Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Автоматики і комп'ютеризованих технологій_____
(повна назва)

Кафедра _Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки_
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
_Розроблення системи автоматизації для управління замовленнями у моделі дропшипінг-бізнесу_____
(тема)

Виконав:
здобувач ___3____ року навчання,
групи _____АКТСІу-22-1_____
_____Артем НОЗДРЯКОВ_____
(власне ім'я, прізвище)

Спеціальність ____151 Автоматизація та___
комп'ютерно-інтегровані технології_____
(код і повна назва спеціальності)
Тип програми __освітньо-професійна____
Освітня програма __Системна інженерія___
( повна назва освітньої програми)

Керівник _доцент Софія ХРУСТАЛЬОВА
(посада, власне ім'я, прізвище)

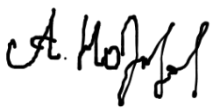
Допускається до захисту

Завідувач кафедри _____ _____ ___Ігор НЕВЛЮДОВ__
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Ноздряков Артем Андрійович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

"09" червня 2025 р. Артем НОЗДРЯКОВ



Харківський національний університет радіоелектроніки

Факультет___Автоматики і комп'ютеризованих технологій _____

Кафедра_Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Рівень вищої освіти_____перший (бакалаврський)_____

Спеціальність_151 Автоматизація та___ комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма_____Системна інженерія_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____20___ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Ноздрякову Артему Андрійовичу_____
(прізвище, ім'я, по батькові)

1.Тема Розроблення системи автоматизації для управління замовленнями у моделі дропшипінг-бізнесу

Затверджена наказом університету від 21 травня 2025 р. № 407 Ст_____

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 липня 2025 р.

3. Вихідні дані до роботи PHP 7.4, фреймворк Yii2, JavaScript, фреймворк Vue.js, система контейнеризації Docker, СУБД MySQL, PhpMyAdmin, REST API, HTML/CSS, браузер Google Chrome, розробницьке середовище Visual Studio Code, Postman, Git._____

4. Перелік питань, що потрібно опрацювати в роботі Аналіз процесу онлайн-замовлень, побудова логіки роботи платформи, проєктування структури бази даних, реалізація клієнтської частини, створення серверної частини з REST API, організація обробки замовлень._____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _Графічний матеріал у вигляді презентації формату .pptx в форматі А4

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

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

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання до кваліфікаційної роботи | 30.04.25 | Виконано |
| 2 | Написання вступу | 03.05 – 09. 05.25 | Виконано |
| 3 | Огляд та аналіз існуючих методів, засобів та автоматизованих систем управління замовленнями | 10.05 – 17.05.25 | Виконано |
| 4 | Розроблення структурної схеми та алгоритму роботи системи автоматизації | 18.05 – 25.05.25 | Виконано |
| 5 | Розроблення системи автоматизації управління замовленнями | 26.05 – 02.06.25 | Виконано |
| 6 | Охорона праці | 03.06 – 05.06.25 | Виконано |
| 7 | Висновки | 06.06 – 08.06.25 | Виконано |
| 8 | Оформлення пояснювальної записки | 09.06 – 14.06.25 | Виконано |

Дата видачі завдання _30_ __квітня_ 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____доцент Софія ХРУСТАЛЬОВА__
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка: 69 с., 2 табл., 13 рис., 2 дод., 19 джерел.

СИСТЕМА АВТОМАТИЗАЦІЇ, МОНІТОРИНГ, МОДЕЛЬ ДРОПШИПІНГ-БІЗНЕС, ОБРОБКА ЗАМОВЛЕНЬ.

Об’єкт розробки – процес управління замовленнями в дропшипінг-моделі бізнесу.

Предмет розробки – програмний засіб для створення, обробки та моніторингу замовлень між продавцем, постачальником і клієнтом.

Мета роботи – підвищення ефективності обслуговування клієнтів за рахунок зменшення кількості помилок, прискорення обробки замовлень і покращення масштабованості бізнесу.

В ході роботи проаналізовано особливості управління замовленнями у дропшипінг-моделі, вимоги до програмного забезпечення, існуючі підходи до автоматизації та інструменти реалізації.

Результатом роботи є система автоматизації реалізована у вигляді програмного засобу, що дозволяє оброблювати, створювати та моніторити замовлення.

Результати роботи відповідають Цілям сталого розвитку, зокрема Цілі 9 «Промисловість, інновації та інфраструктура» (підцілі 9.1 та 9.4), оскільки запропонована система автоматизації покликана підвищити ефективність процесів у сфері електронної комерції, сприяючи цифровій трансформації малого бізнесу та розвитку сучасної ІТ-інфраструктури.

# ABSTRACT

Diploma project, explanatory note**:** 69p., 2 tables,13 fig., 2 add., 19 sourses.

AUTOMATION SYSTEM, MONITORING, DROPSHIPPING BUSINESS MODEL, ORDER PROCESSING.

Object of development – the process of order management in the dropshipping business model.

Subject of development – software tool for creating, processing, and monitoring orders between seller, supplier, and customer.

The aim of the project is to increase customer service efficiency by reducing errors, accelerating order processing, and improving business scalability.

During the project, the peculiarities of order management in the dropshipping model were analyzed, as well as software requirements, existing automation approaches, and implementation tools.

The result of the project is a web application that automates the interaction between seller, supplier, and customer.

The results of the work are in line with the Sustainable Development Goals, in particular Goal 9 “Industry, Innovation and Infrastructure” (subgoals 9.1 and 9.4), as the proposed automation system is designed to increase the efficiency of e-commerce processes, facilitating the digital transformation of small businesses and the development of modern IT infrastructure.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

КІТАР – кафедра комп’ютерно-інтегрованих технологій автоматизації та робототехніки;

ПЗ – програмне забезпечення;

ХНУРЕ – Харківський національний університет радіоелектроніки;

CSS – каскадні таблиці стилів (Cascading Style Sheets);

HTML – мова розмітки гіпертексту (HyperText Markup Language);

JS – JavaScript;

MySQL – система керування реляційними базами даних;

PHP – Personal Home Page (мова сценаріїв на стороні сервера);

RBAC – рольовий контроль доступу (Role-Based Access Control);

REST API – інтерфейс прикладного програмування для передачі даних у форматі REST;

SPA – односторінковий вебзастосунок (Single Page Application);

UML – уніфікована мова моделювання (Unified Modeling Language).

# ВСТУП

Автоматизація бізнес-процесів є ключовим чинником успішного функціонування сучасних моделей електронної комерції. Однією з найперспективніших і водночас складних для реалізації моделей є дропшипінг [1], [2], що дозволяє продавцю уникнути витрат на складування та логістику, передаючи ці функції безпосередньо постачальнику [3].

У моделі дропшипінгу управління замовленнями здійснюється між трьома основними сторонами: продавцем, постачальником та кінцевим клієнтом. Ефективна координація між ними потребує точного обліку, синхронізації залишків, обробки великої кількості транзакцій у реальному часі та забезпечення зворотного зв'язку.

Автоматизація цих процесів є не лише способом зниження витрат, а й засобом підвищення якості обслуговування клієнтів, зменшення кількості помилок, прискорення обробки замовлень і покращення масштабованості бізнесу. У сучасних умовах конкуренції впровадження власних автоматизованих систем управління дозволяє бізнесу досягти нових рівнів ефективності.

Мета роботи – підвищення ефективності обслуговування клієнтів за рахунок зменшення кількості помилок, прискорення обробки замовлень і покращення масштабованості бізнесу.

Об'єкт розробки – процес управління замовленнями в дропшипінг-моделі бізнесу.

Предмет розробки – програмний засіб для створення, обробки та моніторингу замовлень між продавцем, постачальником і клієнтом.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів, засобів та автоматизованих систем управління замовленнями;

– розробити структурну схему та алгоритм роботи системи автоматизації для управління замовленнями;

– провести вибір компонентів системи автоматизації;

– обрати середовище та мову програмування та реалізувати систему автоматизації для управління замовленнями у вигляді програмного засобу;

– протестувати систему у віртуальному середовищі Docker.

Кваліфікаційну роботу виконано відповідно до вимог ДСТУ 3008:2015 [4] та методичних вказівок з підготовки і оформлення кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 "Автоматизація та комп’ютерно-інтегровані технології" освітньої програми «Системна інженерія» [5].

# 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

## 1.1 Особливості управління замовленнями у дропшипінг-моделі

Управління замовленнями є критично важливою складовою будь-якої торговельної платформи, а в моделі дропшипінгу – відіграє центральну роль, оскільки саме через систему замовлень здійснюється комунікація між продавцем, постачальником і клієнтом. Дропшипінг відрізняється від традиційних моделей тим, що продавець не має фізичного контакту з товаром – усе замовлення обробляється та виконується стороннім постачальником. Через це автоматизація управління замовленнями набуває ще більшого значення.

Складність управління дропшипінг-замовленнями полягає в необхідності оперативної обробки великого обсягу замовлень, збереженні актуальності даних про наявність товару, інтеграції з API постачальників, обліку статусів доставки, виставлення рахунків, трекінгу, а також забезпечення зворотного зв’язку з клієнтом. Усе це має відбуватись із мінімальним втручанням оператора.

Типові етапи роботи із замовленнями:

- отримання замовлення з інтерфейсу (frontend);
- перевірка коректності введених даних;
- формування транзакції у базі даних;
- автоматичне надсилання даних постачальнику;
- зміна статусу в міру обробки;
- інформування клієнта;
- архівування завершеного замовлення.

Через те, що в моделі дропшипінгу кількість замовлень часто зростає нелінійно (особливо в періоди розпродажів), ручне опрацювання таких процесів призводить до затримок, помилок і втрати клієнтів. Автоматизовані системи дозволяють уникнути цих ризиків і забезпечити стабільну роботу.

## 1.2 Аналіз існуючих рішень та платформ для керування і моніторингу дропшипінг-процесів

На сучасному ринку існує велика кількість платформ, які забезпечують автоматизоване управління бізнесом у сфері дропшипінгу. До таких рішень відносяться Shopify [6], WooCommerce (з додатками) [7], BigCommerce, Oberlo (до закриття), Spocket, AutoDS, DSers та інші. Вони дозволяють керувати товарами, синхронізувати залишки зі складами постачальників, обробляти замовлення, автоматизувати логістику та відстежувати ключові метрики ефективності.

На рисунку 1.1 представлено основну панель управління замовленнями в Shopify. Вона містить інформацію про покупця, статус оплати, статус виконання, метод доставки та інші деталі, що дозволяють ефективно контролювати процеси дропшипінгу.

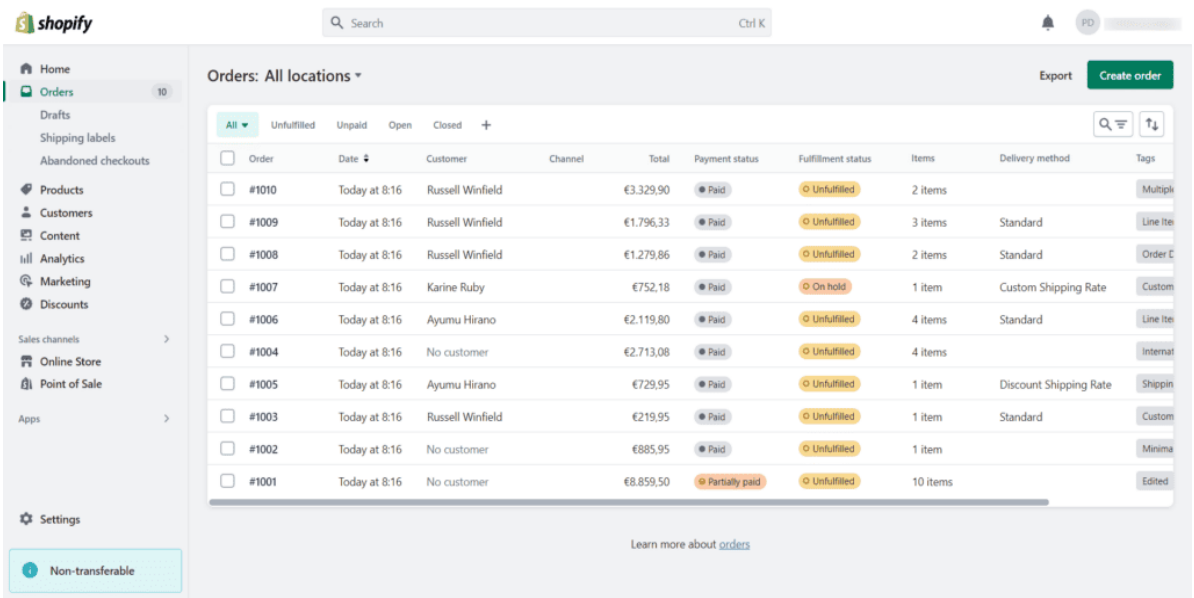



Рисунок 1.1 – Інтерфейс керування замовленнями в системі Shopify

Ці платформи вирішують низку важливих завдань:

– інтеграція з постачальниками (через API або маркетплейси типу AliExpress, CJdropshipping);

– автоматичне оновлення наявності та цін;

– обробка замовлень, зокрема з можливістю автозаповнення даних для постачальника;

– створення трекінгів і сповіщення покупців про статуси доставки;

– базова аналітика щодо продажів, рентабельності, часу обробки тощо.

Порівняння функціональних можливостей популярних дропшипінг-платформ наведено у табл. 1.1.

Таблиця 1.1 – Порівняльна таблиця функціональних можливостей популярних дропшипінг-платформ

| Платформа | Інтеграція з маркетплейсами | Автоматизація замовлень | Трекінг доставки | Підтримка мультивалютності | Аналітика | Вартість |
|---|---|---|---|---|---|---|
| Shopify (з Oberlo / AutoDS) | Так | Так | Так | Так | Середня | Висока |
| WooCommerce + AliDropship | Так | Частково | Частково | Частково | Базова | Низька |
| Spocket | Так | Так | Так | Ні | Базова | Середня |
| DSers | Так | Так | Так | Так | Середня | Низька |

Основні недоліки існуючих рішень:

– обмежена кастомізація логіки обробки замовлень;

– слабка підтримка для великих обсягів даних (масштабування лише через сторонні модулі або API);

– відсутність глибокої бізнес-аналітики та прогнозної аналітики;

– вартість ліцензій для професійних версій або доступу до преміум-функцій.

Перспективи розвитку. Створення універсальної open-source платформи з використанням Vue.js[8], Redis[9], Yii2 [10], WebSockets та MySQL дозволить забезпечити більшу гнучкість та контроль над бізнес-логікою. Така система може включати:

– адаптивну архітектуру черг (для фонових задач і високого навантаження);

– модуль реального часу для сповіщень;

– блок глибокої аналітики з побудовою графіків і KPI;

– API для інтеграції з постачальниками, службами доставки та фінансовими інструментами.

Приклад архітектури типової дропшипінг-системи наведено на рис. 1.2.

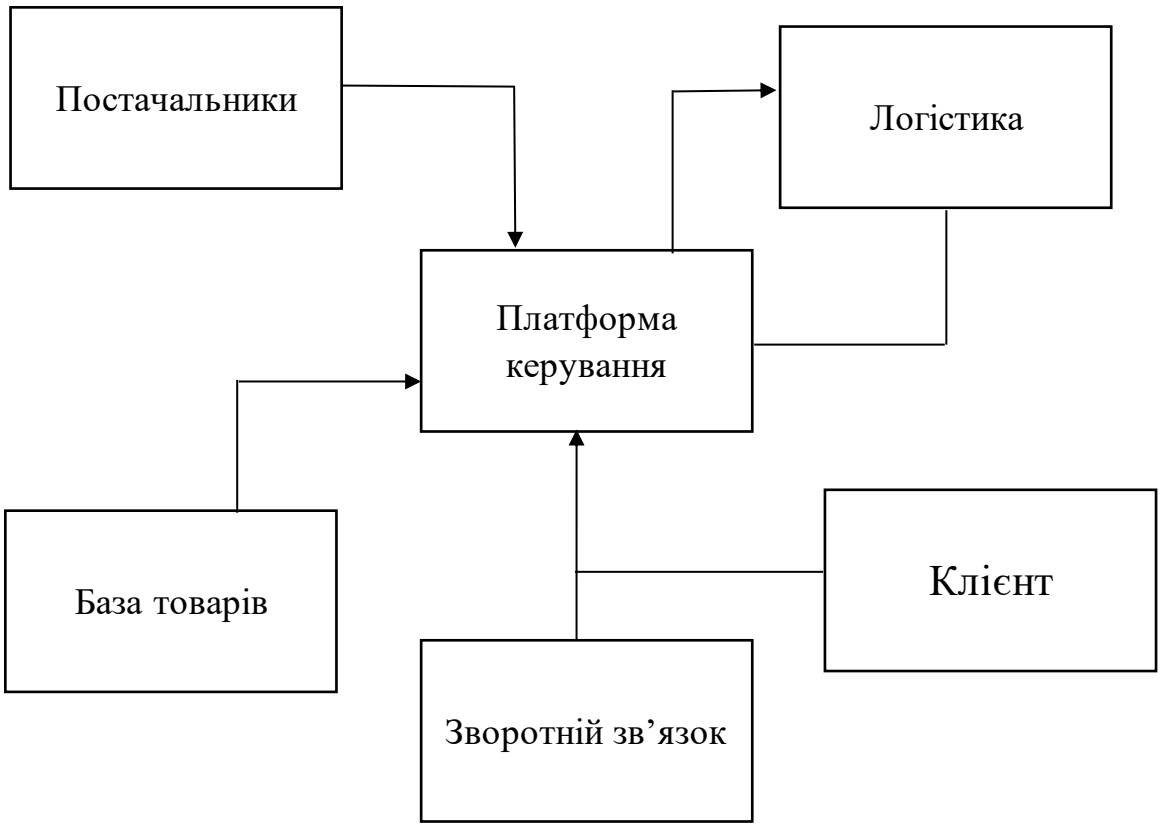



Рисунок 1.2 – Приклад архітектури типової дропшипінг-системи

# 2 РОЗРОБЛЕННЯ СТРУКТУРНОЇ СХЕМИ ТА АЛГОРИТМУ РОБОТИ СИСТЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ

## 2.1 Структурна схема системи автоматизації управління замовленнями

Для ефективної роботи автоматизованої системи потрібно реалізувати низку функціональних та технічних рішень, що забезпечать стабільність, безпеку та масштабованість. Зокрема, необхідно передбачити:

– чітке розділення прав доступу між адміністраторами, менеджерами та клієнтами, із урахуванням ролей і рівнів відповідальності кожного типу користувача;

– повнофункціональний модуль роботи із замовленнями, що включає створення, редагування, перегляд, фільтрацію, пошук та сортування замовлень за різними критеріями;

– централізоване зберігання структурованої інформації про замовлення, товари, постачальників, клієнтів, з можливістю розширення структури у разі масштабування бізнесу;

– логування дій користувачів та ключових подій системи з метою забезпечення аудиту, зворотного аналізу проблем та підвищення безпеки;

– впровадження системи черг обробки для задач, які не вимагають миттєвого виконання (зокрема надсилання листів, синхронізація зі сторонніми сервісами) – планується реалізувати з використанням Redis Queue;

– реалізацію push-нотифікацій через WebSocket на основі Node.js сервера – з метою забезпечення інтерактивного оновлення інтерфейсу в режимі реального часу (наприклад, при зміні статусу замовлення);

– контейнеризацію всіх сервісів у середовищі Docker, що дозволить ізолювати компоненти, спростити процес розгортання та забезпечити гнучке масштабування системи [11].

На рис. 2.1 представлено структурну схему системи автоматизації управління замовленнями.

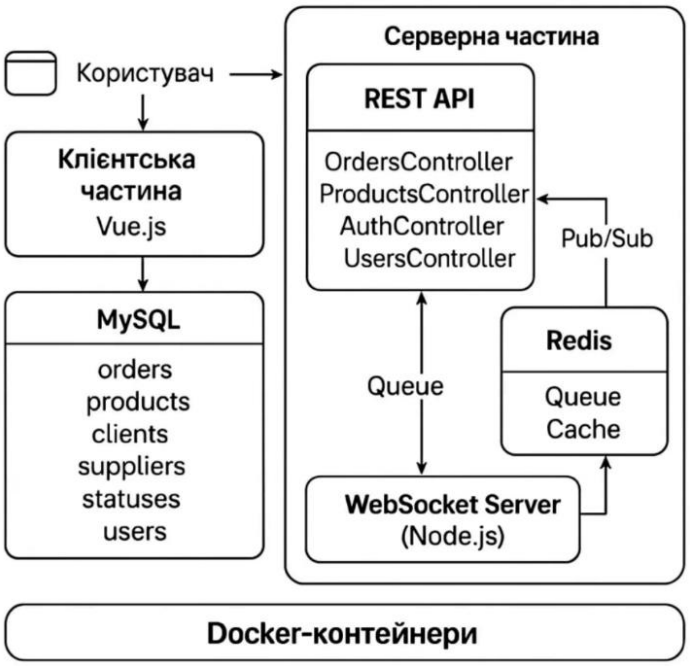



Рисунок 2.1 – Структурна схема системи автоматизації управління замовленнями

Основні компоненти системи:

1. Клієнтська частина (Vue.js).

Використовується для побудови інтерфейсу користувача. Реалізована як SPA, що забезпечує:

– швидке завантаження сторінок без перезавантаження браузера;

– інтерактивну роботу з замовленнями, товарами та повідомленнями;

– запити до REST API для надсилання форм та отримання динамічних даних.

2. Серверна частина (Yii2 – REST API).

PHP-фреймворк Yii2[12] відповідає за бізнес-логіку та обробку запитів від клієнта. Включає кілька контролерів:

– OrdersController – створення, оновлення та перегляд замовлень;
– ProductsController – управління товарами;
– AuthController – авторизація та реєстрація користувачів;
– UsersController – профілі клієнтів і постачальників.

3. База даних (MySQL).

Використовується для зберігання основних сутностей:

– orders – таблиця з усіма замовленнями;
– products – товари, доступні для замовлення;
– clients – користувачі-покупці;
– suppliers – постачальники;
– statuses – статуси замовлень;
– users – загальна таблиця облікових записів.

4. Redis.

Застосовується для:

– кешування тимчасових даних (наприклад, проміжні результати фільтрації);
– роботи з чергами задач (через Queue);
– реалізації публікації/підписки (Pub/Sub) для WebSocket-повідомлень у реальному часі (наприклад, сповіщення про нове замовлення чи зміну статусу).

5. WebSocket-сервер (Node.js).

Працює окремо від основного PHP-сервера та відповідає за:

– трансляцію повідомлень у реальному часі (chat, нові замовлення, зміни статусів);
– прослуховування Redis Pub/Sub;
– комунікацію з frontend через WebSocket-з'єднання.

6. Docker-контейнери.

Вся система розгортається в ізольованих контейнерах:

– кожен компонент (MySQL, Redis, Yii2 backend, Node.js WebSocket-сервер, Nginx, frontend) працює як окремий контейнер;

– спрощується розгортання, CI/CD, масштабування.

Переваги такої структури:

– гнучкість: можна легко змінювати компоненти або масштабувати їх незалежно;

– масштабованість: Redis і черги дозволяють обробляти велику кількість одночасних подій без навантаження на основний сервер;

– безпека: логіка доступу та обробки запитів чітко розділена між фронтендом, бекендом і базою даних;

– модульність: кожна частина (REST API, WebSocket, клієнт, БД) розвивається незалежно, що спрощує командну розробку.

## 2.2 Вибір компонентів системи

У процесі проєктування системи автоматизації обробки замовлень для дропшипінг-моделі було обрано набір компонентів, які найкраще відповідають вимогам до швидкодії, масштабованості, підтримки асинхронних процесів та зручності для подальшої розробки.

Основою серверної частини стала технологія PHP у поєднанні з фреймворком Yii2. Вибір Yii2 пояснюється його продуктивністю, наявністю вбудованої підтримки авторизації, валідації, побудови REST API та ORM ActiveRecord. Це дозволило суттєво скоротити час на розробку бізнес-логіки та зосередитися на реалізації специфіки дропшипінг-процесів.

Для реалізації клієнтської частини було використано фреймворк Vue.js, який забезпечує високу реактивність інтерфейсу, зручне двостороннє зв’язування даних і модульну архітектуру. Такий підхід дозволив створити односторінковий вебзастосунок (SPA), адаптивний до різних пристроїв та масштабований у подальшому.

В якості системи керування базами даних обрано MySQL, що є перевіреним рішенням для роботи з реляційними структурами. Цей вибір обумовлений потребою у надійному зберіганні даних про користувачів, замовлення, товари, постачальників та статуси обробки.

Для кешування, черг фонових задач та обміну повідомленнями між компонентами застосовано Redis. Він забезпечує високу швидкодію і дозволяє реалізувати як класичну систему черг, так і механізм миттєвого сповіщення за допомогою Pub/Sub. Завдяки Redis досягнуто асинхронність обробки деяких критичних операцій, що розвантажує основний потік виконання.

Для забезпечення функціоналу реального часу, зокрема динамічних оновлень статусів замовлень, використано WebSocket-сервер на базі Node.js. Node.js обрано через його ефективну модель обробки подій, що дозволяє обслуговувати велику кількість одночасних з’єднань з мінімальними затримками.

З метою уніфікації середовища розгортання та підвищення надійності всі сервіси упаковані у Docker-контейнери. Це рішення забезпечує незалежність від хост-середовища, спрощує CI/CD та прискорює процес деплою на будь-якому сервері чи у хмарі.

Таким чином, обрані компоненти системи були підібрані на основі принципу оптимального поєднання стабільності, швидкодії, гнучкості та простоти масштабування, що є критично важливим для реалізації ефективної та надійної платформи автоматизації у сфері дропшипінгу.

## 2.3 Моделювання дропшипінг системи в UML

UML – це стандартна мова графічного моделювання, яка використовується для візуалізації структури, поведінки та взаємодії компонентів програмних систем. Вона не є методологією розробки, але виступає універсальним інструментом для фіксації вимог, аналізу та проектування систем [16].

У рамках розробки дропшипінг-платформи UML застосовується для формалізації таких аспектів, як обробка замовлень, статуси виконання, ролі користувачів, логіка взаємодії з постачальниками, механізми авторизації та адміністрування.

Для моделювання використано як структурні, так і поведінкові діаграми:

– діаграми класів і компонентів – для представлення логічної архітектури (моделі користувача, замовлення, товару, постачальника тощо);

– діаграми прецедентів і послідовностей – для опису взаємодії між користувачем, постачальником і системою при оформленні та обробці замовлення.

Застосування UML дозволило системно описати функціональність майбутньої платформи, узгодити вимоги між усіма учасниками проєкту, а також задати чітку основу для реалізації програмних модулів.

### 2.3.1 Структурні діаграми

Структурні діаграми описують статичні аспекти системи – зокрема, її логічну і фізичну структуру, об’єкти, таблиці, контролери та їхні взаємозв’язки. У межах дропшипінг–системи доцільно використовувати такі діаграми:

Діаграма компонентів – дозволяє показати основні частини системи: клієнтську частину (Vue.js), серверну частину (Yii2), базу даних MySQL,

Redis–сервер, WebSocket–сервер. Компонентна діаграма чітко візуалізує логічну взаємодію між модулями. На рисунку 2.2 представлено діаграму компонентів для дропшипінг-платформи з автоматизованим обліком замовлень.

Діаграма класів – використовується для моделювання об'єктів, їхніх властивостей та методів. У системі управління замовленнями необхідно змоделювати такі класи, як Order, Product, User, Supplier, Status, Role, Notification. Ці класи взаємодіють між собою, утворюючи об’єктно–орієнтовану модель для взаємодії з базою даних [17].

На рисунку 2.3 зображено діаграму класів, що описує основні сутності системи.

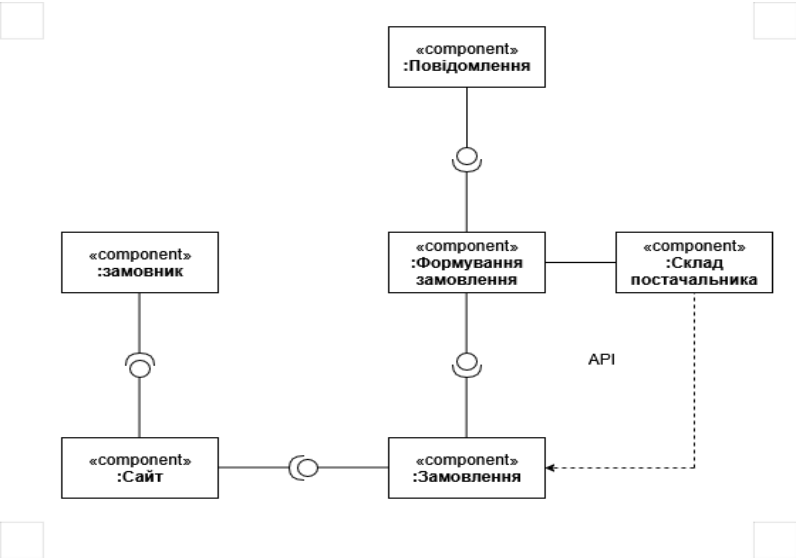



Рисунок 2.2 – Діаграма компонентів

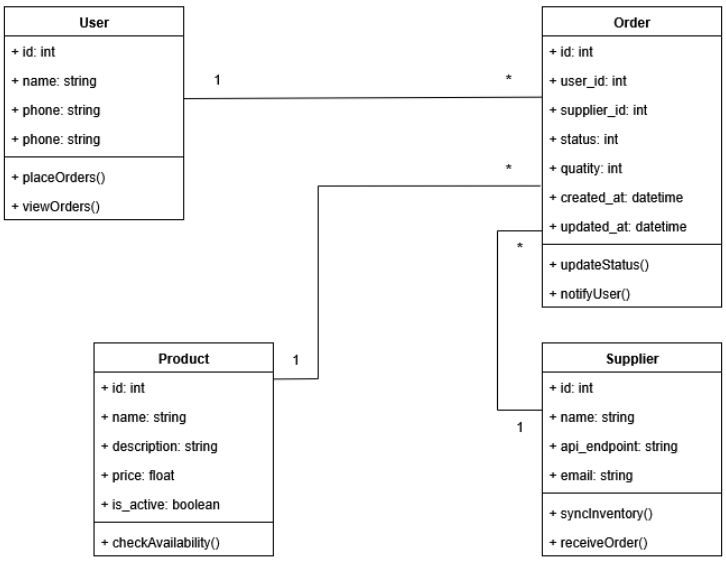



Рисунок 2.3 – Діаграма класів

Діаграма об'єктів – відображає конкретний стан системи в певний момент часу, наприклад, коли є активне замовлення, прив’язане до користувача і постачальника. Це дозволяє візуально проаналізувати реальний випадок роботи системи.

Структурні діаграми особливо корисні під час розробки backend–архітектури системи, оскільки вони дозволяють формалізувати взаємозв'язки між сутностями, забезпечуючи узгодженість даних.

### 2.3.2 Поведінкові діаграми

Поведінкові діаграми UML описують динамічні аспекти системи, зокрема бізнес–логіку, алгоритми роботи, сценарії взаємодії користувачів з платформою. Для системи управління замовленнями доцільно використовувати:

Діаграма прецедентів (use case diagram**)** – відображає функціональність системи з точки зору користувача. У системі дропшипінгу можна виділити три основні актори: «Користувач», «Адміністратор» та «Постачальник», кожен із яких виконує власні сценарії взаємодії із системою.

Користувач має можливість переглядати товари, шукати потрібні позиції, оформлювати замовлення, перевіряти статус доставки, редагувати власний профіль та здійснювати повторне замовлення.

Адміністратор відповідає за керування товарами, обробку замовлень, адміністрування користувачів і підключення або редагування постачальників.

Постачальник отримує інформацію про нові замовлення, змінює статус їх обробки, передає трек–номери та синхронізує залишки товарів на складі.

На рисунку 2.3 зображено діаграму прецедентів, яка ілюструє ключові сценарії взаємодії кожного з акторів із системою управління замовленнями у моделі дропшипінгу.

Діаграма активностей – використовується для опису процесу обробки замовлення: від моменту його створення до зміни статусу, надсилання запиту постачальнику, отримання відповіді, відображення статусу клієнту. Це дозволяє чітко визначити гілки прийняття рішень, перевірки, очікування або повідомлення.

Діаграма послідовності – дозволяє змоделювати покрокову взаємодію об'єктів. Наприклад: користувач надсилає запит на створення замовлення –

контролер зберігає дані в базі – черга Redis виконує асинхронну задачу – WebSocket надсилає оновлення – статус змінюється у реальному часі.

На рисунку 2.4 представлено діаграму послідовності для сценарію «Створення замовлення».

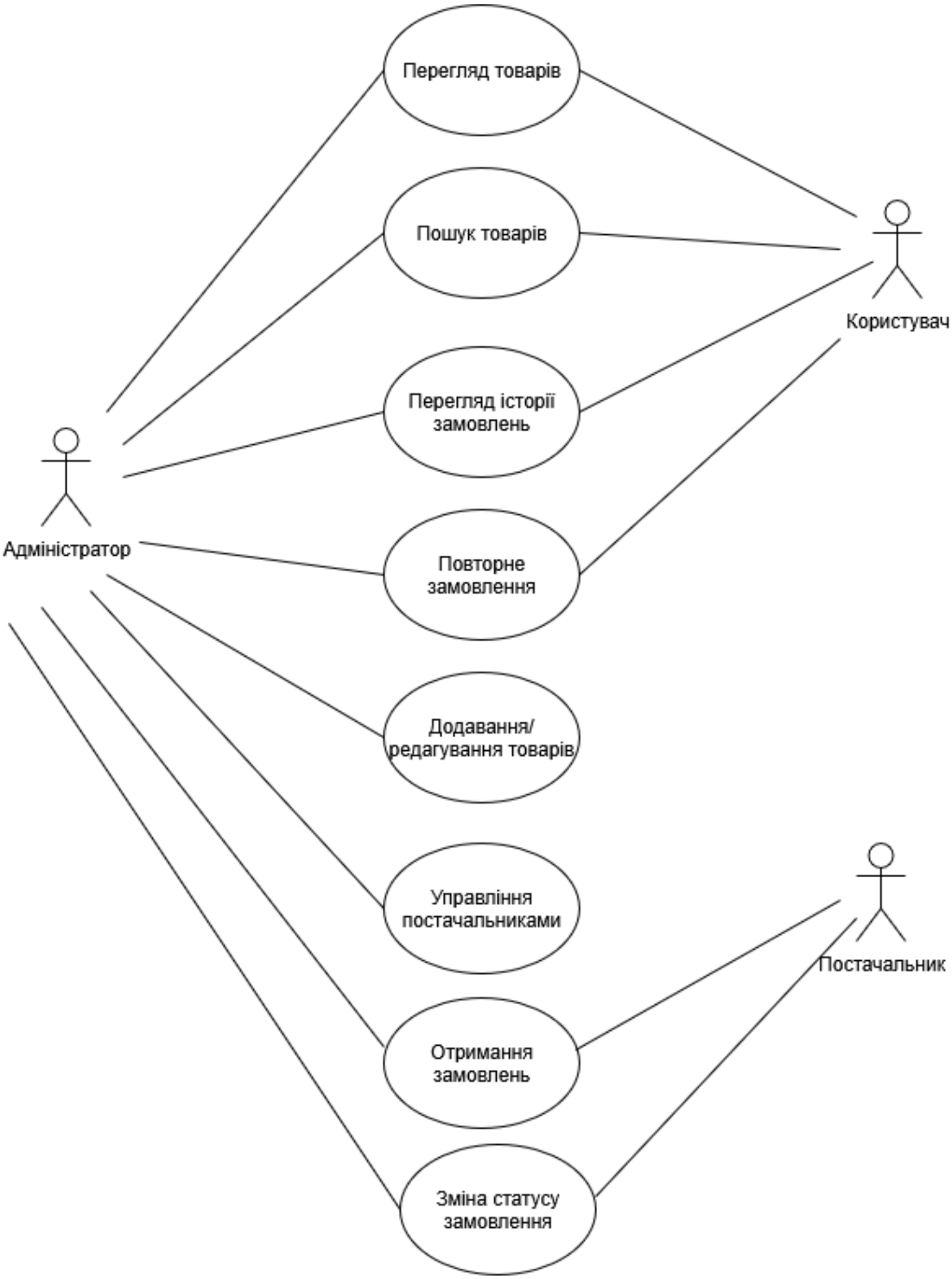



Рисунок 2.3 – UML. Діаграма прецедентів

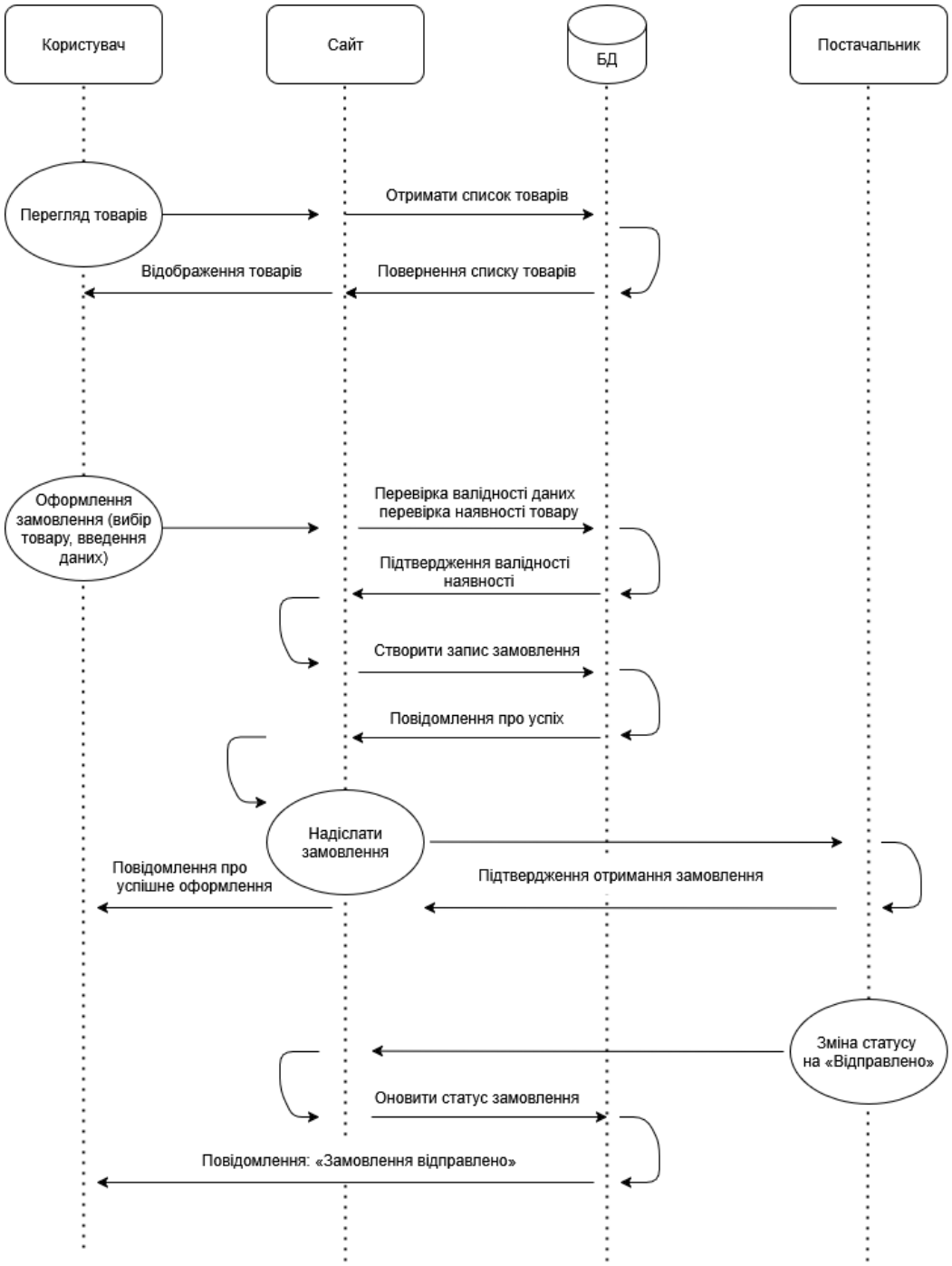



Рисунок 2.4 – UML. Діаграма послідовності

Такі діаграми важливі для командної роботи, налагодження API, створення автотестів і симуляції сценаріїв у бізнес–логіці. Вони також корисні при майбутньому масштабуванні – наприклад, при впровадженні багатопостачальницької моделі або аналітики.

## 2.4 Алгоритм роботи системи автоматизації

У процесі обробки замовлень у дропшипінг-моделі ключову роль відіграє своєчасна передача даних між продавцем і постачальником. На рисунку 2.5 зображено типовий алгоритм взаємодії в рамках такого бізнес-процесу. З-поміж усього процесу в межах даного програмного рішення було автоматизовано два важливих етапи:

– передача замовлення постачальнику;

– оновлення статусів замовлення в особистому кабінеті продавця та клієнта.


Рисунок 2.5 – Процес обробки замовлення в моделі дропшипінгу

Після того як покупець оформлює та оплачує замовлення на сайті продавця, система автоматично формує набір необхідних даних (інформацію про товар, контактні дані покупця, адресу доставки, спосіб доставки тощо) та передає їх постачальнику через інтегрований механізм API або особистий кабінет постачальника. Це дозволяє уникнути ручного перенесення даних і зменшує ризик помилок.

Накладну формує постачальник, оскільки саме він відповідає за фізичну доставку товару та логістичне оформлення. Система лише надає повний набір

вхідних даних, необхідних для створення документів, що суттєво пришвидшує процес та зменшує кількість комунікацій між сторонами.

Після обробки замовлення постачальником та відправки товару система отримує відповідне повідомлення (або оновлює статус вручну через особистий кабінет), після чого статус замовлення змінюється на «відправлено», а згодом – на «доставлено», коли товар отримано покупцем.

Таким чином, автоматизація ключових етапів дозволяє зменшити час на обробку кожного замовлення, забезпечити актуальність інформації для всіх сторін та покращити загальну ефективність дропшипінг-системи.

## 2.5 Теоретичні основи автоматизованого управління замовленнями

Управління замовленнями у дропшипінг-моделі бізнесу є критично важливим процесом, що охоплює створення, передачу, обробку та контроль виконання кожного замовлення. З точки зору теорії автоматичного управління (ТАУ), цей процес можна розглядати як дискретну інформаційно-керовану систему з об’єктами керування, сигналами зворотного зв’язку та логікою прийняття рішень.

У реалізованій системі автоматизації користувач виступає ініціатором замовлення – формує вхідні дані (об'єкт керування), які надходять до обчислювального блоку – бекенд-системи на основі фреймворку Yii2. Система обробляє ці дані, виконує перевірки, проводить валідацію та зберігає інформацію в базу даних. Після цього автоматично відбувається передача замовлення постачальнику, що є ключовим етапом процесу керування – саме на цьому етапі ініціюється зовнішній бізнес-процес, на який система має лише частковий вплив.

Управління відбувається за принципом відкритого контуру з частковим зворотним зв’язком: після передачі замовлення постачальнику останній здійснює його обробку та надсилає відповідь у вигляді статусу (наприклад, «обробляється», «відправлено», «виконано», «скасовано» тощо). Ці статуси

фіксуються у системі автоматично (через API) або вручну (через інтерфейс адміністратора), що забезпечує зворотний зв'язок та дозволяє користувачу відстежувати стан свого замовлення.

З формальної точки зору, поведінку системи можна описати як дискретну динамічну систему, яка змінює свій стан під дією зовнішніх сигналів:

$x(k+1)=f(x(k),u(k))x(k+1) = f(x(k), u(k))x(k+1)=f(x(k),u(k))$

де:

– $x(k)x(k)x(k)$ – стан замовлення на момент часу $kkk$ (наприклад, «створено», «очікує підтвердження», «виконано»);

– $u(k)u(k)u(k)$ – керуючий вплив (дії користувача або адміністратора – наприклад, натискання кнопки "оформити замовлення");

– f – функція переходу між станами, реалізована у вигляді логіки бекенду.

Відповідь постачальника на запит можна формалізувати як сигнал зворотного зв’язку:

$y(k)=h(x(k))y(k) = h(x(k))y(k)=h(x(k))$

де:

– $y(k)y(k)y(k)$ – статус замовлення, отриманий від постачальника;

– h – функція формування зворотного сигналу, залежно від поточного стану системи.

На основі отриманого статусу система виконує прийняття рішень щодо подальших дій:

$u(k+1)=g(y(k))u(k+1) = g(y(k))u(k+1)=g(y(k))$

де:

– g – логіка реагування (наприклад, відправлення сповіщення користувачу або зміна статусу на «архівований»).

У більш складних випадках, коли враховується ймовірність успішного завершення замовлення, можна застосувати елементи імовірнісного моделювання. Наприклад:

P(y(k)="виконано"|x(k)="відправлено")=p

де p – імовірність того, що замовлення буде виконано після відправлення, яка може базуватись на історичних даних платформи.

Таким чином, в межах цієї системи автоматизації реалізовано часткове управління процесом обробки замовлень, яке включає:

- формування керуючого впливу (дані замовлення);
- передавання керуючого сигналу до виконавця (постачальника);
- реєстрацію зворотного сигналу у вигляді статусу;
- прийняття рішень на основі актуального стану замовлення.

Така модель відповідає базовим принципам ТАУ для дискретних подієвих систем. Вона дозволяє мінімізувати втручання оператора (адміністратора), зменшити кількість помилок при передачі замовлення та підвищити загальну ефективність роботи дропшипінг-платформи.

# 3 РОЗРОБЛЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ

## 3.1 Вибір середовища розробки та мова програмування

У процесі розробки автоматизованої системи управління замовленнями електроних компонентів у дропшипінг-моделі було використано сучасне програмне забезпечення та інструменти, які забезпечують зручність налаштування середовища, масштабованість та стабільність роботи. Одним із ключових компонентів став Docker – платформа контейнеризації, що дозволяє створювати ізольовані середовища для кожного сервісу, які легко запускати на будь–якій машині незалежно від ОС.

Весь проєкт був розділений на окремі контейнери:

– app – основний PHP–контейнер на базі Yii2, що відповідає за логіку та обробку запитів;

– mysql – контейнер бази даних MySQL 8.0, у якому зберігаються всі записи про замовлення, клієнтів, товари;

– redis – in–memory–сховище, що використовується для кешування та обробки черг;

– node – допоміжний сервер для подій реального часу (в майбутньому планується додавання WebSocket);

– frontend – Vue.js SPA (Single Page Application) [14], що надає зручний інтерфейс користувачу;

– phpmyadmin – для візуального керування базою даних.

Файл docker-compose.yml описує, як всі ці сервіси взаємодіють між собою. Завдяки Docker було досягнуто стабільної роботи всіх сервісів незалежно від ОС розробника. На рисунку 3.1 зображено консоль Docker під час запуску проєкту.

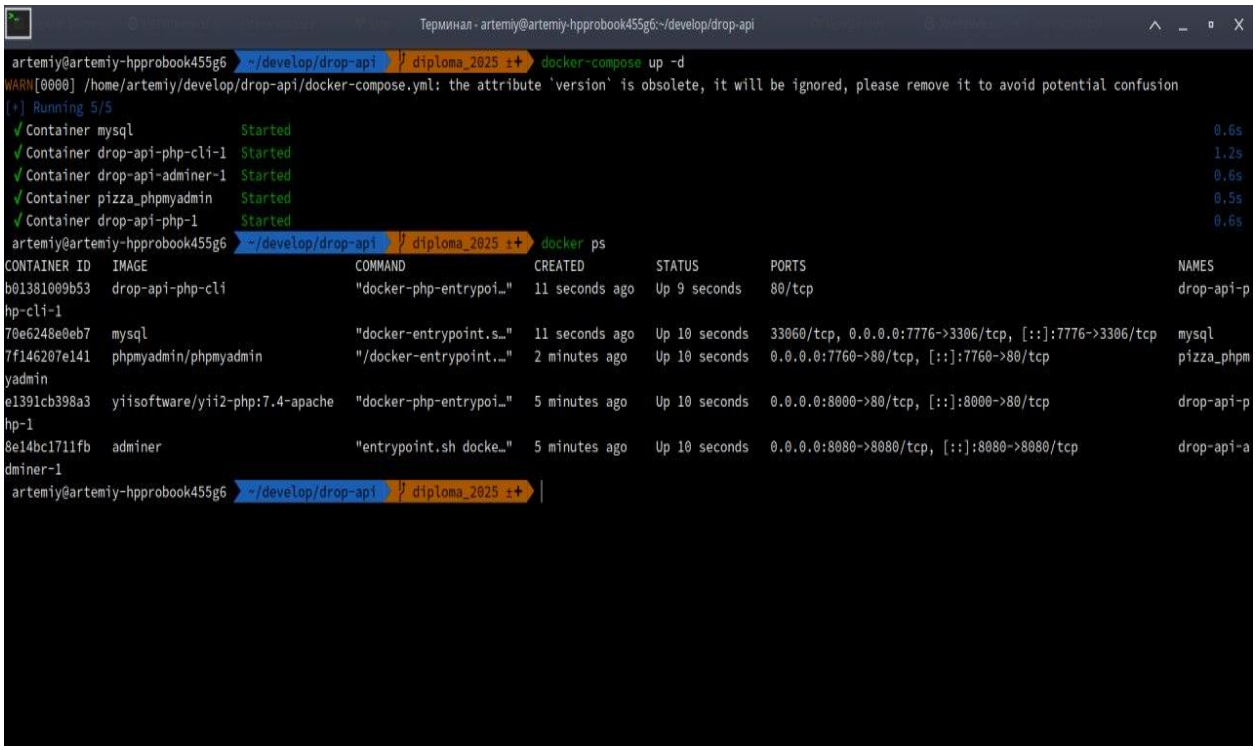



Рисунок 3.1 – Консоль Docker при запуску проєкту

## 3.2 Робота з базою даних

Для збереження даних про замовлення, товари, клієнтів та постачальників було створено реляційну базу даних у MySQL. Доступ до неї здійснюється за допомогою PhpMyAdmin – веб-застосунку, що дозволяє переглядати, редагувати та моделювати структуру БД у зручному графічному інтерфейсі.

На рисунку 3.2 представлено інтерфейс PhpMyAdmin, який демонструє структуру бази даних про замовлення.

Ключовою таблицею є orders, яка містить наступні поля:

- id – унікальний ідентифікатор замовлення;
- user_id – прив’язка до користувача (може бути NULL, якщо гість);
- supplier_id – прив’язка до постачальника;
- product_id – товар у замовленні;
- status – поточний стан (нове, обробляється, доставлено тощо);
- created_at, updated_at – часові мітки.

Додаткові таблиці products, suppliers, users дозволяють реалізувати повноцінну модель для обліку та зв'язків між сутностями [18].

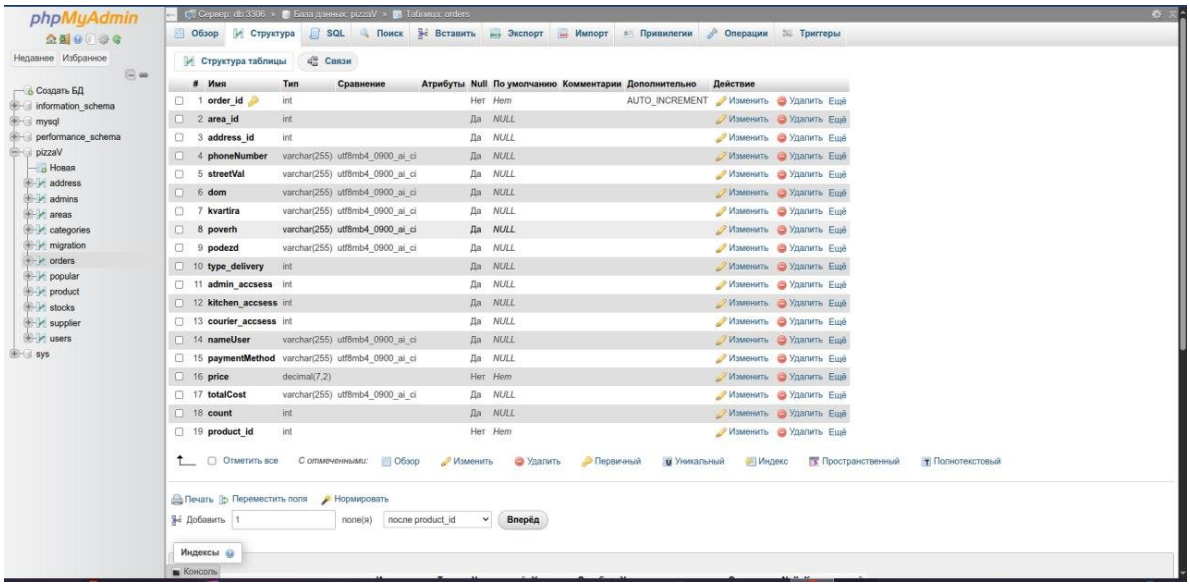



Рисунок 3.2 – Вікно phpMyAdmin зі структурою таблиці orders

## 3.3 Програмна реалізація інтерфейсу користувача

Клієнтська частина розробленої системи реалізована за допомогою сучасного JavaScript–фреймворку Vue.js [16], який широко використовується для створення динамічних односторінкових веб–застосунків. Завдяки своїй компонентній структурі, Vue дозволяє розділити інтерфейс на незалежні логічні блоки, кожен з яких виконує окрему функцію – від форми введення даних до перегляду замовлення.

Таке розділення забезпечує не лише зручність у розробці, а й легке тестування, повторне використання коду та масштабованість при розширенні проєкту. У рамках реалізованої дропшипінг–платформи було створено низку візуальних компонентів, що взаємодіють із REST API, реалізованим на сервері через Yii2.

Основна сторінка оформлення замовлення є ключовим елементом взаємодії з користувачем. Вона містить кілька важливих елементів:

– форма вибору товару – реалізована у вигляді випадаючого списку з автоматичним пошуком. користувач бачить назву товару, його опис та ціну;

– поле введення контактної інформації – передбачає введення імені, email та номеру телефону. додатково реалізована перевірка правильності вводу (наприклад, обов’язкова наявність «@» у полі email);

– вибір способу доставки – дозволяє користувачу вибрати з доступних варіантів (самовивіз, доставка кур’єром, поштове відправлення);

– кнопка підтвердження "замовити" – при натисканні дані відправляються на сервер за допомогою http–запиту типу post, після чого користувач отримує повідомлення про успішне оформлення замовлення або помилки у заповненні форми.

Усі дані перед відправленням проходять первинну валідацію на стороні клієнта – перевіряється обов’язковість полів, правильність формату email, номеру телефону тощо. У разі виявлення помилок, відповідні поля підсвічуються, а система виводить пояснювальні повідомлення. Це зменшує кількість некоректних запитів до сервера і підвищує зручність користування.

На сервері також відбувається додаткова перевірка – щоб виключити можливість фальсифікації запиту або відправлення шкідливих даних. Валідація реалізована у Yii2 через механізми моделей (rules()) з фіксацією помилок у логах.

На рисунку 3.3 представлено приклад інтерфейсу користувача на сторінці оформлення замовлення. Видно компонент Vue, що містить форму з полями для вибору товару, контактної інформації та варіанту доставки. Праворуч розміщене підтвердження дії через кнопку, яка активує серверний запит. Інтерфейс є інтуїтивно зрозумілим навіть для користувачів без технічного досвіду.

Крім основного компонента оформлення замовлення, у фронтенді реалізовано також:

– спливаючі повідомлення (alert–компоненти) про успішне замовлення;

– перевірку наявності користувача у системі (через збереження токенів у localStorage);

– інтеграцію з адмінчастиною (за іншими ролями доступу) – реалізовану в тому ж фреймворку Vue.js.

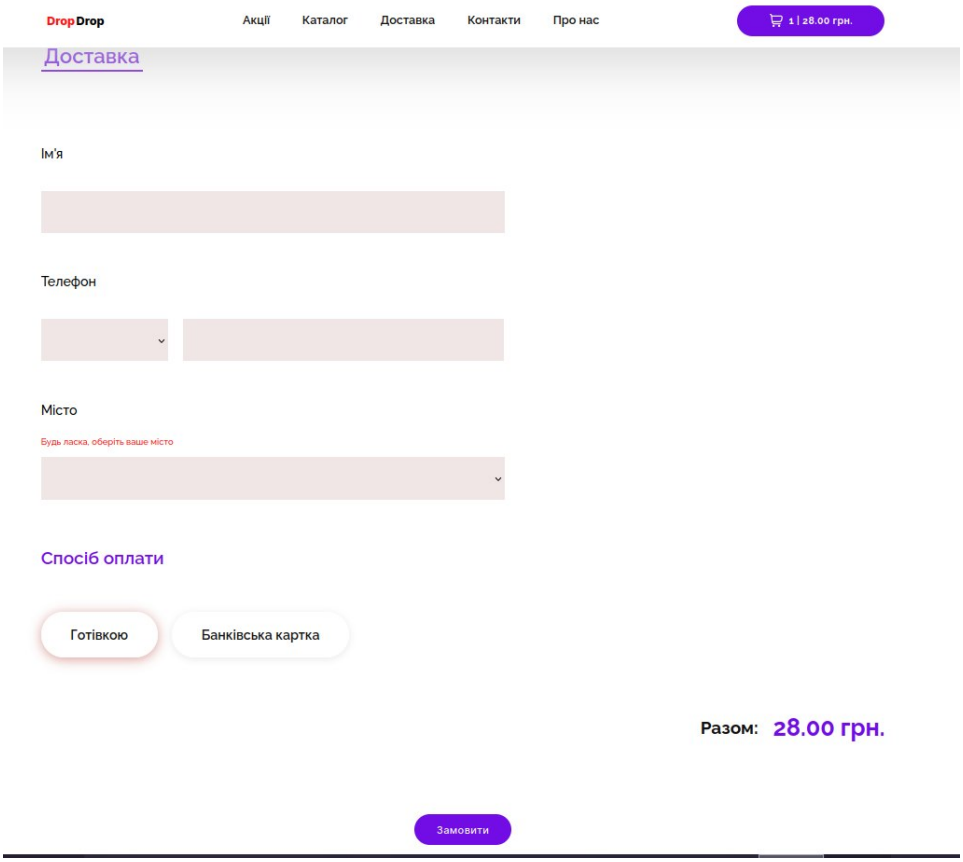



Рисунок 3.3 – Vue.js компоненти на сторінці оформлення замовлення

Таким чином, реалізований інтерфейс забезпечує зручну взаємодію користувача із системою, дозволяє легко і швидко оформити замовлення, а також скорочує кількість помилок за рахунок попередньої перевірки введених даних. У майбутньому можливе розширення функціональності інтерфейсу за рахунок впровадження особистого кабінету, історії замовлень і можливості повторного оформлення.

## 3.4 Функціональність адміністративної панелі

Для управління замовленнями з боку адміністратора реалізована окрема панель з розширеними правами доступу. У цій частині системи доступні:

– перегляд усіх замовлень;

– фільтрація за статусами (нові, оброблені, скасовані);

– пошук за прізвищем, телефоном чи номером замовлення;

– зміна статусу та логування подій;

– окрема вкладка для незареєстрованих користувачів (гостьові замовлення).

На рисунку 3.3 показано інтерфейс адмінпанелі із таблицею замовлень, доступною для керування адміністраторами.

Адмін панель розроблена з використанням тієї ж компонентної моделі Vue.js, що забезпечує швидку навігацію без перезавантаження сторінок.

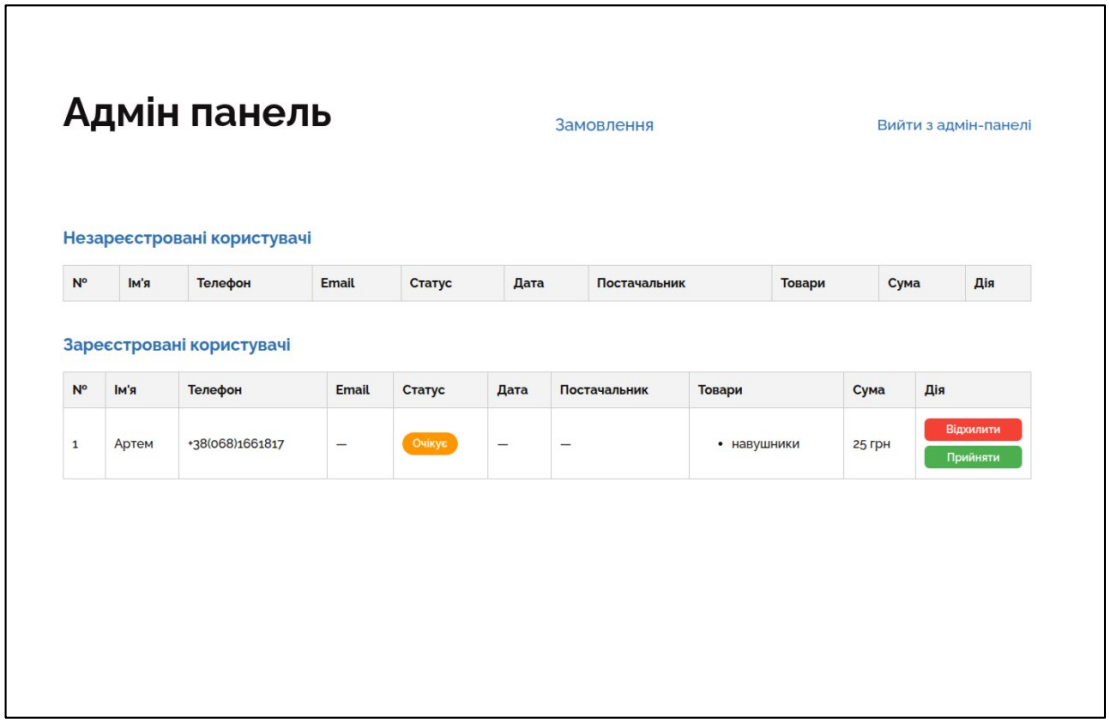



Рисунок 3.4 – Адмін–панель з замовленнями

### 3.5 Серверна логіка: Yii2

Уся бізнес–логіка системи реалізована на основі фреймворку Yii2. Було створено REST API для взаємодії з фронтендом, яке включає:

- GET /orders – список замовлень із фільтрацією;
- POST /orders – створення нового замовлення;
- PUT /orders/:id/status – зміна статусу;
- GET /products – отримання товарів;
- GET /users – перевірка прав доступу.

Моделі, контролери та фільтри побудовані за принципом MVC, що спрощує супровід і масштабування коду. Для захисту використовуються механізми RBAC (рольовий контроль доступу), а також токени авторизації для кожного користувача.

### 3.6 Загальна архітектура програмного рішення

Проєкт реалізовано у вигляді веб–застосунку, який складається з двох основних частин: серверної логіки на основі PHP–фреймворку Yii2 та клієнтської частини, побудованої з використанням фреймворку Vue.js. Для розгортання використовується система контейнеризації Docker, яка забезпечує ізоляцію кожного модуля та спрощує масштабування.

Умовно архітектуру можна поділити на такі складові:

- Backend (Yii2) – реалізує REST API, обробляє запити, зберігає дані у БД, виконує валідацію, роботу з ролями;
- Frontend (Vue.js) – SPA–інтерфейс, який взаємодіє з API та відображає динамічні дані користувачам;
- MySQL – реляційна база даних [15], де зберігаються всі ключові сутності: замовлення, користувачі, товари, постачальники;
- Redis – система кешування і черг, використовується для фонових задач;

– WebSocket–сервер – реалізований на Node.js для підтримки оновлень у реальному часі;

– Docker – забезпечує середовище виконання всіх сервісів.

## 3.7 Реалізація серверної логіки (Yii2)

Розробка серверної логіки є критичним етапом створення системи управління замовленнями у дропшипінг–моделі. У рамках бекенд–частини проєкту реалізовано REST API за допомогою фреймворку Yii2, що забезпечує взаємодію між клієнтською частиною та базою даних. Основні можливості API включають створення, перегляд, оновлення замовлень та логування змін статусів.

### 3.7.1 Реалізація моделі даних

Ключовою складовою є модель Order, яка зберігає інформацію про замовлення. Структура таблиці orders включає наступні поля:

– id – унікальний ідентифікатор;

– user_id – покупець, який створив замовлення (або null, якщо гість);

– product_id – товар;

– supplier_id – постачальник;

– status – статус (нове, підтверджене, відправлене, скасоване);

– created_at, updated_at – часові мітки.

Створення таблиці orders у базі даних:

```
$this->createTable('{{%orders}}', [
    'id' => $this->primaryKey(),
    'user_id' => $this->integer(),
    'product_id' => $this->integer()->notNull(),
    'supplier_id' => $this->integer(),
    'status' => $this->string(20)->defaultValue('new'),
    'created_at' => $this->integer()->notNull(),
    'updated_at' => $this->integer()->notNull(),
]);
```

У цьому фрагменті коду виконується створення таблиці orders, яка є центральною у системі. У ній зберігається кожне оформлене замовлення, з обов'язковим зазначенням товару (product_id) та часових міток (created_at, updated_at). Значення статусу за замовчуванням – new.

Створення таблиці users:

```
$this->createTable('{{%users}}', [
    'id' => $this->primaryKey(),
    'name' => $this->string()->notNull(),
    'email' => $this->string()->notNull()->unique(),
    'phone' => $this->string(),
    'role' => $this->string()->defaultValue('client'),
    'created_at' => $this->integer()->notNull(),
]);
```

У таблиці users зберігається інформація про всіх користувачів системи – як клієнтів, так і адміністраторів чи постачальників. Поле role визначає тип користувача, а email є унікальним ключем, що унеможливлює дублювання.

Створення таблиці product:

```
$this->createTable('{{%products}}', [
    'id' => $this->primaryKey(),
    'name' => $this->string()->notNull(),
    'description' => $this->text(),
    'price' => $this->decimal(10, 2)->notNull(),
    'stock_quantity' => $this->integer()->defaultValue(0),
    'supplier_id' => $this->integer(),
]);
```

Ця таблиця містить перелік товарів. Поле supplier_id дозволяє визначити постачальника кожного товару, а stock_quantity відповідає за кількість одиниць на складі, що дає змогу реалізувати контроль залишків.

Створення таблиці suppliers:

```
$this->createTable('{{%suppliers}}', [
    'id' => $this->primaryKey(),
    'name' => $this->string()->notNull(),
    'email' => $this->string()->notNull(),
    'region' => $this->string(),
    'is_active' => $this->boolean()->defaultValue(true),
]);
```

У цій таблиці міститься інформація про всіх постачальників, які обробляють замовлення. Статус is_active дозволяє адміністраторам деактивувати постачальника без його видалення.

Створення таблиці order_status_log:

```
$this->createTable('{{%order_status_log}}', [
    'id' => $this->primaryKey(),
    'order_id' => $this->integer()->notNull(),
    'status' => $this->string(20)->notNull(),
    'changed_by' => $this->integer(),
    'timestamp' => $this->integer()->notNull(),
]);
```

Ця таблиця використовується для логування кожної зміни статусу замовлення. Це забезпечує прозорість дій адміністраторів або постачальників та дозволяє аналізувати історію обробки замовлень.

### 3.7.2 Модель Order

Модель Order є ключовою у системі, оскільки відповідає за збереження даних про замовлення. Вона включає такі основні поля:

- id – ідентифікатор замовлення;
- user_id – зв’язок із таблицею users;
- product_id – обраний товар;
- supplier_id – постачальник товару;
- status – поточний статус замовлення (наприклад, new, processing, shipped, delivered, canceled);
- created_at, updated_at – дати створення та оновлення запису.

Клас моделі реалізовано наступним чином:

```
namespace app\models;
use yii\db\ActiveRecord;
class Order extends ActiveRecord
{
    public static function tableName()
    {
        return 'orders';
    }

    public function rules()
```

```
    {
        return [
            [['user_id', 'product_id', 'supplier_id'], 'integer'],
            [['status'], 'in', 'range' => ['new', 'processing',
'shipped', 'delivered', 'canceled']],
            [['created_at', 'updated_at'], 'safe'],
        ];
    }

    public function behaviors()
    {
        return [
            'timestamp' => [
                'class' => 'yii\behaviors\TimestampBehavior',
                'attributes' => [
                    ActiveRecord::EVENT_BEFORE_INSERT =>
['created_at', 'updated_at'],
                    ActiveRecord::EVENT_BEFORE_UPDATE =>
['updated_at'],
                ],
                'value' => time(),
            ],
        ];
    }

    public function getProduct()
    {
        return $this->hasOne(Product::class, ['id' => 'product_id']);
    }
}
```

### 3.7.3 Створення замовлення

Метод actionCreate у контролері OrderController відповідає за створення нового замовлення. Він приймає дані у форматі JSON, здійснює валідацію та зберігає інформацію до бази даних. У разі успішного збереження повертається відповідь із кодом 201, інакше – список помилок.

```
public function actionCreate()
{
    $model = new Order();
    $data = Yii::$app->request->post();

    if ($model->load($data, '') && $model->validate()) {
        $model->user_id = Yii::$app->user->id ?? null;
        $model->status = 'new';

        if ($model->save()) {
            Yii::$app->response->statusCode = 201;
```

```
            return ['success' => true, 'order_id' => $model->id];
        }
    }

    Yii::$app->response->statusCode = 422;
    return ['success' => false, 'errors' => $model->errors];
}
```

### 3.7.4 Зміна статусу

Для контролю за виконанням замовлень адміністратор має можливість змінювати їх статус. Метод actionUpdateStatus приймає PUT–запит із новим статусом і записує зміну у таблицю order_status_log, де зберігається історія змін статусів із зазначенням користувача, що їх здійснив.

```
public function actionUpdateStatus($id)
{
    $order = Order::findOne($id);
    if (!$order) {
        throw new NotFoundHttpException('Замовлення не знайдено');
    }

    if (!Yii::$app->user->can('updateOrderStatus')) {
        throw new ForbiddenHttpException('Доступ заборонено');
    }

    $newStatus = Yii::$app->request->post('status');
    $order->status = $newStatus;

    if ($order->save()) {
        $log = new OrderStatusLog([
            'order_id' => $order->id,
            'status' => $newStatus,
            'changed_by' => Yii::$app->user->id,
            'timestamp' => time(),
        ]);
        $log->save();

        return ['success' => true];
    }

    Yii::$app->response->statusCode = 422;
    return ['success' => false, 'errors' => $order->errors];
}
```

### 3.7.5 Перегляд списку замовлень

Метод actionList реалізує можливість перегляду списку замовлень з фільтрацією за статусом та підтримкою пагінації. Це особливо важливо при роботі з великою кількістю записів у базі даних.

```
public function actionList()
{
    $query = Order::find();
    $status = Yii::$app->request->get('status');

    if ($status) {
        $query->andWhere(['status' => $status]);
    }

    return new ActiveDataProvider([
        'query' => $query,
        'pagination' => ['pageSize' => 20],
        'sort' => ['defaultOrder' => ['created_at' => SORT_DESC]],
    ]);
}
```

Таким чином, серверна логіка повністю покриває базову функціональність управління замовленнями: створення, зміну статусів та перегляд із фільтрацією. Усі дії захищені через RBAC–механізм та валідацію даних, що підвищує безпеку системи та її надійність.

## 3.8 Тестування системи на прикладі замовлення Arduino Uno

Для перевірки працездатності реалізованої системи було проведено функціональне тестування на прикладі оформлення замовлення Arduino Uno. Тестування охоплювало всі ключові етапи: від взаємодії користувача з інтерфейсом до збереження замовлення у базі даних і відображення його в адміністративній панелі.

### 3.8.1 Умови тестування:

а) стан користувача: неавторизований (гостьовий режим);

б) обраний товар: Arduino Uno;

в) тип доставки: доставка Новою Поштою;

г) контактні дані:

1) ім’я: Артем Ноздряков;
2) email: artem@example.com;
3) телефон: +380501234568.

### 3.8.2 Покроковий сценарій

Покроковий сценарій наведено у таблиці 3.1.

Таблиця 3.1 – Покроковий сценарій оформлення замовлення Arduino Uno

| Крок | Дія | Опис |
|---|---|---|
| 1 | Користувач відкриває форму замовлення | Інтерфейс автоматично підтягує доступні товари з API /products |
| 2 | Обирає товар | "Arduino Uno" |
| 3 | Заповнює контактні поля | Дані валідуються на стороні клієнта (Vue.js) |
| 4 | Обирає тип доставки | Вибір зі списку: "Нова Пошта" |
| 5 | Натискає кнопку “Замовити” | Дані надсилаються HTTP POST–запитом на /orders |
| 6 | Отримує повідомлення | Успішне замовлення – повідомлення "Ваше замовлення прийнято!" |

### 3.8.3 Передача та обробка даних

Приклад структури даних, що передаються на сервер при оформленні замовлення:

```
{
  "product_id": 17,
```

```
  "name": "Артем Ноздряков",
  "email": "artem@example.com",
  "phone": "+380501234567",
  "delivery_type": "nova_poshta"
}
```

Обробка запиту на сервері відбувається у методі actionCreate контролера OrderController. На першому етапі дані, отримані від клієнта, проходять валідацію: перевіряється коректність формату електронної пошти, наявність номера телефону, а також чи існує товар з відповідним product_id у базі даних. У разі успішної перевірки система створює новий запис у таблиці orders зі статусом new. Після цього автоматично генерується додатковий запис у таблиці order_status_log, який фіксує факт створення замовлення та його початковий статус.

### 3.8.4 Результати тестування

У результаті виконання тестового сценарію система успішно обробила замовлення Arduino Uno. Дані пройшли валідацію, замовлення було збережено в таблиці orders зі статусом new, а також автоматично створено запис у order_status_log. Постачальника визначено автоматично на основі товару, що був попередньо доданий адміністратором.

У адміністративному інтерфейсі нове замовлення з’явилося у списку з усіма ключовими параметрами. Адміністратору доступні дії перегляду, зміни статусу, видалення замовлення та перегляду журналу змін.

На стороні користувача після оформлення було отримано повідомлення про успішне створення замовлення. Всі дані пройшли клієнтську та серверну перевірку, збоїв не виявлено.

# 4 ОХОРОНА ПРАЦІ

У процесі розробки програмного забезпечення важливо враховувати не лише технічні аспекти, а й дотримання вимог охорони праці, що гарантують безпечні умови для працівника. Робота розробника характеризується значним зоровим, емоційним та фізичним навантаженням при переважно сидячому режимі праці. Незважаючи на відсутність безпосередньої фізичної небезпеки, ігнорування санітарно–гігієнічних та ергономічних вимог може призвести до професійних захворювань, таких як порушення зору, хребта, перенапруження нервової системи.

Основними факторами, що впливають на здоров’я програміста, є тривала робота за комп’ютером, недостатнє освітлення, неправильно організоване робоче місце, відсутність перерв у роботі. Відповідно до нормативних документів (ДСанПіН 3.3.2.007–98, Закон України "Про охорону праці", ДСТУ ISO 45001:2019), робоче місце повинно бути організоване з дотриманням ергономіки: зручне регульоване крісло, стіл відповідної висоти, монітор на відстані 50–70 см від очей, рівень освітленості не нижче 300 лк у зоні клавіатури. Температура повітря має становити 20–24 °C, відносна вологість – у межах 40–60%. Щоб уникнути перенапруження, рекомендовано робити короткі перерви кожні 1,5–2 години, виконувати гімнастику для очей і розминку для тіла.

Суттєвими є також вимоги електробезпеки. Усе обладнання повинно бути справним, сертифікованим і мати заземлення. Забороняється використовувати пошкоджені подовжувачі, розетки або кабелі. Для захисту від перенапруги бажано використовувати джерела безперебійного живлення. Робоче місце повинно бути обладнане первинними засобами пожежогасіння (вогнегасником), а також мати доступ до аварійного виходу та чітко визначений план евакуації. У разі виникнення надзвичайної ситуації

(замикання, задимлення, пожежа) слід вимкнути електроживлення, повідомити відповідні служби та покинути приміщення згідно з інструкцією.

До потенційно шкідливих чинників також належать електромагнітне випромінювання від монітора, шум системних блоків, перевтома, стреси через надмірне розумове навантаження. Для їх зменшення необхідно впроваджувати організаційно–профілактичні заходи – обмеження безперервної роботи за комп'ютером, використання якісного обладнання з низьким рівнем шуму та випромінювання, забезпечення належного мікроклімату, підтримка режиму праці та відпочинку [20].

Отже, при дотриманні встановлених санітарних, технічних та організаційних норм, умови праці фахівця у сфері інформаційних технологій можуть вважатись безпечними. Використання сертифікованого обладнання, правильна організація робочого місця, дотримання режиму роботи та наявність інструкцій з безпеки сприяють збереженню здоров'я працівника та створюють комфортне середовище для виконання службових обов'язків.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та реалізовано систему автоматизації управління онлайн-замовленнями. Основною метою створеної системи стало забезпечення ефективного, масштабованого та зручного інструменту, який автоматизує ключові етапи процесу обробки замовлень – від їхнього створення до зміни статусу та передачі даних відповідним виконавцям.

У процесі розробки проведено аналіз основних бізнес-процесів, характерних для онлайн-торгівлі, та визначено етапи, що доцільно автоматизувати. Побудована логіка роботи системи охоплює обробку запитів користувачів, зміну статусів замовлень і фіксацію подій на кожному етапі їх виконання.

Усі компоненти системи інтегровано в єдине контейнеризоване середовище за допомогою технології Docker, що значно спрощує налаштування, розгортання та подальше обслуговування. Проведене тестування підтвердило стабільну роботу системи, зручність користувацького інтерфейсу та відповідність реалізованої функціональності заявленим вимогам.

Розроблене програмне забезпечення орієнтоване на сферу електронної комерції та призначене насамперед для користувачів, які лише починають організовувати онлайн-продаж товарів. Система не потребує складної технічної інфраструктури, є зручною у використанні та може слугувати базою для створення власної торговельної платформи.

Таким чином, усі поставлені в роботі завдання були повністю виконані. Отримані результати підтверджують ефективність запропонованого підходу та готовність системи до подальшого вдосконалення відповідно до потреб цільової аудиторії.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Краус К.М., Манжура О.В., Краус Н.М. Електронна комерція та інтернет–торгівля. – Київ: Аграр Медіа Груп, 2021. – 454 с.

2. Шимановський О. В. Теорія та практика електронної комерції. – Київ: КМ Академія, 2018. – 432 с.

3. Що таке електронна комерція? [Електронний ресурс] / SalesWillBeBest. – Режим доступу: https://sales.willbe.best/blog/shcho–take–elektronna–komertsiia/ (дата звернення: 04.04.2025).

4. ДСТУ 3008:2015. Документація. Звіти у сфері науки і техніки: структура та правила оформлення. – К.: Держстандарт України, 2017. – 29 с.

5. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва, А.І. Бронніков. – Харків: ХНУРЕ, 2022. – 66 с.

6. Shopify. Dropshipping Business Guide [Електронний ресурс]. – Режим доступу: https://www.shopify.com/guides/dropshipping (дата звернення: 01.04.2025).

7. WooCommerce Documentation [Електронний ресурс]. – Режим доступу: https://woocommerce.com/documentation/ (дата звернення: 03.06.2025).

8. Vue.js Guide: Composition API & Components [Електронний ресурс]. – Режим доступу: https://vuejs.org/guide/ (дата звернення: 02.04.2025).

9. Redis Documentation [Електронний ресурс]. – Режим доступу: https://redis.io/docs/ (дата звернення: 02.04.2025).

10. Офіційна документація Yii2 Framework [Електронний ресурс]. – Режим доступу: https://www.yiiframework.com/doc/guide/2.0/en (дата звернення: 03.04.2025).

11. Docker Docs: Containers, Images, Compose [Електронний ресурс]. – Режим доступу: https://docs.docker.com/ (дата звернення: 03.04.2025).

12. FreehostWiki. Основи PHP: поняття, можливості, застосування [Електронний ресурс]. – Режим доступу: https://freehost.com.ua/ukr/faq/wiki/chto–takoe–php/ (дата звернення: 07.04.2025).

13. FreehostWiki. Що таке phpMyAdmin? [Електронний ресурс]. – Режим доступу: https://freehost.com.ua/ukr/faq/wiki/chto–takoe–phpmyadmin/ (дата звернення: 06.04.2025).

14. Що таке JavaScript і для чого він використовується [Електронний ресурс] / GoIT. – Режим доступу: https://goit.global/ua/articles/shcho–take–javascript–i–dlia–choho–vin–potriben/ (дата звернення: 07.04.2025).

15. Модель «сутність–зв'язок» у базах даних: навчальний матеріал [Електронний ресурс] / КПІ. – Режим доступу: https://web.kpi.kharkov.ua/dpm/wp–content/uploads/sites/123/2020/05/LEKTSIYA–10–11–OBD_ukr.pdf (дата звернення: 08.04.2025).

16. UML–базові поняття: KDE Umbrello Manual [Електронний ресурс]. – Режим доступу: https://docs.kde.org/trunk5/uk/umbrello/umbrello/umlbasics.html (дата звернення: 05.04.2025).

17. Як будувати UML–діаграми для IT–проєктів [Електронний ресурс] / DOU.ua. – Режим доступу: https://dou.ua/forums/topic/40575/ (дата звернення: 06.04.2025).

18. Іванов С. П. Проектування програмного забезпечення: навчальний посібник. – Київ: Центр учбової літератури, 2019. – 288 с.

19. Невлюдов І.Ш., Андрусевич А.О. та ін. Технічні засоби автоматизації: Підручник. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 366 с.

20. Закон України «Про охорону праці» [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 05.06.2025).