

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ МАГАЗИНУ**
(назва теми)
АВТОЗАПЧАСТИН

Виконав: студент 4 курсу, групи КНТ-512

Спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація) _____
«Комп'ютерна інженерія»

КУЗНЄЦОВ О.О.
(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.
(ПРИЗВИЩЕ та ініціали)

Рецензент ТЯГУНОВ Д.В.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«» квітня 202 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КУЗНЕЦОВА Олексія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи магазину автозапчастин

керівник проєкту (роботи) к.т.н., доцент, СКРУПСЬКИЙ С.Ю.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » квітня 2026 року № 160

2. Строк подання студентом проєкту (роботи) 27.05.2026 р.

3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області та обґрунтування розробки;

2) Проєктування комп'ютерної системи магазину автозапчастин;

3) Розробка та тестування комп'ютерної системи;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації (15 шт.)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	СКРУПСЬКИЙ С.Ю.		
Нормоконтроль	ДЬЯЧУК Т.С.		

7. Дата видачі завдання «14» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2026	
2	Визначення та аналіз вимог до програмного забезпечення	до 22.04.2026	
3	Розробка специфікації вимог SRS	до 24.04.2026	
4	Розробка діаграми варіантів використання	до 28.04.2026	
5	Розробка діаграми послідовності	до 01.05.2026	
6	Розробка описів прецедентів	до 05.05.2026	
7	Проектування архітектури програмного забезпечення	до 12.05.2026	
8	Проектування інтерфейсу користувача	до 22.05.2026	
9	Оформлення пояснювальної записки	до 25.05.2026	
10	Проходження нормоконтролю	до 01.06.2026	
11	Перевірка на наявність академічного плагіату	до 03.06.2026	
12	Проходження рецензування	до 10.06.2026	

Студент(ка)

(підпис)

КУЗНЕЦОВ О.О.

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

СКРУПСЬКИЙ С.Ю.

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 89 с., 46 рис., 3 табл., 2 додатка, 28 джерел.

КОМП'ЮТЕРНА СИСТЕМА, МАГАЗИН АВТОЗАПЧАСТИН,
АВТОМАТИЗАЦІЯ ТОРГІВЛІ, REST API, ASP.NET CORE, REACT,
TYPESCRIPT, БАЗА ДАНИХ, WEB-ЗАСТОСУНОК, СКЛАДСЬКИЙ ОБЛІК

Предметом розробки є комп'ютерна система автоматизації діяльності магазину автозапчастин.

Метою роботи є підвищення ефективності роботи магазину автозапчастин шляхом проєктування та реалізації комп'ютерної системи, що автоматизує основні торговельні та управлінські процеси та забезпечує зручний веб-інтерфейс для персоналу і клієнтів.

У роботі виконано аналіз предметної області та сучасних підходів до автоматизації торгівлі автозапчастинами, проведено огляд існуючих програмних рішень, визначено їхні переваги та недоліки, а також обґрунтовано доцільність розробки спеціалізованої системи для малого та середнього бізнесу.

Сформовано функціональні та технічні вимоги до системи, спроектовано архітектуру програмного забезпечення та структуру бази даних. Серверну частину реалізовано на платформі ASP.NET Core із використанням REST API, клієнтську частину – на основі React і TypeScript у вигляді SPA-застосунку.

Реалізовано основні модулі системи: управління товарами, категоріями, замовленнями та користувачами. Проведено тестування програмного забезпечення, яке підтвердило його працездатність та відповідність вимогам.

Результатом роботи є створена комп'ютерна система, що підвищує ефективність діяльності магазину автозапчастин, оптимізує облік товарів і покращує якість обслуговування клієнтів.

ЗМІСТ

Вступ.....	8
1 Аналіз предметної області та обґрунтування розробки	10
1.1 Актуальність автоматизації магазину автозапчастин	10
1.2 Роль інформаційних систем у сфері торгівлі автозапчастинами	12
1.3 Аналіз сучасних підходів до автоматизації торговельних процесів.....	15
1.4 Аналіз існуючих систем для торгівлі автозапчастинами.....	21
1.5 Висновки до розділу 1	26
2 Проєктування комп'ютерної системи магазину автозапчастин.....	27
2.1 Характеристика об'єкта дослідження	27
2.2 Функціональні та технічні вимоги до системи	29
2.3 Вибір та обґрунтування серверних технологій	33
2.4 Вибір та обґрунтування технологій клієнтської частини	37
2.5 Проєктування бази даних	40
2.6 Висновки до розділу 2	44
3 Розробка та тестування комп'ютерної системи.....	46
3.1 Реалізація серверної частини системи	46
3.2 Реалізація клієнтської частини системи	49
3.3 Інструкція адміністратора системи	51
3.4 Керівництво користувача системи	55
3.5 Тестування та оцінка працездатності системи.....	59
3.6 Висновки до розділу 3	61
Висновки	62

Перелік джерел посилання	64
Додаток А Код проєкту.....	67
Додаток Б Слайди презентації	82

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IC	– інформаційна система
RTK	– Redux Toolkit, інструментарій для управління станом і взаємодії з API
RTK Query	– модуль Redux Toolkit для організації запитів до серверної частини
REST	– архітектурний стиль побудови веб-сервісів
JSON	– текстовий формат обміну структурованими даними
DTO	– об'єкт передавання даних
SQL	– мова структурованих запитів для роботи з реляційними базами даних
MVC	– архітектурний шаблон Model–View–Controller
ORM	– об'єктно-реляційне відображення даних
EF Core	– фреймворк ORM для роботи з реляційними базами даних
JWT	– маркер автентифікації користувачів у форматі JSON Web Token
SPA	– односторінковий вебзастосунок
UI	– інтерфейс користувача
UX	– користувацький досвід взаємодії з системою
CRUD	– базові операції роботи з даними

ВСТУП

Розвиток інформаційних технологій кардинально змінює підходи до організації торговельної діяльності в усіх галузях економіки. Сфера торгівлі автозапчастинами не є винятком. Тут цифрова трансформація є особливо актуальною через специфічну складність галузі та зростаючу конкуренцію з боку великих онлайн-платформ.

Більшість малих і середніх магазинів автозапчастин в Україні досі функціонують або повністю без автоматизації, або з використанням примітивних засобів обліку, це не дозволяє ефективно управляти підприємством і надавати клієнтам сучасний рівень сервісу. Водночас існуючі спеціалізовані програмні рішення, представлені на ринку, мають суттєві обмеження: більшість із них є або надто дорогими для малого бізнесу, або не мають публічного клієнтського веб-інтерфейсу.

За таких умов розробка спеціалізованої комп'ютерної системи магазину автозапчастин, яка поєднує зручний інтерфейс для персоналу та покупців, повноцінний складський облік і аналітичні інструменти, є практично значущим і своєчасним завданням.

Метою роботи є підвищення ефективності торговельної та управлінської діяльності магазину автозапчастин за рахунок проєктування та розробки спеціалізованої комп'ютерної системи, яка автоматизує ключові бізнес-процеси підприємства та забезпечує зручний веб-інтерфейс для персоналу і покупців.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз предметної області та обґрунтувати актуальність розробки;
- здійснити огляд існуючих програмних рішень для торгівлі автозапчастинами та виявити їх недоліки;
- визначити функціональні та нефункціональні вимоги до розроблюваної системи;

- обґрунтувати вибір технологічного стеку серверної та клієнтської частин;
- спроектувати реляційну базу даних системи;
- реалізувати серверну частину на платформі ASP.NET Core з використанням REST API архітектури;
- розробити клієнтську частину у вигляді SPA-застосунку на базі React та TypeScript;
- провести тестування системи та підтвердити її відповідність сформульованим вимогам.

Об'єктом дослідження є процеси автоматизації торговельної діяльності магазину автозапчастин.

Предметом дослідження є методи, засоби та технології розробки комп'ютерних систем для автоматизації торговельних підприємств галузі продажу автозапчастин.

Методи дослідження. У роботі використано методи системного аналізу, порівняльного аналізу, об'єктно-орієнтованого проектування, практичної розробки програмного забезпечення.

Практична цінність роботи полягає в тому, що розроблена система вирішує реальну прикладну задачу автоматизації магазину автозапчастин і може бути впроваджена на підприємстві малого або середнього масштабу. Система охоплює повний цикл торговельних операцій.

Актуальність даної роботи полягає у необхідності розробки комплексного програмного рішення для автоматизації роботи магазину автозапчастин, яке поєднує клієнтську веб-платформу, адміністративну панель та аналітичні інструменти для керівництва. Це дозволяє не лише оптимізувати внутрішні процеси підприємства, а й підвищити рівень сервісу для кінцевих користувачів.

Виконане дослідження спрямоване на розв'язання актуальної науково-прикладної задачі – удосконалення управління магазином автотоварів за рахунок впровадження ком'ютерних систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ

1.1 Актуальність автоматизації магазину автозапчастин

Сучасний ринок автозапчастин є одним із найбільш динамічних і конкурентних сегментів роздрібної торгівлі. За даними аналітичних досліджень, автомобільний парк України налічує понад 12 мільйонів транспортних засобів [1]. Та навіть попри складну економічну ситуацію останніх років в Україні, попит на запасні частини та витратні матеріали залишається стабільно високим. Власники автомобілів змушені регулярно звертатися до магазинів автозапчастин для проведення технічного обслуговування та ремонту. Ці чинники формують постійний і передбачуваний потік клієнтів. В той же час специфіка цього ринку висуває надзвичайно жорсткі вимоги до організації торговельних процесів, управління асортиментом та обслуговування покупців.

Ключовою особливістю торгівлі автозапчастинами є дуже велика номенклатура товарів. Типовий магазин середнього масштабу може одночасно оперувати сотнями найменувань продукції. Це включає деталі для автомобілів різних марок, моделей, років випуску та модифікацій. Кожна позиція характеризується унікальними ідентифікаторами.

Управління такою номенклатурою вручну або з використанням примітивних засобів обліку, практично унеможливорює оперативне та якісне обслуговування клієнтів.

В контексті цієї теми, варто звернути увагу на проблему підбору запчастин за сумісністю. На відміну від більшості товарних категорій, у торгівлі автозапчастинами покупець рідко знає точний артикул необхідної деталі. Клієнт знає лише марку, модель та рік випуску свого автомобіля, а також назву несправної частини. Менеджер магазину повинен в режимі реального часу ідентифікувати відповідну позицію в каталозі, перевірити наявність на складі, запропонувати аналоги від інших виробників за різними ціновими категоріями. Без автоматизованої системи пошуку та підбору запчастин цей процес займає значно

більше часу, супроводжується помилками та негативно впливає на задоволеність клієнтів та прибутковість підприємства.

Іншою проблемою даного сегменту торгівлі, є управління товарними запасами. Надлишкові залишки заморожують обігові кошти підприємства, займають складські площі та несуть ризик втрати актуальності. Дефіцит же популярних позицій призводить до втрати клієнтів, які звертаються до конкурентів. Балансування між ними вимагає систематичного аналізу статистики продажів, сезонних коливань попиту, термінів постачання від постачальників. Ефективно аналізувати такий обсяг даних без відповідної комп'ютерної підтримки є досить складним завданням.

Важливим фактором, який підсилює актуальність автоматизації, є зростання конкуренції з боку великих мережових гравців та інтернет-магазинів.

Великі платформи такі як Rozetka, Avtopro, AutoBaza, пропонують покупцям зручний онлайн-пошук запчастин, порівняння цін, перегляд наявності та швидку доставку. Невеликі та середні магазини змушені тепер впроваджувати власні цифрові рішення. Це дає можливість запропонувати клієнтам приблизно такий самий рівень сервісу.

Автоматизація торговельних процесів безпосередньо впливає на економічну ефективність підприємства. Дослідження у сфері роздрібної торгівлі свідчать, що впровадження спеціалізованих інформаційних систем дозволяє скоротити час обслуговування одного клієнта в середньому на 30-40%. Це суттєво допомагає знизити кількість помилок при оформленні замовлень і накладних, оптимізувати товарні запаси завдяки аналітичним інструментам прогнозування попиту [2].

Автоматизація бухгалтерського та складського обліку дозволяє суттєво зменшити адміністративне навантаження на персонал і мінімізувати людський фактор як джерело помилок.

Слід також звернути увагу на законодавчий та регуляторний аспект. Сучасні вимоги до ведення господарської діяльності в Україні передбачають обов'язкову фіскалізацію розрахунків, ведення первинної документації у встановлених форматах, подання звітності до контролюючих органів. Ручне ведення обліку в

умовах значного товарообігу несе великі ризики порушення цих вимог. Інтегровані комп'ютерні системи здатні автоматично формувати необхідні документи, вести облік відповідно до чинних стандартів. Це дозволяє знижувати правові ризики для підприємства.

У сфері торгівлі працівники досить часто змінюються, тому без автоматизованої системи багато важливої інформації зберігається лише в пам'яті окремих співробітників. Це можуть бути дані про постачальників, клієнтів або особливості підбору запчастин. Коли працівник звільняється, підприємство може втратити частину цих знань. Використання комп'ютерної системи дозволяє зберігати всю необхідну інформацію в єдиній базі, що полегшує навчання нових працівників і робить роботу підприємства менш залежною від конкретних людей.

Актуальність розробки комп'ютерної системи для магазину автозапчастин обумовлена сукупністю взаємопов'язаних факторів:

- складністю номенклатури та необхідністю швидкого пошуку товарів;
- потребою в ефективному управлінні складськими запасами;
- зростаючою конкуренцією з боку цифрових торговельних платформ;
- тенденцією до цифровізації бізнес-процесів у всіх галузях економіки.

Розробка спеціалізованої комп'ютерної системи, яка враховує галузеву специфіку торгівлі автозапчастинами, є необхідною умовою успішного функціонування сучасного підприємства у цій сфері.

1.2 Роль інформаційних систем у сфері торгівлі автозапчастинами

Інформаційна система у сфері торговельного підприємства є важливим інструментом для організації та автоматизації основних бізнес-процесів. Вона забезпечує обробку, зберігання та аналіз інформації, пов'язаної з товарами, продажами, клієнтами, постачальниками та складським обліком. У сучасних умовах використання інформаційних систем дозволяє підвищити ефективність

роботи підприємства.

У загальному розумінні інформаційна система – це система, яка надає інформацію зацікавленим у ній особам та включає до себе інформаційну мережу, людські, фінансові, технічні ресурси [3]. Це визначення надзвичайно широке. Воно охоплює як найпростіші системи такі як картотеки на паперових носіях, так і складні розподілені програмні комплекси.

У практичному контексті торговельного підприємства під інформаційною системою розуміють насамперед програмний комплекс, що автоматизує обробку інформації, яка виникає в процесі господарської діяльності:

- дані про товари;
- клієнтів;
- замовлення;
- рух коштів;
- складські залишки.

Головна відмінність сучасної комп'ютеризованої ІС від традиційного паперового обліку полягає не просто у швидкості обробки даних, а у якісно іншому рівні управлінських можливостей, які вона надає.

Будь-яка інформаційна система торговельного підприємства, незалежно від масштабу і галузі, складається з кількох взаємопов'язаних підсистем.



Рисунок 1.1 – Типова структура інформаційної системи магазину

Інформаційна система магазину автозапчастин повинна забезпечувати автоматизацію основних бізнес-процесів підприємства та підтримувати ефективне управління торговельною діяльністю. Специфіка сфери продажу автозапчастин визначає особливі вимоги до функціональних можливостей такої системи.

Однією з ключових задач інформаційної системи є управління номенклатурою товарів. Магазины автозапчастин можуть містити десятки тисяч позицій. У зв'язку з цим система повинна забезпечувати швидкий пошук товарів за різними параметрами. Автоматизація цього процесу значно скорочує час пошуку необхідної деталі та підвищує якість обслуговування клієнтів.

Важливою задачею є також управління складськими залишками в режимі реального часу. Інформаційна система повинна забезпечувати постійний контроль кількості товарів на складі. Це дозволяє уникнути помилок, пов'язаних із продажем відсутніх товарів, а також сприяє оптимізації складських запасів і зменшенню фінансових втрат через надлишкове накопичення продукції.

Наступною задачею є автоматизація процесу продажу та супровідного документообігу. Такі можливості дозволяють покращити якість сервісу, підвищити рівень лояльності покупців і забезпечити ефективну взаємодію з постійними клієнтами, зокрема станціями технічного обслуговування та корпоративними замовниками.

Ще однією важливою задачею є формування аналітичної інформації та підтримка управлінських рішень. Отримана аналітична інформація використовується для прогнозування попиту, планування закупівель, оптимізації асортименту та оцінки результативності роботи магазину. Завдяки цьому управлінські рішення приймаються на основі актуальних даних, а не лише суб'єктивних оцінок.

У теорії інформаційних систем прийнято виділяти кілька рівнів автоматизації торговельного підприємства, які відрізняються між собою за ступенем охоплення бізнес-процесів та складністю реалізації програмного забезпечення [4]. Для сфери торгівлі автозапчастинами найбільш доцільно розглядати три взаємопов'язані рівні, які формують цілісну структуру інформаційної системи:

- операційний (front-office) – охоплює процеси безпосередньої взаємодії з клієнтом. До нього належать пошук та підбір товарів, оформлення продажів, проведення розрахункових операцій, а також формування і друк супровідної документації. Саме цей рівень є найбільш наближеним до користувача та визначає швидкість і якість обслуговування;

- управлінський (back-office) – забезпечує внутрішню діяльність підприємства та підтримує роботу операційного рівня. Він включає управління складськими залишками, облік надходжень і повернень товарів, взаємодію з постачальниками, регулювання цінової політики та інші процеси внутрішнього обліку. Цей рівень є основою стабільного функціонування всієї системи, хоча і не є безпосередньо видимим для клієнта;

- аналітичний – призначений для підтримки управлінських рішень на основі накопичених даних. Він передбачає формування звітності щодо продажів, аналіз рентабельності товарних груп, прогнозування попиту та контроль ключових показників ефективності діяльності підприємства. Реалізація цього рівня стає можливою лише за умови достатнього накопичення та структуризації даних на попередніх рівнях автоматизації.

Таким чином, розглянуті рівні автоматизації відображають багаторівневу структуру інформаційної системи торговельного підприємства, де кожен рівень виконує власні функції та у взаємодії забезпечує комплексну підтримку операційної, управлінської та аналітичної діяльності магазину автозапчастин.

1.3 Аналіз сучасних підходів до автоматизації торговельних процесів

Автоматизація торговельних процесів є комплексним завданням, що охоплює широкий спектр технологічних рішень. Ринок програмного забезпечення для торгівлі є досить об'ємний та включає в себе велику кількість існуючих підходів.

Одним із найбільш комплексних підходів до автоматизації діяльності підприємства є використання ERP-систем. ERP являє собою інтегровані інформаційні системи, призначені для автоматизації та координації основних бізнес-процесів організації в межах єдиного інформаційного середовища. Концепція ERP-систем сформувалася як розвиток систем класу MRP та MRP II і на сьогодні є одним із найбільш поширених підходів до автоматизації середніх і великих підприємств [5].

Основною особливістю ERP-систем є використання єдиної бази даних для всіх функціональних модулів системи. Використання спільного інформаційного середовища дозволяє уникнути дублювання даних та забезпечує актуальність інформації в режимі реального часу.

Для магазину автозапчастин використання ERP-системи дає можливість автоматизувати взаємопов'язані процеси торговельної діяльності. Такий підхід сприяє підвищенню ефективності управління підприємством і зменшенню кількості помилок, пов'язаних із ручною обробкою інформації.

Разом із перевагами ERP-системи мають і недоліки. Одним із основних обмежень є висока вартість впровадження та супроводу таких рішень. Використання систем потребує значних фінансових витрат на придбання програмного забезпечення, налаштування системи, технічний супровід і навчання персоналу. Крім того, ERP-системи часто містять значну кількість функціональних модулів, частина яких може залишатися невикористаною на малих підприємствах.

На українському ринку серед ERP-рішень для торговельних підприємств поширення отримали системи BAS Управління торгівлею та Microsoft Dynamics 365. Проте для більшості невеликих магазинів автозапчастин впровадження подібних систем залишається економічно складним.

Іншим поширеним підходом до автоматизації торговельної діяльності є використання POS-систем.

На відміну від ERP-систем, які охоплюють практично всі бізнес-процеси підприємства, POS-системи орієнтовані переважно на автоматизацію касових операцій та процесу оформлення продажів.

Типова POS-система являє собою апаратно-програмний комплекс, до складу якого входять касова програма, сенсорний термінал, сканер штрих-кодів, програмний реєстратор розрахункових операцій, принтер чеків та платіжний термінал для безготівкових розрахунків. Основним призначенням такої системи є швидке оформлення продажу, проведення розрахунків із покупцем та формування фіскальних документів [6].

Для України важливим аспектом функціонування POS-систем є підтримка програмних РРО, оскільки відповідно до чинного законодавства всі розрахункові операції повинні проходити процедуру фіскалізації з передачею даних до органів ДПС у режимі реального часу. У зв'язку з цим сучасні POS-рішення повинні підтримувати інтеграцію з сервісами фіскалізації та забезпечувати коректне формування електронних чеків.

Основним недоліком POS-систем є їх обмежена функціональність у контексті комплексного управління підприємством. Такі системи ефективно вирішують задачі реєстрації продажів та проведення розрахункових операцій, проте зазвичай не забезпечують повноцінних можливостей для аналітики, управління закупівлями, CRM-функцій або організації онлайн-продажів. Для реалізації подібних можливостей необхідне використання додаткових програмних модулів або інтеграцій із зовнішніми системами.

Важливе місце серед засобів автоматизації торговельної діяльності займають CRM-системи. Вони на відміну від ERP-систем, які переважно орієнтовані на автоматизацію внутрішніх бізнес-процесів підприємства, CRM-системи спрямовані на організацію ефективної взаємодії з клієнтами, підвищення якості обслуговування та формування довгострокових відносин із покупцями [7].

У сфері торгівлі автозапчастинами CRM-функціонал має особливе значення через специфіку взаємодії з клієнтами. Якість підбору необхідних деталей, оперативність обслуговування та врахування індивідуальних потреб покупця безпосередньо впливають на рівень лояльності клієнтів і ймовірність повторних звернень до магазину. У зв'язку з цим важливими функціями CRM-систем є збереження історії покупок, ведення клієнтської бази, фіксація інформації про

транспортні засоби клієнтів, а також підтримка програм лояльності та систем персоналізованих знижок.

Для магазинів автозапчастин особливо актуальною є можливість прив'язки клієнта до конкретного автомобіля із збереженням таких характеристик, як марка, модель, рік випуску та VIN-номер. Це дозволяє значно спростити подальший підбір запчастин і підвищити точність обслуговування. Крім того, система може використовувати накопичену інформацію для нагадувань про планове технічне обслуговування або повторні покупки певних товарів.

Базовий функціонал CRM-системи для автомагазину зазвичай включає ведення бази клієнтів із контактною інформацією та історією замовлень, підтримку бонусних і дисконтних програм, а також облік дебіторської заборгованості для корпоративних клієнтів, станцій технічного обслуговування та юридичних осіб, які працюють із відстроченою оплатою.

Разом із цим більшість універсальних CRM-систем недостатньо враховують специфіку сфери продажу автозапчастин. У зв'язку з цим доцільним підходом є інтеграція CRM-функціоналу безпосередньо в загальну торговельну систему підприємства, що дозволяє забезпечити єдине інформаційне середовище та підвищити ефективність взаємодії між окремими модулями системи.

Окрему увагу в сучасних умовах розвитку інформаційних технологій слід приділити SaaS-моделі, яка є одним із найбільш поширених підходів до постачання програмного забезпечення (рис. 1.2). На відміну від традиційної моделі використання програмних продуктів, за якої підприємство самостійно встановлює систему на власному обладнанні та забезпечує її технічну підтримку.



Рисунок 1.2 – Схема сучасної SaaS-моделі

SaaS-модель передбачає надання доступу до програмного забезпечення через мережу Інтернет за підпискою.

У межах даного підходу програмне забезпечення розміщується на серверах постачальника послуг, а користувач отримує доступ до системи через веб-браузер або спеціалізований клієнтський застосунок. Така модель дозволяє підприємствам уникнути значних початкових витрат на придбання серверного обладнання, ліцензійного програмного забезпечення та створення власної IT-інфраструктури.

Для малого та середнього бізнесу SaaS-модель має низку суттєвих переваг. Це зниження початкових фінансових витрат, оскільки оплата здійснюється у формі щомісячної або річної підписки. Крім того, відсутня необхідність у самостійному технічному супроводі системи, адмініструванні серверів та встановленні оновлень, оскільки ці функції виконує постачальник послуг. Важливою перевагою також є можливість швидкого масштабування системи шляхом додавання нових користувачів або функціональних модулів без складних технічних процедур.

Для магазинів автозапчастин SaaS-рішення є особливо актуальними через можливість організації віддаленого доступу до системи, синхронізації даних між кількома торговими точками та забезпечення мобільного доступу для працівників підприємства.

Одним із головних обмежень є залежність роботи системи від стабільного доступу до мережі Інтернет. У разі виникнення проблем із з'єднанням доступ до програмного забезпечення може бути тимчасово втрачений, що негативно впливає на роботу підприємства. Особливої уваги потребують питання інформаційної безпеки, оскільки дані підприємства зберігаються на серверах стороннього провайдера. Це створює ризики, пов'язані із захистом конфіденційної інформації та контролем над даними.

Тісно пов'язаною із SaaS-моделлю є концепція хмарних обчислень, яка має значно ширший характер. Хмарні технології загалом охоплюють надання різних видів обчислювальних ресурсів, зокрема серверних потужностей, систем зберігання даних, мережевої інфраструктури та програмних платформ. Відповідно до класифікації NIST [8]. Виділяють три основні моделі хмарних сервісів:

- IaaS (Infrastructure as a Service);
- PaaS (Platform as a Service);
- SaaS (Software as a Service).

Для розробки сучасних інформаційних систем хмарні технології мають важливе значення насамперед як інфраструктурна основа для розгортання серверної частини програмного забезпечення. Використання хмарних платформ, таких як Amazon Web Services, Google Cloud, Microsoft Azure, а також українських провайдерів Tucha і GigaCloud, дозволяє забезпечити гнучке масштабування системи відповідно до навантаження.

Важливим елементом сучасної хмарної інфраструктури є контейнеризація застосунків, зокрема із використанням технології Docker. Контейнеризація забезпечує створення ізольованого та відтворюваного середовища виконання програмного забезпечення, що дозволяє уникнути проблем сумісності між середовищем розробки та серверною інфраструктурою. Завдяки використанню контейнерів процеси розгортання, оновлення та масштабування інформаційної системи стають більш стандартизованими, керованими та передбачуваними.

Застосування хмарних технологій і контейнеризації є особливо доцільним для малих і середніх підприємств, оскільки дозволяє зменшити витрати на підтримку власної серверної інфраструктури та спростити адміністрування програмного забезпечення.

Мабуть, однією з ключових тенденцій останнього десятиліття у розробці торговельних інформаційних систем є перехід від монолітних десктопних рішень до веб-орієнтованих систем із розподіленою архітектурою. Така трансформація зумовлена розвитком мережових технологій, необхідністю забезпечення багатокористувацького доступу та підтримки роботи з різних пристроїв.

У сучасній практиці поширеним є підхід до поділу системи на клієнтську (frontend) та серверну (backend) частини, які взаємодіють через REST API. REST (Representational State Transfer) базується на використанні стандартних HTTP-методів для обміну даними між компонентами системи та забезпечує простоту інтеграції, масштабованість і технологічну незалежність клієнтських застосунків.

Для торговельних систем, такий підхід є доцільним, оскільки дозволяє використовувати єдиний серверний інтерфейс для різних типів клієнтів: адміністративної панелі, веб-каталогу та можливих мобільних застосунків. Це спрощує супровід і розвиток системи.

Клієнтська частина зазвичай реалізується у вигляді SPA (Single Page Application), що забезпечує динамічне оновлення даних без повного перезавантаження сторінок. Такий підхід підвищує продуктивність інтерфейсу та покращує користувацький досвід.

1.4 Аналіз існуючих систем для торгівлі автозапчастинами

Аналіз існуючих програмних рішень, представлених на ринку, є важливим етапом перед проектуванням власної ком'ютерної системи. Це дозволяє виявити сильні та слабкі сторони наявних продуктів, визначити функціональні прогалини, які вони не закривають. Обґрунтувати доцільність розробки власного рішення. На українському та міжнародному ринках представлено кілька класів програмного забезпечення для автоматизації торгівлі автозапчастинами:

- спеціалізовані галузеві системи;
- універсальні ERP-системи з модулями для торгівлі;
- хмарні SaaS-рішення та POS-системи з можливістю розширення функціоналу.

AutoSelling – це професійна система обліку для автомагазинів, СТО та роздрібної торгівлі, яка вже містить понад 25 мільйонів унікальних запчастин. Система орієнтована виключно на автомобільну галузь і включає інтегрований каталог TecDoc, підбір аналогів у реальному часі, прив'язку деталей до конкретних марок і моделей автомобілів, а також модуль для роботи СТО з графіком та записом клієнтів [9]. Інтерфейс системи AutoSelling наведено на рисунку 1.3.

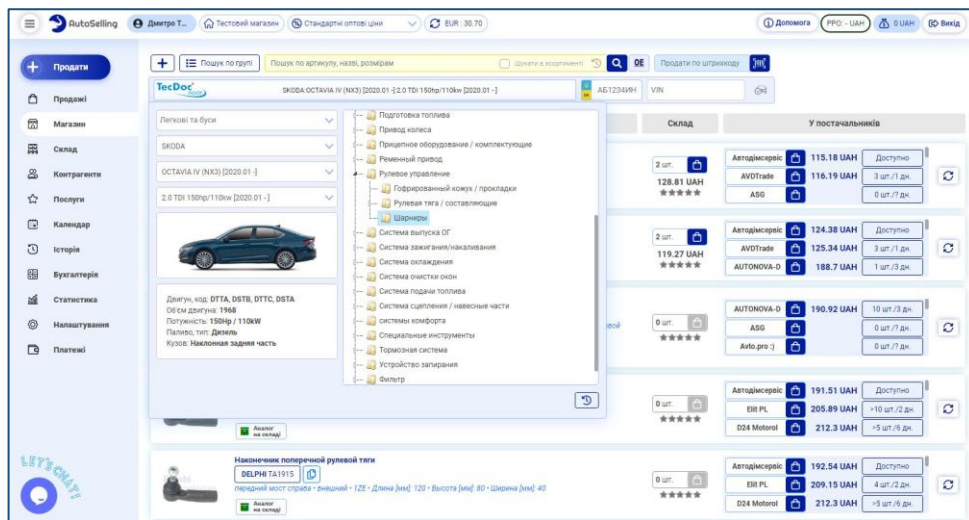


Рисунок 1.3 – Інтерфейс системи AutoSelling

До переваг системи AutoSelling можна віднести її спеціалізацію саме для сфери продажу автозапчастин, наявність великого каталогу товарів із фотографіями, підтримку крос-номерів та аналогів деталей, а також можливість використовувати безкоштовний програмний РРО. Ще однією перевагою є хмарний доступ, завдяки якому працювати із системою можна з будь-якого пристрою.

Проте система має і певні недоліки. Вона більше орієнтована на великі підприємства, тому для невеликих магазинів вартість підписки може бути досить високою. Крім цього, робота AutoSelling залежить від стабільного інтернет-з'єднання та серверної інфраструктури провайдера.

Іншим прикладом є облікова програма Торгсофт, що підходить для малих і великих автомагазинів. Система пропонує широкий спектр функцій.

Торгсофт є одним із найпоширеніших рішень серед українських магазинів автозапчастин завдяки адаптованості до вимог вітчизняного законодавства та прийнятній вартості ліцензії. До переваг системи Торгсофт можна віднести підтримку українського законодавства, доступну цінову політику, наявність як десктопної, так і хмарної версії програми. Також система підтримує інтеграцію з різними торговим обладнанням, що робить її зручною для використання в магазинах.

Серед недоліків варто відзначити дещо застарілий дизайн інтерфейсу та обмежені можливості веб-доступу для клієнтів. Крім цього, у системі немає

повноцінного онлайн-каталогу для покупців, а при розширенні мережі магазинів можуть виникати труднощі з масштабуванням системи.

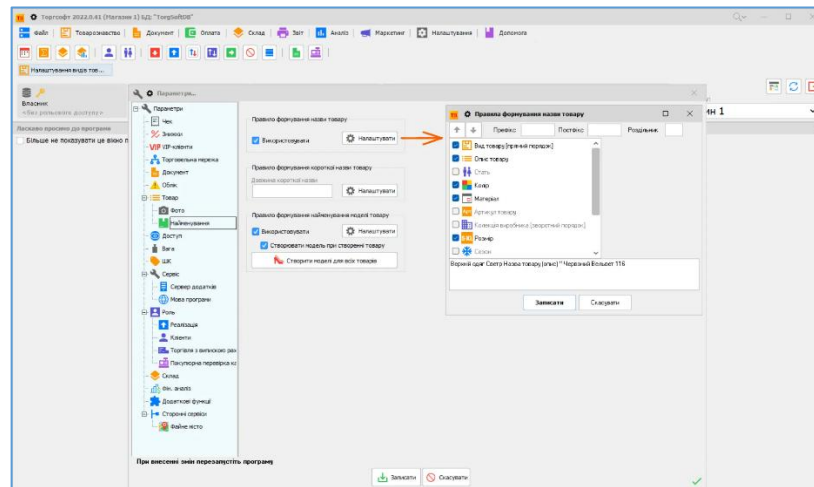


Рисунок 1.4 – Функціонал Торгософт

BAS «Роздрібна торгівля» – програмний продукт на платформі 1С для повного товарного й касового обліку. Він підходить для магазинів із великими обсягами продажів. Продукт надає можливість підключити розподілену інформаційну базу для синхронізації даних між торговими точками. Система охоплює POS-касу, складський облік, CRM та програми лояльності [10].

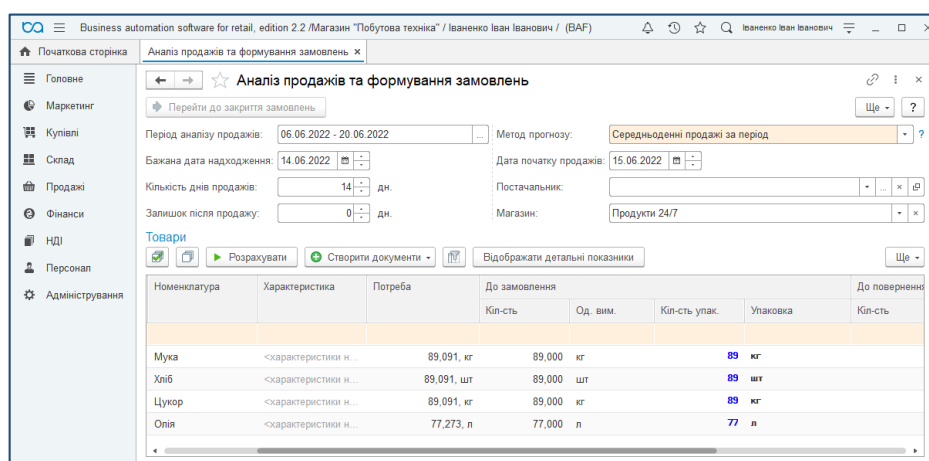


Рисунок 1.5 – Інтерфейс системи BAS Роздрібна торгівля

Ще одна відома хмарна система обліку автозапчастин це – ERP FOSS. Вона використовується для управління автомагазином онлайн. Має зручний інтерфейс і

допомагає автоматизувати продажі та закупівлі. Також система підтримує формування бухгалтерської та фінансової звітності. До переваг можна віднести повністю хмарну архітектуру. Система інтегрується з українськими сервісами, такими як Prom.ua, Нова Пошта та Приват24. Є можливість мобільного доступу. Також присутній довідник номенклатури з аналогами та крос-кодами. Недоліком є менш розвинений каталог у порівнянні з AutoSelling. Також відсутній повноцінний інтернет-магазин для клієнтів.

USAP.ONLINE – сервіс для обліку й контролю робочих процесів, створений на основі хмарних технологій. Його не треба встановлювати чи довго налаштовувати. Доступ до даних здійснюється онлайн із будь-якого пристрою, підключеного до інтернету. Для кожного товару є декілька унікальних ідентифікаторів, за якими його можна знайти. Система також підтримує інтеграцію з СТО, облік виконаних робіт та виставлення рахунків.

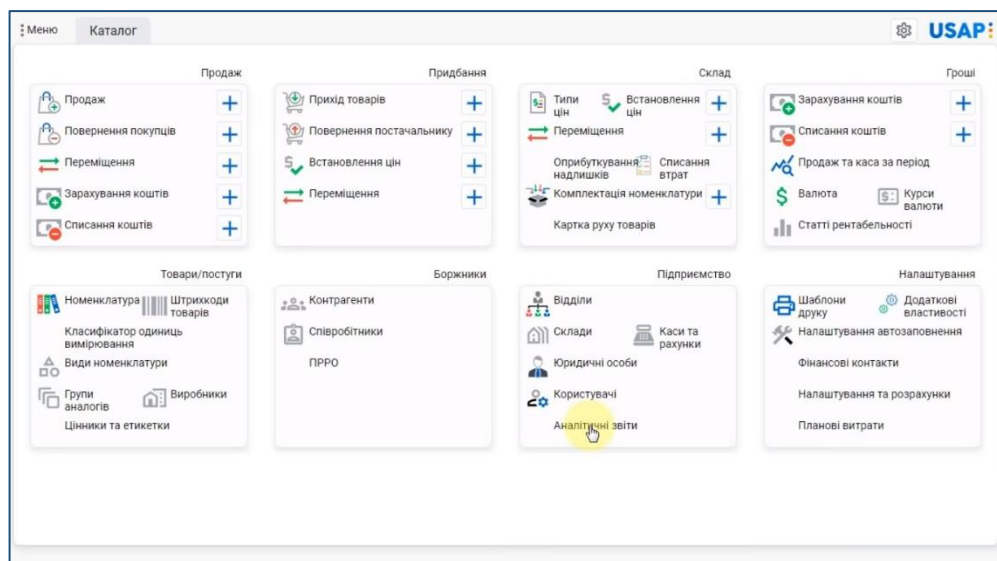


Рисунок 1.6 – Вигляд системи USAP.ONLINE

Для узагальнення результатів проведеного аналізу було створено порівняльну таблицю. Де було порівняно аналоги за основними критеріями, які є найбільш важливими для роботи магазину автозапчастин. Зокрема, враховувалися функціональні можливості, зручність використання, наявність інтеграцій, підтримка онлайн-торгівлі, а також загальна орієнтація системи на потреби бізнесу.

Таблиця 1.1 – Порівняльний аналіз існуючих систем автоматизації

Критерій	AutoSelling	Торгсофт	BAS Роздрібна торгівля	ERP FOSS	USAP.ONLINE
Галузева спеціалізація	Висока	Середня	Низька	Середня	Середня
Інтеграція ТесДос	Так	Частково	Ні	Частково	Ні
Веб-каталог для покупців	Ні	Ні	Ні	Ні	Ні
Хмарний доступ	Так	Частково	Частково	Так	Так
Управління складом	Так	Так	Так	Так	Так
CRM	Частково	Так	Так	Частково	Частково
Інтеграція з Новою Поштою	Ні	Ні	Ні	Так	Частково
Фіскалізація	Так	Так	Так	Так	Так
Мобільний доступ	Так	Частково	Ні	Так	Так
Відкритий вихідний код	Ні	Ні	Ні	Ні	Ні
Вартість	Висока	Середня	Висока	Середня	Низька
Адаптація під конкретний бізнес	Низька	Низька	Висока	Низька	Низька

Аналіз дозволяє виділити кілька основних закономірностей:

- жодна з розглянутих систем не має повноцінного публічного веб-каталогу для клієнтів. Через це покупці не можуть самостійно шукати та замовляти запчастини без участі менеджера;
- більшість рішень є продуктами з обмеженими можливостями налаштування. Для зміни логіки роботи системи або додавання нових функцій та інтеграцій часто потрібно залучати розробників, що призводить до додаткових витрат;
- системи, які мають найбільш розвинену галузеву спеціалізацію, зазвичай є найдорожчими та менш гнучкими у налаштуванні.

Таким чином, аналіз показує, що існує певна незайнята ніша на ринку. Це система для магазину автозапчастин, яка поєднує в собі галузеву спеціалізацію, веб-інтерфейс для клієнтів із можливістю перегляду каталогу та оформлення замовлень, повноцінний адміністративний модуль для управління товарами і складом.

1.5 Висновки до розділу 1

У першому розділі було розглянуто теоретичні та практичні засади, розробки комп'ютерних систем магазину автозапчастин.

Аналіз ринку показав, що автоматизація торгівлі автозапчастинами необхідною умовою конкурентоспроможності підприємства. Велика номенклатура товарів, специфіка підбору запчастин за сумісністю, необхідність точного складського обліку та зростання конкуренції, формує чіткий запит на спеціалізоване програмне рішення.

Огляд існуючих систем виявив, що жодна з них не забезпечує повного поєднання галузевої спеціалізації, клієнтського веб-інтерфейсу та гнучкості у налаштуванні під конкретне підприємство. Більшість рішень є або надто дорогими, або функціонально обмеженими, або не мають публічного каталогу для покупців.

Аналіз підходів до автоматизації торговельних процесів показав, що найбільш обґрунтованим для задач цієї роботи є розробка власної веб-орієнтованої системи на базі REST API архітектури з SPA-клієнтом, яка поєднує переваги ERP, POS та CRM підходів без їхніх характерних недоліків.

2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МАГАЗИНУ АВТОЗАПЧАСТИН

2.1 Характеристика об'єкта дослідження

Невеликі магазини автозапчастин, як правило, належать до категорії малих підприємств. Вони характеризуються простою організаційною структурою, обмеженою кількістю працівників та високим рівнем концентрації управлінських функцій у межах вузького кола осіб, найчастіше – власника або керівника. У таких умовах значна частина рішень приймається на основі особистого досвіду та практичного розуміння ситуації, що підвищує залежність результатів діяльності від індивідуальних управлінських рішень. Відповідно, такі підприємства змушені оперативно реагувати на зміни ринкового середовища, зокрема коливання попиту, цінової політики, умов постачання та поведінки споживачів, що ускладнює стабільне функціонування без належної організації інформаційних процесів.

Практика діяльності магазинів автозапчастин показує, що управлінські рішення часто формуються на основі неповної або фрагментованої інформації, а також особистого досвіду керівника. Такий підхід може бути прийнятним на початкових етапах розвитку бізнесу, однак із збільшенням обсягів продажів та асортименту він поступово знижує ефективність управління. Зокрема, відсутність систематизованого обліку ускладнює аналіз фінансових результатів, контроль асортименту та управління складськими залишками, що є особливо важливим для торговельної діяльності у сфері автозапчастин.

Об'єктом дослідження у межах даної роботи виступає мале підприємство роздрібної торгівлі автозапчастинами. Його діяльність охоплює процеси закупівлі товарів у постачальників, ведення складського обліку, зберігання та реалізацію продукції кінцевим споживачам, а також аналіз товарних і фінансових потоків. Особливістю даного сегмента є необхідність оперативного прийняття управлінських рішень, пов'язаних із формуванням запасів, реагуванням на сезонні коливання попиту та оптимізацією товарного асортименту на основі актуальних

даних.

Існуючі підходи до управління малими підприємствами можна умовно поділити на неформалізовані та частково формалізовані.

Перший підхід ґрунтується переважно на досвіді та інтуїції керівника і не передбачає систематичного збору та аналізу даних. Його перевагами є простота впровадження та низькі витрати, однак він не забезпечує достатнього рівня обґрунтованості управлінських рішень і практично не піддається масштабуванню.

Частково формалізований підхід передбачає використання облікових систем або електронних таблиць для фіксації даних, однак такі інструменти, як правило, не забезпечують повноцінної аналітичної підтримки процесів управління [11].

Недоліки зазначених підходів особливо проявляються у ситуаціях, коли необхідно оцінити альтернативні варіанти рішень або спрогнозувати наслідки управлінських дій. За відсутності інтегрованої інформаційної системи дані зберігаються у різних, часто ізольованих джерелах, що ускладнює їх подальший аналіз і знижує якість прийняття рішень. У зв'язку з цим виникає необхідність у розробці інформаційних систем підтримки прийняття рішень, адаптованих до специфіки малого бізнесу.

Діяльність малого торговельного підприємства генерує значний обсяг операційних даних, які потенційно можуть використовуватися для аналітичної обробки. До таких даних належать відомості про продажі, закупівлі, рух товарів, клієнтську поведінку та результати управлінських рішень. Однак без належної систематизації та структурування ці дані мають обмежену практичну цінність для управління підприємством.

Малі торговельні підприємства доцільно розглядати не лише як об'єкти управління, але й як об'єкти інформаційного моделювання. Такий підхід передбачає формалізацію ключових бізнес-процесів, інформаційних потоків та точок прийняття рішень, що є основою для проектування інформаційних систем підтримки управлінських рішень [12].

Узагальнені характеристики підприємства малого бізнесу наведено в таблиці 2.1.

Таблиця 2.1 – Основні характеристики підприємства у малому бізнесі

Характеристика	Опис
Сфера функціонування	Роздрібна торгівля автозапчастинами
Тип підприємства	Невелика фірма
Ключові бізнес-процеси	Постачання та придбання товарів, ведення складського обліку, здійснення продажів, оцінювання діяльності
Чисельність персоналу	До 10 осіб
Особливості управління	Централізована система управління з основною роллю власника або керівника підприємства
Інформаційне забезпечення	Наявність часткової автоматизації окремих процесів та розрізненого використання інформаційних систем
Характер управлінських рішень	Переважання поточних оперативних і короткострокових тактичних рішень
Основні управлінські труднощі	Обмежений доступ до аналітичних даних, залежність процесу прийняття рішень від особистого досвіду

Таким чином, аналіз показує, що об'єкт дослідження поєднує достатньо складні операційні процеси з обмеженим рівнем інформаційної підтримки.

2.2 Функціональні та технічні вимоги до системи

Формування вимог до інформаційної системи є одним із ключових етапів проєктування, оскільки саме на цьому рівні визначається відповідність майбутнього програмного продукту реальним потребам підприємства. У даному контексті вимоги розглядаються як відображення організаційної структури, специфіки управління та особливостей інформаційних процесів підприємства.

Аналіз діяльності магазинів автозапчастин свідчить про наявність суттєвого розриву між накопиченням операційних даних та їх практичним використанням в

управлінні. Інформація про продажі, закупівлі та складські залишки, як правило, фіксується в облікових системах, однак не об'єднується в єдину інформаційну модель. Це знижує її аналітичну цінність і ускладнює використання для прийняття управлінських рішень. У зв'язку з цим базовою вимогою до системи є забезпечення централізованого зберігання даних та підтримка їх узгодженості.

Основні вимоги до система магазину автозапчастин, по типу малого торговельного підприємства наведено в таблиці 2.2.

Таблиця 2.2 – Основні групи вимог до інформаційної системи

Група вимог	Зміст вимог	Обґрунтування
Функціональні вимоги	Облік продажів, закупівель, клієнтів, товарів та складських залишків	Операційні процеси є джерелом первинних даних для подальшого аналізу
Аналітичні вимоги	Аналіз динаміки продажів, оборотності товарів, стану залишків	Необхідність підтримки управлінських рішень на основі оброблених даних
Інформаційні вимоги	Централізоване зберігання та узгодженість даних	Усунення фрагментації інформації та підвищення її достовірності
Вимоги до інтерфейсу	Простота, інтуїтивність, зручність використання	Користувачами є працівники без спеціальної ІТ-підготовки
Вимоги до безпеки	Розмежування прав доступу користувачів	Необхідність захисту управлінської та фінансової інформації
Технологічні вимоги	Веб-орієнтована архітектура	Забезпечення доступності та зниження вимог до обладнання
Економічні вимоги	Мінімізація витрат на впровадження та підтримку	Обмежені фінансові ресурси малого бізнесу
Обмеження системи	Відсутність складних прогнозних алгоритмів	Орієнтація на навчальний характер та доступні дані

Окремо слід відзначити, що інформаційна система повинна орієнтуватися не лише на зберігання даних, але й на забезпечення їх коректної обробки та використання в межах операційної діяльності підприємства. На відміну від класичних облікових систем, які переважно фіксують господарські операції,

сучасні рішення повинні забезпечувати можливість аналізу основних показників діяльності, виявлення закономірностей у роботі магазину та формування узагальненої інформації для внутрішнього використання.

Для наочного відображення функціональних вимог із позиції кінцевих користувачів системи використовують діаграму варіантів використання. Вона показує, які дії може виконувати кожен тип користувача і як ці дії пов'язані між собою. Діаграма варіантів використання системи наведена на рисунку 2.1.

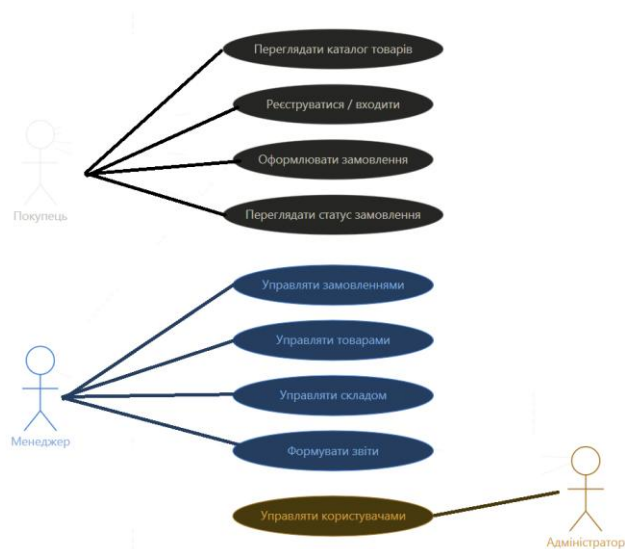


Рисунок 2.1 – Use Case Diagram

Важливою вимогою є також забезпечення зручності використання системи. Надмірна складність функціоналу або перевантажений інтерфейс можуть негативно впливати на ефективність роботи користувачів, особливо за відсутності спеціальної підготовки. Тому інтерфейс повинен бути логічним, зрозумілим та мінімізувати кількість помилок при введенні даних.

Практика впровадження інформаційних систем у малому бізнесі показує, що часто використовуються рішення, адаптовані з систем для середніх або великих підприємств. Такий підхід не завжди є ефективним, оскільки не враховує обмеженість ресурсів та специфіку роботи невеликих торговельних точок. У результаті такі системи можуть використовуватися не повною мірою або вимагати додаткових витрат на налаштування та супровід [13].

Особливого значення набувають можливості обробки та представлення даних. Система повинна забезпечувати не лише зберігання інформації, але й її структурування, узагальнення та перегляд у зручному форматі. Для магазину автозапчастин це включає перегляд продажів за періодами, контроль асортименту, аналіз складських залишків та виявлення товарів із низькою або надмірною оборотністю.

Окрему увагу слід приділити якості та повноті даних. Некоректна або неповна інформація може призводити до помилок у веденні обліку та негативно впливати на загальну ефективність роботи магазину. У зв'язку з цим система повинна забезпечувати контроль цілісності даних та мінімізацію впливу людського фактора.

Процес оформлення замовлення в інформаційній системі магазину автозапчастин є одним із ключових бізнес-процесів, оскільки безпосередньо пов'язаний із взаємодією користувача із системою та фіксацією результатів продажу. Формалізований опис цього процесу у вигляді UML-діаграми діяльності наведено на рисунку 2.2.



Рисунок 2.2 – Діаграма діяльності процесу оформлення замовлення в системі

Крім того, необхідним є розмежування прав доступу користувачів. Навіть у межах малого підприємства існує потреба у диференціації доступу до окремих функцій та даних, зокрема адміністративних розділів. Це підвищує рівень інформаційної безпеки та впорядковує процеси роботи персоналу.

Сучасні тенденції розвитку інформаційних систем передбачають використання веб-орієнтованих рішень, що забезпечують доступність і гнучкість використання. Водночас надмірна залежність від зовнішньої інфраструктури може бути небажаною для малого бізнесу, тому важливим є баланс між сучасними технологічними підходами та збереженням контролю над ключовими даними.

Таким чином, вимоги до інформаційної системи для магазину автозапчастин формуються на основі поєднання операційних потреб, організаційних обмежень та особливостей інформаційних потоків підприємства. Система повинна забезпечувати зручну роботу з даними, їх структурованість і доступність, залишаючись при цьому простою у використанні та економічно доцільною.

2.3 Вибір та обґрунтування серверних технологій

У сучасних веб-орієнтованих інформаційних системах серверна частина виконує значно ширшу роль, ніж просто обробка запитів клієнта або взаємодія з базою даних. Вона фактично є основним логічним ядром системи, де відбувається обробка, перетворення та структуризація первинної інформації у вигляді, придатному для подальшого використання в межах бізнес-процесів торговельного підприємства. Для систем магазину автозапчастин, де характерними є великі обсяги номенклатури та часті зміни складських даних, продуктивність і надійність серверного рівня мають визначальний вплив на стабільність роботи всієї системи.

Одним із найбільш поширених інструментів для реалізації серверної логіки є мова програмування C#. Вона належить до сучасних об'єктно-орієнтованих мов із

суворою системою типізації, що дозволяє виявляти значну частину помилок ще на етапі компіляції. Це особливо важливо для систем, які обробляють комерційні дані, оскільки зменшується ризик виникнення критичних помилок під час виконання. Крім того, об'єктний підхід у C# дозволяє формалізувати предметну область у вигляді класів і сервісів, що спрощує структуру програмного коду та його подальший супровід [14].

Важливою складовою екосистеми C# є технологія LINQ (Language Integrated Query), яка дозволяє виконувати операції вибірки та обробки даних безпосередньо в межах мови програмування (рис. 2.3). Це забезпечує уніфікований підхід до роботи з даними та зменшує необхідність написання великої кількості окремих SQL-запитів. У контексті торговельної системи LINQ особливо корисний для реалізації операцій аналізу продажів, фільтрації товарів, розрахунку оборотності та узагальнення інформації за певними періодами.

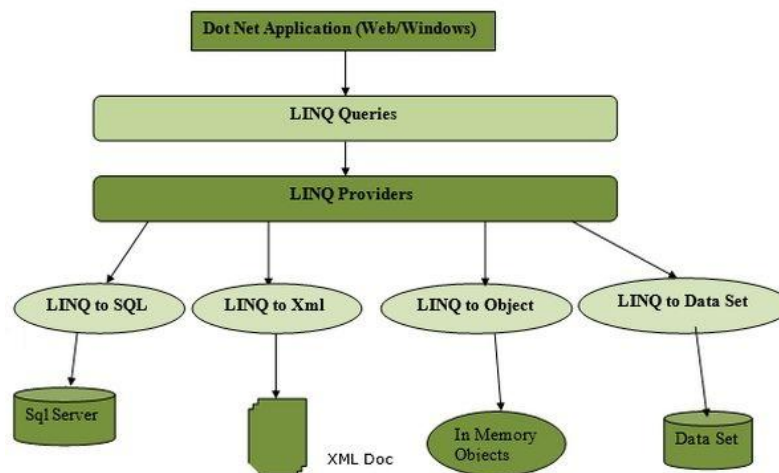


Рисунок 2.3 – Архітектура LINQ

При цьому слід враховувати, що використання керованого середовища виконання .NET має певні особливості, зокрема пов'язані з роботою автоматичного керування пам'яттю. У високонавантажених сценаріях це може призводити до непередбачуваних затримок, що потребує додаткової уваги під час проєктування продуктивних ділянок системи [15].

Серверна частина застосунків на C# базується на платформі .NET, яка надає

набір бібліотек та інструментів для створення багаторівневих інформаційних систем. Сучасні версії платформи орієнтовані на високу продуктивність, підтримку кросплатформенності та ефективне використання ресурсів. Важливою перевагою є модульність архітектури, що дозволяє розділяти систему на окремі логічні компоненти.

Окремо слід відзначити підтримку принципів інверсії керування та впровадження залежностей (Dependency Injection), що дозволяє будувати слабко зв'язані компоненти системи, як показано на рисунку 2.4. Такий підхід спрощує масштабування та модифікацію окремих частин системи без необхідності суттєвого перепроєктування всієї архітектури [16].

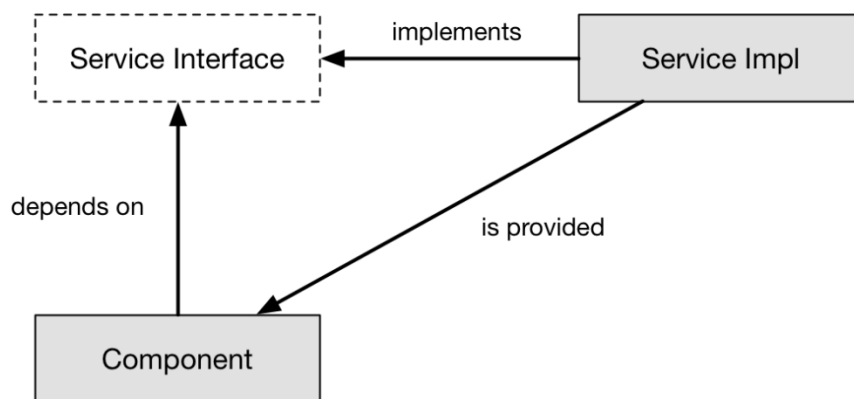


Рисунок 2.4 – Схема роботи Dependency Injection

Для реалізації взаємодії між клієнтською та серверною частинами використовується фреймворк ASP.NET Core, який дозволяє будувати веб-API на основі REST-архітектури. Обмін даними у таких системах зазвичай здійснюється через HTTP-запити з використанням формату JSON. Це забезпечує просту інтеграцію з різними типами клієнтів, включаючи веб-інтерфейси та мобільні застосунки [17].

У подібних системах значну увагу необхідно приділяти питанням безпеки, зокрема автентифікації користувачів та розмежуванню доступу до функціональних можливостей. Це вимагає впровадження відповідних механізмів контролю доступу

та захисту переданих даних.

Для роботи з базою даних у середовищі .NET часто використовується технологія Entity Framework Core, яка реалізує підхід об'єктно-реляційного відображення (рис. 2.5). Вона дозволяє працювати з даними через об'єктну модель, що спрощує розробку та зменшує кількість ручного SQL-коду [18].

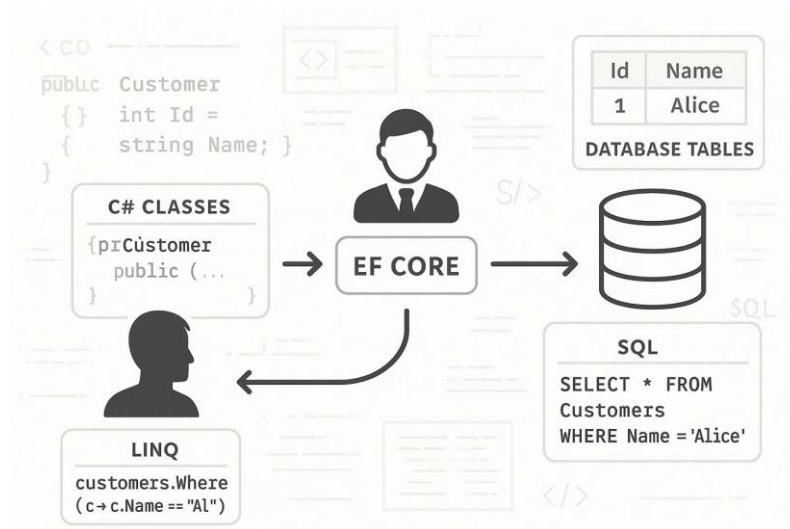


Рисунок 2.5 – Створення SQL-запитів за допомогою EF Core

Водночас застосування ORM має і певні недоліки. У складних сценаріях можлива генерація неефективних запитів або надмірна кількість звернень до бази даних при роботі з пов'язаними сутностями. Це потребує додаткової оптимізації, зокрема використання механізмів асинхронної обробки та контрольованого завантаження даних.

Таким чином, використання C#, платформи .NET, ASP.NET Core та Entity Framework Core забезпечує створення сучасної серверної частини інформаційної системи, здатної ефективно обробляти дані та підтримувати складну бізнес-логіку. Разом із тим, окремі технологічні особливості вимагають зваженого підходу до проєктування архітектури та оптимізації продуктивності.

2.4 Вибір та обґрунтування технологій клієнтської частини

Ефективність роботи сучасних веборієнтованих інформаційних систем у сфері електронної комерції значною мірою залежить від того, як організовано клієнтський рівень та які технології використано для його реалізації. Саме клієнтська частина забезпечує взаємодію користувача із системою, впливаючи на зручність навігації, швидкість отримання необхідної інформації та загальну якість відображення результатів обробки даних. У системах підтримки управлінських рішень цей рівень виконує не лише роль інтерфейсу, а й слугує інструментом подання та інтерпретації аналітичних показників, що формуються на серверній стороні.

У сучасних інформаційних системах, які працюють із великими масивами даних, передбачають складну фільтрацію та інтерактивне подання інформації, дедалі частіше застосовується підхід односторінкових застосунків (SPA). Його особливість полягає в тому, що користувацька оболонка завантажується один раз, після чого всі подальші зміни інтерфейсу відбуваються через асинхронний обмін даними із сервером без повного перезавантаження сторінки. Такий принцип роботи дозволяє скоротити час відгуку системи, підвищити плавність взаємодії та забезпечити більш комфортний користувацький досвід [19].

Однією з найпоширеніших технологій для реалізації SPA є бібліотека React. В її основі лежить компонентний підхід до побудови інтерфейсу, згідно з яким інтерфейс складається з окремих незалежних частин (рис. 2.6). Кожен компонент поєднує в собі структуру, логіку та власний стан, що дозволяє розглядати його як самостійну одиницю. Завдяки такій моделі складні інтерфейси формуються шляхом комбінування простіших елементів, що суттєво полегшує їх супровід і масштабування. Цей підхід є особливо ефективним для систем, де інтерфейс складається з повторюваних елементів, таких як списки товарів, панелі фільтрації, таблиці даних або аналітичні віджети.

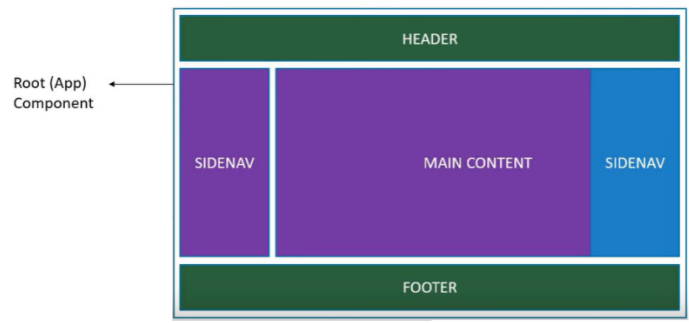


Рисунок 2.6 – Компонентно-орієнтована модель побудови інтерфейсу

У межах React використовується механізм віртуального DOM, який виступає проміжним шаром між логікою застосунку та реальним DOM браузера. Зміни стану спочатку фіксуються у віртуальній моделі, після чого система визначає мінімально необхідні операції для оновлення реального DOM. Це дає змогу зменшити навантаження на браузер і підвищити ефективність оновлення інтерфейсу при роботі з великими обсягами даних [20].

Декларативний підхід, який застосовується у React, передбачає опис інтерфейсу як функції від поточного стану даних. Така модель спрощує логіку оновлення елементів сторінки та зменшує ризик виникнення помилок, пов'язаних із некоректною синхронізацією станів інтерфейсу. Водночас React не нав'язує жорсткої архітектури застосунку, що забезпечує гнучкість розробки, але потребує використання додаткових інструментів для маршрутизації, керування станом і взаємодії з API.

Для підвищення надійності клієнтської частини часто застосовується TypeScript — надбудова над JavaScript із підтримкою статичної типізації. Завдяки статичному аналізу коду більшість помилок можна виявити ще на етапі розробки, до запуску програми [21]. У контексті вебсистем це підвищує стабільність роботи клієнтського рівня та спрощує інтеграцію із серверними інтерфейсами. Чітко визначені типи даних і контрактів обміну сприяють узгодженості між компонентами системи та зменшують ризик некоректної обробки аналітичної інформації.

Окремою проблемою SPA-підходу є організація та управління глобальним станом застосунку. У великих системах різні частини інтерфейсу часто повинні

використовувати спільні дані, наприклад параметри користувача або результати запитів. Для цього застосовують централізовані сховища стану. Одним із найвідоміших рішень є Redux Toolkit, який базується на концепції єдиного джерела істини. Зміни стану відбуваються через чітко визначені дії, що забезпечує прогнозованість роботи застосунку та спрощує його налагодження [22].

Сучасним розвитком цього підходу є RTK Query, який поєднує керування станом із механізмами взаємодії із сервером. Він дозволяє описувати запити декларативно та автоматично забезпечує кешування, повторне використання даних і синхронізацію актуальної інформації без додаткової ручної реалізації цих процесів.

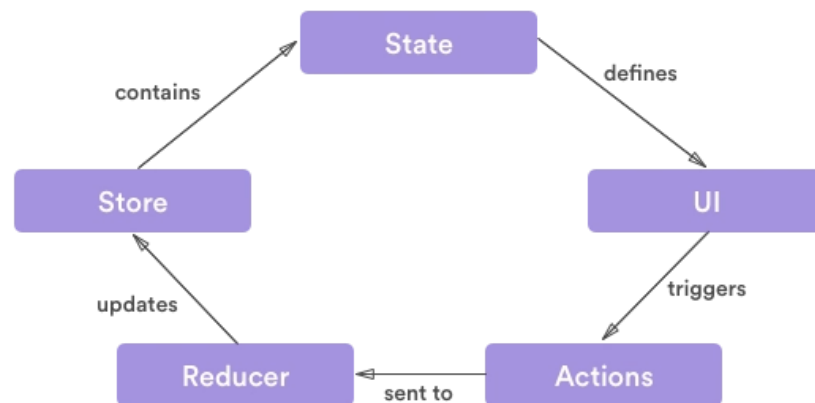


Рисунок 2.7 – Принцип роботи Redux Toolkit

Важливу роль у сприйнятті вебзастосунків відіграє також візуальна складова інтерфейсу. Для React-екосистеми широко використовуються CSS-фреймворки та UI-бібліотеки:

- Tailwind CSS;
- Chakra UI;
- Ant Design;
- Bootstrap.

Вони дають змогу швидко створювати адаптивні та візуально узгоджені інтерфейси, які коректно відображаються на різних пристроях.

Клієнтський рівень також виконує функції базового захисту взаємодії

користувача із системою. У безстанових архітектурах автентифікація найчастіше реалізується через JSON Web Token, який передається разом із кожним запитом до серверного API. Такий підхід спрощує масштабування системи, однак вимагає ретельного контролю за збереженням і використанням токенів, а також механізмів їх оновлення.

Окрему категорію інструментів становлять засоби візуалізації даних. Бібліотеки на кшталт Chart.js дозволяють перетворювати числові показники у графічні представлення, що значно полегшує аналіз динаміки показників, зокрема у сфері продажів або обліку товарів. Використання графіків і діаграм підвищує наочність інформації та сприяє швидшому прийняттю управлінських рішень.

Таким чином, сучасні технології клієнтського рівня SPA-застосунків формують багатокомпонентну екосистему, яка поєднує компонентну архітектуру, ефективне управління станом, типізацію даних і засоби аналітичної візуалізації. Їх використання створює основу для розробки продуктивних інформаційних систем, орієнтованих на високу інтерактивність, масштабованість і якісну обробку аналітичних даних.

2.5 Проєктування бази даних

Проєктування бази даних є одним із найважливіших етапів створення інформаційної системи, саме на цьому рівні визначаються принципи організації, зберігання та обробки інформації. Від правильності побудови структури бази даних залежить цілісність, достовірність і узгодженість даних, які надалі використовуються для виконання операцій та аналітичної обробки.

Основою більшості сучасних інформаційних систем виступають реляційні бази даних. Саме вони призначені для роботи зі структурованою інформацією. Їх особливість полягає у представленні даних у вигляді взаємопов'язаних таблиць, що дозволяє формалізувати предметну область і впорядкувати інформацію відповідно

до логіки бізнес-процесів. Використання первинних і зовнішніх ключів, а також механізмів забезпечення цілісності дає можливість підтримувати узгодженість даних під час виконання операцій та формування аналітичних звітів.

Важливою перевагою реляційної моделі є підтримка транзакцій відповідно до ACID-властивостей. Завдяки цьому забезпечується коректне виконання операцій навіть у багатокористувацькому середовищі. Для інформаційної системи магазину автозапчастин це особливо важливо, оскільки система повинна надійно фіксувати продажі, закупівлі та зміни складських залишків навіть при одночасній роботі декількох користувачів. Крім того, нормалізація структури бази даних дозволяє зменшити дублювання інформації та підвищити ефективність виконання складних аналітичних запитів [23].

Для реалізації зберігання даних, у моїй роботі було обрано SQLite – легку реляційну систему управління базами даних, яка не потребує окремого серверного середовища. Усі дані зберігаються в одному файлі, що значно спрощує розгортання та адміністрування системи. Незважаючи на компактність, SQLite підтримує основні механізми реляційних СУБД, зокрема транзакції, індекси та обмеження цілісності. Це робить її придатною для використання у невеликих веборієнтованих інформаційних системах, де важливими є простота використання та невисокі вимоги до ресурсів.

SQLite має і певні обмеження. Вони стосуються масштабованості та роботи з великою кількістю одночасних операцій запису, що пояснюється файловою організацією зберігання даних. Проте для систем малого бізнесу такі недоліки не є критичними [24].

Проектування структури бази даних починається з аналізу предметної області [25]. У межах даної роботи було розглянуто процеси продажу автозапчастин, обліку товарних залишків, взаємодії з постачальниками та формування управлінських рішень.

На основі цього аналізу створюється концептуальна модель, у якій визначаються основні сутності та зв'язки між ними. Такий підхід дозволяє забезпечити відповідність структури бази даних реальним бізнес-процесам і

створює основу для подальшої реалізації системи.

Центральним елементом бази даних є сутність «Товар», яка описує окремі позиції автозапчастин. Вона містить атрибути, необхідні для ідентифікації продукції, обліку вартості та контролю складських залишків. Під час проектування цієї сутності враховано необхідність зберігання як описових характеристик товару, так і кількісних параметрів, які використовуються для аналітики попиту та обліку запасів.

Для впорядкування асортименту передбачено сутність «Категорія», що реалізує логічний поділ товарів за групами та спрощує навігацію й фільтрацію у клієнтському інтерфейсі.

Окремий блок сутностей відповідає за організацію закупівель і взаємодію з постачальниками. Сутність «Постачальник» зберігає інформацію про контрагентів, які забезпечують постачання товарів. Сутність «Закупівля» використовується для фіксації фактів придбання товарів у певного постачальника із зазначенням дати та загальної вартості. Детальна інформація про склад кожної закупівлі реалізується через таблицю позицій закупівель, у якій зберігаються дані про кількість та собівартість окремих товарів. Така структура дозволяє аналізувати ефективність співпраці з постачальниками та динаміку закупівель.

Для моделювання процесів продажу використовуються сутності «Клієнт», «Замовлення» та «Позиції замовлень». Таблиця клієнтів містить контактну інформацію покупців, а сутність «Замовлення» фіксує сам факт продажу, дату оформлення, суму та поточний статус виконання. Зв'язок між товарами та замовленнями реалізовано через проміжну таблицю позицій замовлень, яка дозволяє зберігати інформацію про кількість і ціну кожної придбаної запчастини. Подібна структура характерна для транзакційних систем і створює основу для подальшого аналізу продажів та поведінки клієнтів.

Важливе місце у структурі бази даних займає сутність «Рух складу», призначена для обліку змін кількості товарів. У ній фіксуються всі операції, пов'язані з надходженням, продажем або коригуванням запасів. Такий підхід дозволяє зберігати історію змін складських залишків та формувати аналітичні звіти

щодо дефіциту товарів або перевищення оптимального рівня запасів. Дані про рух товарів є важливими для формування обґрунтованих управлінських рішень.

Для реалізації механізмів контролю доступу передбачено сутності «Користувач» і «Роль». Це дозволяє розподілити права доступу між адміністраторами, менеджерами та клієнтами системи. Використання ролей забезпечує більш безпечну та впорядковану організацію роботи користувачів у системі.

Особливістю структури бази даних є наявність таблиці журналу управлінських рішень. Вона використовується для збереження результатів аналітичної обробки даних і містить інформацію про тип рішення, опис ситуації, сформовані рекомендації, а також дату й час створення запису.

Зв'язок із таблицею користувачів реалізовано через зовнішній ключ UserID, що дозволяє визначити автора сформованого рішення.

Під час проектування зв'язків між сутностями враховано принципи нормалізації баз даних. Це дозволяє мінімізувати дублювання інформації та підтримувати її цілісність. Використання зв'язків типу один-до-багатьох і багато-до-одного відображає реальні взаємозв'язки між об'єктами предметної області та полегшує виконання складних аналітичних запитів. При цьому структура бази даних залишається достатньо гнучкою для подальшого розширення відповідно до розвитку бізнес-процесів підприємства [26].

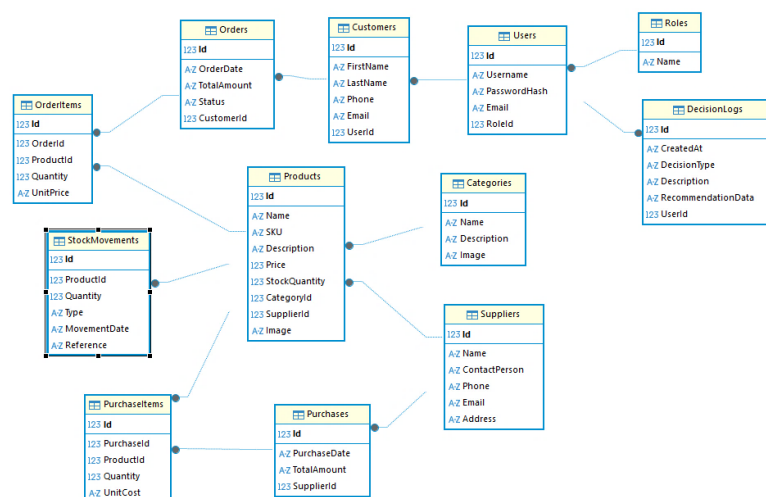


Рисунок 2.8 – Схема таблиць у базі даних

У результаті проєктування була сформована модель бази даних, яка комплексно відображає діяльність магазину автозапчастин та поєднує транзакційні, довідкові й аналітичні аспекти функціонування системи. Розроблена структура створює надійну основу для реалізації модуля підтримки прийняття управлінських рішень і забезпечує можливість накопичення, структуризації та подальшого аналізу інформації, необхідної для ефективного управління закупівлями, продажами та складськими запасами.

2.6 Висновки до розділу 2

У другому розділі було проведено проєктування комп'ютерної системи магазину автозапчастин та розглянуто основні аспекти її побудови. У процесі виконання роботи було проаналізовано предметну область, визначено особливості функціонування магазину автозапчастин і сформовано основні вимоги до майбутньої інформаційної системи. На основі цього було визначено перелік функціональних можливостей системи, а також технічні вимоги, необхідні для забезпечення стабільної та ефективної роботи програмного засобу.

У межах розділу здійснено вибір серверних технологій, які забезпечують обробку запитів, взаємодію з базою даних та реалізацію бізнес-логіки системи. Також було обґрунтовано використання сучасних технологій клієнтської частини, орієнтованих на створення інтерактивного вебінтерфейсу та зручної взаємодії користувача із системою.

Окрему увагу приділено проєктуванню бази даних, яка є основою функціонування інформаційної системи. Було визначено основні сутності предметної області, їх атрибути та взаємозв'язки. Розроблена структура бази даних дозволяє забезпечити зберігання інформації про товари, замовлення, постачальників, клієнтів і складські операції, а також створює основу для подальшого аналітичного опрацювання даних та підтримки прийняття

управлінських рішень.

У результаті виконаного проектування сформовано цілісну модель комп'ютерної системи магазину автозапчастин, яка відповідає поставленим вимогам та може бути використана як основа для подальшої програмної реалізації й впровадження системи у практичну діяльність підприємства.

3 РОЗРОБКА ТА ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Реалізація серверної частини системи

Реалізація серверної частини магазину автотоварів була спрямована на створення стабільного програмного середовища. Воно має забезпечувати обробку запитів користувачів, взаємодію з базою даних та виконання бізнес-логіки системи. Основною задачею під час розробки було побудувати багаторівневу архітектуру, у якій окремі компоненти виконують власні функції та можуть незалежно змінюватися.

Під час реалізації серверної частини використовувалися принципи Clean Architecture та SOLID [27]. Бізнес-логіку було відокремлено від механізмів роботи з базою даних і засобів відображення інформації. Це дало можливість створити більш гнучку структуру застосунку, де зміни в одному рівні не впливають на інші компоненти системи (рис. 3.1).

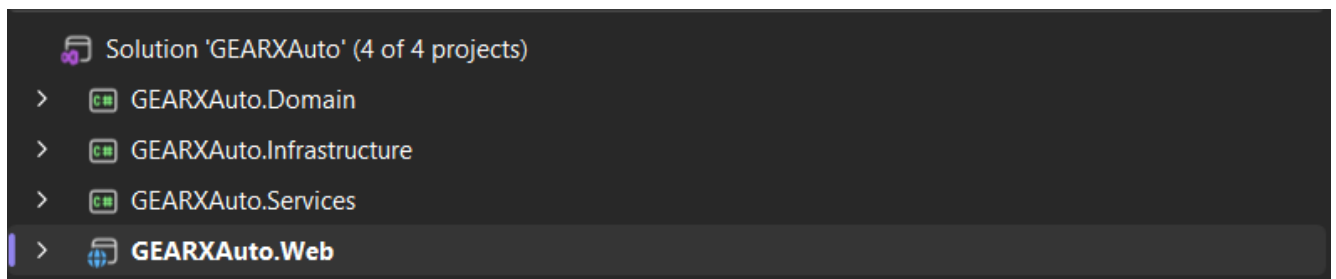


Рисунок 3.1 – Структура бек-енд частини

Серверна частина системи була реалізована на платформі ASP.NET Core із використанням підходу Web API. Така архітектура дозволила організувати взаємодію між клієнтською та серверною частинами через REST-запити у форматі JSON. У структурі застосунку серверні ендпоінти були розподілені між окремими контролерами відповідно до функціонального призначення.

Для роботи з товарами реалізовано ProductsController. Він забезпечує отримання списку товарів, пошук, фільтрацію та перегляд детальної інформації про

конкретну позицію. Під час отримання списку товарів використовується пагінація, що дозволяє обмежити кількість записів, які одночасно передаються клієнту. Також реалізовано фільтрацію за категоріями та пошук за назвою або артикулом товару. Для оптимізації роботи запити формуються через IQueryable, що дозволяє динамічно змінювати умови вибірки ще до виконання SQL-запиту.

Для отримання інформації про категорії товарів створено CategoriesController. У ньому реалізовано механізми пошуку та посторінкового виведення результатів. Це спрощує роботу з великими наборами даних і зменшує навантаження на серверну частину.

Окрему роль у системі відіграє OrdersController, який відповідає за оформлення замовлень та отримання інформації про історію покупок користувача. Під час створення нового замовлення серверна частина перевіряє наявність товарів на складі, актуальність кількості та правильність переданих даних. Якщо товар відсутній або його недостатньо, то система повертає повідомлення про помилку. У разі успішного проходження перевірок формується запис у таблиці Orders та відповідні записи у таблиці OrderItems.

Операція оформлення замовлення реалізована із використанням транзакцій бази даних через BeginTransactionAsync. Якщо під час створення замовлення виникає помилка, усі зміни автоматично скасовуються, що виключає ситуації часткового запису інформації. Потім система автоматично зменшує значення StockQuantity для відповідних товарів.

Для доступу до бази даних використано Entity Framework Core. Робота з даними реалізована через контекст ApplicationDbContext.

ApplicationDbContext містить набір DbSet для всіх сутностей системи: Users, Products, Categories, Orders, OrderItems, Suppliers, Purchases, PurchaseItems, StockMovements. Контекст виступає проміжним рівнем між серверною логікою та фізичною базою даних SQLite.

Під час реалізації використано підхід Code First, за якого структура таблиць формується автоматично на основі C#-моделей. У методі OnModelCreating були налаштовані зв'язки між сутностями, зовнішні ключі та правила каскадного

видалення. Наприклад, для сутності Customer реалізовано зв'язок один-до-одного з таблицею Users, а для Product – зв'язок багато-до-одного з таблицею Categories. Це дозволило забезпечити цілісність даних і коректне відображення зв'язків між об'єктами предметної області.

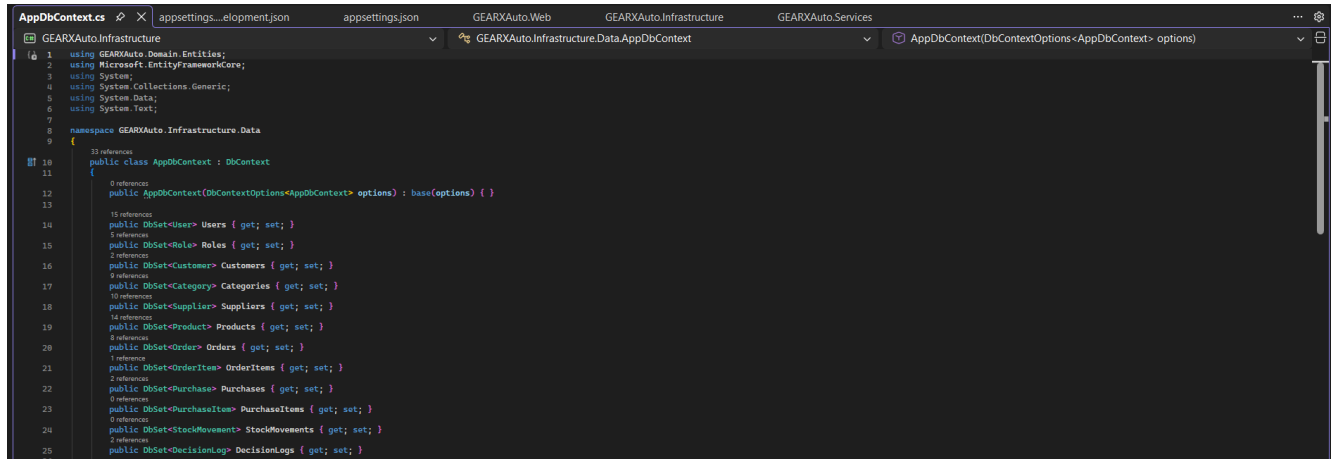


Рисунок 3.2 – Контекст доступу до даних AppDbContext

Важливу роль у роботі серверної частини відіграє система автентифікації та авторизації користувачів. Для адміністративної частини застосунку було використано Cookie Authentication. Для роботи клієнтського API обрано JWT-токени. Такий підхід дозволяє розмежувати доступ до службових та публічних ресурсів системи.

Паролі користувачів не зберігаються у відкритому вигляді. Для їх захисту використовується стандартний механізм IpasswordHasher. Він виконує хешування перед збереженням у базі даних. Це дозволяє підвищити безпеку системи навіть у випадку несанкціонованого доступу до файлу SQLite.

Для підтримки взаємодії між серверною та клієнтською частинами також було налаштовано CORS-політику. Це дозволило надати доступ React-застосунку, який працює на окремому локальному порту, до серверного API.

У процесі реалізації було підключено AutoMapper для автоматичного перетворення моделей бази даних у DTO та ViewModel.

Для тестування та документування API було використано Swagger. Завдяки цьому стало можливим переглядати список доступних ендпоінтів, перевіряти

параметри запитів та тестувати роботу серверної частини без використання сторонніх програм.

У результаті було створено серверну частину системи, яка забезпечує обробку бізнес-логіки, взаємодію з клієнтським застосунком та підтримку основних функцій магазину автозапчастин.

3.2 Реалізація клієнтської частини системи

Клієнтська частина системи була реалізована у вигляді односторінкового вебзастосунку з використанням бібліотеки React. Такий підхід дозволив створити динамічний інтерфейс, у якому оновлення даних відбувається без повного перезавантаження сторінки. Основою структури інтерфейсу стали функціональні компоненти та React-хуки `useState` і `useEffect`, які забезпечують керування станом і реактивне оновлення елементів сторінки [28].

У межах клієнтської частини було створено окремі сторінки для каталогу товарів, профілю користувача та адміністративної панелі.

Структура застосунку побудована таким чином, щоб користувач міг швидко переходити між розділами та отримувати доступ до необхідної інформації (рис. 3.3).

```

16  function App() {
17    return (
18      <BrowserRouter>
19        <Routes>
20          <Route path="/" element={<MainLayout />} />
21          <Route index element={<HomePage />} />
22          <Route path="login" element={<LoginPage />} />
23          <Route path="orders" element={<OrdersPage />} />
24          <Route path="categories" element={<CategoriesPage />} />
25          <Route path="category/:id" element={<CategoryPage />} />
26          <Route path="cart" element={<CartPage />} />
27          <Route path="confirm" element={<OrderSuccessPage />} />
28          <Route path="products" element={<ProductsPage />} />
29          <Route path="product/:id" element={<ProductDetailsPage />} />
30          <Route path="order/:id" element={<OrderDetailsPage />} />
31          <Route path="/register" element={<RegisterPage />} />
32        </Routes>
33      </BrowserRouter>
34    );
35  }
36

```

Рисунок 3.3 – Структура сторінок клієнтського сайту

Для централізованого управління глобальним станом застосунку використовувався Redux Toolkit. За його допомогою було реалізовано зберігання даних кошика, параметрів авторизації та інших спільних станів системи. Використання слайсів дозволило зосередити логіку оновлення станів у єдиному місці та спростити взаємодію між компонентами інтерфейсу. Після додавання товару до кошика зміни відразу відображаються у відповідному лічильнику без повторного звернення до сервера.

Обмін даними між клієнтською та серверною частинами реалізовано через HTTP-запити у форматі JSON.

Для роботи із серверними API використовувався RTK Query, який забезпечує автоматичне кешування та повторне використання результатів запитів. Це дозволило зменшити кількість звернень до сервера та підвищити швидкодію інтерфейсу.

Одним із найскладніших процесів стала реалізація механізму оформлення замовлення. На клієнтському рівні формується список товарів та їх кількостей, після чого інформація передається на сервер. Серверна частина перевіряє актуальність цін, доступність товарів на складі та створює записи в таблицях Orders і OrderItems. Усі операції виконуються в межах однієї транзакції бази даних, що дозволяє уникнути помилок часткового запису даних.

Адміністративна панель була створена з урахуванням основних потреб менеджера магазину. Для її реалізації використовувалися Razor Views у поєднанні з JavaScript-скриптами. Панель керування надає можливість редагувати товари, категорії, постачальників та керувати закупівлями.

Оформлення закупівель реалізовано у вигляді динамічних форм, де користувач може додавати необхідну кількість позицій. Після підтвердження закупівлі система автоматично оновлює значення StockQuantity у таблиці Products, завдяки чому залишки на складі змінюються в реальному часі.

Для підвищення наочності аналітичної інформації у системі реалізовано візуалізацію статистичних даних у вигляді графіків і діаграм. Аналітичні показники

передаються через окремі API-ендпоінти та відображаються на клієнтському рівні, що дозволяє користувачам швидко оцінювати динаміку продажів і популярність товарів.

Було реалізовано повноцінну клієнт-серверну інформаційну систему, яка об'єднує React-інтерфейс, серверну платформу .NET та ORM-рівень доступу до даних у межах єдиної архітектури. Реалізовані механізми дозволяють забезпечити стабільну роботу системи, підтримку аналітичних функцій та подальше розширення програмного продукту.

3.3 Інструкція адміністратора системи

Робота з розробленою інформаційною системою магазину автозапчастин організована шляхом поділу функціоналу між адміністративною частиною та клієнтським вебінтерфейсом. Для персоналу магазину реалізована окрема панель керування на базі ASP.NET Core MVC, а для покупців створено клієнтський сайт у форматі SPA-застосунку на основі React. Такий підхід дозволяє ефективно організувати внутрішні бізнес-процеси та забезпечити швидку взаємодію користувачів із системою.

Робота адміністратора або менеджера починається з проходження авторизації через спеціальну форму входу до системи (рис. 3.4).

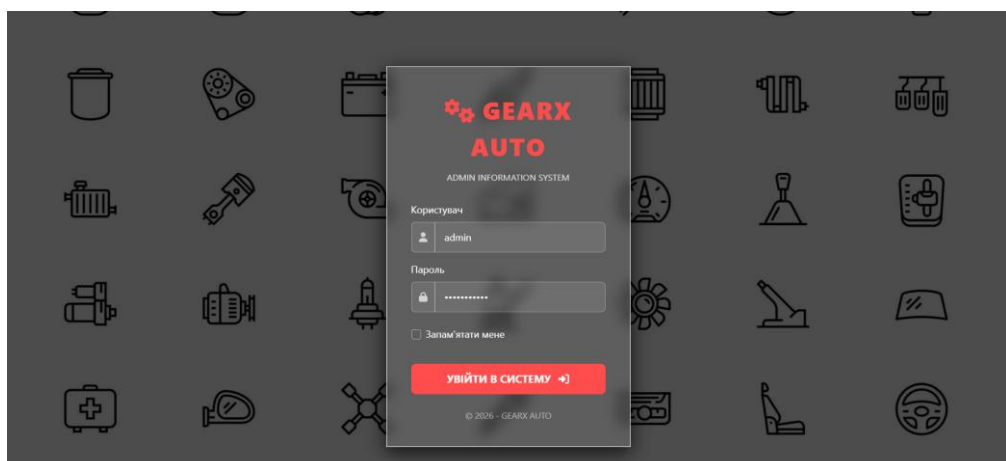


Рисунок 3.4 – Сторінка авторизації у адмін-панелі

Після успішного входу користувач переходить до головної панелі керування, яка виконує роль центрального аналітичного модуля системи (рис. 3.5).

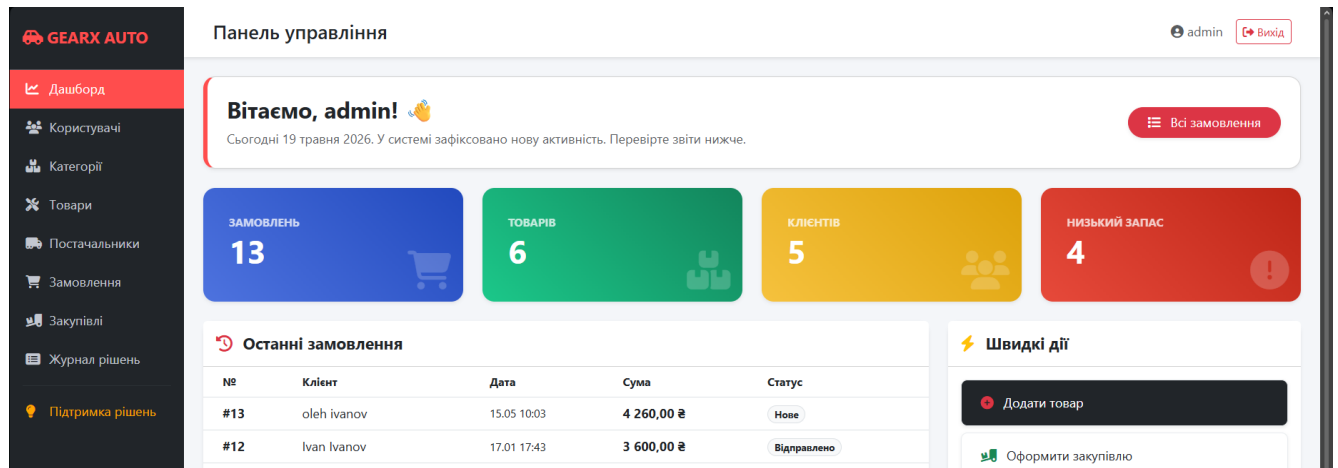


Рисунок 3.5 – Аналітичний дашборд адмін-панелі

На головній сторінці адміністративної панелі відображаються основні статистичні показники та аналітична інформація, необхідна для контролю роботи магазину. Окреме місце займає блок критичних залишків, у якому автоматично виводяться товари з недостатньою кількістю на складі. Це дозволяє менеджерам швидко реагувати на дефіцит продукції без необхідності постійного ручного контролю складських запасів.

Адміністративна панель також забезпечує можливість створення та редагування категорій товарів і карток автозапчастин.

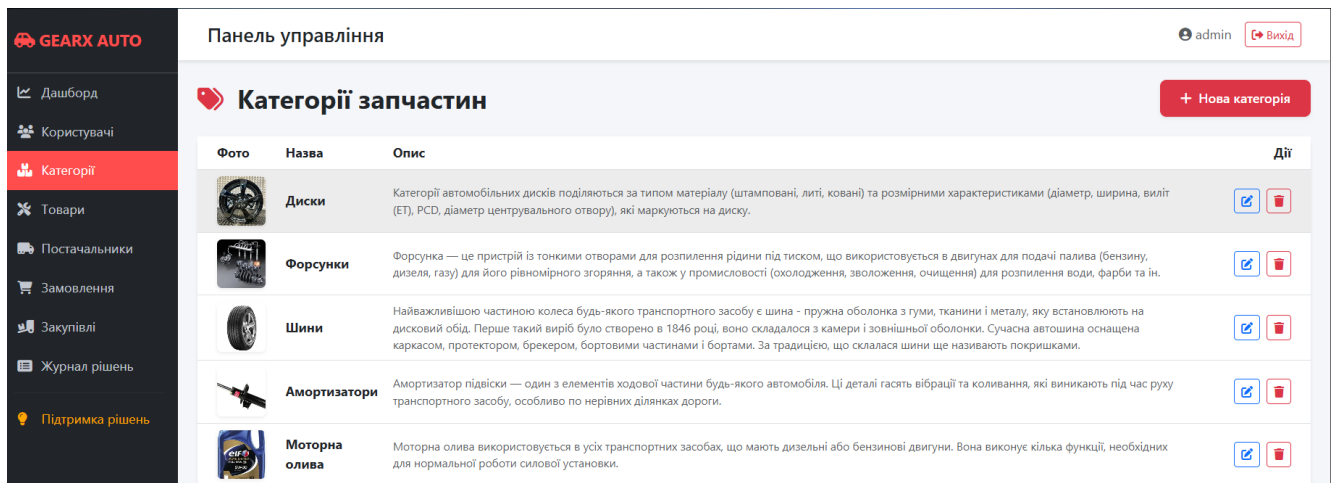


Рисунок 3.6 – Сторінка відображення та управління категоріями

Реалізований функціонал дозволяє переглядати, додавати та змінювати інформацію про товари й категорії через зручний інтерфейс. Завдяки простій структурі панелі керування нові працівники можуть швидко освоїти принцип роботи системи без додаткового навчання.

Для покращення вигляду клієнтського сайту в системі передбачено підтримку завантаження та зміни зображень товарів і категорій. Додавання або редагування фотографій здійснюється безпосередньо під час створення чи зміни відповідної сутності (рис. 3.7).

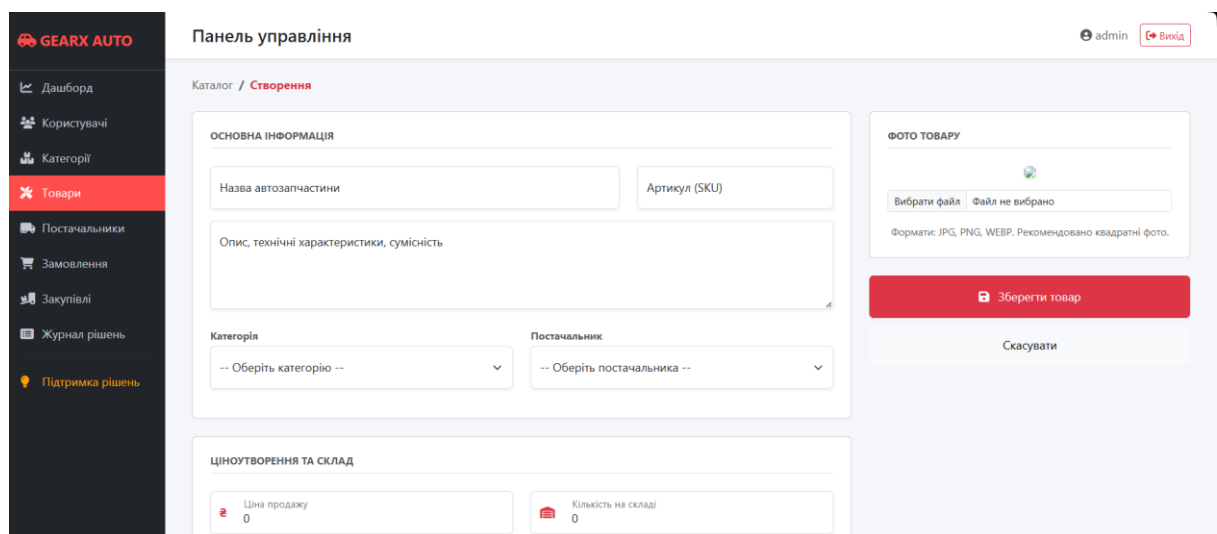


Рисунок 3.7 – Редагування обраного товару

Окрім товарного каталогу, у системі реалізовано окремий модуль роботи з

постачальниками. Він дозволяє зберігати та редагувати контактну інформацію контрагентів для подальшої організації закупівель і логістики (рис. 3.8).

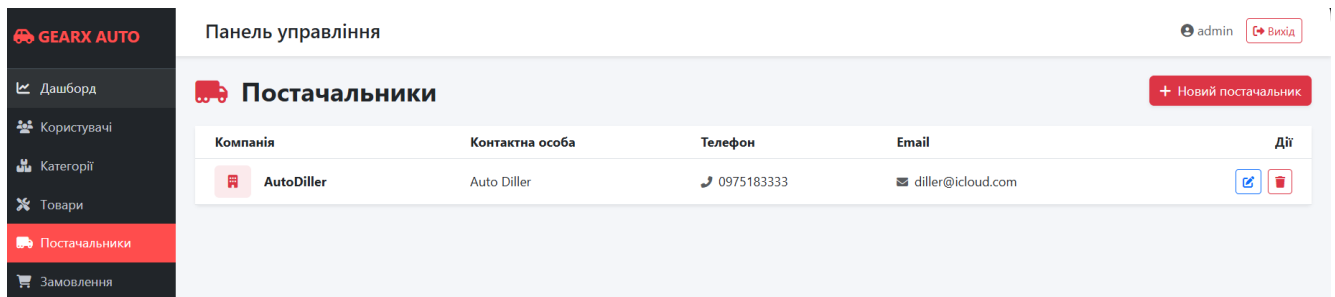


Рисунок 3.8 – Сторінка постачальників

Функціонал поповнення складських запасів реалізований через сторінку закупівель. Під час створення нової поставки менеджер може додавати необмежену кількість товарних позицій із зазначенням кількості та закупівельної вартості. Після підтвердження закупівлі система автоматично оновлює складські залишки, а зміни одразу відображаються у клієнтському інтерфейсі (рис. 3.9).

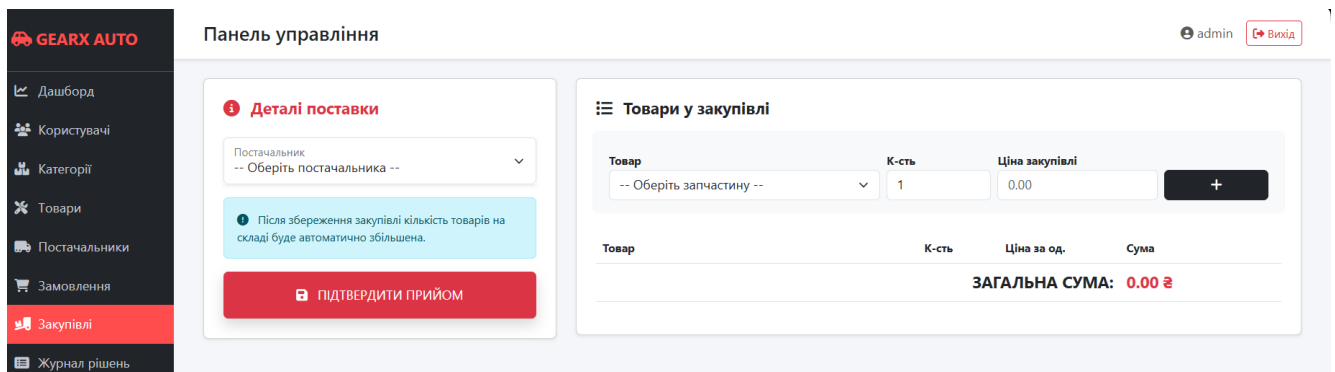


Рисунок 3.9 – Функціонал поповнення складських запасів

Для роботи із замовленнями реалізовано окрему сторінку, де адміністратор може переглядати інформацію про покупки клієнтів, аналізувати склад замовлення та змінювати його статус залежно від етапу виконання (рис. 3.10).

GEARX AUTO

Дашборд

Користувачі

Категорії

Товари

Постачальники

Замовлення

Закупівлі

Журнал рішень

Підтримка рішень

Панель управління

adminВихід

Керування замовленнями

№ / Дата	Клієнт	Сума	Статус	Дії
#13 15.05.2026 10:03	oleh ivanov	4 260,00 ₴	Ново	Деталі
#12 17.01.2026 17:43	Ivan Ivanov	3 600,00 ₴	Відправлено	Деталі
#11 16.01.2026 20:33	Anton Derkach	20,00 ₴	Ново	Деталі
#10 11.01.2026 20:24	string string	10,00 ₴	Ново	Деталі
#9 11.01.2026 20:22	string string	10,00 ₴	Відправлено	Деталі

Рисунок 3.10 – Функціонал обробки замовлень

Система також містить модуль аналітики, який виконує обробку накопичених даних і формує звіти щодо популярності товарів, динаміки продажів та необхідності поповнення запасів. На основі цих даних генеруються рекомендації для менеджерів щодо закупівель та оптимізації асортименту.

Отже, адміністративна панель забезпечує повний контроль над основними процесами роботи магазину автозапчастин. Реалізований функціонал дозволяє швидко керувати товарами, категоріями, постачальниками, закупівлями та замовленнями, а також використовувати аналітичні можливості системи для підтримки управлінських рішень.

3.4 Керівництво користувача системи

Клієнтський інтерфейс розробленої інформаційної системи спрямований на забезпечення зручного та швидкого процесу вибору й придбання автозапчастин кінцевими користувачами. Для доступу до розширених можливостей платформи, таких як робота з кошиком, перегляд замовлень та особистих даних, користувач повинен пройти реєстрацію або виконати вхід до облікового запису (рис. 3.11).

Рисунок 3.11 – Форма реєстрації нового клієнта

Головна сторінка клієнтського вебзастосунку (рис. 3.12) реалізована у вигляді зручного та сучасного інтерфейсу, що забезпечує швидкий доступ до основних розділів каталогу. На ній відображаються категорії товарів та добірка популярних позицій, що дозволяє користувачеві оперативно перейти до потрібного розділу або переглянути актуальні пропозиції магазину.

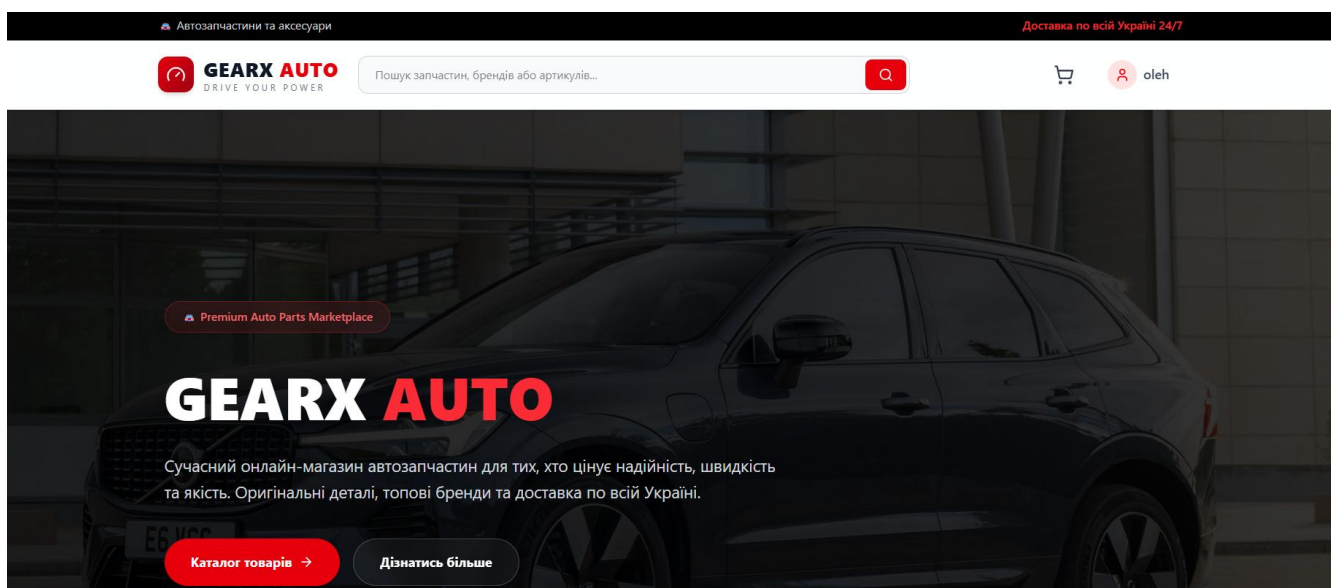


Рисунок 3.12 – Головна сторінка клієнтського сайту

Для полегшення пошуку необхідних автозапчастин реалізовано механізм пошуку та фільтрації за категоріями (рис. 3.13). Користувач може знайти потрібний товар за назвою або артикулом виробника. Каталог підтримує посторінковий вивід даних, що забезпечує зручну навігацію навіть при великій кількості товарів.

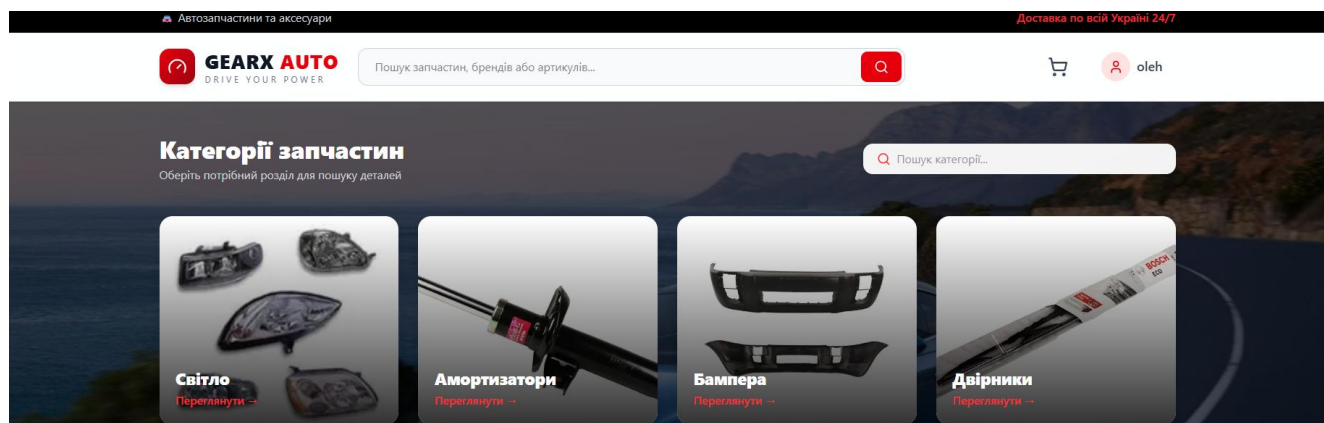


Рисунок 3.13 – Сторінка каталогу та результати пошуку

Сторінка детального перегляду товару містить повну інформацію про обрану позицію, включаючи опис, зображення та актуальну вартість. Також система автоматично відображає наявність товару на складі, що дозволяє користувачеві одразу визначити можливість його придбання та додати необхідну кількість у кошик (рис. 3.14).

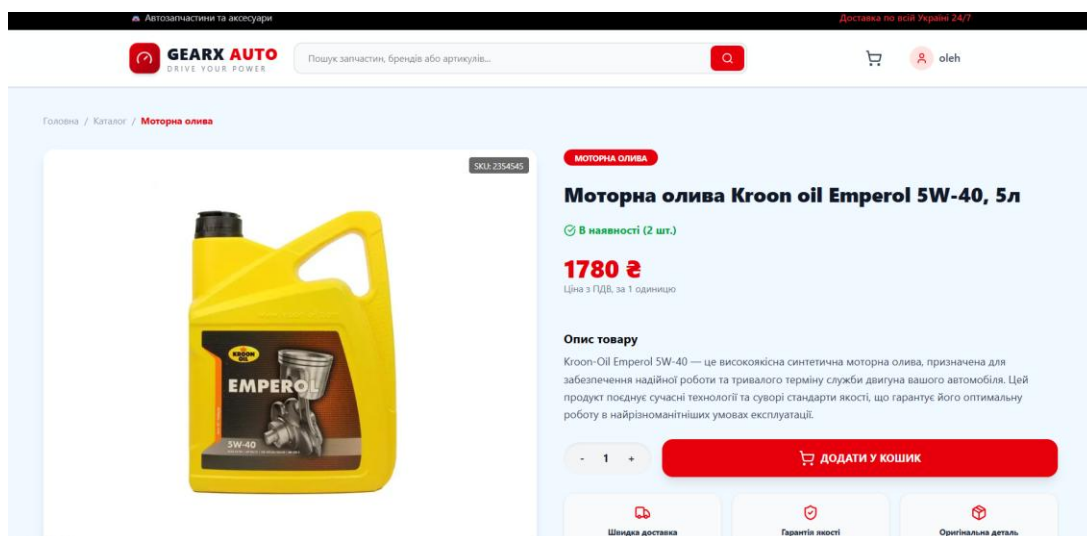


Рисунок 3.14 – Детальна інформація про обраний товар

У кошику покупець може керувати обраними товарами, змінювати їх кількість та в режимі реального часу бачити загальну суму замовлення. Після перевірки сформованого списку користувач переходить до оформлення покупки, після чого інформація передається до серверної частини та адміністративної панелі для подальшої обробки (рис. 3.15).

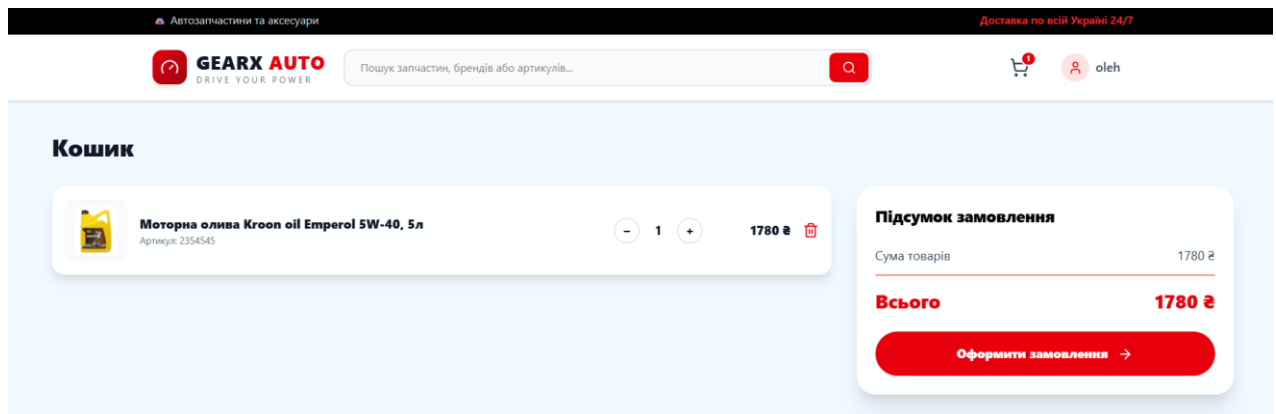


Рисунок 3.15 – Сторінка кошика товарів

Особистий кабінет користувача містить розділ із історією замовлень, де відображається статус кожної покупки. Для кожного замовлення доступна деталізована інформація щодо складу придбаних товарів та етапу його виконання.

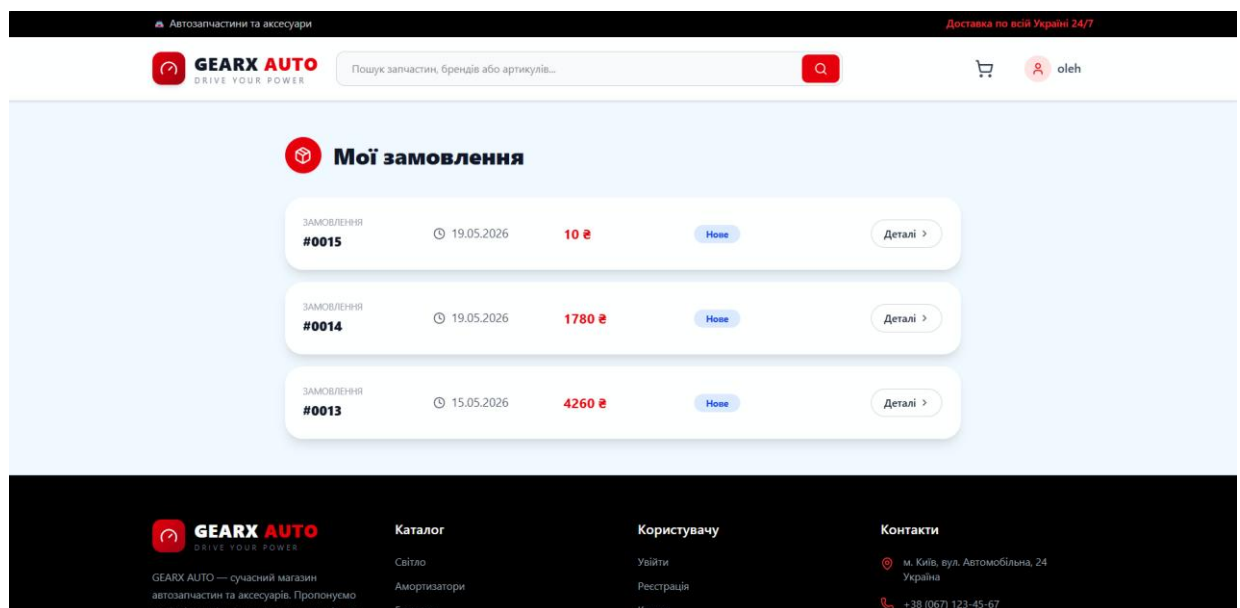


Рисунок 3.16 – Перегляд історії замовлень клієнтом

Додатково користувач може переглянути розширені відомості про кожне замовлення, зокрема перелік товарів, що входять до нього, їх кількість та загальну вартість (рис. 3.17).

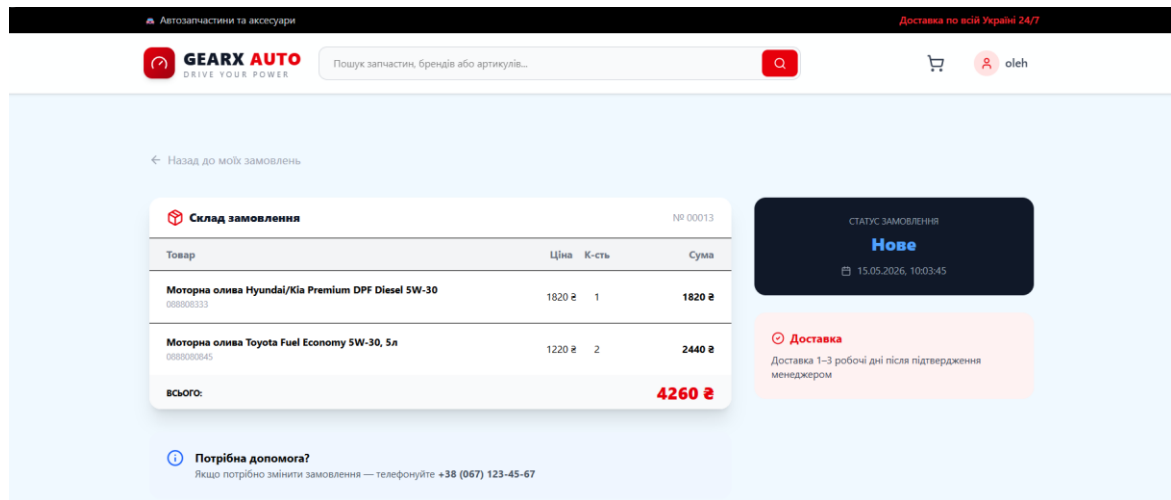


Рисунок 3.17 – Перегляд детальної інформації про замовлення

У цілому клієнтська частина системи забезпечує повноцінний цикл взаємодії користувача з інтернет-магазином: від пошуку товарів до оформлення замовлення та контролю його виконання. Це сприяє підвищенню зручності використання сервісу та рівня обслуговування клієнтів.

3.5 Тестування та оцінка працездатності системи

Тестування є важливим етапом розробки системи, оскільки дозволяє перевірити її відповідність вимогам та коректність роботи всіх функцій. У межах даного проєкту тестування виконувалося на кількох рівнях: функціональному, інтеграційному та користувацькому. Це дало змогу перевірити як окремі модулі системи, так і їх взаємодію між собою.

Спочатку була визначена загальна стратегія тестування, яка включала: перевірку окремих компонентів серверної частини, тестування взаємодії клієнта і

сервера через API, а також перевірку основних сценаріїв роботи користувача.

Серверна частина перевірялася за допомогою Swagger. Тестувалися основні ендпоінти для роботи з товарами, категоріями та замовленнями.

Для ProductsController перевірено:

- отримання списку товарів із пагінацією;
- пошук і фільтрацію;
- перегляд детальної інформації про товар.

Усі запити повертали коректні HTTP-відповіді та правильні дані.

Окремо тестувався OrdersController, оскільки оформлення замовлення є критичною операцією. Перевірялися такі сценарії:

- успішне створення замовлення;
- спроба замовити більше товару, ніж є на складі;
- замовлення з некоректними даними.

У всіх випадках система працювала коректно, а при помилках транзакція повністю скасовувалася, що забезпечує цілісність даних.

Було перевірено роботу транзакцій і зв'язків між таблицями. Після оформлення замовлення кількість товару на складі змінювалась правильно. Також перевірено каскадні зв'язки та цілісність даних – порушень не виявлено.

Клієнтська частина тестувалася через браузер шляхом перевірки основних сценаріїв:

- реєстрація та вхід користувача – працює коректно, токен авторизації зберігається;
- пошук товарів і додавання в кошик – дані оновлюються без перезавантаження сторінки;
- оформлення замовлення – замовлення створюється і відображається в кабінеті;
- робота адміністратора – замовлення змінює статус і синхронізується з клієнтом.

Усі сценарії виконуються без помилок.

Система перевірялась у Google Chrome, Mozilla Firefox та Microsoft Edge.

Інтерфейс і функціональність працюють коректно в усіх браузерах.

У результаті тестування підтверджено, що система працює стабільно, виконує всі основні функції та забезпечує коректну взаємодію між клієнтською і серверною частинами. Система готова до подальшого використання та доопрацювання.

3.6 Висновки до розділу 3

У третьому розділі було описано практичну реалізацію та тестування комп'ютерної системи магазину автозапчастин.

Серверна частина системи реалізована на платформі ASP.NET Core з використанням принципів Clean Architecture та SOLID, що забезпечило чітке розмежування рівнів відповідальності між компонентами. Для роботи з базою даних застосовано Entity Framework Core з підходом Code First, а доступ до ресурсів захищено через поєднання Cookie Authentication для адміністративної панелі та JWT-токенів для клієнтського API.

Клієнтська частина реалізована у вигляді SPA-застосунку на базі React із використанням Redux Toolkit для управління глобальним станом та RTK Query для взаємодії з серверним API. Окремо реалізовано адміністративну панель на базі ASP.NET Core MVC для роботи персоналу магазину.

Тестування системи проводилося на трьох рівнях. Перевірка API-ендпоінтів, механізмів транзакцій, цілісності даних та повних користувацьких сценаріїв підтвердила відповідність системи сформульованим вимогам. Крос-браузерна сумісність підтверджена для Google Chrome, Mozilla Firefox та Microsoft Edge.

Виявлені в процесі тестування незначні недоліки не впливають на основний функціонал системи та визначені як напрями подальшого вдосконалення.

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу підвищення ефективності діяльності магазину автозапчастин шляхом розробки комп'ютерної системи, яка забезпечує комплексну підтримку основних торговельних і управлінських процесів підприємства. У ході виконання роботи було проведено аналіз предметної області, досліджено особливості функціонування магазинів автотоварів та визначено основні проблеми, що виникають при використанні неавтоматизованих або застарілих засобів обліку.

У першому розділі роботи проаналізовано сучасний стан автоматизації торговельної сфери, розглянуто роль інформаційних систем у діяльності підприємств з продажу автозапчастин, а також виконано огляд існуючих програмних рішень. Проведений аналіз показав, що більшість наявних систем або мають надлишковий функціонал і високу вартість впровадження, або не враховують специфіку роботи невеликих магазинів автозапчастин. Це підтвердило актуальність розробки власної спеціалізованої системи.

У другому розділі було сформовано функціональні та технічні вимоги до системи, обґрунтовано вибір технологій для реалізації серверної та клієнтської частин застосунку, а також спроектовано структуру бази даних. Для реалізації серверної частини було обрано платформу ASP.NET Core та архітектуру REST API, що забезпечують надійність, масштабованість і зручність інтеграції. Клієнтську частину реалізовано у вигляді SPA-застосунку на базі React та TypeScript, що дозволило створити сучасний і зручний користувацький інтерфейс.

У третьому розділі виконано практичну реалізацію комп'ютерної системи, розроблено серверну та клієнтську частини застосунку, реалізовано механізми взаємодії з базою даних, автентифікації користувачів та управління товарами й замовленнями. Також було підготовлено інструкцію адміністратора та керівництво користувача, що спрощує впровадження і подальше використання системи.

Проведене тестування підтвердило працездатність програмного забезпечення та відповідність реалізованого функціоналу поставленим вимогам.

Результати виконаної роботи свідчать про те, що розроблена система може бути використана для автоматизації діяльності магазинів автозапчастин малого та середнього масштабу. Використання системи дозволяє підвищити ефективність обліку товарів, прискорити процес обслуговування клієнтів, зменшити кількість помилок під час оформлення замовлень та покращити контроль за складськими залишками.

Цінність роботи полягає у можливості подальшого впровадження та розширення функціональних можливостей системи відповідно до потреб конкретного підприємства. У перспективі система може бути доповнена інтеграцією з платіжними сервісами, службами доставки, CRM-модулями та аналітичними інструментами для прогнозування попиту й підтримки управлінських рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Авто новини від AUTO-Consulting. AUTO-Consulting. URL: <https://autoconsulting.ua/article.php?sid=57949> (дата звернення: 10.04.2026).
2. Інноваційні технології в бізнесі : навч. посіб. Одеса : Міжнар. гуманітар. ун-т, 2025. 87 с.
3. Загальна характеристика інформаційних систем. URL: <https://klaster.ua/ua/stati-i-obzory/kharakteristiki-informacionnykh-sistem-ru/> (дата звернення: 10.04.2026).
4. Що таке автоматизація роздрібної торгівлі? Програма для автоматизації роздрібної торгівлі. URL: <https://datawiz.io/uk/blog/what-is-retail-automation> (дата звернення: 10.04.2026).
5. Що таке ERP? | Oracle Україна. URL: <https://www.oracle.com/ua/erp/what-is-erp/> (дата звернення: 12.04.2026).
6. Найкращі POS-системи в Україні – Незалежний огляд – ChoiceQR. URL: <https://choiceqr.com/uk/news/yaku-pos-systemu-obraty-zakladam-v-ukraini/> (дата звернення: 12.04.2026).
7. CRM | SendPulse Marketing 101 UA. URL: <https://sendpulse.ua/support/glossary/crm> (дата звернення: 13.04.2026).
8. Understanding NIST's Five Essential Characteristics of Cloud Computing | LinkedIn. URL: <https://www.linkedin.com/pulse/understanding-nists-five-essential-characteristics-cloud-computing-uvmie/> (дата звернення: 13.04.2026).
9. Autoselling - спеціалізована онлайн-програма. URL: <https://autoselling.eu> (дата звернення: 10.04.2026).
10. Автоматизація рітейлу: ТОП-6 систем для магазинів і торгових мереж. URL: <https://www.oneservice-consulting.com/avtomatizatsiia-riteilu-top-6-sistem> (дата звернення: 10.04.2026).
11. Кравець Р. Б., Серов Ю. О., Марковець О. В. Інформаційні технології організації бізнесу : посібник. Львів : Львівська політехніка, 2013. 228 с.

12. Менеджмент малих і середніх підприємств. Укладачі: Олійченко І.М., Дітковська М.Ю. Чернігів: НУ «Чернігівська політехніка», 2024. – 60 с.

13. Power D. J. Decision Support Systems: Concepts and Resources for Managers. – Journal of Management Information Systems, 2015.

14. Переваги, тренди та перспективи розробки на C#/.NET | nt.ua. URL: <https://nt.ua/advantages-trends-and-prospects-for-development-in-c-sharp-and-dot-net> (дата звернення: 14.04.2026).

15. Introduction to LINQ in C#. Language Integrated Query (LINQ) | Medium. URL: <https://medium.com/@ravipatel.it/introduction-to-linq-in-c-26bf70607d14> (дата звернення: 14.04.2026).

16. Introduction to .NET Framework – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/c-sharp/introduction-to-net-framework/> (дата звернення: 14.04.2026).

17. Yan, X. (2024). Web API Development with ASP.NET Core 8: Learn techniques, patterns, and tools for building high-performance, robust, and scalable web APIs. Packt Publishing. 804 с.

18. Entity Framework – Wikipedia. URL: https://en.wikipedia.org/wiki/Entity_Framework (дата звернення: 14.04.2026).

19. Single Page Applications: A Powerful Design Pattern for Modern Web Apps | by Mary Boyd | A Lady Dev | Medium. URL: <https://medium.com/a-lady-dev/single-page-applications-a-powerful-design-pattern-for-modern-web-apps-ec3590bb7e7a> (дата звернення: 17.04.2026).

20. Understanding Virtual DOM in React | Refine. URL: <https://refine.dev/blog/react-virtual-dom/#introduction> (дата звернення: 17.04.2026).

21. Статична типізація – React. URL: <https://uk.legacy.reactjs.org/docs/static-type-checking.html> (дата звернення: 17.04.2026).

22. Getting Started with Redux | Redux. URL: <https://redux.js.org/introduction/getting-started> (дата звернення: 17.04.2026).

23. Дія.Освіта – IT-студії. URL: <https://it-osvita.diia.gov.ua/task/item/c1d56d2b-718f-4490-8498-ee209eac30d8> (дата звернення: 19.04.2026).

24. Distinctive Features Of SQLite. URL:
<https://tool.oschina.net/uploads/apidocs/sqlite/different.html> (дата звернення:
 17.04.2026).

25. Етапи проектування бази даних – Вікі ЦДУ. URL:
https://wiki.cusu.edu.ua/index.php/Етапи_проектування_бази_даних (дата
 звернення: 27.04.2026).

26. Зв'язок у базі даних: основні типи та як встановлюється. URL:
<https://foxminded.ua/zviazok-u-bazi-danykh/> (дата звернення: 27.04.2026).

27. Basics of Clean Architecture with C# – We code IT happen. URL:
<https://www.macrix.eu/en/basics-of-clean-architecture-with-c/> (дата звернення:
 05.05.2026).

28. Barklund Morten, Mardan Azat. React Quickly, Second Edition. New York :
 Manning Publications, 2023. 456 с.

ДОДАТОК А

КОД ПРОЄКТУ

Лістинг А.1 – Конфігурація серверної частини Program.cs

```

using GEARXAuto.Domain.Entities;
using GEARXAuto.Infrastructure.Data;
using GEARXAuto.Services.Interfaces;
using GEARXAuto.Services.Services;
using GEARXAuto.Web.Mapper;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using Microsoft.OpenApi.Models;

namespace GEARXAuto.Web
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");

            builder.Services.AddDbContext<AppDbContext>(options =>
                options.UseSqlite(connectionString));

            builder.Services.AddScoped<IImageService,
ImageService>();
            builder.Services.AddScoped<IDecisionService,
DecisionService>();

            builder.Services.AddAutoMapper(cfg => { },
typeof(MappingProfile));
            builder.Services.AddScoped<IPasswordHasher<User>,
PasswordHasher<User>>();

builder.Services.AddAuthentication(CookieAuthenticationDefaults.Auth
enticationScheme)
                .AddCookie(options =>
                {
                    options.LoginPath = "/Account/Login";

```

```

        options.AccessDeniedPath =
"/Account/AccessDenied";
    })

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options =>
{
    options.TokenValidationParameters = new
TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer =
builder.Configuration["Jwt:Issuer"],
        ValidAudience =
builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["J
wt:Key"]))
    };
});

builder.Services.AddControllersWithViews();
builder.Services.AddEndpointsApiExplorer();

builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowReact", policy =>
    {
        policy.WithOrigins("http://localhost:5173") //
порт
        .AllowAnyHeader()
        .AllowCredentials()
        .AllowAnyMethod();
    });
});

builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new
Microsoft.OpenApi.Models.OpenApiInfo
    {
        Title = "GEARXAuto API",
        Version = "v1",
        Description = "API для інтернет-магазину
автозапчастин",
        Contact = new
Microsoft.OpenApi.Models.OpenApiContact
        {
            Name = "GEARXAuto",
            Email = "support@gearxauto.com"
        }
    }
});

```

```

    });

    options.AddSecurityDefinition("Bearer", new
Microsoft.OpenApi.Models.OpenApiSecurityScheme
    {
        Name = "Authorization",
        Type =
Microsoft.OpenApi.Models.SecuritySchemeType.Http,
        Scheme = "bearer",
        BearerFormat = "JWT",
        In =
Microsoft.OpenApi.Models.ParameterLocation.Header,
        Description = "Введіть JWT токен у форматі:
Bearer {token}"
    });

    options.AddSecurityRequirement(new
Microsoft.OpenApi.Models.OpenApiSecurityRequirement
    {
        {
            new
Microsoft.OpenApi.Models.OpenApiSecurityScheme
            {
                Reference = new
Microsoft.OpenApi.Models.OpenApiReference
            {
                Type =
Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,
                Id = "Bearer"
            },
            new string[] {}
        }
    });
});

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
    DbSeeder.SeedAdminUser(scope.ServiceProvider);
}

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}

app.UseSwagger();
app.UseSwaggerUI();

app.UseStaticFiles();

```

```

app.UseRouting();
app.UseCors("AllowReact");

app.UseAuthentication();
app.UseAuthorization();
app.MapStaticAssets();

app.MapControllerRoute(
    name: "areas",
    pattern:
"{area:exists}/{controller=Home}/{action=Index}/{id?}")
    .RequireAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

//app.MapControllerRoute(
//    name: "default",
//    pattern:
"{controller=Account}/{action=Login}/{id?}")
//    .WithStaticAssets();

app.Run();
}
}
}

```

Лістинг А.2 – Контекст бази даних AppDbContext

```

using GEARXAuto.Domain.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Data;
using System.Text;

namespace GEARXAuto.Infrastructure.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
        : base(options) { }

        public DbSet<User> Users { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<Customer> Customers { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }
    }
}

```

```

    public DbSet<Product> Products { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<OrderItem> OrderItems { get; set; }
    public DbSet<Purchase> Purchases { get; set; }
    public DbSet<PurchaseItem> PurchaseItems { get; set; }
    public DbSet<StockMovement> StockMovements { get; set; }
    public DbSet<DecisionLog> DecisionLogs { get; set; }

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Customer>()
            .HasOne(c => c.User)
            .WithOne(u => u.Customer)
            .HasForeignKey<Customer>(c => c.UserId)
            .OnDelete(DeleteBehavior.Cascade);

        modelBuilder.Entity<Product>()
            .HasOne(p => p.Category)
            .WithMany(c => c.Products)
            .HasForeignKey(p => p.CategoryId);

        modelBuilder.Entity<OrderItem>()
            .HasOne(oi => oi.Order)
            .WithMany(o => o.OrderItems)
            .HasForeignKey(oi => oi.OrderId);

        modelBuilder.Entity<Product>()
            .Property(p => p.Price)
            .HasConversion<double>();

        modelBuilder.Entity<Role>().HasData(
            new Role { Id = 1, Name = "Admin" },
            new Role { Id = 2, Name = "Manager" },
            new Role { Id = 3, Name = "Customer" }
        );
    }
}

```

Лістинг А.3 – Маппер для сутностей MappingProfile

```

using AutoMapper;
using GEARXAuto.Domain.Entities;
using GEARXAuto.Web.DTOs;
using GEARXAuto.Web.Models;

namespace GEARXAuto.Web.Mapper
{

```

```

public class MappingProfile : Profile
{
    public MappingProfile()
    {
        ///
        /// Category Mapping
        ///
        CreateMap<Category, CategoryDto>();

        CreateMap<Category, CategoryVM>();

        CreateMap<Category, CategoryEditVM>()
            .ForMember(dest => dest.ExistingImage, opt =>
opt.MapFrom(src => src.Image));

        CreateMap<CategoryEditVM, Category>()
            .ForMember(dest => dest.Image, opt => opt.Ignore());

        ///
        /// Product Mapping
        ///
        CreateMap<Product, ProductDto>()
            .ForMember(dest => dest.CategoryName,
                opt => opt.MapFrom(src =>
src.Category.Name));

        CreateMap<ProductDto, Product>();

        ///
        /// User Mapping
        ///
        CreateMap<User, UserVM>()
            .ForMember(dest => dest.RoleName, opt =>
opt.MapFrom(src => src.Role.Name));

        CreateMap<UserEditVM, User>().ReverseMap();

        ///
        /// Supplier Mapping
        ///
        CreateMap<Supplier, SupplierVM>();
        CreateMap<Supplier, SupplierEditVM>().ReverseMap();

        ///
        /// Product Mapping
        ///
        CreateMap<Product, ProductVM>()
            .ForMember(dest => dest.CategoryName, opt =>
opt.MapFrom(src => src.Category.Name))
            .ForMember(dest => dest.SupplierName, opt =>
opt.MapFrom(src => src.Supplier.Name));
    }
}

```

```

        CreateMap<Product, ProductEditVM>()
            .ForMember(dest => dest.ExistingImage, opt =>
opt.MapFrom(src => src.Image))
            .ForMember(dest => dest.Categories, opt =>
opt.Ignore())
            .ForMember(dest => dest.Suppliers, opt =>
opt.Ignore());

        CreateMap<ProductEditVM, Product>()
            .ForMember(dest => dest.Image, opt => opt.Ignore());

        ///
        /// Purchase Mapping
        ///
        CreateMap<Purchase, PurchaseVM>()
            .ForMember(dest => dest.SupplierName, opt =>
opt.MapFrom(src => src.Supplier.Name))
            .ForMember(dest => dest.ItemsCount, opt =>
opt.MapFrom(src => src.PurchaseItems.Count));

        ///
        /// Order Mapping
        ///
        CreateMap<Order, OrderVM>()
            .ForMember(d => d.CustomerName, opt => opt.MapFrom(s
=> s.Customer.FirstName + " " + s.Customer.LastName));

        CreateMap<Order, OrderDetailsVM>()
            .ForMember(d => d.CustomerName, opt => opt.MapFrom(s
=> s.Customer.FirstName + " " + s.Customer.LastName))
            .ForMember(d => d.CustomerPhone, opt =>
opt.MapFrom(s => s.Customer.Phone))
            .ForMember(d => d.CustomerEmail, opt =>
opt.MapFrom(s => s.Customer.Email))
            .ForMember(d => d.Items, opt => opt.MapFrom(s =>
s.OrderItems));

        CreateMap<OrderItem, OrderItemVM>()
            .ForMember(d => d.ProductName, opt => opt.MapFrom(s
=> s.Product.Name))
            .ForMember(d => d.SKU, opt => opt.MapFrom(s =>
s.Product.SKU));
    }
}
}

```

Лістинг А.4 – Сервіс взаємодії з API orderApi.ts

```

import { createApi } from '@reduxjs/toolkit/query/react';
import { createBaseQuery } from '../../utils/baseQuery';

```

```

export interface IOrderItemRequest {
  productId: number;
  quantity: number;
}

export interface IOrderRequest {
  customerId: number;
  items: IOrderItemRequest[];
}

export interface IOrderResponse {
  id: number;
  orderDate: string;
  totalAmount: number;
  status: string;
}

export interface IOrderItem {
  productName: string;
  sku: string;
  quantity: number;
  unitPrice: number;
  subtotal: number;
}

export interface IOrderDetailsResponse {
  id: number;
  orderDate: string;
  totalAmount: number;
  status: string;

  customerName: string;
  customerPhone: string;
  customerEmail: string;

  items: IOrderItem[];
}

export interface IOrderResponse {
  id: number;
  orderDate: string;
  totalAmount: number;
  status: string;
  customerName: string;
}

export const orderApi = createApi({
  reducerPath: 'orderApi',
  baseQuery: createBaseQuery('orders'),
  tagTypes: ['Orders'],
  endpoints: builder => ({
    createOrder: builder.mutation<
      { orderId: number; message: string },
      IOrderRequest

```

```

>({
  query: orderData => ({
    url: 'checkout',
    method: 'POST',
    body: orderData,
  }),
  invalidatesTags: ['Orders'],
}),

getMyOrders: builder.query<IOrderResponse[], number>({
  query: customerId => ({
    url: 'my-orders',
    params: { customerId },
  }),
  providesTags: ['Orders'],
}),

getOrderDetails: builder.query<IOrderDetailsResponse, number |
string>({
  query: id => `details/${id}`,
  providesTags: ['Orders'],
}),
}),
});

export const {
  useCreateOrderMutation,
  useGetMyOrdersQuery,
  useGetOrderDetailsQuery,
} = orderApi;

```

Лістинг А.5 – Сторінка деталей запчастин ProductDetailsPage.tsx

```

import { useState } from 'react';
import { useParams, Link } from 'react-router-dom';
import { useGetProductByIdQuery } from '../services/product';
import {
  ShoppingCart,
  CheckCircle,
  XCircle,
  Package,
  Truck,
  ShieldCheck,
} from 'lucide-react';

```

```

import { IMAGE_URL } from '../utils/envData.ts';

import { useAppDispatch } from '../app/hooks';
import { addToCart } from '../app/cartSlice';

const ProductDetailsPage = () => {
  const { id } = useParams<{ id: string }>();
  const dispatch = useAppDispatch(); // Ініціалізуємо dispatch

  const { data: product, isLoading, isError } =
    useGetProductByIdQuery(id!);

  const [quantity, setQuantity] = useState(1);

  const handleAddToCart = () => {
    if (product) {
      dispatch(
        addToCart({
          product: product,
          quantity: quantity,
        })
      );
    }
  };

  if (isLoading)
    return (
      <div className="flex justify-center items-center py-24">
        <div className="animate-spin rounded-full h-12 w-12 border-
b-2 border-red-600"></div>
      </div>
    );

  if (isError || !product)
    return (

```

```

    <div className="max-w-7xl mx-auto px-6 py-20 text-center">
      <div className="bg-red-50 text-red-600 p-6 rounded-2xl
inline-block">
        Товар не знайдено або сталася помилка завантаження.
      </div>
    </div>
  );

return (
  <div className="container mx-auto py-10 px-4">
    <nav aria-label="breadcrumb" className="mb-8">
      <ol className="flex flex-wrap gap-2 text-gray-400 text-sm">
        <li>
          <Link to="/" className="hover:text-red-600 transition">
            Головна
          </Link>
        </li>
        <li></li>
        <li>
          <Link to="/products" className="hover:text-red-600
transition">
            Каталог
          </Link>
        </li>
        <li></li>
        <li className="text-red-600 font-
bold">{product.categoryName}</li>
      </ol>
    </nav>

    <div className="grid grid-cols-1 lg:grid-cols-2 gap-10">
      <div className="relative group rounded-2xl overflow-hidden
shadow-lg bg-white p-6 flex items-center justify-center">
        <img
          src={

```

```

        product.image
        ? `${IMAGE_URL}/600_${product.image}`
        : 'https://via.placeholder.com/600'
    }
    alt={product.name}
    className="w-full h-full object-contain transition-
transform duration-500 group-hover:scale-105"
    style={{ maxHeight: '500px' }}
  />
  <span className="absolute top-3 right-3 bg-black/60 text-
white text-xs px-2 py-1 rounded">
    SKU: {product.sku}
  </span>
</div>

<div className="space-y-6">
  <span className="inline-block bg-red-600 text-white px-4
py-1 rounded-full text-xs font-bold uppercase">
    {product.categoryName}
  </span>

  <h1 className="text-3xl font-extrabold text-gray-900">
    {product.name}
  </h1>

  <div className="flex items-center gap-4">
    {product.stockQuantity > 0 ? (
      <span className="flex items-center text-green-600
font-bold">
        <CheckCircle size={18} className="mr-1" /> B
наявності (
          {product.stockQuantity} шт.)
        </span>
      ) : (
        <span className="flex items-center text-red-600 font-
bold">

```

```

        <XCircle size={18} className="mr-1" /> Немає в
наявності

    </span>
  )}
</div>

<div className="space-y-2">
  <span className="text-4xl font-extrabold text-red-600">
    {product.price} ₾
  </span>
  <p className="text-gray-500 text-sm">Ціна з ПДВ, за 1
одиницю</p>
</div>

<hr className="border-gray-200 opacity-30" />

<div>
  <h5 className="text-lg font-bold mb-2">Опис товару</h5>
  <p className="text-gray-700 leading-relaxed">
    {product.description ||
      'Опис для даного товару наразі відсутній. Будь
ласка, зверніться до менеджера для уточнення технічних
характеристик.'}
  </p>
</div>

<div className="flex items-center gap-4">
  <div className="flex items-center bg-gray-100 rounded-
full shadow-inner px-3 py-1">
    <button
      className="text-gray-700 font-bold px-3 py-2
hover:text-red-600 transition"
      onClick={() => setQuantity(Math.max(1, quantity -
1))}
    >
      -
    </button>
  </div>
  <div className="flex items-center bg-gray-100 rounded-
full shadow-inner px-3 py-1">
    <button
      className="text-gray-700 font-bold px-3 py-2
hover:text-red-600 transition"
      onClick={() => setQuantity(Math.max(1, quantity +
1))}
    >
      +
    </button>
  </div>
</div>

```

```

        </button>
        <input
          type="text"
          value={quantity}
          readOnly
          className="w-10 text-center bg-transparent outline-
none font-bold text-gray-900"
        />
        <button
          className="text-gray-700 font-bold px-3 py-2
hover:text-red-600 transition"
          onClick={() =>
            setQuantity(Math.min(product.stockQuantity,
quantity + 1))
          }
        >
          +
        </button>
      </div>

      <button
        onClick={handleAddToCart}
        className="flex items-center justify-center bg-red-600
hover:bg-red-700 text-white px-8 py-4 rounded-2xl font-bold shadow-
lg transition-all active:scale-95 flex-grow disabled:bg-gray-400
disabled:cursor-not-allowed"
        disabled={product.stockQuantity === 0}
      >
        <ShoppingCart className="mr-2" /> ДОДАТИ У КОШИК
      </button>
    </div>

    <div className="grid grid-cols-3 gap-4 mt-4">
      <div className="flex flex-col items-center p-4 bg-gray-
50 rounded-2xl border border-gray-100 shadow-sm">
        <Truck size={24} className="text-red-600 mb-2" />

```

```

        <span className="text-xs font-bold text-gray-800 text-
center">
            Швидка доставка
        </span>
    </div>

    <div className="flex flex-col items-center p-4 bg-gray-
50 rounded-2xl border border-gray-100 shadow-sm">
        <ShieldCheck size={24} className="text-red-600 mb-2"
/>

        <span className="text-xs font-bold text-gray-800 text-
center">
            Гарантія якості
        </span>
    </div>

    <div className="flex flex-col items-center p-4 bg-gray-
50 rounded-2xl border border-gray-100 shadow-sm">
        <Package size={24} className="text-red-600 mb-2" />
        <span className="text-xs font-bold text-gray-800 text-
center">
            Оригінальна деталь
        </span>
    </div>
</div>
</div>
</div>
);
};

export default ProductDetailsPage;

```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

**РОЗРОБКА КОМП'ЮТЕРНОЇ
СИСТЕМИ МАГАЗИНУ
АВТОЗАПЧАСТИН**

дипломна робота бакалавра

Виконав: Олексій КУЗНЕЦОВ
Студент групи КНТ-512

Керівник: Степан СКРУПСЬКИЙ
Доцент, к.т.н

Рисунок Б.1 – Слайд 1

2

Предмет розробки – комп'ютерна система автоматизації торговельної діяльності магазину автозапчастин, що охоплює серверну частину, клієнтський веб-застосунок та адміністративну панель.

Мета роботи – підвищення ефективності торговельної та управлінської діяльності магазину автозапчастин за рахунок розробки спеціалізованої комп'ютерної системи.

Новизна роботи – поєднання функціоналу класичних ERP/POS-систем із публічним клієнтським SPA-застосунком, що дозволяє покупцям самостійно шукати та замовляти запчастини онлайн.

Практична цінність – готове до впровадження програмне рішення для малого та середнього бізнесу, яке оптимізує складський облік, прискорює обробку замовлень та підвищує якість обслуговування клієнтів.

Рисунок Б.2 – Слайд 2

АКТУАЛЬНІСТЬ ПРОБЛЕМИ

3

Чому автоматизація необхідна?

- Величезна номенклатура: типовий магазин оперує сотнями тисяч найменувань, ручний облік яких неможливий.
- Складність підбору: потрібен миттєвий пошук за сумісністю з автомобілем, наявністю аналогів та крос-номерів.
- Конкуренція: малому бізнесу необхідно надавати сервіс на рівні великих онлайн-платформ (Rozetka, AutoBaza), щоб вижити.
- Законодавчі вимоги: обов'язкова фіскалізація, ведення складського та бухгалтерського обліку вимагають надійної ІТ-основи.

Розробка спеціалізованої системи – це не просто покращення, а необхідна умова виживання та розвитку сучасного магазину.

Рисунок Б.3 – Слайд 3

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

4

Критерій		AutoSelling	Торгсофт	BAS	ERP FOSS
Галузева спеціалізація		Висока	Середня	Низька	Середня
Веб-каталог для покупців		Ні	Ні	Ні	Ні
Управління складом		Так	Так	Так	Так
Хмарний доступ		Так	Частково	Частково	Так
Адаптація під бізнес		Низька	Низька	Висока	Низька
Вартість впровадження		Висока	Середня	Висока	Середня
Жодна з популярних систем не надає клієнтам публічний веб-каталог для самостійного пошуку та замовлення запчастин.					

Рисунок Б.4 – Слайд 4

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО СИСТЕМИ

5

Група вимог		Зміст вимог
Для клієнта		Реєстрація, перегляд каталогу, пошук та фільтрація товарів, додавання в кошик, оформлення замовлень, перегляд історії покупок.
Для персоналу		Управління товарами та категоріями, облік закупівель, обробка замовлень (зміна статусів), ведення бази постачальників.
Аналітичні		Аналіз динаміки продажів, контроль складських залишків (виявлення дефіциту), аналітика популярності товарів.
Безпека		Розмежування прав доступу (Адміністратор, Менеджер, Клієнт), JWT-автентифікація для клієнтського API.

Рисунок Б.5 – Слайд 5

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

6

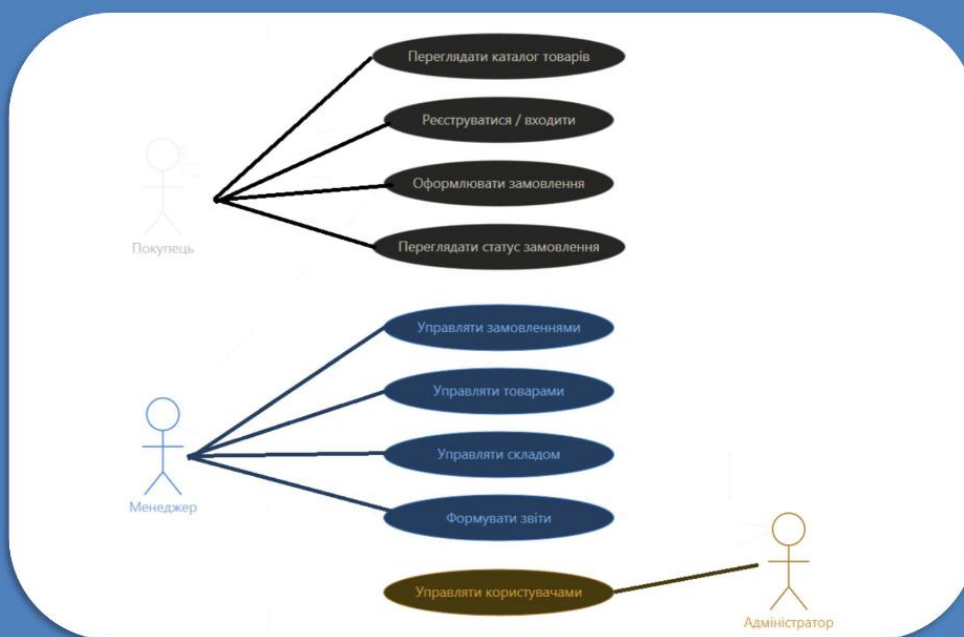


Рисунок Б.6 – Слайд 6

7

Рівень		Технологія
Серверна частина		ASP.NET Core Web API, C#, Entity Framework Core
Клієнтська частина		React (SPA), TypeScript, Redux Toolkit, RTK Query
Адмін-панель		ASP.NET Core MVC, Razor Views
База даних		SQLite
Документування API		Swagger

Рисунок Б.7 – Слайд 7

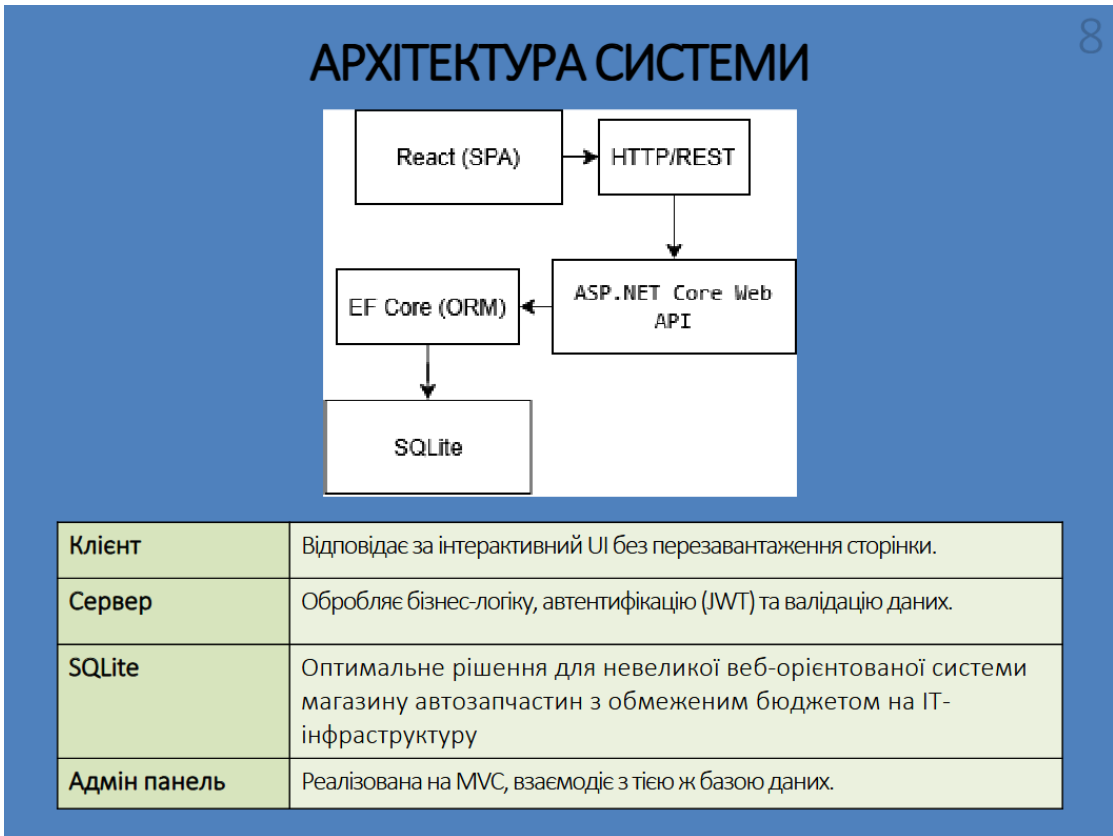


Рисунок Б.8 – Слайд 8

СТРУКТУРА БАЗИ ДАНИХ

9

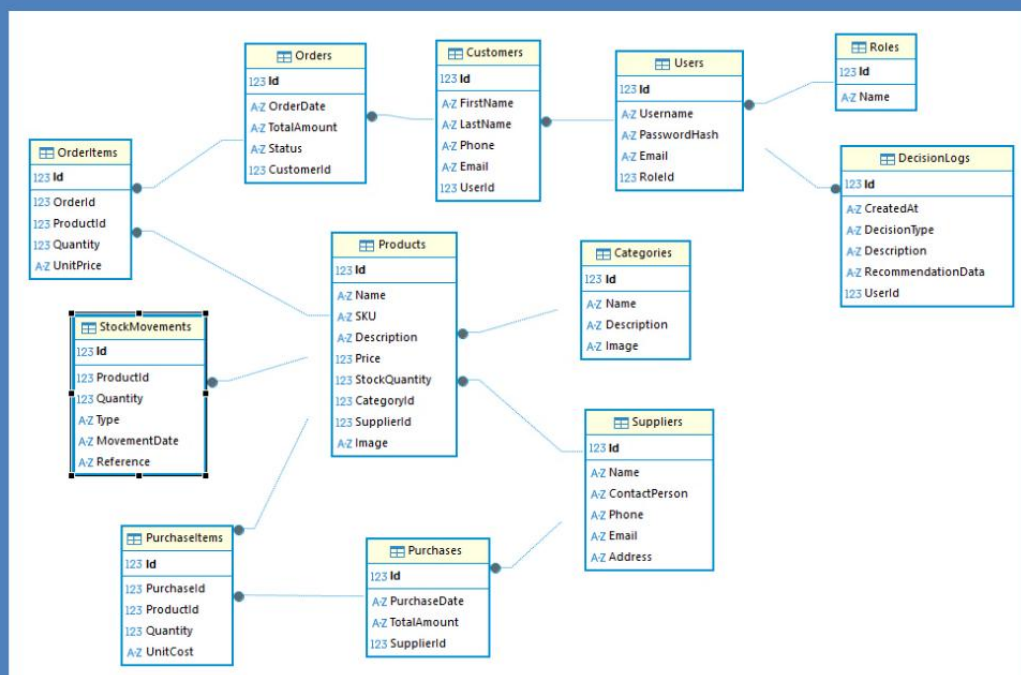
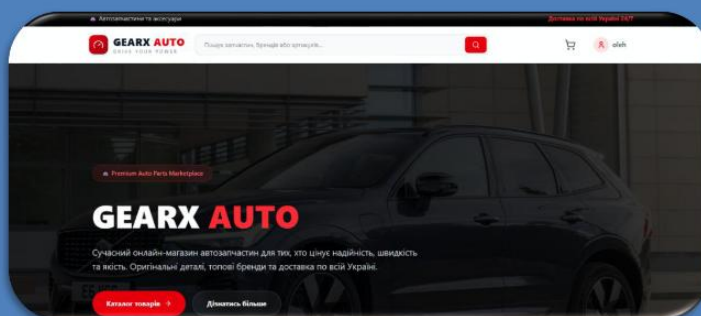


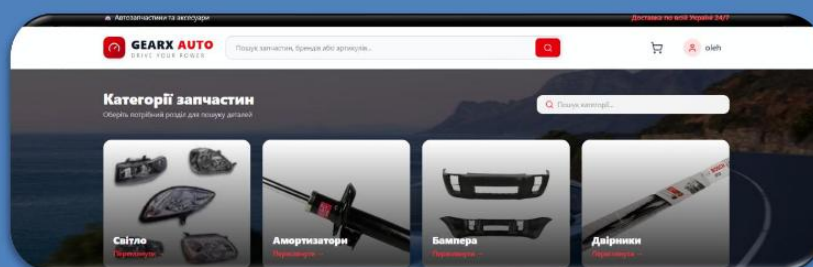
Рисунок Б.9 – Слайд 9

ФУНКЦІОНАЛ КЛІЄНТСЬКОГО ІНТЕРФЕЙСУ

10



Головна сторінка застосунку



Сторінка каталогу товарів

Рисунок Б.10 – Слайд 10

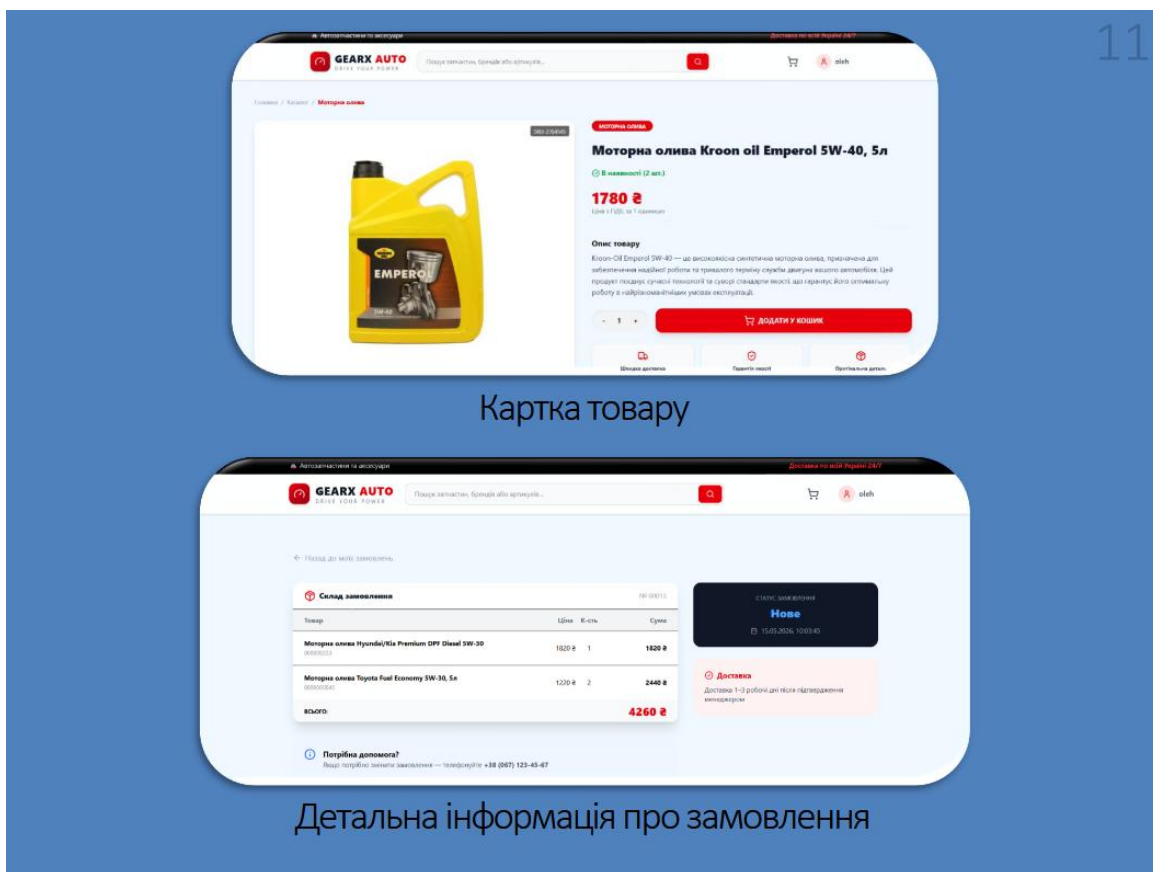


Рисунок Б.11 – Слайд 11

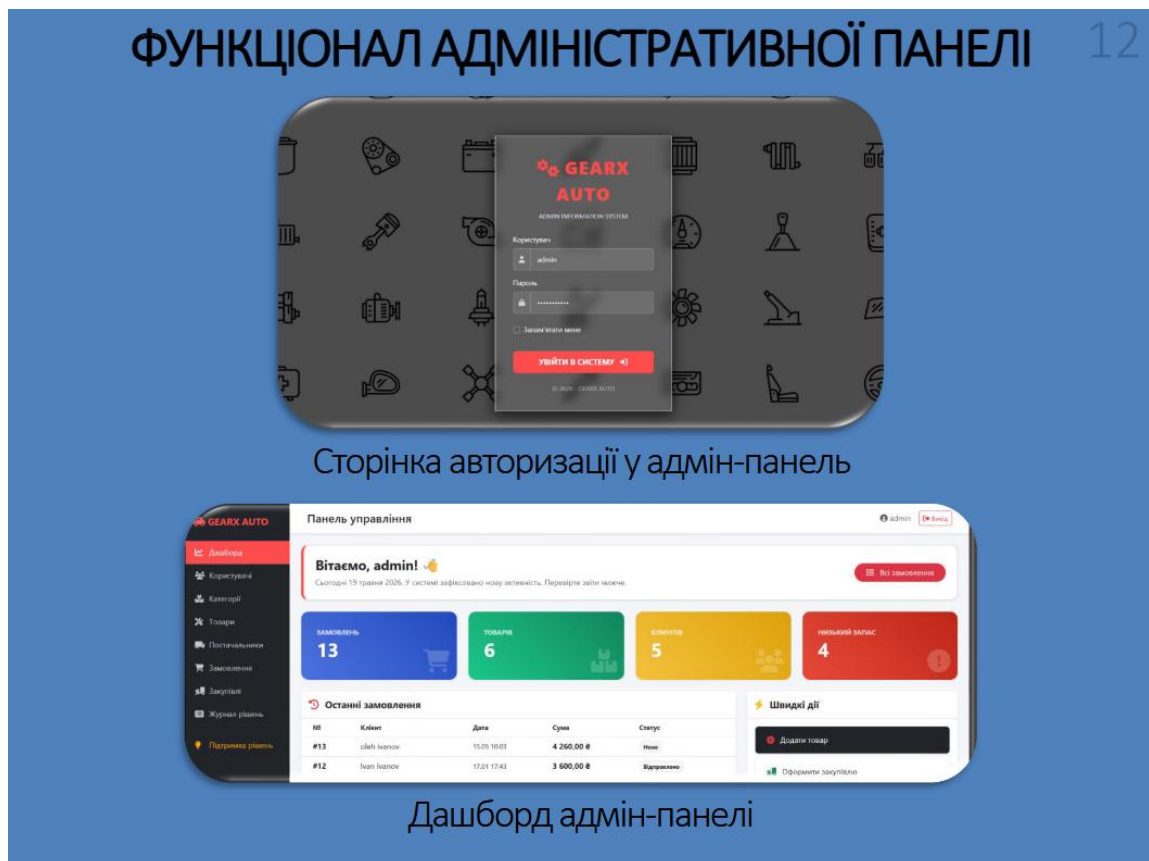


Рисунок Б.12 – Слайд 12

13

ТЕСТУВАННЯ СИСТЕМИ

Вид тестування	Що перевірялось	Результат
API-тестування (Swagger)	Всі ендпоінти, валідація даних, коректність HTTP-відповідей.	Успішно
Інтеграційне	Процес «Оформлення замовлення»: зміна складських залишків, транзакційність, цілісність даних.	Успішно
UI/UX-тестування	Повні користувацькі сценарії для Клієнта та Адміністратора.	Успішно
Крос-браузерне	Chrome, Firefox, Edge.	Успішно

Рисунок Б.13 – Слайд 13

14

ВИСНОВКИ

Проаналізовано предметну область, ринок існуючих рішень та виявлено їх основний недолік — відсутність публічного веб-інтерфейсу для клієнтів.
Спроектовано архітектуру веб-системи на базі ASP.NET Core Web API та React SPA.
Розроблено повноцінну комп'ютерну систему, що складається з клієнтського застосунку, адмін-панелі та серверної частини.
Реалізовано наскрізний процес від пошуку запчастин до оформлення замовлення з автоматичним складським обліком.
Досягнуто мети — підвищено ефективність роботи магазину за рахунок автоматизації рутинних операцій та надання сучасного інструменту продажів.

Рисунок Б.14 – Слайд 14

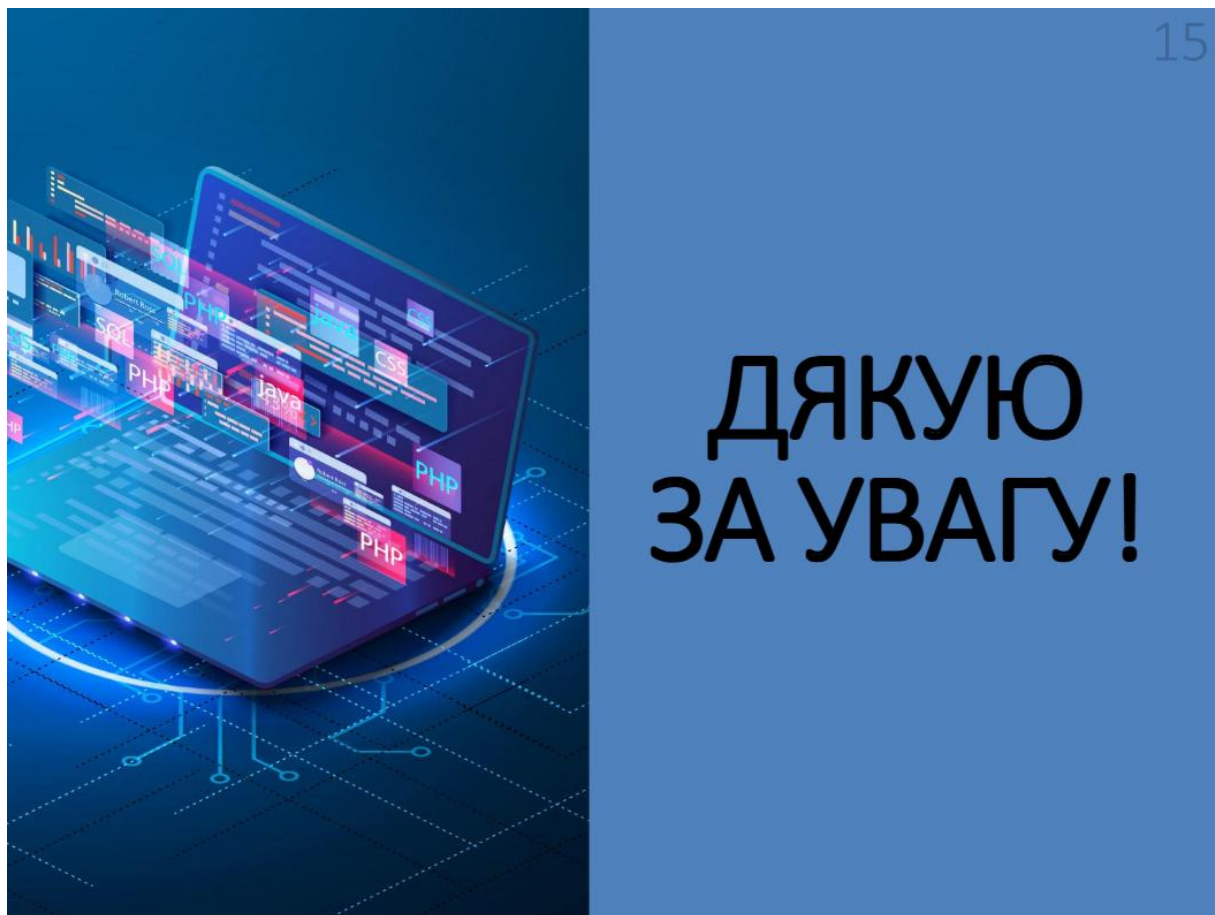


Рисунок Б.15 – Слайд 15