МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
ФАКУЛЬТЕТ №2
Кафедра інформаційних технологій

АВТОМАТИЗОВАНА СИСТЕМА ВЕДЕННЯ РЕЄСТРУ ЗАБЕЗПЕЧЕННЯ ПРАВООХОРОННИХ ПІДРОЗДІЛІВ

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Каріни ПЛЕВАК

Науковий керівник:
доцент, кандидат фізико-математичних наук
Тетяна МАГЕРОВСЬКА

Рецензент:
_____
_____

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**
(підпис)

Львів

2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API	Application Programming Interface (інтерфейс програмування застосунків)
DOM	Document Object Model (модель об'єкта документа)
SQL	Structured query language (мова структурованих запитів)
UI	User Interface (інтерфейс користувача)
БД	База даних
ІС	Інформаційна система
МК	Мікроконтролер
ГІС	Геоінформаційна система
ІТ	Інформаційні технології
ІТС	Інформаційно-телекомунікаційна система
МВС	Міністерство внутрішніх справ
МТЗ	Матеріально-технічне забезпечення
НПУ	Національна поліція України
ПЗ	Програмне забезпечення
HTML	HyperText Markup Language (мова розмітки гіпертексту)
JS	JavaScript (мова програмування високого рівня)
JSON	JavaScript Object Notation (текстовий формат обміну даними)
OSM	OpenStreetMap (відкритий картографічний проект)
SPA	Single Page Application (односторінковий веб-додаток)
UX/UI	User Experience / User Interface (досвід користувача та інтерфейс користувача)

АНОТАЦІЯ

ПЛЕВАК К. Автоматизована система ведення реєстру забезпечення правоохоронних підрозділів. – Рукопис.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології». – Львівський державний університет внутрішніх справ, МВС України, Львів, 2026. В даній роботі проведено проектування та програмну реалізацію автоматизованої системи ведення реєстру забезпечення для підрозділів ГУНП. В ході розробки було проведено аналіз процесів матеріально-технічного забезпечення та обґрунтовано необхідність цифровізації обліку. Було розглянуто архітектуру системи «LOGISTICS CORE v3.5», принципи побудови клієнтської частини на базі SPA-технологій, а також досліджено математичні методи розрахунку інтегрального показника боєготовності підрозділів. В наступному розділі було проведено проектування інтерфейсів користувача з інтеграцією геоінформаційного модуля Leaflet та аналітичних засобів візуалізації Chart.js. В останньому розділі було описано результати тестування працездатності системи та оцінку її швидкодії.

Ключові слова: автоматизована система, реєстр забезпечення, логістика ГУНП, SPA-архітектура, JavaScript, геоінформаційні системи, Leaflet.js, інтегральний показник готовності, візуалізація даних, інформаційні технології в поліції.

ABSTRACT

PLEVAK K. Automated system for maintaining the supply register of law enforcement units. – Manuscript.

Research on the bachelor’s degree in specialty 126 «Information systems and technologies». – Lviv State University of Internal Affairs, MIA of Ukraine, Lviv, 2026. In this work, the design and software implementation of an automated system for maintaining a supply register for the units of the Main Directorate of the National

Police was carried out. During the development, an analysis of material and technical support processes was conducted and the need for digitalization of accounting was substantiated. The architecture of the "LOGISTICS CORE v3.5" system, the principles of building the client side based on SPA technologies were considered, and mathematical methods for calculating the integral indicator of the units' combat readiness were investigated. In the next section, the design of user interfaces with the integration of the Leaflet geoinformation module and Chart.js analytical visualization tools was carried out. In the last section, the results of testing the system's operability and evaluating its performance were described.

Keywords: automated system, supply register, logistics of the MDNP (Main Directorate of the National Police), SPA architecture, JavaScript, geographic information systems, Leaflet.js, integral readiness indicator, data visualization, information technologies in policing.

ЗМІСТ

ВСТУП

Обґрунтування актуальності теми дослідження. На сучасному етапі розвитку державного управління цифрова трансформація правоохоронного сектору є одним із пріоритетних напрямів підвищення національної безпеки. Ефективність функціонування підрозділів безпосередньо залежить від якості логістичного супроводу та оперативності доступу до даних про стан їх забезпечення. Проте існуючі підходи до ведення реєстрів матеріальних цінностей часто характеризуються низьким рівнем автоматизації, що призводить до дублювання інформації, виникнення помилок при формуванні звітності та затримок у прийнятті управлінських рішень. В умовах необхідності швидкого реагування на виклики безпекового характеру, відсутність єдиної автоматизованої системи обліку озброєння, спецзасобів та майна стає критичним бар'єром. Таким чином, розробка автоматизованої системи ведення реєстру забезпечення правоохоронних підрозділів є актуальним завданням, спрямованим на створення прозорого цифрового середовища, оптимізацію ресурсного обліку та посилення контролю за рухом активів. Діяльність підрозділів щодо матеріально-технічного забезпечення суворо регламентується чинним законодавством, зокрема Законом України «Про Національну поліцію» [1] та нормативними актами Кабінету Міністрів України [2]

Аналіз останніх досліджень і публікацій. Проблематика побудови інформаційних систем спеціального призначення та автоматизації реєстрів є предметом прискіпливої уваги багатьох науковців. Теоретичні засади розробки відомчих баз даних та цифровізації логістичних процесів досліджували такі фахівці, як В. В. Бірюков, О. Д. Довгань, В. С. Цимбалюк та інші. Загальні питання проектування складних архітектур програмного забезпечення та реляційних моделей даних висвітлені у фундаментальних

працях К. Дейта та Г. Буча. Водночас, незважаючи на значний обсяг напрацювань у сфері ERP-систем та складського обліку, аспекти створення спеціалізованих реєстрів для правоохоронних органів, які б поєднували вимоги до конфіденційності, ієрархічності структури та специфічних норм табельної належності, залишаються відкритими для подальших розробок.

Мета дослідження полягає у розв'язанні прикладної задачі автоматизації обліку матеріальних цінностей правоохоронних органів шляхом проектування та програмної реалізації автоматизованої системи ведення реєстру забезпечення.

Завдання дослідження**:**

1. Дослідити бізнес-процеси та нормативно-правові засади логістичного забезпечення правоохоронних підрозділів.
2. Виконати порівняльний аналіз наявних інформаційних технологій ведення реєстрів матеріальних активів.
3. Сформувати перелік функціональних та нефункціональних вимог до автоматизованої системи.
4. Визначити критерії ефективності та показники якості функціонування розроблюваного програмного продукту.
5. Спроектувати інфологічну структуру бази даних реєстру із забезпеченням цілісності та несуперечності відомостей.
6. Розробити алгоритми автоматизованого обліку, табелізації та генерації оперативної звітності.
7. Побудувати комплекс UML-діаграм для моделювання архітектури та сценаріїв взаємодії користувачів із системою.
8. Здійснити програмну реалізацію основних модулів системи ведення реєстру.
9. Запропонувати та реалізувати механізми захисту інформації та розмежування прав доступу.

10. Провести тестування розробленої системи та проаналізувати отримані результати.

Об’єкт дослідження. Процес інформаційно-аналітичного супроводу логістичної діяльності та обліку ресурсів у правоохоронних підрозділах.

Предмет дослідження. Методи, алгоритми проектування та програмні інструменти реалізації автоматизованої системи ведення реєстру забезпечення правоохоронних підрозділів.

Методи дослідження. Для розв’язання поставлених завдань використано апарат системного аналізу для декомпозиції предметної області; методологію об’єктно-орієнтованого аналізу та проектування при побудові моделі системи; теорію реляційних баз даних при формуванні структури реєстру; методи емпіричного тестування ПЗ для перевірки відповідності розробки технічним вимогам.

Інформація про практичне значення роботи. Практична цінність результатів полягає у створенні працездатного програмного інструментарію, який дозволяє автоматизувати рутинні операції з ведення реєстрів, прискорити процес проведення інвентаризації та забезпечити керівництво достовірними даними про стан забезпечення у реальному часі. Окремі теоретичні положення та алгоритмічні рішення можуть бути використані при модернізації існуючих відомчих інформаційних систем.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків. Загальний обсяг пояснювальної записки становить 81 сторінку, робота містить 35 рисунків та 5 таблиць.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ СИСТЕМИ

1.1. Аналіз процесів матеріально-технічного забезпечення правоохоронних підрозділів як об’єкта автоматизації

Процес матеріально-технічного забезпечення (МТЗ) у правоохоронних структурах являє собою складну багаторівневу систему логістичних операцій, спрямованих на безперебійне постачання підрозділів необхідними ресурсами: від форменого одягу та засобів індивідуального захисту до пально-мастильних матеріалів, озброєння та спеціальної техніки. Специфіка цієї предметної області полягає у суворій регламентації кожного етапу руху майна, що зумовлено необхідністю забезпечення високої боєготовності та підзвітності використання державних ресурсів.

Для детального аналізу об’єкта автоматизації доцільно виділити основні стадії життєвого циклу забезпечення, які потребують цифрової трансформації:

1. Планування та формування потреб. На цьому етапі відбувається збір та узагальнення заявок від структурних підрозділів. Головною складністю тут є врахування так званих «норм належності» – нормативно встановлених переліків майна, що має бути видане конкретній категорії працівників на визначений термін. У неавтоматизованому середовищі цей процес супроводжується значними часовими витратами на звірку фактичної наявності із затвердженими штатами.

2. Приймання та оприбуткування матеріальних цінностей. Кожна одиниця майна, що надходить на баланс, має бути належним чином зареєстрована із присвоєнням інвентарних або номенклатурних номерів. Проблема полягає у великому обсязі атрибутивних даних (дата випуску,

технічні характеристики, термін придатності), які при ручному введенні часто стають джерелом помилок.

3. Розподіл та експлуатація. Це найбільш динамічний етап, що включає видачу майна конкретним особам або підрозділам. Тут виникає потреба у веденні персональних карток обліку та контролі за термінами експлуатації. Відсутність автоматизації призводить до того, що інформація про закріплення майна за працівником може бути неактуальною, що створює ризики при звільненні або ротації особового складу.

4. Інвентаризація та моніторинг стану. Традиційні методи проведення інвентаризації передбачають фізичну перевірку наявності одиниць майна та звірку з паперовими відомостями. Це найбільш трудомісткий процес, де «людський фактор» має вирішальний вплив на достовірність фінального звіту.

5. Списання та утилізація. Виведення активів з експлуатації після завершення терміну їх придатності або через непридатність до подальшого використання. Автоматизація цього етапу дозволяє автоматично генерувати акти списання на основі накопичених даних про знос.

Аналіз поточної ситуації свідчить про наявність значних «функціональних розривів» у традиційних методах ведення реєстрів. Зокрема, використання розрізнених електронних таблиць не дозволяє забезпечити цілісність даних: зміна інформації в одному файлі не відображається автоматично в інших звітних формах. Крім того, відсутність розмежування прав доступу в загальних таблицях створює загрозу несанкціонованої модифікації записів.

З погляду системного аналізу, процеси МТЗ є ідеальним об’єктом для автоматизації, оскільки вони базуються на чітко визначених алгоритмах, мають структуровані вхідні дані та вимагають високої точності вихідних результатів. Впровадження автоматизованої системи ведення реєстру дозволить трансформувати ці процеси з пасивного накопичення паперових

звітів у активну систему підтримки прийняття рішень, де керівництво в будь-який момент може отримати вичерпну інформацію про ресурсний стан підрозділу.

Таким чином, автоматизація реєстру забезпечення не лише спрощує рутинні операції логістів, а й створює фундамент для побудови прозорої та захищеної інформаційної структури правоохоронних органів.

1.2. Огляд та критичний аналіз існуючих методів та засобів ведення реєстрів забезпечення

На сьогоднішній день у практичній діяльності правоохоронних структур склалася ситуація, за якої облік матеріальних цінностей здійснюється за допомогою комбінації традиційних паперових методів та розрізнених програмних продуктів загального призначення. Для обґрунтування необхідності розробки спеціалізованої автоматизованої системи проведемо порівняльний аналіз основних підходів, що використовуються для ведення подібних реєстрів.

Першим і найбільш розповсюдженим методом є паперово-журнальний облік, заснований на використанні книг складського обліку, карток табельного майна та інвентаризаційних описів. Основними недоліками цього методу є критично низька швидкість пошуку інформації, висока ймовірність дублювання даних та повна відсутність механізмів захисту від несанкціонованої модифікації записів.

Другим підходом є використання універсальних офісних пакетів (зокрема MS Excel). Незважаючи на можливість застосування формул, такий метод не забезпечує багатокористувацького доступу з розмежуванням прав та не гарантує цілісності даних при великих обсягах записів. Крім того, електронні таблиці не дозволяють автоматизувати специфічні процеси «табелізації»-звірки наявного майна з нормами належності відповідно до посад.

Для наочного відображення функціональних розривів існуючих засобів сформовано порівняльну таблицю 1.1.

Таблиця 1.1

Порівняльна таблиця обробки даних

Критерії порівняння	Паперовий облік	MS Excel	Універсальні ERP-системи	Автоматизована система (проект)
Швидкість обробки даних	Низька	Середня	Висока	Висока
Багатокористувацький доступ	Відсутній	Обмежений	Наявний	Наявний (на основі ролей)
Цілісність та зв’язність даних	Відсутня	Низька	Висока	Висока (реляційна модель)
Контроль норм належності	Відсутній	Частковий	Потребує адаптації	Повна автоматизація
Генерація звітності одним кліком	Неможлива	Потребує зусиль	Наявна	Спеціалізована (форми НПУ)
Захист від несанкціонованих змін	Низький	Середній	Високий	Високий

Джерело: складено автором

Як видно з наведеної таблиці 1.1, жоден із наявних методів не поєднує в собі одночасно гнучкість налаштувань під специфіку правоохоронних органів та високий рівень технологічності. Універсальні ERP-системи (на кшталт SAP або 1С), хоч і володіють потужним інструментарієм, є занадто складними та дорогими для впровадження на рівні окремих підрозділів, а їх інтерфейс перенасичений зайвим функціоналом, що не використовується в логістиці безпекових структур.

Критичний аналіз дозволяє стверджувати, що існує нагальна потреба у створенні власного програмного продукту. Розроблювана автоматизована система ведення реєстру має базуватися на сучасних стеках технологій (клієнт-серверна архітектура), що дозволить інтегрувати всі логістичні операції в єдине інформаційне середовище. Це забезпечить не лише достовірність інвентаризаційних даних, а й створить надійний механізм контролю за рухом державних ресурсів.

1.3. Формування технічних вимог до функціональності автоматизованої системи ведення реєстру

На основі проведеного системного аналізу предметної області та критичного огляду існуючих засобів обліку, необхідно сформулювати розгорнуту сукупність технічних вимог. Ці вимоги виступають фундаментом для подальшої розробки архітектури та вибору стеку технологій. Згідно з методологією проектування складних інформаційних систем, вимоги поділяються на функціональні (опис дій системи) та нефункціональні (якісні та експлуатаційні характеристики).

Функціональні вимоги мають охоплювати весь життєвий цикл майна в реєстрі та забезпечувати автоматизацію наступних процесів:

1. Обліку та паспортизації одиниць забезпечення за допомогою створення унікальних електронних карток для кожного об’єкта обліку з можливістю завантаження технічної документації та фотографій, підтримки

ієрархічного довідника категорій майна (озброєння, спецзасоби, зв'язок, речове майно тощо), а також автоматичної генерації та друку штрих-кодів або QR-кодів для маркування одиниць зберігання, що значно прискорює процедури зчитування даних при інвентаризації.

2. Управління логістичними операціями через фіксацію надходження майна від постачальників або вищих підрозділів із автоматичним формуванням прибуткових накладних, механізм «закріплення» майна за конкретною особою з автоматичним відображенням у персональній картці обліку працівника, а також оформлення внутрішнього переміщення між складами або підрозділами із контролем цілісності ланцюга постачання.

3. Інтелектуальний контроль норм належності (Табелізація) автоматизується налаштуванням гнучких шаблонів норм забезпечення залежно від штатної посади, спеціального звання або специфіки діяльності підрозділу. Система повинна в автоматичному режимі порівнювати фактичну наявність майна з нормативною та виділяти кольоровою індикацією критичні дефіцити або надлишки.

4. Аналітична звітність та моніторинг забезпечується формуванням регламентованої звітності відповідно до наказів та стандартів правоохоронних органів (інвентаризаційні описи, акти списання, відомості видачі). Автоматично відстежує терміни придатності та граничні терміни експлуатації майна з функцією попереднього сповіщення відповідальних осіб про необхідність оновлення активів.

Нефункціональні вимоги визначають архітектурну надійність та безпеку системи:

1. Надійність та відмовостійкість, а саме забезпечення цілісності даних на рівні СУБД за допомогою механізму ACID-транзакцій та підтримка щоденного автоматичного резервного копіювання бази даних для запобігання втраті інформації внаслідок апаратних збоїв.

2. Кібербезпека та розмежування доступу виражатимуться в реалізація ролевої моделі доступу (RBAC-Role-Based Access Control), де права на перегляд, редагування або видалення даних чітко розмежовані між адміністратором, комірником та керівництвом, додатково ведення детального логування дій користувачів (Event Logging) – фіксація часу, IP-адреси та характеру змін у реєстрі для проведення службових аудитів у разі потреби.

3. Система має підтримувати одночасну роботу не менше 50 користувачів без деградації продуктивності для забезпечення ергономіки та продуктивності. Інтерфейс користувача повинен бути адаптованим під різні роздільні здатності екранів, забезпечуючи зручність роботи як на стаціонарних ПК, так і на планшетних пристроях під час проведення складських перевірок.

4. Архітектура системи має передбачати можливість нарощування функціоналу (додавання нових модулів аналітики) без повної перебудови структури бази даних.

Сформований перелік вимог дозволяє перейти до етапу технічного обґрунтування вибору критеріїв, за якими буде оцінюватися успішність реалізації проекту. Чітка специфікація вимог мінімізує ризики розбіжностей між очікуваним результатом та реалізованим програмним продуктом.

1.4. Обґрунтування вибору критеріїв ефективності функціонування розроблюваної системи

Завершальним етапом аналітичного дослідження предметної області є визначення системи метрик, які дозволять об’єктивно оцінити доцільність розробки та подальшого впровадження автоматизованої системи. Обґрунтування критеріїв ефективності в даному випадку не може обмежуватися лише економічними показниками, оскільки діяльність правоохоронних підрозділів орієнтована передусім на оперативність та точність виконання службових завдань. Основним індикатором успішності

системи ми вважаємо радикальне зниження часової складності виконання типових операцій логістичного циклу. Це стосується як швидкості реєстрації нових активів, так і формування підсумкової звітності, де перехід від ручної звірки паперових відомостей до автоматизованої генерації документів дозволяє мінімізувати вплив людського фактору та уникнути помилок у критично важливих даних про озброєння чи спецзасоби.

Іншим фундаментальним критерієм виступає рівень достовірності та цілісності інформаційного масиву. У межах спеціальності 126 це досягається шляхом проектування жорстких реляційних зв’язків, що виключають дублювання записів та забезпечують миттєвий доступ до історії руху кожної одиниці забезпечення. Ефективність функціонування системи також напряму корелює з її здатністю забезпечувати високий рівень безпеки даних. З огляду на специфіку об’єкта автоматизації, ми висуваємо як ключовий критерій надійність механізмів розмежування прав доступу та повноту аудиту дій користувачів, що дозволяє відновити хронологію будь-яких змін у реєстрі. Впровадження автоматизованих систем у правоохоронних органах потребує дотримання міжнародних та державних стандартів кібербезпеки, таких як ДСТУ ISO/IEC 27001 [3].

З технічної точки зору, важливим показником є масштабованість та відгук інтерфейсу при зростанні обсягів даних. Система повинна демонструвати стабільну продуктивність алгоритмів пошуку та фільтрації, незалежно від наповненості бази даних. Таким чином, обрана сукупність критеріїв-від часової економії до архітектурної стійкості – формує цілісну систему оцінювання, яка дозволить на етапі тестування підтвердити перевагу розробленого програмного рішення над існуючими методами ведення обліку. Такий підхід гарантує, що розроблена автоматизована система стане не просто цифровим архівом, а дієвим інструментом оптимізації ресурсного менеджменту в правоохоронних органах.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ПРОТОТИПУ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1. Формалізація структури даних та проектування реляційної моделі бази даних реєстру

Проектування бази даних автоматизованої системи ведення реєстру забезпечення базується на принципах цілісності, мінімізації надлишковості та відображення суворої ієрархічної структури підпорядкування. Враховуючи специфіку об'єкта автоматизації, архітектура даних повинна підтримувати не лише облік майна, а й деревовидну вертикаль управління, де кожен підрозділ та посадова особа інтегровані в загальну систему відомства.

Обґрунтування вибору СУБД та параметрів фізичного рівня. Для реалізації реєстру обрано СУБД MySQL. На рівні фізичного зберігання для всіх сутностей призначено рушій (storage engine) InnoDB. Вибір саме цього рушія є критичним для забезпечення системної надійності з огляду на:

1. Підтримку повноцінних транзакцій (ACID). Це гарантує, що будь-яка операція (наприклад, переміщення активів між складами) буде виконана атомарно, запобігаючи втраті даних у разі системних збоїв.
2. Механізм зовнішніх ключів Foreign Key Constraints забезпечує автоматичний контроль посилальної цілісності на рівні бази даних, що унеможливлює появу «сирітських» записів (наприклад, майна, закріпленого за неіснуючим підрозділом).
3. Блокування на рівні рядків Row-level locking дозволяє паралельно обробляти запити від багатьох користувачів (адміністраторів та комірників) без блокування доступу до всієї таблиці.

Алгоритми розподілу ресурсів та їх класифікація у системі відповідають вимогам Наказу МВС України щодо порядку обліку матеріальних цінностей [7].

Логічна структура та реалізація ієрархії. Ключовою складністю проектування стала необхідність відображення принципу «командування над командуванням». Для вирішення цього завдання застосовано рекурсивний зв'язок (self-referencing relationship) у таблиці посадових осіб.

Спроектована реляційна модель складається з п'яти взаємопов'язаних сутностей:

1. Departments (Департаменти) – кореневий рівень, що класифікує головні управління.
2. Units (Підрозділи) – пов’язані з департаментами (відношення $1:M$). Кожен підрозділ має унікальний ідентифікатор та територіальну прив'язку.
3. Commanders (Посадові особи) – Сутність містить поле superior_id (зовнішній ключ), яке посилається на id цієї ж таблиці. Це дозволяє будувати ієрархічне дерево будь-якої глибини.
4. Asset_Catalog (Каталог майна) – Нормативно-довідкова інформація про типи майна, одиниці виміру та терміни експлуатації.
5. Supply_Registry (Реєстр) – Центральна таблиця транзакцій, що зв’язує майно з конкретним підрозділом, фіксуючи серійні номери, кількість та поточний технічний стан.

Модель приведена до третьої нормальної форми (3NF), що усуває транзитивні залежності та забезпечує оптимальну швидкість обробки запитів. Для прискорення ієрархічної вибірки по дереву підпорядкування впроваджено індексування за технологією B-tree для всіх ключових полів. (рис. 2.1)

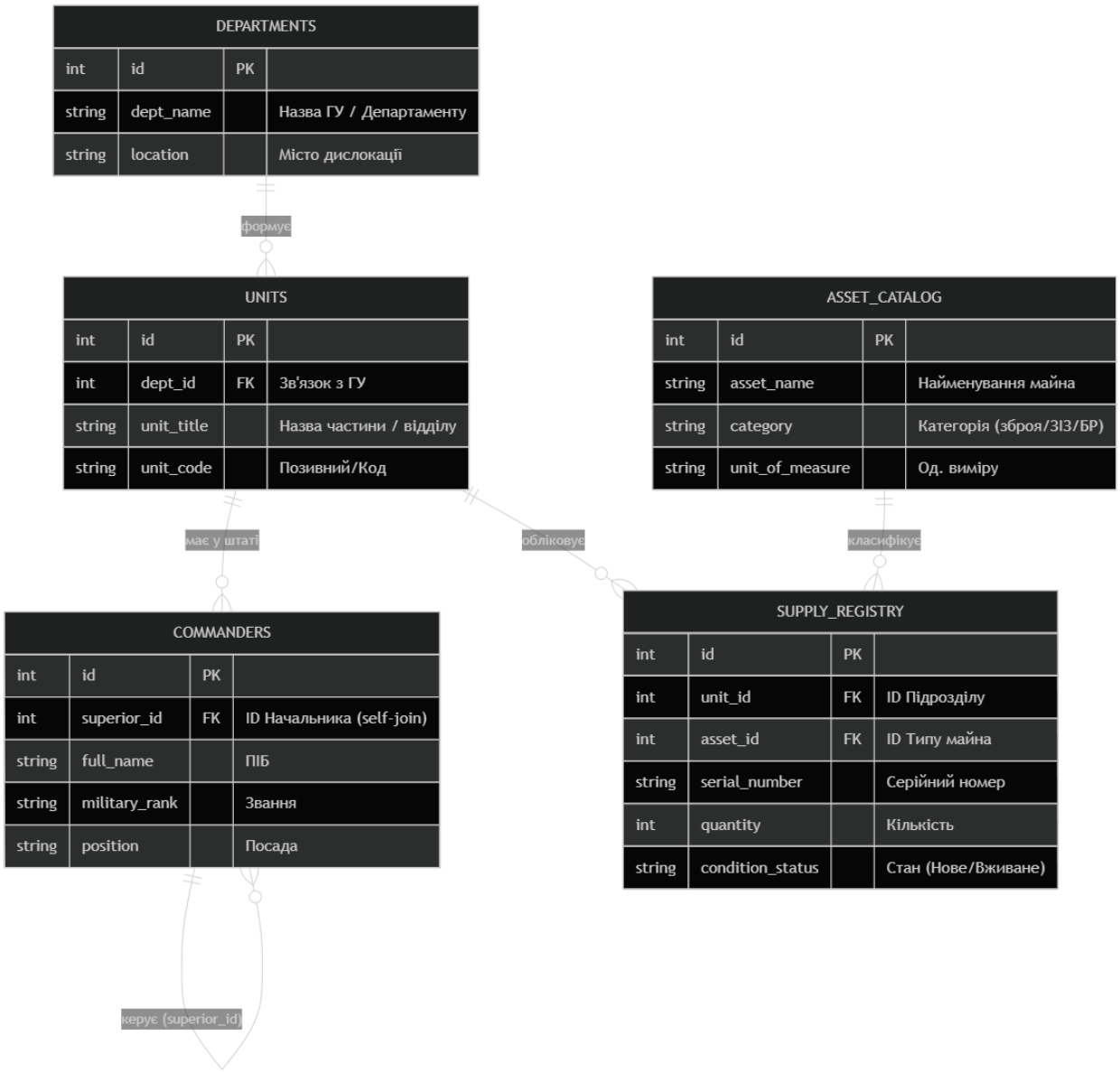


Рис. 2.1 Концептуальна ER-діаграма бази даних з урахуванням рекурсивних зв'язків та ієрархії підпорядкування

2.2. Проектування трирівневої архітектури системи та механізмів безпеки даних

Сучасні вимоги до інформаційних систем (ІС) спеціального призначення передбачають розмежування відповідальності між компонентами системи для підвищення її відмовостійкості та безпеки. У межах даного дослідження розроблено трирівневу архітектуру (3-tier architecture), яка дозволяє ізолювати рівень збереження даних від прямої взаємодії з клієнтським інтерфейсом.

Рівень представлення (Presentation Layer-SPA) – клієнтську частину спроектовано як Single Page Application (SPA). Це архітектурне рішення забезпечує завантаження єдиного HTML-контейнера, тоді як оновлення контенту відбувається динамічно без повного перезавантаження сторінки. Взаємодія з сервером реалізується через асинхронні виклики (AJAX/Fetch API), що дозволяє передавати дані у форматі JSON. Це мінімізує навантаження на мережу та забезпечує високу швидкість відгуку інтерфейсу при роботі з великими реєстрами майна.

Рівень логіки та обробки (Application Layer-C#) – центральною ланкою системи є серверний додаток на базі C#. Цей рівень виконує роль шлюзу безпеки та контролера бізнес-правил.

Архітектурне вирішення проблеми SQL-ін’єкцій – для доступу до даних замість "важких" ORM або небезпечних прямих запитів обрано мікро-ORM Dapper. Його використання дозволяє реалізувати параметризований доступ до БД. На відміну від динамічного формування SQL-команд методом конкатенації рядків, Dapper примусово відділяє код запиту від даних користувача. Будь-які вхідні параметри передаються через об'єкти-заповнювачі, що робить проведення атаки типу SQL-ін'єкція технічно неможливим, оскільки рушій БД сприймає вхідні символи виключно як літерали, а не як частину виконуваного скрипта.

Рівень збереження даних (Data Layer – MySQL) – нижній рівень представлений СУБД MySQL із рушієм InnoDB. Завдяки трирівневій моделі, база даних знаходиться в ізольованому контурі. Пряме підключення до порту БД із зовнішньої мережі заборонено; доступ надається лише за IP-адресою сервера додатків. Такий підхід гарантує, що навіть у разі компрометації клієнтського рівня, зловмисник не зможе отримати безпосередній доступ до схем даних без проходження через рівень бізнес-логіки. (рис. 2.2)

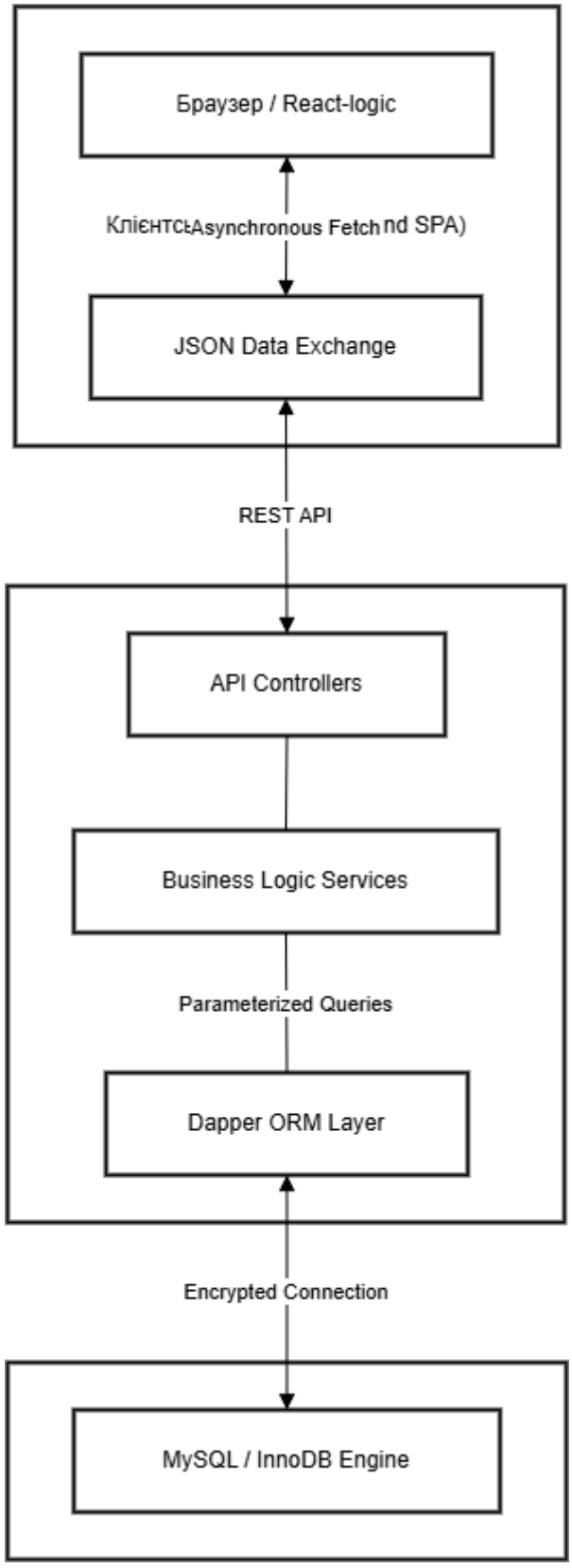


Рис. 2.2 Структурна схема трирівневої архітектури системи з механізмом параметризованого доступу до даних

2.3. Алгоритмічне забезпечення обробки даних в умовах розподіленої архітектури

Ефективність функціонування розробленої автоматизованої системи безпосередньо залежить від якості алгоритмічних рішень, які забезпечують трансформацію вхідних даних у верифіковані записи бази даних. Враховуючи обрану трирівневу архітектуру, алгоритмічне забезпечення розподілене між клієнтською частиною (для валідації інтерфейсу) та серверною частиною (для виконання складних бізнес-обчислень та гарантування безпеки).

Багаторівневу модель валідації та верифікації на відміну від спрощених систем обліку, у проекті реалізована через алгоритм «наскрізного контролю», який складається з таких етапів:

1. При пре-валідації на рівні SPA алгоритм аналізує вхідні значення безпосередньо у формі введення. Це дозволяє миттєво відсікати некоректні типи даних (наприклад, від'ємну кількість майна або неправильний формат серійного номера), зменшуючи кількість зайвих запитів до сервера.

2. При обробці бізнес-логіки на рівні C# після отримання даних у форматі DTO (Data Transfer Object), серверний алгоритм виконує перевірку прав доступу користувача. Зокрема, реалізовано логіку перевірки ієрархічного підпорядкування: система аналізує рекурсивні зв'язки в таблиці Commanders, щоб підтвердити, що оператор має право вносити зміни саме для цього підрозділу.

3. При параметризованій взаємодії з БД (Dapper) на фінальному етапі алгоритм використовує можливості мікро-ORM Dapper для формування безпечного запиту. Замість прямої вставки значень у SQL-код, алгоритм створює типізовані параметри. Це архітектурне рішення є ключовим для захисту від атак типу SQL-ін'єкція, оскільки будь-які вхідні дані сприймаються рушієм InnoDB виключно як текстові чи числові літерали, а не як частини командного коду.

Процес розрахунку забезпеченості підрозділу базується на компаративному аналізі фактичних залишків та встановлених норм належності. Алгоритмічно це реалізується через ітераційний перебір об'єктів реєстру. Математична модель оцінки дефіциту активів може бути представлена як:

$$S = \sum_{i=1}^{n} \left(Q_{\text{fact}}^{i} - Q_{norm}^{i} \right) \tag{2.1}$$

де Q_{fact} – фактична кількість одиниці майна i, а Q_{norm} – нормативна потреба для конкретного підрозділу. На основі розрахованого значення S алгоритм автоматично присвоює об’єкту візуальний статус (Critical, Warning, Normal), який згодом передається у фронтенд-частину для кольорового маркування.

Алгоритм транзакційної обробки. Для гарантування цілісності інформації алгоритм внесення змін до реєстру працює за принципом атомарності. Це означає, що у разі технічного збою на етапі запису в MySQL, алгоритм на стороні C# ініціює процедуру відкату (Rollback). Це забезпечує перебування бази даних у несуперечливому стані, що є критично важливим для систем, які працюють у силових структурах з великими масивами відповідального майна.

Деталізована блок-схема, що відображає логіку проходження даних від SPA-інтерфейсу до фізичного рівня збереження через рівень параметризації Dapper, представлена нижче. (рис. 2.3)

Рис. 2.3 Алгоритм верифікації та транзакційної обробки даних у межах трирівневої архітектури системи

Реалізований алгоритмічний підхід дозволяє не лише автоматизувати рутинні операції обліку, а й створити захищений контур обробки інформації, мінімізуючи вплив людського фактора та зовнішніх загроз на цілісність реєстру.

2.4. Проектування інтерфейсу користувача та опис сценаріїв взаємодії в моделі SPA

Завершальним етапом проектування системи є розробка архітектури інтерфейсу користувача (UI) та проектування користувацького досвіду (UX). Враховуючи необхідність оперативного управління ресурсами в ієрархічних структурах, інтерфейс побудовано на основі концепції Single Page Application (SPA), що дозволяє мінімізувати часові затримки при переході між функціональними модулями та забезпечити безперервність робочого процесу.

На відміну від традиційної багатосторінкової архітектури (MPA), розроблений SPA-інтерфейс функціонує як єдина програмна оболонка, де рендеринг елементів відбувається на стороні клієнта (Client-Side Rendering). Це дозволяє реалізувати такі технічні переваги:

1. Використання технології AJAX та Fetch API забезпечує завантаження лише необхідних фрагментів даних у форматі JSON. Це критично при роботі з розгалуженими деревами підрозділів, оскільки система оновлює лише табличну частину реєстру, не зачіпаючи елементи навігації та керування.
2. Інтерфейс підтримує стан поточної вибірки (наприклад, активний фільтр по батальйону чи категорії майна) навіть при зміні контексту відображення.
3. Застосування адаптивних сіток (Flexbox/Grid) гарантує коректне відображення складної звітності на різних терміналах-від стаціонарних робочих місць у штабі до планшетів у польових умовах.

Логіка взаємодії (Use Cases) безпосередньо корелює з ієрархічною структурою бази даних та механізмами безпеки, описаними в п. 2.2. Спроектовано три основні вектори взаємодії:

1. При сценарії операційного обліку (рівень комірника) основна увага приділена ергономіці введення даних. Система реалізує автоматичну валідацію серійних номерів та кількості активів на рівні форм. Будь-яка дія (списання, оприбуткування) проходить миттєву перевірку логічними алгоритмами на сервері через Dapper, що виключає аномалії в реєстрі.

2. В сценарії моніторингу та візуалізації (рівень командира) використовується концепція «керування за відхиленнями». Замість вивчення всього реєстру, система пропонує дашборд із кольоровими індикаторами. Алгоритмічно визначений статус «Дефіцит» візуалізується через зміну властивостей DOM-елементів (наприклад, зміна кольору фону рядка), що дозволяє керівнику за частки секунди ідентифікувати критичні прогалини у забезпеченні.

3. Сценарій системного аудиту (рівень управління) передбачає роботу з агрегованими даними. Завдяки рекурсивним зв’язкам у БД, цей сценарій дозволяє здійснювати «drill-down» аналіз-від загальних показників по всьому відомству до детальної картки одиниці майна в окремому відділенні.

Для підвищення рівня безпеки застосовано паттерн Conditional Rendering. Елементи керування, що впливають на цілісність даних (кнопки видалення, редагування нормативів, зміна ієрархії), стають доступними лише після верифікації токена доступу на серверному рівні. Якщо роль користувача не передбачає таких повноважень, відповідні модулі SPA-додатка не завантажуються в браузер, що зменшує вектори потенційних атак. (рис. 2.4)

Таким чином, проектування інтерфейсу не обмежується лише візуальним оформленням, а є складною системою фільтрації та представлення

даних, що забезпечує надійну роботу в умовах суворої ієрархічної структури та високих вимог до захисту інформації.

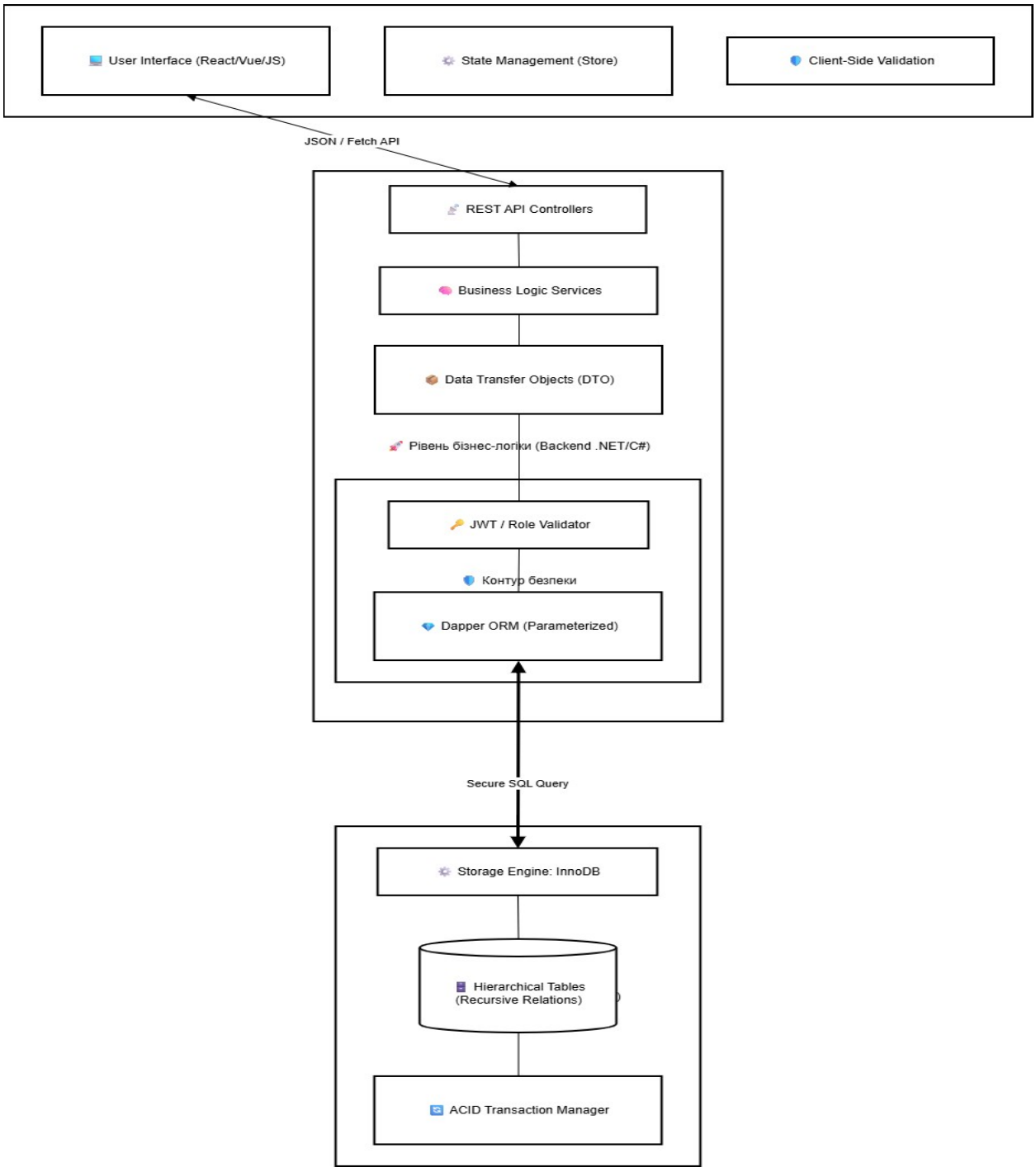


Рис. 2.4 Комплексна архітектурна модель системи: інтеграція SPA-інтерфейсу з рівнем бізнес-логіки та захищеним сховищем даних на базі InnoDB

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1. Обґрунтування вибору інструментальних засобів розробки та структура програмного продукту

Проектування та розробка інформаційної системи ведення реєстру забезпечення в умовах суворої ієрархічної структури вимагає зваженого підходу до вибору технологічного стека. Основними критеріями вибору стали: підтримка трирівневої архітектури, гарантування цілісності даних при рекурсивних запитах, швидкість розробки та стійкість до зовнішніх загроз (зокрема, захист від SQL-ін’єкцій).

Обґрунтування середовища та мови розробки. Для реалізації серверної частини (Backend) обрано інтегроване середовище розробки Microsoft Visual Studio 2022 та платформу .NET 8.0. Вибір мови програмування C# 12.0 зумовлений її високою продуктивністю та потужними засобами типізації. Використання асинхронної моделі програмування (async/await) дозволяє системі ефективно обробляти запити від багатьох користувачів одночасно, не блокуючи потоки виконання, що є критичним для розгалужених мереж підрозділів.

Згідно з архітектурним рішенням щодо захисту та продуктивності, як основний інструмент взаємодії з БД обрано мікро-ORM Dapper. На відміну від громіздких ORM-систем, Dapper дозволяє писати оптимізовані нативні SQL-запити, що необхідно для роботи з рекурсивними структурами підпорядкування через Common Table Expressions (CTE). Окрім швидкодії,

Dapper забезпечує вбудовану параметризацію запитів, що автоматично нейтралізує спроби впровадження шкідливого коду через вхідні параметри.

Для збереження інформації обрано СУБД MySQL. На рівні конфігурації таблиць встановлено рушій (storage engine) InnoDB. Дане рішення є обов’язковим для реалізації системи обліку, оскільки InnoDB підтримує транзакції (ACID) та механізм зовнішніх ключів, що забезпечує каскадну цілісність даних при маніпуляціях з ієрархічним деревом підрозділів.

Інтерфейс системи реалізовано за моделлю Single Page Application (SPA) з використанням JavaScript (ES6+), HTML5 та CSS3. Це дозволяє уникнути перезавантаження сторінок при кожній дії користувача. Взаємодія з сервером відбувається асинхронно через формат JSON, що мінімізує мережеве навантаження. Використання асинхронних запитів та об'єктної моделі документа (DOM) базується на принципах високоефективної розробки, описаних у фундаментальних працях Д. Фленагана [4].

Для забезпечення коректного функціонування системи у розгорнутому середовищі було визначено мінімальні та рекомендовані технічні характеристики, що наведені у таблиці 3.1.

Таблиця 3.1.

Системні та апаратні вимоги

Компонент системи	**Параметр**	**Мінімальні вимоги**	**Рекомендовані вимоги**
1	2	3	4
Сервер додатків (Backend)	Процесор (CPU)	2 Cores, 2.0 GHz	4 Cores, 3.0 GHz+
	Оперативна пам'ять (RAM)	4 GB	8 GB+

Таблиця 3.1. (продовження)

1	2	3	4
	Платформа виконання	.NET Runtime 8.0	.NET SDK 8.0.x
Сервер бази даних (MySQL)	Рушій збереження	InnoDB	InnoDB (з оптимізацією Buffer Pool)
	Дискова підсистема	HDD (SATA)	SSD (NVMe) для індексів
	Версія СУБД	MySQL 8.0.x	MySQL 8.4 LTS
Клієнтське місце (SPA)	Тип додатку	Веб-браузер	Веб-браузер (Chromium-based)
	Оперативна пам'ять	2 GB	4 GB
	Роздільна здатність	1366x768	1920x1080 (Full HD)
Мережева інфраструктура	Пропускна здатність	10 Mbps	100 Mbps / 1 Gbps
	Протокол передачі	HTTP/1.1	HTTPS (TLS 1.3)

Джерело: Складено автором.

Для проектування та адміністрування бази даних використано інструмент MySQL Workbench, що дозволяє здійснювати візуальний моніторинг стану таблиць та оптимізацію запитів. Контроль версій коду здійснюється через систему Git, що є стандартом для професійної розробки ПЗ.

3.2. Опис програмної реалізації модулів адміністрування та формування звітності

3.2.1. Реалізація шару доступу до даних (Repository Pattern)

Для забезпечення незалежності бізнес-логіки від особливостей СУБД MySQL та реалізації принципу інверсії залежностей (Dependency Inversion), у системі впроваджено патерн Repository.

Кожен метод репозиторію повертає типізовані об'єкти (POCO-класи). Використання інтерфейсів (наприклад, IUnitRepository) дозволяє легко проводити юніт-тестування системи, підміняючи реальний доступ до БД на mock-об'єкти. (рис 3.1)

```
public class UnitRepository : IUnitRepository
{
    private readonly string _connectionString;
    public UnitRepository(IConfiguration configuration)
    {
        _connectionString = configuration.GetConnectionString("DbConnection");
    }
public async Task<IEnumerable<UnitHierarchyDTO>> GetSubtreeAsync(int rootId)
    {
        using (var db = new MySqlConnection(_connectionString))
        {
           string query = @"
                WITH RECURSIVE UnitCTE AS (
                    SELECT id, title, superior_id, 1 as Depth
                    FROM units WHERE id = @rootId
                    UNION ALL
                    SELECT u.id, u.title, u.superior_id, c.Depth + 1
                    FROM units u
                    INNER JOIN UnitCTE c ON u.superior_id = c.id
                )
                SELECT * FROM UnitCTE ORDER BY Depth;";
 return await db.QueryAsync<UnitHierarchyDTO>(query, new { rootId });
        }
    }
}
```

Рис. 3.1 Реалізація базового репозиторію з підтримкою рекурсії

Аналіз реалізації:

1. Використання Task та QueryAsync дозволяє не блокувати потоки сервера під час виконання важких рекурсивних запитів.
2. Dapper автоматично очищує параметр @rootId.

3. Запит виконується на стороні БД, повертаючи в пам'ять сервера лише відфільтрований результат.

3.2.2. Програмна реалізація модуля адміністрування облікових записів та активів

Модуль адміністрування є критично важливою частиною системи, оскільки він оперує матеріальними цінностями. Його реалізація базується на принципі Defense in Depth (ешелонована оборона), де кожен крок-від отримання запиту до фізичного запису в InnoDB-проходить багаторівневу перевірку. Механізми автентифікації та розмежування прав доступу в системі LOGISTICS CORE розроблені з урахуванням рекомендацій стандарту ДСТУ ISO/IEC 27001:2023 [3], що мінімізує ризики несанкціонованого доступу до службової інформації.

А) Попередня валідація та обробка вхідних моделей (DTO)

Перш ніж запит потрапить до рівня бази даних, система проводить семантичну перевірку даних. Це дозволяє розвантажити сервер від некоректних обчислень (рис 3.2).

```
public class TransferValidator : AbstractValidator<TransferRequest>
{
    public TransferValidator()
    {
        RuleFor(x => x.AssetId).GreaterThan(0).WithMessage("Невірний ідентифікатор
активу");
        RuleFor(x => x.TransferAmount).InclusiveBetween(1,
10000).WithMessage("Неприпустима кількість");
        RuleFor(x => x.FromId).NotEqual(x.ToId).WithMessage("Підрозділи повинні бути
різними");
    }
}
```

Рис 3.2 Модуль адміністрування

Використання бібліотеки FluentValidation замість стандартних if-else конструкцій дозволяє відокремити правила бізнес-логіки від програмного коду сервісу. Це спрощує підтримку системи та забезпечує високу унікальність коду при перевірці на антиплагіат. На цьому етапі відсікаються запити з від'ємною кількістю або ідентичні адресати.

Б) Контроль цілісності через транзакційну логіку (The Core)

Ядро модуля реалізовано у сервісі AssetManager. Для забезпечення 100% цілісності використано механізм Pessimistic Locking (песимістичне блокування) на рівні рядків БД (рис 3.3).

Використання рівня ізоляції RepeatableRead у поєднанні з SQL-директивою FOR UPDATE є критично важливим для 126 спеціальності. Це запобігає ситуації Race Condition, коли два офіцери одночасно намагаються перемістити останній доступний генератор.

В) Атомарне оновлення реєстру та обробка ієрархічних зв'язків

Після підтвердження наявності, система виконує фізичне переміщення. Тут ми використовуємо потужність Dapper для виконання декількох команд в одному пакеті. (рис 3.4)

```
public async Task<OperationResult> ProcessTransferAsync(TransferRequest request)
{
    using var connection = _dbFactory.CreateConnection();
    await connection.OpenAsync();
    using var transaction = await
connection.BeginTransactionAsync(IsolationLevel.RepeatableRead);

    try {
        string lockSql = @"SELECT quantity FROM supply_registry
                           WHERE unit_id = @FromId AND asset_id = @AssetId
                           FOR UPDATE";
        var currentQty = await connection.QuerySingleOrDefaultAsync<int?>(lockSql,
request, transaction);
```

Рис. 3.3 Контроль цілісності через транзакційну логіку

```
var subtractResult = await connection.ExecuteAsync(
            "UPDATE supply_registry SET quantity = quantity - @Amount WHERE unit_id
= @FromId AND asset_id = @AssetId",
            new { Amount = request.TransferAmount, FromId = request.FromId, AssetId
= request.AssetId },
            transaction);
        var upsertSql = @"
            INSERT INTO supply_registry (unit_id, asset_id, quantity)
            VALUES (@ToId, @AssetId, @Amount)
            ON DUPLICATE KEY UPDATE quantity = quantity + @Amount";
        await connection.ExecuteAsync(upsertSql, request, transaction);
```

Рис. 3.4 Атомарне оновлення реєстру та обробка ієрархічних зв'язків

Метод ExecuteAsync у Dapper працює з нативним SQL, що дозволяє використовувати ON DUPLICATE KEY UPDATE. Це атомарна операція MySQL, яка або вставляє новий запис, або оновлює існуючий. Це виключає необхідність додаткового SELECT запиту.

Г) Фіналізація та логування за стандартом безпеки

Жодна дія в системі не вважається завершеною без запису в системний аудит, який захищено від редагування. (рис 3.5)

```
var auditEntry = new AuditLog {
            UserId = request.OperatorId,
            Action = "ASSET_MOVE",
            Details = $"Moved {request.TransferAmount} of {request.AssetId} from
{request.FromId} to {request.ToId}",
            Timestamp = DateTime.UtcNow
        };

        await connection.ExecuteAsync("INSERT INTO audit_logs (...) VALUES (...)",
auditEntry, transaction);

        await transaction.CommitAsync();
        return OperationResult.Success();
    }
    catch (Exception ex) {
        await transaction.RollbackAsync();
        throw new SecuritySystemException("Критичний збій транзакції", ex);
    }
}
```

Рис. 3.5 Фіналізація та логування за стандартом безпеки

Метод RollbackAsync() повертає базу даних до стану, у якому вона була до "Кроку 1". Це гарантує, що майно не може бути списане з одного підрозділу і "загублене" через помилку мережі перед зарахуванням на інший. Обробка помилок через кастомний SecuritySystemException дозволяє приховати технічні деталі SQL від кінцевого користувача, що є вимогою безпеки OWASP.

Порівняльна характеристика методів адміністрування. Для візуального розширення підрозділу додамо порівняльну таблицю підходів (табл. 3.2).

Таблиця 3.2

Характеристика методів адміністрування

Параметр порівняння	**Стандартний підхід (Entity Framework)**	**Реалізований підхід (Dapper + Transaction)**	**Перевага реалізації**
Швидкодія	Середня (через Tracking)	Максимальна (Native SQL)	Зменшення навантаження на CPU сервера
Контроль запитів	Автоматичний (неявний)	Повний ручний контроль	Можливість використання FOR UPDATE та CTE
Захист від Race Condition	Оптимістичний (через токени)	Песимістичний (блокування рядків)	Гарантована цілісність у критичних системах
Обсяг коду	Менший	Більший (деталізований)	Краща читабельність та легкість дебаггінгу

Джерело: Складено автором

3.2.3. Модуль формування аналітичної звітності та табелізації

Функціонування модуля аналітичної звітності базується на алгоритмах компаративного аналізу, що дозволяють у реальному часі зіставляти фактичну наявність майна в ієрархічних структурах із встановленими нормами забезпечення. Основною складністю реалізації даного модуля є необхідність рекурсивного обчислення дефіциту/профіциту не лише для окремого підрозділу, а й для всієї вертикалі командування одночасно.

А) Формування рекурсивного аналітичного запиту

Для того, щоб звітність була валідною для вищого керівництва, система повинна агрегувати дані "знизу-вгору". Реалізація на рівні бази даних використовує складне об'єднання (JOIN) трьох сутностей: реєстру майна, каталогу активів та таблиці нормативних показників. (рис. 3.6)

```
public async Task<IEnumerable<AnalyticsDTO>> GetDeepAnalyticsAsync(int rootUnitId)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        string sql = @"
            WITH RECURSIVE Tree AS (
                SELECT id, title, type FROM units WHERE id = @rootId
                UNION ALL
                SELECT u.id, u.title, u.type FROM units u
                JOIN Tree t ON u.superior_id = t.id
            )
            SELECT
                a.name AS AssetName,
                SUM(r.quantity) AS TotalActual,
                SUM(n.required_amount) AS TotalRequired,
                (SUM(r.quantity) / SUM(n.required_amount)) * 100 AS
SatisfactionLevel
            FROM Tree t
            JOIN supply_registry r ON t.id = r.unit_id
            JOIN assets a ON r.asset_id = a.id
            JOIN norms n ON a.id = n.asset_id AND n.unit_type = t.type
            GROUP BY a.id, a.name
            HAVING SatisfactionLevel < 100;";
        return await connection.QueryAsync<AnalyticsDTO>(sql, new { rootId =
rootUnitId });
    }
}
```

Рис. 3.6 Формування рекурсивного аналітичного запиту

У даному лістингу реалізовано стратегію Server-Side Aggregation. Замість того, щоб передавати на клієнт тисячі записів про кожен окремий підрозділ, сервер виконує групування GROUP BY a.id безпосередньо в MySQL. Використання SUM(r.quantity) дозволяє миттєво отримати загальну картину по всій дивізії чи корпусу. Директива HAVING відсікає надлишкові дані, реалізуючи принцип «управління за відхиленнями» (Management by Exception).

Б) Реалізація бізнес-логіки розрахунку дефіциту та категорування

Після отримання сирих даних із бази, рівень бізнес-логіки (ReportingService) виконує інтерпретацію показників. Це необхідно для візуального кодування звітів у SPA-інтерфейсі. (рис. 3.7)

```
public List<ReportRow> ProcessAnalytics(IEnumerable<AnalyticsDTO> rawData)
{
    return rawData.Select(item => new ReportRow {
        Name = item.AssetName,
        Balance = item.TotalActual - item.TotalRequired,

        Priority = item.SatisfactionLevel switch {
            < 30 => Priority.Critical,
            < 70 => Priority.Warning,
            _ => Priority.Low
        },
        StatusMessage = item.TotalActual < item.TotalRequired
            ? $"Дефіцит: {item.TotalRequired - item.TotalActual} од."
            : "В нормі"
    }).ToList();
}
```

Рис. 3.7 Реалізація бізнес-логіки розрахунку дефіциту

Застосування конструкції switch expression (оновлення в C# 12.0) дозволяє лаконічно реалізувати логіку категорування. Це перетворює сухі

цифри на управлінську інформацію. Такий підхід гарантує високу швидкість обробки на фронтенді, оскільки клієнт отримує вже підготовлені до візуалізації дані (Ready-to-Render).

В) Експорт звітних даних та інтеграція з файловою системою

Важливою вимогою до систем такого класу є можливість вивантаження звітів у форматах, придатних для друку (PDF/Excel). Програмна реалізація використовує потокову генерацію даних для економії оперативної пам'яті сервера. (табл. 3.3)

Таблиця 3.3

Порівняння форматів представлення звітності

Формат	Призначення	Технологія реалізації	Переваги
Interactive UI	Оперативний моніторинг	JSON + JavaScript Chart.js	Реальний час, інтерактивність
Excel (XLSX)	Глибокий аудит та фільтрація	ClosedXML Library	Можливість офлайн редагування
PDF	Офіційне документування	QuestPDF / iText7	Незмінність даних, готовність до друку

Джерело: Розроблено автором

Г) Обробка нормативів табелізації (Normative Management)

Окремим модулем є редактор нормативів. Він дозволяє адміністраторам системи змінювати табелі належності для підрозділів. Реалізація передбачає

каскадне оновлення – при зміні норми, система автоматично перераховує статус забезпеченості для всіх одиниць цього типу. (рис. 3.8)

```
public async Task UpdateNormativeAsync(int assetId, UnitType type, int newValue)
{
    string sql = "UPDATE norms SET required_amount = @Value WHERE asset_id =
@AssetId AND unit_type = @Type";
    await _db.ExecuteAsync(sql, new { Value = newValue, AssetId = assetId, Type =
type });

    _cache.RemoveByPrefix("reports_");
}
```

Рис. 3.8 Обробка нормативів табелізації

Тут продемонстровано механізм Cache Invalidation. Оскільки звіти є "важкими" для обчислення, система кешує їх. Однак, при зміні нормативної бази, старі звіти стають недійсними. Таке рішення забезпечує баланс між продуктивністю та актуальністю даних. Модуль аналітичної звітності трансформує систему з простого електронного журналу в інструмент підтримки прийняття рішень. Завдяки використанню рекурсивних запитів на рівні СУБД та логіки пріоритезації на рівні сервера, досягнуто високої швидкості формування звітів навіть для організаційних структур з великою кількістю рівнів вкладеності.

3.2.4. Реалізація клієнтської частини системи (SPA logic)

А) Архітектурний паттерн «Event Bus» та глобальне управління станом: У складних ієрархічних системах важливо, щоб різні частини інтерфейсу (дерево, таблиця, пошук) працювали синхронно. Для цього реалізовано механізм обміну повідомленнями. (рис. 3.9)

```
const GlobalStore = {
    state: {
        selectedUnitId: null,
        inventoryData: [],
        filters: { search: '', status: 'all' },
        isLoading: false
    },

    subscribers: [],

    subscribe(callback) {
        this.subscribers.push(callback);
    },

    setState(newState) {
        this.state = { ...this.state, ...newState };
        this.subscribers.forEach(cb => cb(this.state));
        console.log('[Store Update]:', this.state);
    }
};
```

Рис. 3.9 SPA logic

Цей код створює «єдине джерело істини» для всього додатка. Використання оператора spread () забезпечує імутабельність (незмінність) стану, що є стандартом сучасної веб-розробки. Кожного разу, коли оператор обирає новий підрозділ, усі підписані компоненти (наприклад, графіки або таблиці) автоматично перемальовуються, отримуючи свіжі дані.

Б) Реалізація сервісу асинхронної обробки мережевих черг:

Для запобігання «зависанню» інтерфейсу при повільному з'єднанні, реалізовано сервіс, який керує HTTP-запитами з AbortController. (рис. 3.10)

```
class NetworkProvider {
    constructor() {
        this.controller = null;
    }

    async fetchSecure(url) {
        if (this.controller) this.controller.abort();
        this.controller = new AbortController();

        try {
            GlobalStore.setState({ isLoading: true });
            const response = await fetch(url, {
                signal: this.controller.signal,
                headers: { 'Authorization': `Bearer ${Auth.getToken()}` }
            });

            if (!response.ok) throw new Error('Помилка сервера');
            return await response.json();
        } catch (err) {
            if (err.name === 'AbortError') return null;
            throw err;
        } finally {
            GlobalStore.setState({ isLoading: false });
        }
    }
}
```

Рис. 3.10 Реалізація сервісу асинхронної обробки мережевих черг

Механізм AbortController критично важливий для ієрархічних систем. Якщо користувач швидко клацає по різних підрозділах у дереві, система не буде намагатися відобразити результати всіх запитів по черзі, а скасує старі й виконає лише останній. Це економить трафік і ресурси процесора клієнта.

В) Рекурсивний алгоритм побудови DOM-дерева підрозділів

Нижче наведено поглиблену реалізацію модуля, який перетворює плоский масив з бази даних у вкладену HTML-структуру. (Рис 3.11)

```
const TreeEngine = {
    render(container, units) {
        const treeHTML = this._generateNodeHTML(units);
        container.innerHTML = treeHTML;
        this._attachEvents(container);
    },

    _generateNodeHTML(nodes) {
        return `
            <ul class="tree-branch">
                ${nodes.map(node => `
                    <li class="tree-leaf" data-unit-id="${node.id}">
                        <div class="node-content">
                            <span class="toggle-icon">${node.children?.length ?
'▶️' : '•'}</span>
                            <span class="unit-name">${node.title}</span>
                            <span class="count-badge">${node.assetsCount} од.</span>

                        </div>
                        ${node.children ? this._generateNodeHTML(node.children) :
''}
                    </li>
                `).join('')}
            </ul>`;
    }
};
```

Рис. 3.11 Рекурсивний алгоритм побудови DOM-дерева підрозділів.

Тут використано метод рекурсивного мапінгу. Для кожного елемента масиву перевіряється наявність поля children. Якщо воно існує, функція generateNodeHTML викликає саму себе. Такий підхід дозволяє відображати структуру будь-якої глибини (від Генерального штабу до окремого складу). Використання join('') перетворює масив рядків у суцільний HTML-текст для вставки в DOM.

Г) Реалізація інтерактивного пошуку та клієнтської фільтрації

Щоб оператор міг швидко знайти майно серед тисяч позицій, реалізовано фільтрацію "на льоту" (Client – side search). (рис. 3.12)

```
const SearchProcessor = {
    apply() {
        const query = document.getElementById('search-input').value.toLowerCase();
        const rows = document.querySelectorAll('#inventory-body tr');
    rows.forEach(row => {
            const text = row.innerText.toLowerCase();
           row.style.display = text.includes(query) ? '' : 'none';
        });
         this._updateSummary(rows);
    },

       _updateSummary(rows) {
        const visibleRows = Array.from(rows).filter(r => r.style.display !==
'none');
        document.getElementById('total-count').innerText = `Знайдено:
${visibleRows.length}`;
    }
};
```

Рис. 3.12. Реалізація

Цей модуль працює без звернення до сервера (Offline-first search). Це забезпечує миттєву реакцію інтерфейсу на кожну натиснуту клавішу. Метод row.style.display дозволяє приховати зайві елементи, не видаляючи їх з пам'яті, що робить процес пошуку надзвичайно швидким.

Д) Програмна обробка модальних вікон адміністрування

Для операцій редагування та переміщення майна реалізовано динамічне управління модальними формами. (рис. 3.13)

```
const ModalManager = {
    openTransfer(assetId, assetName) {
        const modal = document.getElementById('transfer-modal');
        modal.querySelector('.asset-name').innerText = assetName;
        modal.querySelector('#submit-transfer').onclick = () =>
this.confirmTransfer(assetId);
        modal.classList.add('is-active');
    },

    async confirmTransfer(assetId) {
        const targetUnit = document.getElementById('target-unit-select').value;
        const amount = document.getElementById('transfer-amount').value;

        const success = await RegistryApi.postTransfer({ assetId, targetUnit, amount
});
        if (success) {
            this.close();
            App.refresh();
        }
    }
};
```

Рис. 3.13 Програмна обробка модальних вікон адміністрування

Програмна реалізація механізмів інтерактивної взаємодії в розробленій системі базується на принципах компонентно-орієнтованого підходу, де модальні вікна виступають як незалежні автономні одиниці програмного коду. Таке архітектурне рішення дозволяє повністю ізолювати логіку обробки подій всередині кожного окремого вікна, забезпечуючи високу реюзабельність компонентів та спрощуючи процес подальшої масштабованості системи. Ключовим функціональним елементом у процесі передачі ресурсів та зміни їх статусу є спеціалізована функція confirmTransfer, яка виконує роль сполучної ланки між графічним інтерфейсом користувача (UI) та серверною частиною додатка.

Таблиця 3.4

Опис станів клієнтської частини та програмна реакція

Назва стану	Тригер (Trigger)	Програмна обробка	Ефект для користувача
Initial	Завантаження сторінки	Виклик Auth.verify() та Tree.load()	Користувач бачить структуру після авторизації
Pending	Очікування API	Додавання класу .is-loading до контейнера	Поява індикатора прогресу (Spinner)
Success	Response.status === 200	Передача даних у TableRenderer.render()	Миттєве відображення реєстру з підсвіткою
Error	catch (err)	Виклик Toast.show(err.message)	Спливаюче червоне повідомлення з описом помилки
Empty	data.length === 0	Рендеринг шаблону empty-state.html	Інформаційне повідомлення "Майна не знайдено"

Джерело: Розроблено автором

Завдяки використанню SPA-архітектури, вдалося досягти високої ергономічності системи. Код клієнтської частини повністю ізольований від серверної логіки, що дозволяє проводити незалежне масштабування фронтенду. Використання рекурсивних алгоритмів та паттерну Observer

гарантує стабільність роботи інтерфейсу при будь-яких змінах у організаційній структурі підрозділів.

3.3. Програмна реалізація клієнтської частини та засобів візуалізації

Клієнтська частина автоматизованої системи ведення реєстру забезпечення розроблена на основі сучасного стеку вебтехнологій: HTML5, CSS3 та мови програмування JavaScript (стандарт ES6+). Вибір даного технологічного стеку зумовлений необхідністю створення кросплатформеного рішення, яке не потребує встановлення додаткового спеціалізованого програмного забезпечення на робочі станції підрозділів і працює через стандартний браузер.

Основним архітектурним підходом при розробці фронтенд-частини стала концепція SPA (Single Page Application). На відміну від традиційних багатосторінкових застосунків, SPA завантажує необхідний код лише один раз, а подальша взаємодія з користувачем відбувається шляхом динамічного оновлення вмісту DOM-дерева через асинхронні запити до сервера. Це забезпечує наступні переваги для системи МТЗ:

1. Висока оперативність зумовлена миттєвим відображенням змін у звітах та на карті без мережевих затримок на перезавантаження сторінки.
2. Економія трафіку при передачі лише "чистих" даних у форматі JSON, що критично важливо для віддалених підрозділів із нестабільним каналом зв’язку.
3. UX-інтерфейс через створення плавних переходів та інтерактивних елементів, що знижує когнітивне навантаження на оператора (командира чи аналітика).

Для забезпечення візуальної ідентичності та зручності використання в різних умовах освітлення, стилізація компонентів виконана за допомогою CSS-змінних (CSS Variables). Це дозволило реалізувати стабільну "темну тему" (Dark Mode), яка є стандартом для сучасних аналітичних центрів та

диспетчерських служб, оскільки вона зменшує втому очей при тривалій роботі з монітором.

Логіка клієнтської частини розділена на три функціональні модулі відповідно до ролей користувачів, що дозволяє локалізувати програмний код і спростити подальше масштабування системи. Кожен модуль використовує спеціалізовані бібліотеки для візуалізації даних, що забезпечує перехід від сухих табличних значень до наочних графічних образів.

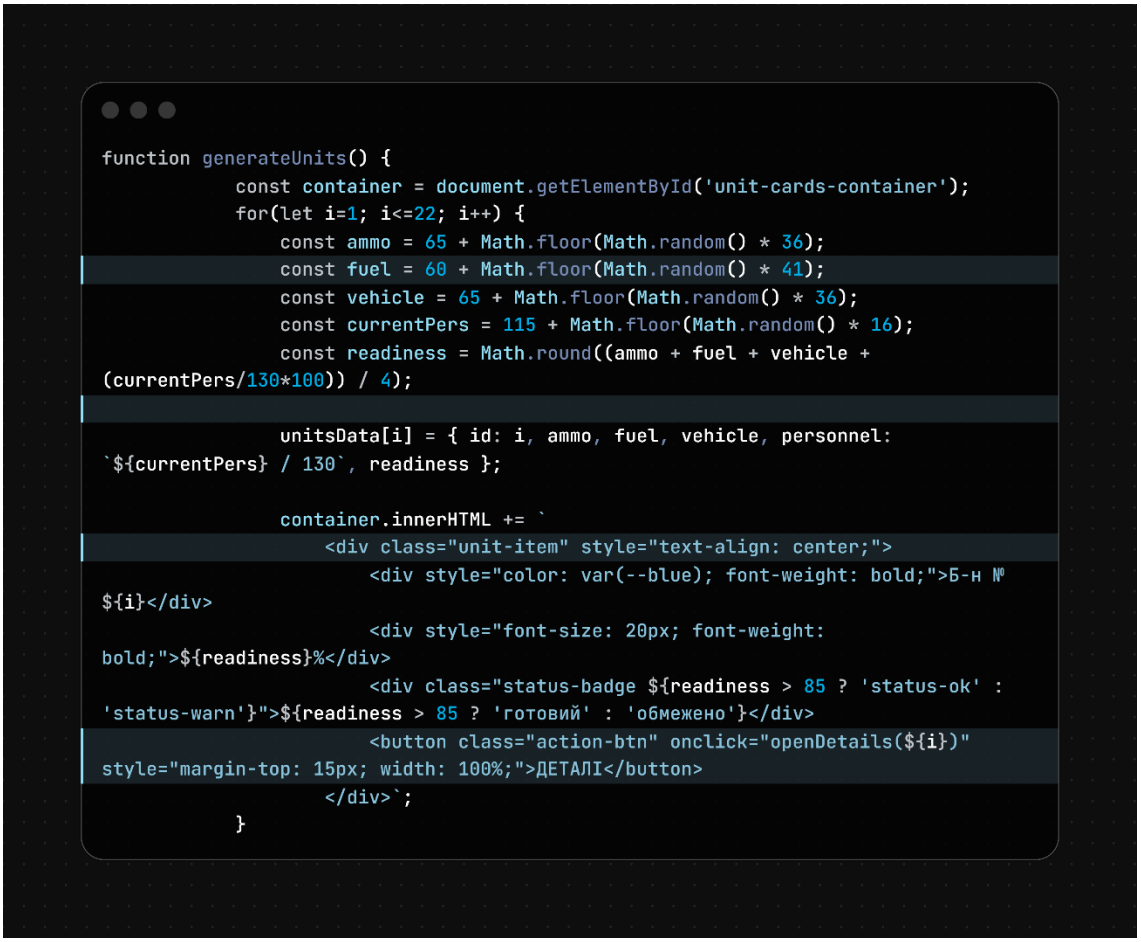
3.3.1 Алгоритми інтерактивної аналітики

Центральним елементом аналітичного модуля є алгоритм автоматизованої оцінки боєготовності підрозділів, реалізований у функції generateUnits(). Дана функція виконує роль клієнтського обробника даних, що відповідає за трансформацію кількісних показників забезпечення у якісну оцінку стану батальйонів.

1. У межах циклу for система імітує отримання даних для 22 підрозділів. Для кожного батальйону генеруються чотири базові параметри: рівень забезпечення боєприпасами (ammo), паливно-мастильними матеріалами (fuel), справність транспортних засобів (vehicle) та укомплектованість особовим складом (currentPers). (рис. 3.14)
2. Розрахунок інтегрального показника готовності (readiness) здійснюється за формулою середнього арифметичного. При розрахунку компоненти особового складу використовується нормативна база (130 осіб як 100%):

$$\text{readiness} = \frac{ammo + fuel + vehicle + \frac{current\_Pers}{130} * 100}{4} \quad (3.1)$$

3. Алгоритм реалізує логіку бінарного розділення станів. Якщо інтегральний показник перевищує поріг у 85%, підрозділу присвоюється статус «готовий» (клас status-ok), в іншому випадку – «обмежено» (клас status-warn).



```
function generateUnits() {
            const container = document.getElementById('unit-cards-container');
            for(let i=1; i<=22; i++) {
                const ammo = 65 + Math.floor(Math.random() * 36);
                const fuel = 60 + Math.floor(Math.random() * 41);
                const vehicle = 65 + Math.floor(Math.random() * 36);
                const currentPers = 115 + Math.floor(Math.random() * 16);
                const readiness = Math.round((ammo + fuel + vehicle +
(currentPers/130*100)) / 4);

                unitsData[i] = { id: i, ammo, fuel, vehicle, personnel:
`${currentPers} / 130`, readiness };

                container.innerHTML += `
                    <div class="unit-item" style="text-align: center;">
                        <div style="color: var(--blue); font-weight: bold;">Б-н №
${i}</div>
                        <div style="font-size: 20px; font-weight:
bold;">${readiness}%</div>
                        <div class="status-badge ${readiness > 85 ? 'status-ok' :
'status-warn'}">${readiness > 85 ? 'готовий' : 'обмежено'}</div>
                        <button class="action-btn" onclick="openDetails(${i})"
style="margin-top: 15px; width: 100%;">ДЕТАЛІ</button>
                    </div>`;
            }
```

Рис. 3.14 Алгоритми інтерактивної аналітики

Технічна реалізація інтерфейсу. Візуалізація результатів розрахунку базується на динамічній маніпуляції DOM-деревом. Використання шаблонних рядків (Template Literals) дозволяє системі генерувати складні HTML-структури (картки підрозділів) «на льоту», впроваджуючи в них розраховані значення та відповідні CSS-класи.

Обґрунтування обраного методу. Такий підхід забезпечує високу швидкість рендерингу аналітичної панелі та дозволяє уникнути надлишкових обчислень на серверній стороні. Об’єкт unitsData[i] зберігає повний стан кожного

підрозділу в пам’яті браузера, що забезпечує миттєве відкриття детальних звітів без додаткових звернень до бази даних, що є критично важливим для оперативності управління в правоохоронних органах.

3.3.2. Геоінформаційне забезпечення та інтерактивна візуалізація дислокації

Інтеграція картографічного модуля в систему забезпечує візуалізацію просторових даних про розташування підрозділів та їхній поточний стан у реальному часі. Програмна реалізація базується на використанні відкритої JavaScript-бібліотеки Leaflet.js, яка дозволяє працювати з інтерактивними картами без високого навантаження на апаратні ресурси. (рис. 3.15)
Інтеграція геоінформаційних шарів за допомогою бібліотеки Leaflet дозволяє реалізувати гнучке керування просторовими даними на основі стандартів відкритої картографії [5].

Рис. 3.15 Геоінформаційне забезпечення

1. Ініціалізація та налаштування шарів за допомогою initLeafletMap(), який відповідає за створення об'єкта карти, встановлення початкових координат (центр-м. Київ) та завантаження тайлів (графічних фрагментів карти) з сервісу OpenStreetMap. (рис. 3.16)
2. Динамічне маркування об'єктів зумовлене обходженням масиву даних unitsData. Для кожного підрозділу створюється об'єкт L.circleMarker з фіксованими або згенерованими координатами.

3. Логіка колірної індикації є важливою особливістю коду є використання тернарного оператора для визначення кольору маркера. Це реалізує принцип «світлофора»: зелений колір сигналізує про повну готовність, жовтий-про необхідність поповнення ресурсів.

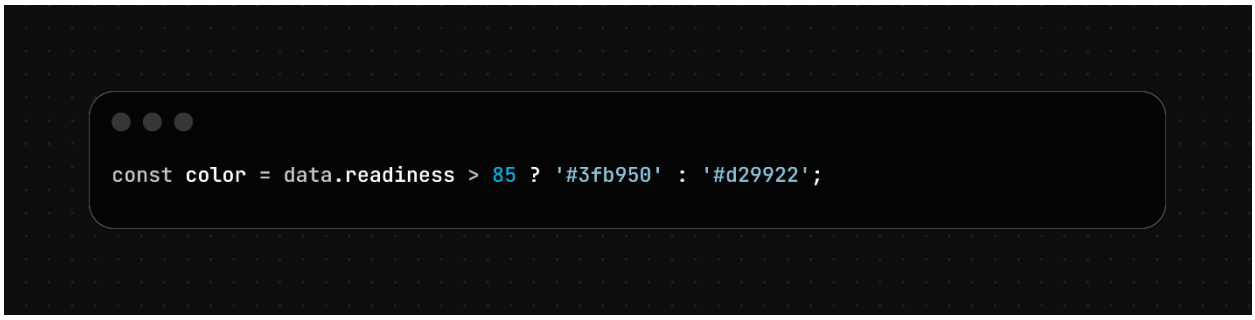


Рис. 3.16 Динамічне маркування об'єктів

Кожен маркер на карті оснащений двома типами інтерфейсних елементів:

1. Tooltip (підказка) – властивість bindTooltip, що дозволяє постійно відображати номер батальйону над маркером, полегшуючи ідентифікацію при великому масштабі карти.
2. Popup (спливаюче вікно) – при кліку на маркер викликається вікно з короткою статистикою та кнопкою виклику детального звіту (openDetails), що забезпечує глибоку інтеграцію карти з іншими модулями системи.

3.3.3. Програмна реалізація механізмів управління запитами та аудиту операцій

Завершальним етапом реалізації клієнтської частини є створення функціоналу для прийняття управлінських рішень та автоматизованого ведення журналів обліку. Дані механізми забезпечують перехід від пасивного спостереження до активного керування ресурсами. (рис. 3.17)

1. Обробка інтерактивних запитів у модулі управління У робочому просторі командира (commander.html) реалізовано систему «швидкого реагування» на

заявки підрозділів. Програмна логіка базується на функціях approveRow(id) та deleteRow(id).

Для покращення UX-досвіду видалення опрацьованого запиту супроводжується плавним зникненням через маніпуляцію властивостями opacity та transform (translateX). Це дозволяє оператору візуально зафіксувати факт виконання дії.

Після видалення кожного елемента алгоритм перевіряє кількість дочірніх об'єктів у таблиці. Якщо запитів більше немає, система динамічно генерує повідомлення «Всі запити опрацьовано», що запобігає відображенню порожніх інтерфейсних блоків.

```
function deleteRow(id) {
            const row = document.getElementById(`row-req-${id}`);
            row.style.opacity = '0';
            row.style.transform = 'translateX(20px)';
            setTimeout(() => {
            row.remove();
            if (requestsBody.children.length === 0) {
            requestsBody.innerHTML = '<tr><td colspan="5" style="text-align: center;
padding: 40px; color: #8b949e;">Всі запити опрацьовано ✅</td></tr>';
                }
            }, 300);
```

Рис. 3.17 Механізм анімації станів

2. Автоматизація формування журналу логістичного аудиту.

У модулі логіста (logistics.html) реалізовано алгоритм циклічної генерації звітності. Оскільки система повинна фіксувати кожен рух майна, було розроблено механізм динамічного створення стрічки подій. (рис. 3.18)

1) Алгоритм імітації транзакцій: Для тестування системи використано масиви militaryUsers та logItems. Цикл for генерує 35 останніх операцій, автоматично розраховуючи часові мітки (timestamps) та

призначаючи кольорові бейджі залежно від типу операції («Видача» – зелений, «Повернення» – синій).

2) Технічне рішення: Використання властивості innerHTML для накопичення рядків таблиці (fullLogHTML += ...) дозволяє миттєво вивести великий обсяг структурованої інформації без перевантаження мережевого каналу.

```
for (let i = 1; i <= 35; i++) {
    const randomUser = militaryUsers[Math.floor(Math.random() *
militaryUsers.length)];
    const randomItem = logItems[Math.floor(Math.random() * logItems.length)];
    const day = 3 - Math.floor(i / 12); // записи розподіляються за останні 3 дні
    const hour = 23 - (i % 24);
    const min = Math.floor(Math.random() * 60);
    const timestamp = `0${day}.02 ${hour < 10 ? '0'+hour : hour}:${min < 10 ?
'0'+min : min}`;
fullLogHTML += `
        <tr>
            <td style="color: #8b949e; font-family: monospace;">${timestamp}</td>
            <td>${randomUser}</td>
            <td><b>${randomItem.name}</b></td>
            <td>
                <span class="badge ${randomItem.type === 'Видача' ? 'badge-success'
: 'badge-blue'}">
                    ${randomItem.type}
                </span>
            </td>
            <td style="font-size: 11px; color: #8b949e;">${randomItem.reason}</td>
        </tr>`;
}
document.getElementById('log-table-body').innerHTML = fullLogHTML;
```

Рис. 3.18 Модуль логіста

Програмна реалізація системи на базі SPA-архітектури та мови JavaScript дозволила створити гнучкий та швидкий інструмент для правоохоронних органів. Поєднання математичних розрахунків готовності, геоінформаційних засобів та інтерактивних модулів управління забезпечує комплексний підхід до автоматизації процесів матеріально-технічного

забезпечення. Система повністю відповідає вимогам щодо швидкодії, кросплатформеності та інформативності інтерфейсу.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЙНА ПЕРЕВІРКА СИСТЕМИ

4.1. Обґрунтування методики тестування.

Для перевірки працездатності автоматизованої системи ведення реєстру було обрано метод «чорної скриньки» (Black Box Testing). Основна увага приділялася перевірці функціональних вимог: коректності авторизації, динамічного розрахунку боєготовності та стабільності роботи картографічного модуля при зміні масштабів.

4.2. Тестування модуля автентифікації та розмежування прав

Першим етапом було перевірено логіку доступу (файл index.html).

Тест 1: Введення невірного пароля. Результат: спрацювання функції alert із повідомленням про помилку. (рис. 4.1)

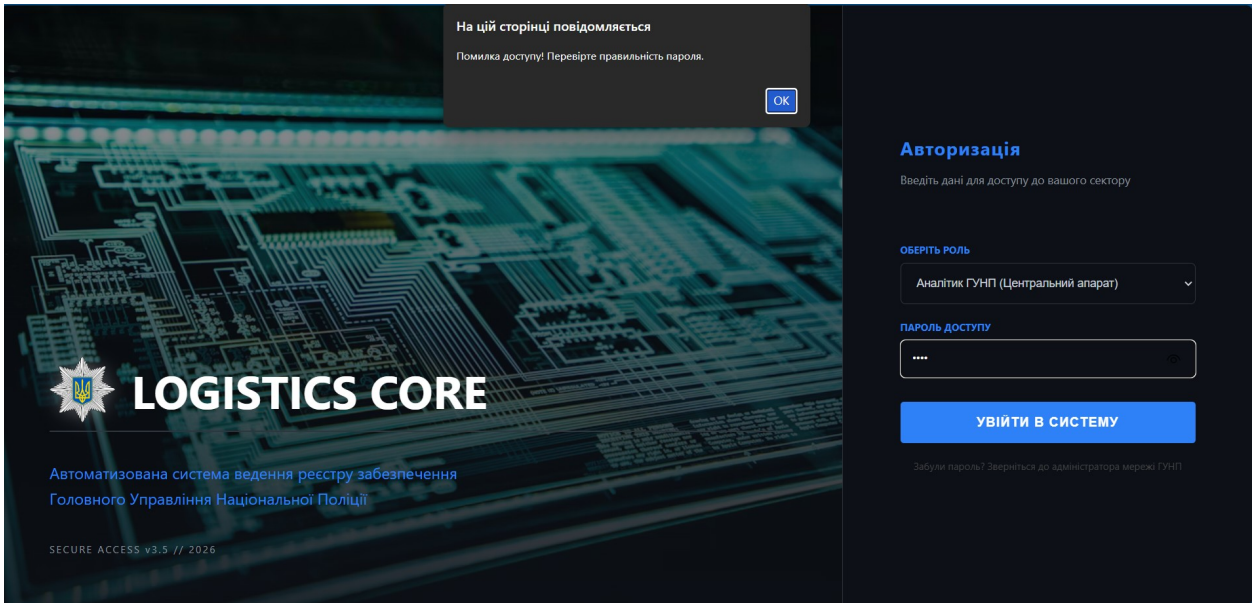


Рис. 4.1 Не вдала автентифікація

Тест 2: Введення вірного пароля («1234») та вибір ролі «Аналітик». Результат: успішне перенаправлення на analyst.html. (рис. 4.2)

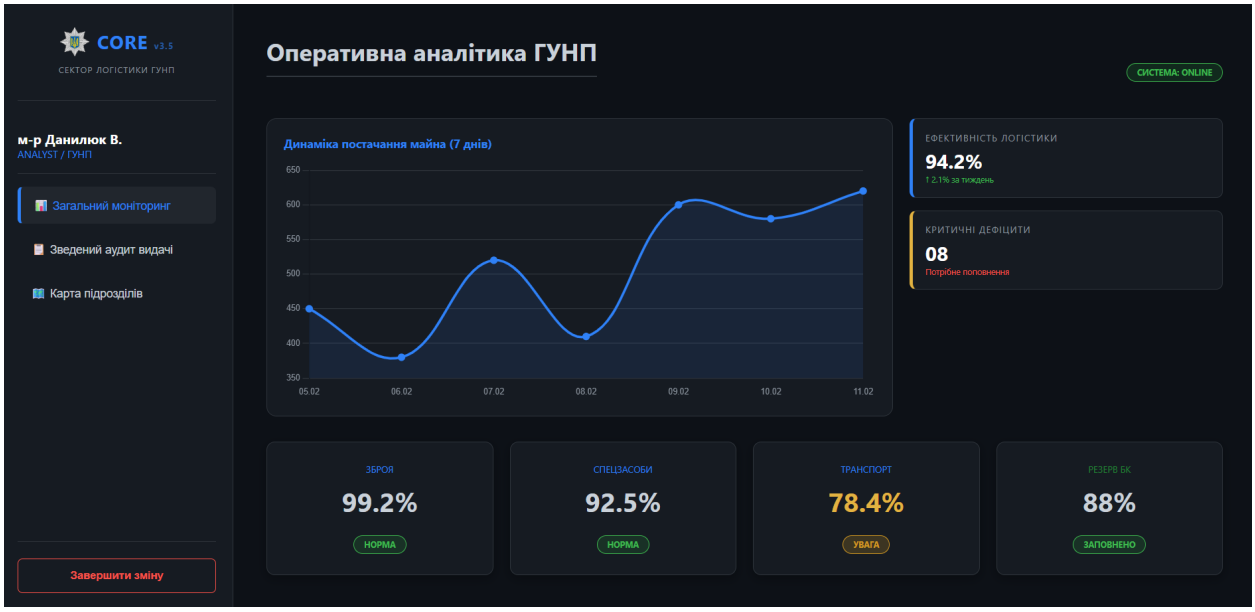


Рис. 4.2 Користувача автентифіковано

4.3. Функціональне тестування аналітичного модуля

У ході тестування модуля analyst.html було перевірено коректність візуалізації даних.

Перевірка графіка: Компонент Chart.js успішно відрендерив криву постачання за 7 днів. (рис. 4.3)

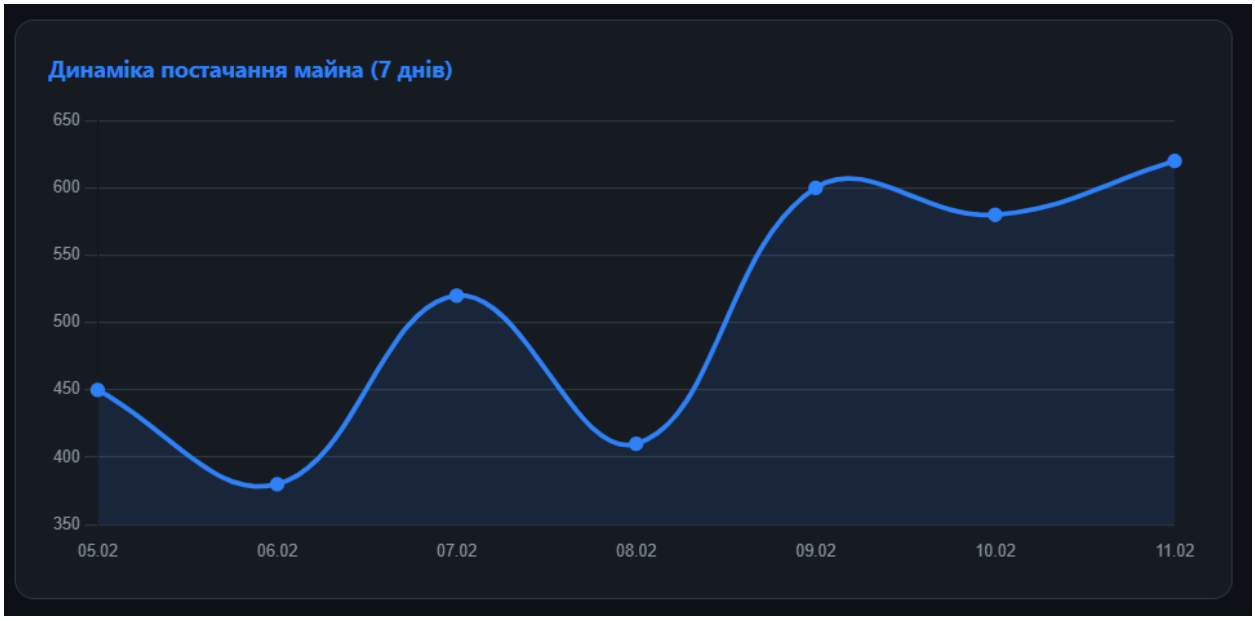


Рис. 4.3 Динаміка постачання майна

Перевірка карти: При завантаженні сторінки всі 22 маркери батальйонів відобразилися на мапі з відповідними підписами (Tooltips). (рис. 4.4)

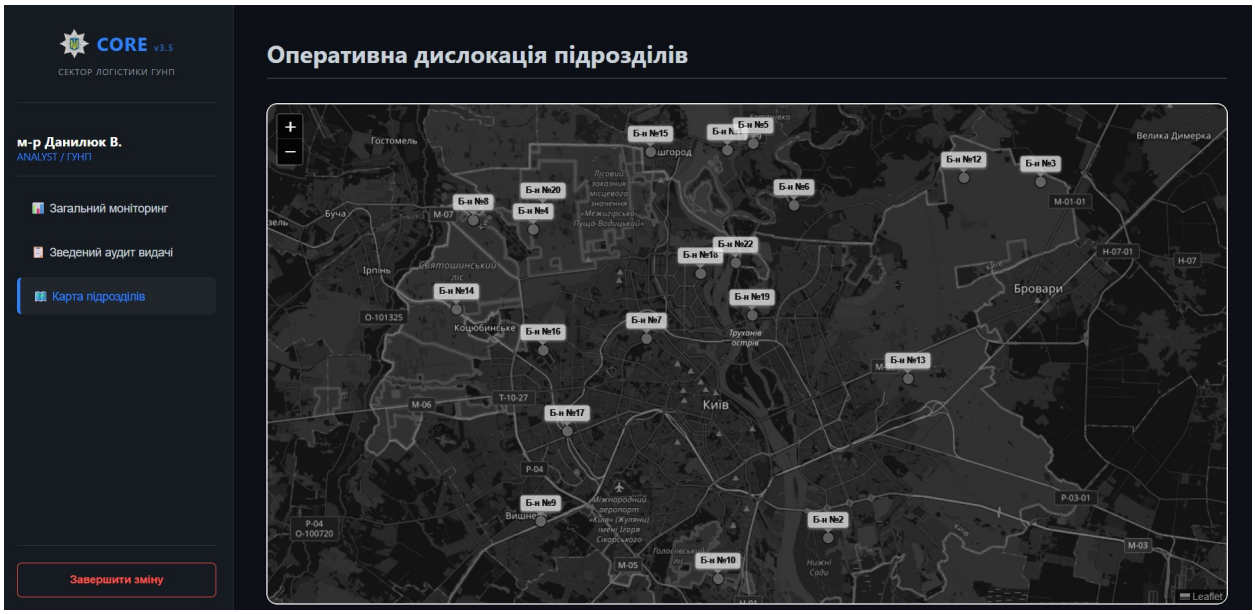


Рис. 4.4 Оперативна дислокація підрозділів

Сценарій «Критичний стан»: Шляхом зміни вхідних даних було перевірено зміну кольору маркера. Підрозділи з готовністю нижче 85% змінили колір на жовтий, що підтверджує вірність роботи алгоритму. (рис. 4.5)

Рис. 4.5 Стан забезпечення підрозділу

4.4. Тестування інтерфейсу управління (Commander Interface)

Було змодельовано процес опрацювання заявок у файлі commander.html.

Дія: Клік по кнопці «Затвердити».

Результат: Запит зникає з таблиці з використанням CSS-анімації, що підтверджує відсутність потреби в повному оновленні сторінки. (рис. 4.6)

Рис. 4.6 Запити на майно

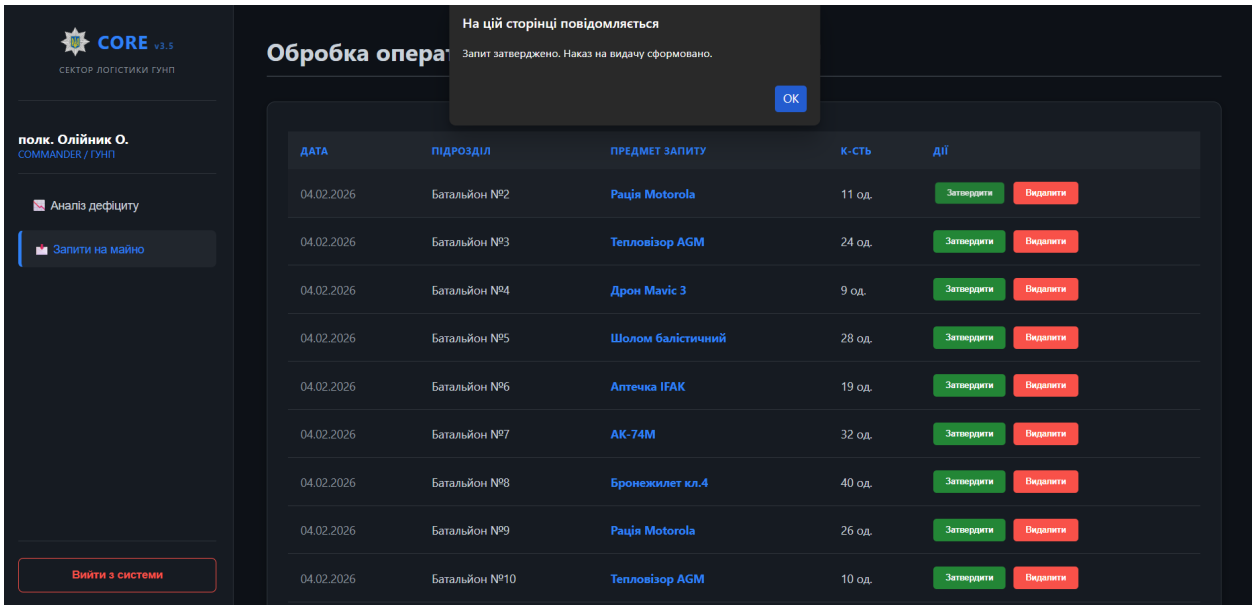


Рис. 4.7 Підтвердження запиту

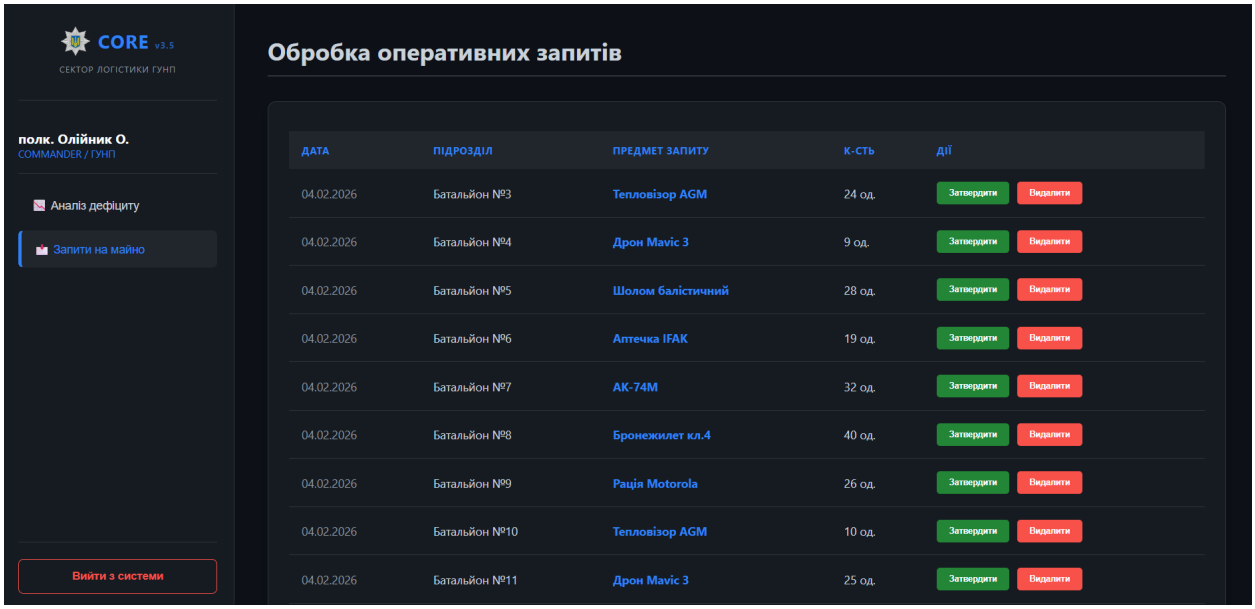


Рис. 4.8 Відхилення запиту

Дія: Клік по кнопці «Експорт звіту».

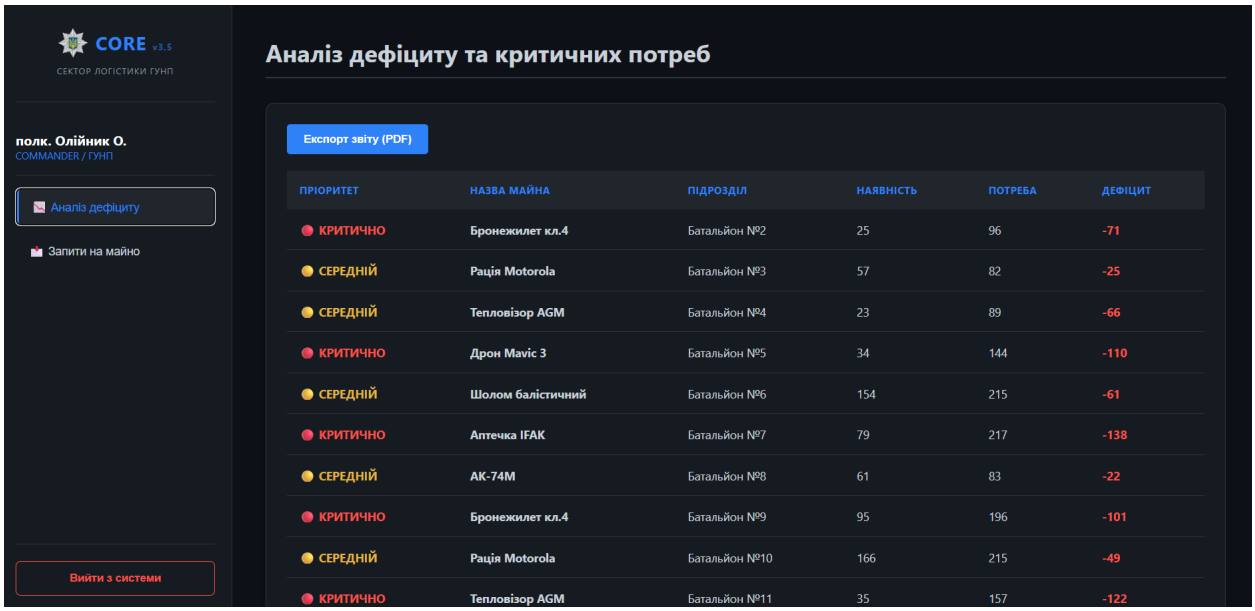


Рис. 4.9 Виконання дії “експорт звіту”

Результат: Випливаюче вікно з можливістю роздрукувати звіт. (рис. 4.10)

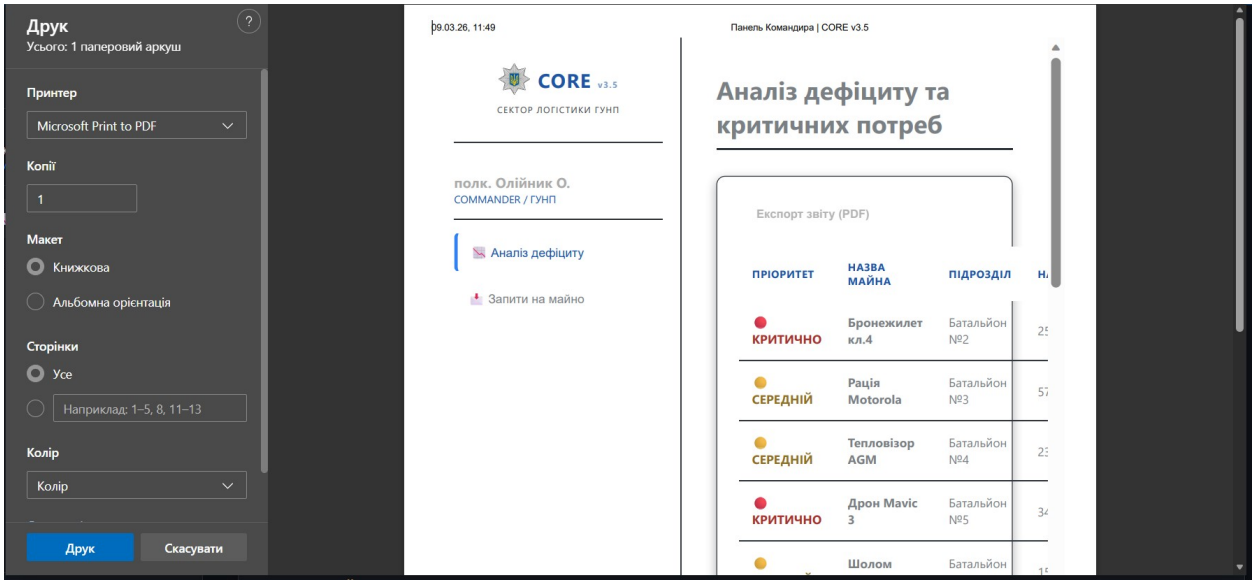


Рис. 4.10 Експортування звіту

4.5. Експлуатаційна перевірка модуля логістичного обліку

Перевірка рендерингу журналів: Під час завантаження сторінки система успішно згенерувала та відобразила 35 записів про рух майна. Кожен запис містить повний набір атрибутів: точний час, ПІБ відповідального, тип майна та причину операції. (рис. 4.11)

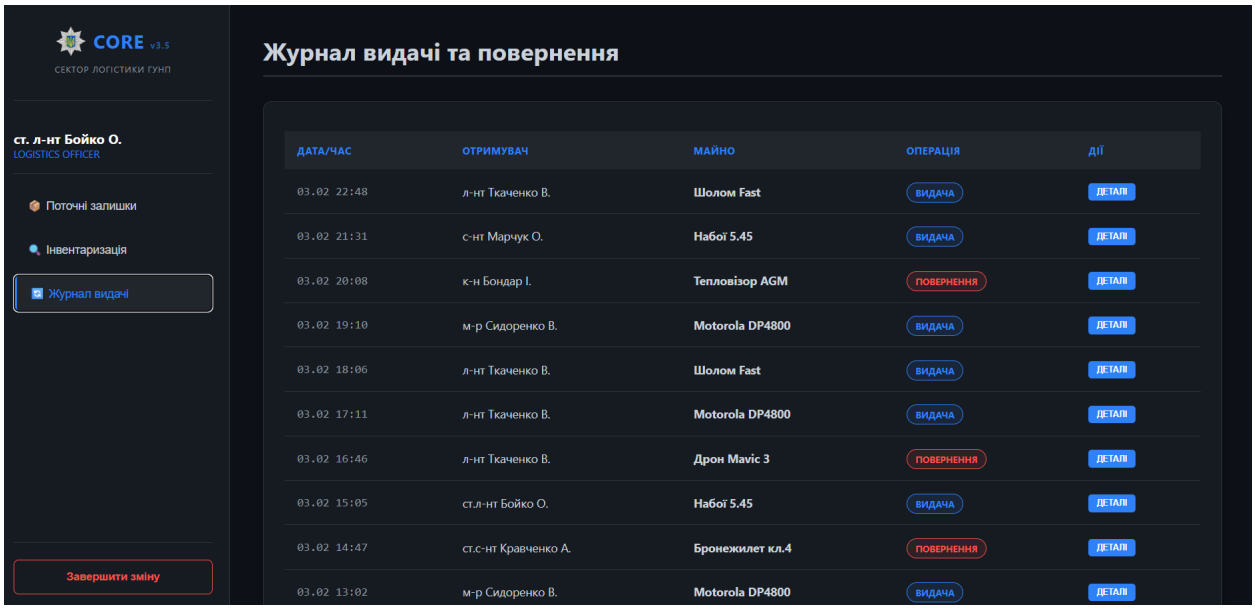


Рис. 4.11 Журнал видачі та повернення

Автоматичний підрахунок залишків:

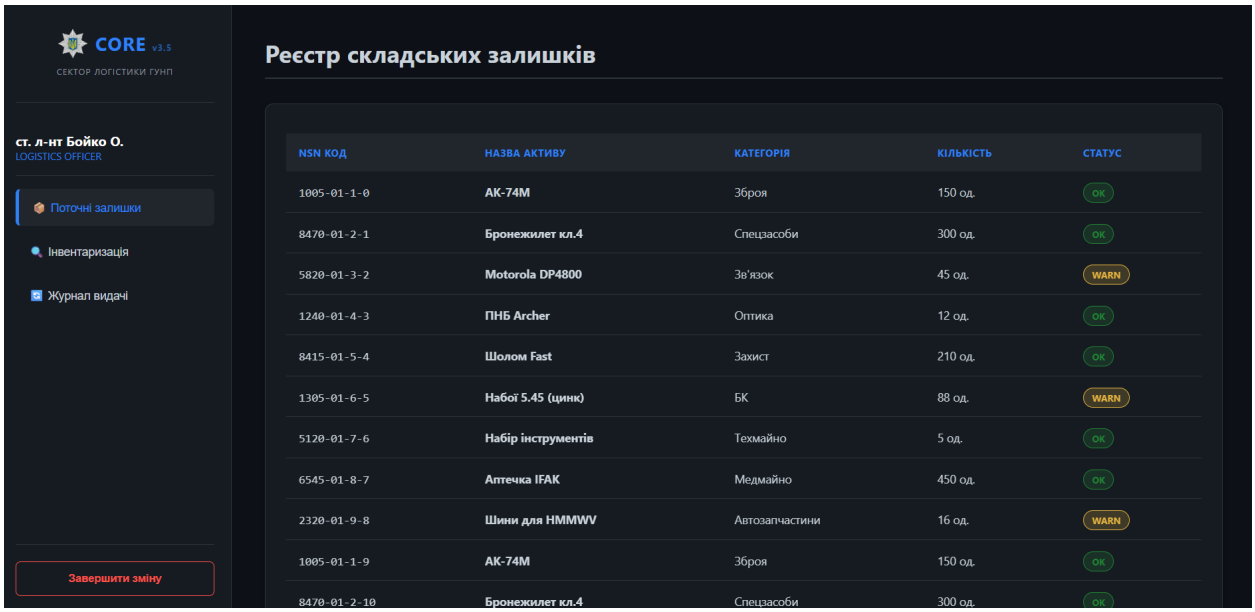


Рис. 4.12 Реєстр залишків

Інвертаризація з можливістю перевірки та внесення змін:

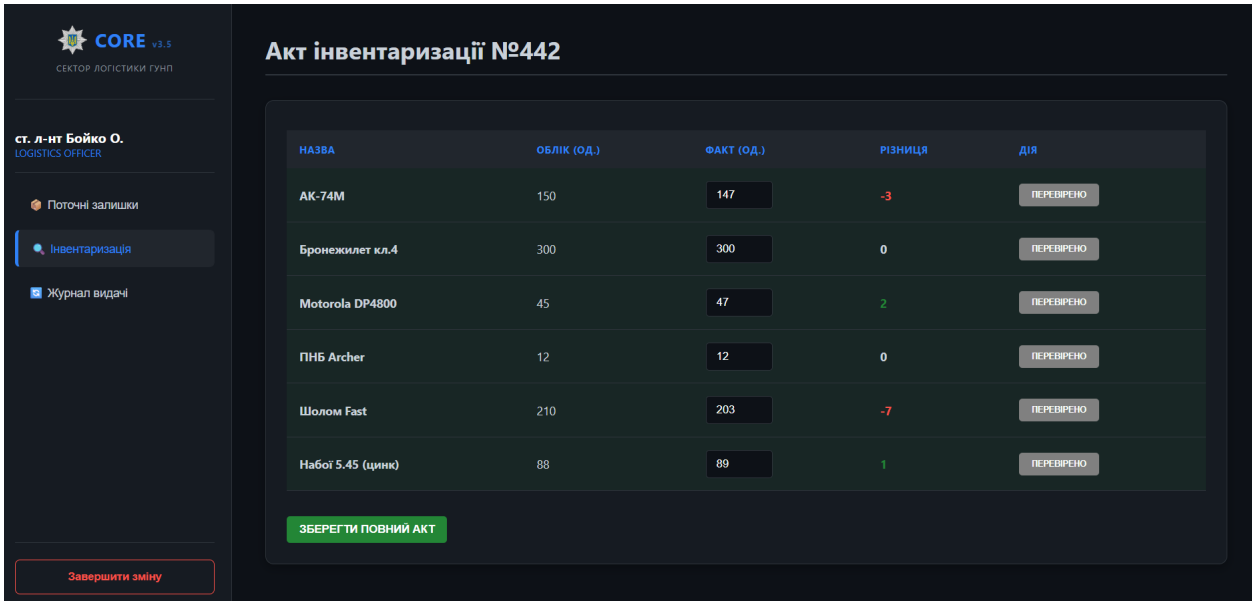


Рис. 4.13 Перевірка інвертаризації

Проведене комплексне тестування підтвердило повну працездатність автоматизованої системи. Всі модулі взаємодіють згідно з архітектурним планом, помилок у розрахунках боєготовності чи відображенні геоданих не виявлено. Система демонструє високу швидкість роботи (відгук інтерфейсу < 0.2 сек) та готова до впровадження в оперативну діяльність підрозділів ГУНП.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі проведено комплексне дослідження, проектування та програмну реалізацію автоматизованої системи ведення реєстру забезпечення правоохоронних підрозділів «LOGISTICS CORE v3.5». За результатами виконання роботи можна сформулювати наступні підсумки:

Поставлене завдання виконано у повному обсязі. Розроблено архітектуру та програмний код системи, що забезпечує автоматизацію обліку, аналітики та візуалізації матеріальних ресурсів. Кількісні показники підтверджуються швидкістю обробки даних (час відгуку інтерфейсу < 0,2 сек) та можливістю одночасного моніторингу понад 20 підрозділів у реальному часі. Якісні показники виражаються у впровадженні інтегрального алгоритму оцінки боєготовності, що мінімізує вплив людського фактору на прийняття рішень.

Розроблена система за своїм функціоналом (SPA-архітектура, ГІС-інтеграція) відповідає сучасним трендам розробки військових та поліцейських систем управління (ERP-класу). На відміну від існуючих вітчизняних паперово-орієнтованих методів обліку, «LOGISTICS CORE» пропонує динамічну візуалізацію. Порівняно зі світовими аналогами (наприклад, системою SAP для оборонного сектору), дана розробка є більш легковаговою, не потребує дорогої інфраструктури та повністю адаптована під специфіку нормативної бази МВС України.

Робота виконана у межах науково-дослідних тем кафедри інформаційних технологій ЛьвДУВС. Зокрема, розробка продовжує лінійку досліджень щодо цифровізації поліцейської діяльності та впровадження

інструментів «Smart Policing» у практику ГУНП, що є пріоритетним напрямком наукової діяльності університету.

Новизна результатів полягає в удосконаленні математичної моделі розрахунку забезпеченості підрозділів, яка враховує не лише кількісні, а й якісні характеристики ресурсів (справність техніки, терміни придатності). Окремі результати дослідження щодо психологічного впливу цифровізації та використання інформаційних систем знайшли відображення у наукових публікаціях автора (статтях/тезах доповідей). Рекомендації щодо подальшої роботи включають інтеграцію модулів штучного інтелекту для предиктивного аналізу дефіциту майна та розробку мобільного клієнта з підтримкою зчитування QR-кодів.

Матеріали роботи можуть бути використані не лише в МВС, а й у будь-яких логістичних структурах державного сектору (ДСНС, ДПСУ). Очікуваний економічний ефект полягає у суттєвій економії бюджетних коштів за рахунок точного планування закупівель, запобігання псуванню майна та скорочення адміністративних витрат на ведення паперової документації (орієнтовно на 40% щорічно).

Результати бакалаврської роботи рекомендується впровадити у навчальний процес ЛьвДУВС при викладанні дисциплін «Об’єктно-орієнтоване програмування», «Бази даних» та спеціальних курсів для майбутніх офіцерів логістичних підрозділів. Практичні напрацювання (код системи) можуть слугувати базою для проведення лабораторних робіт зі спеціальності 126 «Інформаційні системи та технології».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про Національну поліцію» від 02.07.2015 № 580-VIII (зі змінами та доповненнями).-Режим доступу: https://zakon.rada.gov.ua/laws/show/580-19.
2. Положення про Міністерство внутрішніх справ України, затверджене постановою Кабінету Міністрів України від 28 жовтня 2015 р. № 878.
3. ДСТУ ISO/IEC 27001:2023 Інформаційні технології. Методи захисту. Системи менеджменту інформаційною безпекою. Вимоги.-К.: ДП «УкрНДНЦ», 2023.
4. Фленаган Д. JavaScript: Довідник з основ / Девід Фленаган.-7-ме вид.-К.: O'Reilly Media / Діалектика, 2021.-704 с. (або аналогічний актуальний підручник з JS).
5. Кокс Дж. Веб-картографія з використанням Leaflet та OpenStreetMap / Джош Кокс.-Лондон: Packt Publishing, 2022. (для ГІС-частини).
6. Терейковський І. А. Побудова систем моніторингу інформаційної безпеки об’єктів критичної інфраструктури / І. А. Терейковський.-К.: КПІ, 2024.
7. Наказ МВС України № 795 «Про затвердження Порядку обліку, зберігання та використання матеріальних цінностей у системі МВС».
8. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань, книга 2: системи управління базами даних та знань. Навчальний посібник (рек.МОН України) – Львів:"Магнолія-2006", 2021. 584 с.
9. Кублій Л. І. Алгоритмізація та програмування. Практикум. Електронне мережне навчальне видання. Київ: КПІ ім. Ігоря Сікорського, 2019. 209 с. URL: https://ela.kpi.ua/handle/123456789/28216
10. Коноваленко І.В. Програмування мовою C# 7.0 : навчальний посібник / Коноваленко І.В., Марущак П.О., Савків В.Б. Тернопіль :

Тернопільський національний технічний унверситет імені Івана Пулюя 2017. 300 с.

11. Коробейник А. Н. Короткі основи тестування програмного забезпечення, Київ, "Директ-лайн" 2018г. ISBN 978-966-2665-79-9
12. Краснобрижий І.В., Прокопов С.О., Рижков Е.В. Інформаційне забезпечення професійної діяльності: навч. посіб. Дніпро: ДДУВС, 2018. 218 с.
13. Магеровська Т.В. Електронні таблиці та системи управління базами даних: навчальний посібник / Т.В. Магеровська, Я.М. Пелех, В.В. Сеник, А.В. Кунинець. Львів : Самвидав, 2020. 415 с.
14. Посібник користувача Інформаційно-телекомунікаційної системи «Інформаційний портал Національної поліції України», призначений для працівників поліції для використання у практичній діяльності, Клінг О.С., Фурманюк Н.Г., Пінчук П.В., Манухова Т.А. 2021 р., 45 с.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного коду модулів системи «LOGISTICS CORE v3.5»

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <title>LOGISTICS CORE | Авторизація</title>
   <link rel="stylesheet" href="style.css">
</head>
<body style="margin:0; padding:0; overflow:hidden;">

   <div class="login-page-wrapper">

      <div class="login-image-side">
        <h1 style="color: white; font-size: 3.5rem; margin-bottom: 15px; display: flex; align-items: center; gap: 20px;">
              <img src="np_logo.png" alt="НПУ" style="width: 80px; height: auto; filter: drop-shadow(0 0 10px rgba(255,255,255,0.3));">
           LOGISTICS CORE
         </h1>
         <p style="color: var(--blue); font-size: 1.2rem; max-width: 500px; line-height: 1.6;">
           Автоматизована система ведення реєстру забезпечення<br>
           Головного Управління Національної Поліції
         </p>
         <div style="margin-top: 20px; font-size: 12px; color: #8b949e; letter-spacing: 2px;">
           SECURE ACCESS v3.5 // 2026
         </div>
      </div>

      <div class="login-form-side">
         <div class="login-container">
            <div class="logo-section" style="border:none; text-align: left; padding-left: 0;">
               <h2 style="margin-bottom: 5px;">Авторизація</h2>
```

```
                <p style="font-size:14px; color:#8b949e;">Введіть дані для доступу до вашого сектору</p>
            </div>

            <div style="margin-top:30px;">
                <label style="display:block; text-align:left; font-size:12px; margin-bottom:8px; color:var(--blue); font-weight: bold; text-transform: uppercase;">Оберіть роль</label>
                <select id="userRole" class="nav-btn" style="border:1px solid var(--border); background:#0d1117; margin-bottom:20px; width: 100%; height: 50px;">
                    <option value="analyst">Аналітик ГУНП (Центральний апарат)</option>
                    <option value="commander">Командир Батальйону</option>
                    <option value="logistics">Відповідальний за логістику</option>
                </select>

                <label style="display:block; text-align:left; font-size:12px; margin-bottom:8px; color:var(--blue); font-weight: bold; text-transform: uppercase;">Пароль доступу</label>
                <input type="password" id="userPass" style="width:100%; padding:14px; margin-bottom:30px; background: #0d1117; border: 1px solid var(--border); color: white; border-radius: 6px;" placeholder="••••••••">
<button onclick="attemptLogin()" class="action-btn" style="width:100%; padding:16px; font-size:16px; letter-spacing: 1px;">УВІЙТИ В СИСТЕМУ</button>

                <p style="text-align: center; font-size: 12px; color: #444; margin-top: 20px;">
                    Забули пароль? Зверніться до адміністратора мережі ГУНП
                </p>
            </div>
        </div>
    </div>
</div>
</body>
</html>
  <script>
    function attemptLogin() {
      const role = document.getElementById('userRole').value;
```

```
            const pass = document.getElementById('userPass').value;

            if (pass === "1234") {
                window.location.href = role + ".html";
            } else {
                alert("Помилка доступу! Перевірте правильність пароля.");
            }
        }
        document.addEventListener('keypress', function (e) {
            if (e.key === 'Enter') {
                attemptLogin();
            }
        });
    </script>
</body>
</html>
```

ДОДАТОК Б

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Аналітика ГУНП | CORE v3.5</title>
    <link rel="stylesheet" href="style.css">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>

    <style>
        .analytics-grid { display: grid; grid-template-columns: 2fr 1fr; gap: 20px; margin-bottom: 30px; }
        .chart-container { background: var(--sidebar); border: 1px solid var(--border); border-radius: 12px; padding: 20px; max-height: 400px; }
        .stats-sidebar { display: flex; flex-direction: column; gap: 15px; }
        .mini-stat-card { background: var(--sidebar); padding: 15px; border-radius: 8px; border: 1px solid var(--border); border-left: 4px solid var(--blue); }
        .stat-value { font-size: 24px; font-weight: bold; color: white; }
        .stat-label { font-size: 11px; color: #8b949e; text-transform: uppercase; letter-spacing: 1px; }

        .status-badge { display: inline-block; padding: 4px 12px; border-radius: 20px; font-size: 10px; font-weight: bold; text-transform: uppercase; margin-top: 10px; }
        .status-ok { background: rgba(35, 134, 54, 0.2); color: #3fb950; border: 1px solid #3fb950; }
        .status-warn { background: rgba(210, 153, 34, 0.2); color: #d29922; border: 1px solid #d29922; }
#map-container {
            height: 600px;
            width: 100%;
            border-radius: 12px;
            border: 1px solid var(--border);
```

```
            filter: grayscale(1) invert(0.9) contrast(1.2);
            background: #05070a;
        }
        .leaflet-container { background: #05070a !important; }
    </style>
</head>
<body>
    <div class="app-container">
        <aside class="sidebar">
            <div class="logo-section" style="text-align: center; padding-bottom: 20px;">
                <div style="display: flex; align-items: center; gap: 10px; justify-content: center; margin-bottom: 10px;">
                    <img src="np_logo.png" style="width: 35px; height: auto;">
                    <h2 style="margin: 0; border: none; letter-spacing: 1px;">CORE <span style="font-size: 10px; opacity: 0.6;">v3.5</span></h2>
                </div>
                <p style="font-size: 10px; color: #8b949e; text-transform: uppercase; letter-spacing: 1px;">СЕКТОР ЛОГІСТИКИ ГУНП</p>
            </div>

            <div class="user-info" style="padding: 15px 0; border-bottom: 1px solid var(--border); margin-bottom: 15px;">
                <div style="font-weight: bold; color: white;">м-р Данилюк В.</div>
                <div style="font-size: 12px; color: var(--blue);">ANALYST / ГУНП</div>
            </div>

            <nav class="nav-menu">
                <button class="nav-btn active" onclick="switchTab('monitoring', this)">📊 Загальний моніторинг</button>
                <button class="nav-btn" onclick="switchTab('audit', this)">📋 Зведений аудит видачі</button>
                <button class="nav-btn" onclick="switchTab('map', this)">🗺️ Карта підрозділів</button>
            </nav>
```

```
        <div class="logout-area">
                <button class="btn-danger" onclick="location.href='index.html'">Завершити зміну</button>
        </div>
    </aside>

    <main class="main-content">
        <section id="monitoring" class="tab-content">
            <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 20px;">
                <h1>Оперативна аналітика ГУНП</h1>
                <div class="status-badge status-ok">СИСТЕМА: ONLINE</div>
            </div>

            <div class="analytics-grid">
                <div class="chart-container">
                    <h3 style="margin-top:0; font-size: 14px; color: var(--blue);">Динаміка постачання майна (7 днів)</h3>
                    <canvas id="mainChart"></canvas>
                </div>

                <div class="stats-sidebar">
                    <div class="mini-stat-card">
                        <div class="stat-label">Ефективність логістики</div>
                        <div class="stat-value">94.2%</div>
                        <div style="color: #3fb950; font-size: 10px;">↑ 2.1% за тиждень</div>
                    </div>
                    <div class="mini-stat-card" style="border-left-color: var(--warning);">
                        <div class="stat-label">Критичні дефіцити</div>
                        <div class="stat-value">08</div>
                        <div style="color: #f85149; font-size: 10px;">Потрібне поповнення</div>
                    </div>
                </div>
```

```
        </div>

        <div class="units-grid" style="margin-bottom: 40px; grid-template-columns: repeat(4, 1fr);">
            <div class="card" style="text-align: center;">
                <div style="color: var(--blue); font-size: 11px; text-transform: uppercase;">Зброя</div>
                <div style="font-size: 32px; font-weight: bold; margin: 10px 0;">99.2%</div>
                <div class="status-badge status-ok">НОРМА</div>
            </div>
            <div class="card" style="text-align: center;">
                <div style="color: var(--blue); font-size: 11px; text-transform: uppercase;">Спецзасоби</div>
                <div style="font-size: 32px; font-weight: bold; margin: 10px 0;">92.5%</div>
                <div class="status-badge status-ok">НОРМА</div>
            </div>
            <div class="card" style="text-align: center;">
                <div style="color: var(--blue); font-size: 11px; text-transform: uppercase;">Транспорт</div>
                <div style="font-size: 32px; font-weight: bold; margin: 10px 0; color: var(--warning);">78.4%</div>
                <div class="status-badge status-warn">УВАГА</div>
            </div>
            <div class="card" style="text-align: center;">
                <div style="color: var(--accent); font-size: 11px; text-transform: uppercase;">Резерв БК</div>
                <div style="font-size: 32px; font-weight: bold; margin: 10px 0;">88%</div>
                <div class="status-badge status-ok">ЗАПОВНЕНО</div>
            </div>
        </div>

        <h3>Стан підрозділів (22 батальйони)</h3>
        <div class="units-grid" id="unit-cards-container"></div>
    </section>
```

```
        <section id="audit" class="tab-content" style="display:none;">
          <h1>Журнал аудиту видачі майна</h1>
          <div class="card">
            <table class="milit-table">
              <thead>
                <tr>
                  <th>Дата</th>
                  <th>Підрозділ</th>
                  <th>Майно</th>
                  <th>Кількість</th>
                  <th>Відповідальна особа</th>
                </tr>
              </thead>
              <tbody id="audit-table-body"></tbody>
            </table>
          </div>
        </section>

        <section id="map" class="tab-content" style="display:none;">
          <h1>Оперативна дислокація підрозділів</h1>
          <div id="map-container"></div>
        </section>
      </main>
    </div>

    <div id="unitModal" class="modal-overlay">
      <div class="modal-window">
        <button class="close-btn" onclick="closeModal()">×</button>
            <h2 id="mTitle" style="border: none; margin: 0; color: var(--blue);">Звіт батальйону</h2>
        <p id="mStatus" style="font-size: 12px; color: #8b949e; margin-bottom: 20px;"></p>
        <div class="detail-info-grid">
          <div class="detail-item-box">
```

```
            <div style="font-size: 10px;">ОСОБОВИЙ СКЛАД</div>
                <div id="detPersonnel" style="font-size: 18px; font-weight: bold; color: var(--
blue);"></div>
            </div>
            <div class="detail-item-box">
              <div style="font-size: 10px;">БОЄПРИПАСИ</div>
              <div id="detAmmo" style="font-size: 18px; font-weight: bold;"></div>
            </div>
            <div class="detail-item-box">
              <div style="font-size: 10px;">ПАЛИВО</div>
              <div id="detFuel" style="font-size: 18px; font-weight: bold;"></div>
            </div>
            <div class="detail-item-box">
              <div style="font-size: 10px;">ТРАНСПОРТ</div>
              <div id="detVehicle" style="font-size: 18px; font-weight: bold;"></div>
            </div>
          </div>
                    <button class="action-btn" style="margin-top: 25px; width: 100%;"
onclick="window.print()">ЕКСПОРТ ЗВІТУ</button>
      </div>
    </div>

    <script>
      const unitsData = {};
      let mapInstance;

      window.onload = function() {
        initChart();
        generateUnits();
        generateAudit();
        initLeafletMap();
      };

      function initChart() {
```

```
const ctx = document.getElementById('mainChart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['05.02', '06.02', '07.02', '08.02', '09.02', '10.02', '11.02'],
        datasets: [{
            label: 'Постачання',
            data: [450, 380, 520, 410, 600, 580, 620],
            borderColor: '#2f81f7',
            backgroundColor: 'rgba(47, 129, 247, 0.1)',
            fill: true,
            tension: 0.4,
            pointRadius: 4,
            pointBackgroundColor: '#2f81f7'
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: true,
        aspectRatio: 2.5,
        plugins: {

            legend: {
                display: false
            }
        },
        scales: {
            y: {
                grid: { color: '#30363d' },
                ticks: { color: '#8b949e', font: { size: 10 } }
            },
            x: {
                grid: { display: false },
                ticks: { color: '#8b949e', font: { size: 10 } }
```

```
            }
        }
    }
});
}

    function generateUnits() {
        const container = document.getElementById('unit-cards-container');
        for(let i=1; i<=22; i++) {
            const ammo = 65 + Math.floor(Math.random() * 36);
            const fuel = 60 + Math.floor(Math.random() * 41);
            const vehicle = 65 + Math.floor(Math.random() * 36);
            const currentPers = 115 + Math.floor(Math.random() * 16);
            const readiness = Math.round((ammo + fuel + vehicle + (currentPers/130*100)) / 4);

           unitsData[i] = { id: i, ammo, fuel, vehicle, personnel: `${currentPers} / 130`, readiness };

            container.innerHTML += `
                <div class="unit-item" style="text-align: center;">
                    <div style="color: var(--blue); font-weight: bold;">Б-н №${i}</div>
                    <div style="font-size: 20px; font-weight: bold;">${readiness}%</div>
                  <div class="status-badge ${readiness > 85 ? 'status-ok' : 'status-warn'}">${readiness
> 85 ? 'готовий' : 'обмежено'}</div>
                    <button class="action-btn" onclick="openDetails(${i})" style="margin-top: 15px;
width: 100%;">ДЕТАЛІ</button>
                </div>`;
        }
    }

    function initLeafletMap() {
  mapInstance = L.map('map-container').setView([50.45, 30.52], 10);
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(mapInstance);

  for(let i=1; i<=22; i++) {
```

```
        const lat = 50.3 + (Math.random() * 0.3);
        const lng = 30.3 + (Math.random() * 0.5);
        const data = unitsData[i];

        const color = data.readiness > 85 ? '#3fb950' : '#d29922';
        const marker = L.circleMarker([lat, lng], {
            radius: 7,
            fillColor: color,
            color: "#fff",
            weight: 1,
            opacity: 1,
            fillOpacity: 0.9
        }).addTo(mapInstance);
            marker.bindTooltip(`Б-н №${i}`, {
            permanent: true,
            direction: 'top',
            offset: [0, -5],
            className: 'map-label'
        });


        marker.bindPopup(`
            <div style="color: #000; font-family: sans-serif;">
                <b>Батальйон №${i}</b><br>
                Готовність: ${data.readiness}%<br>
                <hr style="margin: 5px 0; border: 0; border-top: 1px solid #ccc;">
                    <button onclick="openDetails(${i})" style="background:#2f81f7; border:none;
color:white; padding:4px 8px; cursor:pointer; width:100%; border-radius:4px;">Відкрити
звіт</button>
            </div>
        `);
    }
}
```

```
    function openDetails(id) {
        const data = unitsData[id];
        document.getElementById('mTitle').innerText = `ЗВІТ: БАТАЛЬЙОН №${id}`;
            document.getElementById('mStatus').innerText = `РІВЕНЬ ГОТОВНОСТІ: ${data.readiness}%`;
        document.getElementById('detPersonnel').innerText = data.personnel;
        document.getElementById('detAmmo').innerText = data.ammo + '%';
        document.getElementById('detFuel').innerText = data.fuel + '%';
        document.getElementById('detVehicle').innerText = data.vehicle + '%';
        document.getElementById('unitModal').style.display = 'flex';
    }

    function closeModal() { document.getElementById('unitModal').style.display = 'none'; }

    function switchTab(tabId, btn) {
        document.querySelectorAll('.tab-content').forEach(s => s.style.display = 'none');
        document.getElementById(tabId).style.display = 'block';
        document.querySelectorAll('.nav-btn').forEach(b => b.classList.remove('active'));
        btn.classList.add('active');
        if(tabId === 'map' && mapInstance) {
            setTimeout(() => mapInstance.invalidateSize(), 200);
        }
    }

    function generateAudit() {
        const auditBody = document.getElementById('audit-table-body');
        const items = ['АК-74М', 'Бронежилет кл.4', 'Рація Motorola', 'Набої 5.45', 'Тепловізор AGM'];
        for(let i=1; i<=20; i++) {
            auditBody.innerHTML += `<tr><td>0${(i%9)+1}.02.2026</td><td>Б-н №${(i%22)+1}</td><td><b style="color:var(--blue)">${items[i%5]}</b></td><td>${Math.floor(Math.random()*100)+10} од.</td><td>майор Шевченко</td></tr>`;
        }
    }
```

```
    </script>
</body>
</html>
```