НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____Ігор БОЛБОТ
(підпис)

«___» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ Белла ГОЛУБ
(підпис)

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення інформаційно-управляючої системи для Станції Технічного Обслуговування»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ (підпис) **Ганна ВАЙГАНГ**

Керівник бакалаврської кваліфікаційної роботи
науковий ступінь та вчене звання _____ (підпис) **Тарас ЛЕНДЄЛ**

Виконав _____ (підпис) **Артем КОМЕНДА**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

к.т.н.,доцент _____ Белла ГОЛУБ

(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Коменді Артему Вадимовичу

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення інформаційно-управляючої системи для Станції Технічного Обслуговування »

затверджена наказом від «19» грудня 2025 р. № 3204 «С»

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

Вихідні дані до бакалаврської кваліфікаційної роботи:

_____

Перелік питань, що підлягають дослідженню:

_____

Перелік графічних документів (за необхідності).

_____

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Тарас ЛЕНДЄЛ**

Завдання взяв до виконання _____ (підпис) **Артем КОМЕНДА**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

СТО – станція технічного обслуговування

СУБД – система управління базами даних

.NET – безкоштовна та з відкритим вихідним кодом платформа для розробки програмного забезпечення

ACID – це набір властивостей, що гарантують надійну роботу транзакцій бази даних

C# – об’єктно-орієнтована мова програмування створена Microsoft

DirectX – це набір компонентів у Windows, які дають змогу програмному забезпеченню працювати безпосередньо з відео- та аудіопристроями

HDD – Hard Disk Drive

MVVM – Model-View-ViewModel, один з MV патернів (архітектурний шаблон у WPF)

ORM – Object-Relational Mapping (об’єктно-реляційне відображення)

PDF – Portable Document Format (переносний формат документів)

SKU – Stock Keeping Unit (складський артикул)

SQL – це декларативна мова запитів для реляційних СУБД

SSD – Solid State Drive

UI – User Interface (інтерфейс користувача)

UML – уніфікована мова моделювання

WMS – система управління складом

WPF – Windows Presentation Foundation (технологія побудови UI в середовищі .NET)

XAML – декларативна мова розмітки

АРМ – автоматизоване робоче місце

БД – база даних

ВСТУП

В умовах інтенсивного розвитку автомобільної галузі та перманентного зростання автотранспортного парку України особливої актуальності набуває вдосконалення процесів технічного обслуговування та ремонту. Станції технічного обслуговування (СТО) є ключовими суб’єктами забезпечення надійності та експлуатаційної безпеки транспортних засобів, надаючи комплексний спектр послуг — від превентивної діагностики до капітального відновлення функціональних систем автомобілів.

Фундаментальним чинником ефективного функціонування СТО є раціональна організація складського господарства. Дана структура забезпечує безперебійність технологічних процесів шляхом своєчасної дистрибуції необхідних компонентів, витратних матеріалів та інструментарію. Рівень оптимізації управління складськими операціями безпосередньо детермінує часові показники виконання ремонтних робіт, якість сервісу та загальні параметри рентабельності підприємства.

Актуальність цієї теми зумовлена тим, що в сучасних умовах ринку автосервісних послуг особливого значення набувають оперативність і якість обслуговування клієнтів. Вони напряму залежать від того, наскільки швидко та в повному обсязі на складі доступні необхідні запасні частини, витратні матеріали та інструменти. Ефективно організоване управління складом дає змогу зменшити час очікування клієнтів, уникнути як надлишкових запасів, так і дефіциту потрібних комплектуючих, що безпосередньо впливає на репутацію СТО та рівень лояльності клієнтів.

Застарілі підходи до ведення складського обліку на автосервісних підприємствах, які базуються на паперовій документації або простих електронних таблицях, уже не відповідають вимогам сучасного бізнесу. Вони є недостатньо швидкими, менш зручними та більш схильними до помилок. У зв’язку з цим розробка

спеціалізованого програмного забезпечення для автоматизації робочого місця завідувача складу СТО є актуальним і необхідним завданням.

Метою розробки є створення програмного забезпечення автоматизованого робочого місця (АРМ) для управління складом станції технічного обслуговування, яке дозволить підвищити ефективність обліку та контролю складських операцій, оптимізувати рівень запасів і покращити якість обслуговування клієнтів СТО.

Проектоване програмне забезпечення спрямоване на вирішення комплексу завдань щодо автоматизації ключових складських операцій, зокрема: обліку надходження та видачі товарно-матеріальних цінностей, моніторингу залишків, а також генерації аналітичної звітності. Пріоритетними аспектами розробки є створення ергономічного інтерфейсу користувача та забезпечення високого рівня цілісності й відмовостійкості бази даних, що є критичним для стабільного функціонування станції технічного обслуговування.

Структура роботи відповідає логічній послідовності дослідження та реалізації проекту.

У першому розділі проаналізовано предметну область, виконано порівняльний огляд існуючих аналогів та сформульовано технічне завдання на розробку.

Другий розділ присвячено концептуальному моделюванню системи. Виконано візуалізацію функціональних вимог і динаміки процесів через побудову діаграм прецедентів, послідовності, активності та класів.

У третьому розділі викладено результати проектування архітектури системи. Розроблено логічну та фізичну моделі даних, а також наведено обґрунтування вибору технологічного стека та інструментарію розробки.

Четвертий розділ охоплює питання практичного впровадження та експлуатації системи. Описано технічні вимоги та наведено результати верифікації й тестування якості програмного продукту.

РОЗДІЛ І.
СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Опис предметної області

Станція технічного обслуговування (СТО) є суб’єктом господарювання, що здійснює комплексне надання послуг із підтримання належного технічного стану та відновлення працездатності транспортних засобів. Провідне місце в інфраструктурі підприємства посідає складське господарство, яке виступає гарантом безперебійного функціонування виробничих підрозділів.

Діяльність складського господарства спрямована на акумулювання, систематизацію та оперативну дистрибуцію запасних частин, витратних матеріалів і технічного інструментарію. Раціоналізація складської логістики сприяє мінімізації латентних простоїв автотранспорту в процесі обслуговування, що корелює з підвищенням загальної якості сервісу.

Просторова структура типового складу СТО базується на принципі функціонального зонування з урахуванням специфіки зберігання різних категорій активів. Проектування передбачає виокремлення ділянок приймання продукції, зберігання вузлів і агрегатів, паливно-мастильних матеріалів, шинокомплектів, інструментарію, а також зони відвантаження. Такий підхід дозволяє оптимізувати внутрішні логістичні потоки та дотримуватися встановлених режимів зберігання для кожної номенклатурної групи.

Асортиментний перелік складських запасів охоплює компоненти ключових систем автомобіля (силових агрегатів, трансмісії, ходової частини), регламентні витратні матеріали (фільтруючі елементи, технічні рідини, засоби автохімії), а також супутнє обладнання та аксесуари.

Механізм управління складом СТО включає низку ітераційних процесів. Стадія закупівлі та приймання базується на детермінації потреб, селекції постачальників і

вхідному контролі якості товарно-матеріальних цінностей. Етап зберігання передбачає організацію раціонального розміщення запасів із забезпеченням їхньої фізичної цілісності.

Ключовим аспектом є облікова політика, що реалізується через документування операцій, проведення періодичних інвентаризацій, моніторинг складських залишків і формування звітної документації. Процес видачі активів ініціюється внутрішніми заявками виробничих ділянок і завершується комплектуванням замовлень із відповідним документальним супроводом.

Впровадження спеціалізованих інформаційних систем дозволяє автоматизувати базові алгоритми управління запасами. Використання таких рішень забезпечує прецизійність обліку, оптимізацію закупівельної діяльності та формування аналітичного базису для прийняття управлінських рішень. Інтеграція складського модуля в загальну інформаційну архітектуру СТО є необхідною умовою підвищення загальної операційної ефективності підприємства.

1.2 Аналіз вимог до програмної системи

Функціональні вимоги до програмного забезпечення автоматизованого робочого місця (АРМ) управління складом станції технічного обслуговування охоплюють сукупність ключових аспектів, що забезпечують ефективну експлуатацію системи. Першочерговим завданням є ведення верифікованого каталогу запасних частин для здійснення оперативного пошуку та навігації. Крім того, система має реалізувати механізми обліку поточних складських залишків та реєстрації надходжень нових партій товарів з метою оптимізації управління запасами.

Додатково функціональні вимоги передбачають можливість опрацювання електронних запитів від технічного персоналу та фіксацію видачі комплектуючих. Це сприяє вдосконаленню координації роботи автосервісу та забезпечує транспарентність технологічних процесів. Важливою умовою є імплементація

системи розмежування прав доступу для різних категорій користувачів, що підвищує рівень безпеки даних.

У контексті компонентної структури кожен елемент системи має відповідати визначеним функціональним вимогам. Зокрема, компонент ведення каталогу повинен забезпечувати ергономічний пошук, а модуль обліку надходжень — коректну реєстрацію нових партій ТМЦ. Проектування кожного компонента спрямоване на досягнення високої ефективності та надійності системи в цілому.

У таблиці 1.1 наведено специфікацію функціональних вимог до автоматизованої системи складу СТО.

Таблиця 1.1

Специфікація функціональних вимог

№	Вимоги	Призначення та використання
1	Ведення каталогу запчастин	Система повинна забезпечувати створення, збереження та навігацію по каталогу запчастин із можливістю пошуку за назвою, артикулом, категорією та виробником.
2	Відображення поточних залишків на складі	Система повинна надавати актуальну інформацію про кількість запчастин у наявності, оновлюючи дані в режимі реального часу для оперативного реагування на запити.
3	Облік надходження партій запчастин	Система повинна дозволяти реєструвати надходження нових партій запчастин із можливістю додавання через інтерфейс.
4	Прийом електронних запитів від автомеханіків	Система повинна приймати заявки від автомеханіків через інтерфейс, автоматизувати перевірку наявності та генерувати повідомлення про статус виконання запиту.
5	Реєстрація видачі запчастин співробітнику	Система повинна фіксувати факт передачі запчастини: кому, коли і для яких робіт видано, формувати видаткові накладні та вести історію видач.
6	Розмежування прав доступу	Система повинна підтримувати ролі користувачів (оператор, менеджер, адміністратор).
7	Редагування та доповнення каталогу запчастин	Система повинна дозволяти уповноваженим користувачам додавати нові позиції, редагувати характеристики існуючих запчастин та підтримувати історію змін.

8	Облік переміщення та використання запчастин	Система повинна автоматично фіксувати переміщення запчастин між зонами зберігання, а також їхнє використання, із вказанням дати, часу та відповідальних осіб.

Нефункціональні вимоги до системи обліку запчастин на автосервісі (табл. 1.2) визначають параметри та властивості системи, які не прямо пов'язані з її функціональністю, але визначають її якість, надійність, безпеку та інші аспекти, що впливають на задоволення потреб користувачів.

Таблиця 1.2

Специфікація нефункціональних вимог

№	Категорія вимог	Вимоги	Призначення та використання
1	Безпека	Автентифікація	Система повинна підтримувати автентифікацію для різних користувачів.
2		Шифрування паролів	Система повинна шифрувати паролі користувачів перед збереженням у базі даних, використовуючи сучасні алгоритми.
3	Інтерфейс	Простий інтерфейс	Інтерфейс повинен бути простим та зрозумілим для користувачів, що дозволяє швидко виконувати необхідні дії без складних налаштувань.
4	Сумісність	Операційні системи	Програмне забезпечення повинно бути сумісне з операційними системами Windows 10 і вище.
5	Продуктивність	Оптимізація запитів до БД	Запити до бази даних повинні бути оптимізовані для забезпечення швидкої обробки даних, без затримок при великих обсягах інформації.
6		Робота з великими обсягами даних	Система повинна ефективно працювати з великими обсягами даних, забезпечуючи швидке завантаження, пошук та обробку даних без значних затримок.

Однією з ключових нефункціональних вимог є інформаційна безпека. Система повинна гарантувати конфіденційність, цілісність та доступність відомостей про

номенклатурні одиниці й складські операції, а також забезпечувати захист від несанкціонованого доступу. Заходи безпеки передбачають надійну автентифікацію користувачів та криптографічний захист чутливих даних.

Важливою нефункціональною вимогою є ергономічність інтерфейсу. Програмне забезпечення має характеризуватися інтуїтивно зрозумілим графічним середовищем, що дозволяє користувачам оперативно виконувати функціональні завдання без необхідності складного конфігурування чи тривалої спеціалізованої підготовки.

Суттєве значення має продуктивність системи, яка повинна забезпечувати високу швидкість обробки запитів та мінімальний час відгуку. Це досягається шляхом оптимізації взаємодії з базою даних та забезпечення ефективного опрацювання великих масивів інформації без суттєвих затримок.

Крім того, система має бути сумісною з операційними системами родини Windows (версії 10 та вище), що дозволяє її інтеграцію в існуючу технічну інфраструктуру автосервісу.

Сукупність зазначених функціональних та нефункціональних вимог детермінує базові можливості та експлуатаційні характеристики системи складського обліку. Системні вимоги є критично важливими для забезпечення стабільного функціонування програмного продукту та його відповідності стандартам безпеки, доступності й ефективності.

1.3 Моделювання предметної області

Моделювання предметної області є детермінуючим етапом життєвого циклу розробки програмного забезпечення. Даний процес передбачає ідентифікацію базових сутностей та встановлення інтерреляцій між ними, що формує фундамент для подальшого проектування системи. Аналіз ключових об'єктів та їхніх взаємодій у межах предметної області дозволяє верифікувати основні бізнес-процеси, що реалізуються в системі.

Мова UML (Unified Modeling Language) виступає універсальним стандартом візуального моделювання, призначеним для специфікації, візуалізації, проектування та документування архітектури програмного забезпечення, бізнес-процесів та інших системних структур. Як потужний інструментарій моделювання, UML забезпечує створення цілісних концептуальних та логічних моделей складних систем. Застосування даної мови дозволяє розробникам чітко дефінувати структуру та динаміку поведінки системи, що оптимізує комунікацію всередині проектної групи та створює уніфікований базис для подальшої розробки й супроводу програмного продукту.

З метою деталізації структури та операційної логіки складського підрозділу СТО було розроблено діаграму прецедентів, яка візуалізує основних акторів та їхні функціональні зв’язки з системою. Центральним елементом взаємодії є інформаційна система управління складом, що підтримує сукупність ключових сценаріїв для різних категорій користувачів: оператора складу, менеджера складу, автомеханіка та постачальника комплектуючих.

Рис.1.1 Діаграма прецедентів складу СТО

Функціональні обов’язки оператора складу охоплюють щоденний операційний цикл. До переліку його повноважень належать:

- модифікація даних про запчастини, що передбачає внесення та редагування записів у довіднику номенклатури згідно з актуальним асортиментом;
- списання товарно-матеріальних цінностей у разі їхнього пошкодження, закінчення терміну придатності або цільового використання для внутрішніх потреб підприємства;
- оприбуткування партій запчастин від контрагентів із верифікацією кількісних і якісних показників, дати надходження та супровідної документації;

- моніторинг наявних партій товарів на складі для аналізу їхньої генези та залишкового ресурсу;
- дистрибуція запчастин для проведення ремонтних робіт, що реалізується на підставі верифікованих запитів від технічного персоналу.

Автомеханік, як кінцевий споживач матеріальних ресурсів у межах виробничого циклу, наділений такими функціями:

- візуалізація актуальних залишків на складі для оперативного контролю наявності необхідних компонентів;
- генерація запитів на отримання запчастин для подальшого опрацювання складською службою;
- здійснення ремонтних операцій на основі отриманих активів, що є результуючим етапом складської логістики.

Менеджер складу реалізує функції стратегічного та контрольного спрямування:

- верифікація заявок на постачання, сформованих операційним персоналом або автоматизованими алгоритмами системи;
- ініціювання нових закупівельних процедур у разі ідентифікації дефіциту ресурсів;
- підготовка аналітичної та статистичної звітності щодо функціонування складу.

Постачальник запчастин виступає як зовнішній суб’єкт, взаємодія з яким детермінує вхідний потік матеріальних цінностей відповідно до узгоджених заявок.

Отже, діаграма прецедентів відображає системні інтерреляції між внутрішніми підрозділами СТО та зовнішніми контрагентами, чітко регламентуючи зони відповідальності персоналу. Така структуризація є підґрунтям для формування технічних вимог до інформаційної системи, покликаної автоматизувати управління запасами та забезпечити єдине комунікаційне середовище.

Для деталізації динаміки процесів складського підрозділу в часовому вимірі було розроблено діаграму послідовності (рис. 1.2), яка візуалізує алгоритм взаємодії акторів у процесі видачі запчастин.

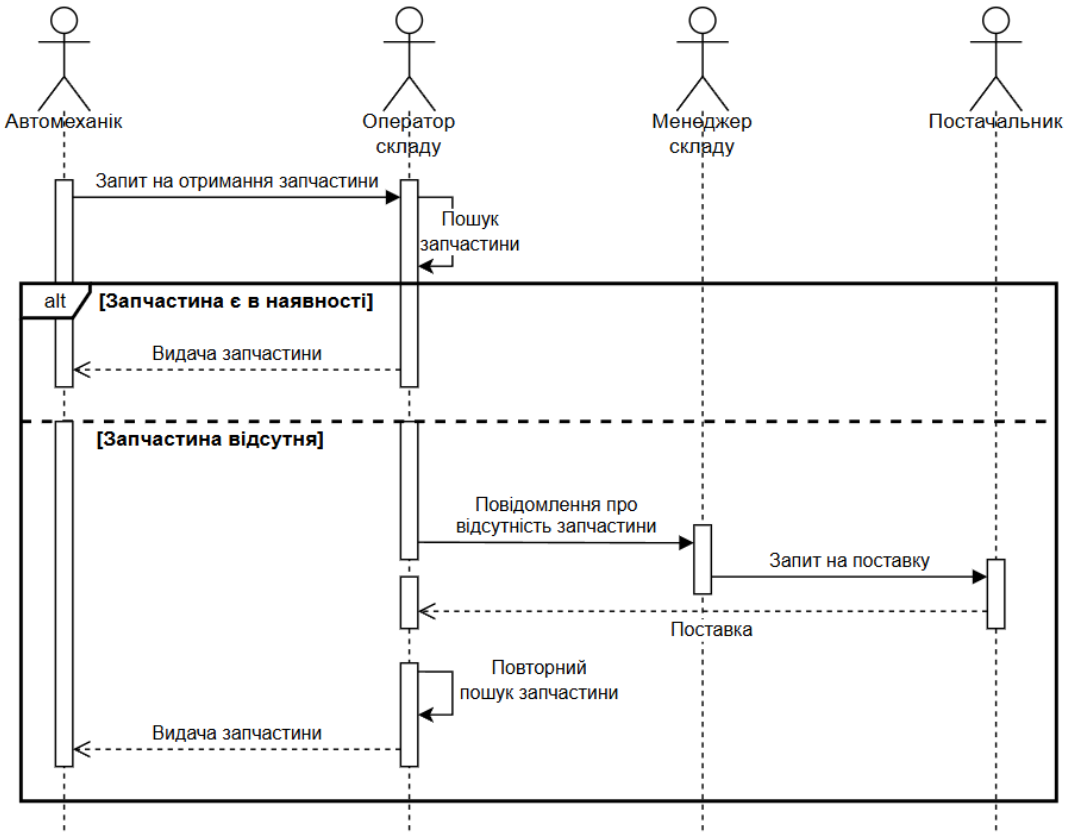


Рис.1.2 Діаграма послідовності складу СТО

Діаграма відображає два основні варіанти розвитку процесу. Якщо необхідна запчастина є на складі, оператор здійснює її видачу автомеханіку, і процес завершується успішно. У разі відсутності запчастини запускається альтернативний сценарій: оператор повідомляє менеджера складу про нестачу потрібної позиції.

Після отримання інформації менеджер формує запит на постачання та надсилає його постачальнику, який виконує доставку необхідної запчастини. Після надходження товару на склад оператор повторно перевіряє наявність і здійснює видачу запчастини автомеханіку.

Таким чином, діаграма послідовності наочно показує логіку роботи складської системи СТО, порядок виконання операцій та взаємодію між усіма учасниками процесу. Вона дозволяє не лише побачити основний сценарій роботи, але й альтернативний варіант у випадку дефіциту запчастин. Це сприяє кращому розумінню процесів, оптимізації логістики, скороченню часу обслуговування клієнтів і підвищенню загальної ефективності роботи СТО.

Для відображення бізнес-процесу видачі запчастин на СТО була створена діаграма активності (рис. 1.3), яка демонструє послідовність дій та взаємодію між ключовими учасниками процесу: автомеханіком, оператором складу, менеджером складу та постачальником.

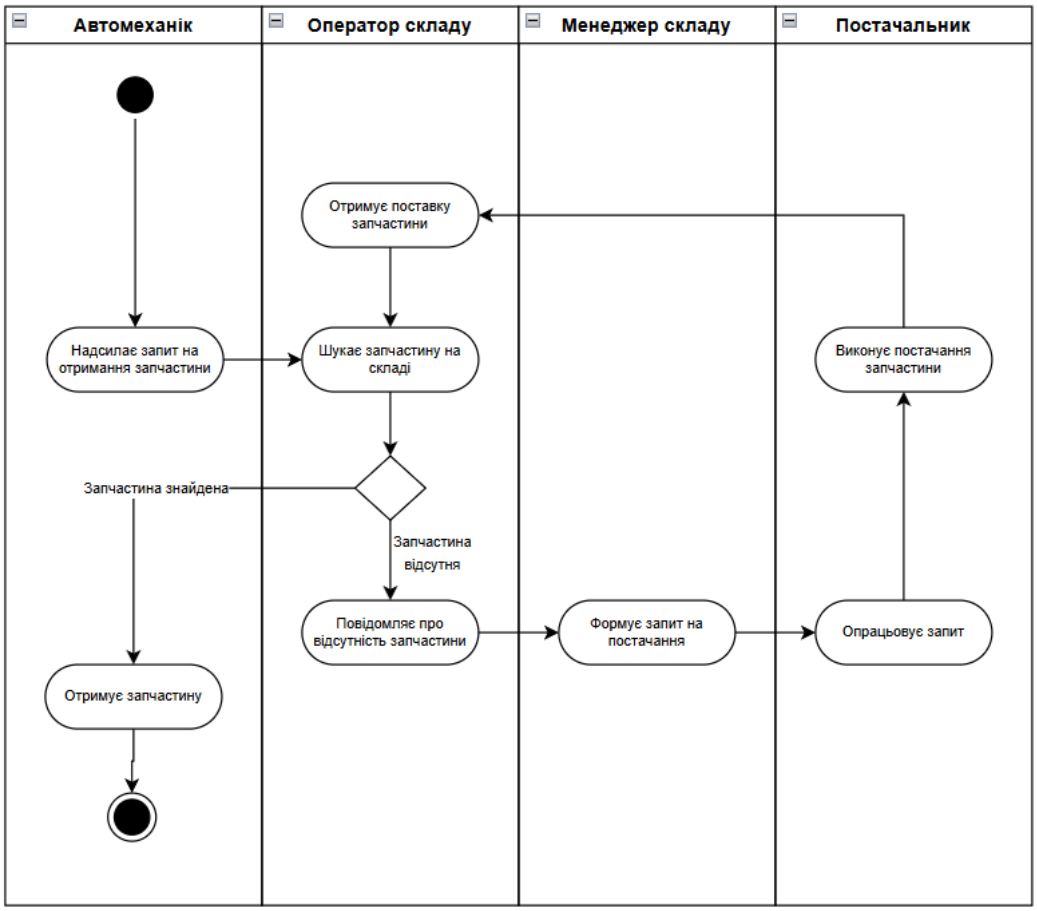


Рис.1.3 Діаграма активності складу СТО

Процес починається з того, що автомеханік формує та надсилає запит на отримання необхідної запчастини. Після цього оператор складу отримує запит і перевіряє її наявність у складських залишках. Якщо потрібна запчастина є в наявності, оператор одразу видає її автомеханіку, після чого процес завершується.

У випадку відсутності запчастини на складі оператор інформує про це менеджера складу. Менеджер, отримавши відповідне повідомлення, формує запит на постачання та надсилає його постачальнику. Після опрацювання запиту постачальник здійснює доставку необхідної запчастини на склад. Далі оператор приймає поставку та повторно виконує видачу запчастини автомеханіку.

Таким чином, діаграма активності відображає як основний сценарій виконання запиту, так і альтернативний варіант розвитку подій у разі відсутності необхідних матеріалів. Така модель дозволяє краще проаналізувати внутрішні логістичні процеси, зменшити час очікування та забезпечити стабільну й безперебійну роботу СТО.

1.4 Огляд інформаційних джерел та існуючих рішень

Етап передінвестиційного аналізу, що включає критичний огляд існуючих інформаційних джерел та аналітику релевантних ринкових рішень, є детермінуючим для успішної реалізації проекту розробки програмного забезпечення. Отримані дані формують концептуальний базис для проектування нової системи або модернізації існуючих рішень. Проведемо порівняльний аналіз програмних продуктів у сегменті систем управління складом (WMS).

Огляд існуючих рішень дозволяє ідентифікувати актуальні технологічні тренди та методологічні підходи до автоматизації складської логістики на підприємствах автосервісу.

Tocan WMS є багатофункціональним програмним комплексом, призначеним для автоматизації повного циклу складських операцій. Система підтримує гібридні моделі розгортання (SaaS та On-premise), що забезпечує масштабованість для бізнесу різного

рівня. Функціональний діапазон охоплює процеси приймання, адресної дистрибуції та відвантаження ТМЦ. Використання 2D/3D-конструкторів дозволяє оптимізувати складську топологію, а підтримка інтеграції з ERP-системами (1С, SAP, MS Dynamics) забезпечує цілісність інформаційного простору підприємства. Модуль аналітики на основі алгоритмів машинного навчання дозволяє розраховувати KPI персоналу та оптимізувати логістичні маршрути.

Поряд із широким функціоналом, Tocan WMS характеризується високою вартістю впровадження та складністю конфігурування візуальних моделей складу, що потребує спеціалізованої підготовки персоналу. Також спостерігаються технічні дефіцити при інтеграції із застарілими системами та обмежена продуктивність мобільних клієнтів на пристроях із низькими апаратними характеристиками.

LSC WMS позиціонується як високотехнологічне рішення для комплексної автоматизації логістики в секторах дистрибуції та виробництва. Система орієнтована на високий рівень механізації, підтримуючи інтеграцію з роботизованими комплексами та автоматизованими складськими системами (AS/RS). Характерною особливістю є впровадження технології голосового керування (voice-picking), що мінімізує помилки за рахунок виключення ручного введення даних. Інтелектуальний модуль прогнозування ресурсів дозволяє автоматично розподіляти навантаження на персонал на основі ретроспективного аналізу даних.

До критичних недоліків LSC WMS належать функціональна надмірність для малих складських комплексів та висока сукупна вартість володіння, зумовлена вимогами до серверної інфраструктури. Користувачі зазначають тривалий період адаптації до інноваційних методів відбору та інертність технічної підтримки при вирішенні специфічних запитів.

Проведений аналіз демонструє необхідність розробки спеціалізованого, економічно ефективного та інтуїтивно зрозумілого рішення, адаптованого до специфічних потреб складського господарства СТО.

1.5 Постановка завдання

Програмне забезпечення автоматизованого робочого місця (АРМ) призначене для забезпечення ефективного управління складською діяльністю станції технічного обслуговування (СТО). СТО є підприємством, що виконує технічне обслуговування та ремонт транспортних засобів, тому її продуктивність значною мірою залежить від правильно організованих процесів постачання, зберігання та обліку запасних частин, витратних матеріалів, інструментів і обладнання. Основною функцією складського господарства є безперебійне забезпечення ремонтних дільниць усіма необхідними ресурсами, що дозволяє скорочувати час простою автомобілів і підвищувати загальний рівень якості сервісу.

Розроблюване програмне рішення реалізується як десктопний застосунок для операційної системи Windows. Воно охоплює повний цикл складських процесів: приймання товарів від постачальників, їх розміщення з урахуванням зонування складу, ведення обліку, проведення інвентаризацій, формування звітності, а також видачу запчастин автомеханікам відповідно до їхніх заявок. Такий підхід дозволяє централізувати управління складом і мінімізувати ймовірність помилок у процесі обліку.

Для підвищення надійності системи та захисту даних від втрати передбачено механізм автоматичного резервного копіювання бази даних. Резервні копії створюються регулярно за встановленим розкладом і зберігаються у захищеному каталозі на локальному або зовнішньому носії. У разі виникнення збоїв або пошкодження основної бази даних адміністратор має можливість швидко відновити інформацію з останньої актуальної копії, що суттєво зменшує ризики простою системи та втрати даних.

Інтерфейс застосунку розробляється з акцентом на простоту, інтуїтивність та зручність використання. Основні функціональні можливості згруповані у вкладки відповідно до ролей користувачів, що забезпечує швидку навігацію навіть для нових працівників. Дизайн інтерфейсу мінімалістичний і структурований, містить логічно організовані форми, таблиці та елементи керування. Усі поля супроводжуються

підказками, що зменшує кількість помилок і сприяє швидкому опануванню системи персоналом.

У системі передбачено три основні ролі користувачів: оператор, менеджер та автомеханік, кожна з яких має визначений набір прав і обов’язків.

Оператор відповідає за повний цикл складського обліку. Його функції включають введення нових надходжень товарів, облік поставок від постачальників, оформлення видачі запчастин на основі заявок, а також постійний контроль залишків у реальному часі. Крім того, оператор може редагувати довідник номенклатури (назва, артикул, категорія, виробник тощо) та здійснювати облік переміщення товарів між складськими зонами.

Менеджер працює з аналітичною та звітною інформацією. Він має доступ до перегляду залишків, аналізу історії операцій, формування звітів за різними критеріями та оцінювання ефективності роботи складських процесів. При цьому менеджер не змінює дані, а використовує їх для прийняття управлінських рішень.

Автомеханік є користувачем, який формує запити на видачу необхідних запчастин для виконання ремонтних робіт. Він може переглядати інформацію про наявні товари, їх характеристики та кількість на складі, після чого створює відповідні запити. Його права обмежуються лише переглядом даних і формуванням заявок без можливості редагування інформації в системі.

Загалом запропонована система дозволяє впорядкувати складські процеси СТО, підвищити точність обліку, прискорити обробку запитів та покращити ефективність роботи всього підприємства.

РОЗДІЛ ІІ.
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Модель «сутність-зв’язок» (ER-модель) являє собою концептуальну схему даних, що базується на використанні уніфікованих структурних блоків. Серед різноманітних методів представлення знань ER-модель виділяється як найбільш універсальний інструмент дескрипції даних, що не залежить від конкретної програмної реалізації. Фундаментальна значущість цієї моделі полягає в її здатності трансформуватися у будь-які існуючі структури даних — ієрархічні, мережеві, реляційні або об’єктні.

ER-моделювання є результатом системної формалізації певної предметної області. Модель не регламентує динаміку процесів, а лише візуалізує статичну архітектуру даних. Інформація структурується у вигляді сутностей, поєднаних детермінованими зв’язками, що відображають логічні залежності та обмеження (наприклад, відношення типу «один-до-багатьох»). Кожна сутність характеризується набором атрибутів, що описують її властивості. Графічна інтерпретація таких структур називається діаграмою «сутність-зв’язок».

Практична імплементація ER-моделі найчастіше здійснюється у форматі реляційних баз даних. У такій системі сутність трансформується в таблицю, де окремий рядок відповідає конкретному екземпляру сутності. Взаємозв’язки між об’єктами реалізуються за допомогою покажчиків (зовнішніх ключів), що посилаються на індекси в інших таблицях.

Термін «сутність» позначає абстракцію об’єкта, явища або процесу реального світу, що має значення для системи. Необхідно розмежовувати поняття «тип сутності» та «екземпляр сутності». Тип сутності визначає категорію однорідних об’єктів

(наприклад, МІСТО), тоді як екземпляр є конкретним представником цієї категорії (наприклад, Київ).

Сутність розглядається як ключовий об’єкт предметної області, інформація про який підлягає акумуляції та зберіганню. Кожен тип сутності володіє унікальним ідентифікатором, що дозволяє диференціювати будь-який екземпляр у межах даної множини.

Концептуальна ER-модель для системи управління складським господарством СТО представлена на рис. 2.1.

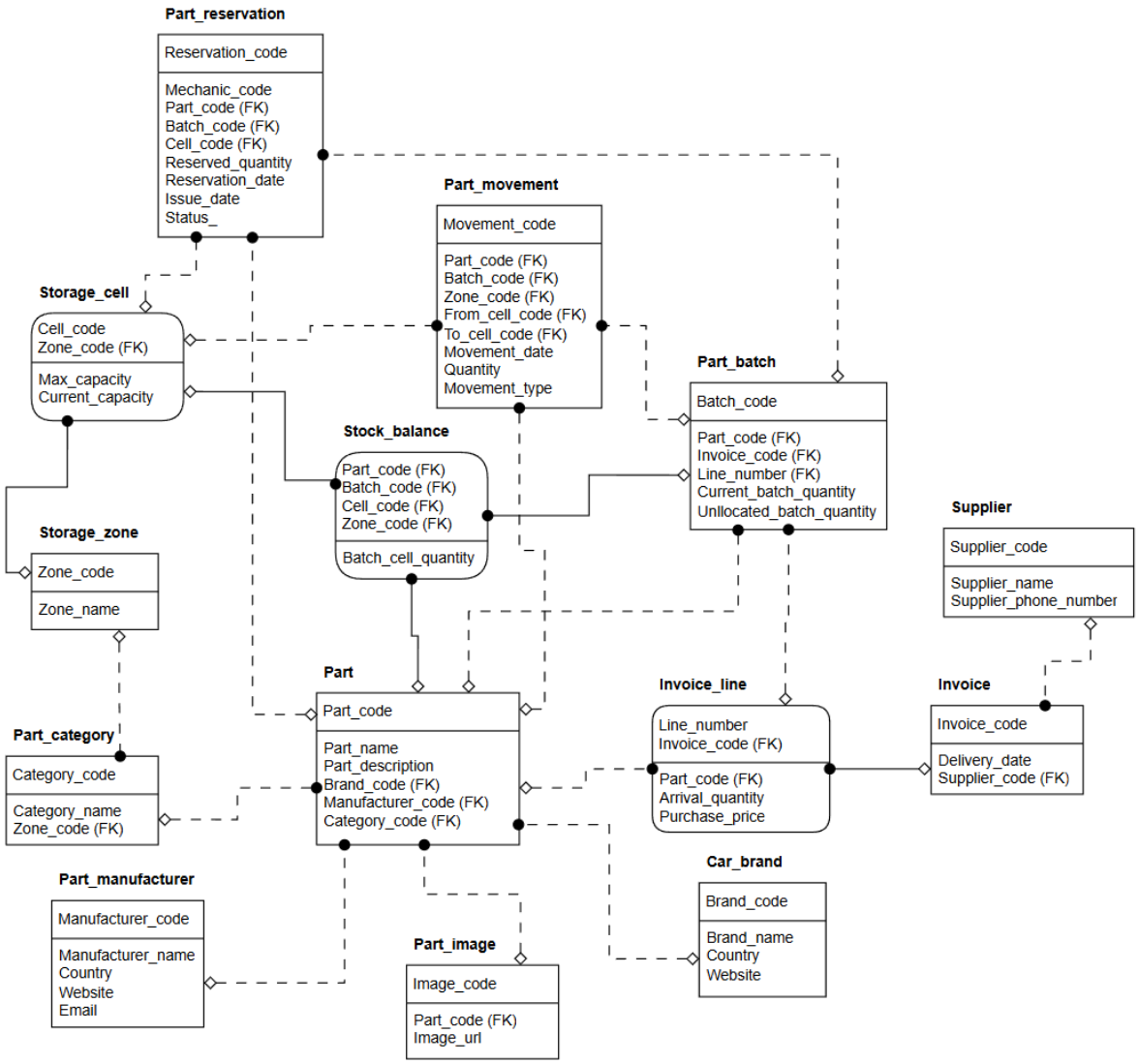


Рис.2.1 ER-діаграма додатку

В табл. 2.1 описано кожну сутніть ER-діаграми додатку.

Таблиця 2.1

Опис сутностей ER-діаграми

№	Сутність	Опис
1	Part	Основна сутність – деталь. Містить назву, опис, бренд, категорію та виробника
2	Part_batch	Партія деталей. Зв’язана з поставкою. Містить кількість деталей у партії і кількість ще не розподілених деталей
3	Part_reservation	Резервація деталі для механіка. Містить кількість, дату бронювання та статус
4	Part_movement	Переміщення деталей між комірками. Вказує звідки/куди, дату, тип руху та кількість
5	Stock_balance	Залишок деталей у комірці для певної партії
6	Storage_cell	Комірка зберігання з максимальною і поточною місткістю
7	Storage_zone	Зона складу, до якої належать комірки
8	Invoice	Документ поставки. Зв’язаний з постачальником та містить дату
9	Invoice_line	Рядок поставки: яка деталь, у якій кількості та за якою ціною була доставлена
10	Supplier	Постачальник деталей
11	Part_category	Категорія деталі (наприклад, двигуни, фільтри), може бути прив’язана до зони
12	Part_manufacturer	Виробник деталі. Має назву, країну, сайт, email
13	Car_brand	Автомобільний бренд, з яким сумісна деталь
14	Part_image	Зображення деталі, зберігається URL. Кожне зображення прив’язане до конкретної деталі

Таким чином, побудована ER-діаграма охоплює всі ключові сутності та взаємозв’язки, що формують основу інформаційної структури системи. Такий підхід дозволяє забезпечити цілісність і узгодженість даних у базі, а також створює надійну базу для розробки механізмів обліку, резервування та переміщення запчастин у межах складу СТО. Визначені сутності охоплюють як логістику зберігання, так і взаємодію

з постачальниками, користувачами та супровідною документацією, що є критично важливим для ефективного функціонування всієї системи.

2.2 Діаграма класів та кооперації

Діаграма класів є статичною структурною UML-діаграмою, яка відображає основні елементи системи — класи, їхні властивості, операції та взаємозв'язки між ними. Вона використовується для опису структури системи в термінах об'єктно-орієнтованого програмування та дозволяє показати як самі класи, так і інтерфейси, об'єкти та кооперації разом із їхніми зв'язками.

На діаграмі кожен клас зображується у вигляді прямокутника, який поділений на три секції:

- верхня секція містить назву класу, яка зазвичай виділяється напівжирним шрифтом, вирівнюється по центру та пишеться з великої літери;
- середня секція включає атрибути класу, які записуються з вирівнюванням по лівому краю та починаються з малої літери;
- нижня секція містить методи або операції класу, також оформлені у вигляді списку, вирівняного по лівому краю.

Важливу роль у діаграмі відіграють зв'язки між класами. Основними типами зв'язків є: асоціація, узагальнення (наслідування), залежність, агрегація та композиція.

Асоціація відображає факт взаємозв'язку між об'єктами різних класів або навіть одного класу. Вона показує, що об'єкти можуть бути пов'язані між собою і зображується суцільною лінією між класами.

Узагальнення (наслідування) визначає ієрархію класів, де дочірній клас успадковує властивості та поведінку батьківського класу. Це означає, що об'єкти нащадка можуть використовуватися замість об'єктів базового класу. При цьому дочірній клас може розширювати функціональність, додаючи власні атрибути та методи.

Залежність показує ситуацію, коли один клас використовує інший, і зміни в одному можуть впливати на інший. Наприклад, клас «Замовлення» може залежати від класу «Клієнт», якщо містить посилання на нього.

Агрегація відображає відношення «частина–ціле», але при цьому частина може існувати незалежно від цілого. Тобто об’єкт вкладеного класу не залежить від життєвого циклу основного об’єкта.

Композиція є більш жорсткою формою агрегації. У цьому випадку частини створюються разом із цілим і не можуть існувати окремо. Наприклад, клас «Автомобіль» може містити клас «Двигун», де двигун є невід’ємною частиною автомобіля і не функціонує самостійно.

Таким чином, діаграма класів дозволяє наочно представити структуру системи та зв’язки між її елементами, що є важливим для проєктування, розробки та підтримки програмного забезпечення.

Діаграма класів ідеально підходить для відображення майбутньої структури класів у розроблюваному програмному забезпеченні. На рисунку 2.2 наведена діаграма класів для декстопного додатку управління складом СТО.

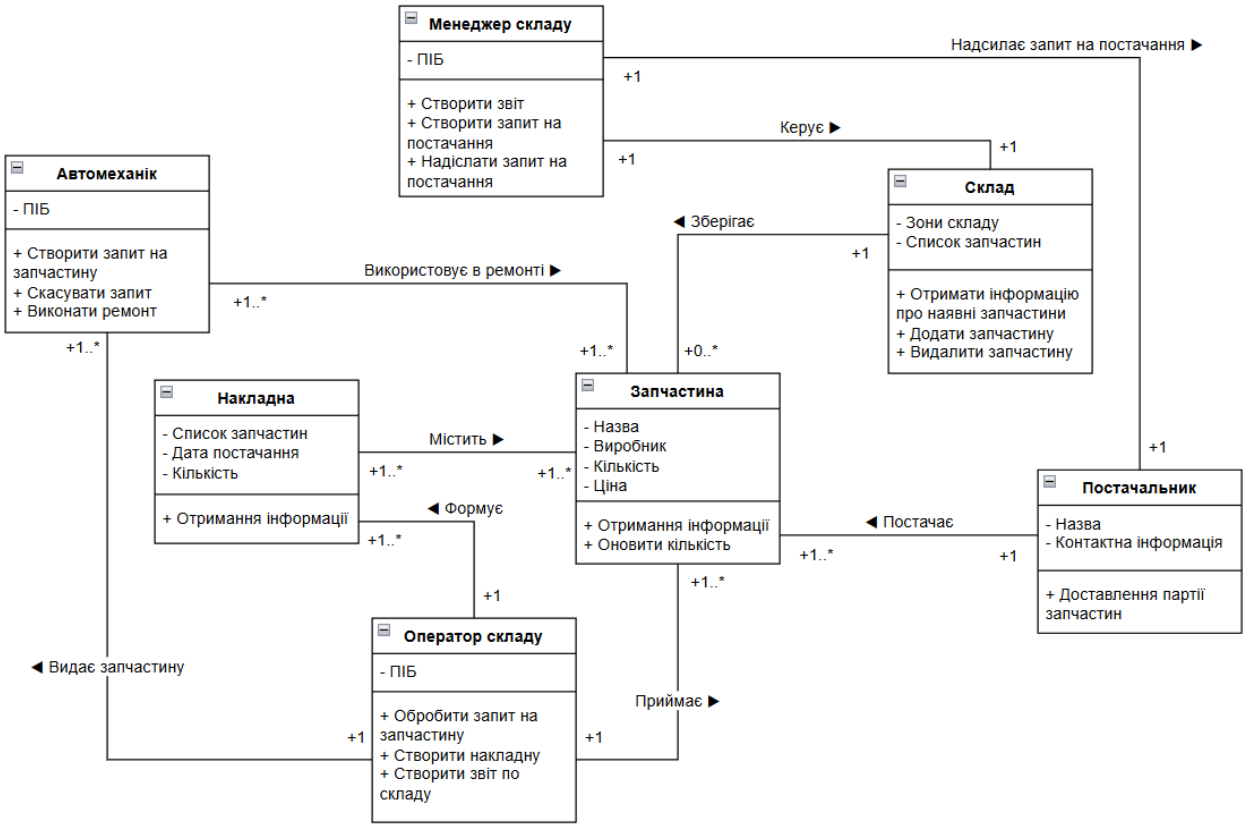


Рис.2.2 Діаграма класів додатку

Архітектура системи представлена сукупністю класів, що описують ключові об’єкти предметної області: запчастини, накладні, складські потужності, а також ролі персоналу (менеджер складу, оператор складу, автомеханік) та зовнішніх контрагентів (постачальник).

У межах виробничого циклу автомеханік ініціює запит на видачу запчастин, необхідних для проведення ремонтних робіт. Оператор складу здійснює верифікацію та опрацювання цих запитів, забезпечуючи дистрибуцію комплектуючих. Також оператор відповідає за документальне супроводження вхідних потоків, формуючи накладні, що містять специфікацію товарів, дату постачання та кількісні показники партій.

Зовнішній логістичний контур замикається на постачальнику, який реалізує відвантаження продукції згідно із замовленнями, сформованими менеджером складу.

Останній виконує функції загального адміністрування складського господарства, зокрема генерує аналітичну звітність та здійснює планування закупівель. Об’єкт «Склад» виступає центральним сховищем, що підтримує операції інвентаризації, додавання та списання активів. Сутність «Запчастина» має асоціативні зв’язки з усіма функціональними елементами системи, включаючи накладні, складські зони, постачальників та кінцевих споживачів у особі автомеханіків.

Опис структури класів наведений в табл. 2.2.

Прості кооперації – взаємодія між кількома об'єктами з метою виконання певної функції або досягнення спільної цілі. Кооперації зручно відображати як діаграми класів за допомогою UML. Проста кооперація являється частиною діаграми класів, яка уточнюється для відображення взаємодії між групою класів. Далі будуть наведені приклади простих кооперацій, побудованих на основі діаграми класів. (див. рис. 2.2)

Таблиця 2.2

Опис структури класів додатку

№	Назва класу	Атрибути	Методи
1	Автомеханік	ПІБ	Створити запит на запчастину, Скасувати запит, Виконати ремонт
2	Менеджер складу	ПІБ	Створити звіт, Створити запит на постачання, Надіслати запит на постачання
3	Склад	Зони складу, Список запчастин	Отримати інформацію про наявні запчастини, Додати запчастину, Видалити запчастину
4	Запчастина	Назва, Виробник, Кількість, Ціна	Отримання інформації, Оновити кількість
5	Постачальник	Назва, Контактна інформація	Доставлення партії запчастин

6	Оператор складу	ПІБ	Обробити запит на запчастину, Створити накладну, Створити звіт по складу
7	Накладна	Список запчастин, Дата постачання, Кількість	Отримання інформації

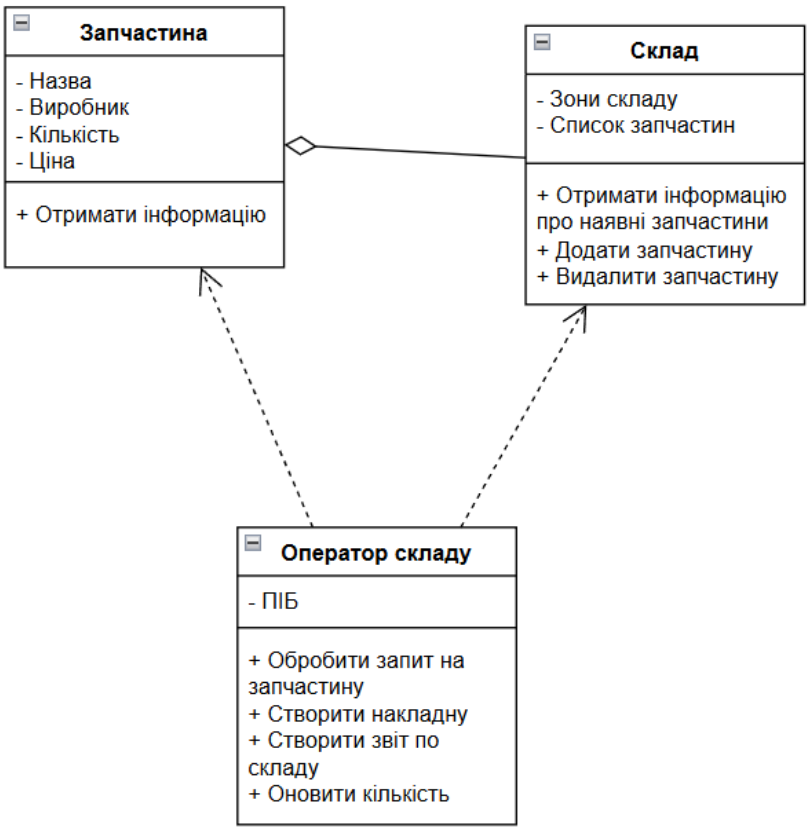


Рис.2.3 Управління запасами

У моделі простої кооперації управління запасами (рис. 2.3) детерміновано загальну логіку адміністрування складських ресурсів. Центральним об'єктом системи є склад, що акумулює відомості про запчастини; таким чином, усі номенклатурні одиниці асоційовані зі складом, який виступає базисом для обліку їхніх кількісних показників, технічних характеристик та атрибутивних даних.

Оператор складу ініціює взаємодію з системою, виступаючи активним суб'єктом управління. Комунікація з об'єктами обліку (запчастинами) реалізується опосередковано через інтерфейс складу: оператор звертається до системних ресурсів

для верифікації наявності ТМЦ або внесення коректив у базу даних. Будь-які операційні дії, як-от актуалізація кількісних залишків, генерація аналітичної звітності чи формування накладних, здійснюються шляхом маніпуляції даними, що зберігаються в межах складської структури.

Отже, склад виконує функцію централізованого інформаційного репозиторію та медіатора між масивом даних про запчастини та користувачем, тоді як оператор здійснює верифікацію та управління логістичними процесами на основі отриманої інформації.

В простій кооперації поставки запчастин (рис.2.4) показано, як організований процес постачання запчастин на склад. Менеджер складу ініціює постачання – він формує та надсилає запит до постачальника. Постачальник, отримавши запит, доставляє партію запчастин. Отримані запчастини обробляє оператор складу: він створює накладну, фіксує інформацію про доставку та додає запчастини на склад. Весь процес постачання закінчується на складі, який є місцем зберігання та обліку усіх запчастин.

Менеджер відповідає за ініціювання постачання, постачальник – за фізичну доставку, оператор складу – за оформлення та облік, а склад – за збереження запчастин.

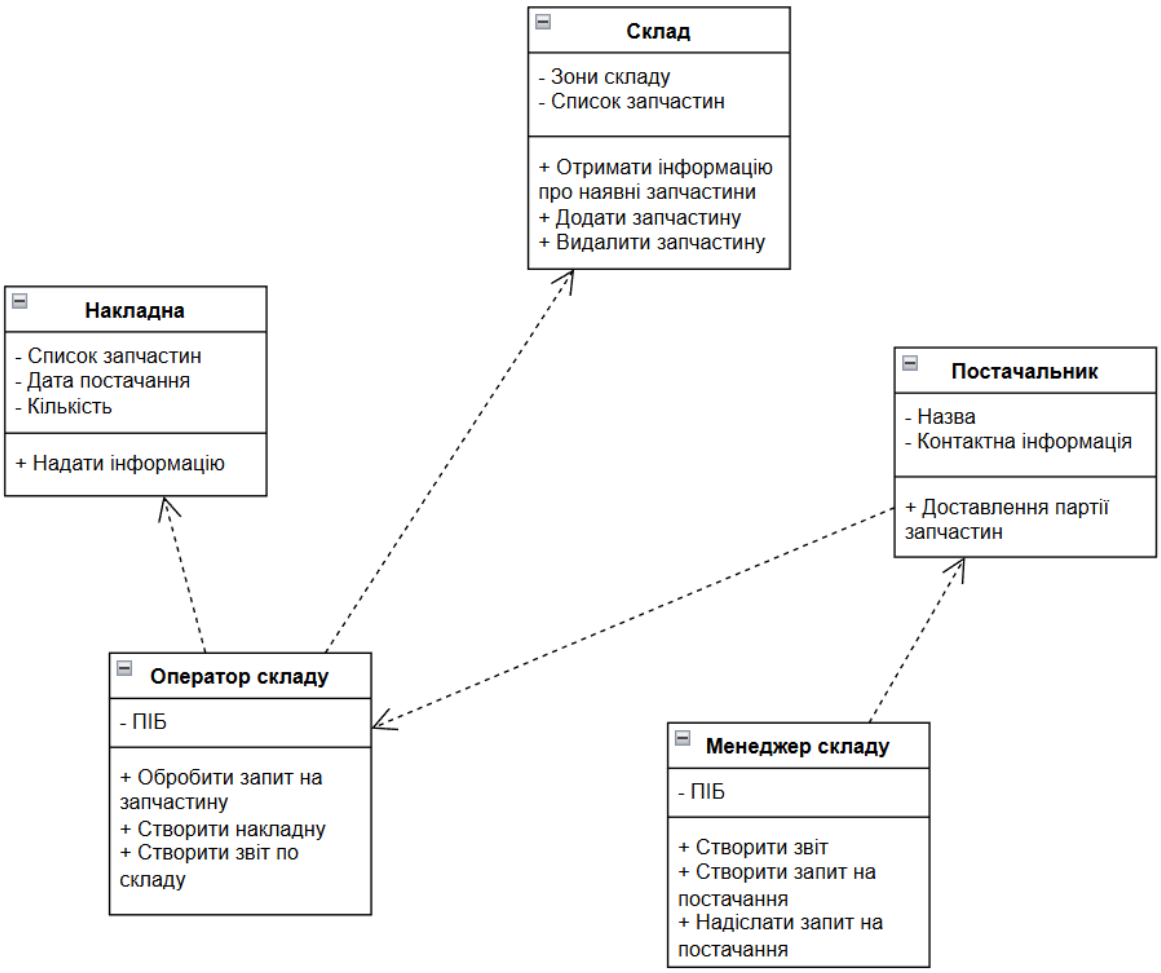


Рис.2.4 Поставка запчастин

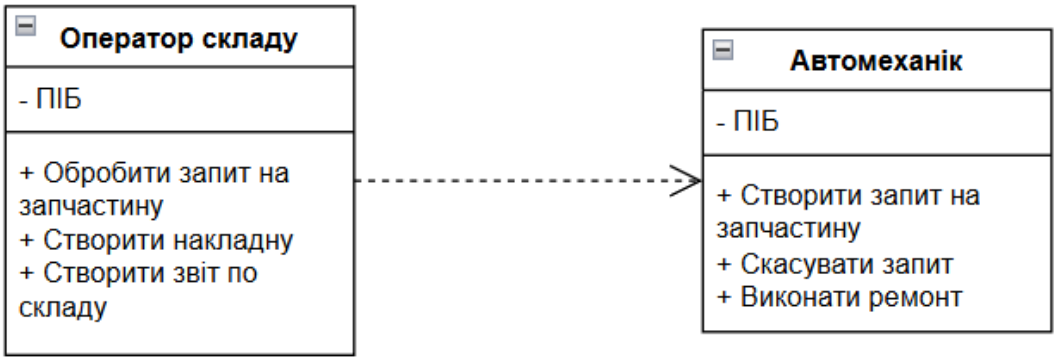


Рис.2.5 Видача запчастин

Проста кооперація видачі запчастин (рис.2.5) демонструє процес видачі запчастин зі складу. Автомеханік створює запит на отримання запчастин, які йому потрібні для виконання ремонту. Оператор складу отримує цей запит, обробляє його та формує накладну на видачу. Після цього запчастини списуються зі складу, а автомеханік може розпочати ремонт.

2.3 Діаграма пакетів

Діаграма пакетів є одним із типів структурних діаграм UML, яка використовується для подання архітектури програмної системи на високому рівні абстракції. Її основним елементом виступає пакет — логічне об’єднання класів, модулів, підсистем або інших структурних компонентів. Такі діаграми застосовуються для відображення модульної організації системи, а також для демонстрації залежностей між її частинами.

Використання діаграм пакетів є особливо важливим на етапі проєктування програмного забезпечення, коли необхідно сформувати загальне уявлення про структуру системи, розподілити її на окремі підсистеми та визначити взаємозв’язки між ними. Це сприяє створенню більш гнучкої архітектури, полегшує масштабування системи та спрощує її подальший супровід і модернізацію.

Діаграма пакетів (рис. 2.6) відображає архітектуру програмної системи, побудовану за принципом багаторівневої (шарової) архітектури. На верхньому рівні знаходиться презентаційний шар, який містить інтерфейси для різних категорій користувачів: автомеханіка, оператора складу та менеджера. Цей шар відповідає за взаємодію з користувачем і передає запити до нижчих рівнів системи.

Наступним є шар бізнес-логіки, який містить основні модулі системи, зокрема компоненти для управління користувачами, обробки замовлень та управління складськими операціями. Саме тут виконується основна логіка обробки даних і прийняття рішень відповідно до правил системи. Бізнес-логіка, у свою чергу, взаємодіє з шаром доступу до даних.

Шар доступу до даних виконує функцію посередника між бізнес-логікою та базою даних. Він відповідає за формування та виконання запитів, отримання та запис інформації, а також забезпечення коректної взаємодії з системою збереження даних. Нижнім рівнем архітектури є шар даних, у якому безпосередньо розміщено базу даних системи.

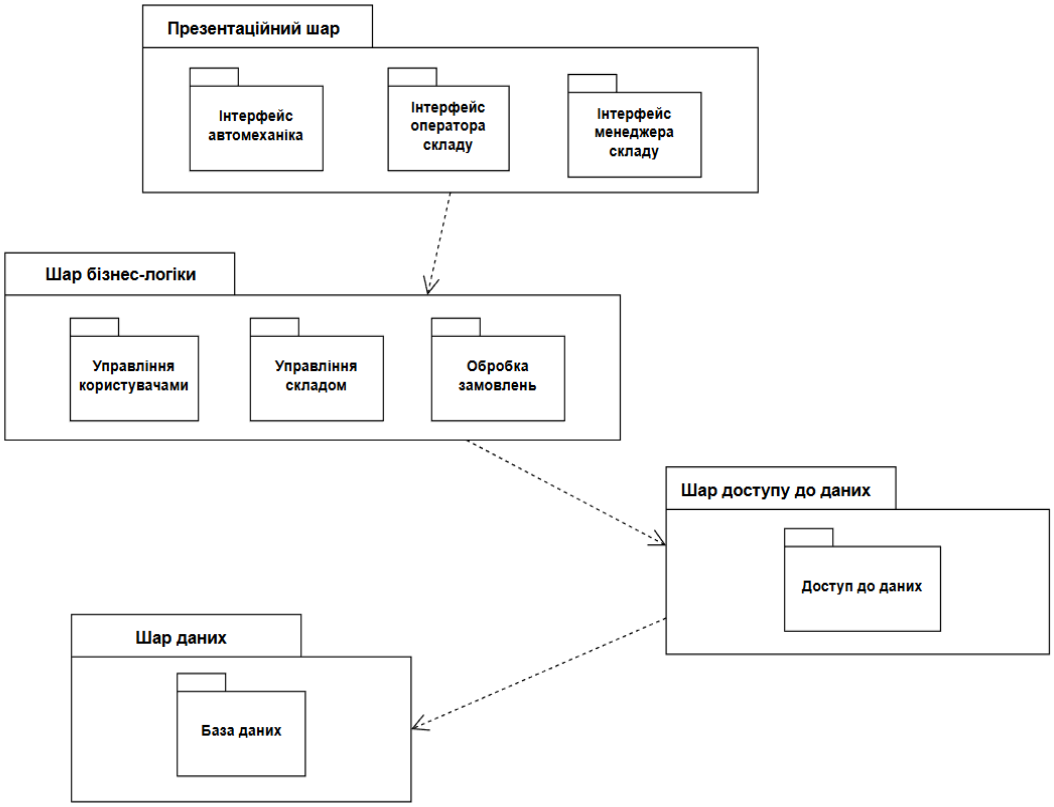


Рис.2.6 Діаграма пакетів додатку

Такий підхід до організації архітектури дозволяє чітко розмежувати відповідальність між компонентами системи. Кожен шар виконує власну функцію та взаємодіє переважно з сусідніми рівнями, що забезпечує ізоляцію компонентів, підвищує масштабованість, спрощує тестування та подальший супровід програмного забезпечення.

2.4 Діаграма компонентів

Діаграма компонентів UML надає концептуальну картину взаємодії між різними системами. Можуть бути присутні як аспекти логічного, так і фізичного моделювання. Крім того, компоненти автономні. Це модульний системний елемент в UML, який можна замінити на альтернативні. Вони містять конструкції будь-якої складності та є самодостатніми. Лише через інтерфейси вкладені частини взаємодіють з іншими компонентами. Крім того, компоненти мають свої інтерфейси, але вони також можуть отримати доступ до операцій і служб інших компонентів за допомогою їхніх інтерфейсів. На діаграмі компонентів інтерфейси також показують зв'язки та залежності в архітектурі програмного забезпечення.

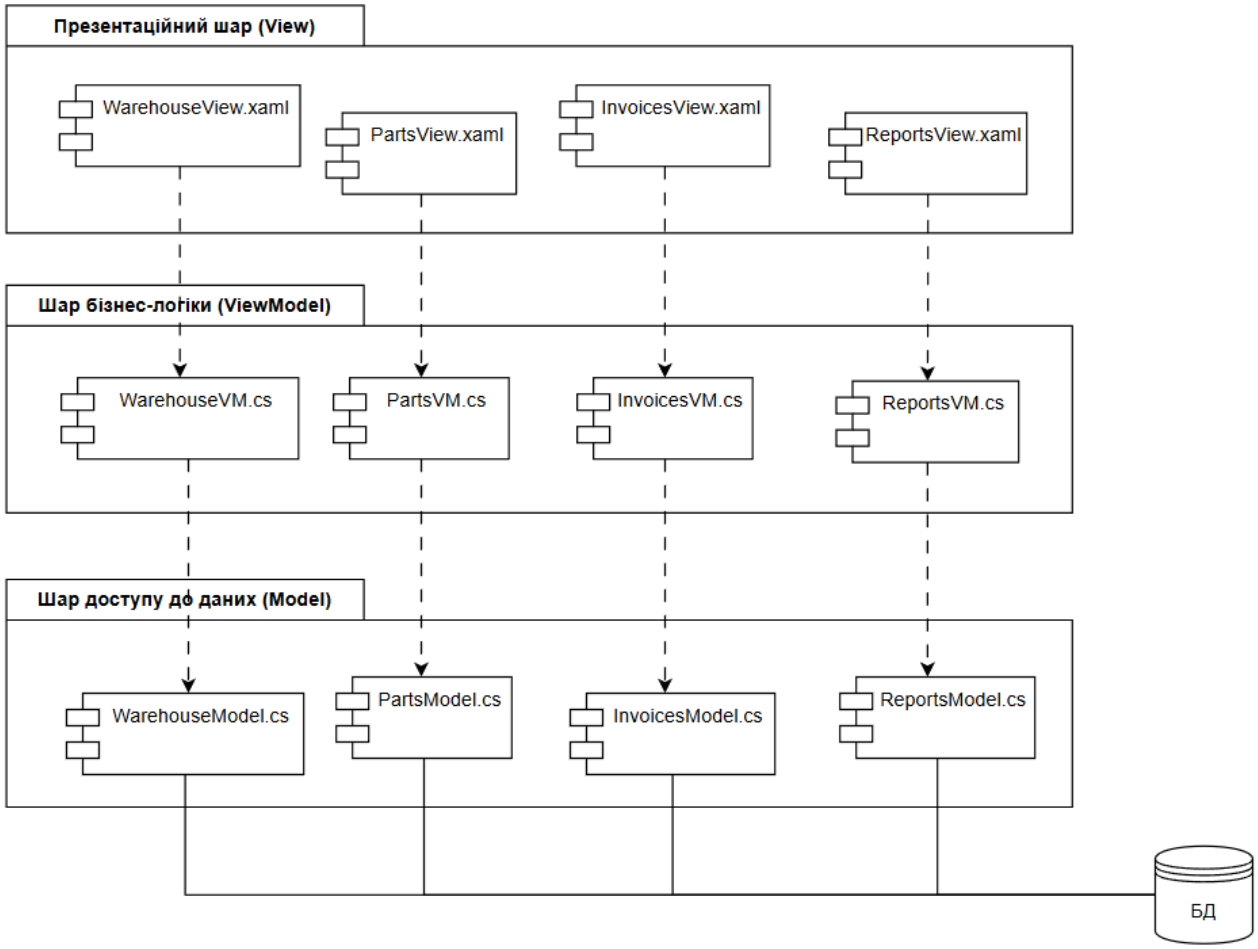


Рис.2.7 Діаграма компонентів додатку

Діаграма компонентів (рис.2.7) демонструє трирівневу архітектуру додатку, реалізовану за принципом MVVM (Model-View-ViewModel). Вона складається з трьох основних шарів: презентаційного (View), бізнес-логіки (ViewModel) та доступу до даних (Model). Кожен шар представлений окремими компонентами, які відповідають за роботу з відповідними сутностями системи: склад (Warehouse), запчастини (Parts), накладні (Invoices) та звіти (Reports). Компоненти верхнього рівня (XAML-представлення) взаємодіють із відповідними ViewModel, які взаємодіють із Model-класами для забезпечення доступу до бази даних. Дана діаграма ілюструє розділення відповідальностей і дотримання принципу слабкого зв’язку між шарами.

РОЗДІЛ ІІІ.
РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЗЕПЕЧЕННЯ

3.1 Система управління базою даних

Система управління базами даних (СУБД) являє собою спеціалізоване програмне забезпечення, призначене для адміністрування процесів створення, супроводу та експлуатації баз даних. СУБД функціонує як інтерфейс між прикладними рішеннями та масивами даних, забезпечуючи структуроване зберігання інформації та оптимізований доступ до неї. До засадничих функцій СУБД належать:

- дефініція даних шляхом проектування та модифікації схем бази даних;
- маніпулювання даними (ітерації додавання, видалення, оновлення та вибірки);
- підтримання цілісності даних через верифікацію коректності та несуперечливості інформаційних потоків;
- менеджмент транзакцій відповідно до протоколів ACID;
- адміністрування доступу для гарантування безпеки та авторизації суб'єктів;
- оптимізація продуктивності обробки запитів.

Класифікація СУБД здійснюється за моделлю даних (реляційні, об'єктно-орієнтовані, ієрархічні, NoSQL) та архітектурною побудовою (файл-серверні, клієнт-серверні, вбудовані, розподілені).

У процесі розробки автоматизованого робочого місця (АРМ) для складського господарства СТО з використанням технології Windows Presentation Foundation (WPF) було обрано реляційну СУБД Microsoft SQL Server. Дане рішення обґрунтоване необхідністю забезпечення прецизійної надійності зберігання даних, високої швидкодії та безшовної інтеграції з обраним технологічним стеком.

Застосування реляційної моделі в межах даного проекту зумовлене низкою чинників. По-перше, інформаційні структури СТО (номенклатура запчастин, контрагенти, логістичні операції) характеризуються високим ступенем формалізації

та наявністю чітко детермінованих інтерреляцій. По-друге, критичне значення має цілісність даних, що реалізується в реляційній моделі через механізми первинних і зовнішніх ключів та транзакційну логіку. По-третє, складський облік потребує виконання складних аналітичних запитів для генерації звітності, що ефективно забезпечується засобами мови SQL. Також система гарантує коректність паралельного доступу при одночасній роботі кількох операторів.

Microsoft SQL Server виступає оптимальним інструментарієм для зберігання та обробки даних завдяки глибокій інтеграції з екосистемою Microsoft, що передбачає повну сумісність із .NET Framework, використання середовища Visual Studio та підтримку Entity Framework як ORM-рішення. SQL Server демонструє високі показники продуктивності та стабільності в Windows-середовищі за рахунок системної оптимізації, ефективних алгоритмів кешування та потужних засобів забезпечення цілісності інформації.

У порівнянні з альтернативними СУБД, Microsoft SQL Server володіє суттєвими перевагами при розробці WPF-застосунків завдяки нативній інтеграції з платформою .NET та єдиному технологічному базису (.NET, XAML, C#), що сприяє інтенсифікації процесів розробки, верифікації та подальшого супроводу системи.

3.2 Розробка інформаційної бази

При розробці інформаційної бази використовувалась СУБД Microsoft SQL Server. Процес створення розпочався з формування структури БД за допомогою мови SQL. У процесі було сформовано 16 взаємопов'язаних таблиць, які охоплюють усі необхідні аспекти обліку та управління запчастинами на складі. Приклад коду для створення таблиці Part (запчастина) наведено на рис.3.1. Код інших таблиць БД наведено в ДОДАТКУ А.

```
-- Таблиця запчастин
CREATE TABLE Part
(
    Part_code CHAR(8) NOT NULL PRIMARY KEY,
    SKU_code VARCHAR(12) NOT NULL UNIQUE,
    Part_name VARCHAR(255) NOT NULL,
    Part_description VARCHAR(1000) NOT NULL,
    Manufacturer_code_FK CHAR(8) NOT NULL,
    Category_code_FK CHAR(8) NOT NULL,
    FOREIGN KEY (Manufacturer_code_FK) REFERENCES Part_manufacturer(Manufacturer_code),
    FOREIGN KEY (Category_code_FK) REFERENCES Part_category(Category_code)
);
```

Рис.3.1 Код для створення таблиці Part

Основу структури БД становлять такі ключові таблиці: Supplier (постачальники), Invoice (накладні), Storage_zone (зони складу), Storage_cell (комірки складу), Part_category (категорії запчастин), Part_manufacturer (виробники запчастин), Part (запчастини), Car_brand (бренди автомобілів), Part_image (зображення запчастин) та Part_batch (партії запчастин). Ці таблиці дозволяють структурувати всю базову інформацію про запчастини, їх характеристики та місцезнаходження.

Для підтримки функціональності системи були розроблені додаткові таблиці, такі як Stock_balance (залишки на складі), Part_movement (переміщення запчастин), Part_reservation (резервування запчастин) та Users (користувачі системи). Ці таблиці забезпечують облік кількості запчастин, відстеження їх переміщень між комірками складу та управління резервуванням запчастин для конкретних механіків.

Фізичну можель даних наведено на рис.3.2.

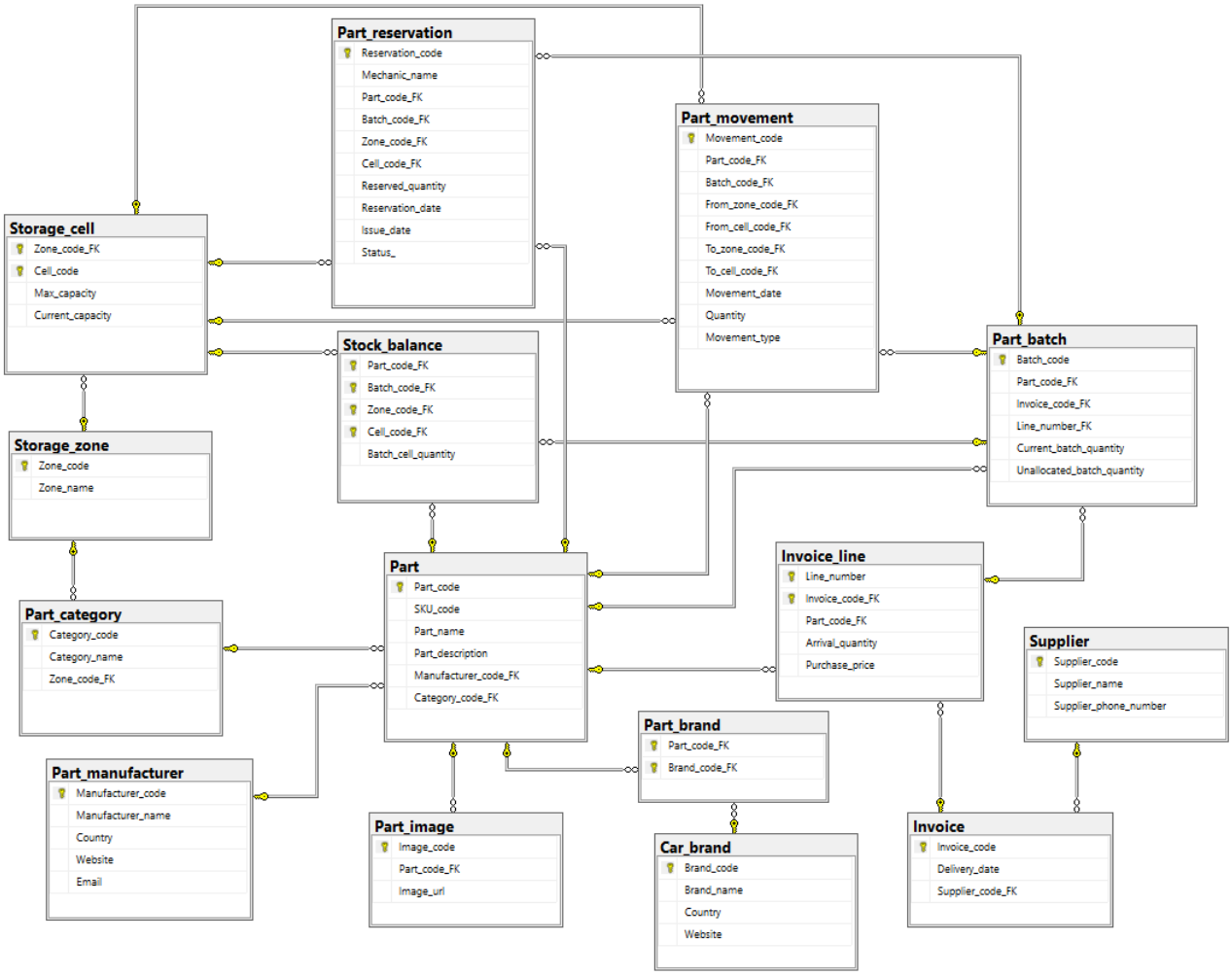


Рис.3.2 Фізична модель даних

При створенні таблиць було використано різні типи даних SQL Server, такі як CHAR та VARCHAR для текстових даних, INT для цілих чисел, DECIMAL для числових значень з фіксованою точністю, DATE та DATETIME для дат і часу, а також VARBINARY для зберігання бінарних даних, а саме хешів паролів.

Для забезпечення функціональності системи було розроблено сім збережених процедур, які потрібні для виконання основних операцій з управління запчастинами. Процедура DistributePartsToCell відповідає за розподіл запчастин з партії по коміркам складу. Процедури MovePartsBetweenCells, MovePartsBetweenCellsForReservation та MovePartsBetweenCellsForCancelReservation реалізують механізми переміщення запчастин між комірками для різних цілей. Процедури CreateReservation,

IssueReservedParts та CancelReservation забезпечують функціональність резервування, видачі та скасування резервацій запчастин для механіків. Код збережених процедур БД наведено в ДОДАТКУ Б.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Проектування настільного застосунку для операційної системи Windows передбачає локальне розгортання бази даних безпосередньо в інфраструктурі складського господарства СТО. Ключовими функціональними завданнями системи є прецизійний облік компонентів, стратегічне управління товарними залишками, верифікація логістичних операцій та генерація аналітичної звітності.

Вибір C# як мови програмування зумовлений її високою продуктивністю. Будучи компільованою мовою, C# забезпечує швидку обробку значних масивів даних, що є критичним для динамічних складських процесів. Статична типізація мови мінімізує виникнення помилок під час виконання (runtime errors), що є суттєвою перевагою для бізнес-орієнтованих систем обліку. Використання ORM-технології Entity Framework дозволяє оптимізувати взаємодію з даними, скорочуючи обсяг вихідного коду та прискорюючи цикл розробки. Технічною перевагою C# є нативний доступ до Windows API, що гарантує безперешкодну інтеграцію з периферійним обладнанням: сканерами штрих-кодів, принтерами етикеток та терміналами збору даних. Автоматичне управління пам'яттю (Garbage Collection) забезпечує стабільність системи протягом тривалих робочих змін без втрати продуктивності.

Для проектування інтерфейсу користувача обрано технологію Windows Presentation Foundation (WPF). Використання апаратного прискорення DirectX забезпечує плавність рендерингу складних табличних структур та об’ємних каталогів. Архітектурна парадигма MVVM (Model-View-ViewModel), покладена в основу WPF, сприяє модульній побудові системи, що полегшує її подальшу модифікацію та масштабування. Декларативний опис інтерфейсу за допомогою XAML дозволяє чітко

розмежувати бізнес-логіку та візуальне представлення. Механізм прив'язки даних (data binding) автоматизує синхронізацію моделі та графічного інтерфейсу, нівелюючи ризик невідповідності відображуваних даних. На відміну від веб-рішень, повноцінний інстальований додаток має необмежений доступ до системних ресурсів, що гарантує стабільну швидкість роботи.

Рівень зберігання даних реалізовано на базі MS SQL Server, що забезпечує повну підтримку транзакційних принципів ACID. Це є критично важливим для збереження цілісності інформації про матеріальні цінності при одночасному доступі кількох користувачів. Використання версії SQL Server Express є економічно обґрунтованим рішенням для малого бізнесу, оскільки вона не потребує ліцензійних витрат. Потужні алгоритми індексації гарантують високу швидкість пошуку в умовах постійного накопичення даних. Вбудовані інструменти резервного копіювання забезпечують збереженість інформації у разі апаратних збоїв, а підтримка збережених процедур дозволяє інкапсулювати складну логіку на рівні БД, мінімізуючи навантаження на мережу.

Отже, інтеграція зазначеного технологічного стека повною мірою відповідає технічним вимогам проекту автоматизації АРМ управління складом СТО, забезпечуючи оптимальний баланс між надійністю, продуктивністю та супроводжуваністю системи.

3.4 Алгоритмізація та програмування програмних модулів

В ході розробки програмної системи було реалізовано багато програмних рішень для вирішення поставлених задач. Далі розглянемо код деяких з них. Код розглянутих програмних рішень надано в ДОДАТКУ В.

Для реалізації функціоналу додавання нової запчастини до бази даних було розроблено відповідний алгоритм, який охоплює усі необхідні етапи перевірки введених даних, формування об’єкта запчастини, обробки асоційованих сутностей та збереження у базу. Цей процес є одним із ключових у системі, адже забезпечує

поповнення каталогу деталей з урахуванням їх категорій, виробників і сумісних брендів. На рисунку 3.3 наведено блок-схему, що ілюструє послідовність дій, які виконуються при додаванні нової запчастини.

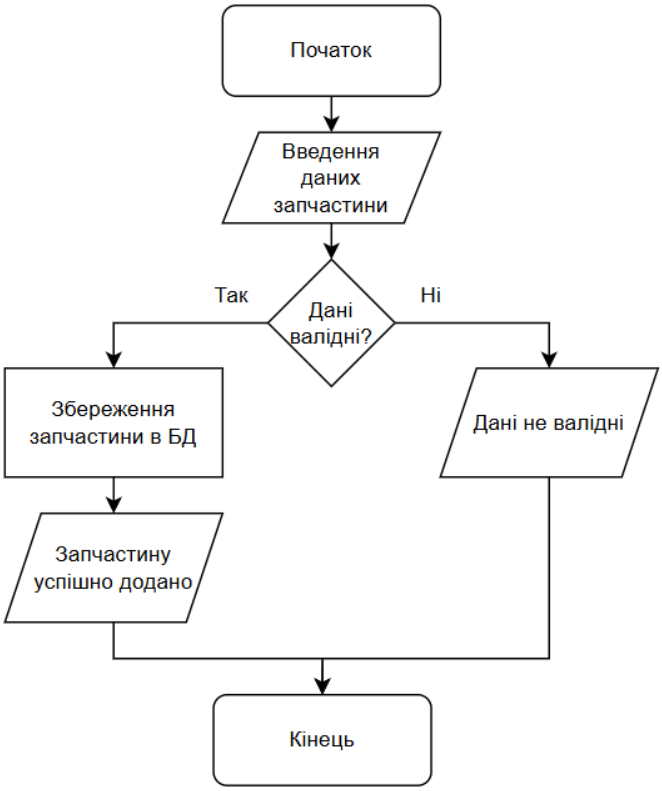


Рис.3.3 Блок-схема алгоритму додавання запчастини

У застосунку є спеціальна форма (рис.3.4) для додавання нової запчастини. Вона містить кілька полів для введення даних: артикул, назва, опис, категорія, марки авто та виробник. Частину з них користувач вводить вручну, а частину обирає з випадаючих списків.

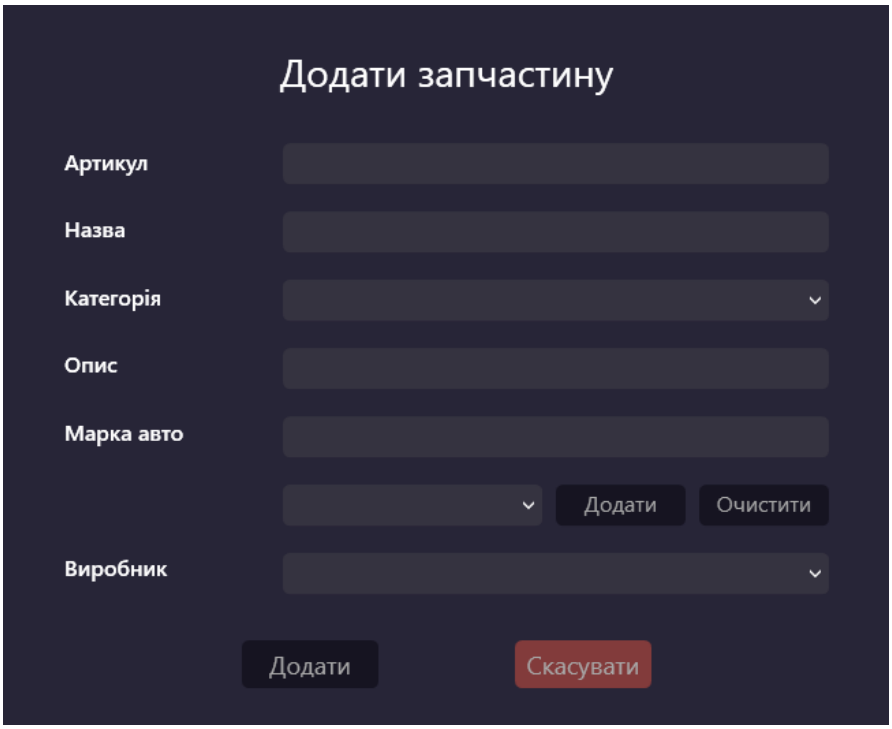


Рис.3.4 Форма додавання запчастини

Для марок автомобілів реалізовано окрему логіку взаємодії: користувач має можливість вибрати одну або кілька марок зі списку, що випадає, та додати їх до запчастини за допомогою відповідної кнопки. Усі обрані значення накопичуються в системі у вигляді текстового представлення, яке надалі обробляється та нормалізується перед збереженням у базу даних. Такий підхід дозволяє гнучко працювати з множинними зв’язками між запчастинами та марками автомобілів.

Після заповнення всіх необхідних полів форми користувач натискає кнопку «Додати». У ViewModel при цьому викликається асинхронний метод AddPartAsync(), який відповідає за обробку введених даних та їх подальше збереження. На першому етапі виконується перевірка обов’язкових полів на заповненість і коректність. Якщо хоча б одне з полів є порожнім або містить некоректні дані, система одразу відображає повідомлення з уточненням, яке саме значення необхідно виправити або додати.

У разі успішного проходження валідації створюється об’єкт Part, у межах якого виконується:

- генерація унікального ідентифікаційного коду запчастини для подальшої однозначної ідентифікації в системі;
- перетворення категорії та виробника у відповідні внутрішні коди через спеціалізовані сервіси, що забезпечує узгодженість даних;
- збереження всіх введених користувачем параметрів у модель даних.

Після цього сформований об’єкт передається до сервісу збереження, який відповідає за формування та виконання SQL-запитів. Спочатку основна інформація про запчастину записується до таблиці Part. Далі для кожної вибраної марки автомобіля здійснюється пошук її ідентифікатора в таблиці Car_brand, після чого створюються відповідні записи у зв’язувальній таблиці Part_brand, що реалізує зв’язок «багато-до-багатьох».

Усі операції виконуються в межах транзакції бази даних, що гарантує цілісність і узгодженість даних. У випадку, якщо хоча б одна з марок не знайдена або виникає інша помилка під час виконання операцій, транзакція повністю скасовується, і жодні зміни не зберігаються. Користувач у такому разі отримує повідомлення про помилку з поясненням причини. Після успішного завершення процесу форма очищується, а користувач бачить підтвердження успішного додавання запчастини.

Додатково варто зазначити, що після внесення запчастин до системи важливим етапом є їх правильне розміщення у складських комірках. Для цього реалізовано механізм ручного розподілу, який дозволяє оператору самостійно визначати кількість одиниць запчастини та обирати відповідну комірку для зберігання з урахуванням її заповненості. Під час цього процесу система виконує перевірку коректності введених даних, а також контролює наявність достатнього вільного місця у вибраній комірці, запобігаючи перевантаженню складу.

Таким чином, описаний алгоритм забезпечує не лише коректне додавання запчастин до системи, але й підтримує цілісність даних, мінімізує ризик помилок і сприяє ефективному управлінню складськими ресурсами. На рисунку 3.5 наведено

блок-схему алгоритму, що детально відображає логіку процесу ручного розподілу запчастин по складських комірках.

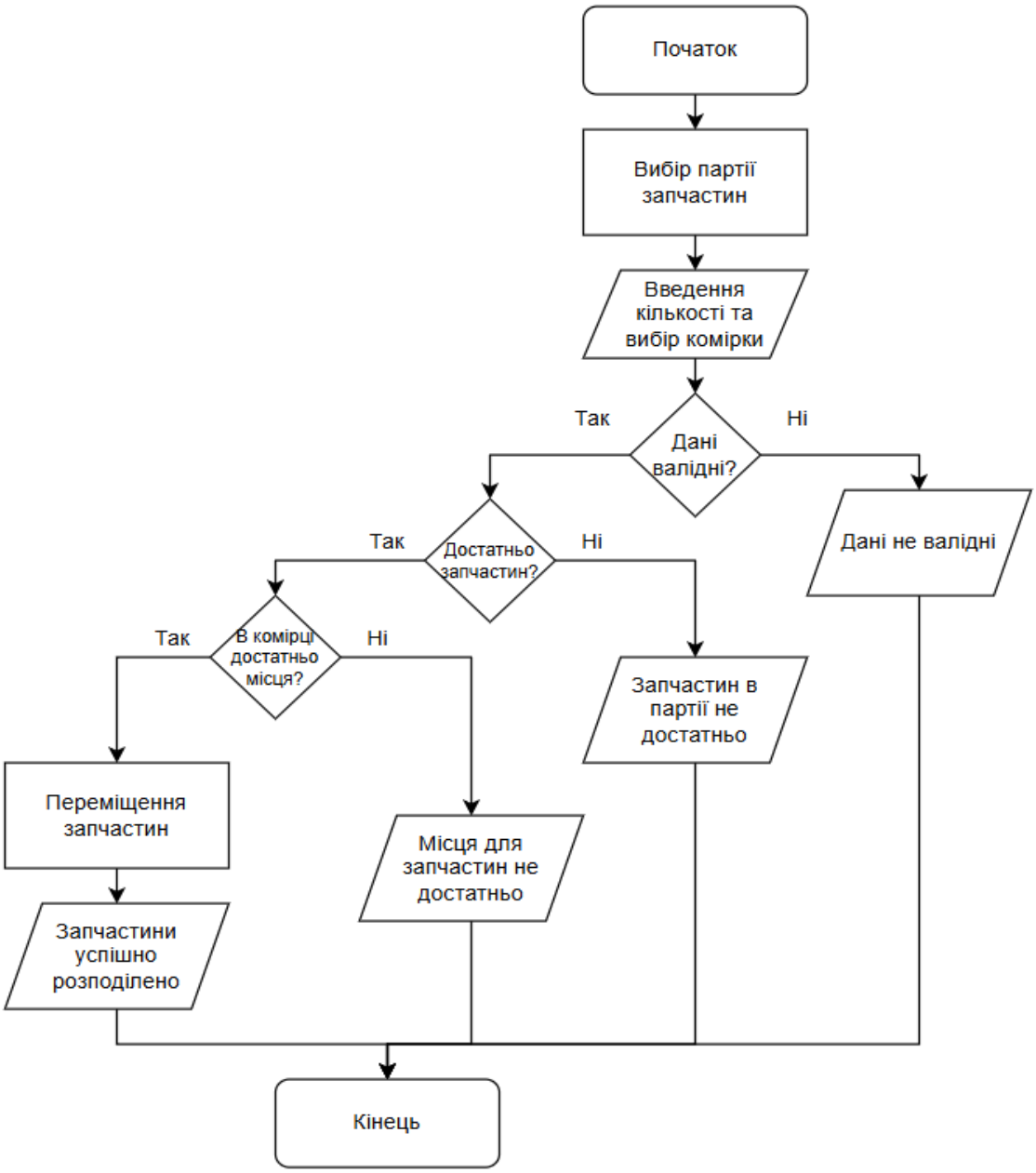


Рис.3.5 Блок-схема алгоритму розподілення запчастин в комірку

Після додавання партії запчастин у систему, її потрібно розподілити по комірках складу. Для цього користувач відкриває спеціальну форму (рис.3.6), де вводить кількість запчастин, яку потрібно перемістити, та обирає комірку зі списку доступних. У формі два поля: "Кількість" і "Комірка", які прив’язані до властивостей

BatchQuantity та SelectedCell відповідно. Після введення даних користувач натискає кнопку "Перемістити", яка активує команду MoveBatchCommand.

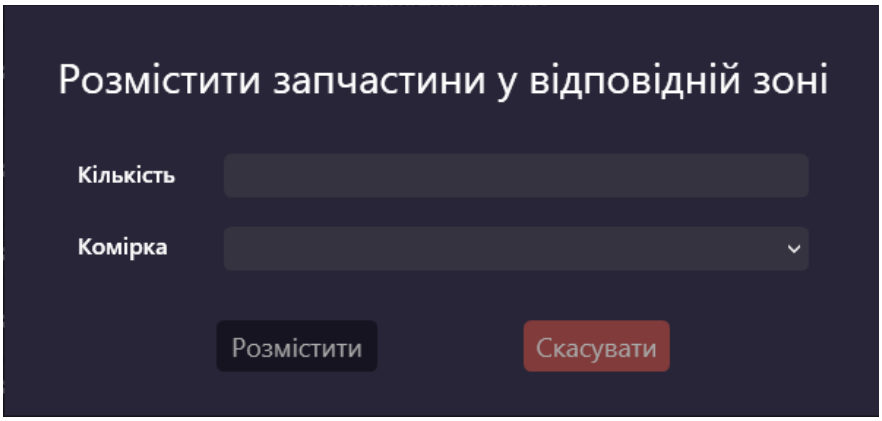


Рис.3.6 Форма розміщення запчастин

Ця команда викликає метод MoveBatchAsync, який відповідає за перевірку введених даних. Якщо хоча б одне поле порожнє, або кількість вказана некоректно, користувачу відображається повідомлення про помилку. У випадку, якщо все заповнено правильно, метод викликає сервіс _warehouseStockMovementService.DistributePartsToCellAsync, куди передає код партії, код зони, код обраної комірки та кількість.

Метод DistributePartsToCellAsync реалізує виклик збереженої процедури DistributePartsToCell у базі даних. Якщо ця процедура завершується успішно, на екран виводиться повідомлення про успішне переміщення і форма закривається. У разі помилки користувач бачить відповідне повідомлення.

Після надходження нових партій запчастин, користувач може скористатися автоматичним розподілом цих партій по комірках складу. У вікні (рис.3.8), яке відкривається, відображається коротке інформаційне повідомлення (Message), а також дві кнопки – "Розподілити" та "Скасувати". Кнопка "Розподілити" активує команду AutoMoveCommand, яка викликає метод AutoMoveBatch.

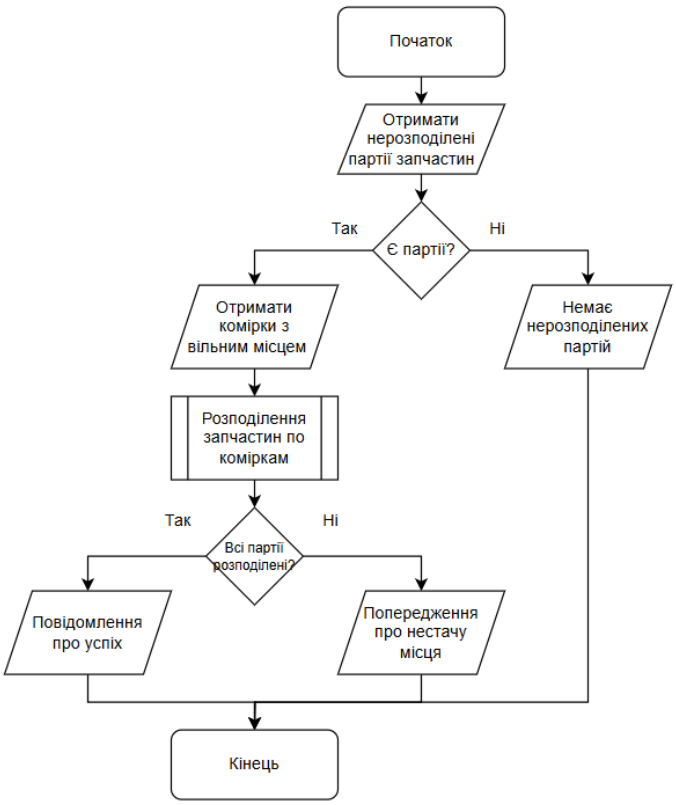


Рис.3.7 Блок-схема алгоритму автоматичного розподілення запчастин по коміркам

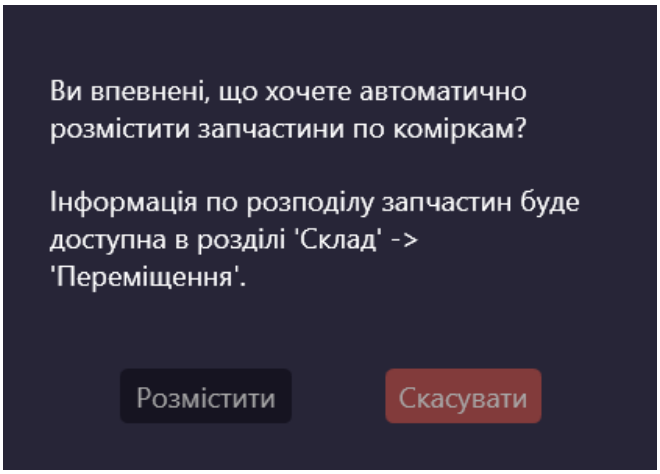


Рис.3.8 Вікно підтвердження автоматичного розподілення запчастин

Метод AutoMoveBatch ініціюється з верифікації наявності партій, що мають нерозподілені залишки (Unallocated_batch_quantity). У разі відсутності таких об’єктів виконання процедури припиняється. При виявленні нерозподілених залишків система

здійснює завантаження переліку доступних комірок зберігання через сервіс _loadDataService.GetStorageCellsAsync(). Отримані дані підлягають фільтрації за ознакою наявності вільного об’єму та сортуванню за показником поточної заповненості для пріоритетного використання найменш завантажених локацій.

Для кожної ідентифікованої партії реалізується алгоритм дистрибуції в межах комірок, що належать до функціональної зони складу, валідованої для конкретної номенклатурної одиниці (визначається за допомогою методу _loadDataService.GetStorageZoneCodeByPartAsync). Процес передбачає ітераційне розміщення запчастин у декількох комірках, де обсяг переміщення обмежений вільним місцем у кожній конкретній точці зберігання.

Фіксація кожної операції переміщення забезпечується викликом методу _warehouseStockMovementService.DistributePartsToCellAsync, який ініціює виконання збереженої процедури DistributePartsToCell на рівні бази даних, передаючи ідентифікатори партії, зони, комірки та кількісні параметри активів.

У ситуації, коли сумарна ємність доступних комірок є недостатньою для повного розміщення партії, система генерує попередження із зазначенням обсягу нерозподілених залишків. За умови успішної інкорпорації всіх партій у складський простір користувач отримує відповідне інформаційне повідомлення про завершення процесу.

З огляду на мультибрендову сумісність окремих запчастин, у системі реалізовано механізм асоціювання одиниці товару з декількома автомобільними марками. Даний функціонал забезпечує релевантність пошукових алгоритмів та фільтрації за виробниками, що оптимізує операційну діяльність персоналу та мінімізує ризики видачі несумісних компонентів. На рисунку 3.9 представлено алгоритмічну модель послідовності дій при додаванні бренду до картки товару.

Модуль генерації звітності представлений інтерактивним інтерфейсом (рис. 3.10), що дозволяє дефінувати часовий діапазон (дати початку та закінчення періоду) та задавати додаткові параметри фільтрації, зокрема категоріальну належність.

Обробка вхідних параметрів та формування фінального документа ініціюється командою CreateReportCommand.

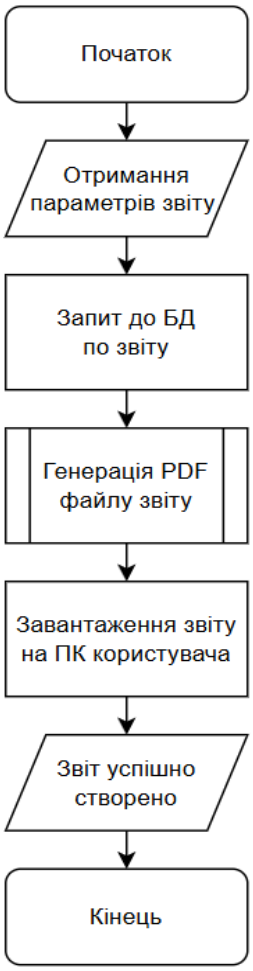


Рис.3.9 Блок-схема алгоритму генерації звітів

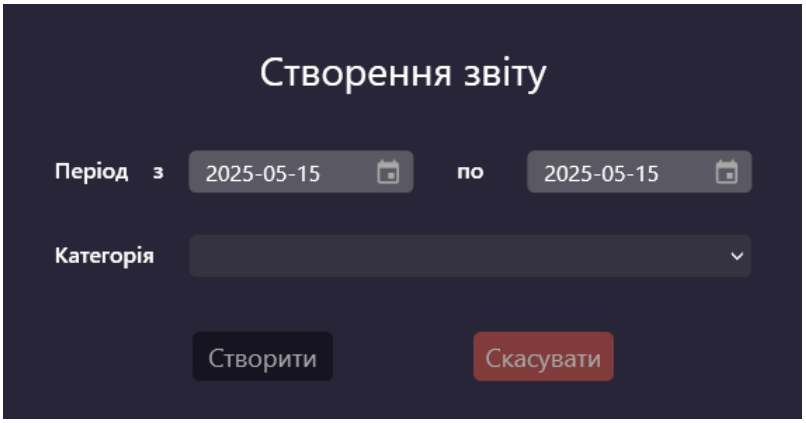


Рис.3.10 Форма генерації звіту

Після ініціювання створення звіту, ViewModel викликає асинхронний метод, який отримує потрібні записи з бази даних за допомогою _loadDataService. Ці записи фільтруються відповідно до вказаного діапазону дат і, за необхідності, категорії.

Далі система формує PDF-документ за допомогою бібліотеки iText7. Створюється заголовок із зазначенням обраного періоду, використовується шрифт Arial (звичайний та напівжирний). Основна частина звіту представлена у вигляді таблиці з фіксованими стовпцями, які заповнюються даними: код запчастини, назва, кількість, загальна сума тощо.

Після основного вмісту додається підсумок наприклад, обчислена загальна сума витрат, а також нижній колонтитул із поточною датою формування звіту та полем для підпису. На завершення викликається функція AddPageNumbers, яка додає номери сторінок на кожну сторінку PDF-документа.

Готовий звіт зберігається у вказане користувачем місце, після чого система покаже повідомлення про успішне завершення процесу або про помилку в разі її виникнення. Приклад звіту представлено в ДОДАТКУ Д.

РОЗДІЛ IV.
РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Тестування системи

Тестування програмного забезпечення є ітераційним процесом верифікації та валідації, спрямованим на виявлення дефектів і помилок у коді. Ця процедура забезпечує підтвердження коректності функціонування системи, її відповідності встановленим технічним специфікаціям, а також гарантує надійність та інформаційну безпеку продукту. Як критичний етап життєвого циклу розробки, тестування дозволяє ідентифікувати та нівелювати ризики до моменту впровадження системи в експлуатаційне середовище.

Сучасна методологія контролю якості програмних продуктів передбачає диференціацію за декількома напрямами.

Функціональне тестування спрямоване на перевірку відповідності ПЗ детермінованим функціональним вимогам. Об'єктом аналізу в даному контексті є коректність виконання операцій, обробка вхідних масивів даних та валідність вихідних результатів згідно з логікою бізнес-процесів.

Нефункціональне тестування оцінює якісні характеристики системи, зокрема її продуктивність, відмовостійкість, безпеку, ергономічність та сумісність з апаратними платформами. До цієї категорії належать навантажувальні випробування, тестування юзабіліті та перевірка вразливостей.

Додатково в інженерній практиці застосовуються такі види контролю:

- інтеграційне тестування — верифікація коректності взаємодії між окремими програмними модулями;
- системне тестування — комплексна оцінка відповідності всього програмного виробу технічному завданню;

- приймальне тестування — фінальна стадія визначення готовності продукту до релізу;
- регресійне тестування — підтвердження цілісності існуючого функціоналу після внесення змін до коду;
- стрес-тестування — аналіз поведінки системи в умовах екстремальних навантажень;
- юніт-тестування — ізольована перевірка працездатності елементарних компонентів коду.

Для оцінки розробленої системи було обрано пріоритетним методом функціональне тестування. Даний вибір аргументований специфікою проекту: наявністю чітко структурованих функціональних вимог, що підлягають верифікації, та необхідністю підтвердження цільового виконання прикладних завдань без аналізу внутрішньої логіки реалізації (метод «чорної скриньки»).

Процедуру тестування ілюструє кейс додавання нової накладної з подальшою дистрибуцією запчастин за складськими комірками. Після проходження процедури автентифікації виконується перехід до модуля управління накладними (рис. 4.1) та ініціюється виклик форми реєстрації нового документа (рис. 4.2).

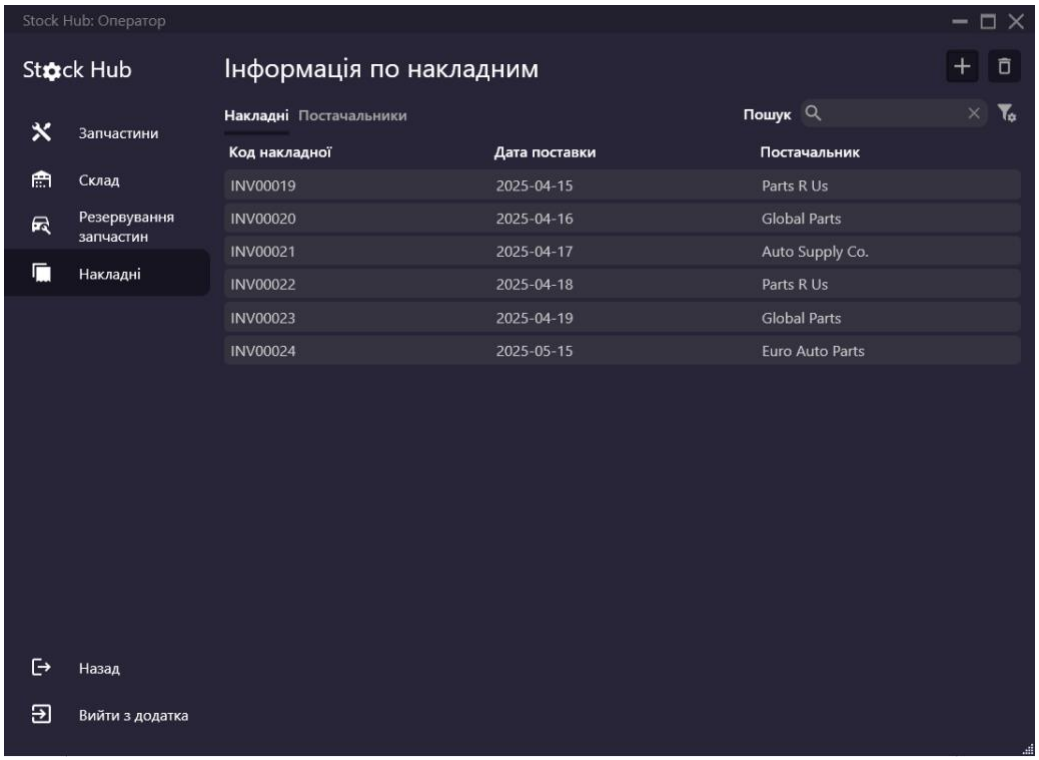


Рис.4.1 Сторінка накладних

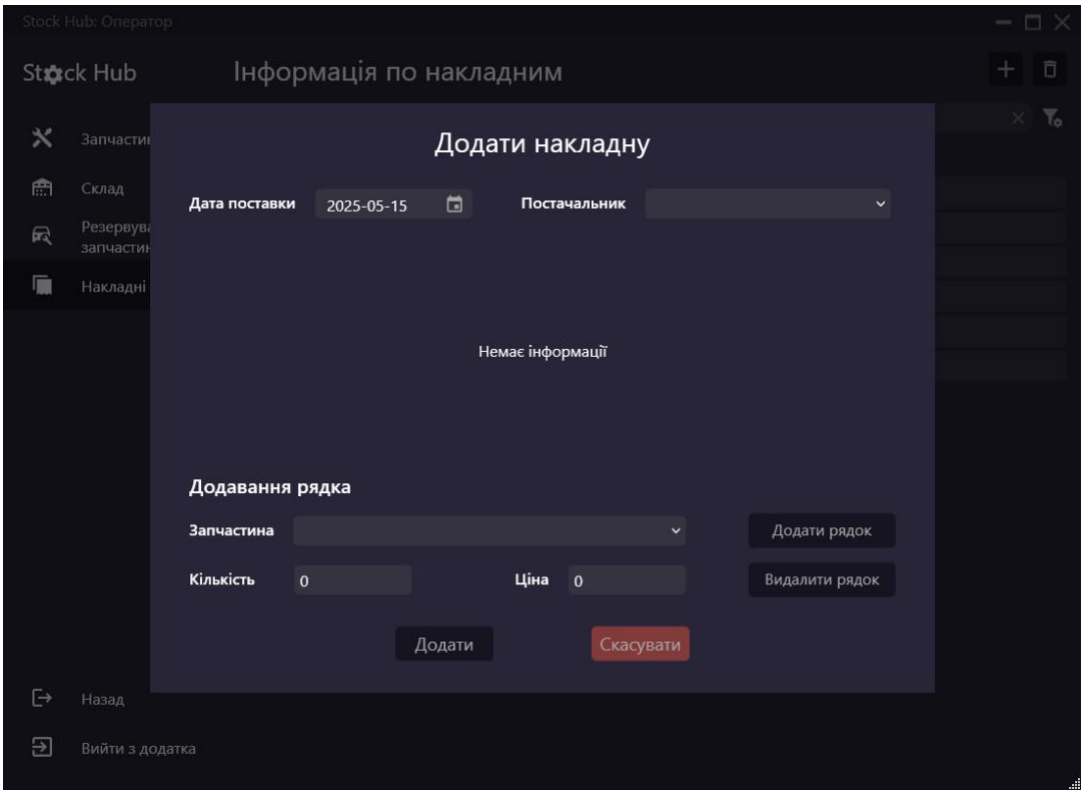


Рис.4.2 Форма додавання накладної

Для успішного додавання накладної необхідно виконати дві умови: обрати постачальника та додати хоча б один рядок накладної. При ігноруванні цих умов видає помилку (рис.4.3).

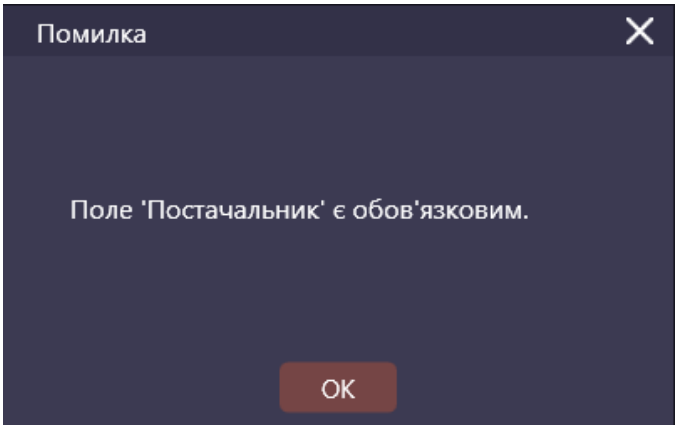


Рис.4.3 Помилка: поле “постачальник” порожнє

Для додавання накладної використовувалася перевірка чи поля заповнені коректно (рис.4.4).

```
var requiredFields = new (object FieldValue, string FieldName)[]
{
        (SelectedDate, "Дата"),
        (SelectedSupplier, "Постачальник"),
        (Lines, "Рядки")
};

foreach (var (value, name) in requiredFields)
{
        if (value == null || (value is string str && string.IsNullOrWhiteSpace(str)) || (value is
ObservableCollection<Invoice_line> coll && coll.Count == 0))
        {
                if (name == "Рядки")
                {
                        _modalNavigation.ShowMessage<ErrorMessageViewModel>(250, 400, "Накладна має
                мати хоча б 1 рядок!");
                }
        else
        {     _modalNavigation.ShowMessage<ErrorMessageViewModel>(250, 400, $"Поле '{name}' є
обов'язковим.");
        }
    return;
        }
}
```

Рис.4.4 Перевірка валідності полів накладної

Розглянемо випадок коли всі умови виконані, тоді накладна буде успішно додана в систему і з’явиться повідомлення про успіх операції (рис.4.5), крім цього, після закриття форми таблиця накладних оновилась (рис.4.6).

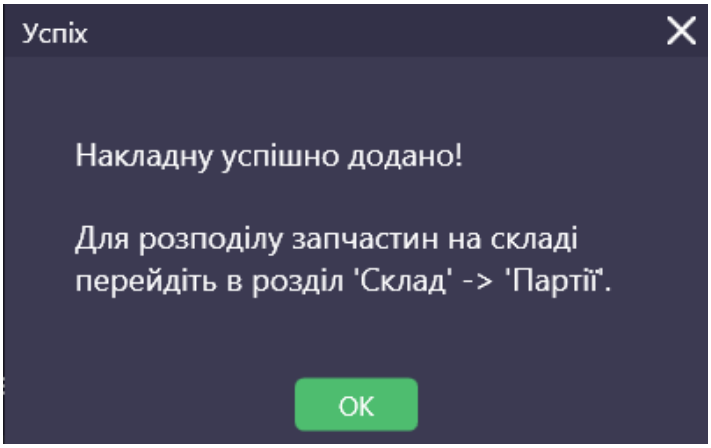


Рис.4.5 Успіх: накладна додана

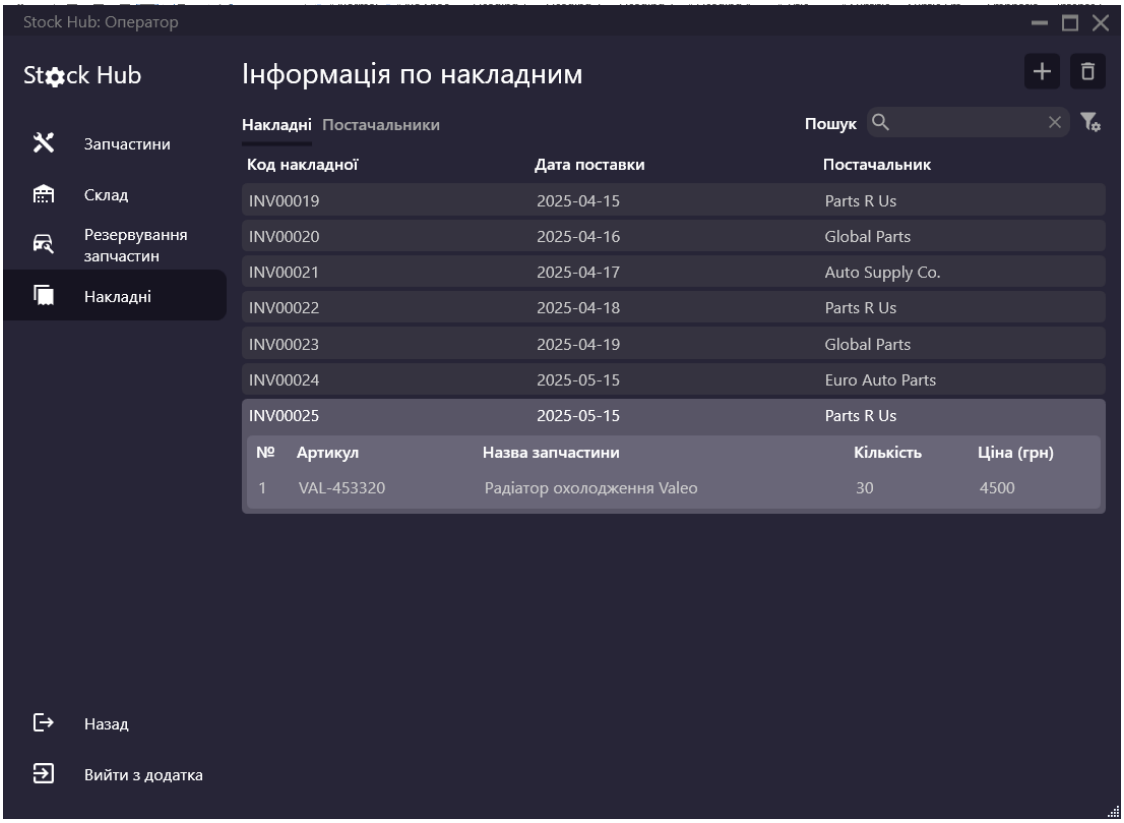


Рис.4.6 Таблиця накладних після оновлення

Оновлення таблиці відбувається після закриття модального вікна форми за допомогою оповіщення ViewModel сторінки з накладними, після сповіщення викликається функція OnModalClosed (рис.4.7) яка оновлює сторінку. Сповіщення було реалізовано за допомогою події, створеної в інтерфейсі сервісу для навігації між модальними вікнами (рис.4.8). Для коректної роботи сповіщення, ViewModel сторінки з накладними підписується на подію закриття модального вікна з сервісу навігації. Для підписання на подію необхідно в конструкторі ViewModel вказати функцію, яка виконується при оповіщенні.

```
private async void OnModalClosed(object sender, EventArgs e)
{
        await LoadInvoicesAsync();
        await FilterInvoicesAsync();
}
```

Рис.4.7 Функція OnModalClosed

```
public interface IModalNavigationService
{
    event EventHandler ModalClosed;
    event EventHandler ModalOpened;
    void ShowModal<TViewModel>(int height, int width) where TViewModel : ViewModel;
    void ShowModal(int height, int width, ViewModel viewModel);
    void CloseModal();
    void ShowMessage<TViewModel>(int height, int width, string message) where TViewModel : ViewModel;
    void ShowMessageForLogin<TViewModel>(int height, int width, string message) where TViewModel : ViewModel;
    void ShowMessageForMainWindow<TViewModel>(int height, int width, string message) where TViewModel :
ViewModel;
    void CloseMessage();
}
```

Рис.4.8 Інтерфейс сервісу навігації між модальними вікнами

Перейдемо до розподілення поставлених запчастин по коміркам складу. Для цього, по вказівці повідомлення про додавання накладної (див. рис.4.5), перейдено на сторінку складу у вкладку “Партії” (рис.4.9), де бачимо додану партію радіаторів охолодження. Далі спробуємо автоматично розподілити запчастини по коміркам (рис.4.10).

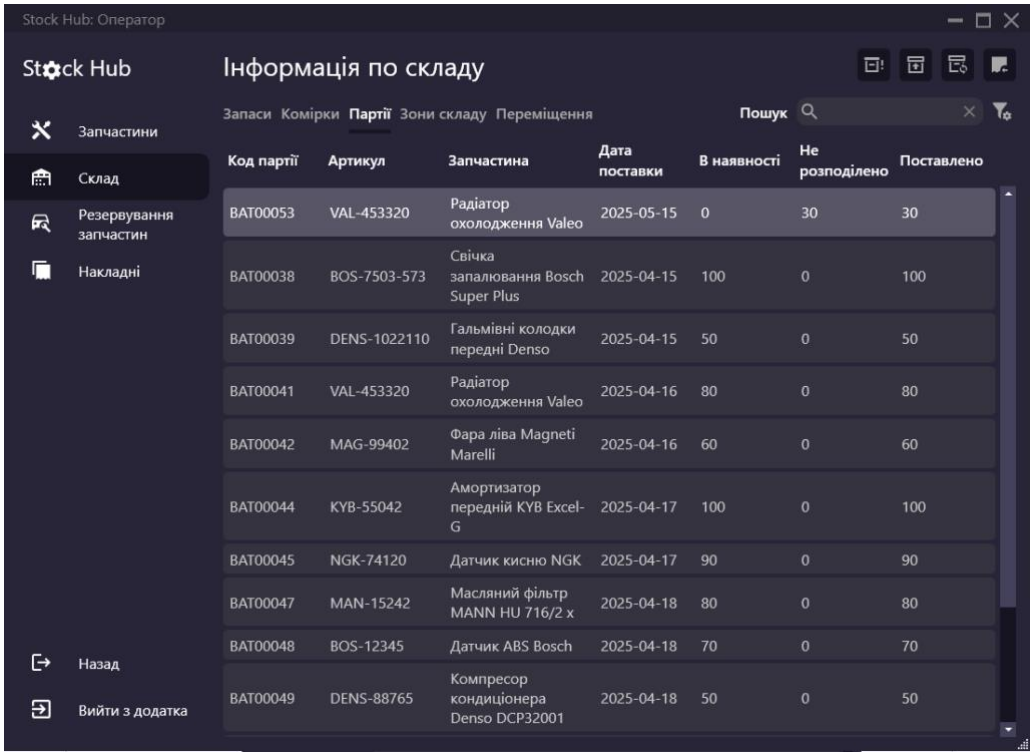
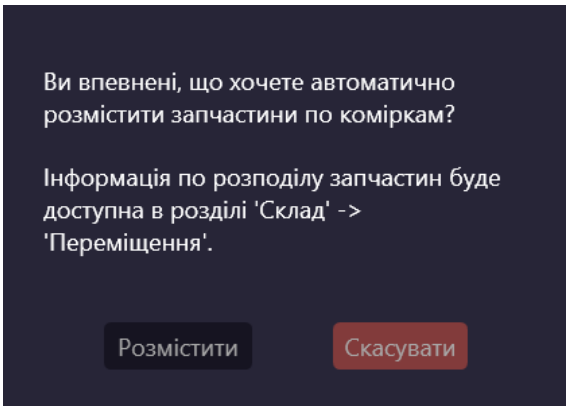


Рис.4.9 Таблиця поставлених партій



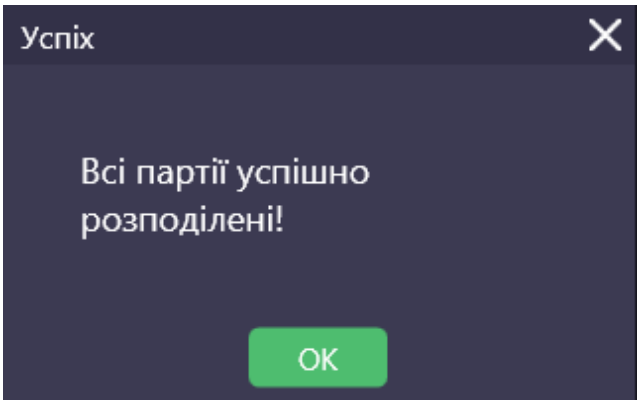


Рис.4.10 Підтвердження розподілення запчастин по коміркам

Рис.4.11 Успіх: партії розподілені по коміркам

Підтверджуємо розподілення. Успіх (рис.4.11). Бачимо, що радіатор змінив статус з “Не розподілено” на “В наявності” (рис.4.12).

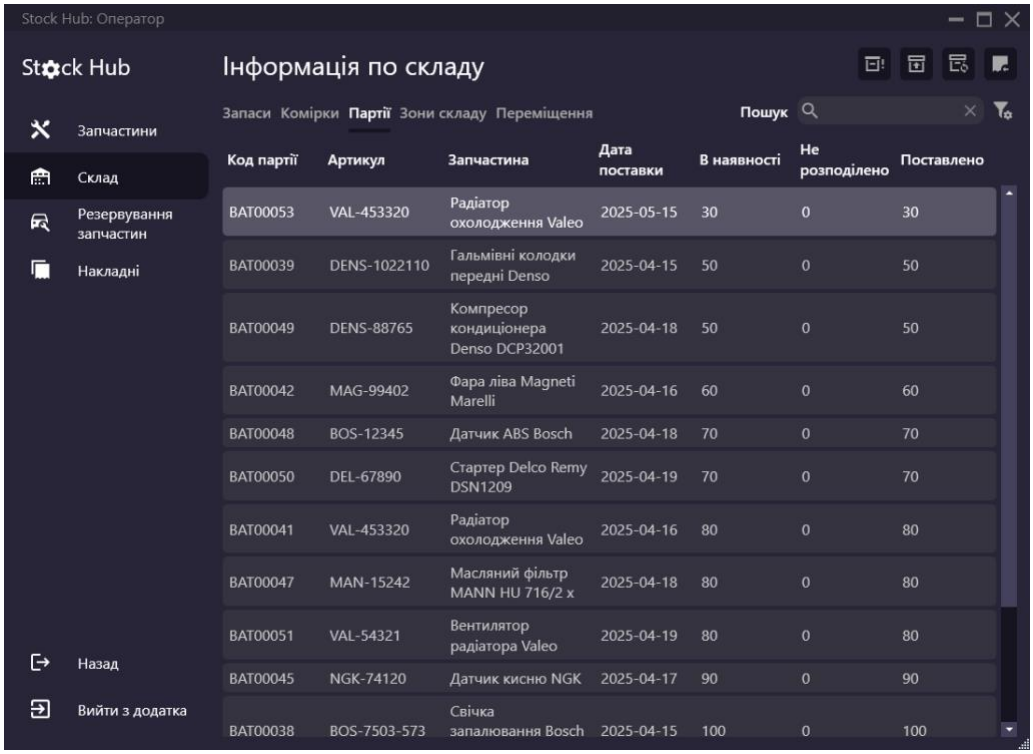


Рис.4.12 Зміна статусу партії

Далі перевіримо чи справді запчастини розподілились по коміркам. Для цього перейдемо у вкладку “Переміщення”, де бачимо лог переміщення запчастин (рис.4.13). Шукаємо “Радіатор охолодження Valeo”. Запчастини переміщені в комірку CELL-040 в зону охолодження та опалення.

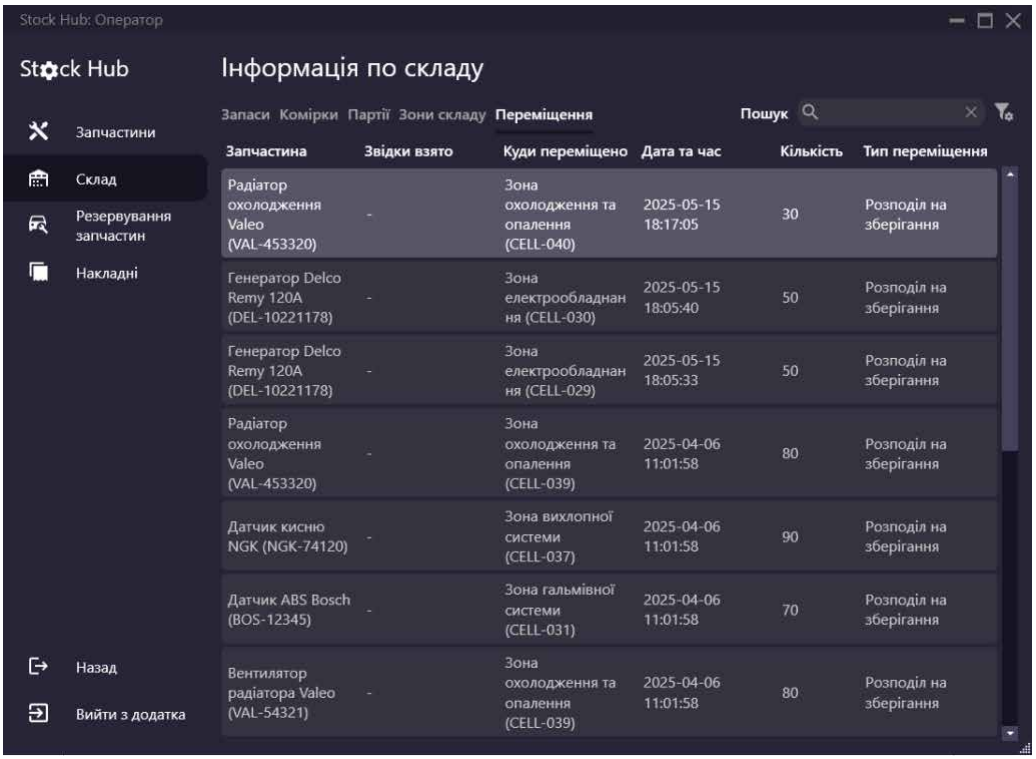


Рис.4.13 Логи переміщень запчастин

Для перевірки переміщення перейдемо у вкладку “Комірки” та знайдемо необхідну (рис.4.14). Переміщення пройшло успішно.

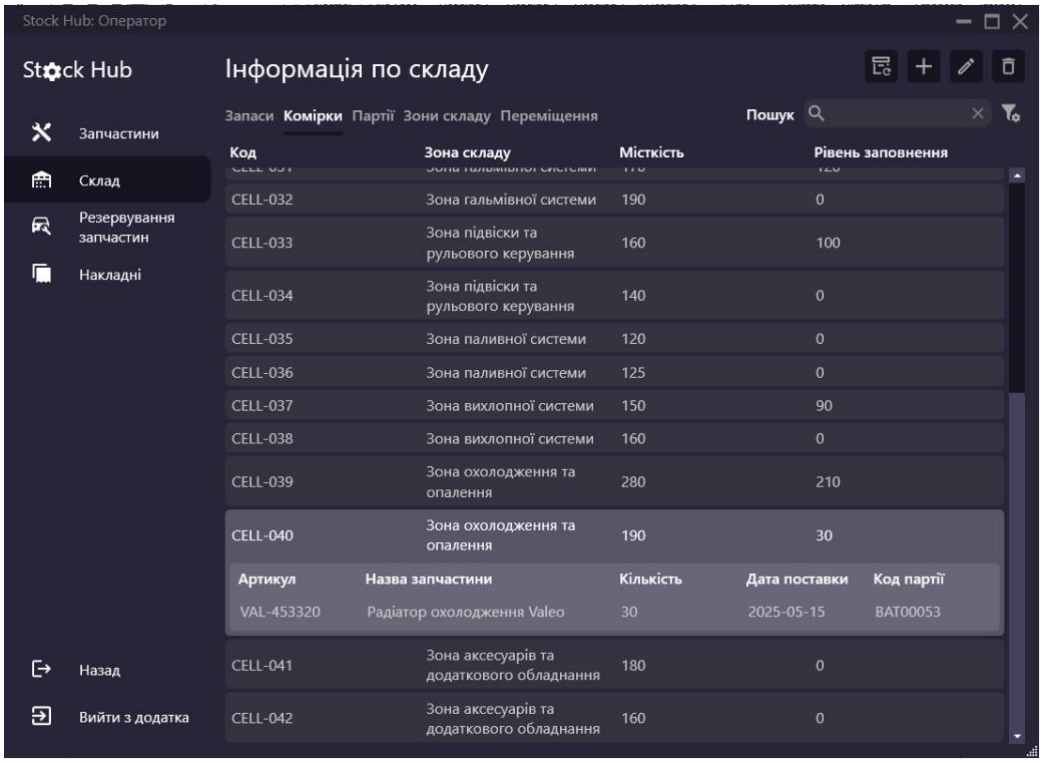


Рис.4.14 Таблиця комірок складу

Отже, функціональне тестування по створенню накладної та розміщення поставлених запчастин було проведено успішно, логіка працює добре та відображення UI частини здійснюється належним чином.

4.2 Вимоги до апаратного та програмного забезпечення

Подальший аналіз зосереджено на системних вимогах з точки зору власника підприємства, що дозволяє акцентувати увагу на стратегічних очікуваннях щодо функціональних можливостей, надійності та ергономічності програмного продукту.

Для забезпечення ефективної експлуатації автоматизованого робочого місця (АРМ) з управління складським господарством СТО необхідно дефінувати його архітектурну побудову, а також встановити регламентні вимоги до апаратного та програмного забезпечення. Ключовим етапом проектування є розробка діаграми розгортання (deployment diagram), яка візуалізує топологію системи та інтерреляції між її базовими компонентами.

Архітектурна модель системи, представлена на рис. 4.15, базується на дворівневій структурі, що охоплює два основні вузли: клієнтський термінал (User PC) та серверний сегмент (Server), функціонування яких забезпечується операційними системами родини Windows (версії 10/11).

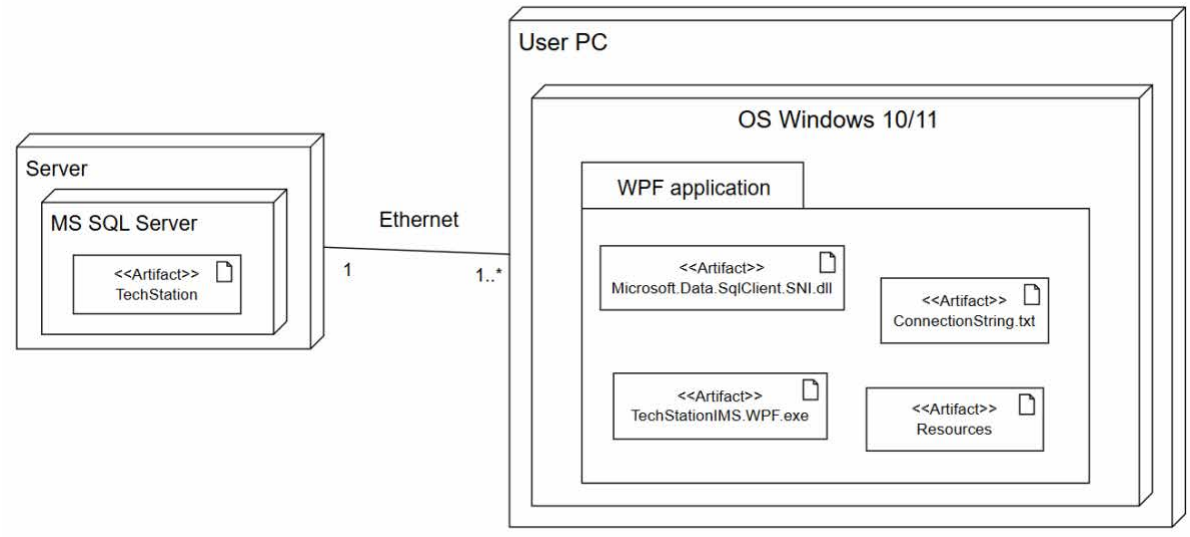


Рис.4.15 Діаграма розміщення

Система реалізована на основі традиційної клієнт-серверної архітектури. На стороні сервера використовується MS SQL Server, який містить базу даних «TechStation». Серверна частина відповідає за збереження інформації, обробку запитів користувачів, реалізацію бізнес-логіки та забезпечення цілісності даних у системі.

Клієнтська частина запускається на комп'ютері користувача під керуванням операційних систем Windows 10/11 і представлена WPF-додатком, який забезпечує графічний інтерфейс для взаємодії з системою. До складу клієнтського застосунку входять бібліотека Microsoft.Data.SqlClient SNI.dll, що забезпечує мережеву взаємодію з SQL Server, файл ConnectionString.txt, у якому зберігаються параметри підключення до сервера, виконуваний файл TechStationIMS.WPF.exe, а також папка Resources, що містить ресурси застосунку.

Після запуску програми на користувацькому ПК застосунок зчитує параметри підключення та встановлює з'єднання з сервером через мережу Ethernet. Далі користувач взаємодіє з графічним інтерфейсом, який формує запити до серверної частини. Сервер обробляє ці запити, виконує необхідні операції з базою даних і повертає результати назад клієнтському застосунку, після чого вони відображаються користувачу у відповідному інтерфейсі.

Апаратні вимоги до впровадження програмної системи визначають мінімальну конфігурацію обладнання, необхідну для стабільної та ефективної роботи всіх компонентів застосунку, включаючи серверну частину та клієнтські пристрої. Деталізовану специфікацію апаратного забезпечення наведено в таблиці 4.1.

Таблиця 4.1

Апаратні вимоги

Компонент	Мінімальні	Рекомендовані
Процесор	4 ядра, 2.0 GHz (x64)	8+ ядер, 3.0+ GHz (x64)
Оперативна пам'ять	8 GB RAM	16+ GB RAM
Жорсткий диск	100 GB вільного місця (HDD 7200 RPM або кращий)	250+ GB вільного місця (SSD, бажано NVMe)
Мережа	Gigabit Ethernet	Gigabit Ethernet або швидша

Програмні вимоги охоплюють перелік необхідного системного та прикладного програмного забезпечення, яке забезпечує коректну роботу інформаційної системи, зокрема операційне середовище, середовище виконання, бібліотеки та інструменти розробки. Повний перелік рекомендованих програмних компонентів подано в таблиці 4.2.

Таблиця 4.2

Програмні вимоги

Компонент	Мінімальні	Рекомендовані
Операційна система	Windows Server 2019 Standard	Windows Server 2022 Datacenter
СУБД	Microsoft SQL Server 2022 Standard Edition	Microsoft SQL Server 2022 Enterprise Edition
Додаткове ПЗ	-	Система моніторингу серверів

Для функціонування додатка необхідні встановлений та налаштований MS SQL Server для зберігання даних, а також створена база даних з необхідними таблицями та зв'язками.

Наступним етапом є визначення вимог до системи з точки зору кінцевого користувача, що дозволяє врахувати зручність взаємодії, швидкість доступу до функцій, а також технічні умови для належного функціонування клієнтської частини застосунку.

Апаратні вимоги для персонального комп’ютера користувача включають мінімальну конфігурацію обладнання, необхідну для стабільної роботи інтерфейсу, обробки запитів та взаємодії з базою даних. Ці характеристики детально наведено в таблиці 4.3.

Таблиця 4.3

Апаратні вимоги

Компонент	Мінімальні вимоги	Рекомендовані вимоги
Процесор	2 ядра, 1.8 GHz	4 ядра, 2.4+ GHz
Оперативна пам'ять	4 GB RAM	8+ GB RAM
Жорсткий диск	10 GB вільного місця (HDD)	20+ GB вільного місця (SSD)
Відеокарта	З підтримкою DirectX 11, 1 GB відеопам'яті	З підтримкою DirectX 12, 2+ GB відеопам'яті
Мережа	100 Mbps Ethernet	Gigabit Ethernet

Програмні вимоги для персонального комп’ютера користувача включають перелік необхідного програмного забезпечення, яке забезпечує запуск та повноцінне використання клієнтської частини системи, зокрема операційну систему, середовище виконання, підтримувані бібліотеки та допоміжні утиліти. Детальний перелік програмних компонентів і їх версій наведено в таблиці 4.4.

Для початку роботи користувачеві необхідно встановити WPF-додаток, який є основним інтерфейсом взаємодії з інформаційною системою. Встановлення передбачає копіювання виконуваних файлів на робочу станцію та, за потреби, попередню інсталяцію необхідних компонентів середовища виконання .NET, що забезпечує коректну роботу всіх елементів застосунку.

Таблиця 4.4

Програмні вимоги

Компонент	Мінімальні вимоги	Рекомендовані вимоги
Операційна система	Windows 10 версія 21H2 або новіша	Windows 11 22H2 або новіша
Системні компоненти	DirectX 11	DirectX 12

Таким чином, ретельне визначення апаратних і програмних вимог як на стороні сервера, так і для кінцевого користувача є необхідною умовою для впровадження ефективної та стабільної системи. Забезпечення відповідності цим вимогам дозволяє уникнути проблем сумісності, забезпечити швидкодію та гарантовану підтримку всіх функціональних можливостей програмного продукту в реальних умовах експлуатації.

4.3 Склад інсталяційного пакету

Для програмного забезпечення автоматизованого робочого місця для управління складом станції технічного обслуговування інсталяційний пакет склається лише з папки самого додатка. На рис. 4.16 показаний вміст папки додатка.

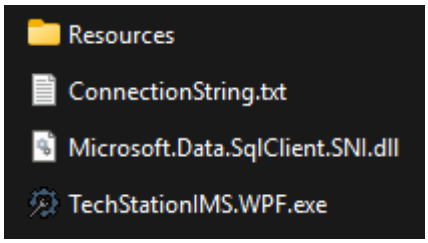


Рис.4.16 Вміст істаляційного пакету

Перед запуском програми необхідно вказати параметри підключення до бази даних у файлі ConnectingString.txt. У ньому слід записати рядок підключення з адресою сервера, назвою бази та обліковими даними. Програма зчитує ці дані під час запуску, тому їх коректність є критично важливою для стабільної роботи системи.

Таким чином, склад інсталяційного пакету є мінімальним, однак достатнім для автономної роботи застосунку на будь-якому клієнтському ПК, що відповідає зазначеним технічним вимогам. Завдяки простій процедурі розгортання система може бути оперативно встановлена та налаштована без додаткового програмного забезпечення або складних інсталяційних сценаріїв.

ВИСНОВКИ

У результаті виконання дипломного проєкту було проведено комплексне науково-технічне дослідження процесів управління складським господарством станції технічного обслуговування (СТО). За результатами роботи реалізовано ключові етапи життєвого циклу розробки прикладного програмного забезпечення для АРМ управління складом.

До основних результатів дослідження належать проведення системного аналізу предметної області, у ході якого здійснено дескрипцію діяльності складу СТО, ідентифіковано базові бізнес-процеси та ролі користувачів. За допомогою уніфікованої мови моделювання (UML) побудовано діаграми прецедентів, активностей, послідовностей та класів, що дозволило сформувати несуперечливу логічну модель функціонування системи.

Також виконано компаративний аналіз існуючих рішень шляхом критичного огляду систем-аналогів, таких як Tocan WMS та LSC WMS. Виявлені функціональні дефіцити та висока вартість існуючих продуктів дозволили обґрунтувати доцільність розробки спеціалізованого економічно ефективного рішення, адаптованого до потреб малих та середніх підприємств автосервісу.

У межах проектування архітектури даних розроблено концептуальну, логічну та фізичну моделі бази даних. Як рівень зберігання обрано реляційну СУБД Microsoft SQL Server, що забезпечує високий ступінь цілісності та відмовостійкості інформації.

Обґрунтування технологічного стека дозволило аргументувати застосування трирівневої архітектури системи. Реалізація клієнтської частини виконана з використанням технології Windows Presentation Foundation (WPF) та патерну проектування MVVM, що забезпечило високу модульність коду, розмежування рівнів представлення та бізнес-логіки, а також ергономічність інтерфейсу.

На етапі програмної реалізації та верифікації розроблено й протестовано функціональні модулі, зокрема підсистеми оприбуткування ТМЦ, інтелектуального

розподілу запчастин за складськими комірками, автоматизації внутрішньої логістики та опрацювання заявок технічного персоналу. Впроваджено механізми валідації вхідних даних та обробки виняткових ситуацій.

Таким чином, розроблена структура та функціональні компоненти АРМ управління складом СТО підтвердили свою відповідність поставленим технічним вимогам. Створена інформаційна система має потенціал для подальшого масштабування та впровадження у реальний сектор економіки як інструмент підвищення операційної ефективності підприємств автомобільної галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Марков О. Д. Організація автосервісу : навч. посіб. Львів: Оріяна-Нова, 2018. 332 с.
2. Канарчук В. Є., Лудченко О. А. Основи технічного обслуговування і ремонту автомобілів: підручник. Київ : Вища школа, 2019. 384 с.
3. Лудченко О. А. Технічне обслуговування і ремонт автомобілів: організація і управління : навч. посіб. Київ: Знання, 2017. 478 с.
4. Волгін В. В. Автосервіс. Створення і компоненти. Київ: Либідь, 2020. 408 с.
5. Фастовцев Г. Ф. Організація технічного обслуговування і ремонту автомобілів. Москва: Машинобудування, 2018. 256 с.
6. Управління автосервісом: навч. посіб. / за ред. Л. Б. Миротіна. Київ: Екзамен, 2021. 464 с.
7. Tocan WMS – система управління складом. URL: https://tocan.com.ua/uk/sistema-upravleniya-skladom-tocan-wms/ (дата звернення: 12.05.2025).
8. LSC WMS – система управління складською логістикою. URL: https://lscenter.com.ua/ua/logistics-optimization/lsc-wms (дата звернення: 12.05.2025).
9. Модель «сутність – зв'язок». Основні поняття моделі «сутність – зв'язок»: сутності, зв'язки, атрибути та їх класифікація. URL: посилання (дата звернення: 12.05.2025).
10. Діаграма класів. Вікіпедія – Вільна енциклопедія. URL: [Електронний ресурс] – Режим доступу: посилання (дата звернення: 12.05.2025).
11. Що таке діаграма класів UML і найкращий творець діаграм класів UML. URL: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення: 12.05.2025).
12. Повне розуміння діаграми компонентів UML за допомогою легкого методу. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/ (дата звернення: 14.05.2025).

13. XAML overview. URL: https://learn.microsoft.com/uk-ua/dotnet/desktop/wpf/xaml/ (дата звернення: 14.05.2025).
14. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань. Львів : Магнолія-2006, 2018. 584 с.
15. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ : Видавнича група BHV, 2016. 384 с.
16. Що таке SQL і де його використовують: веб-сайт. URL: https://goit.global/ua/articles/shcho-take-sql-i-de-yoho-vykorystovuiut/ (дата звернення: 14.05.2025).
17. .NET. Вікіпедія – Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/.NET (дата звернення: 14.05.2025).
18. Завантаження пакета DirectX і його інсталяція: веб-сайт. URL: посилання (дата звернення: 14.05.2025).
19. SQL Server Database Engine. Microsoft SQL Server Documentation: веб-сайт. URL: https://learn.microsoft.com/uk-ua/sql/database-engine/sql-server-database-engine-overview (дата звернення: 14.05.2025).
20. Data Binding Overview. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/uk-ua/dotnet/desktop/wpf/data/?view=netdesktop-8.0 (дата звернення: 14.05.2025).
21. ADO.NET Overview. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/uk-ua/dotnet/framework/data/adonet/ado-net-overview (дата звернення: 14.05.2025).
22. Огляд архітектури клієнт-сервер. Microsoft Learn : веб-сайт. URL: https://learn.microsoft.com/uk-ua/dotnet/architecture/modern-web-apps-azure/common-client-side-web-technologies (дата звернення: 14.05.2025).
23. Smith J. Patterns: WPF Apps With the Model-View-ViewModel Design Pattern. MSDN Magazine. 2009. URL: https://docs.microsoft.com/uk-ua/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern (дата звернення: 14.05.2025).

24. SQL Server technical documentation / Microsoft Corporation. 2023. URL: https://docs.microsoft.com/uk-ua/sql/ (дата звернення: 14.05.2025).
25. Windows Presentation Foundation documentation / Microsoft Corporation. 2023. URL: https://docs.microsoft.com/uk-ua/dotnet/desktop/wpf/ (дата звернення: 14.05.2025).
26. C# documentation / Microsoft Corporation. 2023. URL: https://docs.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 14.05.2025).
27. Тестування програмного забезпечення. Foxminded: веб-сайт. URL: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/ (дата звернення: 15.05.2025).

ДОДАТОК А

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

КОМЕНДИ АРТЕМА ВАДИМОВИЧА

Наказ НУБіП України 3204 «С». 19.12.2025

ДОДАТОК Б

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій Декларація про академічну доброчесність та використання ШІ

Я, Коменда Артем Вадимович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему

«Програмне забезпечення інформаційно-управляючої системи для Станції Технічного Обслуговування»

є результатом моєї особистої праці.

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

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

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

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

«___»_____ 2026 р. _____ Артем Коменда

(підпис)