий кабінет майстра та адміністративну панель. Кожен із зазначених розділів призначений для виконання визначеного кола завдань і надає доступ лише до тих функцій, які відповідають ролі користувача та його повноваженням у системі.

Першою сторінкою, з якою взаємодіє користувач, є головна сторінка вебплатформи, наведена на рис. 3.2. Вона не лише знайомить користувача із

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

сервісом, а й одразу пропонує перейти до оформлення замовлення або попереднього розрахунку вартості прибирання.

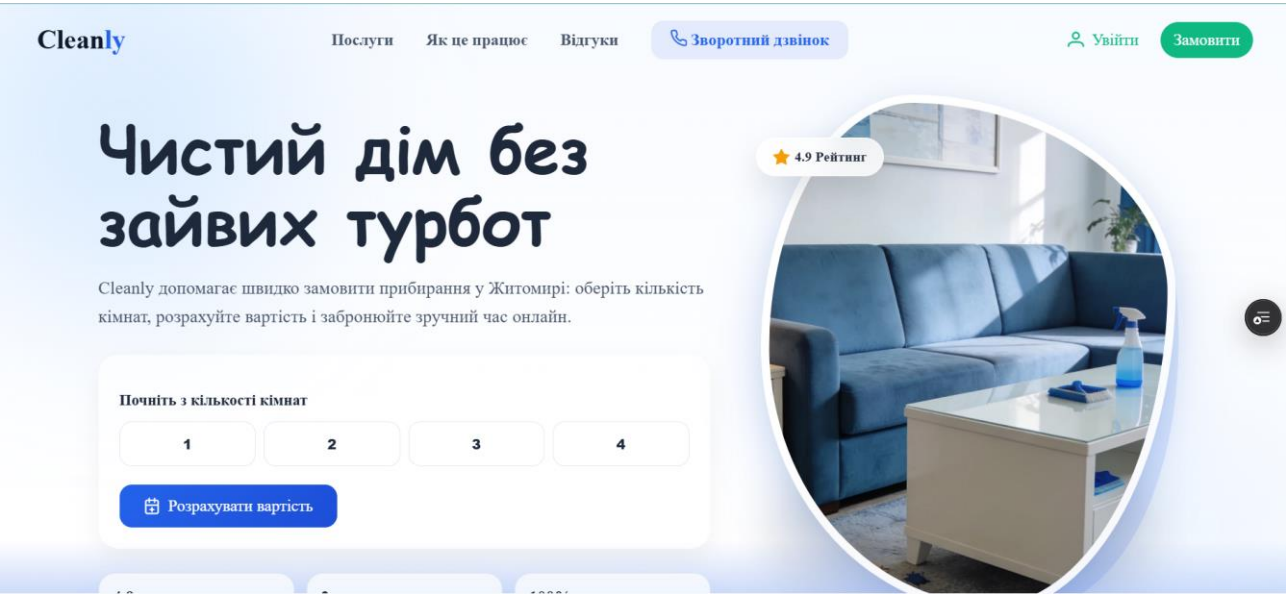


Рис. 3.2 – Головна сторінка вебплатформи

На рис. 3.2 зображено головну сторінку вебплатформи, з якої користувач починає роботу із системою. У верхній частині сторінки розміщено навігаційну панель із логотипом, переходами до розділів «Послуги», «Як це працює» та «Відгуки», а також кнопками для входу, замовлення послуги й запиту зворотного дзвінка. Це дозволяє користувачу одразу перейти до потрібної дії: переглянути інформацію про сервіс, авторизуватися або почати оформлення замовлення.

Центральна частина сторінки містить короткий опис сервісу та блок швидкого розрахунку, де користувач може вибрати кількість кімнат і перейти до розрахунку вартості прибирання. Праворуч розміщено тематичне зображення, яке візуально пов'язує сторінку з клінінговими послугами, а також показник рейтингу. Така структура головної сторінки поєднує інформаційну частину з початковим етапом оформлення заявки, тому користувачу не потрібно окремо шукати форму замовлення.

Після переходу до оформлення замовлення користувач потрапляє на форму вибору послуг, фрагмент якої наведено на рис. 3.3. На цьому етапі послуги згруповано за категоріями, зокрема прибирання, хімчистка, миття вікон і послуги для дому. Для кожної послуги відображається назва, орієнтовна тривалість і

вартість, а після вибору вона додається до кошика замовлення. Під час вибору послуг і параметрів замовлення клієнтська частина надсилає запити до серверного API для розрахунку вартості та отримання доступних часових слотів, використовуючи бібліотеку Axios [28]. У правій частині форми користувач бачить поточний склад кошика, вибрану регулярність виконання, суму до сплати, загальний обсяг робіт, кількість потрібних майстрів і фактичний час візиту. Після натискання кнопки перевірки замовлення система повертає користувача до цього блоку, щоб він міг ще раз переглянути підсумок і натиснути кнопку створення замовлення.

Крок 1
Спочатку оберіть послугу

За кількістю кімнат попередньо підібрано послугу. Перевірте кошик і за потреби змініть вибір.

Прибирання 1/10

Хімчистка 0/8

Миття вікон 0/4

Для дому 0/10

10 послуг

Прибирання

Підтримуюче: 1-кімнатна квартира
180 хв
1200 грн

Підтримуюче: 2-кімнатна квартира
240 хв
1800 грн

Підтримуюче: 3-кімнатна квартира
300 хв
2100 грн

Підтримуюче: 4-кімнатна квартира
360 хв
2700 грн

Генеральне прибирання: 1-кімнатна
360 хв
2500 грн

Генеральне прибирання: 2-кімнатна
480 хв
3500 грн

Кошик замовлення
Послуги, тривалість і підсумок

Підтримуюче: 2-кімнатна квартира
4 год
1800 грн

Регулярність

Разово

Щомісяця -5%

Що 2 тижні -10%

Щотижня -15%

Послуги 1800 грн
Обсяг робіт 4 год
Команда 2 майстри
Час фактичного візиту 2 год

До сплати 1800 грн

Очистити

Замовити

Рис. 3.3 – Вибір послуг та кошик замовлення

Наступним етапом користувач заповнює дані про місце та час виконання замовлення (рис. 3.4). Адреса вводиться структуровано: окремо зазначаються місто, район, вулиця, будинок, квартира або офіс, під'їзд і поверх. Нижче розміщено поля вибору дати й доступного часу виконання, а також поле для коментаря, у якому можна залишити додаткові інструкції для майстрів. Такий поділ даних дозволяє системі зберігати адресу не лише як текстовий опис, а як набір окремих параметрів, що надалі використовується під час обробки замовлення та перевірки місця виконання робіт.

Під час введення адреси система перевіряє коректність заповнених даних. Якщо користувач вказує вулицю, якої немає в Житомирі, система повідомляє про

помилку, оскільки для такої адреси неможливо визначити координати [30]. Це важливо для подальшої обробки замовлення, оскільки координати адреси використовуються для порівняння місцезнаходження майстра з місцем виконання робіт. Також система контролює заповнення обов'язкових полів форми, тому у разі створення замовлення без потрібних даних, під відповідним полем відображається повідомлення червоного кольору, а сторінка автоматично переходить до незаповненого або некоректно заповненого поля.

Адреса

Дані збережемо структуровано (зручно для БД і фільтрів).

Місто *

Житомир

Район (необов'язково)

напр. Богунський

Вулиця *

напр. Велика Бердичівська

Будинок *

напр. 32

Квартира/офіс (необов'язково)

напр. 18

Під'їзд (необов'язково)

напр. 1

Поверх (необов'язково)

напр. 4

Підсумкова адреса:

Житомир

Дата *

Оберіть дату

Час *

Оберіть час

Коментар (необов'язково)

Напр.: код домофону, є тварина, додаткові інструкції...

Рис. 3.4 – Заповнення адреси та часу виконання замовлення

Завершальна частина форми створення замовлення наведена на рис. 3.5. У цьому блоці користувач може додати фото для майстрів, наприклад зображення приміщення, меблів або окремих ділянок, на які потрібно звернути увагу перед виконанням робіт. Завантаження фото не є обов'язковим, однак воно допомагає майстрам підготуватися до замовлення ще до виїзду за адресою. Якщо заявку оформлює гість, який не увійшов у власний обліковий запис, система додатково відображає блок контактних даних із полями імені та номера телефону. Для авторизованого клієнта цей блок не показується, оскільки потрібні контактні дані вже зберігаються в його профілі. Після завершення заповнення всіх даних

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

система автоматично повертає користувача до верхньої частини форми, де розміщено кошик замовлення. Це дає змогу ще раз перевірити вибрані послуги, регулярність виконання, суму, тривалість робіт і кількість майстрів перед остаточним підтвердженням заявки.

Фото для майстрів 0/6
Необов'язково. Додайте фото дивана, плями або приміщення, щоб команда краще підготувалась.

Додати фото
JPG, PNG або WEBP до 6 МБ

Контакти
Після створення замовлення ми прив'яжемо його до номера телефону.

Ім'я **Телефон ***

напр. Яна +38 (0) _ _ - _ _

Вкажіть коректний номер телефону

❶ Всі поля, позначені зірочкою (*), є обов'язковими для заповнення.

↑ Перевірити замовлення
Після заповнення даних поверніться до кошика, щоб перевірити послуги, час і суму перед оформленням.

Рис. 3.5 – Додавання фото та контактних даних замовника

Після підтвердження заявки система створює замовлення та відображає повідомлення про успішне завершення операції (рис. 3.6). На сторінці показується унікальний номер замовлення, вибрана послуга, дата й час виконання, адреса та сума до сплати. Наявність номера замовлення дозволяє однозначно ідентифікувати заявку під час подальшого перегляду в особистому кабінеті або під час звернення до адміністратора. На цьому етапі користувач може перейти до особистого кабінету або повернутися на головну сторінку. Якщо замовлення створював гість, система повідомляє, що після авторизації він зможе переглядати статус замовлення, інформацію про призначених майстрів, дані про оплату та фото виконання робіт. Таким чином, після створення заявки користувач одразу отримує підтвердження, що замовлення збережене в системі та може бути оброблене далі. Це також завершує сценарій оформлення замовлення на стороні клієнта. Надалі заявка стає доступною для подальшої обробки в адміністративній частині системи.

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

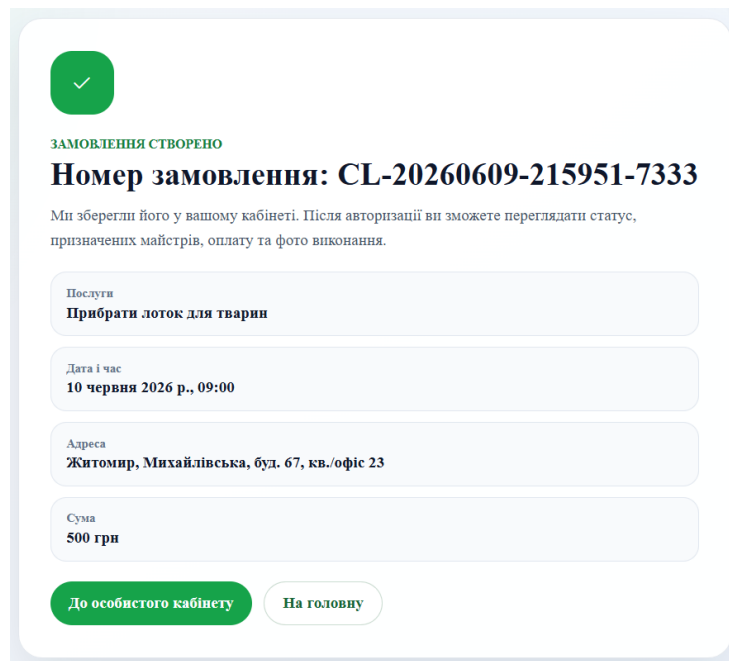


Рис. 3.6 – Повідомлення про успішне створення замовлення

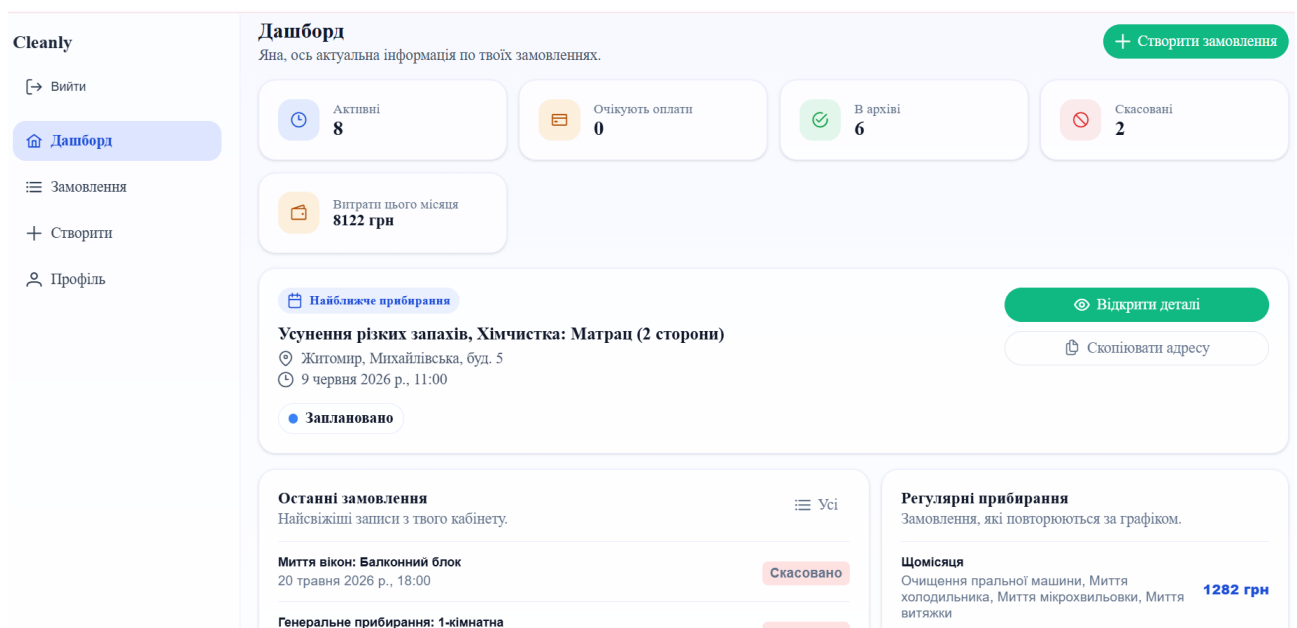


Рис. 3.7 – Головна сторінка особистого кабінету клієнта

На рис. 3.7 наведено головну сторінку особистого кабінету клієнта. У лівій частині інтерфейсу розміщено меню навігації з переходами до дашборду, списку замовлень, сторінки створення нового замовлення та профілю користувача. У центральній частині сторінки відображаються узагальнені показники: кількість активних, архівних і скасованих замовлень, замовлень, що очікують оплати, а також витрати за поточний місяць. Нижче розташовано блок найближчого прибирання із зазначенням послуги, адреси, дати, часу та поточного статусу.

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

Також на сторінці відображаються останні замовлення користувача та список регулярних прибирань.

Деталі замовлення

ID: CL-20260605-173624-5292

НазадСкопіювати адресуРедагуватиСкасувати

Тип

Усунення різких запахів, Хімчистка: Матрац (2 сторони)

Дата і час

9 червня 2026 р., 11:00

Фактична тривалість

2 год

Майстри

Яна, Володимир

Адреса

Житомир, Михайлівська, буд. 5

Статус

Заплановано

Сума

1800 грн

Оплата

Не обрано

Очікує оплати

МАТЕРІАЛИ ДЛЯ КОМАНДИ

Фото від клієнта

0 ФОТО

Фото не додані. За бажанням можна прикріпити зображення меблів, плям або приміщення.

Додати фото

Етап виконання

Час і команда зарезервовані

235

Заплановано

Прийнято

Замовлення створено

Заплановано

Час і команда зарезервовані

Поточний етап

В дорозі

Майстер виїхав до вас

В роботі

Майстри виконують прибирання

Завершено

Прибирання виконано

Рис. 3.8 – Деталі замовлення

На рис. 3.8 наведено сторінку перегляду деталей замовлення. Тут користувач може переглянути основну інформацію про заявку: обрану послугу, дату та час виконання, адресу, тривалість робіт, призначених майстрів, статус замовлення та інформацію про оплату. У нижній частині сторінки відображається поточний етап виконання робіт. Система послідовно змінює статус замовлення від моменту його створення до завершення прибирання, що дозволяє клієнту відстежувати хід виконання роботи. У розділі також передбачено можливість додавання фотографій, які можуть містити додаткову інформацію для майстрів перед виконанням робіт.

Для керування персональними даними призначена сторінка профілю, наведена на рис. 3.9. У цьому розділі користувач може змінити своє ім'я, номер телефону та пароль для входу до системи. Окремо відображається інформація про підключений Telegram-акаунт, який використовується під час авторизації. Після внесення змін оновлені дані зберігаються в обліковому записі користувача та використовуються під час подальшої роботи з вебплатформою.

Рис. 3.9 – Профіль користувача

Створення нового замовлення з особистого кабінету виконується через окремий пункт меню «Створити». При цьому користувач працює з тією самою формою оформлення замовлення, що була описана раніше для гостьового замовлення. Відмінність полягає в тому, що для авторизованого клієнта контактні дані не потрібно вводити повторно, оскільки вони автоматично підтягуються з профілю користувача. Усі інші етапи створення заявки, включаючи вибір послуг, адреси, дати та часу виконання робіт, залишаються без змін.

Рис. 3.10 – Особистий кабінет майстра

Після призначення замовлення майстер отримує доступ до інформації, необхідної для його виконання. Для цього в системі передбачено окремий особистий кабінет, адаптований до потреб працівників (рис. 3.10).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Після авторизації майстер потрапляє до власного кабінету, фрагмент якого наведено на рис. 3.10. На відміну від кабінету клієнта, цей інтерфейс орієнтований насамперед на виконання призначених робіт та містить лише необхідну для цього інформацію. У верхній частині сторінки відображаються швидкі кнопки щодо призначених прибирань: кількість робіт на поточний день, майбутніх замовлень, заявок, що очікують оплати, а також закритих замовлень. Нижче розташовано перелік призначених робіт на сьогодні, із зазначенням часу виконання, адреси, контактних даних клієнта, поточного статусу та суми замовлення. Для кожної заявки передбачено кнопку зміни статусу, яка використовується під час виконання робіт. Окремо в кабінеті відображаються нараховані чайові та архів виконаних замовлень. Це дає змогу майстру переглядати власну історію робіт без переходу до інших розділів системи. Оскільки основна робота майстра виконується безпосередньо на об'єкті замовника, інтерфейс кабінету адаптовано для мобільних пристроїв також.

Деталі замовлення ✕

ЗАМОВЛЕННЯ

Усунення різких запахів, Хімчистка: Матрац (2 сторони)

CL-20260605-173624-5292

9 червня 2026 р., 11:00

В РОБОТІ

АДРЕСА

Житомир, Михайлівська, буд. 5

КЛІЄНТ

Яна

+38096 [REDACTED]

ТРИВАЛІСТЬ

2 год

СУМА

1800 грн

ОПЛАТА

Не обрано

Очікує оплати

ГЕОПЕРЕВІРКА

📍 ГЕОПОЗИЦІЯ НЕ ЗБІГАЄТЬСЯ З АДРЕСОЮ

Фото до та після роботи

Спочатку додайте фото стану до роботи, а перед завершенням - фото результату після прибирання.

📷 Додати фото до
📷 Додати фото після
✓ Завантажити і завершити

Рис. 3.11 – Робота майстра із конкретним замовленням

Після переходу до конкретного замовлення майстер відкриває сторінку його детального перегляду (рис. 3.11). На цій сторінці відображається основна

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

інформація про заявку: обрана послуга, адреса виконання робіт, контактні дані клієнта, тривалість замовлення, сума та стан оплати.

Однією з особливостей системи є механізм геопереверки. Перед початком робіт система порівнює поточне місцезнаходження майстра з адресою замовлення. Якщо координати не збігаються, відображається відповідне повідомлення. У цьому ж розділі майстер завантажує фотографії стану об'єкта до та після виконання робіт. Фото зберігаються разом із замовленням і можуть переглядатися клієнтом та адміністратором після завершення прибирання. Якщо під час виконання робіт виникає потреба у додаткових послугах або збільшенні тривалості прибирання, майстер може сформулювати відповідний запит через окремий блок додаткових робіт. Після погодження клієнтом вартість і тривалість замовлення автоматично перераховуються системою. Після завершення прибирання майстер фіксує спосіб оплати та підтверджує отримання коштів, якщо це готівка. Після оплати замовлення переходить до завершеного стану та переноситься до архіву виконаних робіт.

Окрім функціональних можливостей для клієнтів і майстрів, у системі реалізовано адміністративну панель, яка призначена для керування основними бізнес-процесами. Через неї адміністратор може контролювати замовлення, графіки роботи майстрів, клієнтську базу, перелік послуг, спеціальні пропозиції та відгуки, а також переглядати аналітичну інформацію діяльності компанії.

Після авторизації адміністратор отримує доступ до адміністративної панелі, головну сторінку якої наведено на рис. 3.12. У цьому розділі відображаються основні показники роботи компанії: дохід, кількість замовлень, нараховані чайові, замовлення, що потребують оплати, а також статистика активних, завершених і скасованих заявок. Нижче розміщено блоки аналітики, які відображають динаміку замовлень і доходу за обраний період, розподіл оплат за способами оплати, популярність послуг та поточну структуру статусів замовлень. Окремо система формує перелік задач, що потребують уваги адміністратора, наприклад ручне призначення майстрів або інших дій, які потребують додаткового контролю.

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

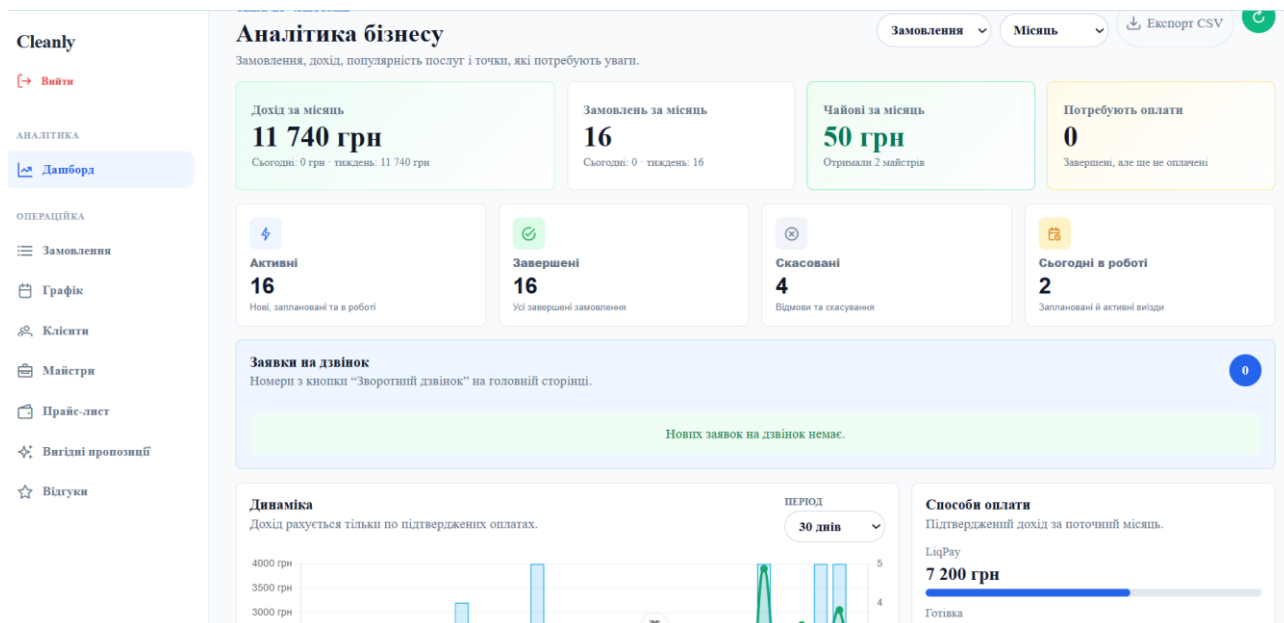


Рис. 3.12 – Головна сторінка адміністративної панелі

Через модуль керування замовленнями (рис. 3.13), адміністратор може переглядати всі заявки, використовувати пошук і фільтрацію, контролювати статуси замовлень, інформацію про оплату та місце виконання робіт. Також передбачено створення нових замовлень, якщо замовник телефонує за номером телефону. У цьому розділі адміністратор може переглядати всі замовлення, створені в системі, незалежно від їх поточного статусу. Для забезпечення комфорту взаємодії, заявки згруповано за категоріями, зокрема активні, в роботі, завершені, оплачені та скасовані. Також передбачено швидкі фільтри для відображення замовлень, які потребують уваги, заплановані на поточний день або майбутні. Кожен запис містить основну інформацію про замовлення: унікальний номер, дату та час виконання, дані клієнта, перелік послуг, адресу, статус виконання, стан геопереверки, інформацію про оплату та підсумкову вартість. Для пошуку потрібної заявки є поле пошуку та можливість фільтрації. Через цей модуль адміністратор також може створювати нові замовлення від імені клієнтів та отримувати доступ до детальної інформації про кожну заявку. Така організація інтерфейсу спрощує оперативний контроль за станом виконання замовлень. Крім того, вона дає змогу швидше знаходити заявки, що потребують втручання адміністратора.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

Cleanly

Вийти

АНАЛІТИКА

Дашборд

ОПЕРАЦІЙКА

Замовлення

Графік

Клієнти

Майстри

Прайс-лист

Вигідні пропозиції

Віагуки

Керування замовленнями

Перегляд та редагування запитів

Житомир

Пошук: ID, клієнт, телефон, адреса або послуга

Створити замовлення

Усі 36

Активні 16

Нові 15

В роботі 1

Завершені 16

Завершені без оплати 0

Оплачені 16

Скасовані 4

16 замовл. у вибраному перегляді

Потребують уваги 0

На сьогодні 2

Майбутні 15

Дата візиту

дд.мм.rrrr

Сортувати

Найближчий візит

ID	ДАТА / ЧАС	КЛІЄНТ	ПОСЛУГА	АДРЕСА	СТАТУС	ГЕО	ПЕРЕНЕСЕННЯ	ОПЛАТА	TARIF*	СУМА
CL-20260607-193633-...	09.06 2026 - 12:00	Яна +3809...	Генеральне прибирання: 2-кімнатна	вул. Лева Качинського, буд. 5	В роботі	Збіжність	Немає	Не обрано - Облікує оп	Зачекавай	3500 ₪
CL-20260609-215951-...	10.06 2026 - 09:00	Валерія +3809...	Прибрати лоток для тварин	вул. Михайлівська, буд. 67 ка.офіс 23	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	500 ₪
CL-20260609-211610-...	10.06 2026 - 10:00	Наталія +3809...	Хімчистка: Звичайний диван	вул. Лева Качинського, буд. 2 ка.офіс 34	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	800 ₪
CL-20260608-095821-...	12.06 2026 - 14:00	Наталія +3809...	Пітримуюче: 2-кімнатна квартира, Хімчистка...	вул. Лева Качинського, буд. 1	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	2992 ₪
CL-20260520-105442-...	21.06 2026 - 09:00	Владислав +3809...	Генеральне прибирання: 1-кімнатна	вул. Лева Качинського, буд. 5 ка.офіс 33 - під. 3 - пом. 1	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	11.5 поточково го	3563 ₪
CL-20260524-143102-...	25.06 2026 - 11:00	Яна +3809...	Очищення пральної машини, Миття...	вул. Лева Качинського, буд. 5	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	1282 ₪
CL-20260608-071526-...	26.06 2026 - 13:00	Іван +3809...	Пітримуюче: 1-кімнатна квартира	вул. Вулиця Лева Качинського, буд. 1 ка.офіс 49 - під. 3 - пом. 4	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	1200 ₪
CL-20260608-232210-...	29.06 2026 - 17:00	Володимир +3809...	Хімчистка: Звичайний диван	вул. Михайлівська, буд. 12 ка.офіс 20	Зачекавай	Облікується	Немає	Не обрано - Облікує оп	Зачекавай	800 ₪

Рис. 3.13 – Управління замовленнями

У розділі графіка майстрів адміністратор обирає майстра та дату, для якої потрібно налаштувати графік роботи. Робочий день поділено на окремі часові проміжки, для кожного з яких можна визначити статус: вільний, робочий або вже заброньований. Для спрощення налаштування передбачено використання тижневого шаблону. Система також відображає кількість обраних годин, кількість заброньованих проміжків та фактичний робочий інтервал майстра на вибрану дату. Інформація з цього модуля використовується під час створення замовлень для автоматичного визначення доступних часових слотів і підбору майстрів, які можуть виконати роботи у зазначений час.

Графік майстрів

Налаштуйте робочі години майстра на конкретний день або змініть його типовий тижневий шаблон.

Шаблон тижня

Зберегти день

1

Дата

серед, 10 червня 2026 р.

2

Майстер

4 активні у Житомирі

В Володимир +3809... Активний

Я Яна +3809... Активний

Яна

Серед, 10 Червня 2026 Р.

Для цієї дати вже є окремо збережений графік.

Обрано годин

2

Заброньовано

2

Робочий проміжок:

09:00-13:00

☐ Вільний

☒ Буде робочим

☐ Заброньовано

☒ За шаблоном

☒ Обрати все

☒ Очистити

Ранок

4

08:00

09:00 ✓

10:00 ✓

11:00 ⏰

День

5

12:00 ⏰

13:00

14:00

15:00

16:00

Вечір

4

Рис. 3.14 – Графік майстрів

На рис. 3.15 відображається перелік усіх зареєстрованих користувачів системи із зазначенням контактних даних, кількості створених замовлень, загальної суми витрат та поточного статусу облікового запису. Для пошуку потрібного клієнта передбачено поле пошуку за ім'ям, електронною поштою або номером телефону, а також фільтрацію за статусом користувачів. Кожен запис містить основну інформацію про клієнта та дозволяє перейти до перегляду його історії замовлень. За потреби адміністратор може швидко отримати інформацію про активність користувача та його взаємодію із сервісом. Це спрощує обробку звернень клієнтів і дозволяє оперативно знаходити потрібні замовлення під час вирішення організаційних питань.

Клієнти					
Управління базою користувачів та фільтрація					
Пошук за іменем, email або телефоном...			Всі клієнти ▾		
КЛІЄНТ	КОНТАКТИ	ЗАМОВЛЕНЬ	СУМА (₴)	СТАТУС	ДІЇ
Без імені ID: f626a7f4-87cd-47ad-9f4e-3cc997692c17	phone-380380-...@cleanly.local +380380-...	0	0 ₴	Заблокований	⚙
Володимир ID: 59bd93dd-e278-4763-9d78-98abfd51fbc5	guest-38098-...@cleanly.local +38098-...	1	800 ₴	Активний	⚙
Світлана ID: 3bf215f9-c606-4126-bcf8-dbf5cc5bd67	guest-38073-...@cleanly.local +3807345-...	1	1200 ₴	Активний	⚙

Рис. 3.15 – Управління клієнтами

У розділі «Команда майстрів» (рис. 3.16) адміністратор може переглядати перелік майстрів, які працюють у системі, а також виконувати пошук за ім'ям, номером телефону або містом. Для кожного працівника відображаються основні дані, зокрема ім'я, контактний номер, місто роботи, наявність доступу до особистого кабінету та поточний статус активності. Через цей модуль адміністратор може додавати нових працівників, редагувати їхні дані, надавати або обмежувати доступ до системи, а також деактивувати облікові записи майстрів у разі потреби. Це дозволяє централізовано керувати персоналом та підтримувати актуальність інформації про працівників.

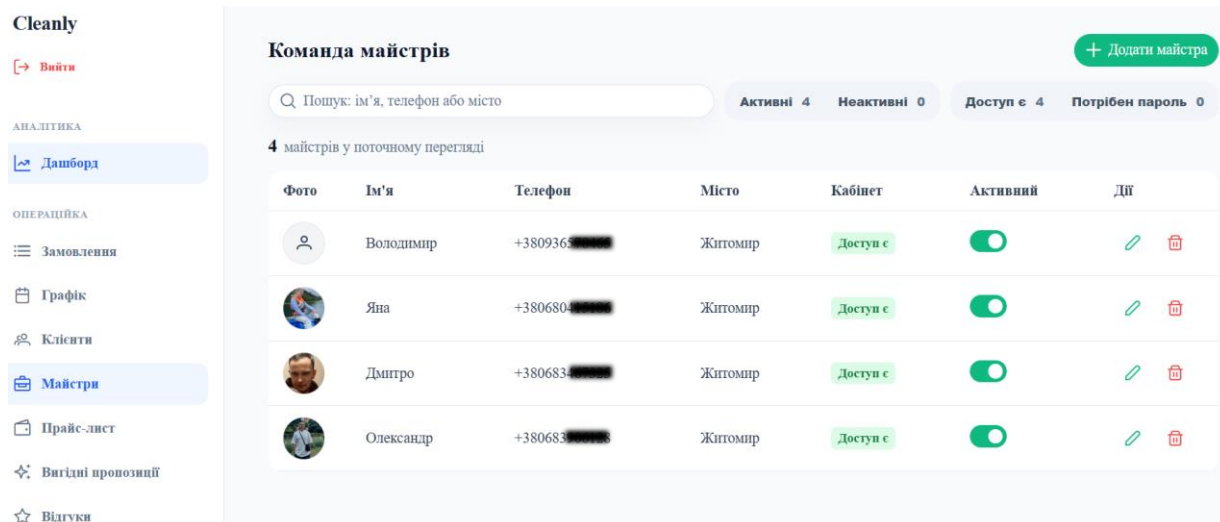


Рис. 3.16 – Управління майстрами

Налаштування переліку послуг виконується через модуль керування послугами, наведений на рис. 3.17. У цьому розділі адміністратор керує прайс-листом вебплатформи. Для кожної послуги зберігаються її назва, категорія, вартість, орієнтовна тривалість виконання та поточний статус. Також для послуг можуть завантажуватися зображення, які використовуються під час відображення на сайті. Модуль підтримує пошук, фільтрацію за категоріями та відображення лише активних або прихованих послуг. Адміністратор може створювати нові послуги, редагувати наявні, приховувати їх від клієнтів або повністю видаляти з системи.

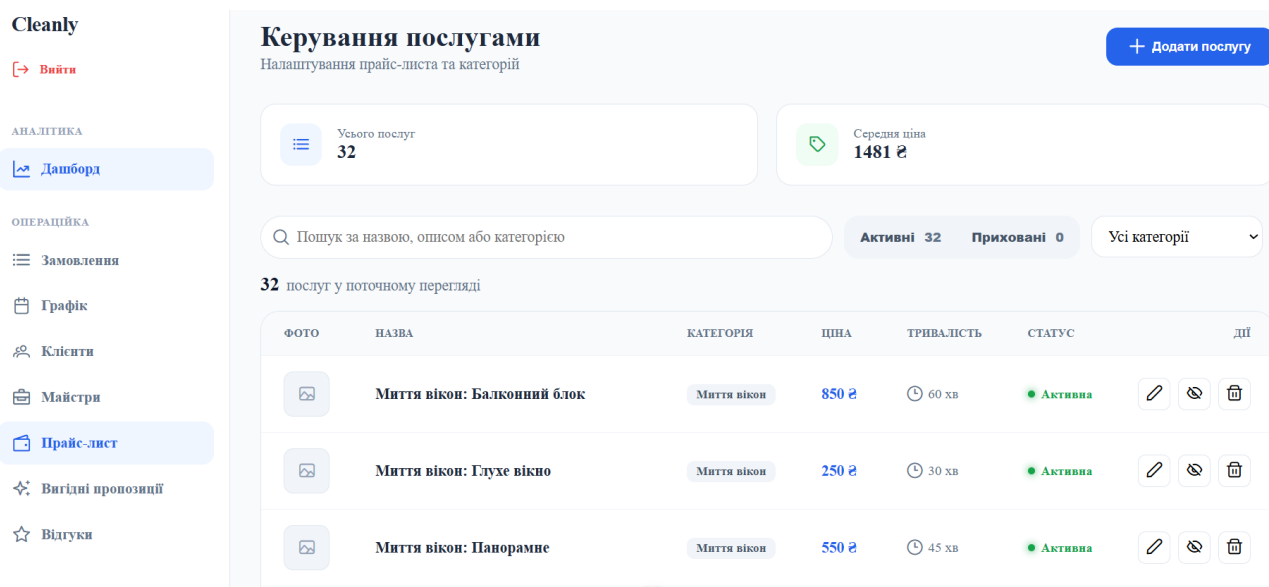


Рис. 3.17 – Управління послугами

Модуль управління вигідними пропозиціями (рис. 3.18) надає можливість створювати та налаштовувати спеціальні пропозиції, які відображаються користувачам під час оформлення замовлення. Для кожної пропозиції задаються умови спрацювання, рекомендована послуга, розмір знижки, пріоритет відображення та поточний статус активності. Система підтримує як рекомендації додаткових послуг, так і автоматичне застосування знижок під час виконання визначених умов. Наприклад, під час замовлення генерального прибирання клієнту може бути запропонована хімічстка меблів зі спеціальною ціною. Це дозволяє автоматизувати механізм додаткових продажів та гнучко керувати маркетинговими пропозиціями без внесення змін до програмного коду.

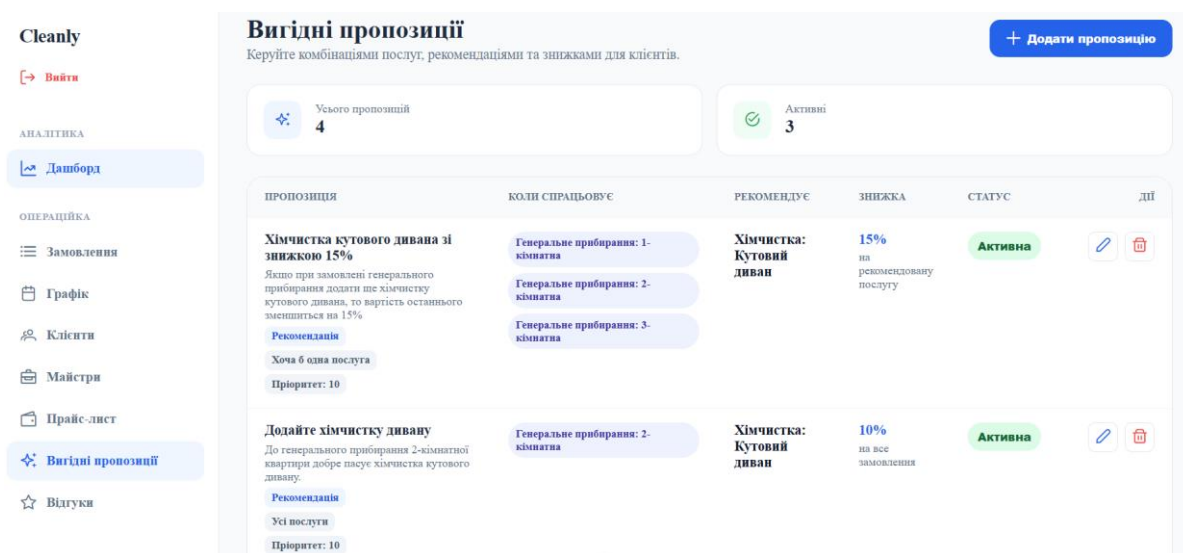


Рис. 3.18 – Управління вигідними пропозиціями

Після завершення замовлення клієнти можуть залишати відгуки про якість виконаних робіт. Для їх перегляду та модерації в адміністративній панелі передбачено окремий розділ, наведений на рис. 3.19. Адміністратор може переглядати всі відгуки, залишені користувачами. Для кожного запису відображаються ім'я клієнта, оцінка, текст відгуку, дата створення та статус публікації. Передбачено можливість пошуку відгуків, перегляду лише опублікованих або прихованих записів, а також зміни їхнього статусу. Це дозволяє контролювати інформацію, яка відображається на головній сторінці вебплатформи, та підтримувати актуальність відгуків про надані послуги. Крім

того, адміністратор може оперативно реагувати на негативні відгуки та контролювати якість надання послуг на основі зворотного зв'язку від клієнтів.

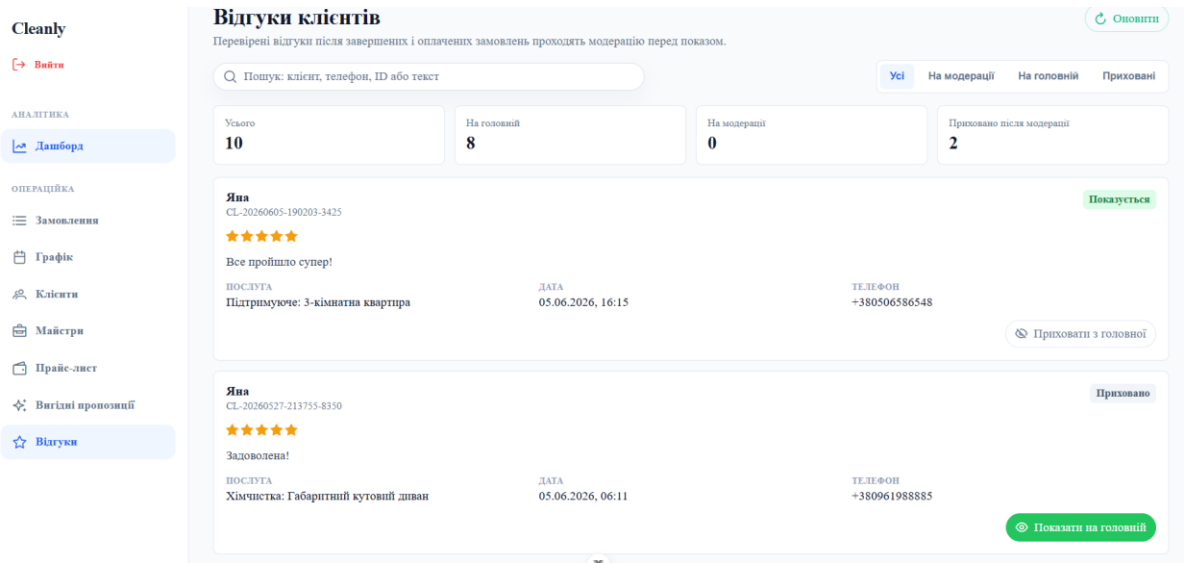


Рис. 3.19 – Модерація відгуків клієнтів

3.3 Тестування роботи програмного продукту

Тестування роботи вебплатформи проводилося з метою перевірки коректності реалізації основних функцій системи, стабільності її роботи та відповідності очікуваній поведінці під час взаємодії користувачів із різними модулями програмного продукту. Основну увагу було зосереджено на перевірці сценаріїв, які є критично важливими для повсякденного використання системи, зокрема авторизації, оформлення замовлення, перевірки введених даних, відображення замовлень у кабінетах користувачів, зміни статусів та роботи адміністративної частини, а також коректність оплати онлайн.

У межах тестування застосовувалась переважно ручна перевірка функціональних можливостей. Для цього послідовно виконувалися типові дії клієнта, майстра та адміністратора, а результати порівнювалися з очікуваною поведінкою системи. Це дає змогу перевірити правильність окремих операцій та загальну логіку взаємодії між інтерфейсом, серверною частиною та базою даних. Особливу увагу було приділено перевірці процесів створення замовлень, авторизації користувачів, призначення майстрів, обробки платежів та зміни статусів замовлень на різних етапах їх виконання. Нижче наведено результати тестування основних функціональних блоків системи у таблицях 3.1–3.10.

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

Спочатку було перевірено можливість входу користувача до системи під клієнтським обліковим записом. Результати тесту авторизації клієнта, під час якого перевірялося коректне введення номера телефону та пароля з подальшим перенаправленням до особистого кабінету, наведено в табл. 3.1.

Таблиця 3.1

Тест «Авторизація клієнта»

Кроки для перевірки	Увести номер телефону і пароль на сторінці входу та натиснути кнопку входу
Вхідні дані	Зареєстрований обліковий запис клієнта, коректний пароль
Очікуваний результат	Користувача авторизовано та перенаправлено до особистого кабінету
Фактичний результат	Відповідає

Після цього було протестовано створення замовлення за умови правильного заповнення всіх необхідних полів. У табл. 3.2 подано результати перевірки сценарію, у якому користувач обирає послугу, вказує адресу, дату, час і, за потреби, додає коментар або фотографії, після чого система формує нове замовлення та відображає сторінку підтвердження з унікальним номером заявки.

Таблиця 3.2

Тест «Створення замовлення з коректними даними»

Кроки для перевірки	Обрати послугу, вказати адресу, дату, час, за потреби додати коментар чи фото і натиснути кнопку створення замовлення
Вхідні дані	Коректно заповнені поля форми замовлення
Очікуваний результат	Замовлення створено, системою сформовано унікальний номер замовлення та відображено сторінку підтвердження
Фактичний результат	Відповідає

Окремо перевірено реакцію системи на порушення вимог до заповнення форми замовлення. У табл. 3.3 наведено результати тесту, що стосується обов'язкових полів: система не дозволяє створити заявку, якщо частина потрібних даних відсутня, позначає проблемні місця та переводить користувача до поля, яке потребує заповнення. Така поведінка дає змогу запобігти збереженню неповних замовлень і забезпечує коректність даних, що надходять до подальшої обробки.

Таблиця 3.3

Тест «Перевірка обов'язкових полів форми замовлення»

Кроки для перевірки	Залишити одне або кілька обов'язкових полів порожніми та спробувати створити замовлення
Вхідні дані	Частково заповнена форма замовлення без обов'язкових полів
Очікуваний результат	Система не створює замовлення, підсвічує незаповнені поля, та переводить користувача до проблемного поля
Фактичний результат	Відповідає

Наступним етапом було перевірено правильність обробки адресних даних. Результати тесту, пов'язаного з введенням некоректної назви вулиці та блокуванням подальшого оформлення замовлення через неможливість визначення координат адреси, подано в табл. 3.4.

Таблиця 3.4

Тест «Перевірка коректності адреси»

Кроки для перевірки	Увести в поле адреси вулицю, якої не існує в Житомирі, та спробувати продовжити оформлення замовлення
Вхідні дані	Некоректна назва вулиці
Очікуваний результат	Система повідомляє про помилку та не дозволяє продовжити створення замовлення, оскільки неможливо визначити координати адреси
Фактичний результат	Відповідає

У табл. 3.5 наведено результати тесту, який підтверджує, що нове замовлення з'являється в кабінеті із зазначенням статусу, дати, часу, адреси та можливістю перегляду детальної інформації.

Таблиця 3.5

Тест «Відображення замовлення в кабінеті клієнта»

Кроки для перевірки	Створити замовлення та перейти до особистого кабінету клієнта
Вхідні дані	Успішно створене замовлення
Очікуваний результат	У кабінеті відображається нове замовлення зі статусом, датою, часом, адресою та можливістю перегляду деталей
Фактичний результат	Відповідає

У табл. 3.6 подано результати перевірки, які підтверджують, що майстер після входу до системи бачить замовлення, закріплені саме за ним, разом з адресою, часом виконання та поточним статусом.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.6

Тест «Відображення замовлення в кабінеті майстра»

Кроки для перевірки	Увійти в систему під обліковим записом майстра та відкрити список призначених замовлень
Вхідні дані	Замовлення, призначене конкретному майстру
Очікуваний результат	У кабінеті майстра відображається призначене замовлення з адресою, часом і поточним статусом
Фактичний результат	Відповідає

Перевірка зміни статусу замовлення безпосередньо майстром у процесі роботи показала, чи система коректно обробляє послідовне оновлення статусу активної заявки та зберігає внесені зміни. Результати цього тесту наведено в табл. 3.7.

Таблиця 3.7

Тест «Зміна статусу замовлення майстром»

Кроки для перевірки	Відкрити замовлення в кабінеті майстра та послідовно змінити його статус
Вхідні дані	Активне замовлення, доступне майстру
Очікуваний результат	Статус замовлення успішно оновлюється, а зміни відображаються в системі коректно
Фактичний результат	Відповідає

Окремо перевірено функції адміністратора, пов'язані з редагуванням графіка роботи майстрів. У табл. 3.8 подано результати тесту, який підтверджує, що після зміни доступних годин для вибраної дати новий графік успішно зберігається, а вже заброньовані часові слоти не втрачаються.

Таблиця 3.8

Тест «Редагування графіку майстра адміністратором»

Кроки для перевірки	Увійти під адміністратором, відкрити сторінку графіка майстрів, змінити доступні години для вибраної дати та зберегти зміни
Вхідні дані	Існуючий обліковий запис адміністратора, вибраний майстер і дата
Очікуваний результат	Новий графік збережено, вільні години оновлено, а вже заброньовані слоти не втрачено
Фактичний результат	Відповідає

Наступний тест стосувався вибору способу оплати для завершеного замовлення. Результати цієї перевірки наведено в табл. 3.9, де зафіксовано, що система коректно обробляє вибір способу оплати та спрямовує подальше виконання замовлення за відповідним сценарієм.

Таблиця 3.9

Тест «Вибір способу оплати замовлення»

Кроки для перевірки	Відкрити завершене замовлення та обрати спосіб оплати
Вхідні дані	Завершене замовлення, доступне для оплати
Очікуваний результат	Система коректно обробляє вибір способу оплати та переводить замовлення до відповідного сценарію оплати
Фактичний результат	Відповідає

Завершальним етапом стало тестування онлайн-оплати через LiqPay. У табл. 3.10 подано результати перевірки, у межах якого користувач переходить до платіжного сервісу, виконує оплату, після чого система отримує результат операції та оновлює статус оплати замовлення.

Таблиця 3.10

Тест «Успішна онлайн-оплата замовлення через LiqPay»

Кроки для перевірки	Відкрити замовлення, обрати оплату через LiqPay, перейти на сторінку платіжного сервісу та завершити оплату
Вхідні дані	Завершене замовлення, доступне для онлайн-оплати, коректні платіжні дані
Очікуваний результат	Користувач успішно виконує оплату, платіжний сервіс повертає результат операції, а система оновлює статус оплати замовлення на оплачено
Фактичний результат	Відповідає

У результаті проведеного тестування було підтверджено коректну роботу основних функціональних можливостей вебплатформи. Перевірка показала, що система правильно обробляє авторизацію користувачів, створення замовлень, валідацію даних, відображення заявок у кабінетах клієнта та майстра, зміну статусів, дії адміністратора під час керування графіком, а також вибір і успішне виконання онлайн-оплати через LiqPay. Отримані результати свідчать про те, що реалізований програмний продукт відповідає основним функціональним

вимогам і може використовуватися для автоматизації процесів оформлення, планування та контролю клінінгових замовлень.

Висновки до третього розділу

У третьому розділі було розглянуто практичні аспекти роботи з вебплатформою, починаючи з порядку її встановлення та налаштування, і завершуючи перевіркою основних функціональних можливостей у процесі тестування. У межах розділу описано структуру розгортання системи, до якої входять клієнтська частина, сервер застосунку, база даних Neon PostgreSQL, сховище завантажених фото та зовнішній платіжний сервіс LiqPay. Така побудова забезпечує розподіл функцій між окремими компонентами системи та дозволяє організувати стабільну взаємодію між інтерфейсом, серверною логікою, збереженням даних і обробкою онлайн-оплати.

Окрему увагу було приділено структурі інтерфейсу вебплатформи для різних категорій користувачів. Було показано, що система містить публічну головну сторінку, форму створення замовлення, особистий кабінет клієнта, кабінет майстра та адміністративну панель, причому кожна з цих частин виконує власну роль у межах загального процесу роботи клінінгової компанії. Завдяки такому поділу користувач отримує доступ лише до тих функцій, які відповідають його ролі, а сам інтерфейс послідовно супроводжує його від початкових дій до завершення потрібної операції. Платформа орієнтована на підтримку повного циклу роботи із замовленням: від вибору послуг до перегляду статусу, деталей виконання та оплати.

У межах тестування було перевірено основні сценарії взаємодії з вебплатформою, зокрема авторизацію користувачів, створення замовлення, валідацію полів форми, перевірку адреси, відображення замовлень у кабінетах клієнта та майстра, зміну статусів, редагування графіка майстрів адміністратором, а також вибір і успішне виконання онлайн-оплати. Проведені перевірки підтвердили, що реалізовані функції працюють узгоджено, а система коректно обробляє дані на різних етапах взаємодії користувача з програмним продуктом.

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

Важливим результатом третього розділу є також підтвердження того, що реалізована вебплатформа забезпечує цілісну підтримку взаємопов'язаних процесів, які в реальних умовах роботи клінінгової компанії не можуть існувати відокремлено один від одного. Створення замовлення, його відображення в особистих кабінетах, призначення виконавця, коригування графіка роботи, контроль статусів та обробка оплати функціонують як частини єдиного інформаційного середовища, у межах якого кожна дія користувача впливає на подальший стан системи.

Отже, третій розділ показує, що вебплатформа є не лише реалізованою на рівні програмного коду, а й придатною до практичного використання. Описані інтерфейсні рішення, порядок роботи із системою та результати тестування підтверджують, що розроблений програмний продукт відповідає поставленим вимогам і може бути використаний для автоматизації, планування та контролю клінінгових замовлень.

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

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено вебплатформу для автоматизації основних процесів клінінгової компанії. Створений програмний комплекс охоплює повний цикл роботи із замовленням: від вибору послуг і формування заявки клієнтом до призначення майстрів, контролю виконання робіт, оплати та аналізу результатів діяльності компанії.

У межах роботи було проаналізовано предметну область, досліджено наявні програмні аналоги та визначено їхні переваги й недоліки. На основі отриманих результатів сформовано функціональні та нефункціональні вимоги до вебплатформи, визначено ролі користувачів, спроектовано архітектуру системи, структуру бази даних і логіку взаємодії між її основними компонентами.

Розроблена вебплатформа забезпечує реєстрацію та авторизацію користувачів, розмежування прав доступу між клієнтом, майстром і адміністратором, оформлення гостьових і звичайних замовлень, автоматичний розрахунок вартості та тривалості робіт, визначення необхідної кількості майстрів і перевірку доступності часових слотів. Реалізовано автоматичне призначення майстрів з урахуванням їхньої зайнятості, можливість ручного коригування призначень адміністратором, керування тижневими й денними графіками роботи, підтримку регулярних замовлень і формування спеціальних пропозицій.

Для контролю виконання замовлень передбачено послідовну зміну статусів, перевірку місцезнаходження майстра, завантаження фотографій до та після виконання робіт, а також відображення актуальної інформації в кабінетах клієнта, майстра й адміністратора. Платформа підтримує оплату готівкою та онлайн через сервіс LiqPay. Для адміністратора реалізовано засоби керування замовленнями, користувачами, майстрами, послугами, спеціальними пропозиціями, відгуками та робочими графіками, а також інструменти формування аналітичної звітності.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		74

Вебплатформу побудовано відповідно до клієнт-серверної архітектури. Клієнтську частину реалізовано як односторінковий застосунок на основі Vue 3 із використанням Composition API, PrimeVue, Pinia, Vue Router і Tailwind CSS. Серверну частину розроблено на платформі Node.js із застосуванням Express, а для зберігання даних використано PostgreSQL на хмарній платформі Neon. Захист системи забезпечується JWT-автентифікацією, хешуванням паролів, перевіркою ролей користувачів і збереженням конфіденційних параметрів у змінних середовища.

Під час функціонального тестування було перевірено авторизацію користувачів, створення замовлень, валідацію введених даних та адреси, відображення замовлень у кабінетах клієнта й майстра, зміну статусів, редагування графіків роботи, вибір способу оплати та проведення онлайн-платежу через LiqPay. Результати тестування підтвердили коректну й узгоджену роботу основних функціональних можливостей системи та її відповідність сформованим вимогам.

Практична цінність розробки полягає в можливості використання вебплатформи клінінговими компаніями для централізованого керування замовленнями, персоналом і розкладом робіт. Її впровадження дає змогу скоротити кількість ручних операцій, зменшити ризик помилок під час планування, підвищити прозорість виконання замовлень і покращити взаємодію між клієнтами, майстрами та адміністраторами.

Подальший розвиток вебплатформи може передбачати розширення географії обслуговування, підключення додаткових платіжних і комунікаційних сервісів, надсилання повідомлень через SMS. Реалізована архітектура створює основу для подальшого масштабування системи та розширення її функціональних можливостей.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OWASP. Password Storage Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата звернення: 05.06.2026).
2. Кабанчик. URL: <https://kabanchik.ua/> (дата звернення: 30.04.2026).
3. Чистий Кит. URL: <https://kytt.com.ua/> (дата звернення: 30.04.2026).
4. Клін Сервіс. URL: <https://clean-service-zhytomur.com.ua/> (дата звернення: 30.04.2026).
5. Neon. Introduction. URL: <https://neon.com/docs/introduction> (дата звернення: 30.04.2026).
6. VueJS. Composition API FAQ. URL: <https://vuejs.org/guide/extras/composition-api-faq.html> (дата звернення: 15.05.2026).
7. PrimeVue. URL: <https://primevue.org/> (дата звернення: 10.06.2026).
8. Vue Router. Guide. URL: <https://router.vuejs.org/guide/> (дата звернення: 15.05.2026).
9. Pinia. Core Concepts. URL: <https://pinia.vuejs.org/core-concepts/> (дата звернення: 15.05.2026).
10. Tailwind CSS. Installation. Using Vite. URL: <https://tailwindcss.com/docs/installation/using-vite> (дата звернення: 15.05.2026).
11. Node.js. URL: <https://nodejs.org/> (дата звернення: 15.05.2026).
12. Haverbeke M. Eloquent JavaScript. 4th ed. San Francisco : No Starch Press, 2024. 459 p. URL: <https://eloquentjavascript.net/> (дата звернення: 26.05.2026).
13. Express. URL: <https://expressjs.com/> (дата звернення: 15.05.2026).
14. node-postgres. URL: <https://node-postgres.com/> (дата звернення: 15.05.2026).
15. Chart.js Documentation. URL: <https://www.chartjs.org/docs/latest/> (дата звернення: 23.05.2026).
16. JSON Web Tokens. Introduction. URL: <https://www.jwt.io/introduction> (дата звернення: 24.05.2026).
17. bcrypt. URL: <https://www.npmjs.com/package/bcrypt> (дата звернення: 24.05.2026).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

- 18.LiqPay. Документація. URL: <https://www.liqpay.ua/doc> (дата звернення: 27.05.2026).
- 19.Multer. URL: <https://www.npmjs.com/package/multer> (дата звернення: 27.05.2026).
- 20.Object Management Group. Unified Modeling Language (UML). URL: <https://www.omg.org/uml/> (дата звернення: 29.05.2026).
- 21.PlantUML. Class Diagram. URL: <https://plantuml.com/class-diagram> (дата звернення: 29.05.2026).
- 22.PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 30.05.2026).
- 23.Бацинко М. О. Захист персональних даних в цифровій економіці. Тези Всеукраїнської науково-практичної онлайн конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки. Житомир : Державний університет «Житомирська політехніка», 2021. С. 84. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2021/01/84.pdf> (дата звернення: 01.06.2026).
- 24.MDN Web Docs. Set-Cookie. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie> (дата звернення: 02.06.2026).
- 25.Express. Using middleware. URL: <https://expressjs.com/en/guide/using-middleware.html> (дата звернення: 02.06.2026).
- 26.MDN Web Docs. Security on the Web. URL: <https://developer.mozilla.org/en-US/docs/Web/Security> (дата звернення: 05.06.2026).
- 27.Vite. Environment Variables and Modes. URL: <https://vite.dev/guide/env-and-mode> (дата звернення: 03.06.2026).
- 28.Axios. First Steps. URL: <https://axios-http.com/docs/intro> (дата звернення: 05.06.2026).
- 29.W3C. Geolocation API Specification. URL: <https://www.w3.org/TR/geolocation/> (дата звернення: 04.06.2026).
- 30.Nominatim. Search API. URL: <https://nominatim.org/release-docs/latest/api/Search/> (дата звернення: 04.06.2026)

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг реалізації авторизації та розмежування ролей користувачів

Лістинг А.1 – Формування JWT-токена та встановлення cookie

server/src/routes/auth.js

```

const JWT_SECRET = process.env.JWT_SECRET
if (!JWT_SECRET) {
  console.warn('[auth] JWT_SECRET is not set in .env')
}

const signToken = (user) => {
  return jwt.sign(
    { sub: user.id, email: user.email },
    JWT_SECRET,
    { expiresIn: '7d' },
  )
}

const cookieOptions = {
  httpOnly: true,
  sameSite: 'lax',
  secure: false,
  maxAge: 2 * 24 * 60 * 60 * 1000,
}

// POST /api/auth/login
router.post('/login', async (req, res) => {
  try {
    const identifier = String(req.body?.phone || req.body?.email || '').trim()
    const email = normalizeEmail(identifier)
    const phoneDigits = normalizePhoneDigits(identifier)
    const looksLikeEmail = identifier.includes('@')
    const password = String(req.body?.password || '')

    if (!identifier || !password || (!looksLikeEmail &&
    !isValidUaPhone(phoneDigits))) {
      return res.status(400).json({ ok: false, error: 'VALIDATION_ERROR' })
    }

    const found = await pool.query(
      `SELECT id, email, name, phone, role, created_at, password_hash,
      telegram_chat_id, telegram_username
      FROM users
      WHERE email = $1
        OR ${phoneDigitsSql} = $2
      ORDER BY (email = $1) DESC, created_at ASC
      LIMIT 1`,
      [email, phoneDigits],
    )

    const userRow = found.rows[0]
    if (!userRow || !userRow.password_hash) {
      return res.status(401).json({ ok: false, error: 'INVALID_CREDENTIALS' })
    }
  }
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const ok = await bcrypt.compare(password, userRow.password_hash)
if (!ok) {
  return res.status(401).json({ ok: false, error: 'INVALID_CREDENTIALS' })
}

const user = toPublicUser(userRow)

const token = signToken(user)
res.cookie('token', token, cookieOptions)

return res.json({ ok: true, user })
} catch (e) {
  console.error('[POST /api/auth/login] error:', e)
  return res.status(500).json({ ok: false, error: 'DB_ERROR' })
}
})

```

Лістинг А.2 – Middleware перевірки авторизації користувача

server/src/middlewares/requireAuth.j

```

import jwt from 'jsonwebtoken'
import { pool } from '../db.js'

export const requireAuth = async (req, res, next) => {
  try {
    const token = req.cookies?.token || req.cookies?.access_token

    if (!token) {
      return res.status(401).json({ ok: false, error: 'NO_AUTH' })
    }

    const payload = jwt.verify(token, process.env.JWT_SECRET)

    const userId = payload.sub || payload.id

    const { rows } = await pool.query(
      'SELECT id, email, role, is_blocked FROM users WHERE id = $1',
      [userId]
    )

    const user = rows[0]

    if (!user) {
      res.clearCookie('token')
      res.clearCookie('access_token')
      return res.status(401).json({ ok: false, error: 'USER_NOT_FOUND' })
    }

    if (user.is_blocked) {
      res.clearCookie('token')
      res.clearCookie('access_token')
      return res.status(403).json({
        ok: false,
        error: 'USER_BLOCKED',
        message: 'Ваш акаунт заблоковано адміністратором.'
      })
    }
  }
}

```

					ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

req.user = {
    id: user.id,
    email: user.email,
    role: user.role
}

next()
} catch (err) {
    if (err.name === 'TokenExpiredError' || err.name ===
'JsonWebTokenError') {
        res.clearCookie('token')
        res.clearCookie('access_token')
    }
    console.error('Auth Middleware Error:', err.message)
    return res.status(401).json({ ok: false, error: 'NO_AUTH' })
}
}

```

Лістинг А.3 – Middleware перевірки ролі адміністратора

server/src/middlewares/requireAdmin.js

```

import { pool } from '../db.js'

export const requireAdmin = async (req, res, next) => {
    try {
        const userId = req.user?.id
        if (!userId) return res.status(401).json({ ok: false, error: 'NO_AUTH' })

        const { rows } = await pool.query(`SELECT role FROM users WHERE id = $1
LIMIT 1`, [userId])
        const role = rows?.[0]?.role

        if (role !== 'admin') {
            return res.status(403).json({ ok: false, error: 'FORBIDDEN' })
        }

        next()
    } catch (e) {
        console.error('[requireAdmin] error:', e)
        return res.status(500).json({ ok: false, error: 'DB_ERROR' })
    }
}

```

Лістинг А.4 – Middleware перевірки ролі майстра

server/src/middlewares/requireWorker.js

```

import { pool } from '../db.js'

export const requireWorker = async (req, res, next) => {
    try {
        if (req.user?.role !== 'worker') {
            return res.status(403).json({ ok: false, error: 'WORKER_ONLY' })
        }

        const { rows } = await pool.query(

```

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

```

        `
        SELECT id, is_active
        FROM workers
        WHERE user_id = $1
        LIMIT 1
        `
        ,
        [req.user.id],
    )

    const worker = rows[0]
    if (!worker) return res.status(403).json({ ok: false, error:
'WORKER_PROFILE_NOT_FOUND' })
    if (worker.is_active === false) return res.status(403).json({ ok: false,
error: 'WORKER_INACTIVE' })

    req.worker = {
        id: worker.id,
        is_active: worker.is_active,
    }

    next()
} catch (e) {
    console.error('[requireWorker] error:', e)
    return res.status(500).json({ ok: false, error: 'DB_ERROR' })
}
}

```

Лістинг А.5 – Перевірка ролей у маршрутизаторі клієнтської частини src/router/index.js

```

const homeRouteForRole = (role) => {
    if (role === 'admin') return { name: 'admin-dashboard' }
    if (role === 'worker') return { name: 'worker-orders' }
    return { name: 'account-dashboard' }
}

router.beforeEach(async (to) => {
    const auth = useAuthStore()

    // відновлення сесії після ф5
    if (!auth.loading && auth.user === null && auth.isAuthenticated === false) {
        await auth.hydrate()
    }

    if (!auth.loading && auth.user === null) {
        try {
            await auth.hydrate()
        } catch (e) {
            if (e.status === 403) alert('Ваш акаунт заблоковано адміністратором')
        }
    }

    if (to.meta.requiresAuth && !auth.isAuthenticated) {
        return { name: 'login', query: { redirect: to.fullPath } }
    }

    if (to.meta.guestOnly && auth.isAuthenticated) {
        return homeRouteForRole(auth.user?.role)
    }
}

```

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

```

    if (to.meta.customerOnly && auth.user?.role !== 'customer') {
      return auth.isAuthenticated ? homeRouteForRole(auth.user?.role) : { name:
'landing' }
    }

    if (to.meta.workerOnly && auth.user?.role !== 'worker') {
      return auth.isAuthenticated ? homeRouteForRole(auth.user?.role) : { name:
'landing' }
    }

    // перевірка адміна
    if (to.meta.requiresAdmin) {
      if (auth.user?.role !== 'admin') {
        return auth.isAuthenticated ? homeRouteForRole(auth.user?.role) : { name:
'landing' }
      }

      try {
        const res = await fetch(getApiUrl('/api/admin/ping'), {
          credentials: 'include',
        })
        const data = await res.json().catch(() => ({}))

        if (!res.ok || data.ok !== true) {
          return homeRouteForRole(auth.user?.role)
        }
      } catch {
        return homeRouteForRole(auth.user?.role)
      }
    }
  })
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг реалізації створення замовлення та розрахунку його параметрів

Лістинг Б.1 – Нормалізація вибраних послуг і отримання їхніх параметрів
server/src/routes/orders.js

```
const normalizeServiceIds = (payload) => {
  const values = [
    ...(Array.isArray(payload?.service_ids) ? payload.service_ids : []),
    ...(Array.isArray(payload?.extra_service_ids) ? payload.extra_service_ids : []),
    payload?.service_id,
  ]

  return [...new Set(values.map(Number).filter((id) => Number.isInteger(id) && id > 0))]
}

const resolveServiceMeta = async (db, payload) => {
  const rawDuration = Number(payload?.duration_minutes)
  if (Number.isFinite(rawDuration) && rawDuration > 0) {
    return {
      durationMinutes: Math.ceil(rawDuration),
      servicePrice: toPriceInt(payload?.price),
    }
  }

  const serviceIds = normalizeServiceIds(payload)
  const serviceId = Number(payload?.service_id)
  const type = String(payload?.type || '').trim()

  let result = null
  if (serviceIds.length > 0) {
    result = await db.query(
      `
      SELECT id, name, base_price, duration_minutes
      FROM services
      WHERE id = ANY($1::int[]) AND is_active = true
      ORDER BY array_position($1::int[], id)
      `,
      [serviceIds],
    )
  } else if (Number.isInteger(serviceId) && serviceId > 0) {
    result = await db.query(
      `SELECT id, name, base_price, duration_minutes FROM services WHERE id = $1
      AND is_active = true LIMIT 1`,
      [serviceId],
    )
  } else if (type) {
    result = await db.query(
      `SELECT id, name, base_price, duration_minutes FROM services WHERE name = $1
      AND is_active = true LIMIT 1`,
      [type],
    )
  }

  const services = result?.rows || []
  const service = services[0]
  const totalDuration = services.reduce((sum, row) => sum +
```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

(Number(row.duration_minutes) || 0), 0)
const totalPrice = services.reduce((sum, row) => sum + (Number(row.base_price)
|| 0), 0)
const durationMinutes = Number(service?.duration_minutes)

return {
  service,
  services,
  serviceIds,
  durationMinutes: totalDuration > 0
    ? totalDuration
    : Number.isFinite(durationMinutes) && durationMinutes > 0 ?
durationMinutes : 60,
  servicePrice: services.length > 0 ? toPriceInt(totalPrice) :
toPriceInt(payload?.price),
}
}

```

Лістинг Б.2 – Розрахунок кількості майстрів і тривалості візиту

server/src/utils/orderScheduling.js

```

export const getRequiredWorkers = (durationMinutes, maxWorkers = null) => {
  const minutes = Math.max(1, Number(durationMinutes) || 1);
  const recommendedWorkers = Math.max(1, Math.ceil(minutes / 120));
  const workerLimit = Number(maxWorkers);

  if (!Number.isFinite(workerLimit) || workerLimit < 1) return
recommendedWorkers;
  return Math.max(1, Math.min(recommendedWorkers, Math.floor(workerLimit)));
};

export const roundToSlotMinutes = (minutes) =>
  Math.ceil(Math.max(1, Number(minutes) || 1) / 60) * 60;

export const getVisitDurationMinutes = (workDurationMinutes, workerCount) => {
  const workers = Math.max(1, Number(workerCount) || 1);
  return roundToSlotMinutes(Math.ceil((Number(workDurationMinutes) || 60) /
workers));
};

export const getDurationPlan = (workDurationMinutes, maxWorkers = null) => {
  const safeWorkDuration = Math.max(60, Math.ceil(Number(workDurationMinutes) ||
60));
  const recommendedWorkers = getRequiredWorkers(safeWorkDuration);
  const requiredWorkers = getRequiredWorkers(safeWorkDuration, maxWorkers);
  const visitDurationMinutes = getVisitDurationMinutes(safeWorkDuration,
requiredWorkers);

  return {
    workDurationMinutes: safeWorkDuration,
    visitDurationMinutes,
    requiredWorkers,
    recommendedWorkers,
  };
};

```

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

Лістинг Б.3 – Перевірка кількості активних майстрів

server/src/routes/orders.js

```
const getActiveWorkerLimit = async (db, city) => {
  const cleanCity = String(city || '').trim()
  if (!cleanCity) return null

  const { rows } = await db.query(
    `
    SELECT COUNT(*)::int AS count
    FROM workers
    WHERE is_active = true
      AND lower(city) = lower($1)
    `
    ,
    [cleanCity],
  )

  const count = Number(rows[0]?.count) || 0
  return count > 0 ? count : null
}

const getCityDurationPlan = async (db, workDurationMinutes, city) => {
  const activeWorkerLimit = await getActiveWorkerLimit(db, city)
  return {
    ...getDurationPlan(workDurationMinutes, activeWorkerLimit),
    activeWorkerLimit: activeWorkerLimit || 0,
  }
}
```

Лістинг Б.4 – Створення або пошук гостьового користувача

server/src/routes/orders.js

```
const normalizePhoneDigits = (v) => {
  const digits = String(v || '').replace(/\D/g, '')
  if (digits.length === 10 && digits.startsWith('0')) return `38${digits}`
  if (digits.length === 9) return `380${digits}`
  if (digits.length === 12 && digits.startsWith('380')) return digits
  return digits
}

const formatPhone = (v) => {
  const digits = normalizePhoneDigits(v)
  return digits.length === 12 && digits.startsWith('380') ? `+${digits}` :
  String(v || '').trim()
}

const makeGuestEmail = (phoneDigits) => `guest-${phoneDigits}@cleanly.local`

const {
  guest_name,
  guest_phone
} = req.body || {}

const guestName = String(guest_name || '').trim() || 'Гість'
const guestPhone = String(guest_phone || '').trim()
const phoneDigits = normalizePhoneDigits(guestPhone)
const formattedGuestPhone = formatPhone(guestPhone)
const guestEmail = makeGuestEmail(phoneDigits)

const existingUser = await client.query(
```

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

```

        \
        SELECT id, email, name, phone
        FROM users
        WHERE ${phoneDigitsSql} = $1
           OR email = $2
        ORDER BY created_at ASC
        LIMIT 1
        \
    `,
    [phoneDigits, guestEmail],
)

let user = existingUser.rows[0]

if (!user) {
    const createdUser = await client.query(
        \
        INSERT INTO users (email, name, phone)
        VALUES ($1, $2, $3)
        RETURNING id, email, name, phone
        \
        ,
        [guestEmail, guestName, formattedGuestPhone],
    )

    user = createdUser.rows[0]
} else if (!user.phone || !user.name) {
    const updatedUser = await client.query(
        \
        UPDATE users
        SET
            phone = COALESCE(NULLIF(phone, ''), $2),
            name = COALESCE(NULLIF(name, ''), $3)
        WHERE id = $1
        RETURNING id, email, name, phone
        \
        ,
        [user.id, formattedGuestPhone, guestName],
    )

    user = updatedUser.rows[0]
}

```

Лістинг Б.5 – Основний маршрут створення замовлення

server/src/routes/orders.js

У наведеному фрагменті змінні `orderStatus` та `reservation` формуються на попередньому етапі перевірки доступності майстрів, який детальніше розглянуто в додатку В.

```

const publicId = makePublicId()
const durationPlan = await getCityDurationPlan(client,
serviceMeta.durationMinutes, city)

const discount = await calculateSpecialOfferDiscount(
    client,
    serviceMeta.serviceIds,
    req.body?.applied_offer_ids,
    serviceMeta.servicePrice,
)

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const frequencyDiscount = calculateFrequencyDiscount(
  Math.max(0, serviceMeta.servicePrice - discount),
  frequency,
)

const totalDiscount = Math.min(serviceMeta.servicePrice, discount +
frequencyDiscount)
const safePrice = Math.max(0, serviceMeta.servicePrice - totalDiscount)

const discountDetails = {
  special_offers: discount,
  frequency: frequencyDiscount,
}
const { rows } = await client.query(
  `
  INSERT INTO orders (
    public_id,
    user_id,
    type,
    date,
    time,
    address,
    address_structured,
    customer_note,
    price,
    payment_amount,
    frequency,
    discount_amount,
    discount_details,
    recurrence_group_id,
    recurrence_sequence,
    recurrence_total,
    recurrence_until,
    status,
    worker_id,
    scheduled_time,
    duration_hours,
    work_duration_minutes,
    visit_duration_minutes,
    required_workers
  )
  VALUES
  ($1,$2,$3,$4,$5::text,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19, ($4::date + $5::time),$20,$21,$22,$23)
  RETURNING
    id AS order_uuid,
    public_id AS id,
    type,
    date::text AS date,
    time::text AS time,
    address,
    address_structured,
    customer_note,
    status,
    price,
    payment_method,
    payment_status,
    payment_amount,
    payment_confirmed_at,
    payment_note,
    frequency,
    discount_amount,
    discount_details,

```

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

```

recurrence_group_id,
recurrence_sequence,
recurrence_total,
recurrence_until,
duration_hours,
work_duration_minutes,
visit_duration_minutes,
required_workers,
created_at
),
[
  orderPublicId,
  req.user.id,
  orderType,
  occurrence.date,
  occurrence.time,
  address,
  enrichedAddressStructured,
  finalNote || null,
  safePrice,
  safePrice,
  frequency,
  totalDiscount,
  discountDetails,
  recurrenceGroupId,
  index + 1,
  recurringOccurrences.length,
  recurrenceUntil,
  orderStatus,
  reservation?.workerIds?.[0] ?? null,
  Math.ceil(durationPlan.visitDurationMinutes / 60),
  durationPlan.workDurationMinutes,
  durationPlan.visitDurationMinutes,
  durationPlan.requiredWorkers,
],
)

const order = rows[0]

```

					ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг реалізації бронювання часових слотів майстрів

Лістинг В.1 – Формування потрібних часових слотів для замовлення

server/src/utils/orderScheduling.js

```

const timeToMinutes = (value) => {
  const normalized = normalizeTime(value);
  if (!normalized) return null;

  const [hours, minutes] = normalized.split(":").map(Number);
  return hours * 60 + minutes;
};

const minutesToTime = (minutes) => {
  const hours = Math.floor(minutes / 60);
  const mins = minutes % 60;
  return `${String(hours).padStart(2, "0")}:${String(mins).padStart(2, "0")}:00`;
};

export const getSlotTimesForWork = (startTime, durationMinutes) => {
  const startMinutes = timeToMinutes(startTime);
  if (startMinutes === null) return [];

  const slotCount = Math.max(1, Math.ceil((Number(durationMinutes) || 60) / 60));
  return Array.from(
    { length: slotCount },
    (_, index) => minutesToTime(startMinutes + index * 60),
  );
};

```

Лістинг В.2 – Створення робочих слотів на основі шаблону графіка майстра

server/src/routes/orders.js

```

const ensureDailySlotsFromTemplates = async (db, date, city) => {
  const normalizedDate = normalizeDate(date);
  const cleanCity = String(city || '').trim();
  if (!normalizedDate || !cleanCity) return

  await db.query(
    `
    INSERT INTO work_slots (worker_id, slot_date, slot_time, is_booked)
    SELECT
      w.id,
      $1::date,
      generated.slot_time::time,
      false
    FROM workers w
    JOIN worker_schedules ws ON ws.worker_id = w.id
    CROSS JOIN LATERAL generate_series(
      $1::date + ws.start_time,
      $1::date + ws.end_time - interval '1 hour',
      interval '1 hour'
    ) AS generated
  `
  );
};

```

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

```

) AS generated(slot_time)
WHERE COALESCE(w.is_active, true) = true
  AND w.city = $2
  AND ws.day_of_week = EXTRACT(DOW FROM $1::date)::int
  AND NOT EXISTS (
    SELECT 1
    FROM work_slots existing
    WHERE existing.worker_id = w.id
      AND existing.slot_date = $1::date
  )
ON CONFLICT DO NOTHING
`,
[normalizedDate, cleanCity],
)
}

```

Лістинг В.3 – Перевірка доступного часу початку замовлення server/src/routes/orders.js

```

const getAvailableStarts = async (db, { date, city, durationMinutes,
requiredWorkers, excludeOrderId = null }) => {
  const { rows } = await db.query(
    `
WITH active_workers AS (
  SELECT id
  FROM workers
  WHERE COALESCE(is_active, true) = true
    AND city = $2
),
available_slots AS (
  SELECT
    ws.worker_id,
    ws.slot_time
  FROM work_slots ws
  JOIN active_workers w ON w.id = ws.worker_id
  WHERE ws.slot_date = $1
    AND COALESCE(ws.is_booked, false) = false

  UNION ALL

  SELECT
    w.id AS worker_id,
    generated.slot_time::time AS slot_time
  FROM active_workers w
  JOIN worker_schedules schedule ON schedule.worker_id = w.id
  CROSS JOIN LATERAL generate_series(
    $1::date + schedule.start_time,
    $1::date + schedule.end_time - interval '1 hour',
    interval '1 hour'
  ) AS generated(slot_time)
  WHERE schedule.day_of_week = EXTRACT(DOW FROM $1::date)::int
    AND NOT EXISTS (
      SELECT 1
      FROM work_slots existing
      WHERE existing.worker_id = w.id
        AND existing.slot_date = $1
    )
)
SELECT
  worker_id,

```

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

```

        slot_time::text AS slot_time
    FROM available_slots
    ORDER BY slot_time, worker_id
    ,
    [date, city],
)

const ownSlots = await getOrderOwnSlots(db, { publicId: excludeOrderId, date
})
const blockedTimes = await getBlockedTimes(db, date)
const slotsByWorker = new Map()
const candidateStarts = new Set()

for (const row of rows) {
    const time = normalizeTime(row.slot_time)
    if (!time || blockedTimes.has(time)) continue

    if (!slotsByWorker.has(row.worker_id)) slotsByWorker.set(row.worker_id, new
Set())
    slotsByWorker.get(row.worker_id).add(time)
    candidateStarts.add(time)
}

for (const row of ownSlots) {
    const time = normalizeTime(row.slot_time)
    if (!time || blockedTimes.has(time)) continue

    if (!slotsByWorker.has(row.worker_id)) slotsByWorker.set(row.worker_id, new
Set())
    slotsByWorker.get(row.worker_id).add(time)
    candidateStarts.add(time)
}

return [...candidateStarts]
    .sort()
    .filter((startTime) => !isPastVisitTime(date, startTime))
    .filter((startTime) => {
        const neededTimes = getSlotTimesForWork(startTime, durationMinutes)
        if (neededTimes.some((time) => blockedTimes.has(time))) return false

        let availableWorkers = 0
        for (const workerSlots of slotsByWorker.values()) {
            if (neededTimes.every((time) => workerSlots.has(time))) availableWorkers
+= 1
            if (availableWorkers >= requiredWorkers) return true
        }

        return false
    })
    .map(formatTime)
}

```

Лістинг В.4 – Резервування майстрів і блокування слотів

server/src/routes/orders.js

```

const reserveWorkersForOrder = async (client, { date, city, startTime,
durationMinutes, requiredWorkers }) => {
    await ensureDailySlotsFromTemplates(client, date, city)

    const neededTimes = getSlotTimesForWork(startTime, durationMinutes)

```

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

```

if (neededTimes.length === 0) return null

const blockedTimes = await getBlockedTimes(client, date)
if (neededTimes.some((time) => blockedTimes.has(time))) return null

const { rows } = await client.query(
  `
SELECT
  ws.id,
  ws.worker_id,
  ws.slot_time::text AS slot_time,
  COALESCE(worker_load.assigned_orders, 0)::int AS assigned_orders,
  COALESCE(worker_load.booked_slots, 0)::int AS booked_slots
FROM work_slots ws
JOIN workers w ON w.id = ws.worker_id
LEFT JOIN LATERAL (
  SELECT
    COUNT(DISTINCT o.id) AS assigned_orders,
    (
      SELECT COUNT(*)
      FROM work_slots booked
      WHERE booked.worker_id = w.id
        AND booked.slot_date = $1
        AND COALESCE(booked.is_booked, false) = true
    ) AS booked_slots
  FROM order_workers ow
  JOIN orders o ON o.id = ow.order_id
  WHERE ow.worker_id = w.id
    AND o.date = $1
    AND o.status IN ('new', 'scheduled', 'on_the_way', 'in_progress')
) worker_load ON true
WHERE ws.slot_date = $1
  AND ws.slot_time = ANY($2::time[])
  AND COALESCE(ws.is_booked, false) = false
  AND COALESCE(w.is_active, true) = true
  AND w.city = $3
ORDER BY
  COALESCE(worker_load.assigned_orders, 0),
  COALESCE(worker_load.booked_slots, 0),
  ws.worker_id,
  ws.slot_time
FOR UPDATE SKIP LOCKED
  `,
  [date, neededTimes, city],
)

const slotsByWorker = new Map()

for (const row of rows) {
  const workerId = Number(row.worker_id)
  const time = normalizeTime(row.slot_time)
  if (!time) continue

  if (!slotsByWorker.has(workerId)) slotsByWorker.set(workerId, [])
  slotsByWorker.get(workerId).push({
    id: Number(row.id),
    time,
    assignedOrders: Number(row.assigned_orders || 0),
    bookedSlots: Number(row.booked_slots || 0),
  })
}

const workerCandidates = [...slotsByWorker.entries()]

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

.map(([workerId, slots]) => ({
  workerId,
  slots,
  assignedOrders: Math.min(...slots.map((slot) => slot.assignedOrders)),
  bookedSlots: Math.min(...slots.map((slot) => slot.bookedSlots)),
}))
.sort((a, b) =>
  a.assignedOrders - b.assignedOrders ||
  a.bookedSlots - b.bookedSlots ||
  a.workerId - b.workerId
)

const selectedWorkers = []
const selectedSlotIds = []

for (const candidate of workerCandidates) {
  if (selectedWorkers.length >= requiredWorkers) break

  const workerTimes = new Set(candidate.slots.map((row) => row.time))
  if (!neededTimes.every((time) => workerTimes.has(time))) continue

  selectedWorkers.push(candidate.workerId)
  selectedSlotIds.push(
    ...candidate.slots
      .filter((row) => neededTimes.includes(row.time))
      .map((row) => row.id),
  )
}

if (selectedWorkers.length < requiredWorkers) return null

await client.query(
  'UPDATE work_slots SET is_booked = true WHERE id = ANY($1::int[])',
  [selectedSlotIds],
)

return { workerIds: selectedWorkers, slotIds: selectedSlotIds }
}

```

Лістинг В.5 – Призначення майстрів до замовлення

server/src/routes/orders.js

```

const assignOrderWorkers = async (client, orderId, workerIds = []) => {
  for (let index = 0; index < workerIds.length; index += 1) {
    const workerId = workerIds[index]

    await client.query(
      `
      INSERT INTO order_workers (order_id, worker_id, is_location_responsible)
      VALUES ($1, $2, $3)
      ON CONFLICT (order_id, worker_id)
      DO UPDATE SET is_location_responsible = EXCLUDED.is_location_responsible
      `,
      [orderId, workerId, index === 0],
    )
  }
}

```

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг реалізації графіка майстрів

Лістинг Г.1 – Отримання слотів майстра на вибрану дату

server/src/routes/schedule.js

```

router.get("/slots", requireAuth, requireAdmin, async (req, res) => {
  const { worker_id, date } = req.query;
  const dayOfWeek = parseDayOfWeek(date);

  if (!worker_id || !date || dayOfWeek === null) {
    return res.status(400).json({ ok: false, error: "VALIDATION_ERROR" });
  }

  try {
    const actualSlots = await pool.query(
      "SELECT slot_time, is_booked FROM work_slots WHERE worker_id = $1 AND slot_date = $2 ORDER BY slot_time",
      [worker_id, date]
    );

    if (actualSlots.rowCount > 0) {
      return res.json({ ok: true, slots: actualSlots.rows, source: "database" });
    }

    const allTemplates = await pool.query(
      "SELECT day_of_week, start_time, end_time FROM worker_schedules WHERE worker_id = $1",
      [worker_id]
    );

    const template = allTemplates.rows.find((row) => row.day_of_week === dayOfWeek);

    if (allTemplates.rowCount > 0 && !template) {
      return res.json({ ok: true, slots: [], source: "template" });
    }

    const start = template?.start_time || DEFAULT_START_TIME;
    const end = template?.end_time || DEFAULT_END_TIME;

    return res.json({
      ok: true,
      slots: generateHourlySlots(start, end),
      source: "template"
    });
  } catch (err) {
    console.error("[GET /api/admin/schedule/slots] error:", err);
    return res.status(500).json({ ok: false, error: err.message });
  }
});

```

Лістинг Г.2 – Збереження денного графіка майстра

server/src/routes/schedule.js

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

router.post("/slots", requireAuth, requireAdmin, async (req, res) => {
  const { worker_id, date, times } = req.body || {};

  if (!worker_id || !date || !Array.isArray(times)) {
    return res.status(400).json({ ok: false, error: "VALIDATION_ERROR" });
  }

  const normalizedTimes = [...new
Set(times.map(normalizeTime).filter(Boolean))].sort();
  const client = await pool.connect();

  try {
    await client.query("BEGIN");

    const worker = await client.query("SELECT id FROM workers WHERE id = $1
LIMIT 1", [worker_id]);
    if (!worker.rows[0]) {
      await client.query("ROLLBACK");
      return res.status(404).json({ ok: false, error: "WORKER_NOT_FOUND" });
    }

    await client.query(
      "DELETE FROM work_slots WHERE worker_id = $1 AND slot_date = $2 AND
is_booked = false",
      [worker_id, date]
    );

    for (const time of normalizedTimes) {
      await client.query(
        `
        INSERT INTO work_slots (worker_id, slot_date, slot_time, is_booked)
        VALUES ($1, $2, $3, false)
        ON CONFLICT DO NOTHING
        `,
        [worker_id, date, time]
      );
    }

    const { rows: slots } = await client.query(
      "SELECT slot_time, is_booked FROM work_slots WHERE worker_id = $1 AND
slot_date = $2 ORDER BY slot_time",
      [worker_id, date]
    );

    await client.query("COMMIT");
    return res.json({ ok: true, slots });
  } catch (err) {
    await client.query("ROLLBACK").catch(() => {});
    console.error("[POST /api/admin/schedule/slots] error:", err);
    return res.status(500).json({ ok: false, error: err.message });
  } finally {
    client.release();
  }
});

```

Лістинг Г.3 – Отримання тижневого шаблону роботи майстра

server/src/routes/schedule.js

```

router.get("/worker-template/:worker_id", requireAuth, requireAdmin, async (req,
res) => {

```

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const { worker_id } = req.params;

try {
  const result = await pool.query(
    "SELECT day_of_week, start_time, end_time FROM worker_schedules WHERE
worker_id = $1 ORDER BY day_of_week",
    [worker_id]
  );

  return res.json({ ok: true, template: result.rows });
} catch (err) {
  console.error("[GET /api/admin/schedule/worker-template/:worker_id] error:",
err);
  return res.status(500).json({ ok: false, error: err.message });
}
});

```

Лістинг Г.4 – Збереження тижневого шаблону роботи майстра

server/src/routes/schedule.js

```

router.post("/worker-template", requireAuth, requireAdmin, async (req, res) => {
  const { worker_id, schedules } = req.body || {};

  if (!worker_id || isNaN(worker_id)) {
    return res.status(400).json({ ok: false, error: "INVALID_WORKER_ID" });
  }

  const client = await pool.connect();

  try {
    await client.query("BEGIN");

    const worker = await client.query("SELECT id FROM workers WHERE id = $1
LIMIT 1", [worker_id]);
    if (!worker.rows[0]) {
      await client.query("ROLLBACK");
      return res.status(404).json({ ok: false, error: "WORKER_NOT_FOUND" });
    }

    await client.query("DELETE FROM worker_schedules WHERE worker_id = $1",
[worker_id]);

    if (Array.isArray(schedules) && schedules.length > 0) {
      for (const s of schedules) {
        const startTime = normalizeTime(s.start_time);
        const endTime = normalizeTime(s.end_time);
        const dayOfWeek = Number(s.day_of_week);

        if (!Number.isInteger(dayOfWeek) || dayOfWeek < 0 || dayOfWeek > 6 ||
!startTime || !endTime) {
          await client.query("ROLLBACK");
          return res.status(400).json({ ok: false, error: "VALIDATION_ERROR" });
        }

        await client.query(
          "INSERT INTO worker_schedules (worker_id, day_of_week, start_time,
end_time) VALUES ($1, $2, $3, $4)",
          [worker_id, dayOfWeek, startTime, endTime]
        );
      }
    }
  }
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    await client.query("COMMIT");
    return res.json({ ok: true });
  } catch (err) {
    await client.query("ROLLBACK").catch(() => {});
    console.error("[POST /api/admin/schedule/worker-template] error:", err);
    return res.status(500).json({ ok: false, error: err.message });
  } finally {
    client.release();
  }
});

```

Лістинг Г.5 – Завантаження та збереження графіка в адміністративному інтерфейсі

src/views/admin/ScheduleView.vue

```

const loadExistingSlots = async () => {
  if (!selectedWorker.value?.id || !selectedDate.value) return;

  const dateStr = toLocalISODate(selectedDate.value);
  selectedTimes.value = [];
  bookedTimes.value = [];

  try {
    const res = await fetch(
      `${getApiUrl("/api/admin/schedule/slots")}?worker_id=${selectedWorker.value.id}&date=${dateStr}`,
      { credentials: "include" },
    );
    const data = await res.json();

    if (data.ok) {
      dataSource.value = data.source || "none";
      const allSlots = Array.isArray(data.slots) ? data.slots : [];
      allSlots.forEach((slot) => {
        const time = String(slot.slot_time).slice(0, 5);
        if (slot.is_booked) bookedTimes.value.push(time);
        else selectedTimes.value.push(time);
      });
    }
  } catch (err) {
    console.error(err);
  }
};

const saveSchedule = async () => {
  if (!selectedWorker.value?.id) return;

  isSaving.value = true;
  const dateStr = toLocalISODate(selectedDate.value);

  try {
    const res = await fetch(getApiUrl("/api/admin/schedule/slots"), {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        worker_id: Number(selectedWorker.value.id),
        date: dateStr,

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        times: selectedTimes.value,
    }},
    credentials: "include",
});

const data = await res.json().catch(() => ({}));
if (!res.ok || data.ok !== true) {
    alert(`Помилка: ${data.error || "SAVE_FAILED"}`);
    return;
}

await loadExistingSlots();
alert("Трафік на день збережено.");
} catch (err) {
    console.error(err);
} finally {
    isSaving.value = false;
}
};

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						99
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг реалізації обробки оплати замовлення

Лістинг Д.1 – Формування даних та підпису LiqPay

server/src/utils/liqpay.js

```

import crypto from "node:crypto";

export const LIQPAY_CHECKOUT_URL = "https://www.liqpay.ua/api/3/checkout";
export const LIQPAY_API_REQUEST_URL = "https://www.liqpay.ua/api/request";

export const encodeLiqPayData = (payload) =>
  Buffer.from(JSON.stringify(payload)).toString("base64");

export const signLiqPayData = (privateKey, data) =>
  crypto
    .createHash("sha3-256")
    .update(`${privateKey}${data}${privateKey}`)
    .digest("base64");

export const createLiqPayCheckout = ({ publicKey, privateKey, payload }) => {
  const data = encodeLiqPayData({
    version: 7,
    public_key: publicKey,
    ...payload,
  });

  return {
    checkout_url: LIQPAY_CHECKOUT_URL,
    data,
    signature: signLiqPayData(privateKey, data),
  };
};

export const createLiqPayApiRequest = ({ publicKey, privateKey, payload }) => {
  const data = encodeLiqPayData({
    version: 7,
    public_key: publicKey,
    ...payload,
  });

  return {
    data,
    signature: signLiqPayData(privateKey, data),
  };
};

export const decodeLiqPayData = (data) => {
  try {
    return JSON.parse(Buffer.from(String(data || ""),
      "base64").toString("utf8"));
  } catch {
    return null;
  }
};

export const isValidLiqPaySignature = ({ privateKey, data, signature }) =>
  Boolean(privateKey && data && signature && signLiqPayData(privateKey, data)
    === signature);

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

export const getCleanlyLiqPayOrderId = (publicId, attemptId = Date.now()) =>
  `cleanly-${publicId}-${attemptId}`;

export const getPublicIdFromLiqPayOrderId = (liqPayOrderId) => {
  const match = String(liqPayOrderId || '').match(
    /^cleanly[:-(CL-\d{8}-\d{6}-\d{4})](?:-.+)?$/
  );
  return match?.[1] || null;
};

```

Лістинг Д.2 – Оновлення статусу онлайн-оплати після відповіді LiqPay

server/src/routes/orders.js

```

const syncLiqPayPaymentStatus = async ({ publicId, status }) => {
  if (!publicId || (!isLiqPaySuccessStatus(status) &&
    !isLiqPayFailureStatus(status))) return false

  const paymentStatus = isLiqPaySuccessStatus(status) ? 'paid' : 'failed'
  const { rowCount } = await pool.query(
    `
    UPDATE orders
    SET
      payment_method = 'liqpay',
      payment_status = $2,
      payment_confirmed_at = CASE WHEN $2 = 'paid' THEN now() ELSE NULL END,
      payment_note = $3
    WHERE public_id = $1
      AND payment_method = 'liqpay'
      AND payment_status <> 'paid'
    `,
    [publicId, paymentStatus, `LiqPay status: ${status}`],
  )

  return rowCount > 0
}

```

Лістинг Д.3 – Обробка callback від LiqPay

server/src/routes/orders.js

```

router.post('/liqpay/callback', async (req, res) => {
  try {
    const config = getLiqPayConfig()
    const data = String(req.body?.data || '').trim()
    const signature = String(req.body?.signature || '').trim()

    if (!hasLiqPayKeys(config) || !isValidLiqPaySignature({ privateKey:
      config.privateKey, data, signature })) {
      return res.status(400).send('invalid signature')
    }

    const payload = decodeLiqPayData(data)
    const publicId = getPublicIdFromLiqPayOrderId(payload?.order_id)
    const status = String(payload?.status || '').trim()

    if (!payload || !publicId || (!isLiqPaySuccessStatus(status) &&
      !isLiqPayFailureStatus(status))) {
      return res.send('ok')
    }
  }
}

```

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    await syncLiqPayPaymentStatus({ publicId, status })

    return res.send('ok')
  } catch (e) {
    console.error('[POST /api/orders/liqpay/callback] error:', e)
    return res.status(500).send('callback error')
  }
})

```

Лістинг Д.4 – Створення checkout для онлайн-оплати замовлення

server/src/routes/orders.js

```

router.post('/:id/liqpay/checkout', requireAuth, requireCustomer, async (req,
res) => {
  try {
    const id = String(req.params.id || '').trim()
    const config = getLiqPayConfig()

    if (!id) return res.status(400).json({ ok: false, error: 'VALIDATION_ERROR'
})
    if (!hasLiqPayKeys(config)) return res.status(503).json({ ok: false, error:
'LIQPAY_NOT_CONFIGURED' })

    const { rows } = await pool.query(
      `
      SELECT
        public_id,
        type,
        status,
        price,
        payment_method,
        payment_status,
        COALESCE(payment_amount, price) AS payment_amount
      FROM orders
      WHERE public_id = $1
        AND user_id = $2
      LIMIT 1
      `
      , [id, req.user.id],
    )

    const order = rows[0]
    if (!order) return res.status(404).json({ ok: false, error: 'NOT_FOUND' })
    if (order.status !== 'done' || order.payment_method !== 'liqpay') {
      return res.status(409).json({ ok: false, error:
'LIQPAY_PAYMENT_NOT_AVAILABLE' })
    }
    if (order.payment_status === 'paid') {
      return res.status(409).json({ ok: false, error: 'ORDER_ALREADY_PAID' })
    }

    const liqPayOrderId = getCleanlyLiqPayOrderId(order.public_id)
    const resultUrl =
`${config.resultOrigin}/account/orders/${encodeURIComponent(order.public_id)}?li
qpay=return`
    const payload = {
      action: 'pay',
      amount: Math.max(1, Number(order.payment_amount) || Number(order.price) ||
1),
      currency: 'UAH',

```

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

```

        description: `Cleanly order ${order.public_id}: ${order.type}`,
        order_id: liqPayOrderId,
        result_url: resultUrl,
        language: 'uk',
    }

    if (config.callbackUrl) payload.server_url = config.callbackUrl

    await pool.query(
        `
        UPDATE orders
        SET discount_details = COALESCE(discount_details, '{} '::jsonb)
        || jsonb_build_object(
            'liqpay',
            jsonb_build_object(
                'order_id', $2::text,
                'checkout_created_at', to_jsonb(now()::text)
            )
        )
        WHERE public_id = $1
        AND user_id = $3
        `,
        [order.public_id, liqPayOrderId, req.user.id],
    )

    return res.json({
        ok: true,
        payment: createLiqPayCheckout({
            publicKey: config.publicKey,
            privateKey: config.privateKey,
            payload,
        }),
    })
} catch (e) {
    console.error('[POST /api/orders/:id/liqpay/checkout] error:', e)
    return res.status(500).json({ ok: false, error: 'DB_ERROR' })
}
})

```

Лістинг Д.5 – Підтвердження оплати майстром

server/src/routes/worker.js

```

const normalizePaymentMethod = (method) => {
    const value = String(method || '').trim()
    return ['cash', 'liqpay'].includes(value) ? value : null
}

const normalizePaymentStatus = (status) => {
    const value = String(status || '').trim()
    return ['pending', 'cash_expected', 'paid', 'failed'].includes(value) ? value : null
}

router.patch('/orders/:id/payment', requireAuth, requireWorker, async (req, res) => {
    try {
        const paymentMethod = normalizePaymentMethod(req.body?.payment_method)
        const requestedStatus = normalizePaymentStatus(req.body?.payment_status)
        const paymentNote = String(req.body?.payment_note || '').trim() || null

        if (!paymentMethod) {

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		103

```

        return res.status(400).json({ ok: false, error: 'INVALID_PAYMENT_METHOD'
    })
    }

    const paymentStatus = requestedStatus || (paymentMethod === 'cash' ?
'cash_expected' : 'pending')
    if (paymentMethod === 'liqpay' && paymentStatus === 'cash_expected') {
        return res.status(400).json({ ok: false, error: 'INVALID_PAYMENT_STATUS'
    })
    }

    const { rows } = await pool.query(
        `
        UPDATE orders o
        SET
            payment_method = $3,
            payment_status = $4,
            payment_amount = COALESCE(payment_amount, price),
            payment_confirmed_at = CASE WHEN $4 = 'paid' THEN
COALESCE(payment_confirmed_at, now()) ELSE payment_confirmed_at END,
            payment_note = $5
        FROM order_workers ow
        WHERE ow.order_id = o.id
            AND ow.worker_id = $1
            AND o.public_id = $2
            AND o.status = 'done'
            AND o.payment_status <> 'paid'
        RETURNING
            o.public_id AS id,
            o.type,
            o.date::text AS date,
            o.time::text AS time,
            o.address,
            o.address_structured,
            o.customer_note,
            o.status,
            o.price,
            o.payment_method,
            o.payment_status,
            o.payment_amount,
            o.payment_confirmed_at,
            o.payment_note,
            o.frequency,
            o.work_duration_minutes,
            o.visit_duration_minutes,
            o.required_workers,
            o.created_at,
            ow.is_location_responsible,
            ow.started_at,
            ow.start_lat,
            ow.start_lng,
            ow.start_accuracy_meters,
            ow.start_distance_meters,
            ow.start_location_match,
            ow.start_location_status
        `
        ,
        [req.worker.id, req.params.id, paymentMethod, paymentStatus, paymentNote],
    )

    if (!rows[0]) return res.status(404).json({ ok: false, error:
'ORDER_NOT_FOUND_OR_NOT_EDITABLE' })
    return res.json({ ok: true, order: { ...rows[0], status:
normalizeStatus(rows[0].status) } })

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		104

```
    } catch (e) {
      console.error('[PATCH /api/worker/orders/:id/payment] error:', e)
      return res.status(500).json({ ok: false, error: 'DB_ERROR' })
    }
  })
```