МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»**

**Факультет комп'ютерних наук і технологій**
**Кафедра комп'ютерних систем та мереж**

# Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему «РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ АВТОКЛІНІНГОВОЇ КОМПАНІЇ»

Виконав: студент 4 курсу, групи КНТз-512
спеціальності 123 Комп'ютерна інженерія

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(код і назва спеціальності)

КАНИГІНА В.С.
(прізвище та ініціали)
Керівник ТІМЕНКО А.В.
(прізвище та ініціали)
Рецензент РОМАНОВСЬКИЙ О.В.
(прізвище та ініціали)

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет «Запорізька політехніка»**

( повне найменування вищого навчального закладу )

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і назва)

Освітня програма (спеціалізація) Комп'ютерна інженерія

(назва)

**ЗАТВЕРДЖУЮ**
**Зав. кафедри**
_____ Кудерметов Р.К.
"14" квітня 2026 року

# З А В Д А Н Н Я
## НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

КАНИГІНА Оксана Сергіївна

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка комп'ютерної системи автоклінінгової компанії»

керівник проекту (роботи) ТІМЕНКО А.В., ст. викл.

( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "14" квітня 2026 року № 160

2. Строк подання студентом проекту (роботи) 28.05. 2026 року

3. Вихідні дані до проекту (роботи) Опис діяльності автоклінінгової компанії, програмні засоби PHP, MySQL.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметної області та існуючих рішень

2) Проєктування архітектури комп'ютерної системи

3) Розробка вимог до програмного забезпечення

4) Проєктування бази даних

5) Реалізація системи

6) Інтерфейс користувача

ДЬ5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проекту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | прийняв виконане завдання |
| 1-6 | ТІМЕНКО А.В., ст. викладач | | |
| Нормоконтроль | ДЬЯЧУК Т.С., ст. викладач | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 13.04. 2026 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту (роботи) | Строк виконання етапів проекту ( роботи ) | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 15.04.2026 | |
| 2 | Огляд існуючих програмних рішень | 20.04.2026 | |
| 3 | Архітектура комп'ютерної системи | 27.04.2026 | |
| 4 | Розробка вимог дот комп'ютерної системи | 07.05.2026 | |
| 5 | Аналіз моделі бази даних | 12.05.2026 | |
| 6 | Апаратна частина системи | 17.05.2026 | |
| 7 | Реалізація системи | 22.05.2026 | |
| 9 | Розробка інтерфейсу | 25.05.2025 | |
| 10 | Тестування системи | 28.05.2026 | |
| | | | |

**Студент** _____ **Оксана КАНИГІНА**
( підпис ) (ініціали та прізвище)

**Керівник проекту (роботи)** _____ **Артур ТІМЕНКО**
( підпис ) (ініціали та прізвище)

# РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 92 с., 11 табл., 34 рис, 11 лістингів, 2 додатки, 32 джерела.

АВТОКЛІНІНГ, ВЕБСИСТЕМА, ІНТЕРФЕЙС КОРИСТУВАЧА, ПРОГРАМУВАННЯ, СКБД, MYSQL, PHP

У першому розділі було виконано аналіз предметної області автоклінінгових послуг, визначено специфіку діяльності автомийок і дітейлінг-центрів, а також показано, що це структурований сервісний бізнес з підвищеною відповідальністю та потребою в регламентах, обліку й контролі ризиків. У першому розділі було проаналізовано предметну область та визначено доцільність розробки спеціалізованої комп'ютерної системи.

У другому розділі обґрунтовано апаратну платформу для розгортання вебсистеми автоклінінгової компанії. Запропонована архітектура передбачає використання фізичного сервера з LAMP-середовищем, локальної або змішаної мережевої інфраструктури та механізмів забезпечення надійності: UPS, резервне копіювання та RAID.

В ході роботи над даною частиною роботи було спроєктовано реляційну базу даних для вебсистеми автоклінінгової компанії, визначено сутності, зв'язки ER-моделі, первинні та зовнішні ключі, обмеження цілісності та індекси.

У п'ятому розділі було описано реалізацію серверної частини вебсистеми автоклінінгової компанії. Серверний рівень побудовано з використанням принципів модульності та шаблону, реалізовано маршрутизацію, доступ до даних.

У заключному розділі було описано процес реалізації, тестування та розгортання вебсистеми автоклінінгової компанії на апаратній платформі.

# ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

| | | |
|---|---|---|
| CPU | - | Central processing unit, Центральний процесор |
| GPU | - | Graphics processing unit, Графічна карта |
| GUI | - | Graphical User Interface, Графічний інтерфейс користувача |
| HDD | - | Hard disk drive, Жорсткий диск |
| LAN | - | Local Area Network, Локальна мережа |
| MVC | - | Model–View–Controller, Модель-Представлення-Контролер |
| MySQL | - | My Structured Query Language, Система керування базами даних |
| NTP | - | Network Time Protocol, Мережевий протокол часу |
| RAID | - | Redundant Array of Independent/Inexpensive Disks, Надлишковий масив незалежних дисків |
| RAM | - | Random Access Memory, Оперативна пам’ять |
| SSD | - | Solid-state drive, Твердотільний накопичувач |
| UI | - | User Interface, Інтерфейс користувача |
| UPS | - | Uninterruptible Power Supply, Джерело безперебійного живлення |
| UX | - | User Experience, Користувацький досвід |
| VPN | - | Virtual Private Network, Віртуальна приватна мережа |
| WAN | - | Wide Area Network, Глобальна мережа |

# ВСТУП

Сучасний ринок автосервісних послуг характеризується високим рівнем конкуренції та постійним зростанням вимог клієнтів до якості й швидкості обслуговування. Однією з важливих складових цього ринку є автоклінінгові компанії, які надають послуги з миття, хімчистки та догляду за автомобілями. Для ефективного функціонування таких підприємств необхідне впровадження сучасних комп'ютерних систем, що забезпечують автоматизацію основних бізнес-процесів.

Актуальність даної кваліфікаційної роботи зумовлена потребою автоклінінгових компаній у зменшенні часу обслуговування клієнтів, оптимізації використання ресурсів, веденні обліку замовлень і персоналу, а також підвищенні якості управлінських рішень. Використання комп'ютерної системи дозволяє зменшити кількість помилок, підвищити рівень сервісу та забезпечити конкурентні переваги на ринку.

Метою дипломної роботи є розробка комп'ютерної системи автоклінінгової компанії**,** яка забезпечить автоматизацію основних процесів її діяльності.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область автоклінінгових послуг;
- дослідити існуючі програмні рішення;
- сформулювати вимоги до системи;
- спроєктувати архітектуру та базу даних;
- реалізувати програмну систему;
- провести тестування розробленої системи.

Об'єктом дослідження є процеси управління діяльністю автоклінінгової компанії. Предметом дослідження є методи та засоби розробки комп'ютерних систем автоматизації.

Практичне значення роботи полягає у можливості використання розробленої системи в реальній діяльності автоклінінгових компаній.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

## 1.1 Характеристика діяльності автоклінінгових компаній

Автоклінінгова компанія (автомийка або детейлінг-центр) є сервісним підприємством, що надає послуги з очищення, відновлення та захисту автомобіля, при цьому результат послуги прямо пов’язаний із якістю матеріалів, кваліфікацією персоналу, дотриманням технологічних карт і організацією потоку замовлень. На відміну від багатьох інших побутових послуг, автоклінінг поєднує операційні задачі сервісу з підвищеною відповідальністю за майно клієнта та необхідністю регламентів, обліку й контролю ризиків, оскільки можливі претензії щодо пошкодження ЛФП, втрати речей із салону або невідповідності очікуваному результату. Крім того, діяльність пов’язана з використанням води, хімії, утворенням стоків, а також із вимогами безпеки праці й організації робочого простору, що додатково ускладнює управління підприємством [1].

У практиці ринку виділяють кілька форматів автоклінінгових послуг, які відрізняються глибиною робіт і бізнес-моделлю. Класичні автомийки забезпечують швидке очищення кузова та, інколи, легке прибирання салону, тоді як детейлінг-центри орієнтовані на глибоке очищення кузова й салону, полірування та нанесення захисних покриттів, а також відновлення матеріалів інтер’єру. Для детейлінгу характерний високий середній чек і потреба в попередньому записі, оскільки послуги можуть тривати від десятків хвилин до кількох годин, а інколи й днів. Типовий технологічний ланцюжок детейлінгу включає етапи ретельного очищення, відновлення, полірування та нанесення захисту, де кожен етап впливає на кінцевий вигляд автомобіля й сприйняття якості клієнтом.

Зміст і параметри послуг доцільно формалізувати у вигляді переліку з характеристиками (вартість, тривалість, витратні матеріали), оскільки саме ці параметри визначають навантаження майстрів, графік боксів, собівартість і прибутковість замовлень.

Для подальшого проєктування комп’ютерної системи в роботі приймається узагальнений набір послуг, наведений у (табл. 1.1.).

Таблиця 1.1 – Типові послуги автоклінінгової компанії та операційні параметри

| № з/п | Назва послуги | Короткий зміст | Тривалість, хв. | Основні ресурси |
|---|---|---|---|---|
| 1 | Зовнішнє миття | мийка кузова, коліс, сушка | 20–40 | пост/бокс, хімія, вода |
| 2 | Прибирання салону | пилосос, протирка, скло | 20–40 | пилосос, мікрофібра |
| 3 | Хімчистка салону | глибока чистка тканин/шкіри | 90–240 | екстрактор, хімія |
| 4 | Полірування кузова | відновлення блиску, корекція | 180–480 | полірувальна машина |
| 5 | Захисне покриття | віск/кераміка/антидощ | 60–240 | покриття, лампи |
| 6 | Детейлінг дисків/шин | очищення, чорніння, захист | 30–90 | щітки, хімія |

Наведені позиції відображають типову структуру робіт у дітейлінгу, де ключовими етапами є очищення, відновлення, полірування та нанесення захисних покриттів [2]. На практиці автоклінінгова компанія комбінує послуги у пакети (базовий/стандарт/преміум), але для автоматизації зручніше зберігати первинні послуги як атомарні елементи, а пакет розглядати як шаблон набору позицій замовлення.

## 1.2 Процеси автоклінінгової компанії

Бізнес-процеси автоклінінгової компанії мають характер сервісного потоку: клієнт звертається, узгоджується час та склад послуг, авто приймається в роботу, виконуються операції з контролем статусів, після чого здійснюється видача автомобіля й оплата. Для побудови комп’ютерної системи важливо виділити

контур процесів, що підлягають автоматизації, та визначити інформаційні потоки між ролями (адміністратор/оператор/майстер/керівник) і об’єктами (клієнт, авто, послуга, замовлення, оплата).

На концептуальному рівні діяльність компанії можна подати як контекстну функціональну модель (аналог IDEF0 рівня A-0), де вхідними даними виступають (рис. 1.1):

а) звернення клієнта:

б) дані про авто;

в) керуючі впливи:

1) прайс;

2) регламенти;

3) графік;

г) механізмами :

1) персонал;

2) обладнання;

Вихідні дані:

– виконана послуга;

– фінансовий результат;

– дані для звітів [3].


Рисунок 1.1 – Контекстна модель діяльності автоклінінгової компанії (A-0)

Функціональний потік “обробка замовлення” доцільно деталізувати як послідовність подій, де на кожному кроці формуються або уточнюються дані, потрібні системі. Нижче наведено укрупнену діаграму активностей, що описує типовий процес від заявки до закриття замовлення (рис. 1.2).

Опис типового процесу:

а) ініціація (Swimlane: Клієнт):

1) починається з активності "Початок”:

– “Звернення клієнта”, що відповідає вхідним даним на концептуальному рівні;

2) клієнт використовує “Зовнішній механізм” (наприклад, веб-інтерфейс), щоб “Подати заявку”:

– проходить перевірку “Заявка заповнена? ”, яка спирається на “Керуючий вплив: Регламенти”;

б) обробка (Swimlanes: Клієнт -> Менеджер -> Бухгалтерія):

1) “Внутрішній механізм» (Менеджер) перевіряє заявку. Якщо вона заповнена, клієнт "Надає дані про авто”;

2) менеджер виконує “Керуючий вплив: Перевірити прайс”, щоб "Розрахувати total_amount” (загальну суму);

3) менеджер створює фізичний запис у таблиці ORDERS з відповідним client_id та “Оновлює статус: “Новий”, використовуючи "Внутрішній механізм” (програмне забезпечення);

в) виконання (Swimlane: Склад / Виробництво):

1) “Внутрішній механізм” (Виробництво/Склад) створює деталізацію замовлення в таблиці ORDER_ITEMS;

2) “Внутрішні механізми” (призначають персонал і обладнання) та "Виконують послугу/Підготовлюють товар”, що відображає механізми з концептуального рівня;

г) фінанси (Swimlanes: Бухгалтерія -> Менеджер):

1) бухгалтерія створює фізичні записи в таблиці PAYMENTS (рахунки та платежі);

2) автоматично “Оновлення статусу: “Оплачено” та фіксації "Фінансового результату”, що є одним із виходів на концептуальному рівні;

д) завершення (Swimlanes: Доставка -> Клієнт):

1) “Внутрішній механізм (Доставка) призначає транспорт (з таблиці VEHICLES) та водія, “Оновлює статус: “В дорозі”;

2) відображаються механізми, які використовуються для надання послуги;

3) передача клієнту, отримання підтвердження та перехід до “Кінець: Закрити замовлення”, що є остаточним виходом.

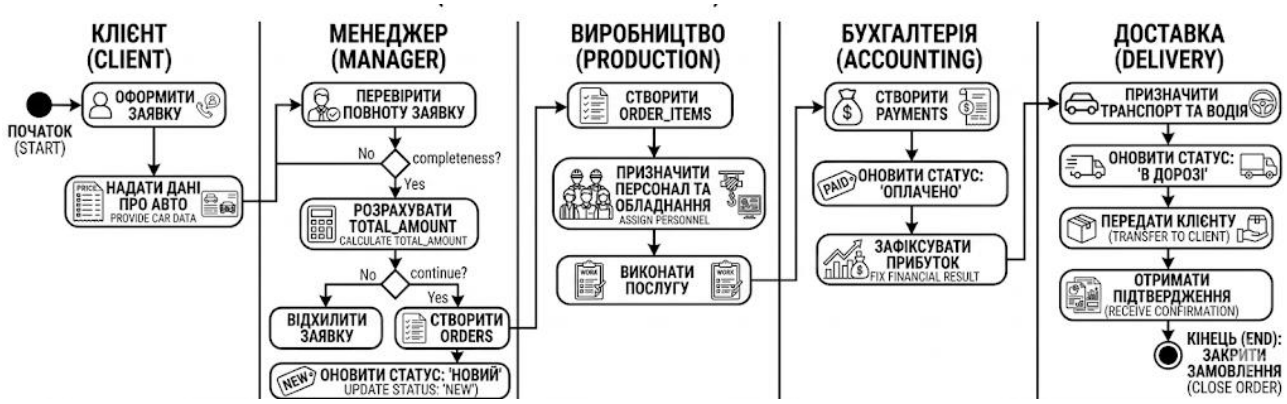


Рисунок 1.2 – Типовий процес обробка замовлення

Оскільки робота є веб-орієнтованою, корисно зафіксувати також взаємодію між ролями та основними функціями системи. Для цього застосовується UML як стандартна графічна мова моделювання, призначена для візуалізації, специфікації та документування систем [4]. На (рис. 1.3) наведено узагальнену діаграму варіантів використання, яка визначає межу системи та основні взаємодії.

На рівні управління персоналом важливо врахувати, що планування відбувається не тільки за часом, але й за ресурсами: боксами, обладнанням, доступністю майстрів та нормами часу.


Рисунок 1.3 – Узагальнена UML діаграма для автоклінінгової системи

Типові рішення на ринку підкреслюють важливість календарного запису, контролю статусів і навантаження майстрів у реальному часі, а також акцентують на онлайн-записі, повідомленнях про статус та історії обслуговування. Це означає, що у моделі бізнес-процесів необхідно передбачити стан замовлення як окрему сутність, а також переходи між станами, оскільки вони є основою оперативного контролю

## 1.3 Аналіз проблем автоматизації

У багатьох невеликих автоклінінгових компаніях облік клієнтів та замовлень часто ведеться у паперових журналах або електронних таблицях, що призводить до втрат інформації, дублювання даних, складності пошуку історії клієнта та неможливості швидко аналізувати показники діяльності. Проблема посилюється в умовах пікового навантаження, коли звернень багато, а оператор змушений одночасно приймати дзвінки, координувати майстрів і оформлювати

замовлення. Типові CRM-підходи для дітейлінгу розглядають централізацію даних клієнтів, керування записом і звітність як ключові чинники конкурентоспроможності, оскільки вони зменшують час обслуговування та підвищують якість сервісу [5].

Однією з системних проблем є “хаос у записах”: накладки в часі, відсутність прозорої картини завантаженості боксів, неузгодженість між оператором і майстрами. Спеціалізовані рішення підкреслюють потребу планувати роботи по годинах, бронювати ресурси та бачити завантаженість у реальному часі. Друга група проблем пов’язана з контрольованістю виконання: клієнти часто очікують прозорості, розуміння “що саме зроблено” і “коли буде готово”, а компанія повинна документувати етапи робіт та результат, зокрема за допомогою фото “до/після”. Третя група проблем стосується економіки: без автоматизованого розрахунку вартості й фіксації позицій замовлення зростає ризик помилок у сумі, некоректних знижок або пропущених додаткових послуг, що прямо впливає на виручку.

Суттєвим фактором є й інформаційна безпека, оскільки веб-система працює з персональними даними клієнтів і фінансовими операціями. OWASP Top 10 підкреслює, що це загальновизнаний перелік найбільш критичних ризиків для вебзастосунків, і його використання є базовим кроком до підвищення безпеки розробки [6]. Тому навіть на етапі аналізу проблем автоматизації необхідно передбачати механізми протидії типовим загрозам, зокрема SQL-ін’єкціям та CSRF, шляхом застосування параметризованих запитів і токенів форм.

Для узагальнення причинно-наслідкових зав’язків між проблемами та їх впливом на бізнес доцільно подати результати у вигляді текстової таблиці (табл. 1.2.), на яку надалі можна посилатись при формуванні вимог.

Таблиця 1.2 – Основні проблеми автоматизації автоклінінгової компанії та наслідки

| № з/п | Проблема | Прояв у роботі | Наслідок |
|---|---|---|---|
| 1 | розрізнений облік клієнтів | контакти в телефоні/ зошиті/таблиці | втрата історії, повторні уточнення |
| 2 | відсутність календарного планування | накладки записів, нерівномірне навантаження | простій або перевантаження боксів |
| 3 | слабкий контроль статусів | неясно, на якому етапі замовлення | конфлікти, падіння лояльності |
| 4 | ручний розрахунок вартості | помилки у сумі, пропуск додаткових послуг | фінансові втрати |
| 5 | відсутність звітності | немає даних про дохід/попит/ефективність | рішення “на інтуїції” |
| 6 | низький рівень безпеки веб-взаємодії | відсутні базові захисти від типових атак | ризик компрометації даних |

## 1.4 Огляд існуючих програмних рішень

Ринок програмних продуктів для сервісних бізнесів пропонує як універсальні CRM/ERP-платформи, так і вузькоспеціалізовані рішення для автомийок та дітейлінгу [6]. Аналіз таких продуктів важливий для визначення “мінімально достатнього” набору функцій, а також для обґрунтування розробки власної системи як дипломного проєкту.

Серед спеціалізованих рішень для дітейлінгу виділяються системи, які поєднують календарний запис, ведення замовлень, комунікацію, оплату та фотофіксацію. Так, RO App позиціонує CRM для дітейлінгу як комплексний інструмент, що охоплює попередній запис у календарі, управління замовленнями, інтеграції каналів комунікації, фінансовий і складський облік, а також можливість фіксації фото “до/після” у замовленні. У Flaber акцент зроблено на онлайн-записі 24/7, контролі етапів робіт, картках авто з історією та фото, оплатах і аналітиці

(популярні послуги, завантаженість, середній чек) [8]. EasyWeek пропонує модель “онлайн-бронювання + календар + повідомлення про статус + оплата онлайн + історія обслуговування”, що фактично відповідає базовим потребам малого бізнесу, орієнтованого на запис і завантаження ресурсів [9]. Універсальні CRM-платформи описують підхід до дітейлінгу як до процесу керування взаємодією з клієнтом, де важливими є централізація клієнтських даних, звітність і автоматизація рутинних комунікацій (нагадування, розсилки, онлайн-запис) [6].

Порівняльний аналіз виконується за критеріями, що безпосередньо відображають бізнес-процеси, а також вимоги до функцій і контролю подано у (табл. 1.3.), де “+” означає наявність функції в явному вигляді, “±” - обмежену підтримку або залежність від тарифу/налаштувань, “–” - відсутність у базовому позиціонуванні.

Таблиця 1.3 – Порівняння існуючих рішень для автомийки/детейлінгу

| № з/п | Критерій | RO App | Flaber | \|EasyWeek | CRM |
|---|---|---|---|---|---|
| 1 | Календарний запис/планувальник | + [3] | + [4] | + [5] | ± [5] |
| 2 | Керування замовленнями/статуси | + [3] | + [4] | | |
| 3 | Картка авто + історія робіт | + [3] | + [4] | + [5] | |
| 4 | Фото “до/після” у замовленні | + [3] | + [4] | ± [5] | |
| 5 | Прийом оплат / інтеграції | + [3] | + [4] | + [5] | |
| 6 | Аналітика (дохід/послуги/чек)\| | + [3] | + [4] | ± [5] | |
| 7 | Складський облік матеріалів | + [3] | ± [4] | – | |
| 8 | Інтеграція месенджерів/каналів | + [3] | ± [4] | ± [5] | |

Як, видно, з таблиці, спеціалізовані рішення прагнуть перекрити весь цикл сервісу, але їх використання може бути обмеженим у навчальному (дипломному) контексті через комерційну ліцензію, закритість внутрішньої архітектури та

залежність від зовнішніх сервісів. Водночас універсальні CRM-системи, хоча й мають сильні сторони в управлінні контактами та комунікаціях, часто потребують значних налаштувань під специфіку дітейлінгу, щоб коректно підтримувати статуси робіт, ресурси боксів, фіксацію наборів послуг і технологічні норми часу.

Для проєкту доцільно розробляти систему, яка покриває мінімальний критичний функціонал: облік клієнтів і авто, формування замовлень із позиціями, календарне планування, контроль статусів, підрахунок вартості, звіти та рольовий доступ. На етапі аналізу предметної області корисно зафіксувати базову інформаційну модель домену, яка надалі стане основою проєктування бази даних і класів у реалізації. Далі наведено концептуальну UML-діаграму класів доменної області (лістинг 1.1), де показано ключові сутності та зв’язки.

Лістинг 1.1 – Модель домена

```
classDiagram
direction LR
    class Client {
        +int client_id
        +string full_name
        +string phone
    }

    class Vehicle {
        +int vehicle_id
        +string brand
        +string plate_number
    }

    class Order {
        +int order_id
        +datetime created_at
        +datetime scheduled_at
        +decimal total_price
    }

    class OrderItem {
        +int quantity
        +decimal price_at_sale
    }

    class Service {
        +int service_id
        +string name
        +decimal base_price
```

```
    +int duration_min
}

class Employee {
    +int employee_id
    +string full_name
    +string position
}

class User {
    +string username
    +boolean is_active
}

class Role {
    +string role_name
}

class UntitledClass {
}

Client "1" -- "0..*" Vehicle : owns
Client "1" -- "0..*" Order : places
Vehicle "1" -- "0..*" Order : serviced_in
Order "1" -- "1..*" OrderItem : consists_of
Service "1" -- "0..*" OrderItem : defines
Order "1" -- "0..*" Employee : assigned_to
User "0..1" -- "1" Employee : identifies
Role "1" -- "0..*" User : grants_permissions
Order -- UntitledClass
```

Концептуальну UML-діаграма класів [10] базується на бізнес-логіці (домені) та інформаційних зв'язках, які забезпечують роботу замовлень, планування та фінансовий контроль (рис.1.4).

Опис доменної моделі:

а) ядро системи:

1) Order – центральний клас, що об'єднує клієнта, автомобіль та статус:

– атрибут планування (scheduled_at);

– атрибут фінансовий підсумок (total_price);

2) OrderItem – специфікація конкретних послуг у замовленні, дозволяє фіксувати ціну на момент продажу;

б) клієнтський блок:

1) Client ;

2) Vehicle;

в) виконавчий блок:

1) Employee – для обліку персоналу;

2) Service – довідник доступних сервісів із базовими цінами та тривалістю;

3) Assignment – реалізує можливість призначення декількох майстрів на одне замовлення;

г) управління та доступ:

1) User – забезпечують рольовий доступ (адмін, менеджер, майстер);

2) Role – забезпечують рольовий доступ (адмін, менеджер, майстер);

3) OrderStatus – дозволяє гнучко керувати життєвим циклом замовлення (від “Заявки» до “Завершено»).

Використання UML як стандартної мови моделювання дозволяє надалі узгодити структуру даних і поведінку системи із вимогами та сценаріями користувачів .

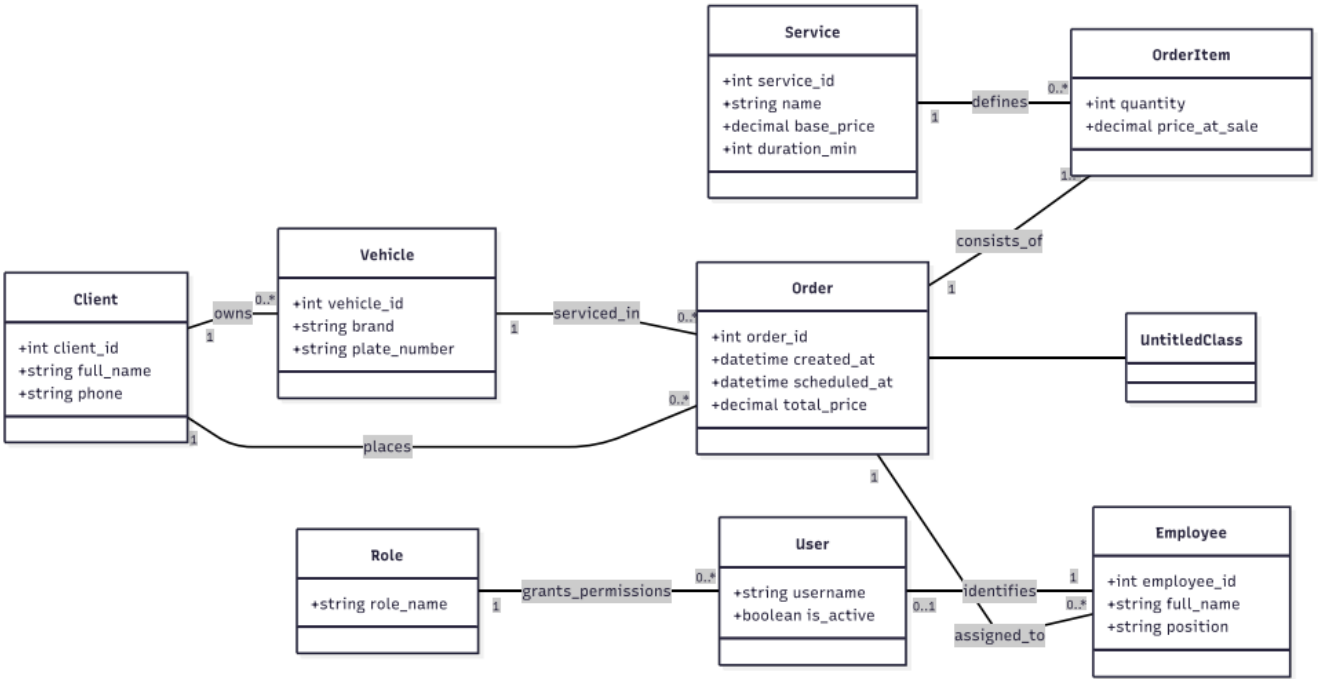


Рисунок 1.4 – Концептуальна UML діаграма класів

У цьому розділі було виконано аналіз предметної області автоклінінгових послуг, визначено специфіку діяльності автомийок і дітейлінг-центрів, а також показано, що це структурований сервісний бізнес з підвищеною відповідальністю та потребою в регламентах, обліку й контролі ризиків. На основі узагальнення типових послуг та процесу “обробка замовлення” сформовано розуміння ключових інформаційних потоків і точок контролю. Проведено аналіз проблем автоматизації та встановлено, що критичними є централізований облік клієнтів і авто, календарний запис, контроль статусів робіт, автоматизований розрахунок вартості та формування звітів. Також зазначено, що для вебсистеми потрібно враховувати базові принципи безпеки, спираючись на загальновизнані підходи, зокрема OWASP Top 10. Порівняння існуючих рішень показало наявність готових продуктів, але обґрунтувало доцільність розробки власної системи в рамках дипломної роботи, оскільки це дозволяє адаптувати функції під конкретні процеси компанії, забезпечити повний контроль архітектури та реалізувати необхідний набір модулів із урахуванням. У першому розділі було проаналізовано предметну область та визначено доцільність розробки спеціалізованої комп’ютерної системи.

# 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ КОМП’ЮТЕРНОЇ СИСТЕМИ

Проєктування архітектури комп’ютерної системи є одним із ключових етапів створення програмного забезпечення, оскільки саме на цьому етапі визначається загальна структура системи, склад її компонентів та принципи їх взаємодії [11]. Правильно спроєктована архітектура забезпечує надійність, масштабованість, зручність супроводу та можливість подальшого розширення функціональності системи.

Архітектура розроблюваної комп’ютерної системи автоклінінгової компанії повинна враховувати специфіку веб-орієнтованих застосунків,

багатокористувацький режим роботи, наявність бази даних, а також можливість розгортання системи на апаратній платформі підприємства.

## 2.1 Загальні вимоги до архітектури системи

Основними вимогами до архітектури комп’ютерної системи є:

- модульність та структурованість;
- підтримка клієнт-серверної взаємодії;
- забезпечення безпеки обміну даними;
- можливість вертикального та горизонтального масштабування;
- простота розгортання та супроводу;
- незалежність клієнтської частини від серверної логіки.

З урахуванням зазначених вимог було обрано клієнт-серверний підхід до побудови архітектури системи.

Розроблювана вебсистема реалізована на основі клієнт-серверної архітектури, яка є стандартом для сучасних інформаційних систем. Згідно з даною архітектурою, обробка даних і бізнес-логіки виконується на сервері, тоді як клієнтська частина відповідає за відображення інтерфейсу користувача.

Основними складовими архітектури є:

- клієнтський рівень;
- сервер прикладної логіки;
- рівень бази даних.

Такий підхід дозволяє ефективно розподілити навантаження, забезпечити централізоване зберігання даних та полегшити масштабування системи.

### 2.1.1 Клієнтський рівень

Клієнтський рівень представлений веббраузером користувача. Доступ до системи здійснюється без встановлення додаткового програмного забезпечення, що значно спрощує експлуатацію та знижує вимоги до робочих станцій.

Функції клієнтського рівня:

– відображення інтерфейсу користувача;
– введення та передача даних на сервер;
– отримання та відображення результатів обробки;
– підтримка інтерактивної взаємодії з користувачем.

Для реалізації клієнтської частини застосовуються стандартні веб-технології: HTML, CSS та JavaScript.

### 2.1.2 Сервер прикладної логіки

Серверна частина системи реалізована з використанням мови програмування PHP та виконує основні обчислювальні й логічні операції. Сервер обробляє HTTP–запити від клієнтів, виконує бізнес–логіку, взаємодіє з базою даних і формує відповіді.

Основні завдання серверного рівня:

– авторизація та автентифікація користувачів;
– перевірка прав доступу;
– обробка замовлень і даних клієнтів;
– виконання розрахунків;
– формування звітів;
– взаємодія з базою даних.

Для впорядкування серверної логіки в системі застосовано архітектурний шаблон MVC (Model–View–Controller).

### 2.1.3 Використання шаблону MVC

Архітектура MVC дозволяє розмежувати різні аспекти системи [12]:

– Model – відповідає за обробку даних та взаємодію з базою даних;
– View – забезпечує відображення інформації користувачеві;
– Controller – обробляє запити користувача та координує взаємодію між моделлю та представленням.

Застосування MVC підвищує читабельність коду, полегшує тестування та забезпечує можливість модифікації окремих компонентів без впливу на

всю систему.

### 2.1.4 Рівень бази даних

Для зберігання інформації використовується реляційна система управління базами даних MySQL [13]. База даних є окремим логічним рівнем архітектури, що забезпечує централізоване та надійне зберігання даних.

Основні функції рівня бази даних:

- зберігання даних користувачів, клієнтів, замовлень та послуг;
- забезпечення цілісності та несуперечності інформації;
- підтримка транзакцій;
- швидкий доступ до даних.

Доступ до бази даних із серверної частини здійснюється за допомогою механізму PDO, що підвищує безпеку та універсальність роботи з СУБД.

### 2.1.5 Фізична та логічна архітектура

З логічної точки зору система складається з взаємопов’язаних програмних модулів, а з фізичної – розгортається на серверному апаратному забезпеченні.

Фізична архітектура включає:

- сервер з операційною системою Linux;
- веб–сервер Apache;
- інтерпретатор PHP;
- сервер бази даних MySQL.

Такий підхід відповідає концепції LAMP–стека та дозволяє забезпечити стабільну та продуктивну роботу системи.

### 2.1.6 Безпека та надійність архітектури

Проєктована архітектура враховує вимоги до безпеки:

- обмеження доступу відповідно до ролей користувачів;
- захист від несанкціонованого доступу до бази даних;
- централізована обробка запитів на сервері;
- мінімізація прямої взаємодії клієнта з базою даних.

Надійність системи забезпечується за рахунок логічного поділу компонентів та можливості розгортання резервних копій бази даних.

### 2.1.7 Масштабованість і можливості розширення

Запропонована архітектура дозволяє:

- збільшувати кількість користувачів без суттєвих змін коду;
- переносити систему на продуктивніше серверне обладнання;
- додавати нові модулі та функції;
- інтегрувати систему з іншими сервісами.

Таким чином, проєктування архітектури комп'ютерної системи забезпечує гнучкість, надійність та ефективність функціонування вебсистеми автоклінінгової компанії і створює основу для її практичного впровадження та подальшого розвитку.

## 2.2 Порівняльний аналіз апаратних компонентів та вибір оптимальної конфігурації

Провівши аналіз апаратних компонентів для виробничого використання в роботі було доцільно обрати Entry Server [14], оскільки він краще підходить для безперервної роботи, резервування та адміністрування (табл..2.1).

Таблиця 2.1 – Вибір класу серверного обладнання

| Параметр | Робочий ПК (Desktop) | Сервер початкового рівня (Entry Server) | Сервер середнього рівня (Mid-range) |
|---|---|---|---|
| Надійність (24/7) | Середня | Висока | Дуже висока |
| Підтримка ECC-пам'яті | Зазвичай ні | Часто так | Так |
| Диски/RAID | Обмежено | RAID1/RAID10 доступні | RAID10/контролери кращі |
| Віддалене керування (iDRAC/iLO) | Ні | Часто так | Так |

Продовження таблиця 2.1

| Параметр | Робочий ПК (Desktop) | Сервер початкового рівня (Entry Server) | Сервер середнього рівня (Mid-range) |
|---|---|---|---|
| Ціна | Низька | Середня | Вища |
| Рекомендація для системи | Лише тест/демо | Оптимально для впровадження | Для росту/філій |

За проаналізованими характеристиками було обрано найбільш відповідну за критеріями модель процесора яка відповідає вимогам для типового автоклінінгу, а саме 6-8 ядер (табл. 2.2) [15].

Таблиця 2.2 – Порівняння процесорів для вебсистеми

| Критерій | 4 ядра / 8 потоків | 6–8 ядер / 12–16 потоків | 12+ ядер |
|---|---|---|---|
| Придатність для 5–10 користувачів | Так | Так | Надлишково |
| Придатність для 10–30 користувачів | Частково (при піках може “просідати”) | Оптимально | Так |
| Пікові навантаження (звіти, масові запити) | Середньо | Добре | Відмінно |
| Вартість | Низька | Середня | Висока |
| Вибір для диплому/реального бізнесу | Мінімум | Рекомендовано | Тільки при масштабі |

З огляду на потреби підприємства, найбільш збалансованим рішенням стала конфігурація з оперативною пам’яттю RAM 16 ГБ [16]. Порівняльна таблиця щодо обрання оперативної пам’яті наведена у таблиці 2.3.

В якості дискової підсистеми [17] було обрано SSD RAID1 як кращий варіант за критеріями 'ціна/швидкодія/надійність' (табл. 2.4), що забезпечує оптимальний баланс між продуктивністю та захистом даних.

Таблиця 2.3 – Порівняння оперативної пам’яті

| RAM | Переваги | Недоліки | Для яких умов підходить | Вибір |
|---|---|---|---|---|
| 8 ГБ | Дешево, достатньо для базового старту | Мало для кешування MySQL та одночасних запитів | 1 оператор , мінімальне навантаження | Мінімум |
| 16 ГБ | Оптимально для MySQL кешу та стабільної роботи | Трохи дорожче | 5–20 користувачів, активні замовлення | Рекомендовано |
| 32 ГБ | Запас під масштаб, швидші звіти | Може бути надлишково | 20–50 користувачів, більше даних/філії | Для росту |

Таблиця 2.4 – Порівняння дискової підсистеми

| Варіант | Швид-кодія | Надійність | Вартість | Коментар для MySQL | Вибір |
|---|---|---|---|---|---|
| HDD (1 диск) | Низька | Низька | Низька | Повільні запити, ризик втрати | Не рекомендовано |
| SSD (1 диск) | Висока | Середня | Середня | Добре для MySQL і логів | Мінімум |
| SSD RAID1 (2 диски) | Висока | Висока | Середня | Оптимально: швидко + захист від відмови диска | Рекомендовано |
| NVMe SSD RAID1 | Дуже висока | Висока | Вища | Для великої кількості операцій/звітів | Для росту |
| RAID10 (4 SSD) | Дуже висока | Дуже висока | Висока | Під серйозне навантаження | Рідко потрібно |

На основі аналізу даних (табл. 2.5) було прийнято рішення застосувати в системі 1 Gbit Ethernet як оптимальне рішення, яке поєднує високу швидкість передачі даних із прийнятною вартістю.

Таблиця 2.5 – Порівняння мережевих інтерфейсів і типу доступу

| Параметр | 100 Мбіт | 1 Гбіт | 2×1 Гбіт (teaming) |
|---|---|---|---|
| Продуктивність у LAN | Середня | Висока | Висока + резерв |
| Доступ через Wi-Fi | Можливо, але не оптимально | Добре | Добре |
| Відмовостійкість | Низька | Середня | Висока |

В якості системи безперебійного живлення використано Line Interactive UPS (табл.2.6), який відповідає вимогам надійності та економічної доцільності, забезпечуючи захист обладнання від збоїв електромережі [18].

Таблиця 2.6 – Порівняння та вибір UPS

| Тип UPS | Плюси | Мінуси | Для сервера |
|---|---|---|---|
| Offline (Standby) | Дешевий | Поганий для частих просідань | Не бажано |
| Line-Interactive | Баланс ціна/якість | Не ідеальний при “важкій” мережі | Добре |
| Online (Double conversion) | Максимальний захист | Дорожчий | Ідеально |

Для вебсистеми автоклінінгової компанії з урахуванням багатокористувацької роботи та вимог до надійності найбільш доцільною буде конфігурація яка наведена в таблиці 2.7.

Таблиця 2.7 – Рекомендована конфігурація  для впровадження

| Компонент | Вибір |
|---|---|
| CPU | 6–8 ядер / 12–16 потоків |
| RAM | 16 ГБ |
| Диски | 2×SSD 512 ГБ (RAID1) |
| Мережа | 1×1GbE (опційно 2× для резерву) |
| UPS | Line-Interactive 1000–1500 VA |
| Призначення | Стабільна робота 10–30 користувачів, з запасом |

Дана конфігурація має наступні переваги:

– забезпечує достатню продуктивність для обробки замовлень та формування звітів;

– гарантує надійність зберігання даних завдяки RAID1;

– підвищує відмовостійкість через використання UPS;

– має запас по ресурсах для зростання кількості користувачів і обсягу даних без негайної модернізації.

## 2.3 Фізична схема розгортання

Вебсистема реалізує класичну модель “клієнт-сервер”, де:

– клієнт – робочі місця персоналу (ПК/ноутбуки/планшети) з веб-браузером;

– сервер застосунку – Apache + PHP, який обробляє запити та реалізує бізнес-логіку;

– сервер бази даних – MySQL (може бути на тому ж фізичному сервері або винесений окремо).

Для малого/середнього автоклінінгу найбільш практичною є монолітна серверна інсталяція (все на одному сервері), оскільки вона:

– дешевша в закупівлі та супроводі;

– простіша в адмініструванні;

– достатня для невеликих навантажень.

Для перспективи масштабування передбачено можливість переходу на дворівневу схему (окремий сервер БД).

Рекомендована конфігурація серверного обладнання залежать від кількості користувачів і обсягу даних. Для більшості автоклінінгових компаній (до 10 – 30 активних користувачів, сотні замовлень на місяць) достатньо конфігурації рівня “entry server”.

Оскільки розроблена система має веб-орієнтовану архітектуру, вимоги до робочих станцій залишаються помірними та не потребують високопродуктивного обладнання. Це дозволяє забезпечити широкі можливості для використання

системи на різних типах пристроїв. Основні вимоги наведено нижче.

Операційна система: допускається використання будь-якої сучасної ОС (Windows, Linux, macOS, Android, iOS), що підтримує актуальні версії веб-браузерів.

Веб-браузер: Chrome, Firefox або Edge останніх версій, які відповідають сучасним стандартам безпеки та сумісності з веб-технологіями.

RAM: мінімальний обсяг 4 ГБ, що забезпечує базову продуктивність; рекомендовано використовувати більший обсяг для комфортної роботи з багатозадачністю.

Мережеве підключення: стабільний доступ до серверу через інтернет або локальну мережу, що гарантує безперервність роботи та швидкий обмін даними.

Завдяки веб-орієнтованій структурі система є кросплатформною та не залежить від конкретного апаратного забезпечення. Це дозволяє користувачам працювати як з персональних комп’ютерів, так і з мобільних пристроїв, зокрема планшетів, безпосередньо в зоні виконання робіт. Такий підхід підвищує мобільність персоналу, оперативність прийняття рішень та ефективність бізнес-процесів. При цьому можливість використання мобільних пристроїв визначається політикою компанії та рівнем доступу до корпоративних ресурсів.

## 2.4 Режими роботи системи

Система може функціонувати у двох основних режимах доступу:

- локальна мережа (LAN);
- доступ через Інтернет (WAN).

У цьому випадку реалізації LAN, сервер розташований безпосередньо в приміщенні компанії, а доступ до нього здійснюється через локальний IP-адрес або внутрішній домен. Такий підхід забезпечує високий рівень контролю над інфраструктурою та мінімальні витрати на організацію доступу.

Для реалізації доступ через WAN необхідний маршрутизатор із підтримкою порт-форвардингу або VPN. З метою безпеки рекомендується використовувати:

– HTTPS (SSL/TLS) для захищеного з’єднання користувачів;

– VPN (WireGuard/OpenVPN**)** для адміністративного доступу.

Основною перевагою WAN-доступу є можливість роботи з будь-якої локації, що особливо актуально для керівників та адміністраторів.

Практичний підхід до безпеки складається з:

– адміністративні сторінки повинні бути доступні лише через VPN або за допомогою “білого списку IP”;

– користувацький доступ має здійснюватися виключно через HTTPS.

## 2.5 Надійність системи

Оскільки база даних містить критично важливі операційні та фінансові дані, необхідно забезпечити комплекс заходів для підвищення надійності [19]:

а) резервне копіювання - рекомендовано застосовувати схему 3-2-1, яка передбачає:

1) наявність трьох копій даних (основна + дві резервні);

2) використання двох типів носіїв (серверний диск + NAS або зовнішній диск);

3) зберігання однієї копії поза сервером (віддалене сховище або інший майданчик);

б) можливі підходи:

1) щоденне логічне резервування за допомогою mysqldump;

2) щотижневий повний бекап у поєднанні з щоденними інкрементальними копіями;

3) зберігання резервних копій протягом 14–30 днів відповідно до політики компанії;

в) використання UPS дозволяє:

1) захистити обладнання від стрибків напруги;

2) забезпечити коректне завершення роботи сервера при відключенні електропостачання;

3) уникнути пошкодження бази даних та файлової системи;

г) RAID1 (дзеркалювання) - це простий та ефективний спосіб підвищення надійності дискової підсистеми. Важливо зазначити, що RAID не замінює резервне копіювання, а лише знижує ризик втрати даних у разі відмови одного з дисків.

## 2.6 Програмно-апаратне середовище (LAMP)

Апаратна частина системи тісно пов’язана з програмним забезпеченням, що формує середовище LAMP:

- операційна система – Linux (Ubuntu Server або Debian), стабільна робота та мінімальні накладні витрати;
- Web-сервер – Apache (альтернативно – Nginx);
- PHP – версія 8.x, що забезпечує сучасний синтаксис та високу продуктивність;
- база даних – MySQL або MariaDB.

Для коректної експлуатації сервера необхідно врахувати:

- налаштування системного часу (NTP);
- права доступу до каталогів проєкту;
- захист конфігураційних файлів (.env, config.php) шляхом винесення їх поза публічний каталог;
- організацію логування (Apache + застосунок).

Моніторинг стабільності роботи системи включає контроль:

- завантаження CPU;

- використання RAM;
- зайнятості диска;
- стану MySQL (кількість з’єднань, повільні запити);
- доступності HTTP/HTTPS.

Рекомендовані практики:

- регулярне оновлення ОС та пакетів безпеки;
- тестове відновлення резервних копій щонайменше раз на місяць;
- журналювання критичних дій користувачів (audit log);
- обмеження доступу до сервера (SSH по ключах, firewall).

## 2.7 Масштабування системи

При зростанні навантаження (збільшення кількості користувачів, замовлень, відкриття нових філій) можливі два основні шляхи масштабування.

Вертикальне масштабування. Передбачає заміну або розширення апаратних ресурсів (CPU, RAM, SSD) на більш потужні. Це простий шлях, який не потребує зміни архітектури системи.

Горизонтальне масштабування.

- винесення MySQL на окремий сервер;
- балансування веб-запитів між кількома серверами застосунку;
- реплікація бази даних (master-replica) для формування звітів без впливу на основні операції.

Таким чином, запропонована архітектура системи поєднує гнучкість у виборі режимів роботи, високий рівень безпеки, надійність зберігання даних та можливість масштабування відповідно до потреб підприємства.

У даному підрозділі обґрунтовано апаратну платформу для розгортання вебсистеми автоклінінгової компанії. Запропонована архітектура передбачає використання фізичного сервера з LAMP–середовищем, локальної або змішаної

мережевої інфраструктури та механізмів забезпечення надійності: UPS, резервне копіювання та RAID. Такий підхід забезпечує стабільну роботу системи в багатокористувацькому режимі, збереження даних та можливість подальшого масштабування при зростанні бізнесу.

# 3 РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Вимоги до комп'ютерної системи

Комп'ютерна система автоклінінгової компанії призначена для автоматизації основних бізнес-процесів підприємства, підвищення ефективності роботи персоналу та покращення якості обслуговування клієнтів. Для досягнення поставлених цілей система повинна відповідати визначеному набору функціональних і нефункціональних вимог, які формуються з урахуванням специфіки діяльності автоклінінгових компаній, умов їх експлуатації та вимог до сучасних веб-орієнтованих інформаційних систем [21].

Визначення вимог до програмного забезпечення є одним із ключових етапів проєктування, оскільки забезпечує відповідність розроблюваної системи реальним потребам користувачів і дозволяє уникнути помилок на етапі реалізації.

### 3.1.1 Функціональні вимоги

Функціональні вимоги визначають перелік основних можливостей комп'ютерної системи та описують, які саме операції вона повинна виконувати.

Однією з базових вимог є реєстрація та авторизація користувачів. Система повинна забезпечувати створення облікових записів для різних категорій користувачів (адміністратор, оператор, виконавець), а також перевірку автентичності при вході до системи. Авторизація повинна супроводжуватись розмежуванням прав доступу відповідно до ролі користувача.

Важливою функцією є управління клієнтами, яке включає зберігання контактної інформації, історії звернень та замовлень. Це дозволяє швидко

отримувати дані про клієнта, аналізувати його активність та покращувати рівень сервісу.

Система повинна підтримувати формування та обробку замовлень. Користувачі мають можливість створювати замовлення, вказувати перелік обраних послуг, призначати відповідальний персонал та відстежувати стан виконання замовлення в режимі реального часу.

Окремою функціональною вимогою є облік наданих послуг. Система повинна зберігати довідник послуг автоклінінгової компанії із зазначенням вартості, опису та інших параметрів, що дозволяє швидко формувати замовлення та уникати помилок при обслуговуванні клієнтів.

Для забезпечення прозорості операцій необхідно реалізувати автоматичний розрахунок вартості замовлення. Загальна сума повинна формуватись на основі вибраних послуг, що зменшує ризик помилок та підвищує довіру клієнтів.

Крім того, система має забезпечувати управління персоналом, що включає облік працівників, їх ролей та участі у виконанні замовлень. Це дозволяє контролювати завантаженість персоналу та підвищити ефективність планування робіт. Останньою, але не менш важливою функціональною вимогою є формування звітів. Система повинна надавати можливість формування зведеної інформації щодо кількості замовлень, доходів, популярності послуг та роботи персоналу, що є основою для прийняття управлінських рішень.

### 3.1.2 Нефункціональні вимоги

Нефункціональні вимоги характеризують якісні показники системи та умови її експлуатації.

Система повинна бути доступною через веб–браузер, що забезпечує незалежність від операційної системи користувача та можливість роботи з будь–якого пристрою, підключеного до мережі.

Однією з ключових вимог є безпечне зберігання даних. Персональна інформація клієнтів і службові дані повинні зберігатись у базі даних з дотриманням вимог інформаційної безпеки, застосуванням механізмів автентифікації та захисту від несанкціонованого доступу.

Система повинна коректно працювати у багатокористувацькому режимі, підтримуючи одночасну роботу декількох користувачів без втрати даних або зниження швидкодії.

Важливою вимогою є можливість розгортання на власному сервері автоклінінгової компанії. Це дозволяє підприємству контролювати апаратне забезпечення, дані та забезпечувати автономну роботу системи без залежності від сторонніх сервісів.

Крім того, система повинна характеризуватись простотою масштабування, що дає можливість розширювати функціонал, збільшувати кількість користувачів або переносити систему на більш продуктивне серверне обладнання без суттєвих змін у програмному коді.

## 3.2 Функціональна модель комп’ютерної системи

Функціональна модель комп’ютерної системи призначена для опису складу функцій, що реалізуються системою, а також взаємодії між користувачами та програмним забезпеченням. Вона дозволяє формалізувати вимоги до системи, визначити основні сценарії її використання та слугує основою для подальшого проєктування програмної архітектури і логіки роботи [22].

Функціональна модель розроблюваної комп’ютерної системи автоклінінгової компанії орієнтована на автоматизацію процесів обслуговування клієнтів, управління замовленнями та контролю діяльності персоналу. Для побудови моделі використовується підхід, заснований на використанні ролей користувачів і потоків виконання основних операцій.

У системі передбачено декілька типів користувачів, кожен з яких має власні функціональні повноваження:

– “Адміністратор” – має повний доступ до системи, здійснює управління користувачами, довідниками послуг, персоналом та перегляд

узагальнених звітів.

– “Оператор” – здійснює реєстрацію клієнтів, створення та супровід замовлень, взаємодіє з клієнтами.

– “Виконавець” – має доступ до перегляду призначених йому замовлень та статусу їх виконання.

Поділ користувачів за ролями дозволяє забезпечити безпеку даних та впорядкувати бізнес-процеси відповідно до функціональних обов’язків персоналу.

Функціональна модель включає декілька ключових функцій:

– авторизація та автентифікація користувачів - перед початком роботи користувач проходить процедуру входу до системи, під час якої перевіряються його облікові дані. Залежно від ролі користувача надається відповідний рівень доступу;

– управління клієнтами - система забезпечує створення, редагування та зберігання інформації про клієнтів, наявність централізованої бази клієнтів дозволяє швидко працювати з історією замовлень та поліпшувати якість обслуговування;

– формування та супровід замовлень - оператор має можливість створювати нові замовлення, обирати перелік послуг, призначати виконавців та встановлювати статус виконання замовлення, система зберігає інформацію про кожен етап обробки замовлення;

– облік наданих послуг - функція обліку послуг передбачає використання довідника послуг із фіксованими цінами, це дозволяє стандартизувати процес обслуговування та виключити помилки при формуванні вартості;

– автоматичний розрахунок вартості замовлення - загальна вартість замовлення формується автоматично на основі обраного набору послуг, що спрощує роботу персоналу та зменшує вплив людського фактора;

– управління персоналом - адміністратор системи може додавати працівників, призначати ролі та контролювати їх участь у виконанні замовлень;

– формування аналітичних звітів - система забезпечує формування

звітів щодо кількості замовлень, фінансових показників, популярності послуг та ефективності роботи персоналу.

Одним із ключових сценаріїв функціональної моделі є процес обробки замовлення. Він включає такі етапи:

- авторизація оператора у системі;
- вибір або створення нового клієнта;
- формування замовлення та вибір послуг;
- автоматичний розрахунок загальної вартості;
- призначення виконавця;
- виконання замовлення;
- завершення та збереження результатів роботи.

Даний сценарій відображає основну логіку роботи автоклінінгової компанії та є базовим для побудови програмної реалізації.

Запропонована функціональна модель:

- забезпечує логічну структуру системи;
- зменшує складність реалізації програмного забезпечення;
- підвищує надійність і керованість бізнес-процесів;
- є масштабованою та може бути розширена додатковими функціями.

## 3.3 Моделювання бізнес-процесів комп’ютерної системи

Моделювання бізнес-процесів є важливим етапом проєктування комп’ютерної системи, оскільки дозволяє формалізувати логіку роботи підприємства, визначити основні сценарії взаємодії користувачів із системою та виявити взаємозв’язки між окремими функціональними компонентами. Результати моделювання використовуються як основа для побудови архітектури програмного забезпечення та подальшої його реалізації [23].

Для моделювання бізнес–процесів у даній дипломній роботі використано

уніфіковану мову моделювання UML , яка є загальноприйнятим стандартом опису програмних систем. UML дозволяє наочно та формалізовано описати структуру системи, поведінку користувачів і послідовність виконання операцій.

Основними цілями використання UML у межах даної роботи є:

- опис функціональних вимог у графічній формі;
- виявлення основних акторів і варіантів використання;
- демонстрація логіки взаємодії користувачів із системою;
- зменшення складності подальшої реалізації програмної логіки;
- забезпечення масштабованості та зрозумілості системи.

Найбільш доцільним для опису бізнес–процесів автоклінінгової компанії є використання Use Case-діаграм [24], які відображають функціональні можливості системи з точки зору кінцевого користувача.

Актор у UML представляє роль, з якою взаємодіє система. Кожен із акторів взаємодіє з системою лише в межах дозволених йому повноважень, що забезпечує безпеку та структурованість бізнес–процесів.

На основі аналізу предметної області було визначено такі ключові варіанти використання системи (рис.3.1):

- авторизація користувача – перед початком роботи користувач повинен пройти процедуру входу до системи;
- управління клієнтами – передбачає додавання, редагування та перегляд інформації про клієнтів;
- створення замовлення – оператор формує нове замовлення, обираючи клієнта та перелік послуг;
- облік послуг – система використовує довідник послуг із фіксованими цінами;
- розрахунок вартості – замовлення автоматично визначає підсумкову вартість на основі обраних послуг;
- призначення виконавця – до замовлення закріплюється працівник, відповідальний за виконання робіт;
- контроль виконання – замовлення виконавець змінює статус

замовлення у процесі його виконання;

– формування звітів – адміністратор отримує аналітичну інформацію про діяльність компанії.


Рисунок 3.1 – Система управління автоклінінгом

Одним із центральних бізнес–процесів є варіант використання “Обробка замовлення”. Його алгоритм можна описати наступною послідовністю дій:

– оператор проходить авторизацію в системі;

– обирає існуючого клієнта або реєструє нового;

– створює замовлення та додає до нього перелік послуг;

– система автоматично розраховує загальну вартість;

– оператор призначає виконавця;

– виконавець переглядає призначене замовлення та виконує роботи;

– після завершення замовлення змінюється його статус на “Виконано”.

Даний Use Case є базовим сценарієм, який відображає основну діяльність автоклінінгової компанії.

Застосування UML–діаграм у процесі проєктування надає такі переваги:

– чітке відображення структури бізнес-процесів;

– спрощення комунікації між розробником і замовником;

– можливість швидкого внесення змін;

– зручна основа для реалізації серверної логіки;

– відповідність сучасним стандартам проєктування програмних систем.

Результати моделювання бізнес–процесів використовуються під час:

– розробки архітектури системи;

– проєктування бази даних;

– написання серверних модулів на PHP;

– формування інтерфейсу користувача.

Таким чином, моделювання бізнес–процесів є обов'язковим і ключовим етапом створення комп'ютерної системи автоклінінгової компанії.

# 4 ПРОЄКТУВАННЯ БАЗИ ДАНИХ

Проєктування бази даних є критично важливим етапом створення комп'ютерної системи автоклінінгової компанії, оскільки саме база даних забезпечує централізоване зберігання інформації, її цілісність, швидкий доступ та можливість формування аналітичних звітів. Для розроблюваної вебсистеми обрано реляційну модель даних з використанням СУБД MySQL, що є доцільним для задач обліку замовлень, клієнтів, послуг і персоналу [25].

Реляційний підхід дозволяє описати дані у вигляді взаємопов'язаних таблиць, застосувати механізми первинних та зовнішніх ключів, а також забезпечити цілісність даних через обмеження (constraints) та транзакції.

База даних системи повинна забезпечувати:

– зберігання основних сутностей: користувачі, клієнти, транспортні засоби, послуги, замовлення, виконавці, оплати;

– підтримку багатокористувацької роботи з коректним паралельним доступом;

– цілісність даних (посилальна цілісність, уникнення "висячих"

записів);

– швидке виконання типових запитів (перелік замовлень, фільтрація за статусом/датою, звіти);

– можливість масштабування: розширення довідників, додавання нових модулів (наприклад, онлайн–оплата, склад матеріалів);

– безпеку: збереження паролів у хешованому вигляді, контроль доступу по ролях.

## 4.1 Концептуальна модель даних (ER-модель)

Для автоклінінгової компанії логічно виділити такі сутності:

– User (Користувач) – обліковий запис для входу в систему (адмін/оператор/виконавець);

– Role (Роль) – набір прав доступу;

– Employee (Працівник) – дані про персонал (може бути пов’язаний із User);

– Client (Клієнт) – контактні дані клієнта;

– Vehicle (Автомобіль) – транспортний засіб клієнта;

– Service (Послуга) – довідник послуг і базових цін;

– Order (Замовлення) – факт звернення/обслуговування;

– OrderItem (Позиція замовлення) – перелік послуг у межах замовлення;

– OrderAssignment (Призначення виконавців) – хто виконує замовлення (можлива участь кількох працівників);

– Payment (Оплата) – інформація про оплату;

– Status (Статус замовлення) – довідник станів виконання (нове/в роботі/виконано/скасовано).

На рисунку 4.1 наведено ER-діаграму сутностей системи.

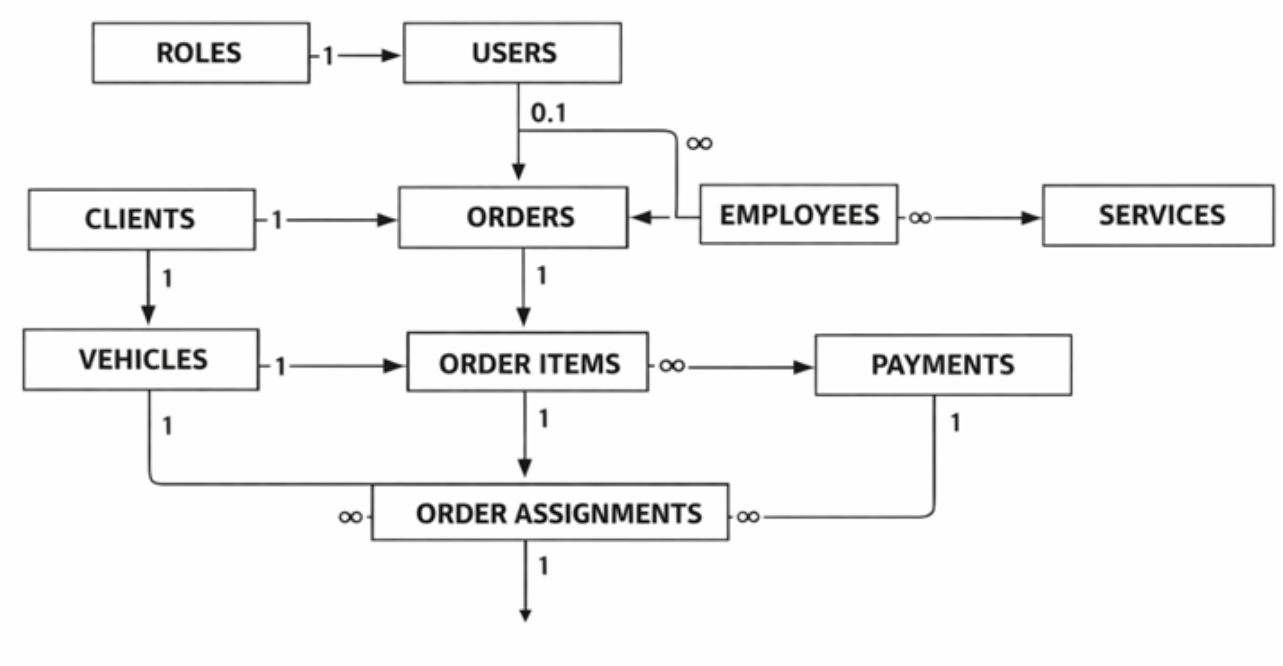


Рисунок 4.1 – ER–діаграму сутностей системи

Текстовий опис ER–зв’язків:

– Role 1–∞ User → одна роль може належати багатьом користувачам;

– Client 1–∞ Vehicle → клієнт має кілька автомобілів;

– Client 1–∞ Order → клієнт створює багато замовлень;

– Vehicle 1–∞ Order → авто може бути пов’язане з багатьма замовленнями;

– Order 1–∞ OrderItem → замовлення складається з багатьох позицій;

– Service 1–∞ OrderItem → послуга може входити до багатьох замовлень;

– Order ∞–∞ Employee (через OrderAssignment) → замовлення виконується кількома працівниками;

– Order 1–∞ Payment → замовлення може мати кілька оплат;

– Status 1–∞ Order → кожне замовлення має один статус.

Концептуальна модель даних забезпечує формалізований опис предметної області та є основою для подальшого створення логічної та фізичної моделі бази даних у СУБД MySQL.

## 4.2 Логічна модель даних

Логічна модель бази даних визначає структуру таблиць, їх ключові поля, типи в’язків та правила забезпечення цілісності [26]. Вона є основою для фізичної реалізації в СУБД MySQL.

В таблиці 4.1 наведено структуру бази даних автоклінінгової компанії їх ключові поля та призначення.

Таблиця 4.1 – Структура бази даних автоклінінгової компанії

| № | Таблиця | Первинний ключ (PK) | Зовнішні ключі (FK) | Основні поля | Призначення |
|---|---|---|---|---|---|
| 1 | roles | role_id | — | name, description | Довідник ролей користувачів |
| 2 | users | user_id | role_id → roles.role_id | username, password_hash, email, is_active, created_at, last_login | Облікові записи користувачів |
| 3 | employees | employee_id | user_id → users.user_id (опційно) | full_name, phone, position, hire_date, is_active | Дані про працівників |
| 4 | clients | client_id | — | full_name, phone, email, notes, created_at | Контактна інформація клієнтів |
| 5 | vehicles | vehicle_id | client_id → clients.client_id | brand, model, plate_number, color, year | Автомобілі клієнтів |
| 6 | services | service_id | — | name, base_price, duration_min, is_active, description | Довідник послуг |

Продовження таблиці 4.1

| № | Таблиця | Первинний ключ (PK) | Зовнішні ключі (FK) | Основні поля | Призначення |
|---|---|---|---|---|---|
| 7 | order_statuses | status_id | — | name, sort_order | Довідник статусів замовлень |
| 8 | orders | order_id | client_id → clients.client_id, vehicle_id → vehicles.vehicle_id, status_id → order_statuses.status_id | created_at, scheduled_at, total_amount, comment | Замовлення клієнтів |
| 9 | order_items | order_item_id | order_id → orders.order_id, service_id → services.service_id | qty, price_at_time, line_total | Позиції замовлення |
| 10 | order_assignments | assignment_id | order_id → orders.order_id, employee_id → employees.employee_id | assigned_at, role_in_order | Призначення виконавців |
| 11 | payments | payment_id | order_id → orders.order_id | amount, method, paid_at, status | Оплати за замовлення |

Усі таблиці мають первинний ключ INT AUTO_INCREMENT**,** який:

- гарантує унікальність запису;
- пришвидшує з’єднання таблиць (JOIN);
- спрощує посилання у зовнішніх ключах.

У SQL (переважно у зовнішніх ключах / FOREIGN KEY) стратегії ON DELETE та ON UPDATE визначають, що робити із залежними записами, коли батьківський запис видаляється або його ключ змінюється.

Зовнішні ключі (Foreign Key, FK) задають зв’язки та забезпечують посилальну цілісність.

Стратегія ON DELETE визначає поведінку при видаленні батьківського запису.

Стратегія ON UPDATE забороняє видалення батька, якщо є дочірні записи.

Проєктування бази даних виконано з урахуванням принципів нормалізації, що зменшує дублювання даних і підвищує узгодженість:

– 1НФ – усі поля атомарні (немає списків у полі “Послуги” вони винесені в order_items);

– 2НФ – атрибути залежать від первинного ключа (позиція замовлення залежить від order_item_id, а не від частини ключа);

– 3НФ – відсутні транзитивні залежності (наприклад, назва послуги не дублюється в orders, а зберігається в services).

Для забезпечення надійної та коректної роботи бази даних важливо правильно обрати механізм зберігання та кодування. Використання механізму InnoDB забезпечує підтримку транзакцій, зовнішніх ключів і механізмів відновлення даних, що підвищує цілісність та безпеку інформації. Кодування utf8mb4 дозволяє коректно зберігати українські символи, спеціальні знаки та символи інших мов, забезпечуючи повну підтримку Unicode і сумісність із сучасними вебзастосунками. Індекси для продуктивності

Для пришвидшення типових запитів доцільно створити індекси:

```
orders(created_at), orders(status_id), orders(client_id)
order_items(order_id), order_items(service_id)
vehicles(client_id), опційно vehicles(plate_number)
payments(order_id)
```

Це прискорює:

– перегляд замовлень за датою;

– фільтрацію за статусом;

– побудову звітів по клієнтах/послугах.

Проєктування структури та індексів орієнтоване на часті сценарії:

– список замовлень за датою/статусом (індекси в `orders`);

– детальний склад замовлення (`JOIN orders+order_items+services`);

– звіт “топ-послуги” (агрегація `order_items по service_id`);

– завантаженість працівників (`JOIN order_assignments+orders`).

### 4.2.1 ER-діаграми

Нижче наведено текстовий та графічне представлення (рис.4.2–4.12) опис ER–моделей розроблюваної системи у форматі “сутність – атрибути – зв’язки” [27]. Такий опис дозволяє чітко визначити структуру даних, кардинальної та ключові залежності.

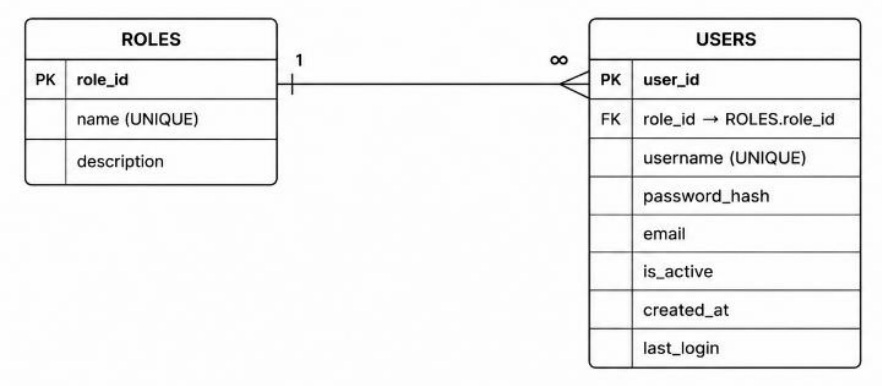


Рисунок 4.2 – ER–діаграма ROLES

Сутність ROLES (рис.4.2):

- атрибути – role_id (PK), name (UNIQUE), description;
- зв’язки – ROLES (1) - (∞) USERS.

Одна роль відповідає багатьом користувачам.

Сутність USERS (рис.4.3):

- атрибути – user_id (PK), role_id (FK→ROLES), username (UNIQUE), password_hash, email, is_active, created_at, last_login;
- зв’язки – USERS (∞) - (1) ROLES USERS (1) - (0..1) EMPLOYEES (якщо працівник має акаунт).

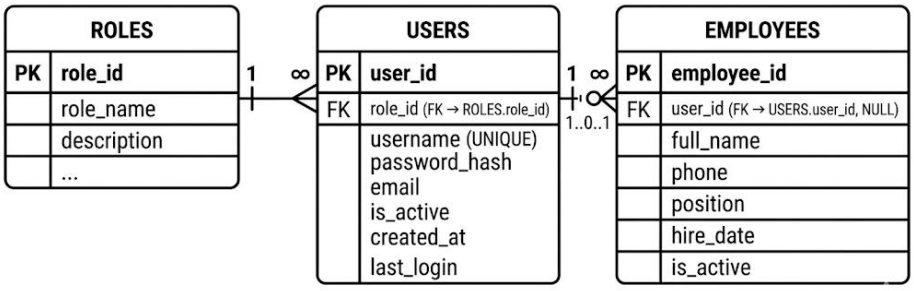


Рисунок 4.3 – ER–діаграма USERS

Сутність EMPLOYEES (рис.4.4):

– атрибути – employee_id (PK), user_id (FK→USERS, NULL), full_name, phone, position, hire_date, is_active;

– зв’язки – EMPLOYEES (∞) → (∞) ORDERS через ORDER_ASSIGNMENTS (працівник може виконувати багато замовлень, і одне замовлення можуть виконувати декілька працівників).

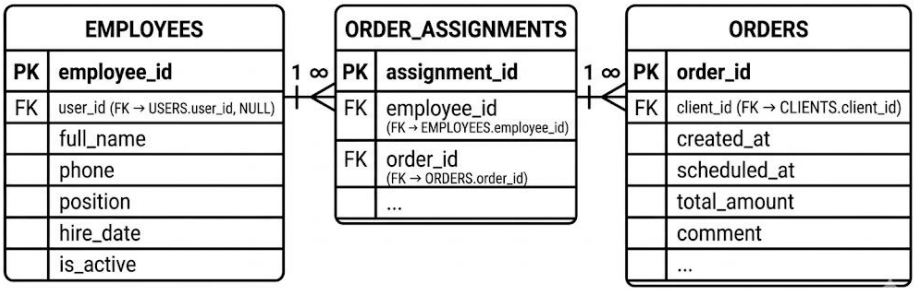


Рисунок 4.4 – ER–діаграма EMPLOYEES

Сутність: CLIENTS (рис.4.5)

– атрибути – client_id (PK), full_name, phone (UNIQUE), email, notes, created_at;

– зв’язки – CLIENTS (1) → (∞) VEHICLES CLIENTS (1) → (∞) ORDERS.

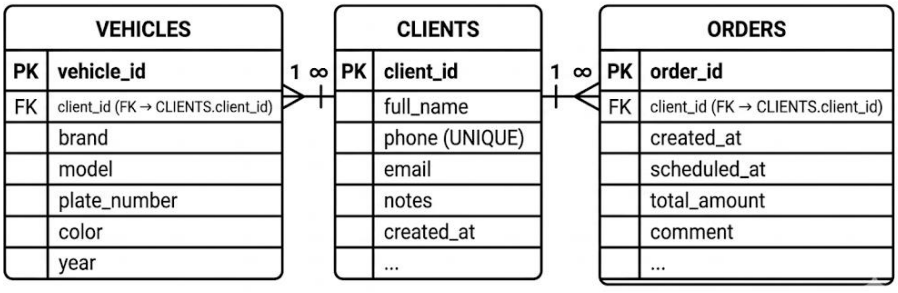


Рисунок 4.5 – ER–діаграма CLIENTS

Сутність: VEHICLES (рис.4.6):

– атрибути – vehicle_id (PK), client_id (FK→CLIENTS), brand, model, plate_number, color, year;

– зв’язки – VEHICLES (∞) → (1) CLIENTS VEHICLES (1) → (∞) ORDERS (одне авто може мати багато замовлень).

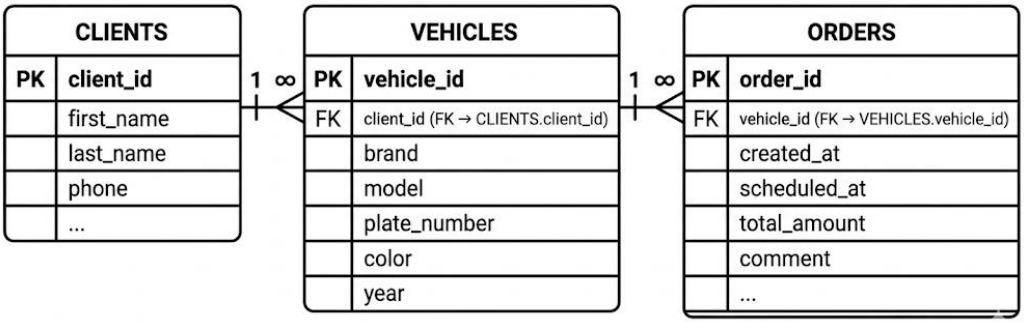


Рисунок 4.6 – ER–діаграма VEHICLES

Сутність: SERVICES (рис.4.7):

– атрибути – service_id (PK), name (UNIQUE), base_price, duration_min, description, is_active;

– зв’язки – SERVICES (1) → (∞) ORDER_ITEMS (одна послуга може входити до багатьох замовлень).

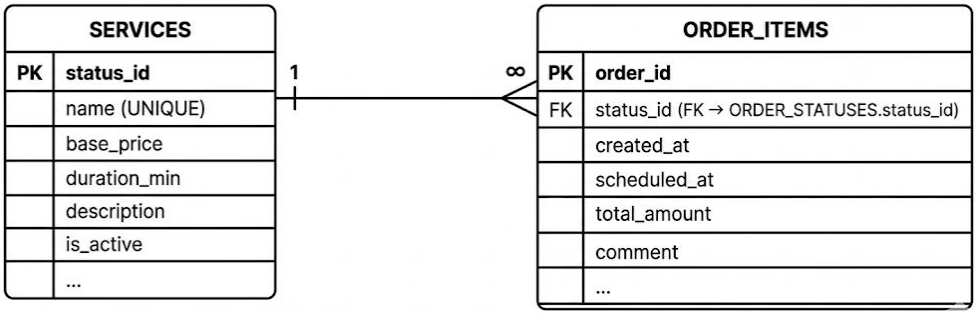


Рисунок 4.7 – ER–діаграма SERVICES

Сутність: ORDER_STATUSES (рис. 4.8):

– атрибути – status_id (PK), name (UNIQUE), sort_order;

– зв’язки – ORDER_STATUSES (1) → (∞) ORDERS.

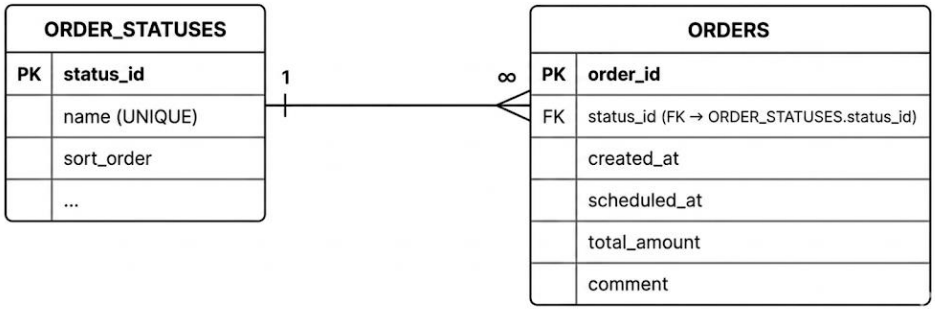


Рисунок 4.8 – ER–діаграма ORDER_STATUSES

Сутність: ORDERS (рис. 4.9):

– атрибути – order_id (PK), client_id (FK→CLIENTS), vehicle_id (FK→VEHICLES), status_id (FK→ORDER_STATUSES),created_at, scheduled_at, total_amount, comment;

– зв’язки – ORDERS (∞) – (1) CLIENTSORDERS (∞) – (1) VEHICLES ORDERS (∞) – (1) ORDER_STATUSES ORDERS (1) – (∞) ORDER_ITEMS ORDERS (1) – (∞) PAYMENTS ORDERS (∞) – (∞) EMPLOYEES через ORDER_ASSIGNMENTS.

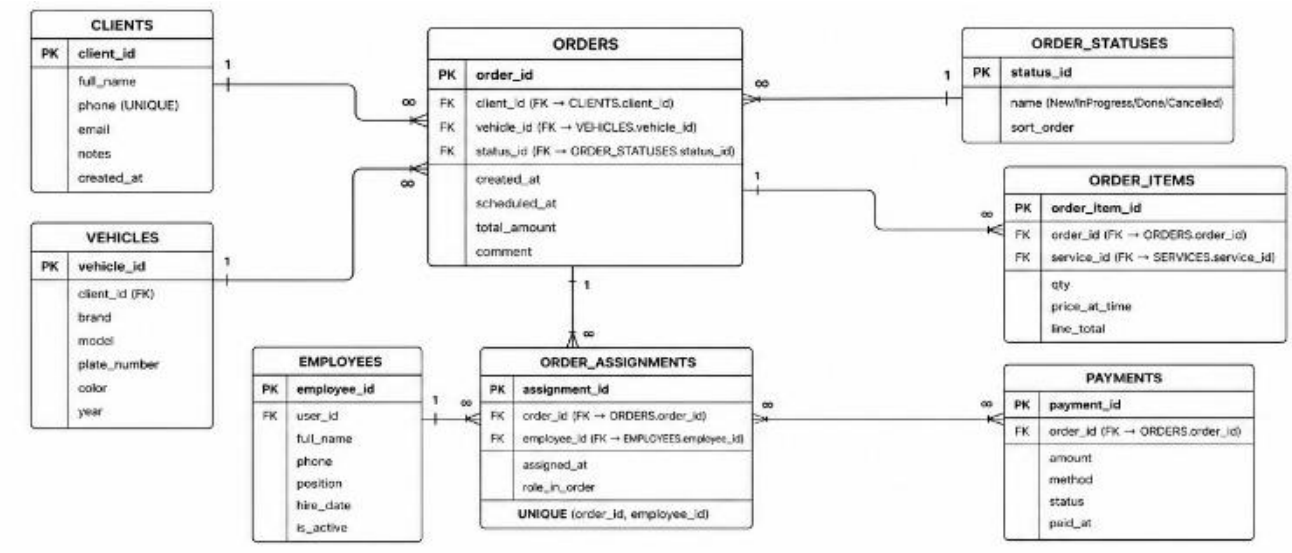


Рисунок 4.9 – ER–діаграма ORDERS

Сутність: ORDER_ITEMS (рис.4.10):

– атрибути – order_item_id (PK), order_id (FK→ORDERS), service_id

(FK→SERVICES), qty, price_at_time, line_total;

– зв’язки – ORDER_ITEMS (∞) – (1) ORDERS ORDER_ITEMS (∞) – (1) SERVICES.

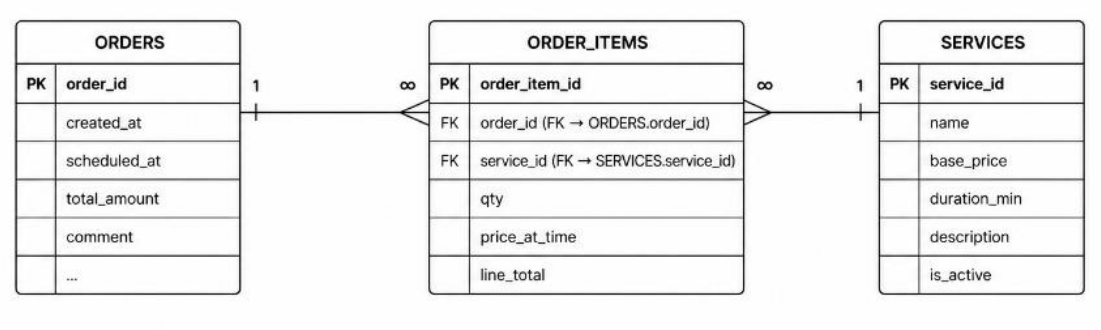


Рисунок 4.10 – ER–діаграма ORDER_ITEMS

Сутність: ORDER_ASSIGNMENTS (рис.4.11):

– атрибути – assignment_id (PK), order_id (FK→ORDERS), employee_id (FK→EMPLOYEES), assigned_at, role_in_order, (UNIQUE: order_id + employee_id);

– зв’язки – ORDER_ASSIGNMENTS (∞) – (1) ORDERS ORDER_ASSIGNMENTS (∞) - (1) EMPLOYEES.

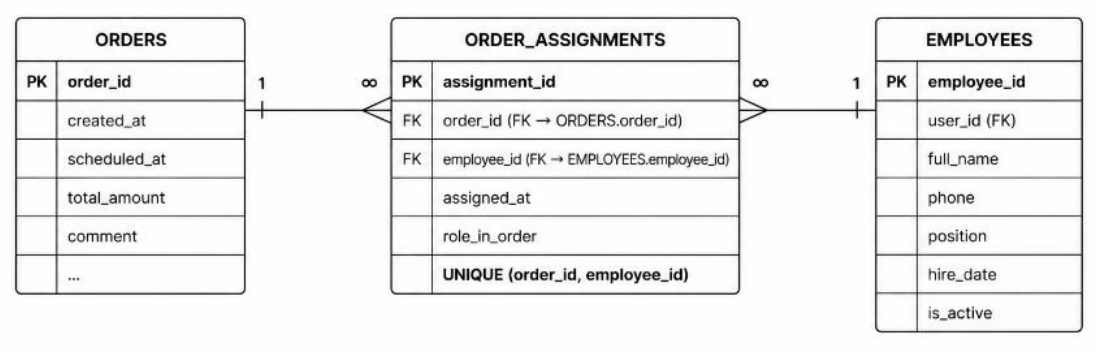


Рисунок 4.11 – ER–діаграма ORDER_ASSIGNMENTS

Сутність: PAYMENTS (рис.4.12):

– атрибути – payment_id (PK), order_id (FK→ORDERS), amount, method, status, paid_at;

– зв’язки – PAYMENTS (∞) – (1) ORDERS (один заказ може мати

декілька оплат (передоплата/доплата).

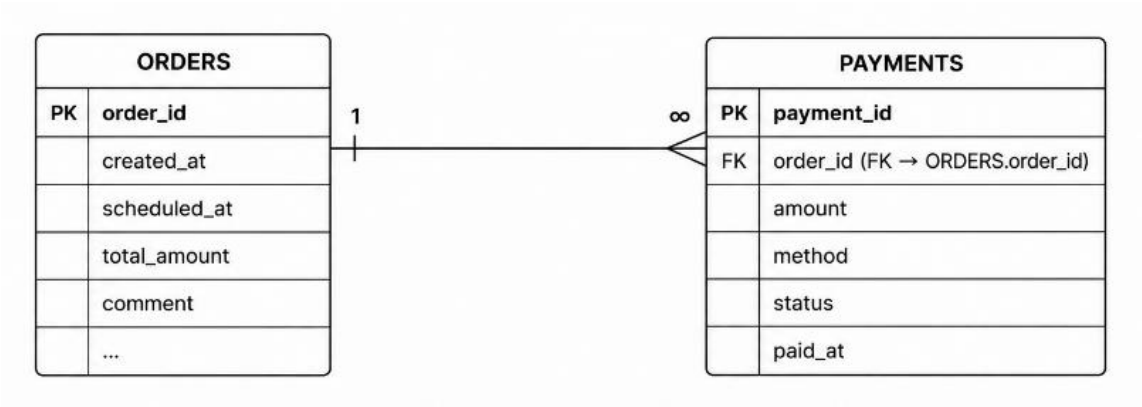


Рисунок 4.12 – ER–діаграма PAYMENTS

Таким чином, логічна модель бази даних забезпечує цілісне уявлення про структуру таблиць, ключові поля та типи зв’язків, що дозволяє узгоджено представити загальну архітектуру системи у вигляді ER-діаграм та текстового опису. Загальна структура моделі бази даних представлена на рисунку 4.13.

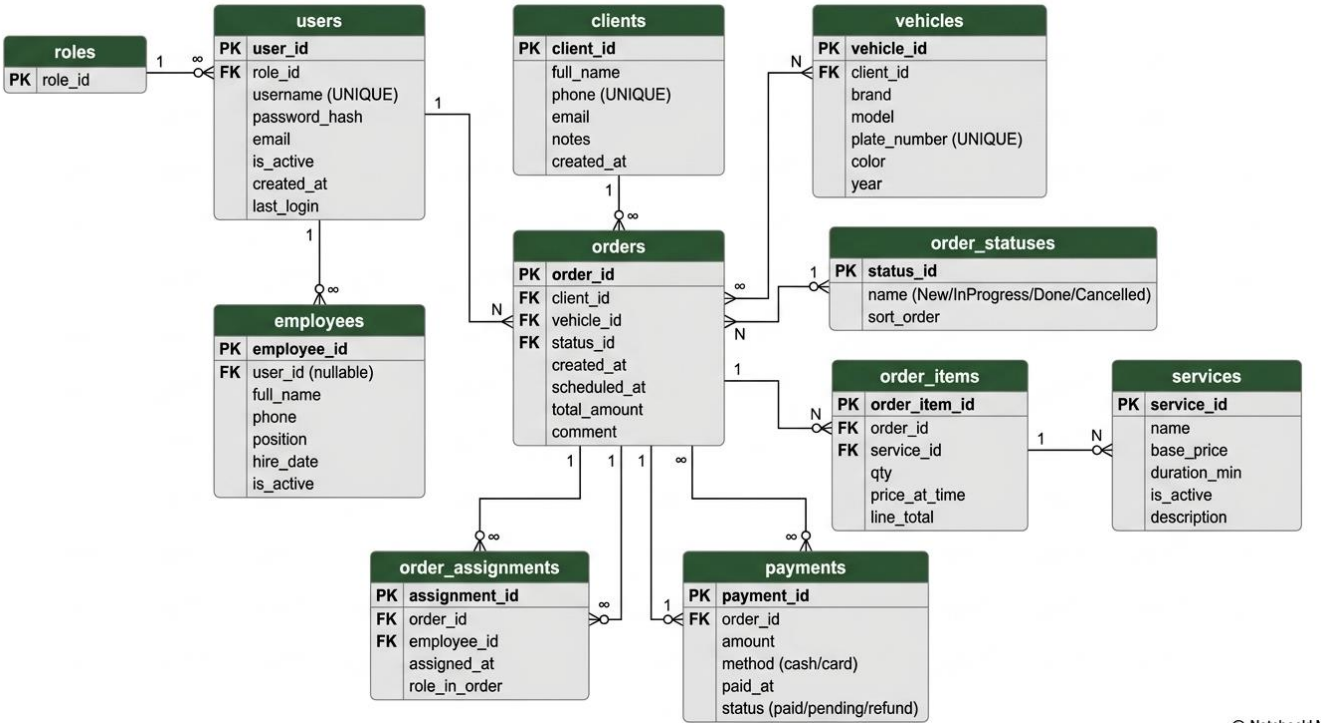


Рисунок 4.13 – ER–діаграма розроблювальної системи

### 4.3 Опис транзакцій

Операція створення замовлення в системі є складеною і включає кілька логічно пов’язаних кроків:

– додавання запису в `orders;`

– додавання набору позицій у `order_items;`

– перерахунок `orders.total_amount;`

– (за потреби) призначення виконавця в `order_assignments.`

Оскільки всі ці кроки повинні виконуватись як єдине ціле, необхідно використовувати транзакцію (рис.4.14). Це гарантує властивості ACID: якщо хоча б один крок не вдався - система повертається до початкового стану (ROLLBACK).

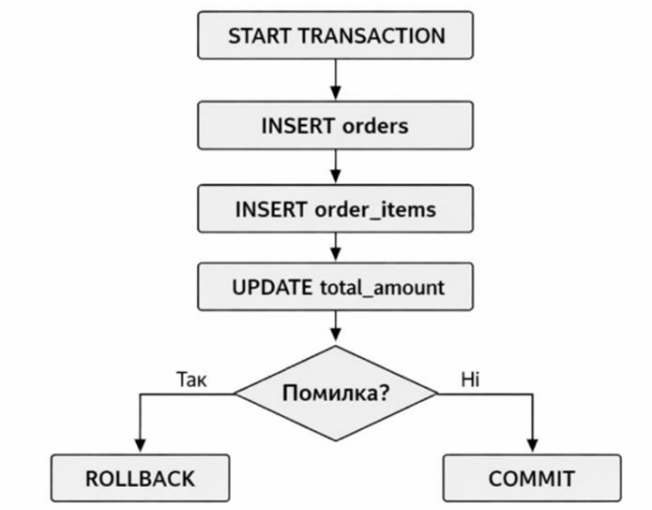


Рисунок 4.14 – Блок–схема транзакції створення замовлення

Логіка транзакції включає наступні елементи:

– START TRANSACTION – початок транзакції;

– INSERT INTO orders – створюється “шапка” замовлення;

– INSERT INTO order_items (N разів) – додаються послуги;

– INSERT INTO order_assignments – призначення виконавців;

– перерахунок total_amount – виконується агрегуванням

SUM(line_total);

– COMMIT – фіксація змін;

– у разі помилки: ROLLBACK – відкат.

Приклад транзакції наведено на рисунку 4.15.

```
START TRANSACTION;

-- 1) Створюємо замовлення (total_amount тимчасово 0)
INSERT INTO orders (client_id, vehicle_id, status_id, scheduled_at, total_amo
VALUES (1, 1, 1, '2026-04-27 12:00:00', 0.00, 'Звичайне миття + хімчистка');

SET @new_order_id = LAST_INSERT_ID();

-- 2) Додаємо позиції замовлення
INSERT INTO order_items (order_id, service_id, qty, price_at_time, line_total
VALUES
(@new_order_id, 2, 1, 250.00, 250.00),
(@new_order_id, 5, 1, 600.00, 600.00);

-- 3) Перерахунок підсумкової суми
UPDATE orders o
SET o.total_amount = (
  SELECT COALESCE(SUM(oi.line_total), 0)
  FROM order_items oi
  WHERE oi.order_id = @new_order_id
)
WHERE o.order_id = @new_order_id;
```

Рисунок 4.15 – Транзакції SQL-послідовність

У PHP доцільно використовувати у наступному вигляді:

- `$pdo->beginTransaction();`
- `try/catch з $pdo->commit(); або $pdo->rollBack();`
- `підготовлені запити (prepared statements) для безпеки й уникнення SQL-ін»єкцій.`

Щоб не покладатися виключно на прикладний рівень (PHP), підрахунок `orders.total_amount` доцільно автоматизувати на рівні СУБД. Для цього застосовується (рис.4.16):

– збережена процедура `recalc_order_total(order_id)` – перераховує суму замовлення;

– тригери на таблицю `order_items` – викликають процедуру при вставці/оновленні/видаленні позиції.

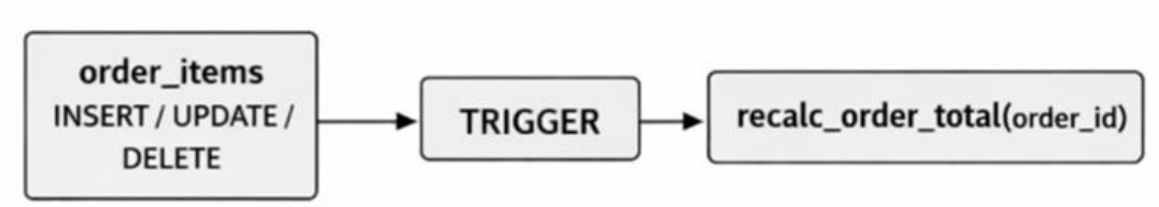


Рисунок 4.16 – Схема взаємодії тригерів і процедури перерахунку суми

Такий підхід забезпечує цілісність: незалежно від того, як змінюються позиції – сума завжди коректна.

Процедура перерахунку підсумкової суми представлено на наступних рисунках 4.17 та 4.18.

```
CREATE PROCEDURE recalc_order_total(IN p_order_id INT)
BEGIN
  UPDATE orders o
  SET o.total_amount = (
    SELECT COALESCE(SUM(oi.line_total), 0)
    FROM order_items oi
    WHERE oi.order_id = p_order_id
  )
  WHERE o.order_id = p_order_id;
END;
```

Рисунок 4.17 – Перерахунок підсумкової суми

```
CREATE TRIGGER trg_order_items_after_change
AFTER INSERT OR UPDATE OR DELETE ON order_items
FOR EACH ROW
BEGIN
  CALL recalc_order_total(NEW.order_id);
END;
```

Рисунок 4.18 – Тригери для автоматичного виклику перерахунку після додавання позиції

```
DELIMITER $

CREATE TRIGGER trg_ordder_items_au
AFTER UPDATE ON order_items
FOR EACH ROW
 BEGIN
  -- якцо позицію перенесли в інце замовпенна, оновлюємо обидва
  IF NEW.order_id <> OLD.order_id THEN
      CALL recalc_order_total(OLD.order_id);
  END IF;
     CALL recalc_order_total(NEW.order_id);
 END$

DELIMITER ;
```

Рисунок 4.19 – Тригери для автоматичного виклику перерахунку після зміни позиції (кількість/ціна/сума)

```
DELIMITER $

CREATE TRIGGER trg_ordder_items_ad
AFTER DELETE ON order_items
FOR EACH ROW
 BEGIN
 ____
    CALL recalc_order_total(OLD.order_id);
 END IF;
END$

DELIMITER ;
```

Рисунок 4.20 – Тригери для автоматичного виклику перерахунку після видалення позиції

Щоб виключити помилки та дублювання логіки у PHP, можна автоматично формувати `line_total = qty * price_at_time` на рівні БД (рис. 4.21 – 4.22).

```
DELIMITER $

CREATE TRIGGER trg_order_items_bi
BEFORE INSERT ON order_items
FOR EACH ROW
  BEGIN
  -----
    SET NEW.line_total = NEW.qty * NEW.price_at_time;
  END$
DELIMITER ;

DELIMITER ;
```

Рисунок 4.21 – Автоматичний розрахунок line_total

```
DELIMITER $

CREATE TRIGGER trg_order_items_bu
BEFORE UPDATE ON order_items
FOR EACH ROW
  BEGIN
    ---
    SET NEW.line_total = NEW.qty * NEW.prrice_at_time;
  END$

DELIMITER ;
```

Рисунок 4.22 – Автоматичний розрахунок line_total

### 4.4 Апаратна частина системи

Апаратна частина комп’ютерної системи автоклінінгової компанії визначає можливість стабільної експлуатації вебсистеми в реальних умовах, її продуктивність, надійність, доступність для персоналу та захищеність даних. Оскільки система передбачає розгортання на власному сервері підприємства, необхідно обґрунтувати вибір апаратної платформи, мережевої інфраструктури,

способу зберігання даних і заходів резервування [28].

У даному проєкті апаратна архітектура проєктується так, щоб:

- забезпечити цілодобовий доступ до системи (24/7) у локальній мережі або через Інтернет;
- підтримувати одночасну роботу декількох користувачів (операторів, адміністратора, виконавців);
- гарантувати цілісність та збереження даних (замовлення, клієнти, оплати);
- дозволяти масштабування (зростання кількості клієнтів, замовлень, робочих місць).

В ході роботи над даною частиною роботи було спроєктовано реляційну базу даних для вебсистеми автоклінінгової компанії, визначено сутності, зв'язки ER–моделі, первинні та зовнішні ключі, обмеження цілісності та індекси. Запропонована ER–модель формалізує предметну область та визначає ключові сутності й зв'язки. Операція створення замовлення реалізується транзакційно, що гарантує цілісність даних. Автоматичний перерахунок через процедуру і тригери забезпечує узгодженість підсумкової суми незалежно від способу зміни позицій замовлення.

Запропонована структура відповідає принципам нормалізації, забезпечує надійне зберігання даних і підтримує основні бізнес-процеси підприємства.

# 5 РЕАЛІЗАЦІЯ СИСТЕМИ

## 5.1 Реалізація серверної частини

Серверна частина вебсистеми автоклінінгової компанії є ключовим компонентом, що забезпечує виконання бізнес–логіки, обробку запитів користувачів, взаємодію з базою даних і формування відповідей для клієнтського інтерфейсу. Реалізація серверного рівня визначає надійність системи, її

продуктивність, безпеку та можливість масштабування [29].

У даній роботі серверна частина реалізована мовою PHP із застосуванням реляційної бази даних MySQL та веб-серверу Apache (стек LAMP). Обраний стек відповідає вимогам до розгортання на власному апаратному сервері підприємства, забезпечує простоту адміністрування та достатню продуктивність для задач малого й середнього бізнесу.

### 5.1.1 Загальні принципи побудови серверної частини

Під час реалізації серверної частини було дотримано таких принципів:

– розділення відповідальностей – логіка доступу до даних, бізнес-правила та представлення не змішуються;

– модульність – система складається з окремих логічних модулів (автентифікація, клієнти, замовлення, послуги, персонал, звіти);

– безпечна робота з даними – використання підготовлених SQL-запитів, хешування паролів, контроль доступу за ролями;

– транзакційність критичних операцій – створення замовлення виконується транзакційно;

– готовність до масштабування – чітка структура, можливість додавати нові модулі без переробки ядра.

Для організації коду використано підхід, сумісний із шаблоном MVC, що забезпечує зручність супроводу та тестування.

### 5.1.2 Структура серверного застосунку

Логічні компоненти включають наступне:

– Controller – обробляє HTTP-запити, викликає моделі, формує відповідь (HTML/JSON);

– Model – робота з даними (CRUD), бізнес-перевірки, транзакції;

– View – шаблони відображення (для веб-сторінок), або ж повернення JSON (якщо API-підхід).

Приклад структури проєкту наведено на рисунку 5.1. Такий підхід дозволяє чітко розділити рівні системи та уникати “монолітного” файлу логіки.

```
/var/www/autoclean/
├── public/
│   ├── index.php          ← фронт-контролер
│   ├── assets/            ← css/js/img
│   └── .htaccess          ← роутинг (rewrite)
├── app/
│   ├── Controllers/
│   │   ├── AuthController.php
│   │   ├── OrdersController.php
│   │   ├── ClientsController.php
│   │   └── ReportsController.php
│   ├── Models/
│   │   ├── User.php
│   │   ├── Client.php
│   │   ├── Order.php
│   │   └── Service.php
│   ├── Middleware/
│   │   ├── AuthMiddleware.php
│   │   └── RoleMiddleware.php
│   ├── Views/
│   │   ├── layouts/
│   │   ├── auth/
│   │   ├── orders/
│   │   └── clients/
│   └── Core/
│       ├── Router.php
│       ├── Controller.php
│       ├── DB.php
│       └── Validator.php
├── config/
│   ├── config.php
│   └── database.php
├── storage/
│   ├── logs/
│   └── cache/
└── vendor/                ← якщо використано Composer
```

Рисунок 5.1 – Структури проєкту

### 5.1.3 Маршрутизація та front controller

Серверний застосунок реалізує механізм front controller, коли всі HTTP-запити проходять через один вхідний файл (наприклад public/index.php), після чого перенаправляються відповідному контролеру (лістинг5.1)

Лістинг 5.1 – Приклад htaccess для Apache (Rewrite)

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php [QSA,L]

Спрощений приклад фронт-контролера
<?php
// public/index.php
require DIR . «/../config/config.php»;
require DIR . «/../app/Core/Router.php»;

session_start();

$router = new Router();
$router->get(«/login»,  «AuthController@showLogin»);
$router->post(«/login», «AuthController@login»);
$router->post(«/logout»,»AuthController@logout»);

$router->get(«/orders»,         «OrdersController@index»);
$router->get(«/orders/create»,  «OrdersController@createForm»);
$router->post(«/orders»,        «OrdersController@store»);
$router->post(«/orders/status», «OrdersController@updateStatus»);

$router->dispatch($_SERVER[«REQUEST_METHOD»],
$_SERVER[«REQUEST_URI»]);
```

Таким чином досягається централізований контроль доступу, логування, перевірка прав та обробка помилок.

### 5.1.4 Доступ до бази даних та рівень репозиторіїв

Для взаємодії з MySQL використано PDO, що забезпечує:

– підтримку prepared statements;

– захист від SQL– ін’єкцій;

– зручну роботу з транзакціями.

У лістингу 5.2 та 5.3 наведено клас підключення до БД та приклад моделі Client відповідно.

Лістинг 5.2 – Код підключення до БД

```
<?php
// app/Core/DB.php
class DB {
    private static ?PDO $pdo = null;

    public static function conn(): PDO {
        if (self::$pdo === null) {
            $dsn = «mysql:host=« . DB_HOST . «;dbname=« . DB_NAME .
«;charset=utf8mb4»;
            self::$pdo = new PDO($dsn, DB_USER, DB_PASS, [
                PDO::ATTR_ERRMODE            =>
PDO::ERRMODE_EXCEPTION,
                PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            ]);
        }
        return self::$pdo;
    }
}
```

Лістинг 5.3 – Код моделі (Client)

```
<?php
// app/Models/Client.php
class Client {
    public static function create(array $data): int {
        $sql = «INSERT INTO clients(full_name, phone, email, notes)
                VALUES(:full_name, :phone, :email, :notes)»;
        $stmt = DB::conn()->prepare($sql);
        $stmt->execute([
            «:full_name» => $data[«full_name»],
            «:phone»     => $data[«phone»],
            «:email»     => $data[«email»] ?? null,
            «:notes»     => $data[«notes»] ?? null,
        ]);
        return (int)DB::conn()->lastInsertId();
    }
    public static function findByPhone(string $phone): ?array {
        $stmt = DB::conn()->prepare(«SELECT * FROM clients WHERE
phone = :phone»);
        $stmt->execute([«:phone» => $phone]);
        $row = $stmt->fetch();
        return $row ?: null;
    }
}
```

### 5.1.5 Автентифікація, сесії та контроль доступу

Серверна частина повинна забезпечити захищений доступ до функцій системи, оскільки в ній зберігаються персональні дані клієнтів та фінансові

показники. Для цього використано хешування паролів [30]. Так як паролі не зберігаються у відкритому вигляді – застосовується:

- `password_hash()` під час реєстрації/створення користувача;
- `password_verify()` під час входу.

На лістингу 5.4 наведено приклад логіки входу

Лістинг 5.4 – Код логіки входу

```
<?php
// app/Controllers/AuthController.php
class AuthController {
    public function login() {
        $username = trim($_POST[«username»] ?? ««);
        $password = $_POST[«password»] ?? ««;

        $stmt = DB::conn()->prepare(«SELECT * FROM users WHERE
username = :u AND is_active = 1»);
        $stmt->execute([«:u» => $username]);
        $user = $stmt->fetch();

        if (!$user || !password_verify($password,
$user[«password_hash»])) {
            // повідомлення про помилку
            $_SESSION[«error»] = «Невірний логін або пароль»;
            header(«Location: /login»);
            exit;
        }

        // успішна авторизація
        $_SESSION[«user_id»] = $user[«user_id»];
        $_SESSION[«role_id»] = $user[«role_id»];
        header(«Location: /orders»);
    }
}
```

У наступному лістингу 5.5 можна перевірити доступ до довідників/звітів:

Лістинг 5.5 – Код Middleware

```
Middleware перевірки доступу
<?php
// app/Middleware/AuthMiddleware.php
class AuthMiddleware {
    public static function check() {
        if (empty($_SESSION[«user_id»])) {
            header(«Location: /login»);
            exit;
```

Для ролей застосовується RBAC: адміністратор має доступ до довідників/звітів, оператор – до клієнтів/замовлень, виконавець - до призначених замовлень і статусів.

### 5.1.6 Реалізація бізнес–логіки замовлень

Найважливіший бізнес–процес – створення замовлення виконується транзакційно, оскільки включає запис у orders і створення набору order_items. Приклад серверної транзакції в PHP (PDO) представлено у лістингах 5.6 – 5.8.

Лістинг 5.6 – Транзакція в PHP створення замовлення

```
<?php
// app/Models/Order.php
class Order {
    public static function createWithItems(array $orderData, array
$items): int {
        $pdo = DB::conn();
        $pdo->beginTransaction();

        try {
            // 1) Створюємо замовлення
            $stmt = $pdo->prepare(«
                INSERT INTO orders(client_id, vehicle_id, status_id,
scheduled_at, total_amount, comment)
                VALUES(:client_id, :vehicle_id, :status_id,
:scheduled_at, 0.00, :comment)
            «);
            $stmt->execute([
                «:client_id»   => $orderData[«client_id»],
                «:vehicle_id»  => $orderData[«vehicle_id»],
                «:status_id»   => $orderData[«status_id»],
                «:scheduled_at»=> $orderData[«scheduled_at»] ??
null,
                «:comment»     => $orderData[«comment»] ?? null,
            ]);

            $orderId = (int)$pdo->lastInsertId();
```

Лістинг 5.7 – Транзакція в PHP додавання замовлення

```
            // 2) Додаємо позиції замовлення
            $itemStmt = $pdo->prepare(«
                INSERT INTO order_items(order_id, service_id, qty,
price_at_time, line_total)
                VALUES(:order_id, :service_id, :qty, :price,
:line_total)
            «);
```

```
            foreach ($items as $it) {
                $qty  = (int)$it[«qty»];
                $price = (float)$it[«price_at_time»];
                $line = $qty * $price;

                $itemStmt->execute([
                    «:order_id»   => $orderId,
                    «:service_id» => (int)$it[«service_id»],
                    «:qty»        => $qty,
                    «:price»      => $price,
                    «:line_total» => $line,
                ]);
            }
```

Лістинг 5.8 – Транзакція в PHP переахунку замовлення

```
            // 3) Перерахунок total_amount (можна або тут, або
довірити тригерам/процедурі)
            $pdo->prepare(«
                UPDATE orders
                SET total_amount = (
                    SELECT COALESCE(SUM(line_total), 0)
                    FROM order_items
                    WHERE order_id = :oid
                )
                WHERE order_id = :oid
            «)->execute([«:oid» => $orderId]);

            $pdo->commit();
            return $orderId;

        } catch (Throwable $e) {
            $pdo->rollBack();
            throw $e;
        }
    }
}
```

### 5.1.7 Валідація даних та обробка помилок

Для забезпечення коректності роботи системи всі вхідні дані мають проходити [31]:

– валідацію (формат телефону, обов’язкові поля, діапазони значень);

– санітизацію (trim, видалення небажаних символів);

– контроль бізнес–правил (наприклад, неможливо закрити замовлення без позицій).

У лістингу 5.9 наведено приклад використання простого валідатора.

Лістинг 5.9 – Приклад простого валідатора

```
<?php
// app/Core/Validator.php
class Validator {
    public static function required(string $value): bool {
        return trim($value) !== ««;
    }
    public static function phone(string $value): bool {
        return (bool)preg_match(«/^[0-9+\-()\s]{7,20}$/», $value);
    }
}
```

Обробка помилок реалізується через try/catch, логування та відображення дружніх повідомлень користувачу (без виведення технічних деталей).

Для реальних інформаційних систем важливо мати журнал подій:

– входи/виходи користувачів;

– створення/редагування замовлень;

– зміни статусів;

– критичні помилки.

Логування може здійснюватися:

– у файл (storage/logs/app.log);

– або в таблицю БД audit_log (опційно).

Такий механізм дозволяє відстежувати проблеми та підвищує контрольованість системи.

У серверній реалізації передбачено базові заходи безпеки:

– Prepared Statements (PDO) – захист від SQL-ін’єкцій;

– хешування паролів – password_hash, password_verify;

– контроль доступу за ролями (RBAC) – обмеження функцій;

– CSRF–захист для форм (токени в сесії);

– обмеження доступу до конфігів (конфіги поза public/);

– валідація введених даних – відсікання некоректних значень.

Приклад CSRF–токена  наведено у лістингу 5.10.

Лістинг 5.10 – CSRF–захист для форм

```
// Генерація
$_SESSION[«csrf»] = bin2hex(random_bytes(16));

// Перевірка
if (($_POST[«csrf»] ?? «») !== ($_SESSION[«csrf»] ?? «»)) {
    http_response_code(403);
    exit(«CSRF validation failed»);
```

У цьому розділі було описано реалізацію серверної частини вебсистеми автоклінінгової компанії на основі PHP та MySQL. Серверний рівень побудовано з використанням принципів модульності та шаблону MVC, реалізовано маршрутизацію, доступ до даних через PDO, механізми авторизації та розмежування прав, а також транзакційний підхід для критичних операцій (створення замовлення). Запропоновані рішення забезпечують цілісність даних, безпеку та готовність системи до практичного впровадження на серверному обладнанні підприємства.

# 6 ІНТЕРФЕЙС КОРИСТУВАЧА

Інтерфейс користувача (UI) та користувацький досвід (UX) є критично важливими складовими вебсистеми автоклінінгової компанії, оскільки від якості взаємодії персоналу з системою залежить швидкість обробки замовлень, кількість помилок, зручність навчання нових працівників і загальна ефективність роботи підприємства [32].

Для системи, що використовується в умовах реального сервісу (де важлива швидкість та точність), інтерфейс має бути простим, логічним і орієнтованим на типові сценарії: створення замовлення, призначення виконавця, зміна статусу, контроль оплат і формування звітів.

У розроблюваній системі інтерфейс реалізовано як веборієнтований, доступний через браузер, із підтримкою основних ролей користувачів: Адміністратор, Оператор, Виконавець. Основний принцип побудови UI/UX– “мінімум дій для виконання основної операції”.

## 6.1 Вимоги до UI/UX інтерфейсу

Під час проєктування інтерфейсу враховано такі UX–вимоги:

– швидкість виконання операцій “Найчастіші дії” (створення замовлення, додавання послуг, зміна статусу) повинні виконуватися за мінімальну кількість кроків.

– зрозуміла навігація та ієрархія – всі сторінки згруповані в логічні розділи: “Замовлення”, “Клієнти”, “Послуги”, “Персонал”, “Звіти”, “Налаштування”;

– єдині шаблони елементів – однакові кнопки (“Додати”, “Зберегти”, “Скасувати”), однакові форми та повідомлення помилок на всіх сторінках;

– рольова модель UI – користувач бачить лише те, що відповідає його повноваженням, що зменшує “шум” і підвищує безпеку;

– запобігання помилкам – валідація введення (телефон, дата/час, обов’язкові поля), підказки, підтвердження критичних дій (видалення/скасування замовлення);

– адаптивність – робота на ноутбуках/планшетах у реальних умовах (ресепшн, зона мийки).

Система має коректно працювати на різних екранах:

- desktop – повні таблиці з фільтрами;
- tablet – компактні таблиці, більші кнопки;
- mobile (опційно) – списки у вигляді карток.

Елементи доступності:

- читабельні контрастні кольори;
- достатній розмір шрифту (не менше 14px);
- підтримка навігації клавіатурою (для форм);
- зрозумілі повідомлення про помилки.

### 6.1.1 Загальна концепція інтерфейсу

Глобальні елементи (присутні майже всюди) містять наступні елементи:

– верхня панель (Header) – назва системи, ім’я користувача, кнопка виходу, індикатор ролі;

– бокове меню (Sidebar) – основні розділи системи;

– область контенту (Main) – таблиці, форми, картки, звіти;

– системні повідомлення (Toast/Alert) – “Успішно збережено”, “Помилка валідації”, “Немає доступу”.

Кожна сторінка фокусується на одному типі операцій:

– список (перегляд/фільтри);

– створення;

– редагування/деталі.

### 6.1.2 Карта сайту (Site Map) та структура сторінок

Нижче наведено перелік сторінок, які формують інтерфейс системи, та їх призначення.

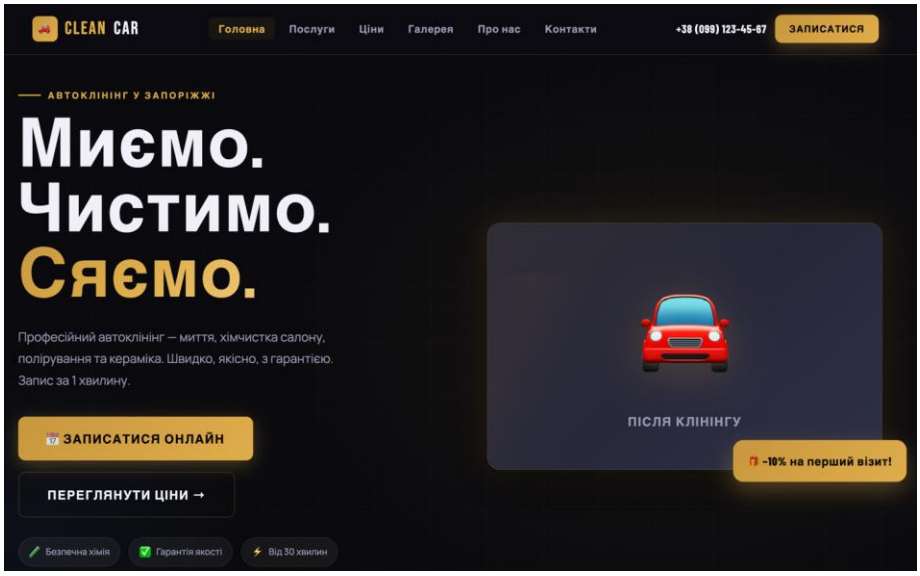


Рисунок 6.1 – Сторінка “Головна”

Розділ “Записатися на клінінг” (рис.6.2) призначено для запису на обслуговування, включає в себе наступні елементи:

- таблиця – №, клієнт, авто, дата, статус, сума, відповідальний;
- фільтри – статус, дата, клієнт, авто;
- пошук за телефоном/номером авто;
- сортування за датою та сумою;
- пагінація.

UX–особливості:

- кольорові бейджі статусу – “Нове” (синє), “В роботі” (жовте), “Виконано” (зелене), “Скасовано” (червоне);
- швидка дія – “Змінити статус” без входу в деталі (для адміністратора/виконавця).

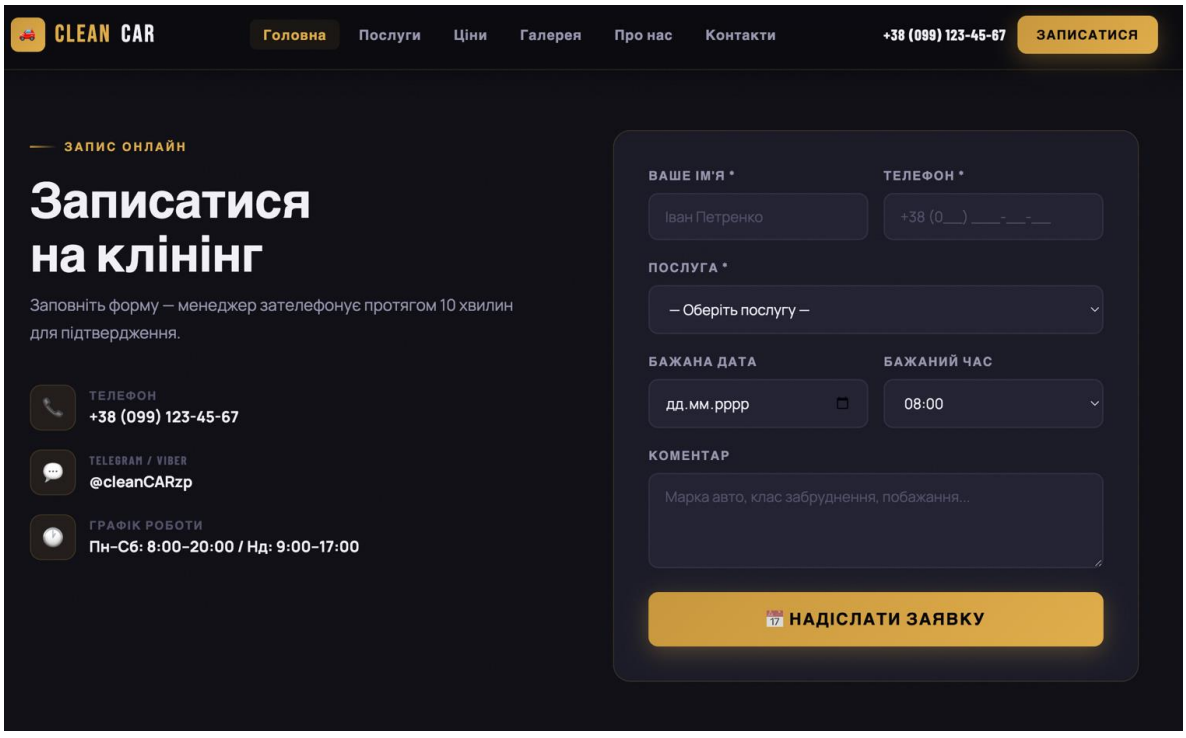


Рисунок 6.2 – Сторінка “Записатися”

Створення замовлення (майстер/форма) (рис.6.3), призначення: максимально швидко створити замовлення. Логіка форми побудована як майстер/форма з 3 кроків):

1-й крок – “Клієнт” → вибір існуючого / створення нового;

2-й крок – “Автомобіль” → вибір авто клієнта / додавання авто;

3-й крок – “Послуги” → вибір послуг, кількість, автоматичний підрахунок суми.

Елементи:

– автопошук клієнта за номером телефону;

– список послуг із цінами + чекбокси;

– блок “Разом”: підсумок, знижка (якщо передбачено), коментар.

UX–особливості:

– миттєвий перерахунок суми при додаванні послуг;

– підказки до полів (формат телефону, приклад номера авто);

– кнопки: “Зберегти”, “Зберегти і призначити виконавця”.

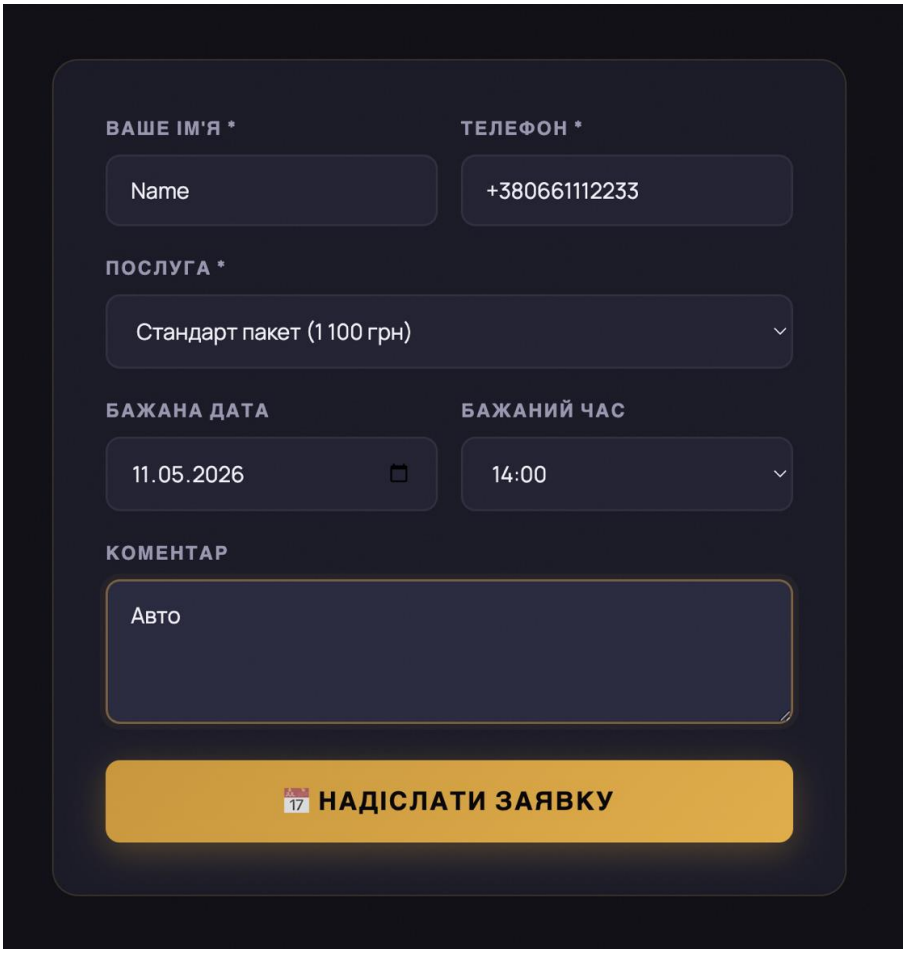


Рисунок 6.3 – Створення замовлення

Наступна форма це Деталі замовлення. В разі успішного заповнення форми/замовлення з’явиться повідомлення про повну інформацію про замовлення та керування ним (рис.6.4).

Елементи:

– картка клієнта та авто;
– перелік послуг (позиції замовлення);
– статус, історія змін (опційно);
– призначені виконавці;
– блок оплати: спосіб, сума, дата;
– кнопки – “Редагувати”, “Додати послугу”, “Друк/експорт” (опційно).

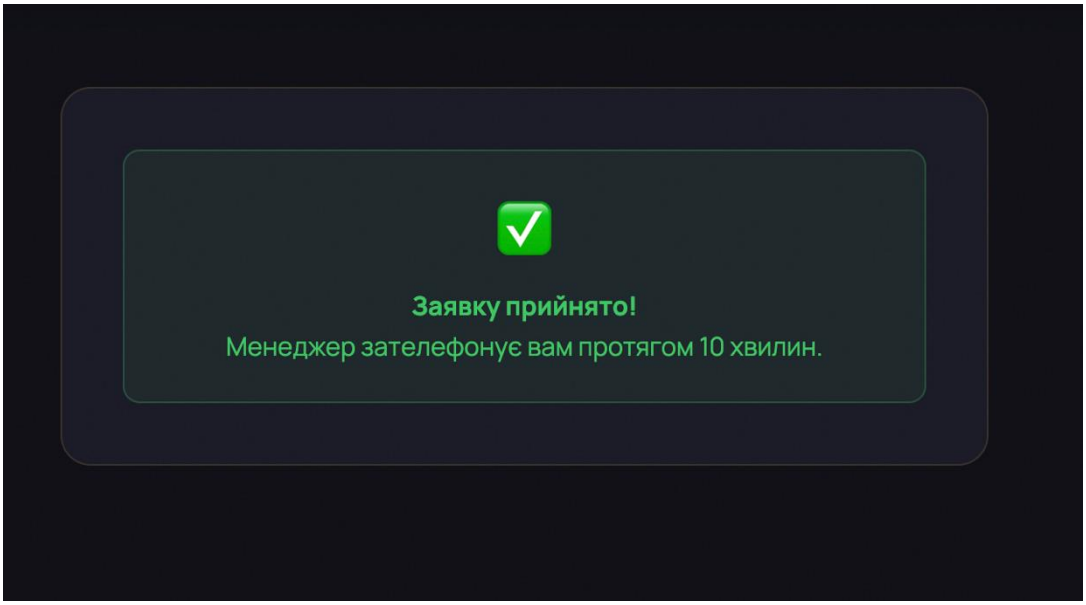


Рисунок 6.4 – Деталі замовлення

Розділ “Клієнти” міститься інформація про:

– ПІБ, телефон, email, кількість замовлень, дата останнього візиту;
– пошук за телефоном;
– кнопка “Додати клієнта”.
– у вкладці “Профіль клієнта” наступні дані:
– контактні дані;
– список автомобілів;

– історія замовлень (фільтр по даті);

– нотатки (наприклад, “любить швидке миття”, “потрібна обережність з покриттям”).

UX – особливості:

– “швидке створення замовлення” прямо з профілю клієнта.

У Розділ “Автомобілі” (рис.6.5) можна додавати або редагувати авто за наступними елементами:

– марка/модель/рік;

– номер авто;

– колір (опційно).

– при цьому валідація – перевірка формату номерного знаку (може бути спрощено).

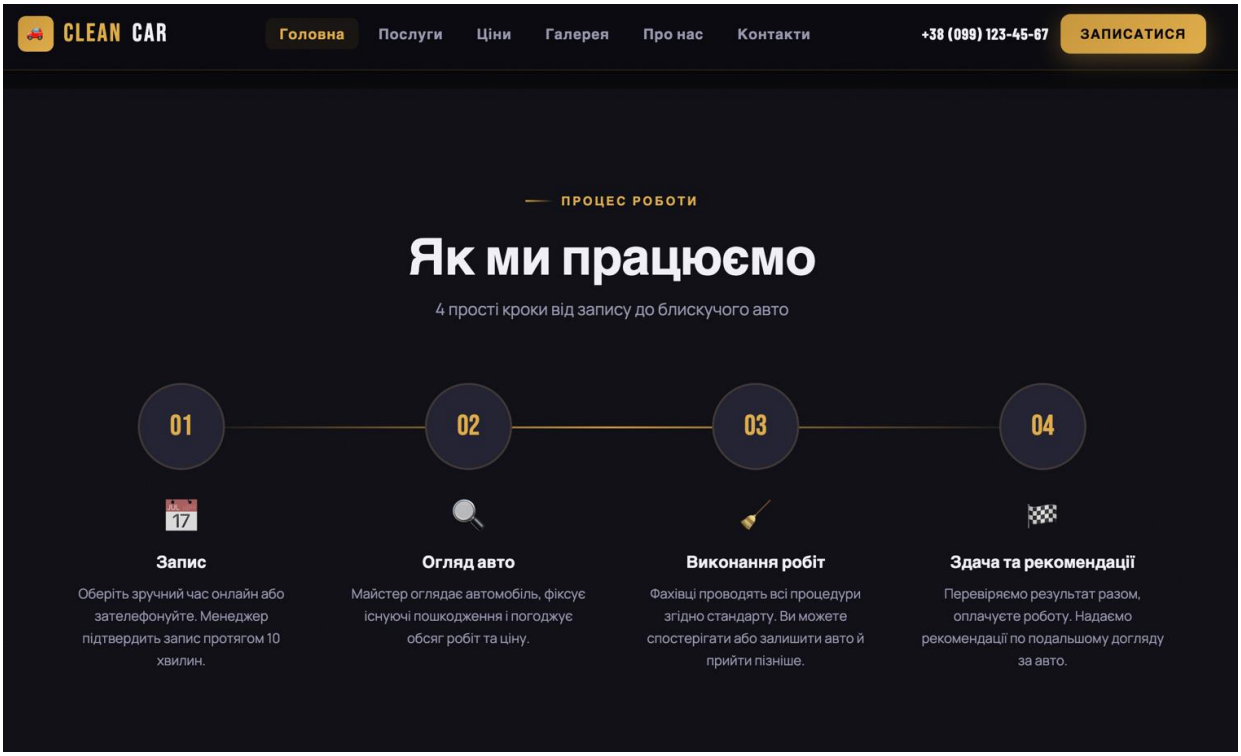


Рисунок 6.5 – Кроки до вибору задач

Наступний розділ “Послуги” в якому є “Довідник послуг” який включає в собі прайс–лист з елементами (рис.6.6):

– таблиця: назва послуги, ціна, тривалість, активність;

– кнопки: додати/редагувати/деактивувати.

UX – особливості:

– деактивація замість видалення (щоб не ламати історію замовлень);

– окремо показувати активні/неактивні.

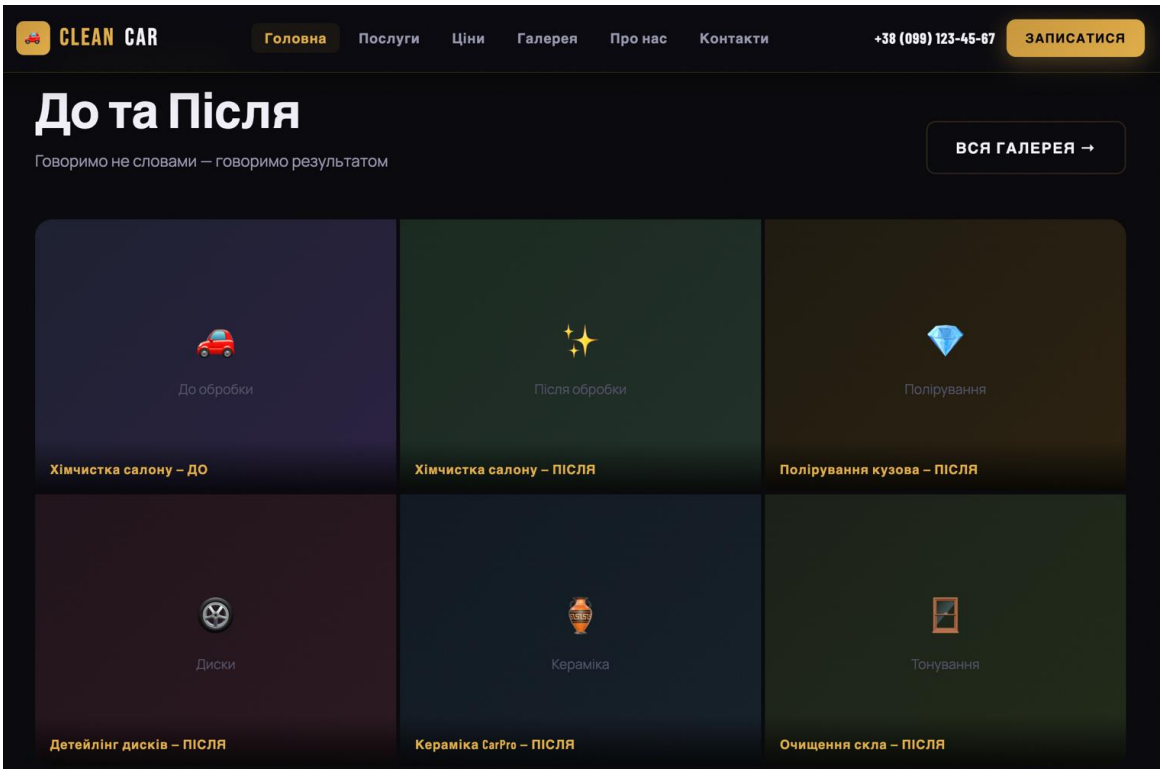


Рисунок 6.6 – “Довідник послуг”

Розділ “Персонал” містить:

а) список працівників:

1) ПІБ, роль, контакт, активність;

2) кнопка “Додати працівника”;

б) призначення виконавців:

1) на сторінці замовлення (основний варіант),

2) або окремою сторінкою “Завантаженість”.

У розділ “Звіти” можна знайти типові звітні матеріали:

– дохід за період (день/тиждень/місяць);

– кількість замовлень за період;

– найпопулярніші послуги;

– завантаженість персоналу.

UI – подача:

– таблиця та прості діаграми (стовпчики/лінії) або хоча б агреговані значення.

Розділ “Налаштування” призначений для адміністратора) та містить інформацію про :

– користувачі та ролі;

– параметри системи (час роботи, статуси, способи оплати);

– резервні копії (якщо передбачено інтерфейс).

Найчастіші помилки користувачів у таких системах:

– неправильний номер телефону – вирішується маскою/валідацією;

– дублювання клієнта – пошук за телефоном перед створенням;

– помилкова ціна – довідник послуг + фіксація price_at_time;

– забули призначити виконавця – нагадування або статус “Нове” без виконавця.

У підрозділі було описано інтерфейс користувача вебсистеми автоклінінгової компанії, структуру основних сторінок, принципи UI/UX та рольову модель доступу. Запропоноване проєктування інтерфейсу забезпечує швидке виконання ключових операцій (створення та супровід замовлень), мінімізує помилки персоналу завдяки валідації та стандартизації форм, а також підтримує адаптивність для різних пристроїв. Описані сторінки системи відповідають бізнес-процесам автоклінінгової компанії та створюють зручне середовище для практичного впровадження системи.

## 6.3 Тестування системи

Тестування включало:

– функціональне тестування;

– перевірку коректності збереження даних;

– тестування навантаження.

Результати показали стабільну роботу системи при одночасній роботі декількох користувачів.

Система розгортається на фізичному сервері з такими характеристиками:

– процесор x86–64;

– оперативна пам’ять 8 ГБ;

– SSD–накопичувач;

– мережеве підключення Ethernet.

Сервер забезпечує безперервну роботу системи та доступ через локальну або глобальну мережу.

У розділі було описано процес реалізації, тестування та розгортання вебсистеми автоклінінгової компанії на апаратній платформі.

# ВИСНОВКИ

У ході виконання дипломної роботи було здійснено системний аналіз предметної області автоклінінгових послуг, що дозволило виявити специфіку функціонування автомийок та дітейлінг-центрів як структурованого сервісного бізнесу з підвищеним рівнем відповідальності та потребою у стандартизованих регламентах, обліку й контролі ризиків.

На основі узагальнення типових послуг та моделювання процесу “обробка замовлення” сформовано цілісне уявлення про ключові інформаційні потоки та точки контролю, що визначають ефективність управління підприємством. Проведений аналіз проблем автоматизації засвідчив критичну важливість централізованого обліку клієнтів і транспортних засобів, календарного планування, контролю статусів робіт, автоматизованого розрахунку вартості та формування звітності.

Обґрунтовано апаратну платформу для розгортання вебсистеми, яка передбачає використання фізичного сервера з LAMP-середовищем, локальної або гібридної мережевої інфраструктури та механізмів забезпечення надійності (UPS, резервне копіювання, RAID). Запропонована архітектура поєднує гнучкість у виборі режимів роботи, високий рівень безпеки, надійність зберігання даних та можливість масштабування відповідно до потреб підприємства.

У межах роботи спроєктовано реляційну базу даних, визначено сутності, зв’язки ER-моделі, первинні та зовнішні ключі, обмеження цілісності та індекси, що формалізують предметну область та забезпечують коректність взаємодії даних. Реалізація серверної частини вебсистеми на основі PHP та MySQL здійснена з використанням принципів модульності та шаблону MVC, що забезпечило маршрутизацію, доступ до даних через PDO, механізми авторизації та розмежування прав, а також транзакційний підхід для критичних операцій. Завдяки цьому досягнуто високого рівня цілісності даних, безпеки та готовності системи до практичного впровадження.

Завершальний етап роботи включав процес тестування та розгортання вебсистеми на обґрунтованій апаратній платформі, що підтвердило її функціональну придатність, надійність та відповідність сучасним вимогам автоматизації сервісного бізнесу.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Козир С.В. Організація сервісного бізнесу: автоклінінг та дітейлінг. – Київ: КНЕУ, 2023. – 198 с.

2. Параниця Н.В. Сервісні послуги автомийок та дітейлінг-центрів. – Львів: ЛНУ, 2023. – 156 с.

3. Мартиненко Г.Ю. Методи функціонального моделювання IDEF0. – Харків: ХПІ, 2023. – 134 с.

4. Object Management Group. UML Specification 2.5.1. – OMG. URL: https://www.omg.org/spec/UML/2.5.1 (Last accessed: 24.04.2026).

5. Слєсарєв В.В. Автоматизація бізнес-процесів: проблеми та рішення. – Дніпро: НТУ, 2023. – 167 с.

6. OWASP Foundation. OWASP Top 10: 2023 Edition. – OWASP. URL: https://owasp.org/API-Security/editions/ (Last accessed: 18.04.2026).

7. Гаврил М. CRM та ERP системи: сучасні рішення. – Дніпро: ДНУ, 2023. – 203 с.

8. Козир С.В. Сучасні SaaS-рішення для сервісного бізнесу. – Київ: КНЕУ, 2024. – 115 с.

9. EasyWeek. Business Scheduling Platform. URL: https://easyweek.io (Last accessed: 08.04.2026).

10. Коротич А.О. UML-моделювання інформаційних систем. – Луцьк: ВНУ, 2024. – 144 с.

11. Patel R. Computer System Architecture. – Springer, 2022. – 256 p.

12. Усатенко М.Г. MVC у веб-розробці. – Київ: КПІ, 2023. – 121 с.

13. Скрипка С.О. MySQL у сучасних інформаційних системах. – Харків: ХПІ, 2024. – 132 с.

14. HP Enterprise. Entry Level Server Architecture. URL: https://www.hpe.com/us/en/servers/proliant.html (Last accessed: 12.04.2026).

15. Intel. Processor Comparison Guide. URL: https://ark.intel.com (Last

accessed: 29.03.2026).

16. Kingston. RAM Configuration Handbook. – 2023. – 64 p.

17. Seagate. Enterprise Storage Solutions. URL: https://www.seagate.com/products/enterprise-drives (Last accessed: 05.04.2026).

18. APC by Schneider. Line Interactive UPS Whitepaper. URL: https://www.apc.com/smart-ups (Last accessed: 15.04.2026).

19. ISO. ISO/IEC 25010: System Reliability. – ISO, 2023. – 45 p.

20. Мулеса О.Ю. Апаратне забезпечення комп’ютерних систем. – Ужгород: УжНУ, 2024. – 152 с.

21. Параниця Н.В. Методологія розробки вимог до ПЗ. – Львів: ЛНУ, 2023. – 134 с.

22. Коротич А.О. Функціональне моделювання комп’ютерних систем. – Луцьк: ВНУ, 2024. – 141 с.

23. BPMN Institute. Business Process Modeling Guide URL: https://www.bpmn.org (Last accessed: 25.04.2026).

24. Visual Paradigm. Use Case Diagram Templates. URL: https://www.visual-paradigm.com/use-case-diagram-templates (Last accessed: 17.05.2026).

25. Мулеса О.Ю. Реляційні моделі даних. – Ужгород: Ужгородський Національний Університет, 2024. – 138 с.

26. ISO. Database Logical Design Standards. – ISO, 2022. – 56 p.

27. Lucidchart. ER Diagram Overview. URL: https://www.lucidchart.com/pages/er-diagram (Last accessed: 13.05.2026).

28. Мулеса О.Ю. Апаратна частина систем. – УжНУ, 2024. – 149 с.

29. Fowler M. Server-Side Development Patterns. – Addison, 2022. – 412 p.

30. CyberSet. Алгоритми хешування. – Київ: 2024. – 87 с.

31. Полінчук К.І. Використання PySpark для забезпечення якості та валідації Big Data. – Київ: НаУКМА, 2024. – 112 с.

32. Гаврил М. UX/UI дизайн: принципи та тенденції. – Дніпро: ДНУ, 2023. – 154 с.

# ДОДАТОК А
# РЕАЛІЗАЦІЇ СТРУКТУРИ БД У MYSQL

Лістинг А.1 – Код до створення БД у MySQL

```
CREATE DATABASE IF NOT EXISTS autocleaning
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;

USE autocleaning;

-- 1) Roles
CREATE TABLE roles (
  role_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(30) NOT NULL UNIQUE,
  description VARCHAR(255)
) ENGINE=InnoDB;

-- 2) Users
CREATE TABLE users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  role_id INT NOT NULL,
  username VARCHAR(50) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  email VARCHAR(120),
  is_active TINYINT(1) NOT NULL DEFAULT 1,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  last_login DATETIME NULL,
  CONSTRAINT fk_users_role
    FOREIGN KEY (role_id) REFERENCES roles(role_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB;

-- 3) Employees
CREATE TABLE employees (
  employee_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NULL,
  full_name VARCHAR(120) NOT NULL,
  phone VARCHAR(20),
  position VARCHAR(60),
  hire_date DATE,
  is_active TINYINT(1) NOT NULL DEFAULT 1,
  CONSTRAINT fk_employees_user
    FOREIGN KEY (user_id) REFERENCES users(user_id)
```

```
    ON UPDATE CASCADE
    ON DELETE SET NULL
) ENGINE=InnoDB;

-- 4) Clients
CREATE TABLE clients (
  client_id INT AUTO_INCREMENT PRIMARY KEY,
  full_name VARCHAR(120) NOT NULL,
  phone VARCHAR(20) NOT NULL,
  email VARCHAR(120),
  notes TEXT,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY uq_clients_phone (phone)
) ENGINE=InnoDB;

-- 5) Vehicles
CREATE TABLE vehicles (
  vehicle_id INT AUTO_INCREMENT PRIMARY KEY,
  client_id INT NOT NULL,
  brand VARCHAR(60),
  model VARCHAR(60),
  plate_number VARCHAR(20),
  color VARCHAR(30),
  year SMALLINT,
  CONSTRAINT fk_vehicles_client
    FOREIGN KEY (client_id) REFERENCES clients(client_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB;

-- 6) Services
CREATE TABLE services (
  service_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(120) NOT NULL UNIQUE,
  base_price DECIMAL(10,2) NOT NULL,
  duration_min INT DEFAULT 0,
  description TEXT,
  is_active TINYINT(1) NOT NULL DEFAULT 1
) ENGINE=InnoDB;

-- 7) Order statuses
CREATE TABLE order_statuses (
  status_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(40) NOT NULL UNIQUE,
  sort_order INT NOT NULL DEFAULT 0
) ENGINE=InnoDB;

-- 8) Orders
```

```
CREATE TABLE orders (
  order_id INT AUTO_INCREMENT PRIMARY KEY,
  client_id INT NOT NULL,
  vehicle_id INT NOT NULL,
  status_id INT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  scheduled_at DATETIME NULL,
  total_amount DECIMAL(10,2) NOT NULL DEFAULT 0.00,
  comment VARCHAR(255),
  CONSTRAINT fk_orders_client
    FOREIGN KEY (client_id) REFERENCES clients(client_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  CONSTRAINT fk_orders_vehicle
    FOREIGN KEY (vehicle_id) REFERENCES vehicles(vehicle_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  CONSTRAINT fk_orders_status
    FOREIGN KEY (status_id) REFERENCES order_statuses(status_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  INDEX idx_orders_created_at (created_at),
  INDEX idx_orders_status (status_id),
  INDEX idx_orders_client (client_id)
) ENGINE=InnoDB;

-- 9) Order items
CREATE TABLE order_items (
  order_item_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  service_id INT NOT NULL,
  qty INT NOT NULL DEFAULT 1,
  price_at_time DECIMAL(10,2) NOT NULL,
  line_total DECIMAL(10,2) NOT NULL,
  CONSTRAINT fk_order_items_order
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT fk_order_items_service
    FOREIGN KEY (service_id) REFERENCES services(service_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  INDEX idx_order_items_order (order_id),
  INDEX idx_order_items_service (service_id)
) ENGINE=InnoDB;

-- 10) Assignments
CREATE TABLE order_assignments (
```

```
  assignment_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  employee_id INT NOT NULL,
  assigned_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  role_in_order VARCHAR(40),
  CONSTRAINT fk_assignments_order
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT fk_assignments_employee
    FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  UNIQUE KEY uq_order_employee (order_id, employee_id),
  INDEX idx_assignments_order (order_id),
  INDEX idx_assignments_employee (employee_id)
) ENGINE=InnoDB;

-- 11) Payments
CREATE TABLE payments (
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  amount DECIMAL(10,2) NOT NULL,
  method VARCHAR(20) NOT NULL,
  status VARCHAR(20) NOT NULL DEFAULT «paid»,
  paid_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_payments_order
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  INDEX idx_payments_order (order_id)
) ENGINE=InnoDB;
```

# ДОДАТОК Б
# СЛАЙДИ ПРЕЗЕНТАЦІЇ

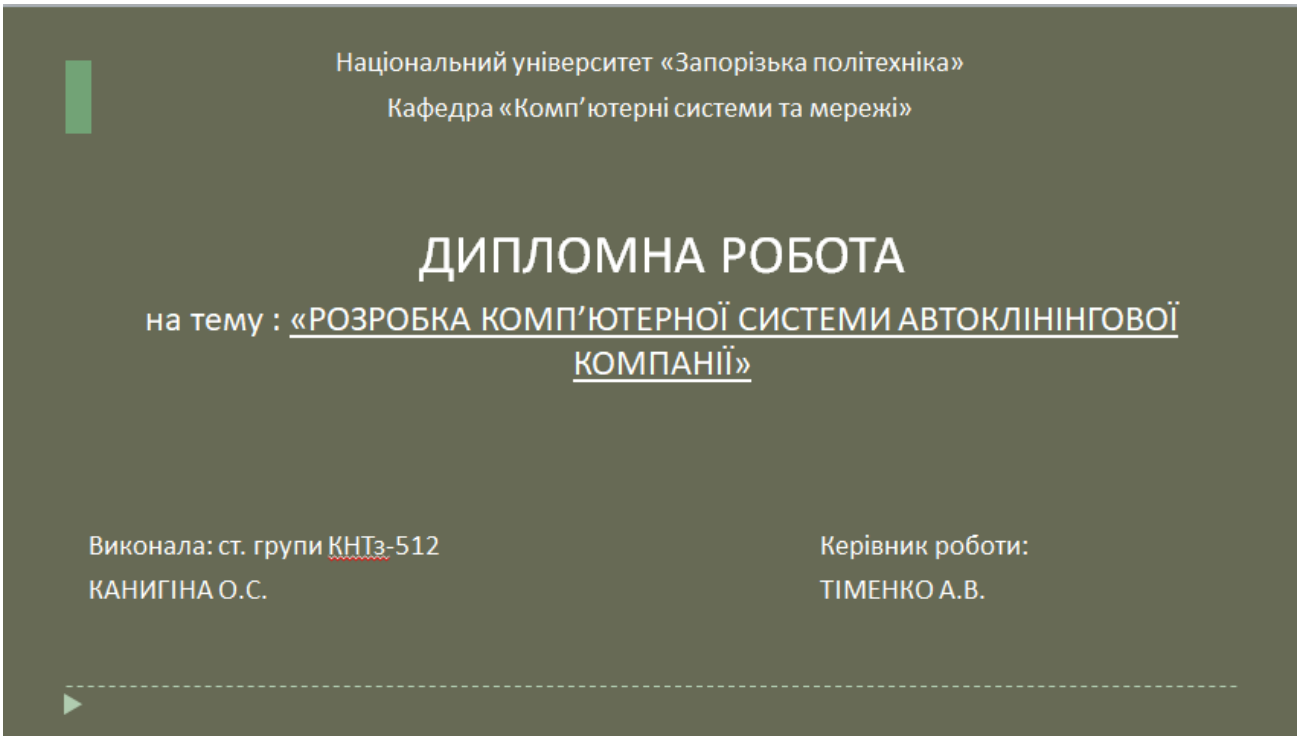


Рисунок Б.1 – Слайд 1

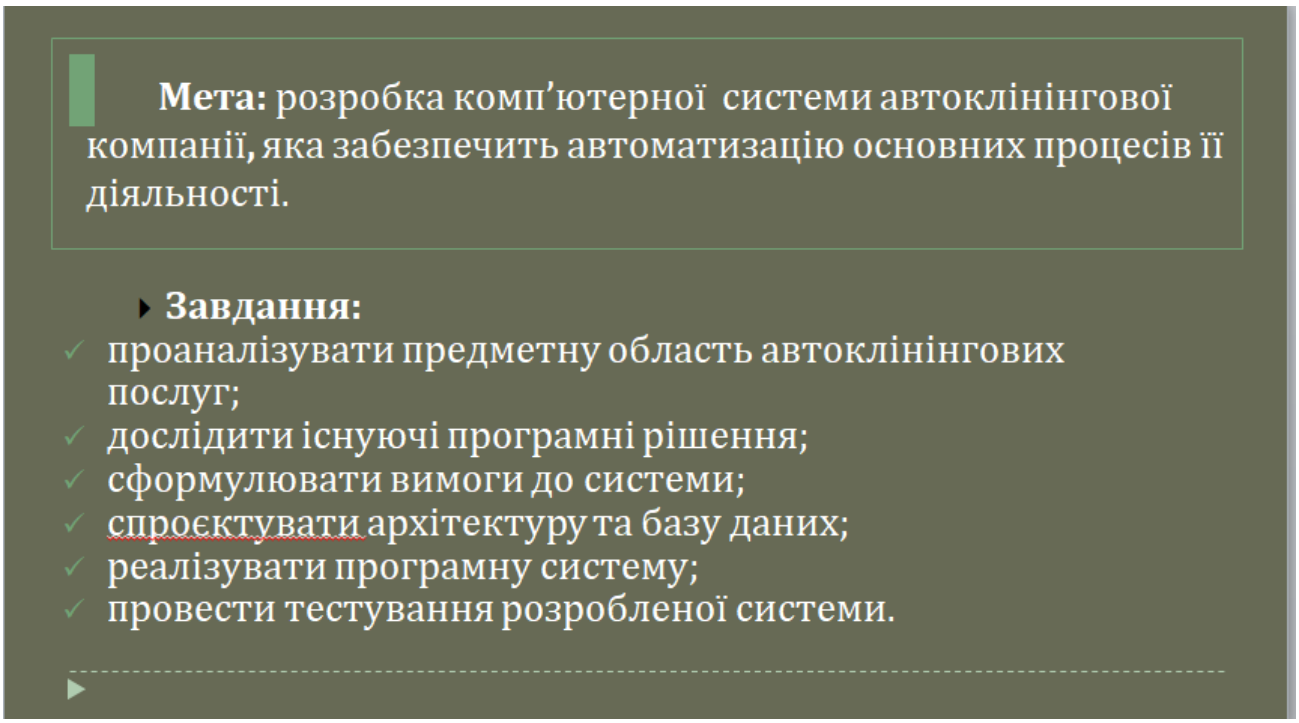


Рисунок Б.2 – Слайд 2

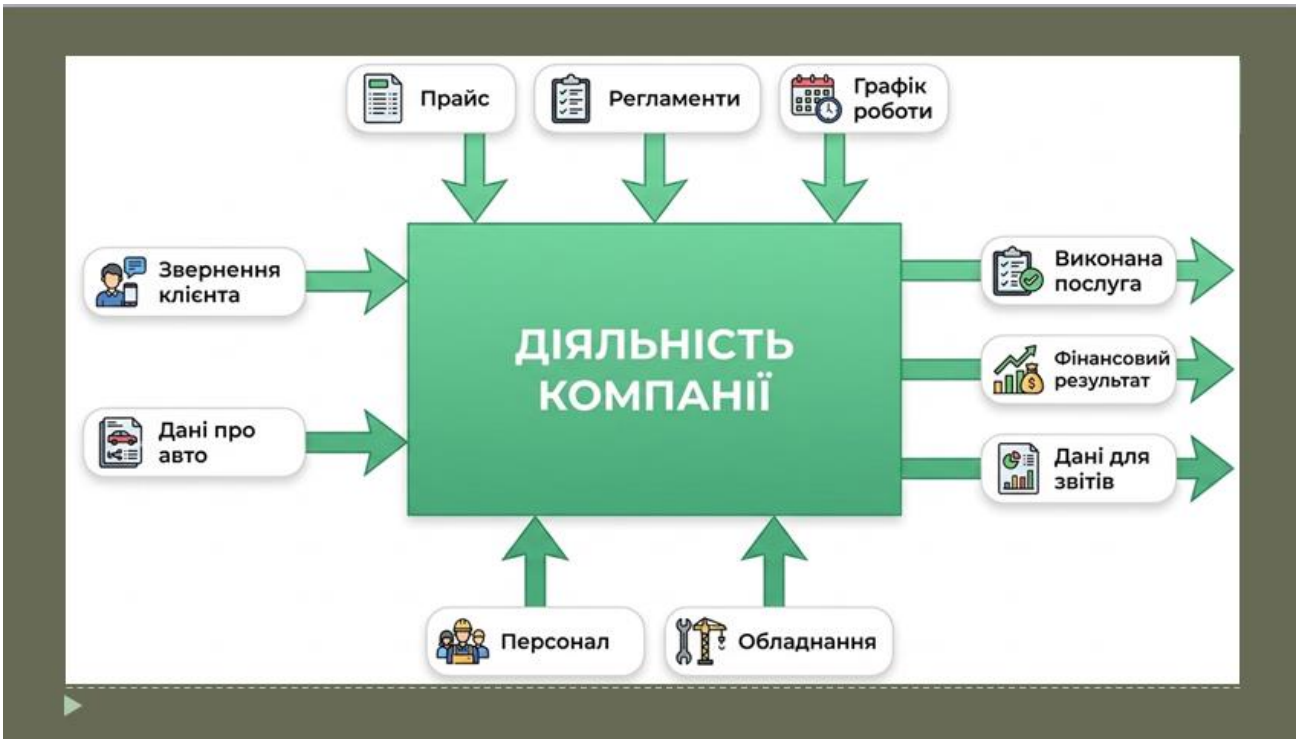


Рисунок Б.3 – Слайд 3


Рисунок Б.4 – Слайд 4


Рисунок Б.5 – Слайд 5

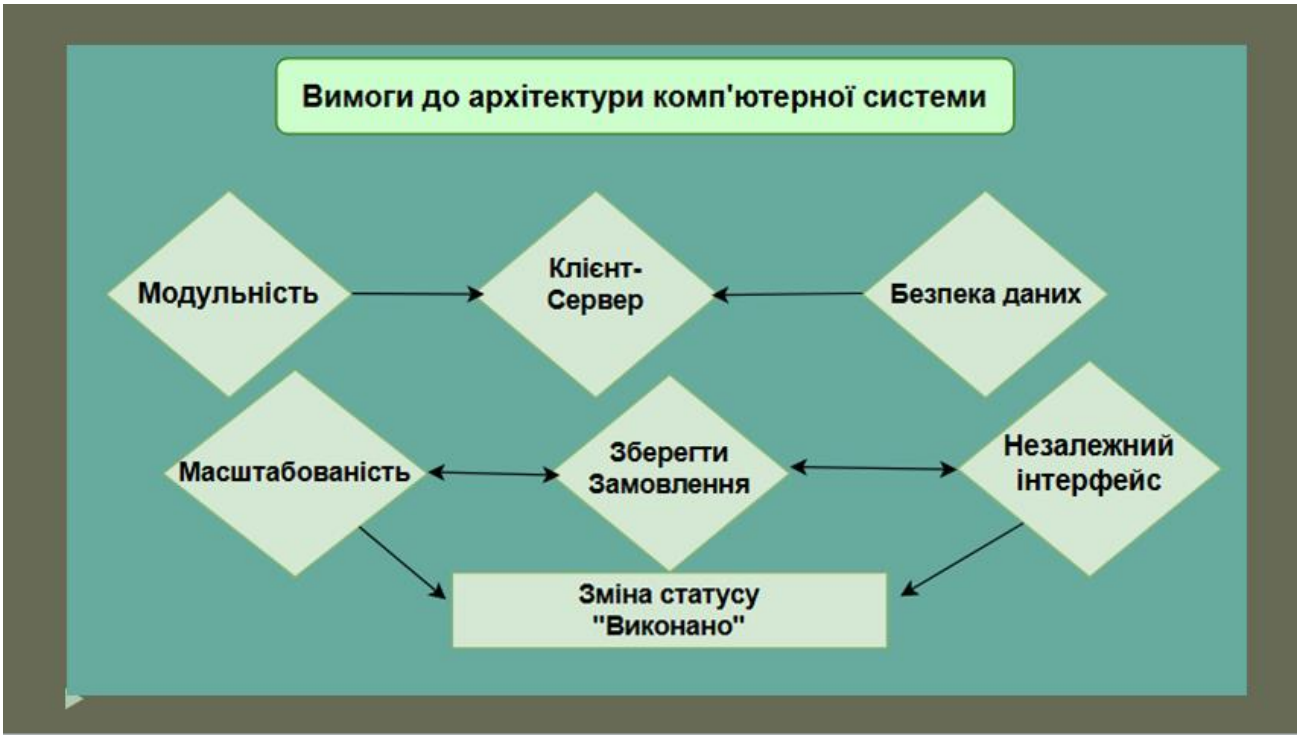


Рисунок Б.6 – Слайд 6

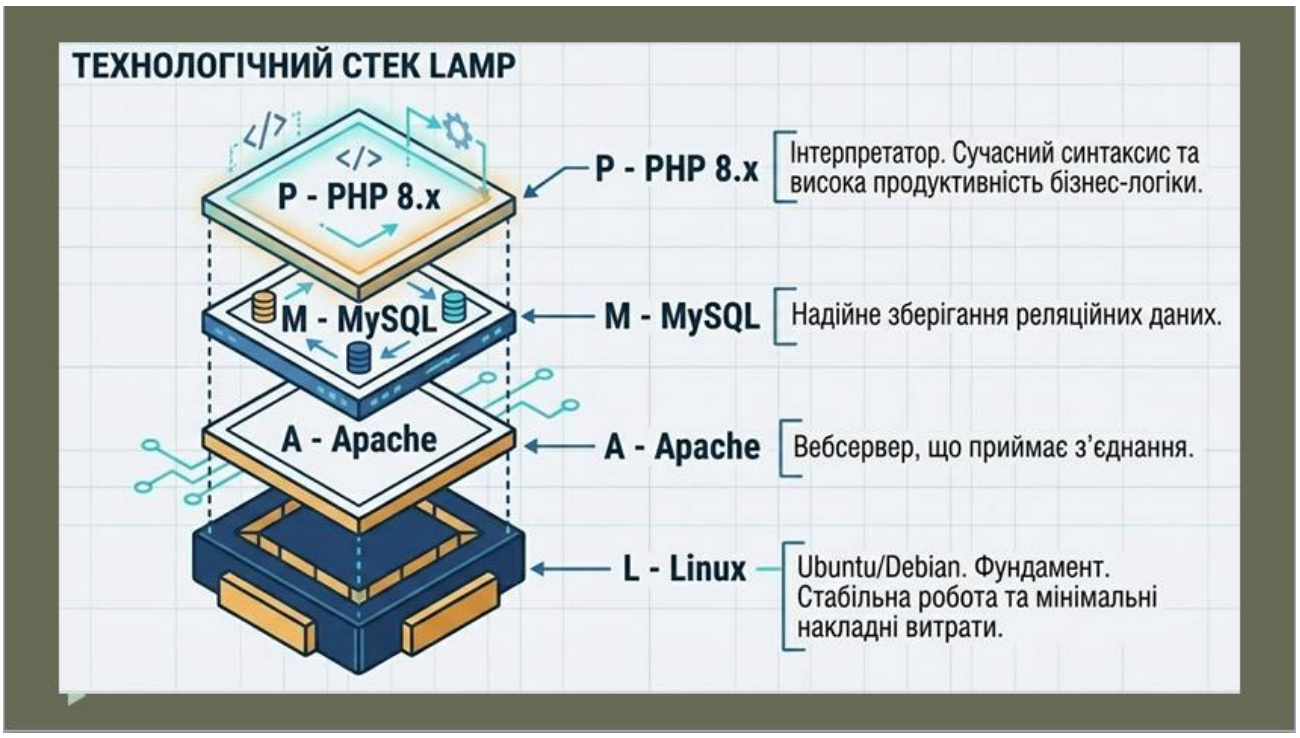


Рисунок Б.7 – Слайд 7

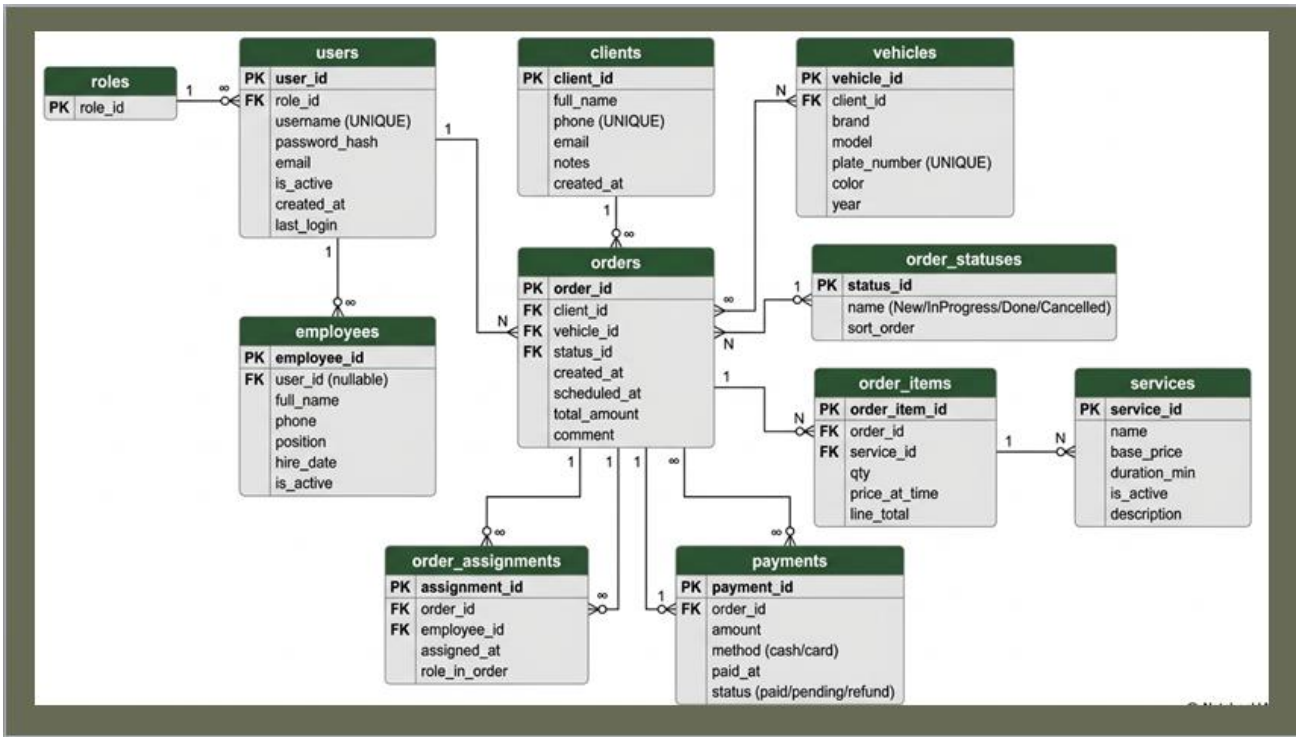


Рисунок Б.8 – Слайд 8


Рисунок Б.9 – Слайд 9

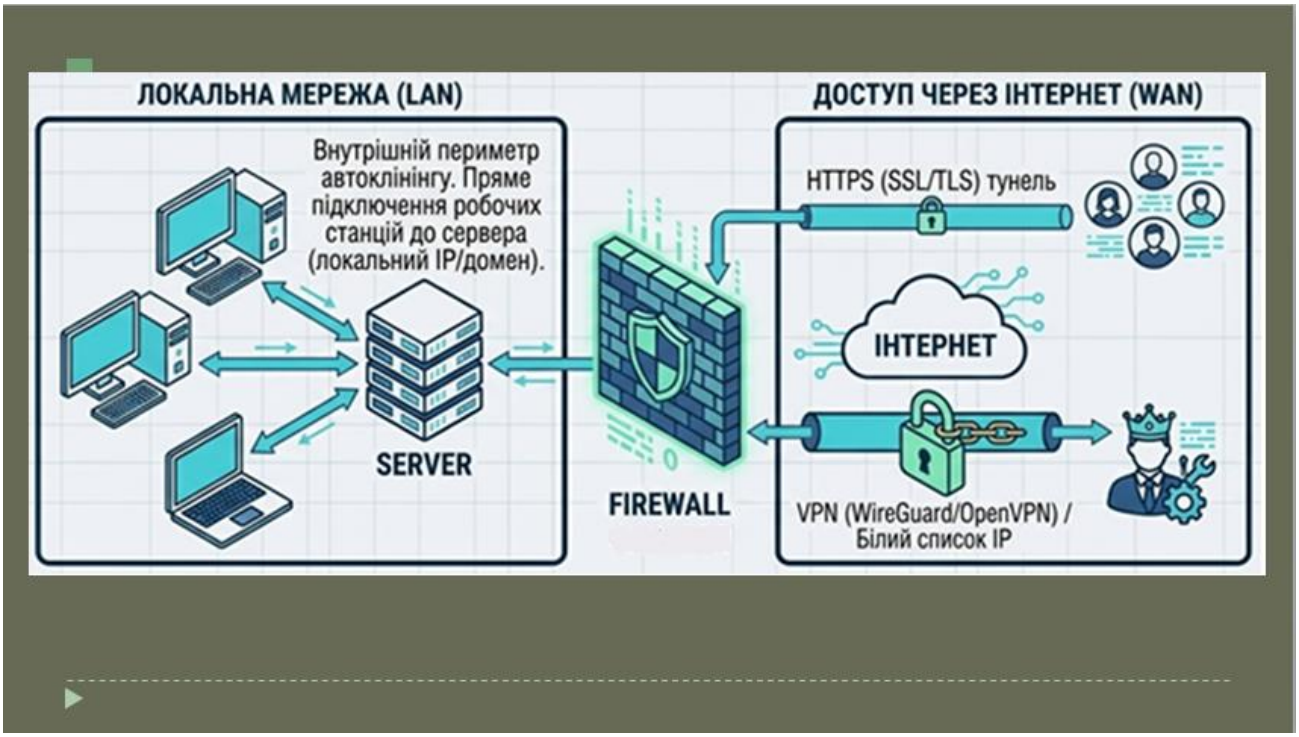


Рисунок Б.10 – Слайд 10

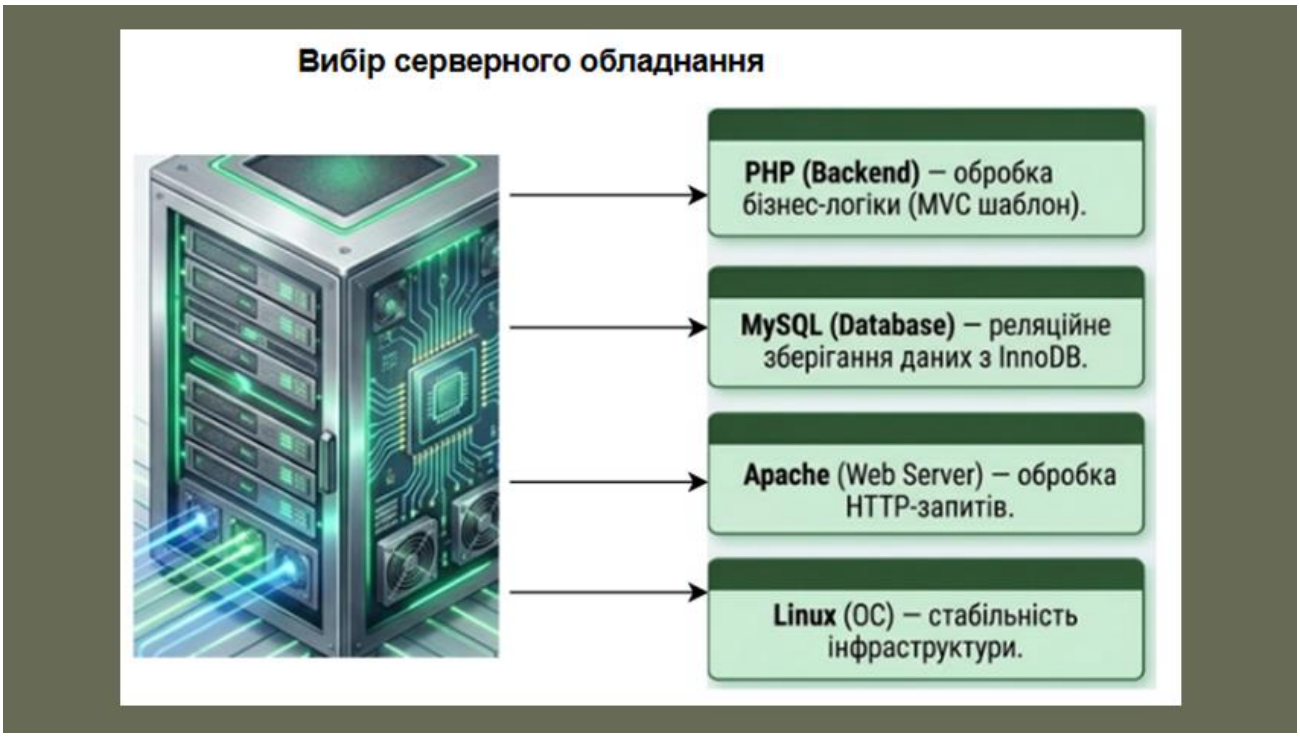


Рисунок Б.11 – Слайд 11

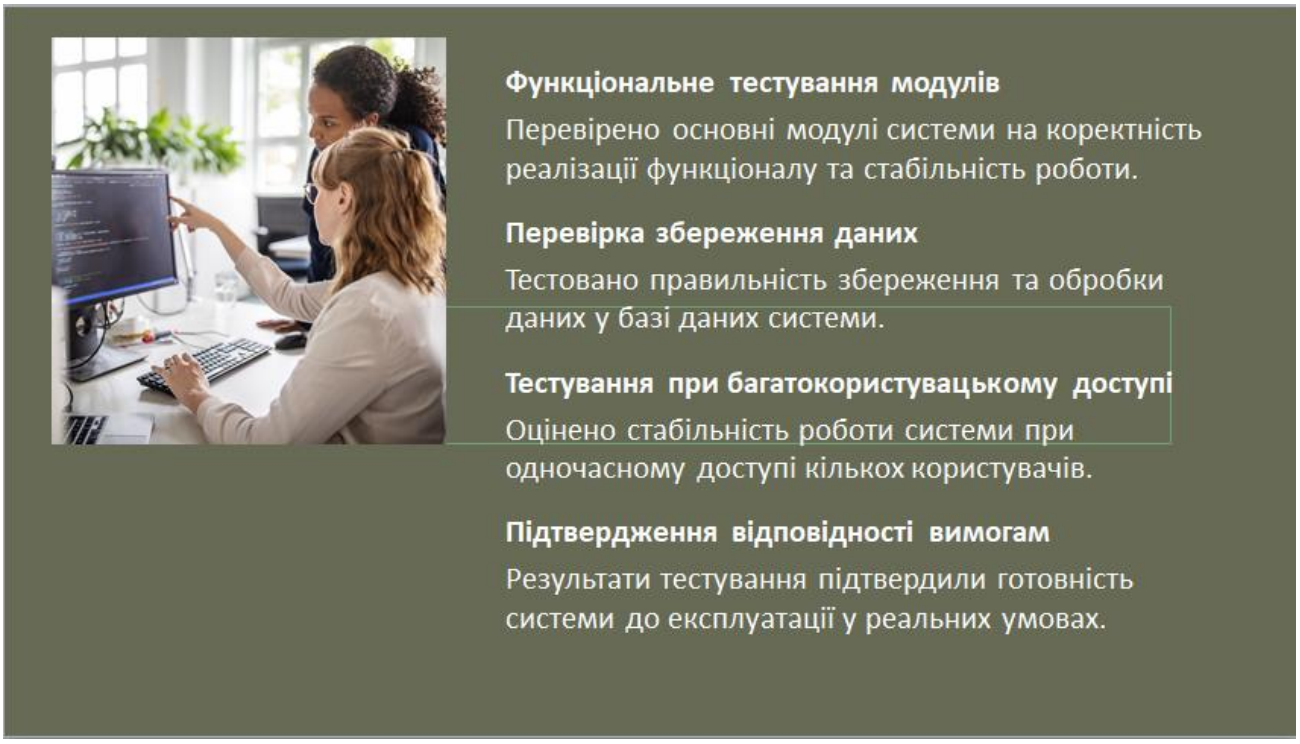


Рисунок Б.12 – Слайд 12

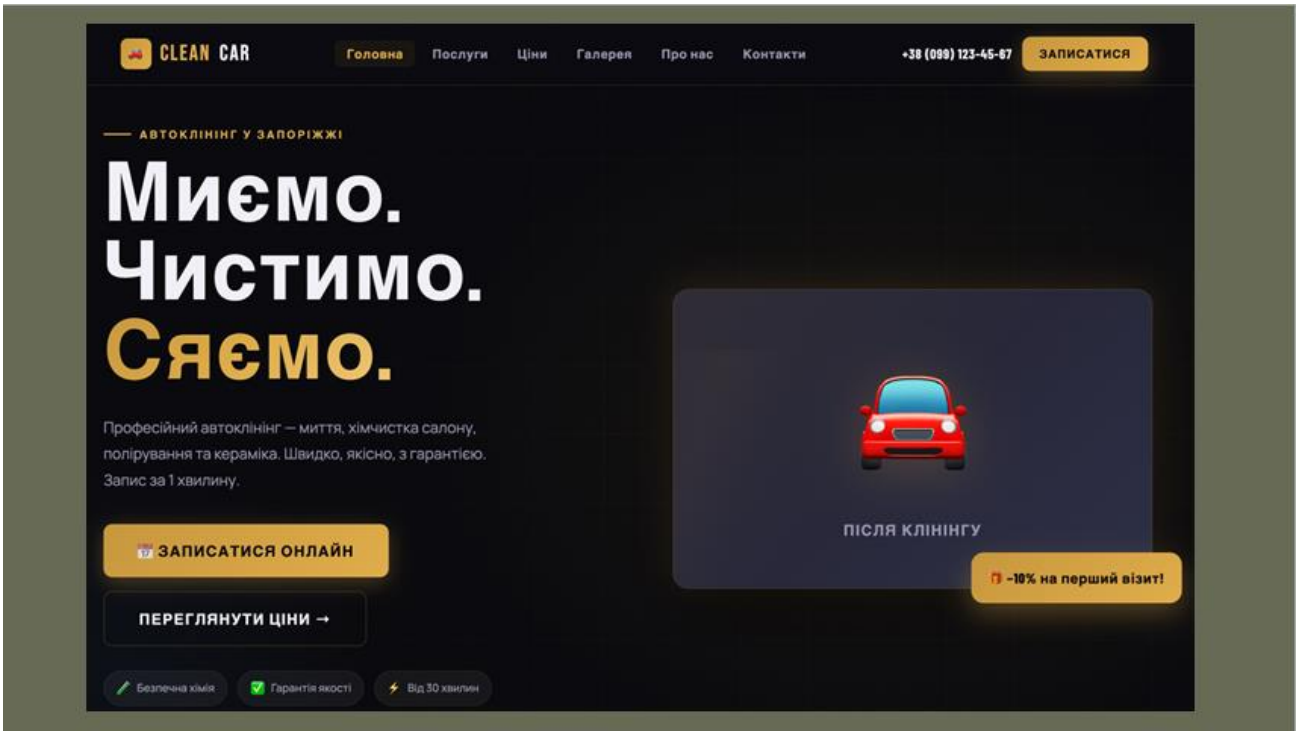


Рисунок Б.13 – Слайд 13

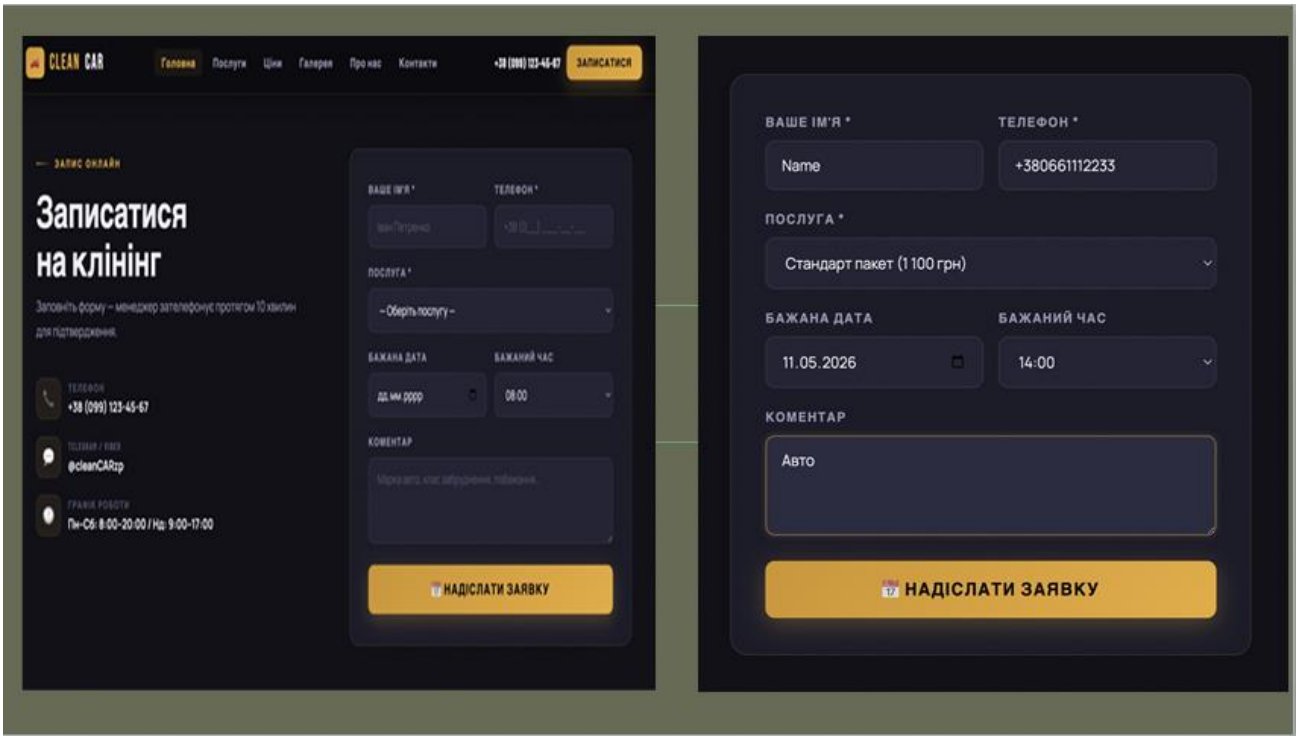


Рисунок Б.14 – Слайд 14


Рисунок Б.15 – Слайд 15


Рисунок Б.16 – Слайд 16