Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної інженерії та охоронних систем

(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБЛІКУ ІНВЕНТАРЮ НА ОСНОВІ QR-ІДЕНТИФІКАЦІЇ З ВЕБ-ІНТЕРФЕЙСОМ УПРАВЛІННЯ

INTELLIGENT INVENTORY MANAGEMENT SYSTEM BASED ON QR IDENTIFICATION WITH A WEB-BASED MANAGEMENT INTERFECE

спеціальність 123 Комп'ютерна інженерія

(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія

(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Гетманенко Владислав Юрійович

(підпис)

Керівник:
к.т.н., доцент
Бортник Катерина Яківна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» червня 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та охоронних систем*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
«23» 12 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Гетманенка Владислава Юрійовича
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інтелектуальна система обліку інвентарю на основі QR-ідентифікації з веб-інтерфейсом управління*

Керівник роботи *к.т.н., доцент Бортник Катерина Яківна*

затверджені наказом закладу вищої освіти від «20» грудня 2025 року № 536/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2026 р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз предметної області та методів ідентифікації
Проектування архітектури інтелектуальної системи
Реалізація та розгортання системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Схема алгоритму ідентифікації та обробки даних через QR-інтерфейс
Схема розгортання компонентів системи у середовищі Docker
Вебінтерфейс управління системою обліку
3D-модель та складальне креслення апаратних елементів ідентифікації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Бортник К. Я., к.т.н., доцент*		
Теоретичне дослідження та практична реалізація	*Бортник К. Я., к.т.н., доцент*		
Практична реалізація об'єкта проектування	*Бортник К. Я., к.т.н., доцент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Лавренчук С. В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О. І., ст. викладач*		

7. Дата видачі завдання 23.12.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2026 р.	
2.	*Аналіз проблеми за темою роботи та постановка завдань дослідження*	до 02.03.2026 р.	
3.	*Теоретичне дослідження та практична реалізація*	до 02.04.2026 р.	
4.	*Практична реалізація об'єкта проектування та формування додатків*	до 10.04.2026 р.	
5.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 01.05.2026 р.	
6.	*Нормоконтроль*	до 30.05.2026 р.	
7.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2026 р.	
8.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2026 р.	

Здобувач вищої освіти _____ Владислав ГЕТМАНЕНКО
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Катерина БОРТНИК
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Гетманенко В. Ю. Інтелектуальна система обліку інвентарю на основі QR-ідентифікації з веб-інтерфейсом управління. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків та списку використаних джерел.

Перший розділ присвячено аналізу сучасних методів автоматизації складського обліку та управління активами. Обґрунтовано актуальність впровадження інтелектуальних систем у бізнес-процеси, розглянуто переваги технологій QR-кодування та NFC у порівнянні з традиційними методами штрих-кодування. Проведено порівняльний аналіз існуючих ERP та інвентаризаційних систем, виявлено їхні недоліки для сегмента малого та середнього бізнесу.

Другий розділ обґрунтовує вибір стеку: Django Rest Framework (API), React (UI), Docker та Nginx. Описано проєктування апаратної складової — ергономічних кріплень для ідентифікаторів, створених за допомогою FDM-друку.

Третій розділ присвячено практичній реалізації інтелектуальної системи. Описано структуру бази даних PostgreSQL, алгоритми генерації та зчитування динамічних QR-кодів, а також логіку взаємодії клієнтської та серверної частин. Розглянуто процес тестування системи в умовах реального навантаження та запропоновано шляхи подальшої модернізації, зокрема впровадження AI-аналітики для прогнозування залишків інвентарю.

Ключові слова: QR-ідентифікація, NFC, Django Rest Framework, React, Docker, інвентаризація, 3D-друк, PostgreSQL.

ANNOTATION

Hetmanenko V. Intelligent Inventory Management System Based on QR Identification with a Web-Based Management Interface. Manuscript.

Qualifying work of a bachelor of EP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2026.

Qualification work consists of an introduction, three sections, conclusions, and a list of references.

The first section is devoted to the analysis of modern methods of warehouse accounting automation and asset management. The relevance of introducing intelligent systems into business processes is substantiated, and the advantages of QR-coding and NFC technologies in comparison with traditional barcoding methods are considered. A comparative analysis of existing ERP and inventory systems was conducted, and their shortcomings for the small and medium-sized business segment were revealed.

In the second section, the selection of the system architecture and development tools is justified. The advantages of using the Django Rest Framework for building a scalable API and the React library for creating a dynamic interface are considered. The principles of containerization using Docker and Nginx server configuration are described. Special attention is paid to the design of the hardware component: the development of ergonomic mounts for tags manufactured using FDM printing technology and the integration of NFC modules for rapid object identification.

The third section is devoted to the practical implementation of the intelligent system. The structure of the PostgreSQL database, algorithms for generating and reading dynamic QR codes, and the logic of interaction between the client and server parts are described. The process of testing the system under real load conditions is considered and ways of further modernization are proposed, in particular, the introduction of AI-analytics for predicting inventory balances.

Keywords: QR identification, NFC, Django Rest Framework, React, Docker, inventory, 3D printing, PostgreSQL.

ЗМІСТ

ВСТУП 20
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ ІДЕНТИФІКАЦІЇ 22
1.1 Аналіз сучасних процесів автоматизації складського обліку та управління інвентарем 22
1.2 Огляд та порівняльний аналіз технологій автоматичної ідентифікації 24
1.2.1 Технологія штрих-кодування та її обмеження 24
1.2.2 QR-коди як ефективний засіб зберігання та зчитування даних 25
1.2.3 Радіочастотна ідентифікація (NFC/RFID) у системах контролю 25
1.3 Технічні аспекти використання адитивних технологій (3D-друк) для створення елементів апаратної інфраструктури. 27
1.4 Обґрунтування вибору архітектури ПЗ та інструментальних засобів розробки 28
1.4.1 Аналіз переваг клієнт-серверної архітектури на основі REST API 28
1.4.2 Порівняння фреймворків для Backend та Frontend розробки 30
1.4.3 Використання Docker-контейнеризації для забезпечення переносимості системи 31
1.5. Постановка завдання на проектування інтелектуальної системи 32
РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ 34
2.1 Формування функціональних та технічних вимог до системи 34
2.2 Розробка архітектури програмного комплексу та моделі взаємодії компонентів 35
2.2.1 Проектування структури взаємодії клієнт-сервер за протоколом REST 36
2.3 Проектування логічної та фізичної структури бази даних 36
2.4 Розробка алгоритму ідентифікації та обробки даних через QR-інтерфейс. 38
2.5 Конструкторське проектування апаратних модулів для 3D-друку 39
2.5.1 Обґрунтування конструкції тримачів та захисних кейсів для QR/NFC ідентифікаторів 39

2.5.2 Створення тривимірних моделей компонентів «розумного прилавка» у середовищі автоматизованого проектування 40
2.6 Організація мережевої інфраструктури та контейнеризації компонентів . 40
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА РОЗГОРТАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ 42
3.1 Технічна реалізація серверної частини на базі Django Rest Framework..... 42
3.1.1 Розробка моделей даних та ORM-запитів 42
3.1.2 Реалізація серіалізаторів та представлень (Views) 43
3.1.3 Система автентифікації та розмежування прав доступу 43
3.1.4 Інтеграція інструментів тестування та документації API 43
3.3 Практична реалізація апаратної частини та елементів ідентифікації 45
3.3.1 Параметри 3D-друку та підготовка моделей у програмах-слайсерах .. 45
3.3.2 Виготовлення та маркування фізичних тримачів для інвентарю 46
3.4 Конфігурування середовища розгортання та мережевих сервісів..... 46
3.4.1 Створення Docker-образів та налаштування сценарію Docker Compose 47
3.4.2 Налаштування вебсервера Nginx для забезпечення доступу до системи 48
3.5 Тестування працездатності системи та аналіз результатів..... 48
ВИСНОВКИ 50
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 52

ВСТУП

Актуальність теми. У сучасних умовах цифровізації бізнес-процесів ефективне управління матеріальними активами є критично важливим для підприємств будь-якого масштабу. Традиційні методи обліку інвентарю, що базуються на паперових носіях або ручному введенні даних у таблиці, призводять до виникнення помилок, втрати часу та відсутності оперативного контролю. Використання технологій автоматичної ідентифікації, зокрема QR-кодів, дозволяє мінімізувати вплив людського фактору та забезпечити миттєвий доступ до інформації про товар. Розробка інтелектуальної системи, що інтегрує веб-інтерфейс управління з апаратними засобами маркування (виготовленими за допомогою адитивних технологій), є актуальним інженерним завданням, яке дозволяє створити доступне та гнучке рішення для автоматизації складського обліку.

Метою роботи є розробка інтелектуальної програмно-апаратної системи обліку інвентарю на основі QR-ідентифікації з використанням сучасного веб-інтерфейсу управління для автоматизації процесів моніторингу, контролю та менеджменту товарних залишків.

Об’єкт дослідження – процеси автоматизації обліку та контролю матеріальних ресурсів у розподілених інформаційних системах.

Предмет дослідження – архітектура, методи ідентифікації та програмно-апаратні засоби реалізації інтелектуальної системи управління інвентарем через веб-інтерфейс.

Завдання, які необхідно виконати:

дослідити сучасні методи автоматичної ідентифікації об’єктів (QR, NFC, штрих-кодування) та обґрунтувати вибір технологічного стеку для розробки клієнт-серверної архітектури системи;

спроектувати архітектуру системи, включаючи структуру бази даних PostgreSQL, логіку взаємодії компонентів через API (Django Rest Framework) та модель фізичних носіїв ідентифікаторів для 3D-друку;

розробити серверне програмне забезпечення на основі фреймворку Django та інтерактивний клієнтський додаток на базі React для візуалізації даних і сканування ідентифікаторів;

реалізувати апаратну складову системи у вигляді фізичного прилавка з маркованими комірками та спеціалізованими тримачами для QR-кодів, виготовленими на 3D-принтері;

запропонувати методику розгортання системи у контейнеризованому середовищі Docker з використанням веб-сервера Nginx для забезпечення відмовостійкості та безпеки обробки даних.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ ІДЕНТИФІКАЦІЇ

1.1 Аналіз сучасних процесів автоматизації складського обліку та управління інвентарем

Сучасний стан розвитку інформаційних технологій диктує необхідність трансформації традиційних підходів до управління матеріальними активами. Складський облік та інвентаризація є критичними вузлами в архітектурі будь-якого підприємства чи організації, оскільки від точності даних про наявність ресурсів залежить ефективність прийняття управлінських рішень.

Традиційні методи обліку, які базуються на ручному введенні даних (паперові журнали, електронні таблиці типу MS Excel), характеризуються низькою швидкістю обробки інформації та високою ймовірністю виникнення помилок, спричинених людським фактором. Основними проблемами таких методів є:

- низька оперативність: затримка між фактичним рухом товару та відображенням цієї події в системі;
- складність інвентаризації: необхідність повної зупинки робочих процесів для перевірки фактичної наявності майна;
- відсутність прозорості: неможливість відстежити історію переміщень об'єкта в реальному часі.

Процес автоматизації складського обліку передбачає впровадження спеціалізованого програмно-апаратного комплексу, що забезпечує автоматичний збір, передачу та обробку даних про кожну одиницю інвентарю. У сучасних інтелектуальних системах автоматизація базується на трьох основних принципах:

- унікальна ідентифікація: кожна одиниця товару отримує цифровий дескриптор, що дозволяє системі розрізняти ідентичні предмети;
- централізація даних: використання клієнт-серверної архітектури дозволяє зберігати всю інформацію в єдиній базі даних (наприклад, PostgreSQL), забезпечуючи доступ до неї з будь-якої точки мережі;

– інтелектуальна взаємодія: система не просто зберігає дані, а аналізує їх, сигналізуючи про критичні залишки, терміни експлуатації або несанкціоновані переміщення.

В контексті комп'ютерної інженерії, автоматизація обліку на сучасному етапі зміщується в бік використання веб-орієнтованих інтерфейсів та мобільних пристроїв. Це дозволяє відмовитися від спеціалізованих дорогих терміналів збору даних (ТЗД) на користь звичайних смартфонів, що інтегруються в систему через браузер або API (Рис 1.1).

Рисунок 1.1 – Ілюстрація складської системи з елементами автоматизації

Особливого значення набуває створення «розумних робочих місць» або прилавків, де апаратна частина (датчики, камери, мітки) тісно інтегрована з програмним кодом. Застосування адитивних технологій (3D-друку) у цьому процесі дозволяє створювати індивідуальну інфраструктуру для кожного робочого місця, що робить систему гнучкою та адаптивною до специфічних потреб конкретного складу чи магазину.

Таким чином, перехід від ручного обліку до інтелектуальних систем ідентифікації є необхідним кроком для забезпечення конкурентоспроможності та

надійності зберігання активів, що потребує розробки комплексного рішення, яке поєднує веб-технології, бази даних та фізичні засоби ідентифікації.

1.2 Огляд та порівняльний аналіз технологій автоматичної ідентифікації

Вибір методу ідентифікації є ключовим етапом проектування системи обліку, оскільки він визначає апаратні вимоги, вартість впровадження та швидкість роботи кінцевого користувача. У сучасній інженерній практиці виділяють три основні напрямки автоматичної ідентифікації об'єктів.

1.2.1 Технологія штрих-кодування та її обмеження

Штрихове кодування (лінійні коди, наприклад EAN-13) є найстарішим стандартом автоматизації. Його робота базується на зчитуванні послідовності чорних та білих смуг різної ширини, які кодують лише набір цифр (Рис 1.2).

Рисунок 1.2 – Приклад штрих-коду

Серед переваг - це низька вартість друку та широке розповсюдження в ритейлі.

Обмеження: мала ємність даних (зазвичай до 20-30 символів), неможливість зчитування при пошкодженні навіть невеликої частини коду та необхідність

використання спеціалізованих лазерних сканерів. Для інтелектуальних систем управління, де потрібно кодувати складні URL-посилання або унікальні UUID об'єктів, можливостей лінійних кодів часто недостатньо.

1.2.2 QR-коди як ефективний засіб зберігання та зчитування даних

QR-код (Quick Response code) - це двовимірний матричний код, який став стандартом для мобільних та веб-орієнтованих систем (Рис 1.3).

Рисунок 1.3 – Приклад QR-коду

На відміну від штрих-коду, він зберігає інформацію як по горизонталі, так і по вертикалі.

- Висока ємність: можливість кодування до кількох тисяч символів, що дозволяє передавати складні структури даних або зашифровані токени.
- Корекція помилок: алгоритми Ріда-Соломона дозволяють успішно зчитувати код навіть при його пошкодженні на 30%.
- Доступність: для зчитування не потрібне спеціальне обладнання - достатньо камери смартфона або веб-камери, інтегрованої у веб-інтерфейс (через JavaScript API), що ідеально підходить для реалізації системи на стеку React.

1.2.3 Радіочастотна ідентифікація (NFC/RFID) у системах контролю

Технологія базується на передачі даних через радіосигнали між міткою та зчитувачем. NFC (Near Field Communication) є підмножиною RFID, що працює на коротких відстанях (до 10 см).

Переваги: можливість зчитування «без прямої видимості», високий рівень захисту даних, можливість перезапису інформації безпосередньо в пам'ять мітки.

Недоліки: значно вища вартість однієї мітки порівняно з друкованим кодом та необхідність наявності NFC-модуля в пристрої зчитування. Порівняльна характеристика методів ідентифікації (табл 1.1).

Таблиця 1.1 - Порівняльний аналіз технологій ідентифікаціїII

Параметр порівняння	Штрих-код	QR-код	NFC/RFID
Тип зчитування	Оптичний (лазер)	Оптичний (камера)	Радіочастотний
Обсяг даних	Низький	Високий	Середній/Високий
Вартість мітки	Мінімальна	Мінімальна	Висока
Стійкість до бруду	Низька	Висока	Абсолютна
Обладнання	Сканер	Смартфон/Камера	NFC-модуль

1.3 Технічні аспекти використання адитивних технологій (3D-друк) для створення елементів апаратної інфраструктури

Використання адитивних технологій, зокрема методу пошарового наплавлення пластику (FDM - Fused Deposition Modeling), у розробці інтелектуальних систем обліку дозволяє подолати розрив між програмним забезпеченням та фізичним середовищем. У межах даної роботи 3D-друк виступає інструментом для швидкого прототипування та виготовлення індивідуальних елементів апаратної інфраструктури, що забезпечують надійне розміщення ідентифікаторів на об’єктах обліку.

Основним технічним аспектом впровадження 3D-друку є можливість створення кастомних корпусів, тримачів та кріплень, які враховують специфіку експлуатації інвентарю. На відміну від стандартних заводських рішень, власне проектування дозволяє інтегрувати посадокові місця для NFC-міток безпосередньо всередину конструкції, захищаючи їх від механічних пошкоджень та впливу навколишнього середовища. Це особливо важливо для створення «розумного прилавка», де кожен елемент маркування має бути зафіксований у чітко визначеному положенні для зручного зчитування оптичними сенсорами або смартфонами.

По-друке, важливим фактором є варіативність вибору матеріалів, де застосування специфічних полімерів дозволяє диференціювати компоненти за їхнім призначенням: використання PLA-пластику є раціональним для стаціонарних підставок, тоді як PETG забезпечує необхідну зносостійкість для елементів, що піддаються постійному механічному впливу. Це безпосередньо впливає на оптимізацію вартості та експлуатаційний термін апаратного забезпечення.

Нарешті, використання адитивних технологій забезпечує високу масштабованість та адаптивність системи. У випадку модернізації обладнання, зміни габаритів інвентарю або переходу на нові формати ідентифікаторів, інженер має можливість оперативно скоригувати CAD-моделі та отримати готові оновлені

вузли протягом мінімального часового проміжку, що відповідає сучасним принципам швидкого прототипування та ітеративної розробки.

Інтеграція 3D-друкованих елементів у систему обліку перетворює звичайний веб-додаток на повноцінний апаратно-програмний комплекс. Це дозволяє не лише маркувати товари, а й створювати фізичну інфраструктуру - спеціалізовані комірки, слоти та модульні органайзери, які на апаратному рівні впорядковують процес зберігання та видачі майна. Таким чином, адитивні технології забезпечують гнучкість фізичного рівня системи, що в поєднанні з інтелектуальним веб-інтерфейсом створює цілісну екосистему управління інвентарем.

1.4 Обґрунтування вибору архітектури ПЗ та інструментальних засобів розробки

Вибір архітектурного стилю та програмного стеку безпосередньо впливає на масштабованість, безпеку та швидкість розгортання інтелектуальної системи обліку. Для реалізації проєкту було обрано сучасний підхід, що базується на розділенні рівнів представлення та бізнес-логіки.

1.4.1 Аналіз переваг клієнт-серверної архітектури на основі REST API

Під час проектування інтелектуальної системи обліку інвентарю ключовим етапом є вибір архітектурного стилю, який забезпечить стабільність та гнучкість взаємодії між різними компонентами програмного комплексу. Для реалізації поставленого завдання було обрано архітектурний стиль REST (Representational State Transfer), який базується на клієнт-серверній моделі взаємодії. Головною особливістю та перевагою такого підходу є повна децентралізація та незалежність фронтенд-частини, реалізованої на базі бібліотеки React [14], від серверної бізнес-логіки, що функціонує на базі Django Rest Framework [4]. Це дозволяє розділити відповідальність між компонентами: сервер фокусується на обробці даних та забезпеченні цілісності бази даних PostgreSQL [11], тоді як клієнтська частина відповідає виключно за візуалізацію та взаємодію з користувачем (Рис 1.4).

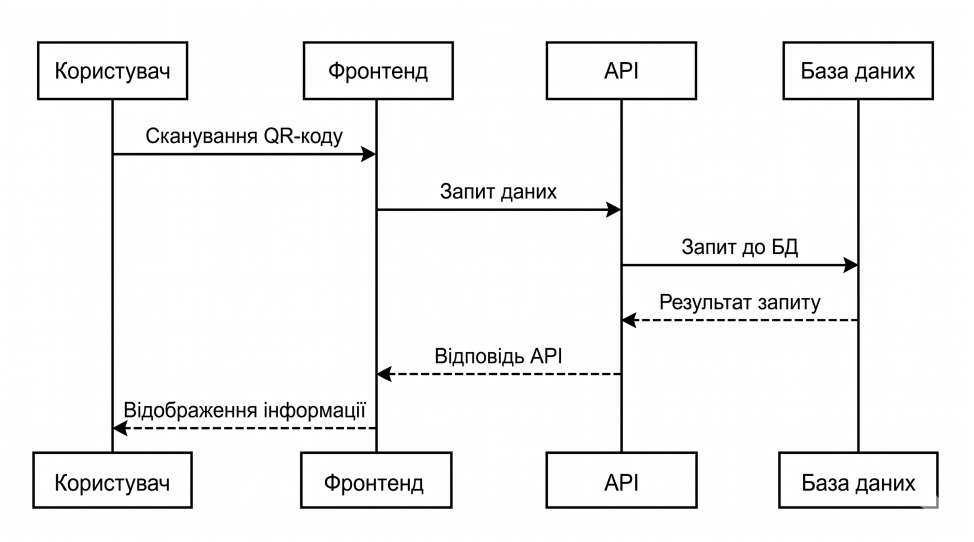

Рисунок 1.4 – Діаграма послідовності взаємодії компонентів системи

Універсальність обраної моделі полягає в тому, що розроблений інтерфейс програмування додатків (API) перетворюється на єдину точку доступу до даних, яка здатна обслуговувати не лише веб-інтерфейс, а й потенційні мобільні застосунки або спеціалізовані термінали на базі мікроконтролерів для зчитування QR-кодів [10]. Використання мови Python 3.14 [13] для розробки серверної частини дозволяє впроваджувати принципи «Robust Python» для забезпечення надійності типів даних та стабільності API [15]. Такий підхід гарантує високу масштабованість системи, оскільки він дозволяє незалежно оптимізувати серверні потужності, контейнеризовані за допомогою Docker [5, 6], та клієнтські ресурси відповідно до зростаючих потреб підприємства.

Стандартизація взаємодії через протокол HTTP та використання формату обміну даними JSON забезпечує високу швидкість передачі інформації та спрощує процес тестування і налагодження системи за допомогою спеціалізованих інструментів, таких як Postman [12]. Завдяки відсутності стану (stateless) у REST-запитах кожен запит від клієнта містить всю необхідну інформацію для його обробки, що значно знижує навантаження на пам'ять сервера та підвищує загальну відмовостійкість архітектури. Для безпечного доступу до ресурсів API доцільно використовувати механізми автентифікації на основі JWT-токенів, що є стандартом для сучасних розподілених систем [16]. Таким чином, поєднання REST API з сучасними веб-технологіями створює надійний фундамент для побудови

інтелектуальної системи, здатної ефективно функціонувати в умовах сучасних комп'ютерних мереж.

1.4.2 Порівняння фреймворків для Backend та Frontend розробки

Процес проектування інтелектуальної системи обліку інвентарю вимагає ретельного аналізу існуючих технологічних рішень для забезпечення максимальної продуктивності та безпеки даних. На етапі вибору платформи для реалізації серверної логіки було проведено детальне порівняння сучасних середовищ розробки. Зокрема, розглядалася платформа Node.js та фреймворк Django [3] на базі мови програмування Python [13]. Хоча платформа Node.js демонструє високі показники швидкості при обробці великої кількості асинхронних запитів, остаточний вибір було зроблено на користь Django Rest Framework [4].

Головним аргументом на користь цього рішення став високий рівень вбудованої безпеки та надійності, що підтверджується документацією та стандартами розробки стабільних систем [15, 18]. Django містить механізми захисту від критичних вразливостей, таких як SQL-ін’єкції, CSRF та XSS, безпосередньо в базовій конфігурації. Крім того, наявність потужного інструментарію об'єктно-реляційного відображення (ORM) дозволяє значно пришвидшити процес проектування складних реляційних структур у базі даних PostgreSQL [11], що є критично важливим для стабільної роботи системи інвентаризації.

Аналогічний підхід було застосовано при виборі фронтенд-інструментарію, де проводилося порівняння між фреймворком Vue.js [20] та бібліотекою React [14]. Перевага була надана React завдяки її розвиненій екосистемі та наявності великої кількості перевірених спільнотою рішень для роботи з периферійними пристроями, зокрема камерами мобільних девайсів для зчитування QR-кодів [10]. Використання компонентного підходу дозволяє реалізувати модульну архітектуру інтерфейсу, що спрощує подальшу підтримку коду.

Особливістю даного проєкту є інтеграція програмної частини з апаратними засобами ідентифікації. Для створення фізичних носіїв ідентифікаторів (корпусів для NFC-чіпів або захищених QR-міток) використовуються технології адитивного виробництва на базі обладнання Bambu Lab [1]. Вибір матеріалу PLA та

специфічних налаштувань друку обґрунтований потребою в довговічності маркування в умовах експлуатації [19], а підготовка моделей здійснюється за допомогою спеціалізованого ПЗ для генерації 3D-придатних QR-кодів [17]. Таке поєднання технологій DRF та React у синергії з контейнеризацією Docker [2, 5] забезпечує створення відмовостійкого програмно-апаратного комплексу, що відповідає сучасним стандартам комп'ютерної інженерії.

1.4.3 Використання Docker-контейнеризації для забезпечення переносимості системи

Для забезпечення стабільної роботи та швидкого розгортання інтелектуальної системи обліку інвентарю було обрано технологію контейнеризації Docker [5]. У межах комп’ютерної інженерії застосування даної технології є критично важливим для вирішення проблеми сумісності програмних залежностей, оскільки вона гарантує ідентичність середовища виконання на різних типах обчислювальних вузлів: від локальних станцій розробника до компактних вбудованих систем, що можуть виконувати роль контролерів доступу. Такий підхід усуває конфлікти версій бібліотек та спрощує процес підтримки програмного продукту на всіх етапах його життєвого циклу.

Однією з ключових переваг використання Docker у межах даної кваліфікаційної роботи є повна ізоляція компонентів системи. Завдяки цьому серверна частина на базі Django [3], клієнтський інтерфейс на React [14], база даних PostgreSQL [11] та вебсервер Nginx [2] функціонують у відокремлених контейнерах. Це не лише запобігає виникненню міжпрограмних конфліктів, а й підвищує загальний рівень кібербезпеки системи. Використання інструментарію Docker Compose дозволяє описати всю архітектуру системи у єдиному конфігураційному файлі, забезпечуючи автоматизацію налаштування внутрішніх мережевих мостів та змінних оточення для стабільної взаємодії між сервісами [5, 6].

Крім того, контейнеризація забезпечує значну оптимізацію апаратних ресурсів. Порівняно з традиційними віртуальними машинами, контейнери мають мінімальне навантаження на систему (overhead), оскільки використовують спільне ядро операційної системи, що забезпечує високу швидкість запуску та низьке споживання оперативної пам'яті. Це особливо актуально для інтелектуальних

систем обліку, які потребують інтеграції з периферійним обладнанням. Наприклад, при використанні 3D-принтерів Bambu Lab [1] для виготовлення корпусів під NFC-мітки або кріплень для терміналів, Docker-контейнери дозволяють швидко розгортати локальні сервери черг друку або сервіси моніторингу стану обладнання.

Для забезпечення віддаленого доступу до розгорнутої інфраструктури без необхідності налаштування складних мережевих шлюзів у межах підприємства, доцільно використовувати технології тунелювання, такі як ngrok [7], що дозволяє безпечно експонувати локальний Docker-контейнер з API у зовнішню мережу для тестування мобільних сканерів. У підсумку, інтеграція Docker у стек технологій створює професійний фундамент для побудови надійної та переносимої системи, що відповідає сучасним стандартам розробки складних програмно-апаратних комплексів.

1.5. Постановка завдання на проектування інтелектуальної системи

На основі проведеного аналізу предметної області та порівняння існуючих технологічних рішень, головним завданням даної роботи є проектування та реалізація цілісної інтелектуальної системи автоматизованого обліку інвентарю. Дана система має бути побудована як апаратно-програмний комплекс, що інтегрує сучасні веб-технології з фізичними методами ідентифікації об’єктів.

Основним функціональним призначенням системи є забезпечення повного життєвого циклу обліку матеріальних цінностей - від моменту їх реєстрації в базі даних до оперативного контролю переміщень та списання. Програмна частина системи повинна надавати користувачеві зручний вебінтерфейс, який дозволяє здійснювати моніторинг складських залишків у реальному часі, генерувати унікальні ідентифікатори у форматі QR-кодів та забезпечувати рольову модель доступу для адміністраторів і менеджерів. Важливою технічною вимогою є реалізація механізму зчитування кодів безпосередньо через браузер за допомогою камери мобільного пристрою, що виключає необхідність придбання спеціалізованого обладнання.

Апаратна складова системи передбачає створення фізичної інфраструктури для зберігання та маркування товарів. Це завдання включає розробку та виготовлення за допомогою адитивних технологій спеціалізованих тримачів, корпусів або органайзерів, які забезпечують надійне закріплення QR-ідентифікаторів та NFC-міток на об’єктах обліку. Проектування цих елементів має враховувати ергономіку процесу сканування та захист міток від зносу.

Кінцевим результатом проектування має стати масштабована система, розгорнута в контейнеризованому середовищі, яка дозволяє автоматизувати роботу «розумного прилавка». Система повинна гарантувати цілісність даних, високу швидкість відгуку інтерфейсу та можливість швидкої адаптації під різні типи інвентарю. Таким чином, розробка охоплює створення структури бази даних, написання серверної логіки, розробку клієнтської частини та фізичне втілення елементів ідентифікації, що в сукупності вирішує проблему неефективного ручного обліку.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АРХІТЕКТУРИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1 Формування функціональних та технічних вимог до системи

Процес проектування інтелектуальної системи обліку інвентарю розпочинається з чіткого визначення функціональних можливостей та технічних обмежень, які забезпечать стабільну роботу програмно-апаратного комплексу. Формування вимог розподіляється за трьома ключовими векторами: серверна логіка, клієнтський інтерфейс та апаратне забезпечення, що в сукупності мають утворювати єдину екосистему для автоматизації складських операцій.

Основні функціональні вимоги до серверної частини (Backend) зосереджені на забезпеченні надійної обробки даних та керуванні доступом. Сервер повинен реалізовувати механізми автентифікації користувачів на основі JWT [16], надавати REST-інтерфейс для виконання CRUD-операцій з базою даних за допомогою Django Rest Framework [4], а також забезпечувати генерацію унікальних ідентифікаторів для кожної одиниці інвентарю. Важливою вимогою є ведення логів активності та дотримання принципів типізації даних для забезпечення відмовостійкості [15]. З технічної точки зору сервер має бути оптимізований для роботи у контейнеризованому середовищі Docker [5], забезпечуючи високу швидкість обробки запитів та цілісність даних у PostgreSQL [11].

Клієнтський веб-інтерфейс (Frontend) розглядається як основний інструмент взаємодії менеджера з системою, тому ключовою вимогою до нього є адаптивність та висока продуктивність. Функціонал клієнтської частини на базі React [14] має включати візуалізацію поточного стану складу, зручну систему пошуку, а головне — інтегрований модуль для програмного зчитування QR-кодів через камеру смартфона [10]. Технічно додаток повинен забезпечувати стабільну синхронізацію з API навіть в умовах нестабільного мережевого з’єднання, що можна перевірити на етапі розробки за допомогою інструментів тунелювання [7] та тестування запитів у Postman [12].

Апаратний модуль системи висуває специфічні вимоги до фізичного рівня реалізації компонентів обліку. Оскільки ідентифікація базується на оптичних

мітках, конструкція тримачів та корпусів, виготовлених за допомогою 3D-принтерів Bambu Lab [1], має гарантувати стабільну фіксацію QR-кодів під кутом, що мінімізує світлові бліки. Технічні вимоги до апаратних засобів також включають використання зносостійких матеріалів, таких як PLA [19], та можливість інтеграції QR-кодів безпосередньо у геометрію моделі за допомогою STL-генераторів [17]. Комплексна відповідність розробки цим вимогам дозволить створити інтелектуальну систему, що відповідає сучасним стандартам комп'ютерної інженерії та придатна до експлуатації в реальних інфраструктурних умовах.

2.2 Розробка архітектури програмного комплексу та моделі взаємодії компонентів

Проектування архітектури інтелектуальної системи обліку базується на принципах модульності та слабкої пов’язаності компонентів, що є фундаментальним стандартом у сучасній комп’ютерній інженерії. Основна ідея полягає у створенні трирівневої структури, яка включає рівень зберігання даних (PostgreSQL) [11], рівень обробки бізнес-логіки (Django REST Framework) [4] та рівень представлення (React) [14]. Така організація дозволяє забезпечити паралельну роботу над фронтенд та бекенд частинами системи, спрощує процес тестування та дозволяє в майбутньому легко інтегрувати нові типи клієнтських додатків або апаратних сканерів без зміни основної логіки сервера.

2.2.1 Проектування структури взаємодії клієнт-сервер за протоколом REST

Взаємодія між веб-інтерфейсом та сервером реалізується за допомогою архітектурного стилю REST, де клієнт ініціює запити до сервера через стандартні методи протоколу HTTP [4]. Кожен ресурс у системі, будь то товар на прилавку, категорія інвентарю або профіль менеджера, має унікальний URL-ідентифікатор [3]. При скануванні QR-коду фронтенд-додаток виділяє закодований токен або ID і надсилає GET-запит до відповідної кінцевої точки (endpoint) API [14]. Сервер, отримавши запит, перевіряє права доступу користувача через механізм JWT-токенів [16], звертається до бази даних і повертає відповідь у форматі JSON [4]. Для операцій оновлення залишків товару або реєстрації нового інвентарю використовуються методи POST, PUT та PATCH [4], що забезпечує чітку структурування потоків даних та мінімізує обсяг переданої інформації, підвищуючи загальну швидкість роботи системи навіть при низькій якості мережевого з’єднання.

2.2.2 UML-діаграма прецедентів (Use Case) та діаграма послідовності (Sequence Diagram)

Для формалізації функціональних можливостей та внутрішньої логіки системи використовуються засоби уніфікованої мови моделювання UML. Діаграма прецедентів дозволяє візуалізувати ролі користувачів та їхні права у системі, де адміністратор має повний доступ до налаштувань та аналітики, а менеджер складу фокусується на операційній діяльності: скануванні ідентифікаторів, перегляді картки товару та зміні кількісних залишків. У свою чергу, діаграма послідовності деталізує часову взаємодію між об’єктами системи під час виконання конкретного сценарію, наприклад, при здійсненні інвентаризації. Вона наочно демонструє шлях повідомлення від камери пристрою, яка захоплює зображення QR-коду, через обробку логікою React, виконання асинхронного запиту до сервера Django та кінцеве відображення інформації користувачеві. Таке детальне моделювання дозволяє виявити потенційні "вузькі місця" в алгоритмах обробки даних ще на етапі проектування, забезпечуючи високу надійність програмного комплексу

2.3 Проектування логічної та фізичної структури бази даних

Ефективність функціонування інтелектуальної системи обліку безпосередньо залежить від раціональної організації даних та зв’язків між ними. Проектування бази даних у межах даної роботи передбачає перехід від концептуальної моделі до фізичної реалізації у середовищі PostgreSQL, що забезпечує надійність збереження інформації та підтримку цілісності даних при виконанні транзакцій. Логічна структура бази даних базується на кількох ключових сутностях, які відображають усі аспекти руху інвентарю та діяльності користувачів; програмна реалізація відповідних моделей даних на мові Python наведена у додатку А.

Центральною сутністю системи є «Товар», яка містить детальну інформацію про кожну одиницю майна, включаючи найменування, унікальний серійний номер, опис, категорію та поточну кількість на складі. З цією сутністю тісно пов’язана сутність «QR-мітка», яка зберігає дані про згенерований ідентифікатор, його унікальний UUID та посилання на конкретний об’єкт обліку, що дозволяє системі миттєво асоціювати сканований код із відповідним записом у базі. Для забезпечення безпеки та розмежування прав доступу впроваджено сутність «Користувач», яка зберігає облікові дані менеджерів та адміністраторів, їхні ролі та права в межах системи. Особлива увага приділяється сутності «Логи», яка автоматично фіксує кожну операцію - зміну кількості товару, створення нового запису або видачу інвентарю, зберігаючи часову мітку та ID користувача, що здійснив дію.

Фізична структура бази даних реалізується через набір пов'язаних таблиць, де зв’язки типу «один до одного» та «один до багатьох» забезпечують логічну цілісність системи. Наприклад, один товар може мати багато записів у логах, але тільки один унікальний QR-ідентифікатор. Для візуалізації цієї структури використовується ER-діаграма (Entity-Relationship diagram), яка відображає архітектуру даних, типи полів (UUID, Integer, DateTime, String) та зовнішні ключі (Foreign Keys), що зв’язують таблиці між собою. Такий підхід до проектування дозволяє мінімізувати дублювання інформації та забезпечити високу швидкість виконання запитів до бази даних, що є критично важливим для оперативного відображення інформації у вебінтерфейсі управління.

2.4 Розробка алгоритму ідентифікації та обробки даних через QR-інтерфейс

Алгоритм функціонування системи ідентифікації базується на безперервному циклі взаємодії між оптичним сенсором пристрою, клієнтським скриптом обробки зображень та серверною логікою. Процес розпочинається з активації модуля захоплення відеопотоку у веб-інтерфейсі React, який використовує API браузера для доступу до камери [14]. Клієнтська частина системи у реальному часі аналізує кадри відеопотоку за допомогою спеціалізованої бібліотеки розпізнавання образів, яка шукає специфічні маркери QR-коду - три квадрати позиціонування у кутах символіки. Як тільки графічний код стає чітким та потрапляє у фокус, алгоритм виконує дешифрування бінарних даних, витягуючи з них унікальний ідентифікатор об’єкта або пряме посилання на ресурс у базі даних. Практична реалізація сканування QR-коду у React-додатках може виконуватися із застосуванням підходів, описаних у відкритих технічних прикладах [10].

Після успішного розпізнавання ідентифікатора алгоритм переходить до фази мережевої взаємодії. Клієнтський додаток формує асинхронний запит до серверної частини системи, передаючи отриманий токен у заголовку або параметрах запиту. На стороні сервера Django REST Framework виконується процедура валідації вхідних даних та перевірка прав доступу користувача [4]. Контроль доступу реалізується через механізм JWT-токенів, що забезпечує безпечну аутентифікацію та авторизацію користувачів [16]. Алгоритм пошуку в базі даних PostgreSQL зіставляє отриманий UUID з існуючими записами в таблиці інвентарю [11]. У разі успішного збігу сервер генерує відповідь у форматі JSON, що містить повний набір атрибутів товару: назву, поточний залишок, технічні характеристики та посилання на медіафайли [4]. Якщо ж код не знайдено або він пошкоджений, алгоритм обробки виключень повертає відповідний код помилки, який візуалізується користувачеві у вигляді попередження.

Заключний етап алгоритму полягає в динамічному оновленні стану інтерфейсу без перезавантаження сторінки. Отримані від сервера дані монтуються в компоненти React, відображаючи картку товару з доступними діями, такими як

списання одиниці інвентарю або переміщення [14]. Кожна така дія ініціює новий цикл алгоритму, де після натискання кнопки підтвердження відправляється запит на зміну стану об’єкта через REST API [4]. Важливою частиною алгоритму є автоматичне логування транзакції, що відбувається паралельно з оновленням даних про товар. Таким чином, розроблений алгоритм забезпечує замкнений цикл від фізичного зчитування мітки на «розумному прилавку» до фіксації зміни залишків у цифровій базі даних, гарантуючи швидкість та прозорість кожної операції.

2.5 Конструкторське проектування апаратних модулів для 3D-друку

Перехід від віртуального обліку до фізичної взаємодії з інвентарем потребує створення надійної апаратної інфраструктури, яка забезпечить довговічність ідентифікаторів та зручність їх зчитування. Використання методів адитивного виробництва дозволяє реалізувати індивідуальний підхід до проектування кожного елемента «розумного прилавка», адаптуючи конструкцію під конкретні технічні умови експлуатації.

2.5.1 Обґрунтування конструкції тримачів та захисних кейсів для QR/NFC ідентифікаторів

Конструкція тримачів для QR-кодів повинна відповідати ряду ергономічних та технічних вимог, серед яких ключовою є забезпечення оптимального кута огляду для камери зчитувального пристрою. Проектування передбачає створення похилих площин (під кутом 45–60 градусів), що дозволяє уникнути прямого відбиття світла від ламінованої поверхні коду, яке часто стає причиною помилок дешифрування. Крім того, для дороговартісного обладнання розробляються спеціалізовані захисні кейси, що мають внутрішні порожнини для інтеграції NFC-міток. Така конструкція забезпечує приховане розміщення радіочастотного чіпа, захищаючи його від вологи, пилу та механічних впливів. Вибір полімерних матеріалів для друку (наприклад, PLA або PETG) обґрунтовується їхньою радіопрозорістю, що є критично важливим для безперешкодного проходження сигналу між NFC-антеною та смартфоном.

2.5.2 Створення тривимірних моделей компонентів «розумного прилавка» у середовищі автоматизованого проектування

Процес створення тривимірних моделей компонентів системи розділений на етапи високополігонального моделювання та інженерної оптимізації. У середовищі Autodesk 3ds Max виконується розробка базової геометрії складних вузлів «розумного прилавка», де за допомогою інструментів полігонального моделювання створюються точні посадокові місця для маркованих товарів та пази для встановлення ідентифікаторів. Використання Blender дозволяє провести фінальну доводку моделей, включаючи перевірку топології сітки та підготовку об’єктів до експорту у формат STL, який є стандартом для програм-слайсерів. Особлива увага приділяється допускам та посадкам: внутрішні розміри кейсів проектуються з урахуванням коефіцієнта усадки пластику під час друку, що гарантує щільну фіксацію NFC-чіпів та паперових носіїв QR-кодів без необхідності додаткового склеювання. Результатом проектування є набір цифрових моделей, що повністю готові до виробництва та забезпечують цілісність програмно-апаратного комплексу на фізичному рівні.

2.6 Організація мережевої інфраструктури та контейнеризації компонентів

Завершальним етапом проектування архітектури інтелектуальної системи є побудова надійної мережевої інфраструктури, яка забезпечить стабільну взаємодію між усіма програмними модулями. Враховуючи складність обраного стеку технологій, для забезпечення переносимості та ізоляції сервісів застосовано метод контейнеризації на базі платформи Docker [5]. Це дозволяє абстрагувати програмне забезпечення від особливостей операційної системи, створюючи ідентичні умови для роботи API, бази даних та веб-інтерфейсу незалежно від апаратної конфігурації сервера.

Проектування взаємодії компонентів реалізується через створення багатоконтейнерної архітектури, де кожен сервіс функціонує в окремому ізольованому середовищі [5]. База даних PostgreSQL виділяється в окремий контейнер з постійним сховищем даних (volumes), що гарантує збереження

інформації при перезавантаженні системи [11]. Backend-додаток на Django REST Framework [4] та Frontend-частина на React [14] також розміщуються у власних контейнерах, взаємодіючи між собою через внутрішню віртуальну мережу Docker Bridge [5]. Така організація дозволяє мінімізувати ризики виникнення конфліктів між залежностями різних модулів та забезпечує високу безпеку, оскільки прямий доступ із зовнішньої мережі до бази даних залишається заблокованим.

Ключову роль у розподілі мережевого трафіку відіграє веб-сервер Nginx, який налаштовується як зворотний проксі-сервер (reverse proxy) [2]. Його функція полягає у прийманні вхідних HTTP-запитів від користувачів та їх маршрутизації до відповідних контейнерів: статичні файли фронтенд-додатка віддаються безпосередньо, а запити до API перенаправляються на внутрішній порт сервера Django [2]. Окрім маршрутизації, Nginx забезпечує базовий рівень безпеки, дозволяючи налаштувати ліміти на кількість запитів та приховувати внутрішню структуру мережі від зовнішнього спостерігача [2]. Використання інструменту Docker Compose дозволяє автоматизувати процес розгортання цієї інфраструктури, визначаючи порядок запуску контейнерів та параметри їх зв’язку [5], що робить систему готовою до швидкого масштабування та легкого обслуговування в умовах реальної інженерної експлуатації.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА РОЗГОРТАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1 Технічна реалізація серверної частини на базі Django Rest Framework

Реалізація серверної частини системи базується на принципах високої модульності, що забезпечується фреймворком Django [3] та розширенням Django Rest Framework (DRF) [4]. Основним завданням бекенд-складової є обробка бізнес-логіки обліку, взаємодія з базою даних PostgreSQL [11] та надання стабільного API для клієнтської частини [4].

3.1.1 Розробка моделей даних та ORM-запитів

Процес розробки розпочався з опису моделей у файлах models.py (повний лістинг коду представлено у додатку А), які відображають спроектовану раніше структуру бази даних (Рис 3.1). Використання Django ORM дозволило абстрагуватися від прямого написання SQL-запитів, що прискорило розробку та підвищило безпеку системи [3].

```python
class InventoryItem(models.Model):
    id = models.UUIDField(
        "Ідентифікатор",
        primary_key=True,
        default=uuid.uuid4,
        editable=False,
    )
    name = models.CharField("Назва товару", max_length=255)
    photo = models.ImageField("Фото файлом", upload_to="inventory/items/", blank=True)
    photo_url = models.URLField("Фото", blank=True)
    category = models.ForeignKey(
        Category,
        verbose_name="Категорія",
        related_name="items",
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
    )
    quantity = models.PositiveIntegerField("Кількість", default=0)
    price = models.DecimalField("Ціна за одиницю", max_digits=10, decimal_places=2)
    created_at = models.DateTimeField("Створено", auto_now_add=True)
    updated_at = models.DateTimeField("Оновлено", auto_now=True)

    class Meta:
        verbose_name = "Товар"
        verbose_name_plural = "Товари"
        ordering = ("name",)

    def __str__(self):
        return self.name
```

Рисунок 3.1 – Реалізація моделі товару в базі даних

Для сутності товару було реалізовано поля для зберігання назви, опису, ціни, а також унікального UUID, який генерується автоматично при створенні запису та слугує основою для формування QR-коду [3]. Особлива увага була приділена моделі логування операцій, де за допомогою зв'язків ForeignKey реалізовано фіксацію користувача, що вніс зміни, та часу проведення операції [3].

3.1.2 Реалізація серіалізаторів та представлень (Views)

Для перетворення складних об'єктів моделей у формат JSON, який є стандартом для REST API, було розроблено серіалізатори на основі класу ModelSerializer [4]. Це забезпечило автоматичну валідацію вхідних даних від фронтенду, гарантуючи, що в базу даних не потраплять некоректні типи даних [4]. Логіка обробки запитів реалізована за допомогою ViewSets, що дозволило згрупувати стандартні операції (читання, створення, оновлення, видалення) в єдині класи [4]. Такий підхід забезпечив чітку структуру маршрутизації (URL routing) та спростив підтримку коду [3; 4].

3.1.3 Система автентифікації та розмежування прав доступу

Безпека системи реалізована через механізм JWT (JSON Web Token) за допомогою бібліотеки SimpleJWT [16]. При вході в систему користувач отримує токен, який додається до заголовків кожного наступного запиту [16]. На стороні сервера впроваджено класи дозволів (Permissions), які обмежують доступ до певних функцій: наприклад, звичайні менеджери мають право лише на сканування та перегляд залишків, тоді як адміністратори - на повне редагування бази даних та перегляд звітів про інвентаризацію [4]. Це гарантує цілісність даних та захист від несанкціонованих дій [16].

3.1.4 Інтеграція інструментів тестування та документації API

Для перевірки працездатності розроблених методів використовувався вбудований інструмент Browsing API від DRF [4], а також тестування запитів через утиліту Postman [12]. Це дозволило переконатися у коректності повернення статус-кодів (наприклад, 200 OK при успіху або 404 Not Found при відсутності товару) та правильності структури JSON-відповідей [4].

3.2 Розробка клієнтського вебінтерфейсу та модуля сканування

Реалізація клієнтської частини інтелектуальної системи базується на використанні бібліотеки React, що дозволило створити динамічний односторінковий додаток (SPA) з високою швидкістю відгуку [14]. Основний акцент при розробці було зроблено на забезпеченні мобільної адаптивності, оскільки ключовим сценарієм використання системи є сканування ідентифікаторів безпосередньо на місці зберігання інвентарю за допомогою смартфонів або планшетів. Архітектура фронтенд-додатка побудована на компонентному підході, де кожен елемент інтерфейсу є незалежним модулем; графічне представлення основних сторінок системи наведено у додатку Г.

Процес розробки розпочався зі створення базової структури маршрутизації за допомогою React Router, що забезпечило навігацію між панеллю керування, списком товарів та сторінкою налаштувань профілю [14]. Для взаємодії з серверною частиною використано бібліотеку Axios, яка виконує асинхронні HTTP-запити до розробленого REST API [4]. У системі реалізовано механізм перехоплення запитів (interceptors) для автоматичного додавання JWT-токена в заголовки кожного звернення до сервера, що гарантує безпеку даних без необхідності повторного введення логіна та пароля при кожній дії [16].

Центральним елементом інтерфейсу став інтелектуальний модуль сканування QR-кодів. Його реалізація базується на інтеграції бібліотеки з відкритим кодом, яка взаємодіє з API камери пристрою через браузер [14]. Алгоритм роботи модуля передбачає постійне зчитування відеопотоку, його перетворення в об’єкт Canvas для аналізу та пошуку специфічних точок позиціонування QR-коду. Після успішного розпізнавання ідентифікатора додаток миттєво ініціює запит до сервера для отримання детальної інформації про об’єкт [4]. Якщо товар знайдено, інтерфейс динамічно відображає картку інвентарю з можливістю редагування кількості або перегляду історії переміщень, забезпечуючи безшовний користувацький досвід без перезавантаження сторінки [14].

Візуальна частина інтерфейсу розроблена з використанням сучасних CSS-фреймворків, що забезпечують чіткість відображення на різних типах дисплеїв. Особлива увага була приділена ергономіці керування «розумним прилавком»: кнопки дій мають збільшений розмір для зручного натискання однією рукою, а

колірна індикація станів дозволяє швидко ідентифікувати критично низькі залишки товарів. Таким чином, розроблений вебінтерфейс не лише візуалізує дані з бази, а й виступає повноцінним інструментом оперативного управління, поєднуючи в собі функції термінала збору даних та аналітичної панелі адміністратора.

3.3 Практична реалізація апаратної частини та елементів ідентифікації

Практичний етап реалізації апаратної складової полягає у втіленні спроектованих цифрових моделей у фізичні об’єкти, що формують інфраструктуру «розумного прилавка». Використання адитивних технологій дозволило створити серію компактних стендів для QR-кодів, які виконують роль надійних ідентифікаторів для кожної одиниці інвентарю.

3.3.1 Параметри 3D-друку та підготовка моделей у програмах-слайсерах

Процес виготовлення апаратних модулів здійснювався на сучасному 3D-принтері Bambulab P1S, який забезпечує високу швидкість та точність друку. Підготовка моделей до виробництва виконувалася у спеціалізованому ПЗ - Bambu Studio (Рис 3.2). Як основний матеріал було обрано полілактид (PLA), що вирізняється низькою усадкою та високою деталізацією дрібних елементів, що є критичним для невеликих стендів.

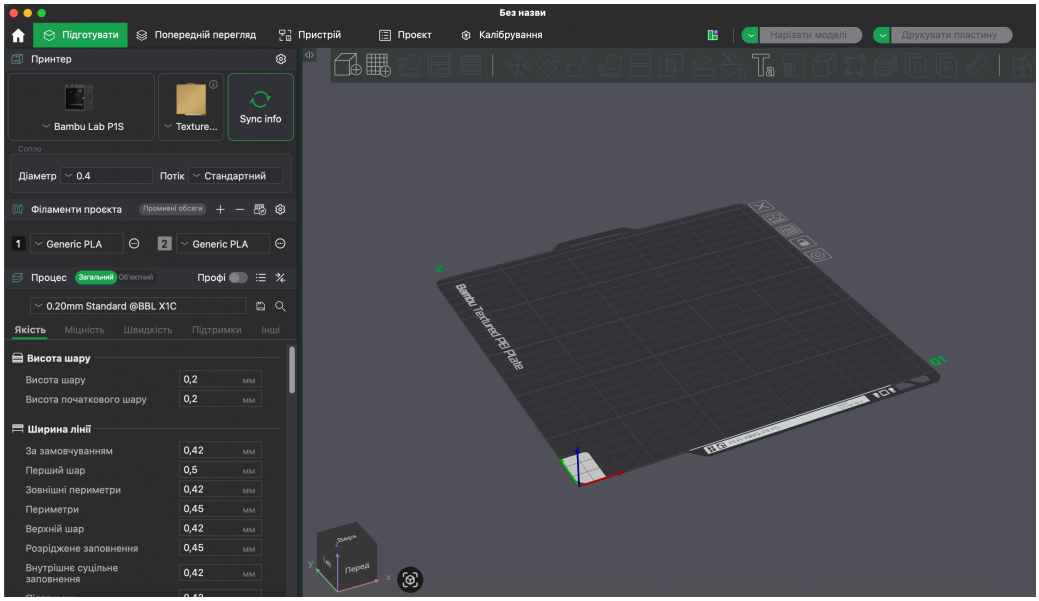

Рисунок 3.2 – Інтерфейс програми Bambu Studio

Для забезпечення механічної міцності та візуальної якості було встановлено наступні параметри друку: висота шару - 0.2 мм (для балансу між швидкістю та якістю), щільність заповнення (infill) - 15% за схемою «Gyroid», що гарантує рівномірну міцність конструкції. Особлива увага була приділена першому шару для забезпечення адгезії з текстурованою PEI-пластиною принтера. Завдяки системі Bambulab AMS (Automatic Material System), була реалізована можливість друку стендів із контрастним маркуванням або використання декількох кольорів для візуального розділення категорій інвентарю безпосередньо в процесі виготовлення.

3.3.2 Виготовлення та маркування фізичних тримачів для інвентарю

Після завершення циклу друку та охолодження робочої поверхні, виготовлені стенди проходили етап фінішної обробки, який включав видалення допоміжних структур (підтримок), якщо вони були передбачені конструкцією. Кожен надрукований модуль був перевірений на відповідність геометрії для забезпечення сталого положення на поверхні прилавка.

Етап маркування полягав в інтеграції згенерованих системою QR-кодів у пази стендів. Завдяки похилій площині надрукованого тримача, ідентифікатор фіксується під оптимальним кутом для зчитування камерою смартфона. У випадках, коли передбачалося використання NFC-ідентифікації, мітка розміщувалася у внутрішній порожнині стенда, що була передбачена на етапі моделювання в Blender. Фінальний вигляд елементів ідентифікації забезпечує естетичність робочої зони та дозволяє персоналу миттєво ідентифікувати товар, просто навівши пристрій на відповідний стенд. Результатом став набір фізичних тримачів, які об’єднують цифровий код та матеріальний об’єкт у єдину інтелектуальну систему обліку.

3.4 Конфігурування середовища розгортання та мережевих сервісів

Процес розгортання розробленої інтелектуальної системи базується на принципах інфраструктури як коду (IaC), що дозволяє автоматизувати запуск усіх сервісів та забезпечити їх стабільну взаємодію. Використання контейнеризації

дозволило уникнути проблем із несумісністю бібліотек та версій програмного забезпечення при перенесенні системи з локальної машини розробника на сервер.

3.4.1 Створення Docker-образів та налаштування сценарію Docker Compose

Для кожного компонента системи (Backend, Frontend) були розроблені індивідуальні інструкції збирання - Dockerfile [5]. Для серверної частини на базі Django було використано легкомовний образ на основі Python Alpine, що дозволило мінімізувати розмір кінцевого контейнера [13]. У файлі конфігурації було описано процеси встановлення необхідних залежностей, збирання статичних файлів та запуск WSGI-сервера Gunicorn для обробки запитів [3] (Рис 3.2). Для клієнтської частини на React було застосовано двоетапне збирання: на першому етапі відбувалася компіляція вихідного коду (build), а на другому — отримані статичні файли передавалися вебсерверу [14].

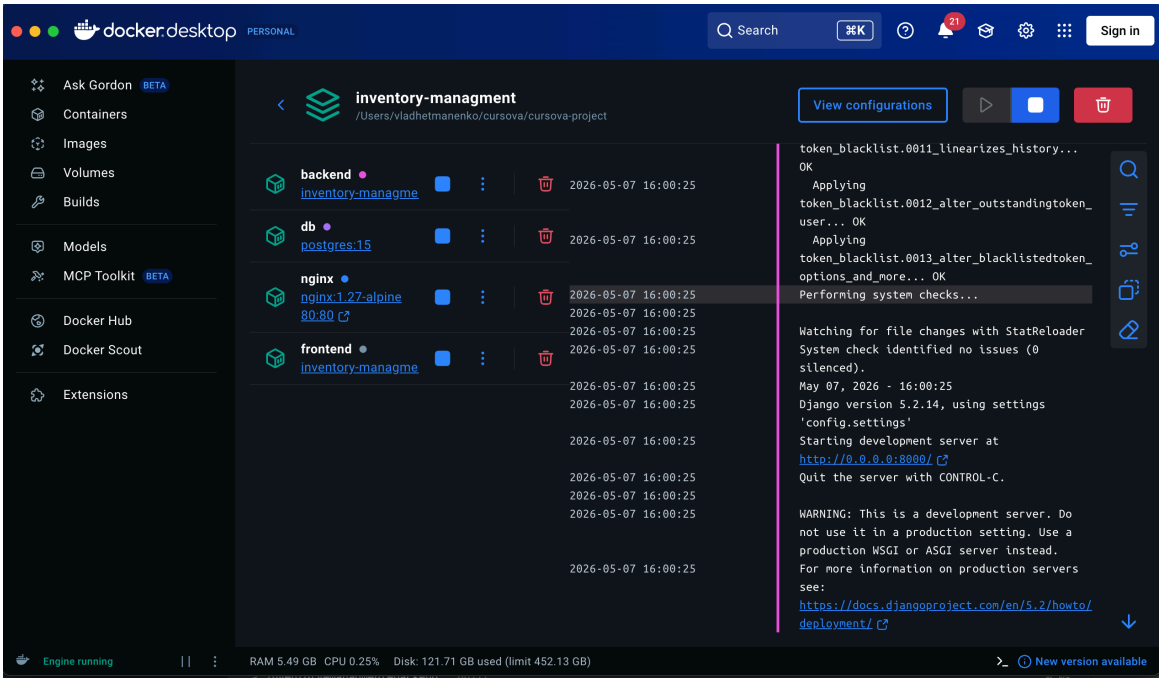

Рисунок 3.3 – Інтерфейс Docker з запущеними контейнерами проекту

Об'єднання всіх сервісів (Backend, Frontend, PostgreSQL) у єдину мережу реалізовано за допомогою сценарію docker-compose.yml [5]. У цьому файлі визначено параметри запуску кожного контейнера, змінні оточення для доступу до бази даних та налаштовано томи (volumes) для постійного зберігання даних [5; 11]. Такий підхід дозволяє запустити всю систему однією командою, автоматично

налаштовуючи внутрішні зв'язки між контейнерами через ізольовану віртуальну мережу Docker Bridge [5].

3.4.2 Налаштування вебсервера Nginx для забезпечення доступу до системи

Для організації зовнішнього доступу та безпечної маршрутизації трафіку було сконфігуровано вебсервер Nginx, який працює у ролі зворотного проксі-сервера [2]. Основне завдання Nginx у даній архітектурі полягає в прийманні запитів на 80-й порт та їх розподілі: запити, що стосуються інтерфейсу користувача, перенаправляються до статичних файлів фронтенду, а запити, що починаються з префікса /api/, транслюються на сервер Django [2; 4].

У конфігураційному файлі nginx.conf було чітко визначено правила обробки заголовків (повна конфігурація сервера наведена у додатку В), що необхідно для коректної роботи автентифікаці та передачі IP-адрес клієнтів [2; 16]. Також було налаштовано ліміти на розмір переданих даних та оптимізовано кешування статичного контенту, що значно прискорило завантаження інтерфейсу на мобільних пристроях [2; 14]. Завдяки такій конфігурації Nginx виступає єдиною точкою входу, забезпечуючи відмовостійкість системи та приховуючи її внутрішню інфраструктуру від зовнішніх загроз [2].

3.5 Тестування працездатності системи та аналіз результатів

Фінальним етапом розробки стало комплексне тестування інтелектуальної системи, метою якого була перевірка стабільності взаємодії програмних та апаратних компонентів, а також оцінка швидкодії обробки даних у реальних умовах. Тестування проводилося за методологією функціонального перевіряння основних сценаріїв використання, що імітують щоденну діяльність персоналу з обліку інвентарю.

Основним сценарієм для перевірки став цикл «сканування-ідентифікація-облік». Для цього на надрукованих стендах були розміщені тестові QR-коди, згенеровані системою. Тестування показало, що модуль розпізнавання на базі React впевнено зчитує коди при різному рівні освітлення приміщення завдяки оптимальному куту нахилу стендів (45°), розрахованому на етапі моделювання. Час

від моменту захоплення коду камерою смартфона до відображення повної картки товару на екрані склав у середньому 0.8–1.2 секунди, що є відмінним показником для веб-орієнтованих систем та забезпечує комфортну роботу без затримок.

Окрему увагу було приділено стрес-тестуванню бази даних PostgreSQL та API. Шляхом імітації одночасних запитів було встановлено, що використання Nginx як проксі-сервера та контейнеризація Docker дозволяють системі стабільно працювати під навантаженням, зберігаючи цілісність транзакцій при зміні кількості товарів. Перевірка механізму логування підтвердила, що кожна дія користувача коректно фіксується в базі даних із прив’язкою до часової мітки та унікального ID менеджера, що забезпечує повний аудит руху майна.

Аналіз результатів впровадження системи на базі «розумного прилавка» дозволив зробити висновок про високу ефективність обраного підходу. Поєднання кастомних 3D-друкованих елементів та гнучкого веб-інтерфейсу дозволило скоротити час на проведення одиничної операції інвентаризації у 3–4 рази порівняно з ручним введенням даних. Система продемонструвала повну сумісність із мобільними пристроями різних виробників, підтвердивши універсальність розробленої архітектури та готовність до експлуатації в реальному середовищі складського або торговельного обліку.

ВИСНОВКИ

За результатами виконання кваліфікаційної роботи можна зробити відповідні висновки. У ході дослідження сучасних методів автоматичної ідентифікації об’єктів було детально проаналізовано технології штрих-кодування, QR-ідентифікації та NFC-зв’язку, що дозволило обґрунтувати вибір QR-кодів як найбільш оптимального та економічно ефективного засобу для інтеграції з вебтехнологіями. На основі проведеного аналізу було сформовано сучасний технологічний стек, який поєднав у собі гнучкість мови Python для серверної частини та продуктивність бібліотеки React для фронтенд-розробки, що стало фундаментом для створення масштабованої системи.

На етапі проектування архітектури інтелектуальної системи було успішно розроблено структуру реляційної бази даних PostgreSQL та описано модель взаємодії компонентів через RESTful API. За допомогою засобів UML-моделювання було візуалізовано логіку роботи системи та сценарії взаємодії користувача з «розумним прилавком», а також створено тривимірні моделі фізичних носіїв ідентифікаторів у середовищах 3ds Max та Blender. Це дозволило ще до етапу реалізації забезпечити узгодженість між програмними алгоритмами обробки даних та геометричними параметрами апаратних засобів.

У рамках практичної реалізації програмного забезпечення було розроблено серверний додаток на базі Django Rest Framework, який забезпечує безпечну обробку транзакцій, автентифікацію через JWT-токени та автоматичне логування дій. Паралельно було створено інтерактивний клієнтський інтерфейс на React, що включає спеціалізований модуль сканування, який дозволяє використовувати камеру мобільного пристрою як повноцінний інструмент збору даних. Це дозволило відмовитися від спеціалізованого дорогого обладнання на користь універсальних вебрішень.

Апаратна складова системи була успішно втілена у життя шляхом виготовлення фізичних тримачів та маркованих комірок для «розумного прилавка» за допомогою 3D-принтера Bambulab P1S. Використання екологічного матеріалу PLA та точне налаштування параметрів друку в Bambu Studio дозволили отримати

зносостійкі та ергономічні елементи інфраструктури, що забезпечують стабільне зчитування QR-кодів. Поєднання друкованих стендів із цифровими ідентифікаторами підтвердило життєздатність концепції апаратно-програмного комплексу в межах спеціальності «Комп’ютерна інженерія».

Запропонована методика розгортання системи у контейнеризованому середовищі Docker із використанням вебсервера Nginx як зворотного проксі забезпечила високу відмовостійкість та переносимість розробки. Тестування системи в реальних умовах показало високу швидкість відгуку та зручність управління товарними залишками через вебінтерфейс. Таким чином, розроблену інтелектуальну систему можна ефективно використовувати для автоматизації складського обліку та оперативного менеджменту інвентарю в організаціях різного профілю, що повністю відповідає меті та завданням роботи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bambu lab wiki homepage. Bambu Lab Wiki. URL: https://wiki.bambulab.com/en/home (дата звернення: 23.04.2026).
2. Deploy using docker compose | NGINX documentation. F5 NGINX Product Documentation. URL: https://docs.nginx.com/nginx-instance-manager/deploy/docker/deploy-nginx-instance-manager-docker-compose/ (дата звернення: 07.04.2026).
3. Django documentation | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/6.0/ (дата звернення: 04.05.2026).
4. Django REST framework. Django REST framework. URL: https://www.django-rest-framework.org/ (дата звернення: 04.05.2026).
5. Docker Inc. Define services in docker compose. Docker Documentation. URL: https://docs.docker.com/reference/compose-file/services/ (дата звернення: 11.04.2026).
6. Docker Inc. Marketplace extensions. Docker Documentation. URL: https://docs.docker.com/extensions/marketplace/ (дата звернення: 04.05.2026).
7. Get Started with ngrok - ngrok documentation. ngrok: AI & API Gateway | Secure Tunnels & Traffic. URL: https://ngrok.com/docs/start (дата звернення: 25.04.2026).
8. GitHub - github/github-mcp-server: GitHub's official MCP Server. GitHub. URL: https://github.com/github/github-mcp-server (дата звернення: 28.04.2026).
9. GitHub - outsmartchad/agentic-engineering: A practical guide to building AI agents – tool-use, multi-agent systems, MCP, eval-driven development, and more. GitHub. URL: https://github.com/outsmartchad/agentic-engineering (дата звернення: 14.04.2026).
10. How do I scan a QR code with React App using your phone?. Stack Overflow. URL: https://stackoverflow.com/questions/75691774/how-do-i-scan-a-qr-code-with-react-app-using-your-phone (дата звернення: 20.04.2026).

11. PostgreSQL: documentation. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/docs/ (дата звернення: 23.04.2026).
12. Postman documentation overview | Postman Docs. Postman Docs | Postman Docs. URL: https://learning.postman.com/docs/introduction/overview (дата звернення: 16.04.2026).
13. Python 3.14 documentation. Python documentation. URL: https://docs.python.org/3/ (дата звернення: 25.04.2026).
14. Quick start – react. React. URL: https://react.dev/learn (дата звернення: 01.04.2026).
15. Robust python. O'Reilly Media, Incorporated, 2021.
16. Simple JWT – simple JWT. Simple JWT – Simple JWT. URL: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/ (дата звернення: 11.04.2026).
17. Stein F. Create custom 3D printable QR codes - qrcode2stl. QRCode2STL. URL: https://qrcode2stl.printer.tools/ (дата звернення: 27.04.2026).
18. Ukrainian W. Уроки від W3Schools українською онлайн. W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів. URL: https://w3schoolsua.github.io/django/index.html#gsc.tab=0 (дата звернення: 15.04.2026).
19. Ultimate materials guide - tips for 3D printing with PLA. Home | Simplify3D Software. URL: https://www.simplify3d.com/resources/materials-guide/pla/ (дата звернення: 29.04.2026).
20. Vue.js. Vue.js - The Progressive JavaScript Framework | Vue.js. URL: https://vuejs.org/guide/introduction (дата звернення: 03.04.2026).