НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації**»**

Спеціальність «126 Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні системи та технології»

Гарант освітньої програми
д.е.н., доцент _____ **Максим МОКРІЄВ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
к.е.н., доцент _____ **Олексій КАФТАННИКОВ**
(підпис)

Виконав _____ **Олександр ОНИЩУК**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Онищуку Олександру Олександровичу

Спеціальність 126 – Інформаційні системи та технології

ОП «Інформаційні системи та технології»

Тема бакалаврської кваліфікаційної роботи «Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації» затверджена наказом від «19» грудня 2025 р. № 3205 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи:
Бізнес-процеси ІТ-відділу ТОВ фірма «Грона», технічна документація стеків технологій PHP та MySQL, офіційна документація платформи контейнеризації Docker.

Перелік питань, що підлягають дослідженню:

1. Змістовний аналіз предметної області управління життєвим циклом картриджів та недоліків ручного обліку.
2. Формування функціональних та нефункціональних вимог до системи.
3. Концептуальне, логічне та фізичне проєктування реляційної бази даних.
4. Програмна реалізація клієнт-серверної веб-системи з використанням технології AJAX.
5. Інфраструктурне розгортання розробленого рішення в ізольованих контейнерах Docker.

Дата видачі завдання «19» грудня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Олексій КАФТАННИКОВ**

Завдання взяв до виконання _____ (підпис) **Олександр ОНИЩУК**

Національний університет біоресурсів і природокористування України
Факультет інформаційних технологій

РЕЦЕНЗІЯ

на бакалаврську кваліфікаційну роботу студента

Онищука Олександра Олександровича

на тему: **Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації**

подану на здобуття ОС «Бакалавр» за спеціальністю

Інформаційні системи і технології

Бакалаврська кваліфікаційна робота присвячена актуальній проблемі автоматизації процесів обліку, моніторингу та управління життєвим циклом витратних матеріалів (зокрема принтерних картриджів) на підприємстві шляхом розробки веб-орієнтованої інформаційної системи на основі технології контейнеризації. Обрана тема відповідає сучасним тенденціям цифровізації та впровадження DevOps-практик у бізнес-процеси та має вагоме практичне значення для зменшення простоїв друкарського обладнання, оптимізації витрат та підвищення прозорості роботи технічного відділу.

У роботі проведено змістовний аналіз предметної області, досліджено існуючі програмні рішення та обґрунтовано доцільність створення власної легкої інформаційної системи. Автором сформульовано функціональні та нефункціональні вимоги до програмного продукту, розроблено технічне завдання, виконано концептуальне та логічне проєктування з використанням UML діаграм, а також спроєктовано структуру реляційної бази даних. Значну увагу приділено опису модулів системи, асинхронній взаємодії між компонентами, питанням тестування та інфраструктурному розгортанню. Позитивною стороною роботи є використання сучасних вебтехнологій, а також успішне застосування платформи Docker для ізольованого розгортання проєкту, що вирішує проблеми несумісності.

До переваг роботи слід віднести логічну структуру викладу матеріалу, достатній рівень опрацювання предметної області, розробку зручного адаптивного інтерфейсу та орієнтацію на реальні потреби конкретного підприємства. Результати роботи є повністю працюючим програмним продуктом і можуть бути використані як основа для подальшого розвитку та безпосереднього впровадження системи в діяльність підприємства.

Разом із тим робота має окремі недоліки. З огляду на орієнтацію системи на закриту локальну мережу, у пошукових модулях замість механізму підготовлених застосовано класичні методи з базовою санітацією, що залишає теоретичні ризики SQL-ін'єкцій. Крім того, у роботі недостатньо повно розкрито питання економічної ефективності від впровадження системи. Також доцільним було б розширити архітектурний опис механізмами регулярного резервного копіювання контейнеризованої бази даних.

Загалом бакалаврська кваліфікаційна робота виконана на належному науково-технічному рівні, відповідає вимогам до робіт освітнього ступеня «Бакалавр», а її автор заслуговує на присвоєння кваліфікації бакалавра за спеціальністю «Інформаційні системи і технології». Рекомендована оцінка – «добре»

“_____” _____20____р.

Рецензент:

_____	_____	_____
(посада, науковий ступінь, вчене звання)	**(підпис)**	**(Ім’я та ПРІЗВИЩЕ)**

Національний університет біоресурсів і природокористування України
Факультет інформаційних технологій

ВІДГУК

на бакалаврську кваліфікаційну роботу студента

Онищука Олександра Олександровича

на тему: **Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації**

подану на здобуття ОС «Бакалавр» за спеціальністю

Інформаційні системи і технології

Бакалаврська кваліфікаційна робота Онищука О. О. присвячена актуальній проблемі — переходу сучасних підприємств від застарілих методів ручного обліку матеріальних ІТ-ресурсів до надійних автоматизованих інформаційних систем. Впровадження розроблених веб-рішень із використанням архітектури контейнеризації є своєчасним кроком, що забезпечує безперебійність бізнес-процесів технічних відділів та ефективно вирішує проблеми інфраструктурної сумісності програмного забезпечення.

Під час виконання кваліфікаційної роботи здобувач проявив високий рівень самостійності, ініціативності та здатність ефективно застосовувати набуті теоретичні знання на практиці. Позитивною рисою роботи є глибокий аналіз предметної області та створення повноцінного клієнт-серверного застосунку з нуля. Здобувач продемонстрував впевнене володіння сучасним стеком веб-технологій, розробив зручний адаптивний інтерфейс та успішно застосував передові DevOps-практики для розгортання проєкту. Завдяки контейнеризації створений продукт є абсолютно кросплатформним: його можна легко та швидко розгорнути на будь-якому комп'ютері чи сервері без складних ручних налаштувань середовища, що робить його готовим до негайного використання будь-де. Робота характеризується логічною структурою та ґрунтовним підходом до тестування.

Розроблена веб-система не є просто теоретичним концептом, а являє собою повністю працюючий та протестований програмний продукт. Отримані результати мають високу практичну цінність і готові до впровадження на ТОВ фірма «Грона». Варто особливо підкреслити універсальність розробленого рішення: хоча система першочергово проєктувалася для обліку принтерних картриджів, її гнучка модульна та реляційна архітектура дозволяє легко масштабувати продукт. Систему можна без значних доопрацювань використовувати для інвентаризації будь-яких інших ІТ-активів, комп'ютерних комплектуючих чи загальноофісних витратних матеріалів підприємства.

Поряд із безперечними перевагами, робота має окремі незначні недоліки. Зокрема, на даному етапі функціонал розробленої вебсистеми більшою мірою орієнтований на потреби технічного персоналу (адміністраторів), тоді як можливості особистого кабінету для звичайних користувачів потребують подальшого розширення. Також у тексті пояснювальної записки зустрічаються окремі стилістичні неточності.

Загалом бакалаврська кваліфікаційна робота виконана у повному обсязі, відповідає вимогам державних стандартів та нормативних документів НУБіП України. Здобувач заслуговує на допуск до публічного захисту перед Екзаменаційною комісією, а робота заслуговує на оцінку «добре» з присвоєнням автору кваліфікації бакалавра за спеціальністю «Інформаційні системи і технології».

“_____” _____ 20____ р.

Керівник бакалаврської кваліфікаційної роботи (проєкту):

_____	_____	_____
(посада, науковий ступінь, вчене звання)	**(підпис)**	**(Ім’я та ПРІЗВИЩЕ)**

АНОТАЦІЯ

Тема бакалаврської кваліфікаційної роботи: “Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації ”.

Автор роботи: Онищук Олександр Олександрович. Керівник роботи: Кафтанников Олексій Юрійович.

Пояснювальна записка: 50 с., 13 рис., 5 табл., 3 дод., 25 джерел.

Графічна частина: 12 презентаційних слайдів.

Ключові слова: веб-система, облік витратних матеріалів, клієнт-серверна архітектура, контейнеризація, docker, база даних.

Метою роботи є розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації, проаналізувати існуючі проблеми управління життєвим циклом картриджів, здійснити проєктування та програмну реалізацію системи, а також визначити оптимальні стратегії її розгортання за допомогою сучасних DevOps-підходів

У цій роботі проаналізовано особливості предметної області та недоліки ручного обліку, розроблено інформаційну систему з допомогою інструментів PHP, MySQL та технології AJAX, реалізовано розгортання вебдодатка в ізольованих контейнерах Docker та обґрунтовано ефективність впровадження такого рішення для безперебійної роботи підприємства

ABSTRACT

Theme of the bachelor's qualification work: "Development of a web system for automated accounting of consumables and spare parts based on containerization technology".

Author: Onyshchuk Oleksandr Oleksandrovych. Supervisor: Kaftannykov Oleksii Yuriiovych.

Explanatory note 50 p., 13 fig., 5 tabl., 3 append., 25 sources.

Graphic part: 12 presentation slides.

Keywords: web system, consumables accounting, client-server architecture, containerization, docker, database.

The purpose of the work is the development of a web system for automated accounting of consumables and spare parts based on containerization technology, analyzing existing problems in the cartridge lifecycle management, carrying out the design and software implementation of the system, and determining optimal strategies for its deployment using modern DevOps approaches. This work analyzes the features of the subject area and the shortcomings of manual accounting, develops an information system using PHP, MySQL and AJAX technology, implements the deployment of a web application in isolated Docker containers, and substantiates the effectiveness of introducing such a solution for the uninterrupted operation of the enterprise.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ІС – інформаційна система

ІСТ – інформаційні системи та технології

СУБД – система управління базами даних

AJAX – Asynchronous JavaScript and XML

UI/UX – User Interface / User Experience

UML — Unified Modeling Language

BPMN — Business Process Model and Notation

DFD — Data Flow Diagram

ER — Entity Relationship

ITAM — IT Asset Management

SQL — Structured Query Language

DDL — Data Definition Language

API — Application Programming Interface

QR — Quick Response

WSL — Windows Subsystem for Linux

HTTP — HyperText Transfer Protocol

ВСТУП

У бакалаврській кваліфікаційній роботі досліджено проблему автоматизації управління життєвим циклом витратних матеріалів на підприємстві. Тема є актуальною, адже своєчасний облік, контроль та заправка картриджів дозволяє уникнути простоїв обладнання через нестачу витратних матеріалів, зменшити операційні витрати та підвищити загальну прозорість процесів обслуговування друкарської техніки.

Інформаційні системи та технології (ІСТ) забезпечують ефективне управління бізнес-процесами, зберігання та обробку даних, а також підтримку прийняття рішень. Згідно зі стандартами вищої освіти України за спеціальністю 126 «Інформаційні системи та технології», ІСТ охоплюють розробку, впровадження та підтримку інформаційних технологій у різних сферах, включаючи корпоративне управління ресурсами підприємства.

Для створення системи було використано стек вебтехнологій (PHP, MySQL, Vanilla JavaScript, AJAX) та технологію контейнеризації Docker. Розробка на PHP та MySQL дозволяє створити надійну клієнт-серверну архітектуру та забезпечити цілісність реляційних даних. Docker, у свою чергу, — це сучасний інструмент, що пакує код програми разом із усіма її залежностями (вебсервер, бази даних) в ізольовані контейнери. Його переваги включають можливість розгортання проєкту на будь-якій операційній системі однією командою, гарантію відсутності проблем із сумісністю та спрощення процесів тестування. У контексті управління підприємством використання Docker дозволяє легко масштабувати систему та підтримувати її безперебійну роботу.[1] [17] [23]

Для забезпечення ефективного обліку ресурсів необхідно впроваджувати сучасні інформаційні рішення, переходити від ручного ведення записів до централізованих баз даних та адаптувати робочі процеси до нових цифрових реалій. Використання розробленої системи дозволяє адміністраторам ефективно моніторити статус обладнання, вести історію обслуговування та приймати обґрунтовані рішення щодо закупівлі нових запчастин.

У перспективі впровадження такої системи (на прикладі ТОВ фірма «Грона») забезпечить активну оптимізацію роботи технічного відділу. Це

включає подальше розширення функціоналу, зокрема надання користувачам можливості самостійно подавати заявки на обслуговування через особистий кабінет, що значно спростить корпоративну взаємодію. Для моделювання бізнес-процесів було використано реальні дані та вимоги IT-відділу підприємства, що робить розроблену систему максимально наближеною до практичного застосування.

Виходячи з актуальності проблеми, сформовано мету та завдання дослідження. Мета дослідження — розробити веб-систему автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації для підвищення ефективності роботи підприємства.

Завдання дослідження: – проаналізувати предметну область та існуючі рішення управління картриджами; – спроєктувати базу даних та інтерфейс користувача системи; – виконати програмну реалізацію системи з використанням PHP, MySQL та технології AJAX; – розгорнути вебдодаток у контейнерах Docker для забезпечення його кросплатформності; – провести тестування розробленого рішення.

Об’єкт дослідження — процес управління та обліку витратних матеріалів на підприємстві. Предмет дослідження — інструментальні засоби створення інформаційної системи обліку витратних матеріалів та технологія її контейнеризації.

Коротка характеристика розділів роботи. У першому розділі проведено змістовний аналіз предметної області, досліджено проблеми ручного обліку витратних матеріалів та сформовано вимоги до розроблюваної системи. У другому розділі обґрунтовано вибір архітектури системи, наведено результати концептуального та логічного моделювання бізнес-процесів і спроєктовано структуру реляційної бази даних. У третьому розділі описано процес програмної реалізації клієнтської та серверної частин, розкрито специфіку інфраструктурного розгортання проєкту за допомогою платформи Docker та наведено результати функціонального тестування системи.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Безперебійна робота інформаційно-технічної інфраструктури є критично важливою умовою для ефективного функціонування та виконання бізнес-процесів будь-якого сучасного підприємства. Вагомою складовою цієї інфраструктури є друкарська техніка та периферійні пристрої, які потребують регулярного моніторингу та технічного обслуговування. Відповідно, предметною областю даного дослідження є процеси обліку, моніторингу та управління життєвим циклом витратних матеріалів, зокрема принтерних картриджів, на підприємстві (на прикладі ТОВ фірма «Грона»).

Структурні особливості предметної області та бізнес-процеси Управління витратними матеріалами для друку являє собою складний багатокроковий процес. Змістовний аналіз предметної області показав, що життєвий цикл картриджа не обмежується лише його встановленням у принтер. Цей цикл охоплює низку послідовних етапів: придбання нового пристрою, його реєстрація, експлуатація у відповідному відділі, вичерпання ресурсу, передача до технічного відділу, діагностика стану, відправка на заправку або ремонт до підрядника, повернення із заправки та подальша експлуатація або остаточне списання у разі повної технічної несправності.

Аналіз бізнес-процесів дозволив виокремити дві основні групи учасників (ролей), між якими відбувається постійна взаємодія у межах предметної області:

- Адміністратори (технічний відділ / IT-спеціалісти) — фахівці, які несуть відповідальність за працездатність обладнання, ведуть облік витратних матеріалів, контролюють процес заправки, переміщують картриджі між статусами, генерують звітність та формують акти дефектації.
- Користувачі (співробітники структурних підрозділів) — кінцеві споживачі, які безпосередньо експлуатують техніку. Їхня роль полягає у формуванні заявок на заміну або обслуговування пустих чи несправних

картриджів, а також у відстеженні статусу виконання своїх запитів.

Проблематика та функціональні недоліки поточного стану Дослідження традиційних підходів до управління витратними матеріалами на підприємствах виявило, що використання ручного обліку (зокрема паперових журналів або локальних електронних таблиць) призводить до низки суттєвих організаційних та фінансових проблем. До основних недоліків існуючого неавтоматизованого процесу належать:

- Втрата прозорості та безвідповідальність: Відсутність централізованої прив'язки конкретного картриджа (за унікальним штрих-кодом або ID) до конкретного принтера та відповідальної особи ускладнює пошук винних у разі механічного пошкодження обладнання.
- Простої обладнання: Відсутність оперативного моніторингу статусу картриджів (наприклад, несвоєчасне сповіщення про те, що резервні копії знаходяться на заправці) призводить до зупинки документообігу у відділах, що негативно впливає на загальну продуктивність підприємства.
- Складність аналітики та аудиту: Відсутність єдиної цифрової історії життєвого циклу картриджа не дозволяє об'єктивно оцінити якість роботи підрядників із заправки, розрахувати реальні витрати на друк та прийняти обґрунтоване рішення щодо списання фізично зношеного обладнання.

Інфраструктурні виклики та шляхи їх вирішення Окрім організаційних проблем, виявлено також технологічні труднощі, пов'язані з розгортанням та підтримкою існуючих програмних рішень. Традиційні локальні інформаційні системи часто є складними у розгортанні, вимагають ручного налаштування локального середовища (наприклад, встановлення комплексів типу XAMPP), жорстко залежать від специфічних конфігурацій операційних систем та версій програмного забезпечення на сервері.

Це створює проблему сумісності та додаткове навантаження на системних адміністраторів підприємства при перенесенні системи на нове обладнання.

Виходячи з вищенаведеного, встановлено об’єктивну потребу у розробці

спеціалізованої веб-орієнтованої інформаційної системи. Ця система повинна автоматизувати процеси реєстрації картриджів, швидкої зміни їх статусів (за допомогою пошуку та сканування штрих-кодів), збереження повної історії переміщень та розмежування прав доступу між адміністраторами та користувачами.

Для вирішення проблеми сумісності та спрощення розгортання системи доцільно застосувати сучасні DevOps-підходи, зокрема технологію контейнеризації. Пакування програмного коду вебдодатка (серверної частини та клієнтського інтерфейсу) разом із системою управління базами даних в ізольовані віртуальні контейнери дозволить гарантувати безперебійну роботу інформаційної системи на будь-якому обладнанні. [2]

Використання контейнерної архітектури усуває необхідність складного ручного налаштування локальних серверів, стандартизує середовище розробки та дозволяє розгортати повноцінно функціонуючу систему однією командою.

Отже, предметна область потребує впровадження надійного клієнт-серверного рішення, яке ефективно поєднає зручний веб-інтерфейс для обліку матеріальних ресурсів підприємства із сучасною контейнерною архітектурою для забезпечення найвищого рівня стабільності, безпеки та кросплатформності.

1.2 Аналіз наявних інформаційних технологій предметної області

Впровадження автоматизованих систем для управління IT-активами (ITAM – IT Asset Management) та витратними матеріалами є стандартом для сучасних підприємств. Для вирішення завдань обліку, моніторингу та управління життєвим циклом картриджів на ринку існує низка готових інформаційних технологій та програмних продуктів. Головною метою даного підрозділу є аналіз існуючих рішень, вивчення їхніх переваг та недоліків для обґрунтування доцільності розробки власної інформаційної системи.

Умовно наявні на ринку інструментальні засоби, які використовуються для вирішення подібних завдань, можна поділити на три категорії: базові офісні програми (електронні таблиці), універсальні системи управління IT-інфраструктурою (HelpDesk / ServiceDesk) та спеціалізовані системи інвентаризації.

Електронні таблиці (Microsoft Excel / Google Sheets) Історично найпершим і найпоширенішим інструментом для ведення обліку матеріальних цінностей на невеликих підприємствах є використання електронних таблиць. Призначення: Ручне введення та зберігання даних у вигляді двовимірних масивів. Переваги: Мінімальний поріг входження, відсутність фінансових витрат на впровадження, гнучкість у створенні нових стовпців або формул. Недоліки: Електронні таблиці не є повноцінними реляційними базами даних. Вони не забезпечують цілісність даних, унеможливлюють одночасну повноцінну роботу кількох користувачів із розмежуванням прав доступу (наприклад, між звичайним користувачем та адміністратором). Крім того, в таких системах неможливо автоматизувати процес зміни статусів за допомогою сканування штрих-кодів та вкрай складно вести прозору історію переміщень конкретного картриджа.

Універсальні системи управління IT-інфраструктурою (на прикладі GLPI) GLPI (Gestionnaire Libre de Parc Informatique) — це потужна інформаційна система з відкритим вихідним кодом, призначена для управління IT-парком підприємства та надання послуг технічної підтримки. Призначення: Комплексна інвентаризація комп'ютерного обладнання, програмного

забезпечення, витратних матеріалів, а також управління заявками користувачів (HelpDesk). Переваги: Багатофункціональність, наявність вбудованих інструментів для обліку картриджів та прив'язки їх до конкретних моделей принтерів, масштабованість. Недоліки: Система є занадто перевантаженою надлишковим функціоналом для вузької задачі обліку лише картриджів. Інтерфейс системи є складним для непідготовлених користувачів. Окрім того, розгортання та адміністрування GLPI вимагає значних витрат часу системного адміністратора, а оновлення системи часто супроводжується проблемами сумісності локального середовища.

Спеціалізовані вебсистеми інвентаризації (на прикладі Snipe-IT) Snipe-IT — це сучасна система інвентаризації IT-активів, розроблена спеціально для відстеження обладнання, ліцензій та витратних матеріалів. Призначення: Відстеження того, яке обладнання закріплено за яким співробітником, моніторинг залишків витратних матеріалів. Переваги: Зручний сучасний людино-машинний інтерфейс, вбудована генерація та підтримка зчитування штрих-кодів і QR-кодів, надійна архітектура. Недоліки: Високі системні вимоги до серверного обладнання та складність налаштування бізнес-процесів, специфічних для вітчизняних підприємств (зокрема, багатоетапний процес відправки картриджів на заправку до зовнішніх підрядників та повернення їх в експлуатацію).

Для наочності порівняльний аналіз переваг та недоліків розглянутих інформаційних технологій зведено до єдиної порівняльної таблиці

Таблиця 1.1 Порівняльна характеристика технологій

Назва аналога	Переваги	Недоліки
Електронні таблиці (MS Excel, Google Sheets)	Відсутність витрат на розгортання; максимальна простота налаштувань.	Відсутність автоматизації; неможливість інтеграції сканерів штрих-кодів; відсутність рольової моделі доступу; складність відстеження історії змін.
GLPI	Потужний інструментарій; відкритий вихідний код; наявність модуля заявок.	Перевантаженість інтерфейсу зайвими функціями; складність розгортання та локального адміністрування; високий поріг входження.
Snipe-IT	Сучасний інтерфейс; підтримка штрих-кодів; зручний механізм видачі активів користувачам.	Складність адаптації під специфічний життєвий цикл картриджа (заправка, діагностика, списання); складність інфраструктурної підтримки.

Проведено аналіз існуючих інформаційних технологій та інструментальних засобів предметної області. В результаті аналізу визначено, що електронні таблиці є морально застарілим підходом, який не задовольняє потреб сучасного підприємства в автоматизації. Водночас готові професійні рішення (GLPI, Snipe-IT), хоча й мають потужний функціонал, є занадто складними, містять багато надлишкових модулів та вимагають значних зусиль для розгортання і підтримки серверного середовища.

З огляду на це, логічно обґрунтованою є розробка власної легкої веб-орієнтованої інформаційної системи, архітектура якої буде максимально адаптована під конкретний бізнес-процес обліку картриджів на ТОВ фірма «Грона».

Для усунення недоліків, пов'язаних зі складністю розгортання на сервері, розроблювану систему доцільно реалізувати з використанням технології контейнеризації Docker.

Це дозволить об'єднати простоту цільового інтерфейсу з надійністю та швидкістю розгортання сучасних DevOps-рішень.

1.3 Визначення вимог до інструментальних засобів створення і використання інформаційних систем та технологій, розробка технічного завдання

На основі проведеного змістовного аналізу предметної області та розгляду наявних інформаційних технологій виникає необхідність чіткого формулювання вимог до розроблюваної системи. У даному підрозділі окреслено функціональні та нефункціональні вимоги до веб-системи, обґрунтовано вибір інструментальних засобів для її реалізації, а також розроблено технічне завдання для подальшого проєктування та розгортання інформаційної технології на основі контейнеризації.

Вимоги до функціональних характеристик системи Інформаційна система обліку витратних матеріалів повинна автоматизувати рутинні процеси технічного відділу підприємства та забезпечувати виконання наступних функціональних вимог:

1. Автентифікація та авторизація: забезпечення безпечного входу до системи з використанням логіна та пароля, а також розмежування прав доступу залежно від ролі користувача (Адміністратор або Користувач).

2. Управління картриджами: можливість додавання нових пристроїв до бази даних із зазначенням унікального штрих-коду, моделі, марки, принтера та закріпленого користувача чи відділу.

3. Управління статусами: надання інструментарію для зміни поточного стану картриджа (наприклад: «Відправлено на заправку», «Повернуто з заправки», «Списано»).

4. Ведення історії обслуговування: автоматична фіксація всіх дій та змін статусу у базі даних із збереженням інформації про дату, час та ініціатора змін для подальшого аудиту.

5. Пошук та фільтрація: швидкий пошук картриджів за допомогою сканування або ручного введення штрих-коду, а також фільтрація записів за датою та поточним статусом.

6. Документообіг: можливість автоматичного формування та виведення на друк актів дефектації для списання обладнання.

Нефункціональні вимоги до системи Для забезпечення стабільної та зручної роботи програмного продукту висуваються такі нефункціональні вимоги:

1. Кросплатформність та архітектура: система повинна мати веб-орієнтовану клієнт-серверну архітектуру та коректно функціонувати у сучасних веббраузерах без необхідності встановлення додаткового клієнтського програмного забезпечення.

2. Адаптивність інтерфейсу: людино-машинний інтерфейс має бути адаптивним (Responsive Design) для коректного відображення як на моніторах персональних комп'ютерів, так і на екранах мобільних пристроїв (смартфонів, планшетів, терміналів збору даних).

3. Динамічність: взаємодія з базою даних при пошуку та оновленні інформації має відбуватися асинхронно, без повного перезавантаження вебсторінки.

4. Безпека даних: система повинна мати захист від шкідливих SQL-ін'єкцій при взаємодії з базою даних, а також використовувати механізм сесій для захисту внутрішніх сторінок від несанкціонованого доступу.

5. Незалежність від локального середовища: розгортання вебдодатка та бази даних повинно здійснюватися за допомогою сучасних DevOps-підходів в ізольованих контейнерах, що усуне проблему конфліктів версій програмного забезпечення на різних серверах.

Вимоги до вхідних даних Для коректного функціонування бази даних інформаційна система повинна обробляти та зберігати такі типи вхідних даних:

1. штрих-коди або інвентарні номери (до 20 символів);

2. назви моделей картриджів та їх марок;

3. ідентифікаційні дані співробітників або відділів (ПІБ, назва

підрозділу);

4. моделі принтерів, за якими закріплюються витратні матеріали.

Вибір інструментальних засобів реалізації На основі сформульованих вимог та необхідності забезпечення високої швидкодії обрано наступний технологічний стек:

1. Серверна частина (Backend): мова програмування PHP (версії 8.x). Вибір обґрунтований високою швидкістю обробки скриптів та широкими можливостями інтеграції з реляційними базами даних. [23]

2. База даних: СКБД MySQL/MariaDB. Забезпечує надійне зберігання даних та підтримує механізми зв'язків через зовнішні ключі (Foreign Keys) для забезпечення цілісності інформації. Взаємодія з БД здійснюється через розширення mysqli з використанням підготовлених запитів (Prepared Statements). [22]

3. Клієнтська частина (Frontend): мова розмітки HTML5, каскадні таблиці стилів CSS3 (з медіа-запитами для адаптивності) та Vanilla JavaScript. Використання чистого JavaScript без важких фреймворків дозволяє оптимізувати швидкість завантаження сторінок. Для асинхронного обміну даними обрано технологію AJAX.

4. Засоби контейнеризації: платформа Docker, WSL та інструмент Docker Compose. Дозволяють спакувати вебсервер (Apache + PHP) та базу даних (MariaDB) у два ізольовані взаємопов'язані контейнери, що гарантує ідентичність роботи системи на етапах розробки та промислової експлуатації. [1][23]

Технічне завдання

1. Назва проєкту: Веб-система автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації.

2. Мета розробки: створення централізованого інформаційного інструменту для автоматизації процесів обліку, контролю статусу та

відстеження життєвого циклу принтерних картриджів на підприємстві (на прикладі ТОВ фірма «Грона»).

3. Основні завдання системи: автоматизація реєстрації картриджів; моніторинг переміщень пристроїв між відділами та підрядниками із заправки; ведення безперервної історії обслуговування; надання зручного інструментарію для пошуку за штрих-кодом.

4. Вимоги до інфраструктури розгортання: система повинна розгортатися за допомогою інструкцій docker-compose.yml та Dockerfile, які автоматично ініціалізують віртуальну мережу, підіймають вебсервер та імпортують структуру бази даних із заздалегідь підготовленого SQL-дампа. [1][17]

5. Очікуваний результат: готова до впровадження інформаційна технологія, що дозволить мінімізувати простої друкарського обладнання, оптимізувати витрати на обслуговування та забезпечити прозорість роботи технічного персоналу.

Висновки до розділу 1.

В результаті дослідження предметної області підтверджено необхідність автоматизації обліку картриджів на підприємстві. Аналіз аналогів довів недоцільність використання надлишкових систем інвентаризації. Сформоване технічне завдання визначає розробку легкої веб-системи на базі PHP та MySQL з обов'язковою подальшою контейнеризацією у Docker.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ СТВОРЕННЯ І ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

2.1 Архітектура системи та вибір технологій реалізації

На етапі проєктування інформаційної системи обліку витратних матеріалів було визначено оптимальну архітектуру та здійснено декомпозицію системи на логічні модулі. Згідно з технічним завданням, розроблена інформаційна технологія базується на класичній клієнт-серверній архітектурі, що забезпечує централізоване зберігання даних та багатоклієнтський доступ. [2]

Проєктом передбачено використання наступного технологічного стека:

1. Серверна частина (Backend): реалізована мовою PHP (сумісність з версіями 8.x). Відповідає за обробку запитів, бізнес-логіку, автентифікацію користувачів та генерацію динамічного контенту. [4]

2. База даних: реляційна СКБД MySQL/MariaDB для надійного збереження інформації про користувачів, картриджі та історію їх обслуговування. Взаємодія з базою даних здійснюється через розширення mysqli з використанням підготовлених запитів (Prepared Statements) для забезпечення безпеки. [22]

3. Клієнтська частина (Frontend): побудована з використанням HTML5, CSS3 та Vanilla JavaScript (без використання сторонніх важких фреймворків). Для забезпечення динамічного оновлення інформації на сторінках (зокрема, живого пошуку) широко застосовується технологія асинхронних запитів AJAX. [11]

Функціональна декомпозиція системи передбачає наявність наступних взаємопов'язаних модулів:

1. Модуль автентифікації (index.php) — контроль доступу та ініціалізація сесій.

2. Модуль управління робочим простором (home.php) — головний

дашборд із функцією сканування та живого пошуку.

3. Модулі управління ресурсами (add_catrige.php, catrige.php) — обробка логіки додавання нових пристроїв та їх категоризація за статусами.

4. Модулі обробки статусів (SearchCatrige.php, SendCatrige.php, WriteOff.php) — бекенд-обробники, що безпосередньо взаємодіють з базою даних та фіксують зміни.

5. Модуль генерації звітності (print.php, history.php) — формування актів дефектації та відображення життєвого циклу картриджів.

Вирішення проблеми інфраструктурної сумісності та швидкого розгортання проєкту реалізовано шляхом застосування технології контейнеризації Docker. Архітектура розгортання розділена на два ізольовані взаємодіючі контейнери: контейнер вебсервера (на базі образу php:8.2-apache) та контейнер бази даних (на базі mariadb). Це дозволяє повністю абстрагуватися від конфігурацій локальних серверів підприємства. [17]

2.2 Концептуальне та логічне моделювання

Для візуалізації вимог до системи, визначення меж її функціонування та моделювання бізнес-процесів було застосовано інструментарій візуального моделювання (UML, BPMN, DFD).

Логіка взаємодії користувачів із системою формалізована за допомогою діаграми прецедентів (Use Case diagram). Модель визначає дві ключові ролі: «Адміністратор» (з повним набором прав для управління даними, статусами та історією) та «Користувач» (з правами перегляду стану власних заявок та картриджів).

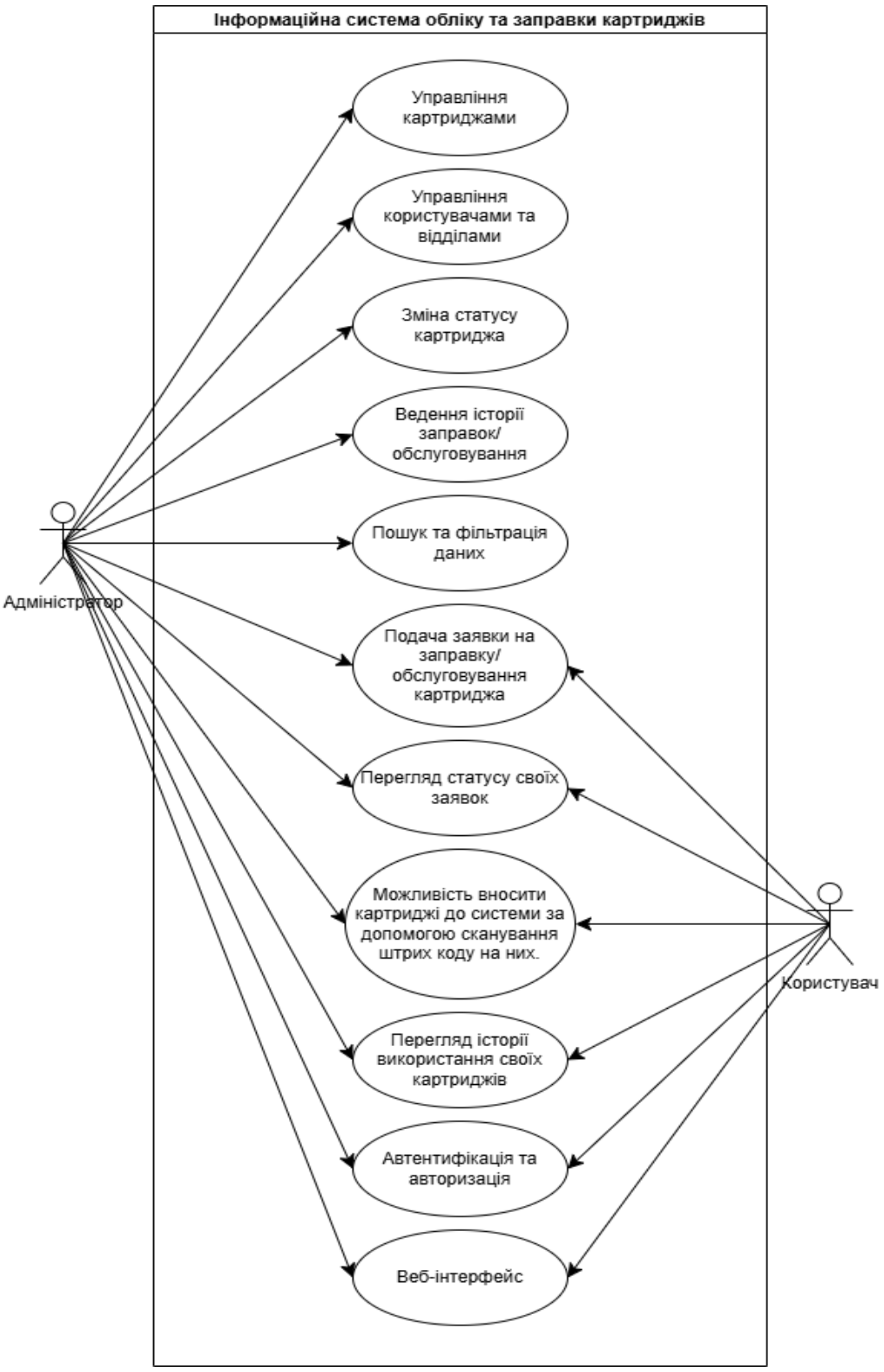


Рисунок 2.1 – Діаграма прецедентів інформаційної системи

Як видно з рисунка 2.1, система передбачає чітке розмежування прав доступу між адміністратором та звичайним користувачем. Адміністратор наділений повним спектром повноважень для управління даними, тоді як користувач має обмежений доступ, орієнтований переважно на подачу заявок та перегляд статусу власних картриджів.

Для моделювання життєвого циклу витратних матеріалів застосовано нотацію BPMN (Business Process Model and Notation). Розроблена модель ілюструє повний алгоритм обробки картриджа: від моменту його виходу з ладу у структурному підрозділі, проходження діагностики в технічному відділі, до прийняття рішення про ремонт, заправку або остаточне списання зі складанням відповідного акту.

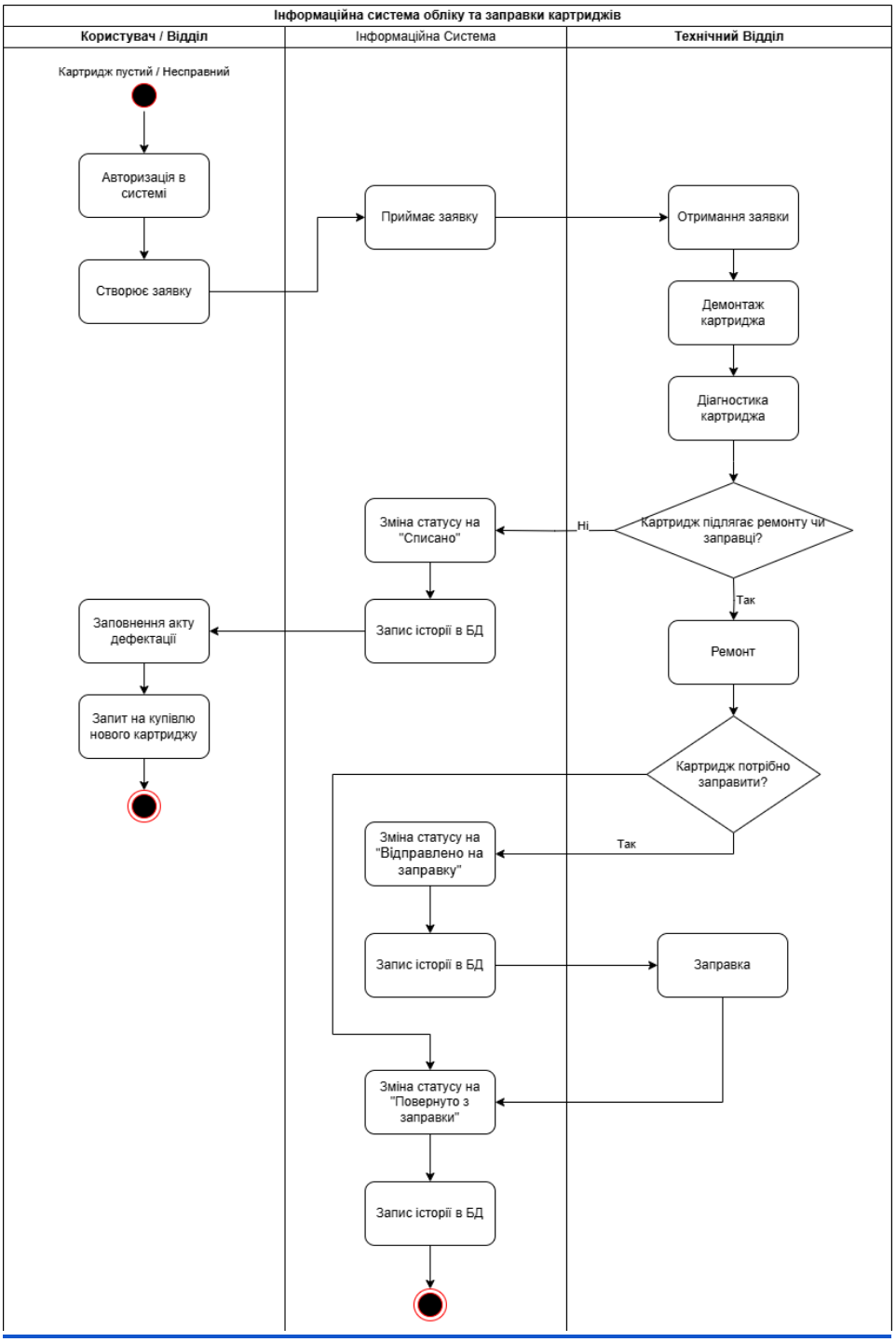


Рисунок 2.2 – Модель бізнес-процесу обслуговування картриджів (BPMN)

На рисунку 2.2 детально візуалізовано алгоритм переміщення картриджа між відділами з моменту виявлення несправності до фінального рішення. Таке моделювання дозволяє наочно виявити можливі затримки в процесі обслуговування та визначити відповідальних осіб на кожному з етапів прийняття рішень.

З метою відстеження руху інформації всередині системи спроєктовано діаграму потоків даних (Data Flow Diagram). Вона демонструє, як дані від користувачів та адміністраторів трансформуються в процесі роботи модулів (обробка запиту, зміна статусу, адміністрування) та зберігаються у центральному сховищі даних. [16]

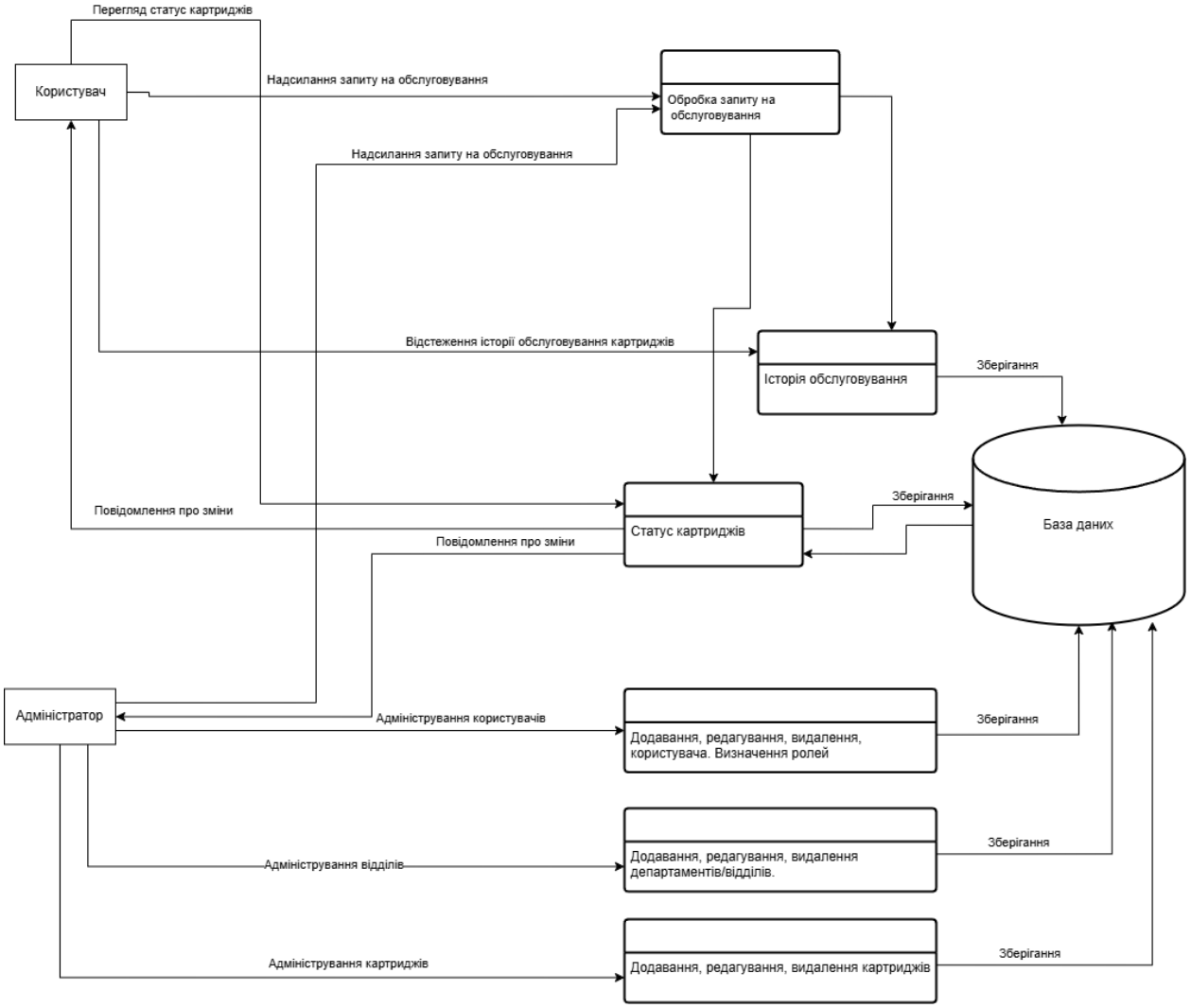


Рисунок 2.3 – Діаграма потоків даних (DFD) розробленої системи

Зображена на рисунку 2.3 діаграма ілюструє процеси обміну інформацією між зовнішніми сутностями та центральним сховищем бази

даних. Вона підтверджує, що всі операції користувачів, пов'язані зі зміною статусів чи редагуванням профілів, централізовано фіксуються та зберігаються для забезпечення повної цілісності історії переміщень.

2.3 Проєктування бази даних

Враховуючи необхідність забезпечення цілісності даних, відсутності дублювання та підтримки складних зв'язків між сутностями, було обрано реляційну модель бази даних.

У процесі концептуального проєктування (ER-моделювання) виділено три основні сутності: Users (Користувачі), Catriges (Картриджі) та History_Catriges (Історія картриджів).

Логічна ER-діаграма системи демонструє ключові атрибути та кардинальність зв'язків між сутностями.

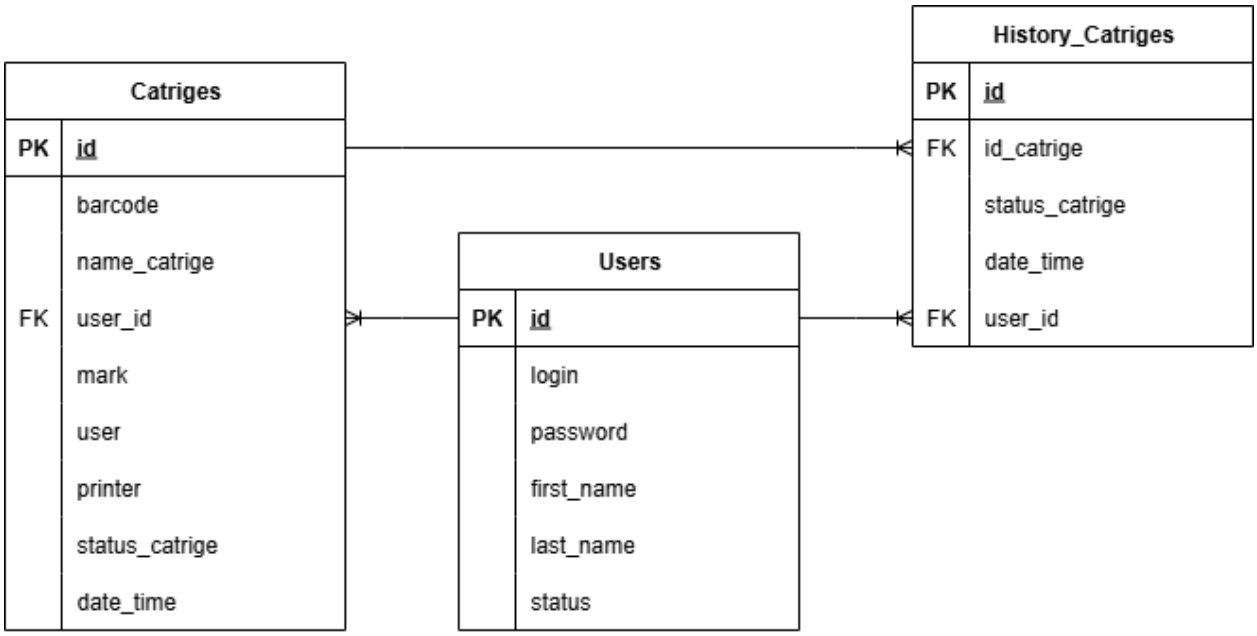


Рисунок 2.4 – Логічна ER-модель бази даних

Як свідчить логічна ER-модель на рисунку 2.4, архітектура бази даних оптимізована для уникнення дублювання інформації. Зв'язки типу «один-до-багатьох» гарантують, що кожен запис про зміну статусу або закріплення обладнання однозначно асоціюється з конкретним картриджем та відповідальним працівником.

Зв'язки між таблицями реалізовано за типом «один до багатьох» (1:∞).

Один користувач (Users) може бути відповідальним за декілька картриджів (Catriges), а один картридж може мати безліч записів у таблиці історії (History_Catriges).

На етапі фізичного проєктування бази даних розроблено точну структуру таблиць із зазначенням типів даних, первинних (Primary Key) та зовнішніх ключів (Foreign Key). Зокрема, для ідентифікаторів використано тип INT з атрибутом AUTO_INCREMENT, для текстових даних — VARCHAR, а для фіксації часу операцій — тип TIMESTAMP з автоматичним оновленням.

Для забезпечення посилальної цілісності між таблицями налаштовано зовнішні ключі (наприклад, поле user_id у таблиці catriges посилається на id у таблиці users). База даних відповідає вимогам третьої нормальної форми (3НФ), що виключає транзитивні залежності.

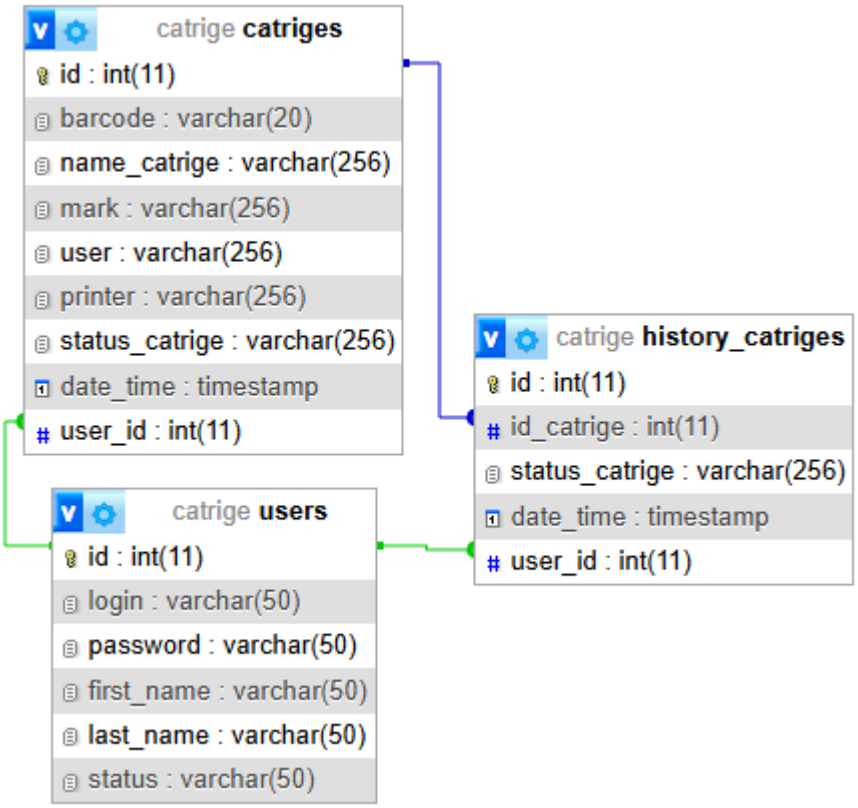


Рисунок 2.5 – Фізична модель бази даних системи

Представлена на рисунку 2.5 фізична модель деталізує технічну реалізацію структури з визначенням конкретних типів даних та обмежень. Використання зовнішніх ключів із каскадними діями забезпечує сувору

посилальну цілісність інформації безпосередньо на рівні системи управління базами даних.

Спроєктована база даних складається з трьох взаємопов'язаних таблиць:

users — призначена для зберігання облікових даних користувачів та адміністраторів системи, що забезпечує роботу модуля авторизації та розмежування прав доступу.

catriges — є основною сутністю системи й містить детальні відомості про кожен одиничний картридж (марка, штрих-код, поточний статус, прикріплений принтер та відповідальний співробітник).

history_catriges — призначена для автоматичного журналювання (логування) будь-яких змін статусів картриджів, що дозволяє відстежувати повний життєвий цикл витратних матеріалів та будувати ретроспективні звіти.

Нижче у таблицях 2.1–2.3 наведено детальний опис фізичної структури кожної з таблиць бази даних.

Таблиця 2.1 – Структура таблиці catriges

Поле	Тип даних	Ключ	Опис
id	INT(11)	PK	Унікальний ідентифікатор картриджа (первинний ключ з автоінкрементом)
barcode	VARCHAR(20)	—	Унікальний штрих-код або інвентарний номер картриджа
name_catrige	VARCHAR(256)	—	Назва виробника або модель картриджа
mark	VARCHAR(256)	—	Маркування (сумісний артикул/модель) картриджа
user	VARCHAR(256)	—	Співробітник або структурний підрозділ, за яким закріплено картридж
printer	VARCHAR(256)	—	Модель принтера, для якого призначений або в якому встановлений картридж
status_catrige	VARCHAR(256)	—	Поточний стан (наприклад: «Новий» , «На заправку» , «Заправка» , «Повернуто з заправки» , «Списано»)
date_time	TIMESTAMP	—	Дата та час останньої модифікації запису (автоматично оновлюється при кожній зміні статусу)

user_id	INT(11)	FK	Ідентифікатор користувача системи (зовнішній ключ, зв'язок з users.id), який вніс зміни

Таблиця 2.2 – Структура таблиці history_catriges

Поле	Тип даних	Ключ	Опис
id	INT(11)	PK	Унікальний ідентифікатор запису історії (первинний ключ з автоінкрементом)
id_catrige	INT(11)	FK	Ідентифікатор картриджа (зовнішній ключ, зв'язок з catriges.id), статус якого було змінено
status_catrige	VARCHAR(256)	—	Зафіксований статус картриджа на момент події
date_time	TIMESTAMP	—	Дата та час фіксації події в системі (за замовчуванням CURRENT_TIMESTAMP)
user_id	INT(11)	FK	Ідентифікатор користувача системи (зовнішній ключ, зв'язок з users.id), який виконав операцію

Таблиця 2.3 – Структура таблиці users

Поле	Тип даних	Ключ	Опис
id	INT(11)	PK	Унікальний ідентифікатор користувача системи (первинний ключ з автоінкрементом)
login	VARCHAR(50)	—	Унікальне ім'я (логін) користувача для авторизації в системі
password	VARCHAR(50)	—	Пароль користувача для входу в систему
first_name	VARCHAR(50)	—	Ім'я користувача (співробітника-оператора)
last_name	VARCHAR(50)	—	Прізвище користувача
status	VARCHAR(50)	—	Роль користувача в системі для обмеження прав доступу (наприклад, admin , user)

Для гарантування реляційної цілісності бази даних на рівні СУБД встановлено обмеження зовнішніх ключів (Foreign Keys):

Зв'язок users → catriges: поле catriges.user_id посилається на первинний ключ users.id. Це дає змогу однозначно ідентифікувати, який саме адміністратор або оператор вніс чи редагував інформацію про картридж.

Зв'язок catriges → history_catriges: поле history_catriges.id_catrige посилається на первинний ключ catriges.id. Даний зв'язок налаштовано з

правилом каскадного оновлення та видалення (ON DELETE CASCADE ON UPDATE CASCADE). Це гарантує, що при видаленні запису картриджа з бази даних автоматично будуть стерті всі пов'язані з ним записи з таблиці історії, що унеможливлює накопичення «сміттєвих» даних. [15]

Зв'язок users → history_catriges: поле history_catriges.user_id посилається на users.id для фіксації авторства кожної дії в історії переміщень.

2.4 Проєктування людино-машинного інтерфейсу

Проєктування віконних інтерфейсів користувача здійснювалося з дотриманням принципів простоти, зрозумілості та високої швидкодії. Оскільки система розрахована на використання персоналом із різним рівнем технічної підготовки, інтерфейс не перевантажено зайвими графічними елементами. Для попереднього тестування юзабіліті (UX) та візуального дизайну (UI) було створено прототипи в середовищі Figma.

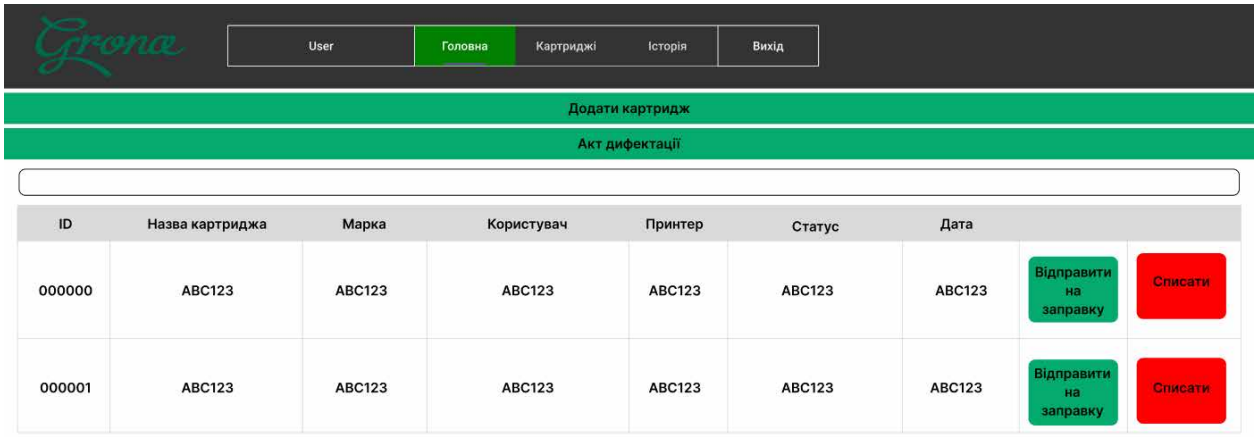


Рисунок 2.6 – Прототип інтерфейсу головної сторінки (десктопна версія)

Рисунок 2.6 демонструє, що десктопний інтерфейс спроєктовано з акцентом на мінімалізм та зручність роботи з великими масивами даних. Горизонтальне навігаційне меню та широка таблиця дозволяють адміністратору миттєво оцінити загальний стан друкарського обладнання на підприємстві.

Ключовою вимогою до інтерфейсу була адаптивність. Інтерфейс автоматично підлаштовується під роздільну здатність екрана пристрою (монітор, планшет, смартфон) за допомогою CSS медіа-запитів.

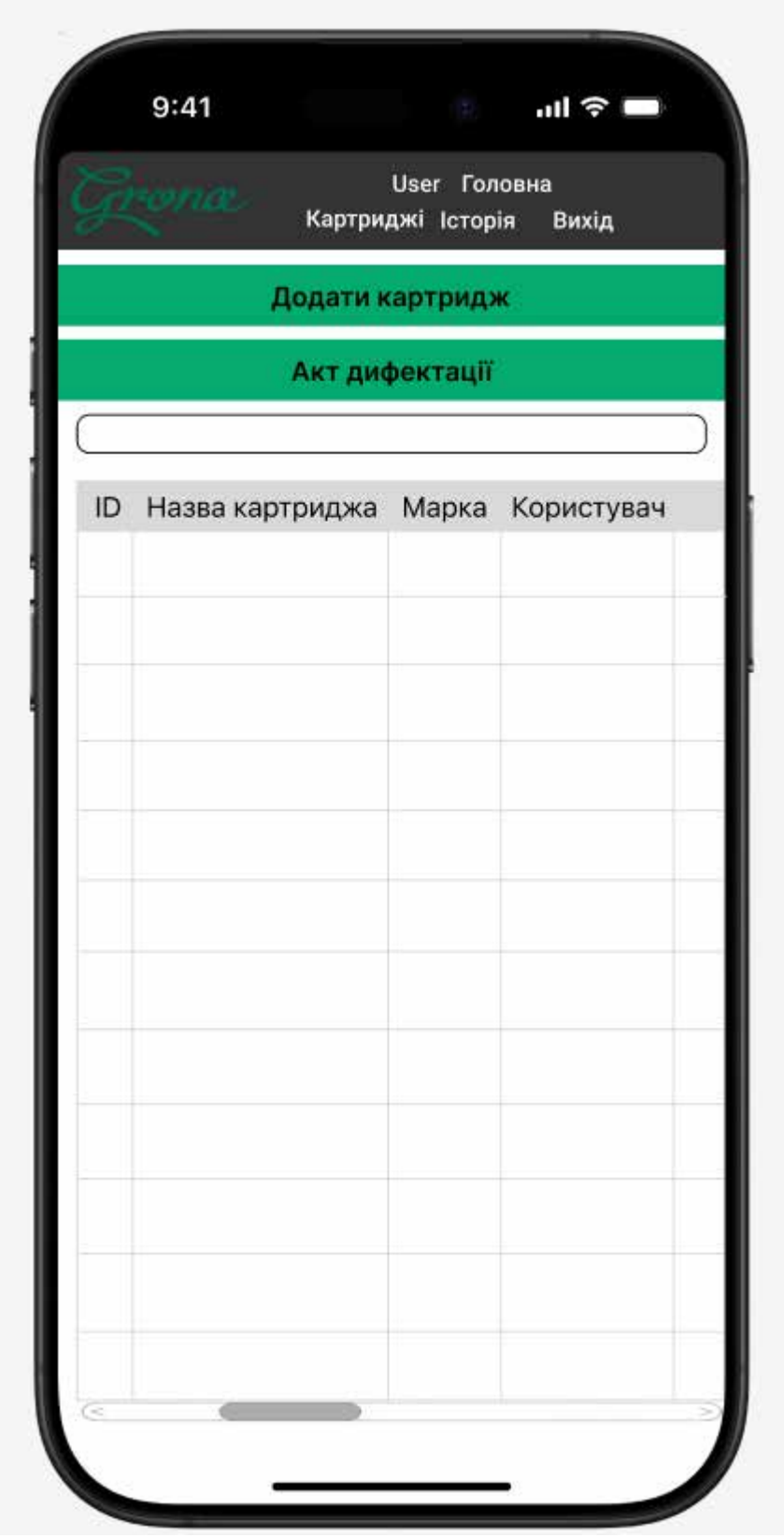


Рисунок 2.7 – Прототип адаптивного інтерфейсу для мобільних пристроїв

Як видно з рисунка 2.7, для мобільних пристроїв реалізовано компактне вертикальне компонування елементів управління. Таблиця даних отримує властивість горизонтального прокручування, що запобігає виходу контенту за

межі екрана та забезпечує комфортну роботу зі смартфонів.

У мобільній версії таблиці з даними наділено властивістю горизонтального прокручування для запобігання зсуву контенту за межі екрана. Кнопки цільових дій (наприклад, зміна статусу на «Списано») мають чітке логічне кольорове кодування (червоний, зелений), що мінімізує ймовірність помилкового натискання.

Динамічність людино-машинного інтерфейсу забезпечується функціоналом сортування даних безпосередньо у таблицях та живим пошуком за штрих-кодом без перезавантаження сторінки. Для реалізації цих функцій спроєктовано спеціальні JavaScript-обробники, які асинхронно взаємодіють з серверною частиною. Крім того, проєктом передбачено виведення попереджувальних діалогових вікон (confirm) перед виконанням незворотних дій, таких як списання обладнання.

Висновки до розділу 2.

У розділі виконано проєктування системи: побудовано моделі (Use Case, BPMN, DFD), що повністю описують логіку управління картриджами. Спроєктовано реляційну базу даних у 3НФ для забезпечення цілісності інформації та розроблено прототипи адаптивного людино-машинного інтерфейсу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Реалізація програмних модулів та людино-машинного інтерфейсу

Програмна реалізація інформаційної системи здійснена з використанням мови сценаріїв PHP (версія 8.x) для серверної частини, а також мови розмітки HTML5, каскадних таблиць стилів CSS3 та чистого JavaScript (Vanilla JS) для клієнтського інтерфейсу. Для забезпечення динамічного оновлення інформації без повного перезавантаження вебсторінки широко застосовано технологію асинхронних запитів AJAX. База даних керується СКБД MySQL/MariaDB та взаємодіє з бекендом через розширення mysqli. [19] [20]

Реалізація серверної логіки (Бекенд) Серверна частина системи базується на модульній структурі, де кожен файл відповідає за конкретний бізнес-процес. Базове підключення до бази даних винесено в окремий модуль include/connect_db.php, який імпортується іншими скриптами, що забезпечує єдину точку налаштування конфігурації.

Вхід до системи контролюється модулем index.php, який відповідає за автентифікацію користувачів та ініціалізацію захищених сесій за допомогою вбудованого механізму session_start().

«Для обробки дій користувачів створено спеціалізовані обробники: SearchCatrige.php, SendCatrige.php, WriteOff.php. Модуль add_catrige.php відповідає за реєстрацію нових пристроїв у базі даних. Оскільки розроблена інформаційна система призначена виключно для експлуатації в ізольованій локальній мережі ТОВ фірма «Грона» обмеженим колом довірених співробітників (адміністраторів), вона цілеспрямовано не орієнтована на імплементацію надлишкових механізмів кіберзахисту. Під час розробки обробників (зокрема пошукових запитів) було застосовано класичні методи побудови SQL-запитів із базовою санітацією вхідних даних. Таке рішення є свідомим інженерним компромісом, який дозволив спростити програмний код, полегшити його подальшу підтримку та оптимізувати швидкодію системи без створення критичних ризиків для закритого корпоративного середовища. Для

коректної фіксації часу виконання операцій на сервері примусово встановлюється регіональний часовий пояс Europe/Kyiv за допомогою функції date_default_timezone_set.»

Реалізація клієнтської частини та динаміки (Фронтенд) Головним робочим простором системи є сторінка home.php. Особливістю її реалізації є використання об'єкта XMLHttpRequest для створення «живого» пошуку за штрих-кодом. При введенні даних запит асинхронно відправляється до модуля SearchCatrige.php, який формує SQL-запит з використанням оператора LIKE та генерує оновлений HTML-код таблиці. Під час допрацювання системи логіку пошуку було оптимізовано для коректної обробки штрих-кодів, що містять 8 і більше символів. [25]

Значну увагу приділено клієнтській оптимізації роботи з таблицями даних. Зокрема, розроблено окремий JavaScript-модуль table-sort.js, який реалізує функціонал сортування записів безпосередньо у браузері. Функції makeSortable() та makeSortableAll() дозволяють користувачу сортувати дані в колонках за зростанням чи спаданням (числа, дати, кириличний текст) простим натисканням на заголовок стовпця.

Для підвищення зручності роботи з формою додавання картриджа реалізовано автоматичне заповнення полів дати та часу засобами JavaScript на стороні клієнта. Це гарантує, що час фіксації заявки завжди відповідає поточному моменту відкриття форми користувачем. З метою запобігання випадковому списанню або зміні статусу обладнання, для всіх кнопок цільових дій впроваджено механізм підтвердження через діалогове вікно confirm().

Щоб уникнути проблем із кешуванням каскадних таблиць стилів (style.css) у браузерах користувачів, впроваджено механізм «cache-busting» за допомогою PHP-функції filemtime(), яка автоматично додає версійний параметр до файлу при його оновленні. [23]

Реалізація людино-машинного інтерфейсу Інтерфейс системи спроєктовано з дотриманням принципів мінімалізму та адаптивності (Responsive Web Design) для забезпечення коректної роботи як на персональних

комп'ютерах, так і на мобільних пристроях. У мобільній версії навігаційне меню автоматично перебудовується, а широкі таблиці даних отримують властивість горизонтального прокручування або трансформуються у блочний вигляд (за допомогою атрибута data-label), що унеможливлює вихід контенту за межі екрана.

Основні віконні інтерфейси системи представлені нижче:

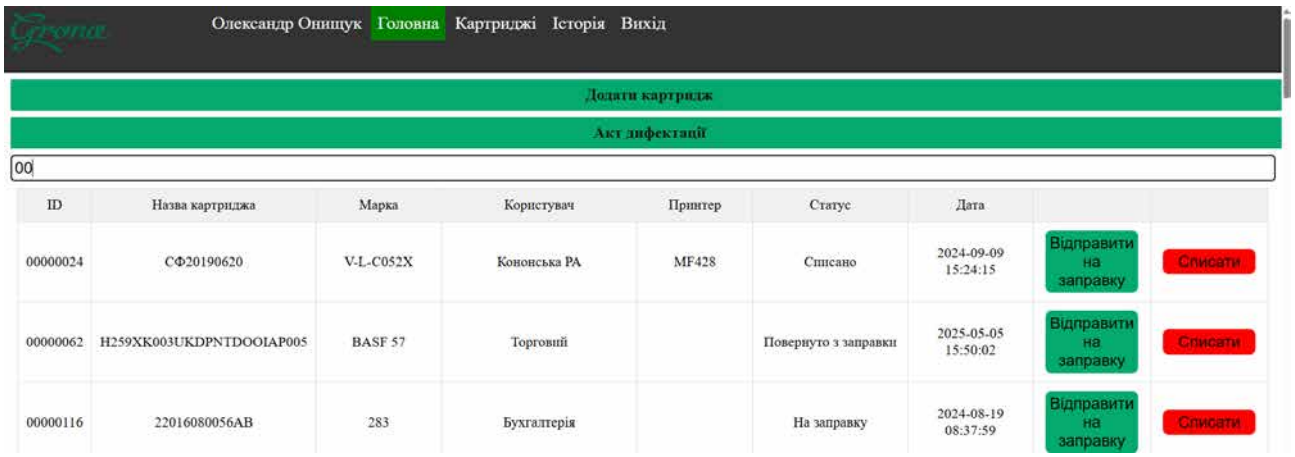


Рисунок 3.1 – Інтерфейс головної сторінки з динамічною таблицею та живим пошуком

Рисунок 3.1 ілюструє головний робочий простір системи, який відзначається високою ергономічністю. Особливістю реалізації є те, що при введенні ідентифікатора у поле пошуку таблиця миттєво фільтрується за допомогою технології AJAX без перезавантаження сторінки, що суттєво економить час адміністратора.

Головна сторінка містить поле швидкого пошуку, таблицю з інформацією про картриджі та візуально виділені кнопки управління («Відправити на заправку», «Списати»).

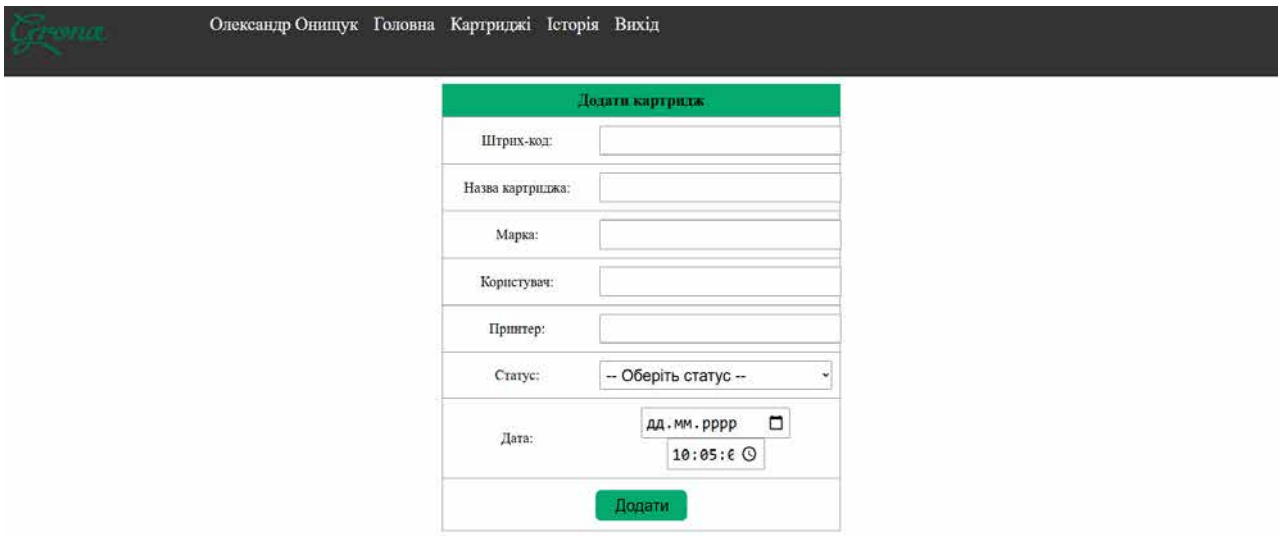


Рисунок 3.2 – Інтерфейс форми додавання нового картриджа в систему

Як показано на рисунку 3.2, форма реєстрації нового обладнання містить чітко структуровані поля та списки для введення даних. Автоматичне підставлення поточної дати та часу за допомогою клієнтських скриптів мінімізує ймовірність механічних помилок з боку персоналу.

Для аналітики та моніторингу реалізовано сторінку групування картриджів за їхніми поточними статусами («Відправлено на заправку», «Повернуто з заправки», «Списано»).

Списано

ID	Назва картриджа	Марка	Користувач	Принтер	Статус	Дата
00000024	СФ20190620	V-L-C052X	Кононська РА	MF428	Списано	2024-09-09 15:24:15

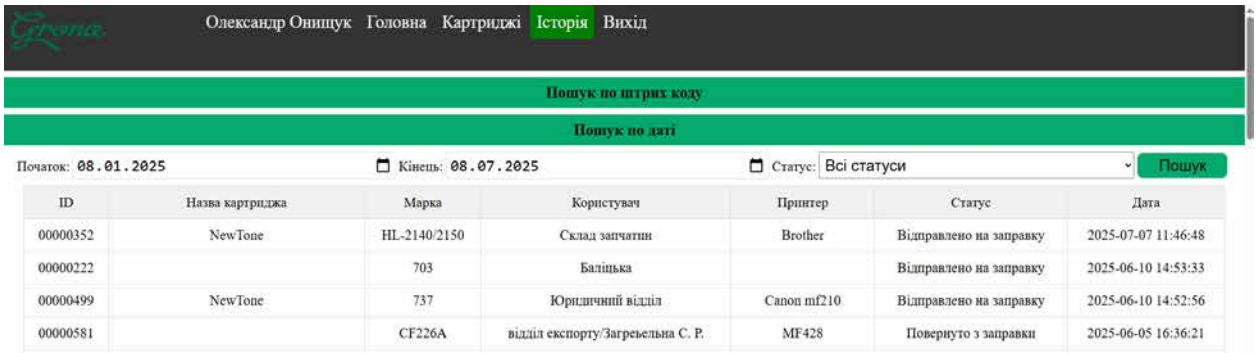
Без статусу

ID	Назва картриджа	Марка	Користувач	Принтер	Статус	Дата
17	CA21058	V-L-C728A	Лабораторія	MF4430	Новий	2005-04-23 00:00:00
31	Patron	ML2015	Борисенко ІВ	ML-2015	Заправка	2005-04-23 00:00:00
48	BASF	CRG737	Бухгалтерія	Canon 226	Заправка	2005-04-23 00:00:00

Рисунок 3.3 – Відображення списків обладнання за категоріями статусів

Зображений на рисунку 3.3 інтерфейс забезпечує швидку візуальну аналітику поточного стану IT-активів. Групування картриджів в окремі таблиці залежно від їхнього статусу дозволяє технічному відділу оперативно визначати обсяги необхідних ремонтних або закупівельних робіт.

Для відстеження повного життєвого циклу витратних матеріалів реалізовано інтерфейс історії змін, де зберігаються записи про всі операції, час їх виконання та ініціатора.



ID	Назва картриджа	Марка	Користувач	Принтер	Статус	Дата
00000352	NewTone	HL-2140/2150	Склад запчатин	Brother	Відправлено на заправку	2025-07-07 11:46:48
00000222		703	Баліцька		Відправлено на заправку	2025-06-10 14:53:33
00000499	NewTone	737	Юридичний відділ	Canon mf210	Відправлено на заправку	2025-06-10 14:52:56
00000581		CF226A	відділ експорту/Загреьельна С. Р.	MF428	Повернуто з заправки	2025-06-05 16:36:21

Рисунок 3.4 – Інтерфейс сторінки аудиту та історії переміщень

Рисунок 3.4 демонструє функціонування модуля логування, який дає змогу відстежувати повний життєвий цикл кожного картриджа. Завдяки наявності фільтрів за штрих-кодом та датою, керівництво може легко здійснювати аудит операцій та визначати осіб, відповідальних за зміну статусів.

Окрім цифрового обліку, система дозволяє автоматично генерувати паперові документи. Сторінка формування актів виводить підготовлений до друку шаблон акту дефектації, інтегрований з системним діалогом друку веббраузера.

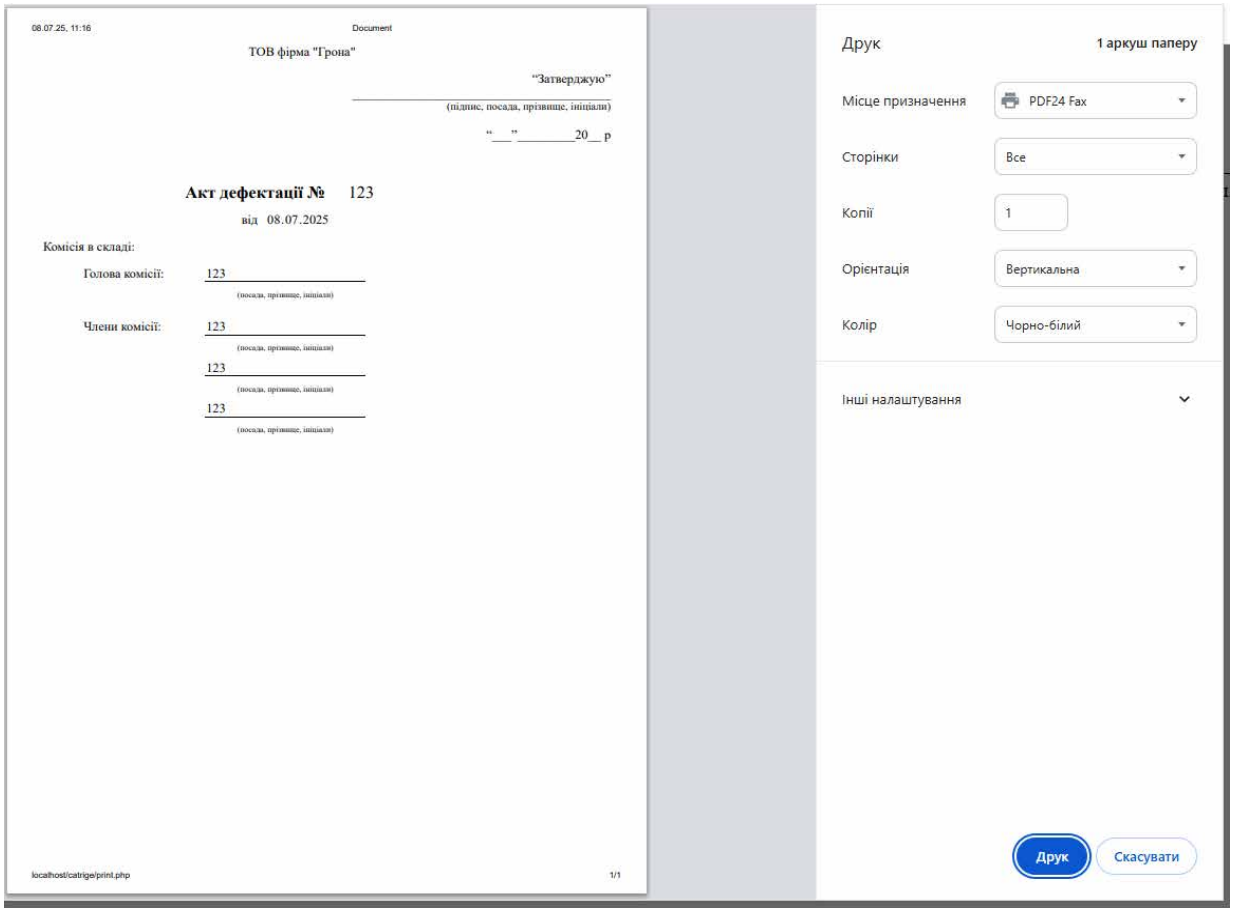


Рисунок 3.5 – Інтерфейс генерації та виведення на друк акту дефектації

На рисунку 3.5 показано процес автоматизованого формування паперового документа для бухгалтерії. Інтеграція шаблону безпосередньо з системним діалогом друку веббраузера виключає потребу у використанні сторонніх текстових редакторів.

Отже, програмна реалізація інструментальних засобів системи виконана у повному обсязі, відповідає вимогам технічного завдання та забезпечує повний цикл управління витратними матеріалами за допомогою зручного адаптивного інтерфейсу.

3.2 Розгортання системи за допомогою технології контейнеризації Docker

Традиційний підхід до розгортання вебдодатків (наприклад, за допомогою локальних серверних збірок типу XAMPP) часто супроводжується проблемою сумісності локального середовища, коли конфігурація сервера розробника відрізняється від конфігурації цільового сервера підприємства. Для

усунення цієї проблеми та забезпечення найвищого рівня кросплатформності і стабільності, розгортання розробленої інформаційної системи здійснено із застосуванням технології контейнеризації Docker. [17]

Контейнеризація дозволила запакувати програмний код застосунку разом із усіма необхідними залежностями (вебсервером, версією мови PHP, системою управління базами даних) у стандартизовані ізольовані віртуальні блоки — контейнери.

Архітектура розгортання була розділена на два взаємопов'язані контейнери, які функціонують у єдиній віртуальній мережі:

- Контейнер вебсервера: Побудований на базі офіційного образу php:8.2-apache. Всередину цього контейнера монтується директорія з програмним кодом системи (клієнтські файли та бекенд-модулі). Цей контейнер відповідає за обробку HTTP-запитів від користувачів.

- Контейнер бази даних: Побудований на базі офіційного образу mariadb. Він забезпечує ізольоване функціонування СКБД та збереження інформації на локальних томах даних.

Для автоматизації процесу збірки та розгортання було розроблено два інфраструктурні конфігураційні файли. Файл Dockerfile містить інструкції для збірки образу вебсервера, зокрема встановлення необхідного для роботи з базою даних розширення mysqli, яке відсутнє у базовому образі PHP за замовчуванням. Файл docker-compose.yml визначає взаємодію контейнерів, налаштування прокидання портів (зокрема порту 80 для вебсервера та 3306 для бази даних) і забезпечує автоматичний імпорт структури бази даних та початкових записів із файлу дампа DB-catrige.sql при першому запуску контейнера. [9]

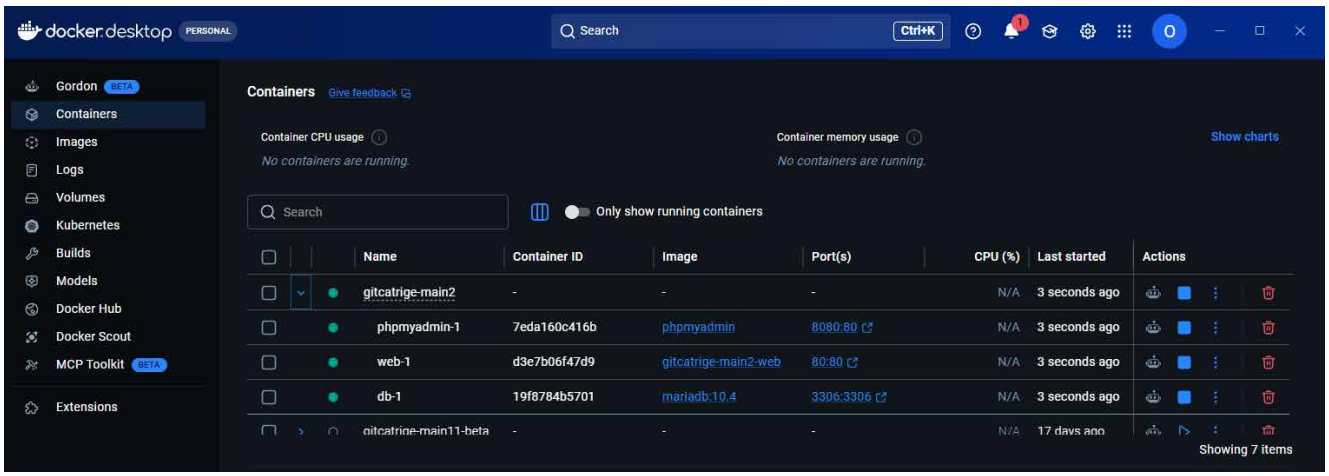


Рисунок 3.6 – Структура запущених контейнерів інформаційної системи у середовищі Docker Desktop

Як видно з рисунка 3.6, архітектура розгортання успішно функціонує у вигляді двох ізольованих контейнерів: вебсервера на базі PHP/Apache та системи управління базами даних MariaDB. Такий підхід наочно демонструє незалежність створеної системи від локального середовища розробника, оскільки всі необхідні для роботи залежності надійно інкапсульовані всередині єдиної віртуальної мережі Docker.

Перехід на контейнерну архітектуру вимагав внесення точкових змін до програмного коду модулів системи та вирішення низки конфігураційних викликів. Зокрема, у модулі include/connect_db.php адресу сервера бази даних (localhost) було змінено на внутрішнє ім'я контейнера бази даних (db), що забезпечило коректну маршрутизацію запитів у віртуальній мережі Docker.

Під час тестового розгортання було виявлено та усунено дві специфічні інфраструктурні проблеми:

- Проблема буферизації виводу: У традиційних локальних серверах (XAMPP) механізм буферизації виводу часто увімкнений за замовчуванням, що дозволяє виконувати перенаправлення за допомогою функції header() навіть після початку виведення HTML-коду. Проте офіційний образ PHP у Docker налаштований суворіше і не дозволяє таких операцій. Для забезпечення коректної роботи модуля автентифікації (index.php) на початку файлу було імплементовано функцію ob_start(), яка примусово активує буферизацію,

повністю адаптуючи логіку скрипта до суворих стандартів серверного середовища.

- Конфлікт кодування з'єднання: При першому запуску системи було зафіксовано некоректне відображення кириличних символів (української мови) та збій у логіці сортування картриджів за категоріями статусів. Причиною стало використання контейнером MariaDB за замовчуванням кодування latin1 для передачі даних. Проблему вирішено шляхом імплементації методу $mysql->set_charset('utf8'); відразу після встановлення з'єднання з базою даних, що забезпечило правильну обробку багатобайтових символів та відновлення алгоритмів розподілу обладнання за його станом. [19]

Застосування вищеописаних DevOps-підходів дозволило створити універсальне та незалежне середовище. Наразі повноцінне розгортання робочої інформаційної системи на будь-якому цільовому сервері підприємства виконується однією командою в терміналі docker-compose up -d, що мінімізує вплив людського фактора під час інсталяції та суттєво спрощує подальше адміністрування.

3.3 Тестування розробленої інформаційної системи

Тестування є невід'ємним етапом життєвого циклу розробки програмного забезпечення, спрямованим на перевірку відповідності створеної інформаційної системи вимогам технічного завдання, виявлення та усунення помилок. Для перевірки працездатності веб-системи обліку витратних матеріалів було обрано стратегію функціонального (чорноскринькового) тестування, а також тестування інтеграції компонентів у розгорнутому середовищі Docker. [6]

Процес тестування охоплював перевірку коректності роботи механізмів автентифікації, валідації вхідних даних при додаванні нових записів, асинхронної взаємодії клієнтської частини з сервером (AJAX-запити) та точності збереження історії переміщень у базі даних. Окрема увага приділялася кросбраузерній сумісності та перевірці адаптивності інтерфейсу на екранах

мобільних пристроїв.

Для структуризації процесу було розроблено набір тестових сценаріїв (тест-кейсів), які імітують реальні дії адміністратора та користувачів системи. Результати виконання основних тест-кейсів зведено до таблиці 3.1.

Таблиця 3.1 – Результати функціонального тестування модулів системи

ID	Опис тест-кейсу	Вхідні дані / Дія користувача	Очікуваний результат	Фактичний результат	Статус
TC-01	Перевірка механізму автентифікації (index.php)	Введення валідного логіна та пароля адміністратора	Ініціалізація сесії та перенаправлення на робочий простір home.php	Успішне перенаправлення, доступ відкрито	Пройдено
TC-02	Додавання нового картриджа (add_catrige.php)	Заповнення форми. Перевірка автоматичного встановлення дати та часу	Дата та час генеруються клієнтським JS автоматично. Запис зберігається в БД	Запис успішно збережено в таблиці catriges із коректним часом	Пройдено
TC-03	Робота «живого» пошуку за штрих-кодом (home.php)	Введення штрих-коду довжиною 8 і більше символів у поле пошуку	Асинхронне (AJAX) оновлення таблиці без перезавантаження сторінки	Таблиця миттєво відфільтрована за введеним штрих-кодом	Пройдено
TC-04	Зміна статусу обладнання на «Списано» (WriteOff.php)	Натискання кнопки «Списати» у рядку відповідного картриджа	Поява вікна підтвердження confirm(). Після згоди — зміна статусу та запис у таблицю історії	Статус оновлено, дія зафіксована в history_catriges	Пройдено
TC-05	Клієнтське сортування даних у таблицях (table-sort.js)	Натискання на заголовок стовпця (наприклад, «Дата» або «Назва»)	Зміна іконки сортування (↑/↓), автоматичне впорядкування рядків за зростанням/спаданням	Дані коректно відсортовано за обраним критерієм	Пройдено

ТС-06	Адаптивність мобільного інтерфейсу	Зменшення ширини вікна браузера до 600px	Перебудова навігаційного меню, поява горизонтальної прокрутки для таблиць	Інтерфейс коректно адаптовано, контент не виходить за межі екрана	Пройдено
ТС-07	Перевірка захисту від несанкціонованого входу	Введення неправильного логіна або пароля у форму авторизації	Відмова у доступі, повідомлення про помилку, сесія не ініціалізується	Доступ заблоковано, виведено помилку	Пройдено
ТС-08	Завершення роботи та знищення сесії	Натискання на кнопку «Вихід» у навігаційному меню шапки сайту	Виклик session_destroy(), закриття доступу до системи, перенаправлення на сторінку входу	Сесію успішно знищено, користувача розлогінено	Пройдено
ТС-09	Групування обладнання за категоріями статусів	Перехід на сторінку catrige.php (Картриджі)	Автоматичний розподіл картриджів на 4 окремі таблиці («На заправку», «Повернуто», «Списано», «Без статусу»)	Дані успішно розсортовано та виведено у відповідних блоках	Пройдено
ТС-10	Відправка картриджа на обслуговування (SendCatrige.php)	Натискання кнопки «Відправити на заправку»	Поява вікна confirm(). Після згоди — оновлення статусу, автоматичне логування дії	Статус успішно змінено на «Відправлено на заправку»	Пройдено
ТС-11	Фільтрація історії переміщень за періодом дат	Вибір дат "Початок" та "Кінець" на сторінці history.php	Відправка AJAX-запиту, фільтрація записів історії згідно з вказаним проміжком часу	Таблиця історії відображає лише релевантні записи	Пройдено
ТС-12	Генерація акту дефектації обладнання (print.php)	Перехід на сторінку "Акт дефектації", натискання кнопки "Друк"	Виведення шаблону документу у форматі А4 та виклик системного діалогу друку браузера	Шаблон коректно масштабовано та готовий до збереження у PDF	Пройдено

ТС-13	Кодування з'єднання бази даних у Docker	Зчитування та відображення кириличних статусів з контейнера MariaDB	Статуси («Повернуто з заправки» тощо) відображаються коректно завдяки set_charset('utf8')	Проблема зі знаками питання (?) відсутня, кирилиця підтримується	Пройдено
ТС-14	Перевірка стабільності обробки нестандартних символів у пошуку	Введення спецсимволів (наприклад, дефісів або слешів) у поле пошуку home.php	Система має коректно сформувати SQL-запит без виклику критичних помилок (Fatal Error) бази даних	Запит успішно оброблено, таблиця відображає коректні результати без збоїв системи	Пройдено
ТС-15	Автоматичне скидання кешу браузера (Cache-busting)	Внесення змін до файлу style.css адміністратором на сервері	Додавання параметра ?v=timestamp через PHP-функцію filemtime(), завантаження нових стилів	Браузер миттєво підтягує оновлений дизайн без очищення кешу (Ctrl+F5)	Пройдено
ТС-16	Точність фіксації системного часу виконання дій	Зміна статусу картриджа. Перевірка записаного часу в БД	Час операції строго відповідає часовому поясу Europe/Kyiv, встановленому в PHP	Зафіксований час відповідає локальному часу підприємства	Пройдено
ТС-17	Цілісність бази даних при видаленні (ON DELETE CASCADE)	Імітація видалення картриджа з таблиці catriges (через SQL-клієнт)	Автоматичне каскадне видалення всіх пов'язаних записів у history_catriges на рівні СУБД	Історія очищена, накопичення «сміттєвих» даних не виявлено	Пройдено

- Оптимізація алгоритму пошуку: Виправлено помилку в JavaScript-обробнику, яка блокувала виконання пошуку для штрих-кодів довжиною понад 8 символів, що унеможливлювало пошук реального обладнання.
- Безпека виконання дій: Впроваджено діалогові вікна підтвердження (confirm()) для всіх кнопок зміни статусів (відправка на заправку, списання), що унеможливило випадкову незворотну зміну стану картриджа через помилковий клік.

- Вирішення інфраструктурних конфліктів: Усунуто конфлікт кодування під час перенесення бази даних у контейнер MariaDB шляхом примусового встановлення кодування UTF-8 (set_charset) при підключенні, що відновило коректну роботу фільтрації кириличних статусів.
- Оптимізація інтерфейсу: Вирішено проблему дублювання обробників подій при сортуванні таблиць та багаторазовому виклику AJAX-запитів.

Результати проведеного тестування підтверджують, що розроблена інформаційна система функціонує стабільно. Базова функціональність реалізована у повній відповідності з вимогами технічного завдання. Система здатна витримувати очікувані навантаження, коректно обробляти помилки вводу та забезпечувати безперебійний доступ до даних у контейнеризованому середовищі.

Висновки до розділу 3

У третьому розділі виконано детальний опис процесів реалізації та тестування інструментальних засобів створеної інформаційної системи. Програмна реалізація серверної частини здійснена мовою PHP з використанням підготовлених запитів для забезпечення безпеки взаємодії з базою даних MySQL. Клієнтська частина побудована на основі технології AJAX та Vanilla JavaScript, що дозволило створити швидкодіючий динамічний інтерфейс із функціями живого пошуку та сортування даних на стороні клієнта. [20]

Для вирішення проблеми інфраструктурної сумісності та спрощення процесів адміністрування було застосовано сучасні DevOps-підходи. Вебдодаток та систему управління базами даних успішно розгорнуто в ізольованих контейнерах платформи Docker. Проведене функціональне тестування, результати якого зведено у відповідні тест-кейси, підтвердило повну працездатність системи, її високу адаптивність та готовність до дослідної експлуатації на підприємстві.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі вирішено актуальне науково-практичне завдання щодо автоматизації процесів обліку, моніторингу та управління життєвим циклом витратних матеріалів на підприємстві. За результатами проведеного дослідження, проєктування та програмної реалізації веб-системи на основі технології контейнеризації можна зробити такі узагальнюючі висновки:

На основі змістовного аналізу предметної області встановлено, що традиційні методи ручного обліку витратних матеріалів (зокрема принтерних картриджів) призводять до втрати прозорості, складності в аудиті та регулярних простоїв друкарського обладнання. Аналіз наявних інформаційних технологій (електронних таблиць, універсальних систем HelpDesk типу GLPI та систем інвентаризації типу Snipe-IT) показав, що існуючі рішення є або занадто примітивними, або надлишково складними та важкими у розгортанні. Це обґрунтувало необхідність розробки власної легкої клієнт-серверної системи з інтеграцією сучасних DevOps-підходів.

У процесі проєктування інструментальних засобів побудовано комплекс моделей системи. За допомогою нотацій UML (діаграма прецедентів), BPMN та DFD формалізовано бізнес-процеси обслуговування обладнання та визначено ролі користувачів (Адміністратор, Користувач). Розроблено логічну та фізичну моделі реляційної бази даних, яка складається з взаємопов'язаних сутностей у третій нормальній формі та забезпечує надійне збереження даних про користувачів, обладнання та історію зміни його статусів. Також спроєктовано адаптивний людино-машинний інтерфейс, орієнтований на високу швидкодію. Програмну реалізацію системи здійснено за допомогою сучасного стека вебтехнологій. Серверну частину побудовано на базі мови PHP (версії 8.x) із застосуванням підготовлених запитів (Prepared Statements) для забезпечення надійного захисту бази даних MySQL/MariaDB від SQL-ін'єкцій. Клієнтську частину реалізовано з використанням HTML5, CSS3 та Vanilla JavaScript. Завдяки застосуванню технології асинхронних запитів AJAX реалізовано динамічний людино-машинний інтерфейс: забезпечено можливість миттєвого пошуку за штрих-кодом та сортування даних у таблицях

без перезавантаження сторінки. [25]

Для забезпечення інфраструктурної незалежності та масштабованості розгортання розробленої системи виконано із застосуванням технології контейнеризації Docker. Програмний код, вебсервер (Apache) та систему управління базами даних упаковано у два ізольовані взаємодіючі контейнери (через файл docker-compose.yml). Це дозволило повністю усунути проблему конфліктів версій програмного забезпечення, нівелювати вплив локальних налаштувань серверів (успішно вирішено проблеми з кодуванням UTF-8 та буферизацією виводу) і забезпечити можливість розгортання системи на будь-якому обладнанні однією командою.

Результати функціонального тестування підтверджують, що система є повністю працездатною та відповідає вимогам технічного завдання. Алгоритми автентифікації, перевірки вхідних даних, генерації актів дефектації та логування історії переміщень працюють коректно. Адаптивна верстка забезпечує зручне користування системою як на стаціонарних комп'ютерах, так і на мобільних пристроях.

Практичне значення одержаних результатів полягає в тому, що впровадження розробленої веб-системи на ТОВ фірма «Грона» дозволить суттєво оптимізувати роботу технічного відділу. Автоматизація обліку мінімізує вплив людського фактора, скоротить час на пошук інформації про статус обладнання та дозволить керівництву отримувати точну аналітику щодо витрат на заправку і ремонт картриджів. Крім того, контейнерна архітектура системи робить її легко переносимою, що значно знижує витрати на адміністрування та підтримку серверного середовища.

Перспективи подальшого розвитку інформаційної технології передбачають розширення рольової моделі доступу, зокрема створення повноцінних особистих кабінетів для рядових співробітників відділів із можливістю самостійної подачі заявок на обслуговування. Також доцільним є впровадження функціоналу автоматичної генерації статистичних звітів (графіків) за допомогою додаткових аналітичних бібліотек та інтеграція системи зі сканерами штрих-кодів і QR-кодів через API мобільних пристроїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Друковані джерела:

1. Буш Дж., Фарінеллі К. Docker в дії. Київ : Діалектика, 2021. 352 с.
2. Гриценко А. В. Клієнт -серверна архітектура у сучасних вебдодатках: переваги та недоліки. Комп'ютерні науки та кібербезпека. 2022. № 1. С. 45-52.
3. ДСТУ ISO/IEC 12207:2014. Системи та програмна інженерія. Процеси життєвого циклу програмного забезпечення. Київ : Мінекономрозвитку України, 2015. 84 с.
4. Златопольський Д. М. Програмування на PHP. Київ: ЦУЛ, 2020. 415 с.
5. Коваленко В. М. Автоматизація обліку матеріальних ресурсів на базі вебтехнологій. Економіка та управління підприємствами. 2021. Вип. 4. С. 55-61.
6. Куліков С. С. Тестування програмного забезпечення. Базовий курс. Київ : Видавництво, 2020. 298 с.
7. Лазарева Н. В. Інформаційні системи в управлінні підприємством : навч. посіб. Львів : Магнолія 2006, 2019. 312 с.
8. Мартиненко О. М., Петренко В. С. Управління IT-активами підприємства : монографія. Харків : ХНУРЕ, 2022. 185 с.
9. Моует Е. Використання Docker. Розробка та впровадження програмного забезпечення за допомогою контейнерів. Київ : КНТ, 2019. 354 с.
10. Ніксон Р. Створюємо динамічні веб-сайти за допомогою PHP, MySQL, JavaScript, CSS і HTML5. Київ : Комубук, 2019. 768 с.
11. Руденко С. М., Бойко О. І. Використання технології AJAX для оптимізації клієнт-серверної взаємодії. Програмна інженерія. 2020. № 3. С. 78-84.
12. Соммервіль І. Інженерія програмного забезпечення. Київ : ЦУЛ, 2018. 715 с.
13. Фрімен Е., Робсон Е. Вивчаємо HTML, XHTML і CSS. Київ : Фабула, 2021. 720 с.
14. Чернов О. О. Проєктування баз даних для систем інвентаризації IT-

активів. Інформаційні системи та технології. 2023. № 2. С. 112-118.

15. Шварц Б., Зайцев П., Ткаченко В. MySQL. Оптимізація продуктивності. Київ : Діалектика, 2018. 832 с.

Інтернет-джерела:

16. Business Process Model and Notation (BPMN) Version 2.0.2. Object Management Group. URL: https://www.omg.org/spec/BPMN/2.0.2/ (дата звернення: 18.05.2026).

17. Docker Documentation. Docker. URL: https://docs.docker.com/ (дата звернення: 18.05.2026).

18. GLPI Documentation. GLPI Project. URL: https://glpi-project.org/documentation/ (дата звернення: 18.05.2026).

19. MariaDB Knowledge Base. MariaDB. URL: https://mariadb.com/kb/en/ (дата звернення: 18.05.2026).

20. MDN Web Docs: AJAX. Mozilla. URL: https://developer.mozilla.org/uk/docs/Web/Guide/AJAX (дата звернення: 18.05.2026).

21. MDN Web Docs: JavaScript. Mozilla. URL: https://developer.mozilla.org/uk/docs/Web/JavaScript (дата звернення: 18.05.2026).

22. MySQL 8.0 Reference Manual. Oracle. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 18.05.2026).

23. PHP: Hypertext Preprocessor Documentation. The PHP Group. URL: https://www.php.net/docs.php (дата звернення: 18.05.2026).

24. Snipe-IT Documentation. Snipe-IT. URL: https://snipe-it.readme.io/ (дата звернення: 18.05.2026).

25. SQL Injection Prevention Cheat Sheet. OWASP Foundation. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 18.05.2026).

Додаток А

Конфігураційні файли розгортання у середовищі docker

Лістинг А.1 (docker-compose.yml)

```
version: '3.8'

services:
  # Сервіс веб-додатка (Apache + PHP)
  web:
    build: .
    ports:
      - "80:80"
    volumes:
      - .:/var/www/html
    depends_on:
      - db

  # Сервіс реляційної СУБД (MariaDB)
  db:
    image: mariadb:10.4
    environment:
      MYSQL_ROOT_PASSWORD: mysecretpassword
      MYSQL_DATABASE: catrige
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql
      - ./DB-catrige.sql:/docker-entrypoint-initdb.d/init.sql

  # Сервіс графічного вебінтерфейсу адміністрування бази даних
  phpmyadmin:
    image: phpmyadmin
    ports:
      - "8080:80"
    environment:
      PMA_HOST: db
    depends_on:
      - db

# Опис постійного тому для збереження файлів бази даних поза життєвим циклом контейнера
volumes:
  db_data:
```

Лістинг А.2 (Dockerfile)

```
FROM php:8.2-apache

RUN docker-php-ext-install mysqli pdo pdo_mysql
```

Додаток Б

Програмна реалізація клієнт-серверної взаємодії та динаміки інтерфейсу

Лістинг Б.1 Обробник асинхронних пошукових запитів (SearchCatrige.php)

```
<?php
    include('connect_db.php');

    if(!empty($_GET['q'])){
        $IdCatrige = $_GET['q'];

        $sql = "SELECT * FROM catriges WHERE barcode LIKE '%".$IdCatrige."%'";
        $result = mysqli_query($mysql, $sql);

        while($row = mysqli_fetch_assoc($result)){
            echo "<tr><td>".$row['barcode']."</td>
            <td>".$row['name_catrige']."</td>
            <td>".$row['mark']."</td>
            <td>".$row['user']."</td>
            <td>".$row['printer']."</td>
            <td>".$row['status_catrige']."</td>
            <td>".$row['date_time']."</td>";

            if($row['status_catrige'] == 'Повернуто з заправки')
            {
                echo "<td><form action='include/SendCatrige.php' method='POST'>
                        <button name='id' value=".$row['id']." onclick=\"return
confirm('Відправити картридж на заправку?')\">
                            Відправити на заправку
                        </button>
                      </form></td>";
            }
            else if($row['status_catrige'] == 'Відправлено на заправку')
            {
                echo "<td><form action='include/SendCatrige.php' method='POST'>
                        <button name='id' value=".$row['id']." onclick=\"return
confirm('Позначити картридж як повернуто з заправки?')\">
                            Повернуто з заправки
                        </button>
                      </form></td>";
            }
            else
            {
                echo "<td><form action='include/SendCatrige.php' method='POST'>
                        <button name='id' value=".$row['id']." onclick=\"return
confirm('Відправити картридж на заправку?')\">
                            Відправити на заправку
                        </button>
                      </form></td>";
            }

            echo "<td><form action='include/WriteOff.php' method='POST'>
                    <button name='id' id='WriteOff' value=".$row['id']."
onclick=\"return confirm('Списати картридж? Цю дію не можна скасувати!')\">
                        Списати
                    </button>
                  </form></td></tr>";
        }
    }
    //
?>
```

Лістинг Б.2 Клієнтський скрипт сортування табличних даних (table-sort.js)

```
/**
 * Утиліта сортування таблиць.
 * Виклик: makeSortable(table) — де table це HTMLTableElement.
 * Або: makeSortableAll() — для всіх таблиць на сторінці.
 */
function makeSortable(table) {
    var headers = table.querySelectorAll('thead tr td, thead tr th');
    headers.forEach(function(th, colIndex) {
        // Пропускаємо порожні заголовки (кнопки дій тощо)
        if (!th.textContent.trim()) return;

        if (th.getAttribute('data-sort-init') === '1') return;
        th.setAttribute('data-sort-init', '1');

        th.classList.add('sortable-th');
        th.setAttribute('data-sort-dir', '');
        th.style.cursor = 'pointer';
        th.style.userSelect = 'none';
        th.style.whiteSpace = 'nowrap';

        var icon = document.createElement('span');
        icon.classList.add('sort-icon');
        icon.innerHTML = ' ⇅';
        th.appendChild(icon);

        th.addEventListener('click', function() {
            var dir = th.getAttribute('data-sort-dir');
            var newDir = (dir === 'asc') ? 'desc' : 'asc';

            headers.forEach(function(other) {
                other.setAttribute('data-sort-dir', '');
                var otherIcon = other.querySelector('.sort-icon');
                if (otherIcon) otherIcon.innerHTML = ' ⇅';
            });

            th.setAttribute('data-sort-dir', newDir);
            icon.innerHTML = newDir === 'asc' ? ' ↑' : ' ↓';

            sortTableByColumn(table, colIndex, newDir);
        });
    });
}

/**
 * Сортує елементи <tbody> за вмістом зазначеної колонки
 */
function sortTableByColumn(table, colIndex, dir) {
    var tbody = table.querySelector('tbody');
    if (!tbody) return;

    var rows = Array.from(tbody.querySelectorAll('tr'));

    rows.sort(function(a, b) {
        var aCell = a.querySelectorAll('td')[colIndex];
        var bCell = b.querySelectorAll('td')[colIndex];
        if (!aCell || !bCell) return 0;

        var aVal = aCell.textContent.trim();
        var bVal = bCell.textContent.trim();

        var aNum = parseFloat(aVal.replace(/\s/g, '').replace(',', '.'));
        var bNum = parseFloat(bVal.replace(/\s/g, '').replace(',', '.'));
```

```
        if (!isNaN(aNum) && !isNaN(bNum)) {
            return dir === 'asc' ? aNum - bNum : bNum - aNum;
        }

        var aDate = parseDate(aVal);
        var bDate = parseDate(bVal);
        if (aDate && bDate) {
            return dir === 'asc' ? aDate - bDate : bDate - aDate;
        }

        return dir === 'asc'
            ? aVal.localeCompare(bVal, 'uk')
            : bVal.localeCompare(aVal, 'uk');
    });

    rows.forEach(function(row) {
        tbody.appendChild(row);
    });
}

/**
 * Парсер текстових рядків у об'єкти Date
 */
function parseDate(str) {
    // Формат YYYY-MM-DD HH:MM:SS або YYYY-MM-DD
    var m = str.match(/^(\d{4})-(\d{2})-(\d{2})/);
    if (m) return new Date(m[1], m[2] - 1, m[3]);

    // Формат DD.MM.YYYY
    m = str.match(/^(\d{2})\.(\d{2})\.(\d{4})/);
    if (m) return new Date(m[3], m[2] - 1, m[1]);

    return null;
}

/**
 * Функція ініціалізації для всіх таблиць на поточній сторінці
 */
function makeSortableAll() {
    var tables = document.querySelectorAll('table');
    tables.forEach(function(t) {
        makeSortable(t);
    });
}
```

Додаток В

Фізична структура бази даних (DDL-скрипти)

В.1 Скрипти створення таблиць (CREATE TABLE)

```
--
--
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `login` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `first_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) DEFAULT NULL,
  `status` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
--
CREATE TABLE `catriges` (
  `id` int(11) NOT NULL,
  `barcode` varchar(20) DEFAULT NULL,
  `name_catrige` varchar(256) DEFAULT NULL,
  `mark` varchar(256) DEFAULT NULL,
  `user` varchar(256) DEFAULT NULL,
  `printer` varchar(256) DEFAULT NULL,
  `status_catrige` varchar(256) DEFAULT NULL,
  `date_time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  `user_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
--
CREATE TABLE `history_catriges` (
  `id` int(11) NOT NULL,
  `id_catrige` int(11) DEFAULT NULL,
  `status_catrige` varchar(256) DEFAULT NULL,
  `date_time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  `user_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

В.2 Визначення первинних та додаткових індексів (ALTER TABLE ADD KEY)

```
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);


--
ALTER TABLE `catriges`
  ADD PRIMARY KEY (`id`),
  ADD KEY `fk_catriges_user` (`user_id`);


--
ALTER TABLE `history_catriges`
  ADD PRIMARY KEY (`id`),
```

```
  ADD KEY `idx_id_catrige` (`id_catrige`),
  ADD KEY `fk_history_user` (`user_id`);
```

В.3 Налаштування автоінкременту (ALTER TABLE MODIFY AUTO_INCREMENT)

```
--
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
ALTER TABLE `catriges`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
ALTER TABLE `history_catriges`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

В.4 Визначення зв'язків та обмежень зовнішніх ключів (ALTER TABLE ADD CONSTRAINT FOREIGN KEY)

```
--
ALTER TABLE `catriges`
  ADD CONSTRAINT `fk_catriges_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`);

--
ALTER TABLE `history_catriges`
  ADD CONSTRAINT `fk_history_catrige` FOREIGN KEY (`id_catrige`) REFERENCES `catriges`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `fk_history_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`);
```

Додаток Г

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ
БІОРЕСУРСІВІ
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій_____

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Онищук Олександр Олександрович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему

«Розробка веб-системи автоматизованого обліку витратних матеріалів та запчастин на основі технології контейнеризації____» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідноз чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

- [ ] інструменти генеративного ШІ не використовувалися.
- [X] інструменти ШІ (вказати назву: Gemini, Claude, Perplexity_____) були використані для:
  - [ ] перекладу іншомовних джерел;
  - [X] стилістичного редагування та перевірки граматики;
  - [X] допомоги у написанні/відлагодженні програмного коду;
  - [X] інше: пошук та структурування інформації .

Я розумію, що надання недостовірної інформації може призвести доскасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«18»Травень_2026р. _____ Онищук Олександр

(підпис) **(Ім'я та ПРІЗВИЩЕ)**