МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Кафедра інженерії програмного забезпечення

КУРСОВА РОБОТА

з дисципліни «Моделювання та аналіз програмного забезпечення»

на тему:

«Онлайн-сервіс для підбору автозапчастин»

студента 4 курсу групи ІПЗ-22-2
спеціальності 121 «Інженерія
програмного забезпечення»
Трофімчука Миколи Олександровича
(прізвище, ім'я та по-батькові)

Керівник: доцент кафедри КН
Левківський В.Л.

Дата захисту: "22" грудня 2025р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ Сугоняк І.І.
(підпис) (прізвище та ініціали)
_____ Панаріна І.В.
(підпис) (прізвище та ініціали)
_____ Левківський В.Л.
(підпис) (прізвище та ініціали)

Житомир – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Факультет інформаційно-комп'ютерних технологій
Кафедра інженерії програмного забезпечення
Освітній рівень: бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Завідувач кафедри ІПЗ
_____ Тетяна Вакалюк
"____"_____2025 р.

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТУ
Трофімчуку Миколі Олександровичу

1. Тема роботи: Онлайн-сервіс для підбору автозапчастин,
   керівник роботи: доцент кафедри КН Левківський В.Л.
2. Строк подання студентом: "22" грудня 2025 р.
3. Вихідні дані до роботи: Розробити архітектуру для онлайн-сервісу підбору автозапчастин.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці):
   1. Постановка завдання
   2. Аналіз вимог користувача
   3. Модель програмного комплексу на логічному рівні
   4. Фізична модель
   5. Прототип програмного комплексу

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
   1. Презентація
   2. Діаграми
6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "___" _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсової роботи	Строк виконання етапів проєкту	Примітки
1	Постановка задачі	20.09.2025	Виконано
2	Аналіз вимог користувача	13.10.2025	Виконано
3	Формулювання технічного завдання	16.10.2025	Виконано
4	Опрацювання літературних джерел	28.10.2025	Виконано
5	Моделювання системи на логічному рівні	01.11.2025	Виконано
6	Фізична модель	18.11.2025	Виконано
7	Генерування програмного коду для прототипу	12.12.2025	Виконано
8	Написання пояснювальної записки	14.12.2025	Виконано
9	Захист	22.12.2025	Виконано

Студент _____ Трофімчук М. О.
(підпис) (прізвище та ініціали)

Керівник проекту _____ Левківський В. Л.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсової роботи на тему «Онлайн-сервіс підбору автозапчастин» складається зі вступу, трьох розділів, висновків, джерел інформації та додатків.

Текстова частина викладена на 43 сторінках друкованого тексту.

Джерела інформації містять 15 найменувань і займають 2 сторінки. Пояснювальна записка має 4 сторінки додатків. У роботі наведено 15 рисунків. Загальний обсяг роботи – 51 сторінки.

У першому розділі було проаналізовано предметну область, розроблено технічне завдання на створення онлайн-сервісу, обґрунтовано вибір CASE-засобів для моделювання та детально описано функціональні й нефункціональні вимоги до програмного продукту.

У другому розділі розроблено логічну модель програмного комплексу. Описано алгоритми роботи та діаграму станів системи, побудовано об’єктно-орієнтовану модель (діаграму класів), а також визначено схеми комунікації та послідовності взаємодії об’єктів у процесі підбору та замовлення автозапчастин.

У третьому розділі спроєктовано фізичну модель системи: визначено діаграму компонентів та архітектуру розгортання програмного комплексу. Також виконано та описано процес генерації програмного коду для створення робочого прототипу системи.

Висновки підсумовують результати проєктування: від етапу аналізу вимог користувача до створення повноцінних UML-моделей та отримання каркасу програмного коду засобами автоматизованого проєктування.

У додатках наведено лістинг згенерованого програмного коду основних класів та глосарій термінів.

ЗМІСТ

ВСТУП

Станом на 2025 рік, сфера електронної комерції стрімко розвивається, охоплюючи все більше аспектів повсякденного життя. Автомобільний ринок не є виключенням: власники транспортних засобів дедалі частіше віддають перевагу онлайн-покупкам замість традиційних візитів до авторинків чи фізичних магазинів. Зростаюча цифровізація бізнес-процесів посилює значення спеціалізованих онлайн-сервісів, які дозволяють швидко та точно знаходити необхідні товари. Вплив таких систем на ринок автозапчастин є значною перспективою в сучасному світі та стає стандартом обслуговування клієнтів.

Система онлайн-підбору автозапчастин є як ніколи актуальним інструментом, який не лише спрощує процес купівлі, але й мінімізує ризики помилок при виборі сумісних деталей. Завдяки цій системі, автовласники та постачальники отримують єдину платформу для взаємодії через мережу Інтернет, що значно пришвидшує логістичні процеси та обмін інформацією про наявність товару.

Користувачі можуть ефективно працювати з каталогами, здійснювати пошук за VIN-кодом, маркою чи моделлю авто, порівнювати ціни та характеристики деталей, замовляючи їх у зручний час. Для постачальників та адміністраторів відкриваються можливості автоматизованого керування складом, обробки замовлень та аналізу попиту. Така система не лише заощаджує час учасників процесу, а й гарантує високу точність підбору компонентів, що є критично важливим для безпечної експлуатації автомобіля.

Метою курсової роботи є дослідження особливостей моделювання та аналізу програмних комплексів для онлайн-сервісу підбору автозапчастин з використанням CASE-технологій.

Завданням на курсову роботу є:

– аналіз теоретичних засад моделювання програмного забезпечення у сфері електронної комерції;

– аналіз та опис вимог користувачів до сервісу підбору запчастин;

– методи моделювання функцій та поведінки системи (Use Case, Activity, Statechart);

– проєктування об’єктної структури системи (Class Diagram);

– фізичне моделювання програмних комплексів (Component, Deployment Diagrams);

– кодогенерація із розроблених моделей.

Предметом дослідження є можливості застосування CASE-засобів проєктування програмного забезпечення для автоматизації продажів.

Об’єктом дослідження є методи та засоби проєктування програмного забезпечення та уніфікація процесу розробки інформаційних систем.

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1. Технічне завдання на розробку системи

Найменування програмного засобу

Повне найменування програмної системи: «Онлайн-сервіс підбору автозапчастин» (надалі «програма» або «Система»). Коротка назва програмної системи – «AutoParts».

Призначення розробки та область застосування

Програмна система призначена для автоматизації процесу пошуку, підбору та придбання автомобільних компонентів. Вона дозволяє покупцям знаходити запчастини за параметрами авто (VIN, марка, модель), а постачальникам — керувати складськими запасами та обробляти замовлення.

Мета

Онлайн-сервіс дозволить спростити взаємодію між власниками авто та постачальниками деталей, мінімізувати помилки при підборі сумісних запчастин та пришвидшити процес оформлення доставки.

Найменування розробника та замовника

Розробник даного продукту - студент групи ІПЗ-22-2 (надалі «розробник»).

Замовник програмного продукту - кафедра інженерії програмного забезпечення Державного університету «Житомирська політехніка» в межах виконання курсової з дисципліни «Моделювання та аналіз програмного забезпечення» (надалі «замовник»).

Документ на підставі якого ведеться розробка

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

Вимоги до функціональних характеристик

Програмна система має забезпечувати:

– аутентифікацію та авторизацію користувачів (Покупець, Постачальник, Адмін);

– ведення профілю автомобіля («Гараж») для швидкого пошуку;

– пошук запчастин за назвою, категорією або сумісністю;

– управління інвентарем (додавання/видалення/редагування запчастин постачальниками);

– оформлення замовлення з вибором стратегії доставки (самовивіз, пошта, кур'єр);

– відстеження статусу замовлення в реальному часі;

– систему відгуків та рейтингів товарів.

Організація вхідних і вихідних даних

Вхідними даними є реєстраційні дані користувачів, параметри автомобілів (VIN, рік, модель), характеристики запчастин (ціна, сумісність), параметри замовлення. Введення даних виконується через веб-форми.

Вихідними даними є результати пошуку, сформовані замовлення, історія покупок, інвойси, звіти про наявність товарів на складі. Виведення здійснюється у вигляді списків, карток товарів та статусних повідомлень.

Часові характеристики і розмір пам’яті, необхідної для роботи програми

Час реакції програми на дії користувача не повинен перевищувати 1 с.

Час пошуку запчастин у базі даних — не більше 2 с.

Доступність БД – 99% цілодобово.

Операції з’єднання з БД не більше 1 хвилини.

Обсяг оперативної пам’яті сервера, необхідний для роботи програми — не менше 4 Гб (для кешування результатів пошуку).

Вимоги до надійного функціонування

Програма повинна функціонувати у режимі 24/7. При апаратних збоях сервера: відновлення роботи шляхом перезавантаження та підняття резервної копії БД. При збоях ПЗ:

а) система повинна забезпечувати цілісність даних (ACID-транзакції при замовленні);

б) коректна обробка винятків (наприклад, спроба замовити відсутній товар).

Контроль вхідної і вихідної інформації

Для контролю коректності вхідної інформації та захисту від помилок оператора:

– валідація форм (формат email, перевірка VIN-коду, перевірка числових значень ціни);

– захист від SQL-ін’єкцій та XSS-атак;

– підтвердження дій користувача (наприклад, при видаленні акаунту).

Час відновлення після відмови

Час відновлення після критичного збою не повинен перевищувати 30 хвилин (час розгортання бекапу).

Умови експлуатації і збереження

Програма розгортається на веб-сервері, доступ здійснюється через браузер. Збереження даних забезпечується регулярним резервним копіюванням бази даних.

Вимоги до інформаційної і програмної сумісності.

Кросбраузерність: Chrome, Firefox, Safari, Edge. Адаптивність під мобільні пристрої.

Вимоги до технічних засобів.

Серверна частина: .NET Core.

Клієнтська частина: React.js / Vue.js / Blazor.

СКБД: MSSQL / PostgreSQL.

Вимоги до програмної документації

Програмна документація повинна включати наступні відомості:

1. Технічна документація (опис класів, API).

2. Інструкція користувача (для покупця та постачальника).

1.2. Обґрунтування вибору засобів моделювання

У процесі вибору засобів для моделювання програмного забезпечення на мові UML, було проведено порівняльний аналіз чотирьох популярних інструментів: Draw.io, Lucidchart, StarUML та PlantText.

Draw.io

Переваги:

- безкоштовний;
- працює у браузері, інтегрується з Google Drive;
- простий у використанні.

Недоліки:

- більше інструмент для малювання, а не повноцінне CASE-засіб;
- відсутня валідація UML-правил та генерація коду.

Lucidchart

Переваги:

- зручна колаборація в реальному часі;
- приємний інтерфейс;
- інтеграція з багатьма сервісами (Slack, Jira).

Недоліки:

- обмежений функціонал у безкоштовній версії;
- залежність від інтернет-з'єднання.

StarUML

Переваги:

- спеціалізований інструмент для UML;
- підтримує всі типи діаграм (Class, Use Case, Sequence тощо);
- має розширювану архітектуру через плагіни, підтримує генерацію коду (C#, Java, Python)..

Недоліки:

- немає функцій спільного редагування діаграм;
- обмежений набір стилів, тем та кастомізації діаграм у порівнянні з деякими конкурентами.

PlantUML

Переваги:

- текстовий (Plaintext) підхід до створення UML-діаграм, що дозволяє зберігати їх у вигляді коду;
- підтримує основні типи UML-діаграм (Class, Use Case, Sequence, Activity, Component тощо);
- легко інтегрується з IDE (IntelliJ IDEA, VS Code), системами контролю версій та CI/CD;
- забезпечує швидке створення та редагування діаграм без використання графічного редактора;
- безкоштовний та з відкритим вихідним кодом.

Недоліки:

- відсутній візуальний drag-and-drop редактор;
- обмежені можливості ручного візуального налаштування діаграм;
- не підтримує пряму генерацію програмного коду з діаграм.

У якості засобу моделювання було обрано PlantUML, оскільки він використовує текстовий (plaintext) підхід до побудови UML-діаграм, що забезпечує простоту зберігання, редагування та версіонування моделей. Використання текстового опису дозволяє ефективно інтегрувати діаграми з системами контролю версій, а також з середовищами розробки програмного забезпечення. Крім того, PlantUML є безкоштовним інструментом з відкритим вихідним кодом, підтримує основні типи UML-діаграм та є зручним для документування архітектури програмних систем у рамках навчальних і практичних проєктів.

1.3. Аналіз вимог до програмного продукту

Для забезпечення конкурентоспроможності сервісу на ринку автозапчастин, система повинна задовольняти ряд вимог, сформованих на основі аналізу потреб цільової аудиторії. Каталог вимог і глосарій наведено в додатках А та Б відповідно.

Аналіз вимог користувачів дозволив визначити основні варіанти використання системи.

У системі «Онлайн-сервіс підбору автозапчастин» розрізняють трьох ключових акторів:

1. Адміністратор (Admin) — відповідає за технічну підтримку, модерацію контенту та управління обліковими записами.
2. Постачальник (Supplier) — партнер, який розміщує товари на платформі, керує складом (Inventory) та змінює статус замовлень.
3. Покупець (Buyer) — кінцевий користувач, який шукає деталі, додає свої авто («Гараж»), оформлює замовлення та залишає відгуки.

Короткий опис акторів представлено в табл. 1.1.

Таблиця 1.1

Характеристика акторів системи

Актор	**Опис ролі**	**Основні функції**
Покупець	Зареєстрований користувач, власник авто.	Пошук запчастин, додавання авто в гараж, оформлення замовлення, вибір доставки, відстеження статусу, написання відгуків.
Постачальник	Представник магазину або складу запчастин.	Додавання нових товарів (Part), оновлення цін та наявності, обробка отриманих замовлень.
Адміністратор	Керуючий системою.	Блокування порушників, перегляд статистики продажів, вирішення спорів.

Таблиця 1.2

Виявлення варіантів використання

Основний актор	**Найменування**	**Формулювання**
Покупець	Реєстрація та авторизація	Користувач створює новий обліковий запис або входить у систему, використовуючи логін та пароль.
	Управління профілем авто («Гараж»)	Користувач додає транспортний засіб (VIN, марка, модель, рік) до свого профілю для швидкого підбору сумісних деталей.
	Пошук автозапчастин	Користувач здійснює пошук необхідних деталей за назвою, категорією або сумісністю з конкретним автомобілем.
	Оформлення замовлення	Користувач обирає необхідні запчастини, вказує стратегію доставки (самовивіз, пошта, кур’єр) та підтверджує покупку.
	Написання відгуку	Користувач залишає текстовий коментар та виставляє рейтинг придбаному товару.
Постачальник	Управління інвентарем	Постачальник додає нові запчастини до каталогу, редагує їх опис, сумісність та актуалізує ціни.
	Оновлення залишків на складі	Постачальник змінює інформацію про наявну кількість товару (Update Stock).
	Обробка замовлення	Постачальник приймає замовлення в роботу, комплектує його та змінює статус замовлення (наприклад, на «Відправлено»).
Адміністратор	Управління користувачами	Адміністратор переглядає список зареєстрованих користувачів, має право блокувати або видаляти акаунти порушників.

	Модерація контенту	Адміністратор перевіряє відгуки та описи товарів на відповідність правилам сервісу.
	Генерація звітів	Адміністратор формує аналітичні звіти щодо продажів, активності користувачів та наповненості каталогу.

На рис. 1.1 наведено діаграму варіантів використання (Use Case Diagram).

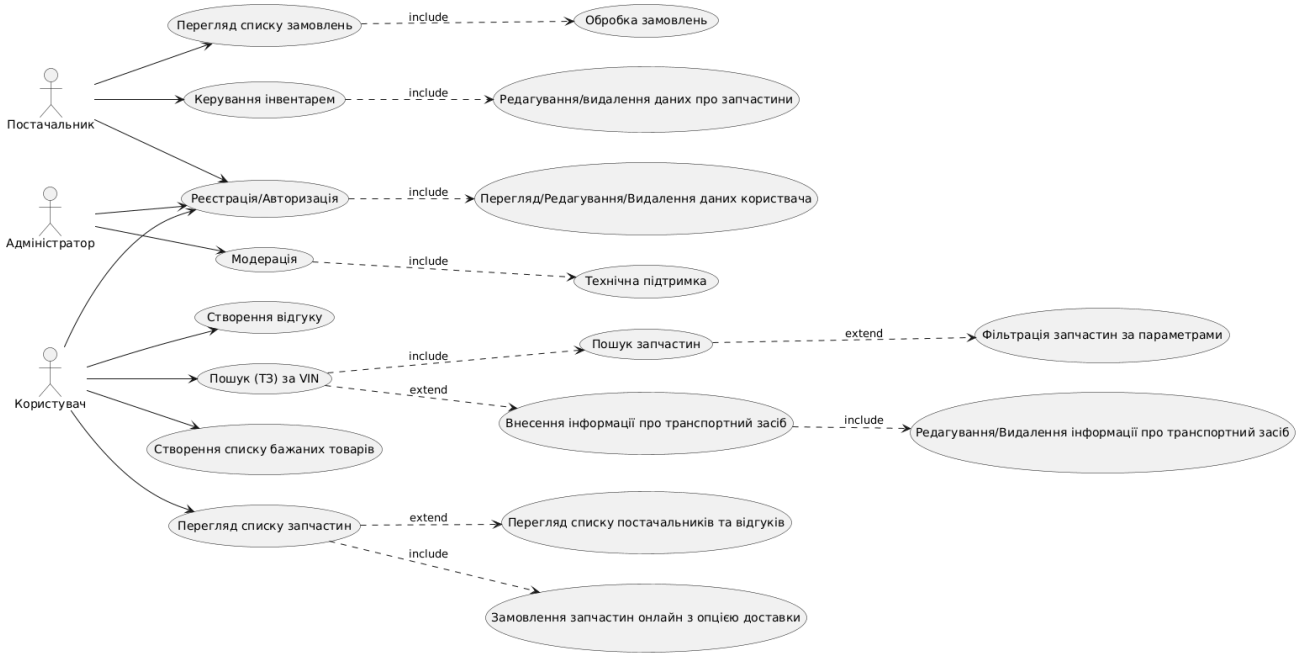

Рис. 1.1 – Діаграма варіантів використання системи

Висновки до першого розділу

Проаналізовано вимоги до програмного продукту, на основі яких сформовано технічне завдання на розробку системи. Для моделювання структури та поведінки системи обрано PlantUML, оскільки даний інструмент використовує текстовий підхід до побудови UML-діаграм, що забезпечує зручність редагування, зберігання та версіонування моделей. На основі розробленого технічного завдання було визначено три основні актори системи, а також сформовано відповідні варіанти використання, які описують їхню взаємодію з програмним продуктом.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ПРОГРАМНОГО КОМПЛЕКСУ НА ЛОГІЧНОМУ РІВНІ

2.1. Алгоритм роботи та стани програмної системи

Діаграми активності, разом з діаграмами варіантів використання та діаграмами станів, вважаються діаграмами поведінки, оскільки вони описують те, що має відбуватися в системі, яка моделюється.

Замовники та розробники мають багато запитань та завдань, в яких потрібно дійти згоди, тому важливо спілкуватися зрозуміло й лаконічно для обох сторін. Діаграми активності допомагають людям з різним рівнем технічної освіти зрозуміти один і той самий процес та прийняти оптимальні рішення в розробці системи.

Діаграма активності (діаграма діяльності) допомагає моделювати послідовності дій у бізнес-процесах. Вона показує, як методи або дії взаємодіють між собою. Ці послідовності можуть бути різними галузями обробки даних або діями, які відбуваються паралельно. Діаграми активності схожі на блок-схеми для алгоритмів. Вони представлені у вигляді графу, де дії – це вузли, а зв’язки між ними – переходи.

Кожний стан на діаграмі активності відповідає виконанню деякої елементарної операції, а перехід в наступний стан виконується тільки після завершення цієї операції. Таким чином, діаграму активності можна вважати приватним випадком діаграми станів.

Функції та застосування діаграм активності:

– демонстрація логіки алгоритмів;

– ілюстрація бізнес-процесів між користувачами та системою;

– оптимізація та удосконалення процесів, роз’яснення складних сценаріїв використання.

Першочерговим етапом роботи з системою є ідентифікація користувача. Діаграма активності на рис. 2.1 відображає комплексний процес реєстрації та авторизації, а також розгалуження логіки залежно від ролі користувача.

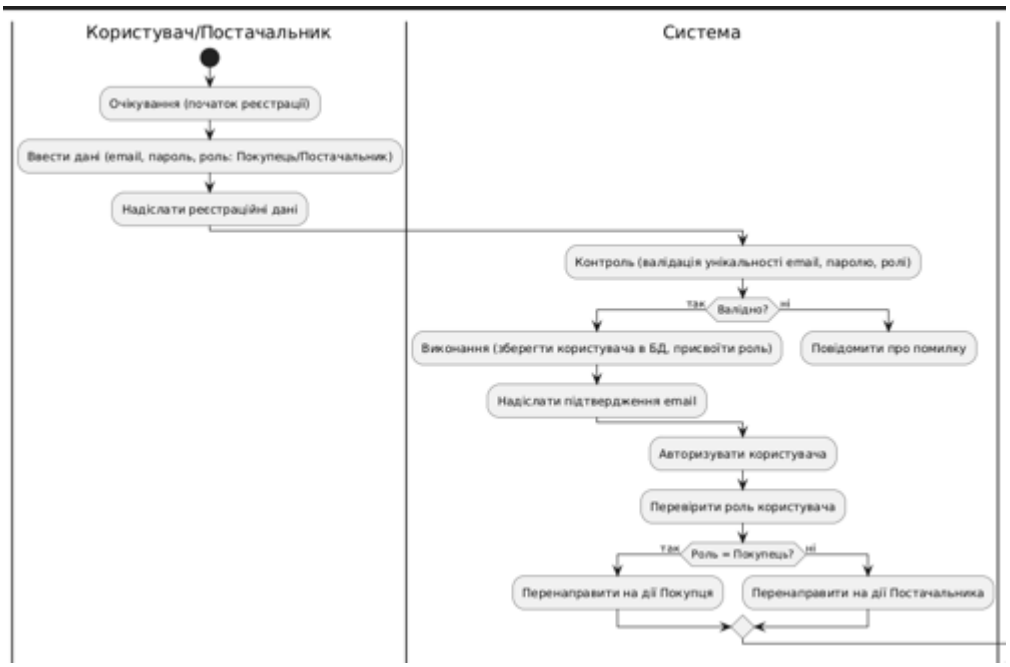

Рис. 2.1 – Діаграма активності для реєстрації та авторизації

Процес починається з очікування початку реєстрації. Користувач або потенційний постачальник вводить необхідні дані (email, пароль) та обирає бажану роль. Система виконує контроль отриманих даних: відбувається валідація унікальності електронної пошти, коректності пароля та вибраної ролі. Якщо дані не є валідними, система повідомляє про помилку, і процес повертається на етап введення. У разі успішної перевірки дані зберігаються в базі даних, користувачу присвоюється роль і надсилається підтвердження на email.

Після авторизації система перевіряє роль користувача, що визначає подальший сценарій:

1. Якщо роль «Покупець»: Користувач перенаправляється на етап внесення інформації про транспортний засіб (марка, модель, рік). Після цього він отримує доступ до пошуку та підбору запчастин, порівняння цін та оформлення замовлення.
2. Якщо роль «Постачальник»: Користувач отримує доступ до функцій управління інвентарем, де він може переглядати каталог, редагувати дані про запчастини та надсилати зміни до системи.

На рис. 2.2 зображено ключовий бізнес-процес системи — обробку замовлення. Дана діаграма демонструє взаємодію між Покупцем, Системою та Постачальником, використовуючи механізм асинхронних сповіщень.

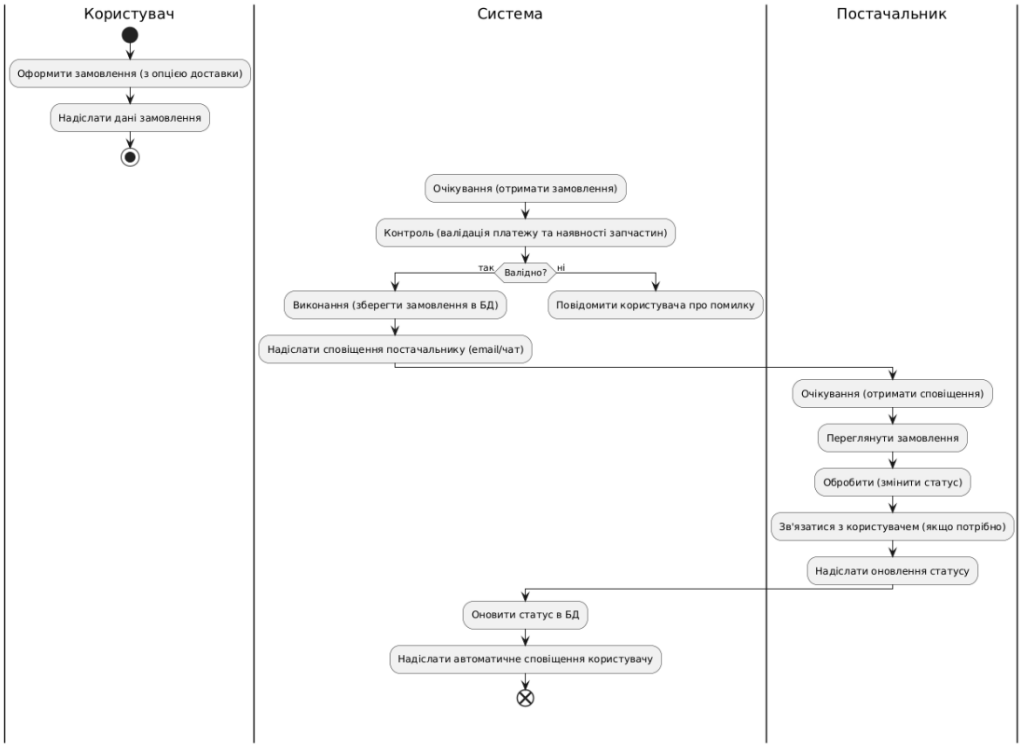

Рис. 2.2 – Діаграма активності для обробки замовлення

Користувач ініціює процес, оформлюючи замовлення з обраною опцією доставки та надсилаючи дані. Система переходить у стан очікування, після чого виконує контроль: валідацію платіжних даних та перевірку фактичної наявності запчастин на складі.

– Якщо перевірка не пройдена, користувач отримує повідомлення про помилку.

– Якщо дані валідні, замовлення зберігається в базі даних, а постачальнику надсилається сповіщення (email або повідомлення у внутрішній чат).

Постачальник, отримавши сповіщення, переглядає деталі замовлення та розпочинає його обробку (змінює статус). За потреби він може зв'язатися з покупцем для уточнення деталей. Після надсилання оновлення статусу,

система фіксує зміни в БД та автоматично сповіщає покупця про те, що його замовлення прийнято в роботу або відправлено.

Третя діаграма активності (рис. 2.3) описує процеси адміністрування та управління контентом. Вона деталізує дії Постачальника щодо наповнення каталогу та функції Адміністратора.

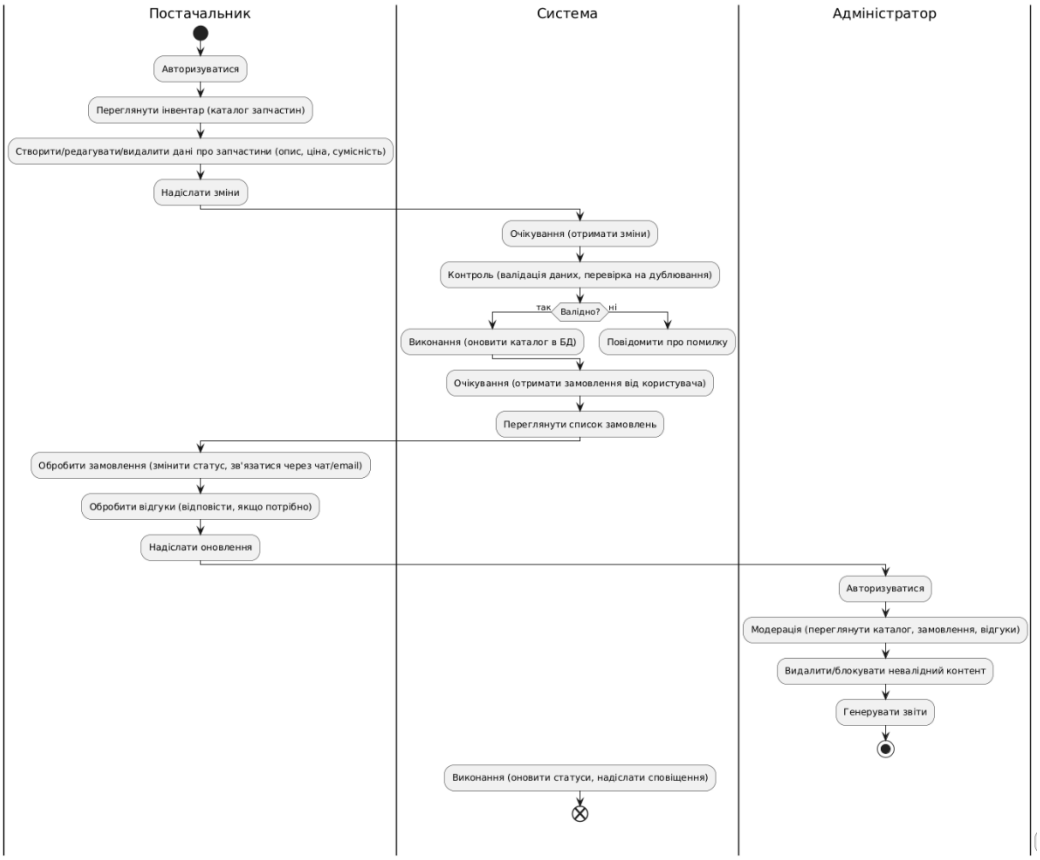

Рис. 2.3 – Діаграма активності для процесу адміністрування та управління контентом

Діяльність Постачальника зосереджена на роботі з інвентарем: авторизація, перегляд каталогу, створення або редагування карток товару (опис, ціна, сумісність). Надіслані зміни проходять автоматичний контроль системою (перевірка на дублювання, коректність форматів даних). Лише валідні зміни оновлюють каталог у базі даних. Також постачальник у цій же доріжці обробляє потік вхідних замовлень та відгуків.

Окрема доріжка виділена для Адміністратора. Його функції включають модерацію контенту: перегляд каталогу, замовлень та відгуків. Адміністратор має повноваження видаляти або блокувати невалідний контент, а також

генерувати аналітичні звіти щодо роботи сервісу. Усі критичні зміни статусів або контенту завершуються оновленням записів у базі даних.

2.2. Об’єктно-орієнтована модель системи

Тепер, коли вже відомі дійові особи й варіанти використання, створено ілюстрацію поведінки системи і послідовність виконання операцій з описом, можна створити попередню об’єктно-орієнтовану модель системи. Почнемо створення діаграми класів з визначення основних сутностей та їх зв’язків у системі.

Основні сутності системи:

- User – абстрактна сутність користувача системи;
- Buyer – клас покупця, який здійснює пошук та замовлення;
- Supplier – клас постачальника, який керує складом;
- Admin – адміністратор системи;
- Vehicle – транспортний засіб користувача (автомобіль);
- Part – автозапчастина (товар);
- Inventory – складський облік (інвентар) постачальника;
- Order – замовлення на придбання запчастин;
- Review – відгук про товар.

В основі моделі лежить механізм успадкування (генералізації). Абстрактний клас User містить спільні атрибути (email, password, role) та методи (register(), login()) для всіх користувачів. Класи Buyer, Supplier та Admin успадковують цей функціонал, розширюючи його специфічними можливостями. Це дозволяє уникнути дублювання коду та спростити масштабування системи.

Між об’єктами «Supplier» і «Inventory», а також «Buyer» і «Vehicle» використовується зв’язок композиція. Це суворий різновид агрегації, коли частина не може існувати без цілого. Особливість цього зв’язку полягає в тому, що компонент може існувати лише як частина контейнера. Тобто, інвентар (Inventory) є невід'ємною частиною постачальника (Supplier), і при видаленні акаунту постачальника його інвентар також знищується.

Аналогічно, автомобілі (Vehicle) в «гаражі» покупця прив’язані до його профілю.

Зв’язок агрегації (або слабкої композиції) простежується між «Inventory» та «Part», а також «Order» і «Part». Агрегація описує відношення «частина-ціле», де дочірні об'єкти можуть існувати самостійно. Наприклад, запчастина (Part) міститься в замовленні (Order), але видалення замовлення не призводить до видалення самої запчастини з бази даних системи.

Також на діаграмі широко використовується **асоціація** — зв’язок, що показує взаємодію між класами без прямого володіння:

- Buyer асоційований з Order (покупець створює замовлення);
- Buyer асоційований з Review (покупець пише відгук);
- Supplier обробляє Order (зв'язок processes);
- Review посилається на Part (відгук залишається про конкретний товар).

Клас Order виступає центральною ланкою, що об'єднує покупця, постачальника та список обраних товарів, містячи методи для розрахунку вартості (calculateTotal()) та оновлення статусу (updateStatus()).

На діаграмі також визначені методи бізнес-логіки. Наприклад, клас Part має метод addCompatibility(), що дозволяє встановлювати зв'язок між деталлю та моделями авто, а клас Inventory містить метод updateStock() для актуалізації кількості товару на складі.

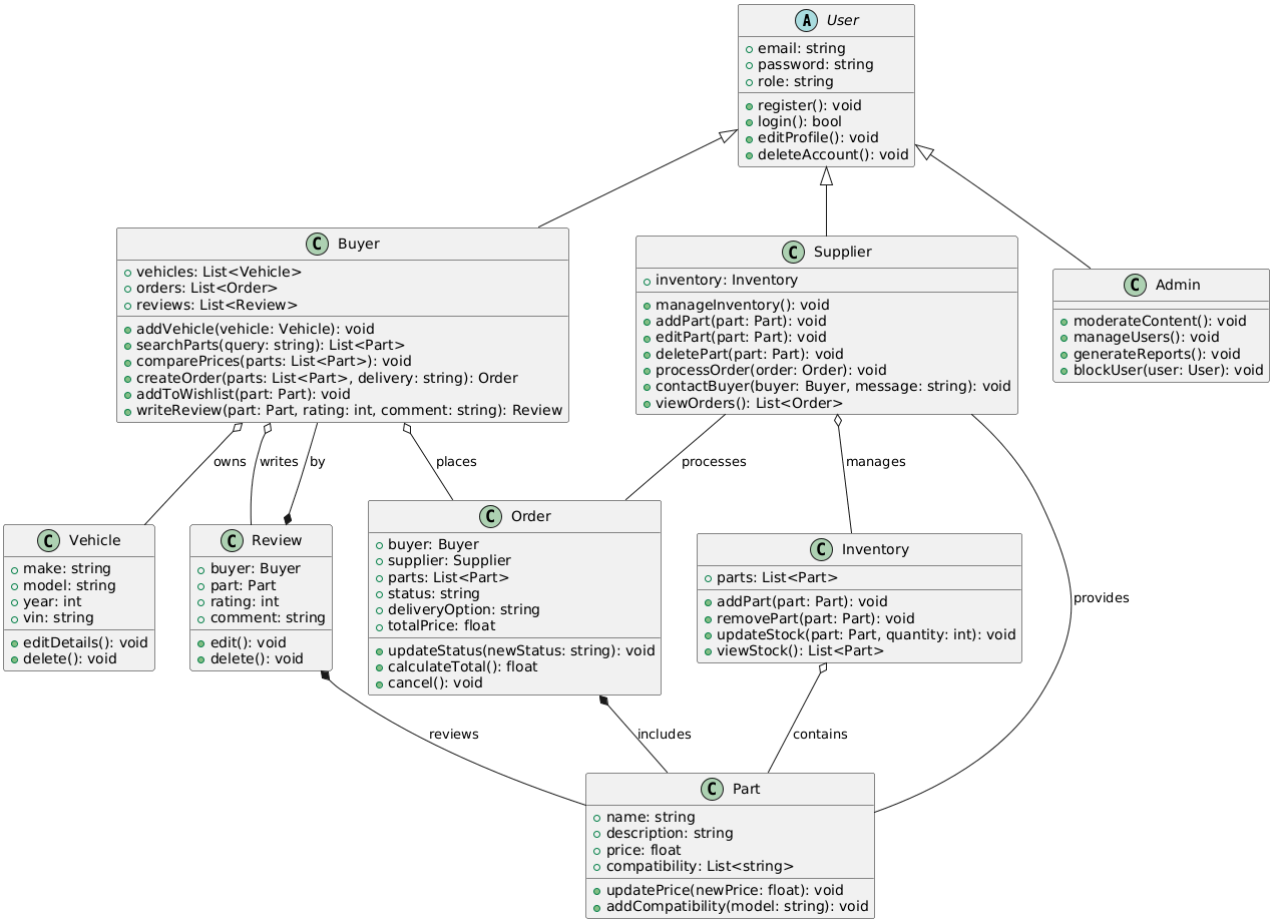

Рис. 2.4 – Діаграма класів системи підбору автозапчастин

Для підвищення гнучкості, розширюваності та зручності супроводу програмного комплексу, до базової діаграми класів було інтегровано три класичні патерни проектування: Factory Method (Фабричний метод), Strategy (Стратегія) та Observer (Спостерігач). Використання цих архітектурних рішень дозволяє зменшити зв’язність компонентів (loose coupling) та забезпечити дотримання принципів SOLID.

Оновлена діаграма класів із застосуванням патернів зображена на рис. 2.5.

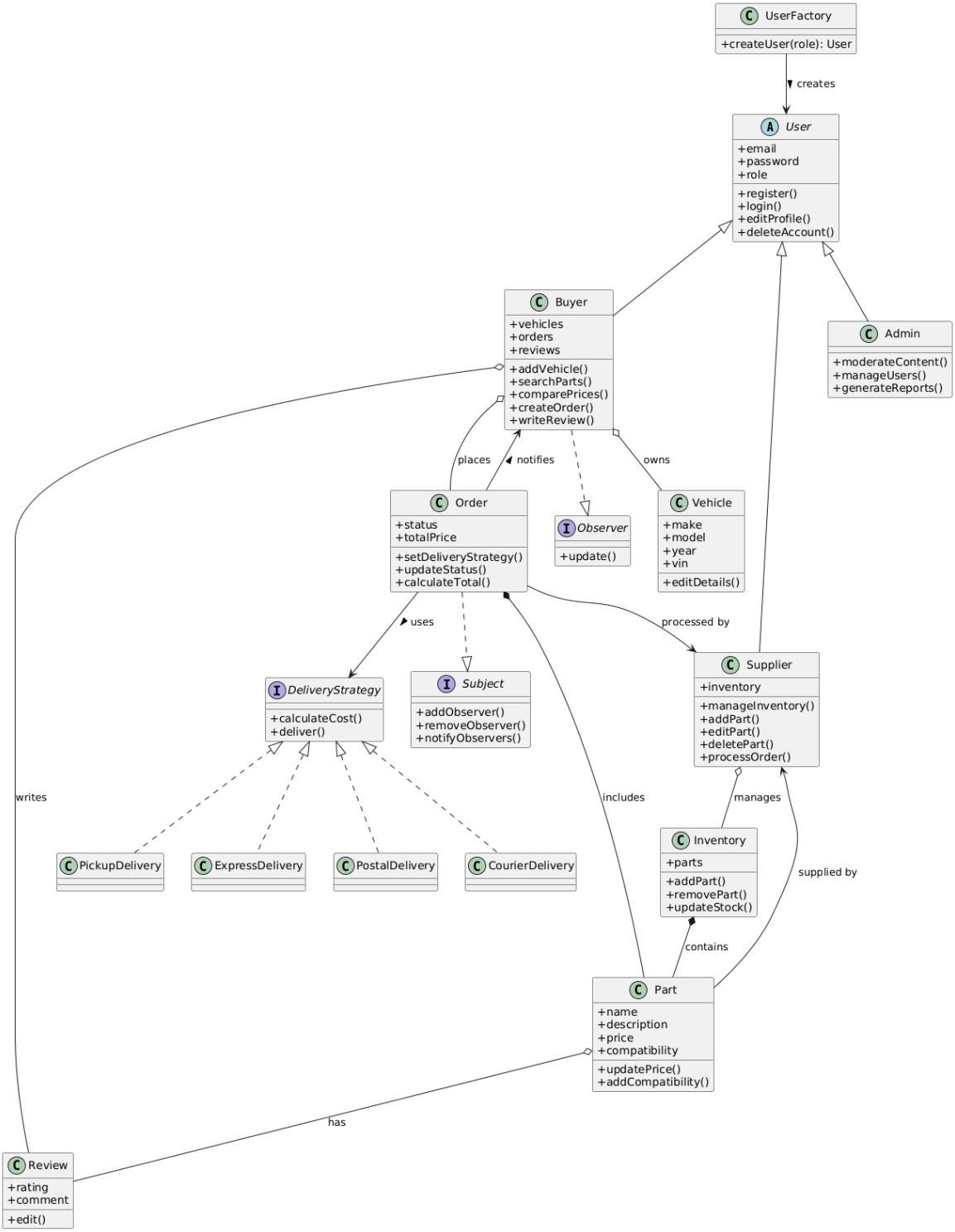

Рис. 2.5 – Діаграма класів системи з використанням патернів проектування

Оскільки система передбачає роботу з різними типами користувачів (Buyer, Supplier, Admin), які мають спільний базовий клас User, але відрізняються набором прав та специфічною поведінкою, було застосовано породжувальний патерн Factory Method.

На діаграмі введено клас UserFactory, який містить статичний метод createUser(role: string): User. Цей метод приймає текстовий ідентифікатор ролі та повертає екземпляр відповідного класу. Це дозволяє інкапсулювати логіку створення об’єктів в одному місці.

Переваги застосування:

- Централізація: Код створення об’єктів не розкиданий по всій системі, а зосереджений у фабриці.
- Розширюваність: Додавання нової ролі (наприклад, «Manager») потребує змін лише у фабриці, не зачіпаючи основний код програми.
- Абстрагування: Клієнтський код працює з абстрактним класом User, не знаючи деталей реалізації конкретних підкласів.

Функціонал оформлення замовлення передбачає варіативність способів доставки (кур’єр, пошта, самовивіз, експрес-доставка), кожен з яких має власний алгоритм розрахунку вартості та термінів. Щоб уникнути використання громіздких умовних операторів (if-else або switch) у класі Order, було використано поведінковий патерн Strategy.

Виділено інтерфейс IDeliveryStrategy, який декларує методи calculateCost() та deliver(). Конкретні реалізації алгоритмів винесено в окремі класи:

- PickupDelivery (Самовивіз);
- ExpressDelivery (Експрес);
- PostalDelivery (Пошта);
- CourierDelivery (Кур’єр).

Клас Order тепер містить посилання на інтерфейс стратегії, а не на конкретну реалізацію, що дозволяє змінювати спосіб доставки динамічно («на льоту») під час виконання програми.

Переваги застосування:

- Інкапсуляція алгоритмів: Кожен алгоритм розрахунку доставки ізольований у власному класі.

• Легка заміна: Можливість додавати нові методи доставки без зміни коду класу Order.

Для реалізації механізму сповіщень про зміну статусу замовлення було обрано поведінковий патерн Observer. Він визначає залежність «один-до-багатьох» між об’єктами таким чином, що при зміні стану одного об’єкта всі залежні від нього інформуються про це автоматично.

У системі клас Order виступає у ролі Суб’єкта (Subject), реалізуючи методи приєднання та сповіщення спостерігачів (addObserver, notifyObservers). Клас Buyer (Покупець) реалізує інтерфейс IObserver з методом update(). Коли постачальник змінює статус замовлення (метод updateStatus у класі Order), автоматично викликається метод сповіщення, який розсилає повідомлення всім підписаним покупцям.

Переваги застосування:

• Реактивність: Покупець миттєво дізнається про зміни стану замовлення без необхідності постійно опитувати сервер.

• Слабка зв’язність: Клас Order не знає деталей реалізації класу Buyer, він лише знає, що той реалізує інтерфейс IObserver.

2.3. Комунікації і послідовність взаємодії об’єктів системи

Діаграми послідовності та взаємодії є важливими інструментами для документування динамічної поведінки системи. Вони демонструють, як об’єкти обмінюються повідомленнями у хронологічному порядку для виконання конкретних бізнес-сценаріїв. Це дозволяє розробникам чітко зрозуміти розподіл відповідальності між компонентами та порядок виклику методів.

На рис. 2.6 представлено діаграму послідовності реєстрації нового користувача. Процес ініціюється на клієнтській частині (Web-UI), де користувач вводить реєстраційні дані (email, пароль, бажана роль). Запит передається до сервісу аутентифікації (AuthService), який викликає метод реєстрації. Система перевіряє наявність користувача в базі даних через UserRepository. Якщо користувача не знайдено (повертається null),

створюється новий запис із хешованим паролем. Після успішного збереження (INSERT INTO users) сервіс ініціює відправку підтвердження на пошту через EmailService та повертає повідомлення про успішну реєстрацію на інтерфейс користувача.

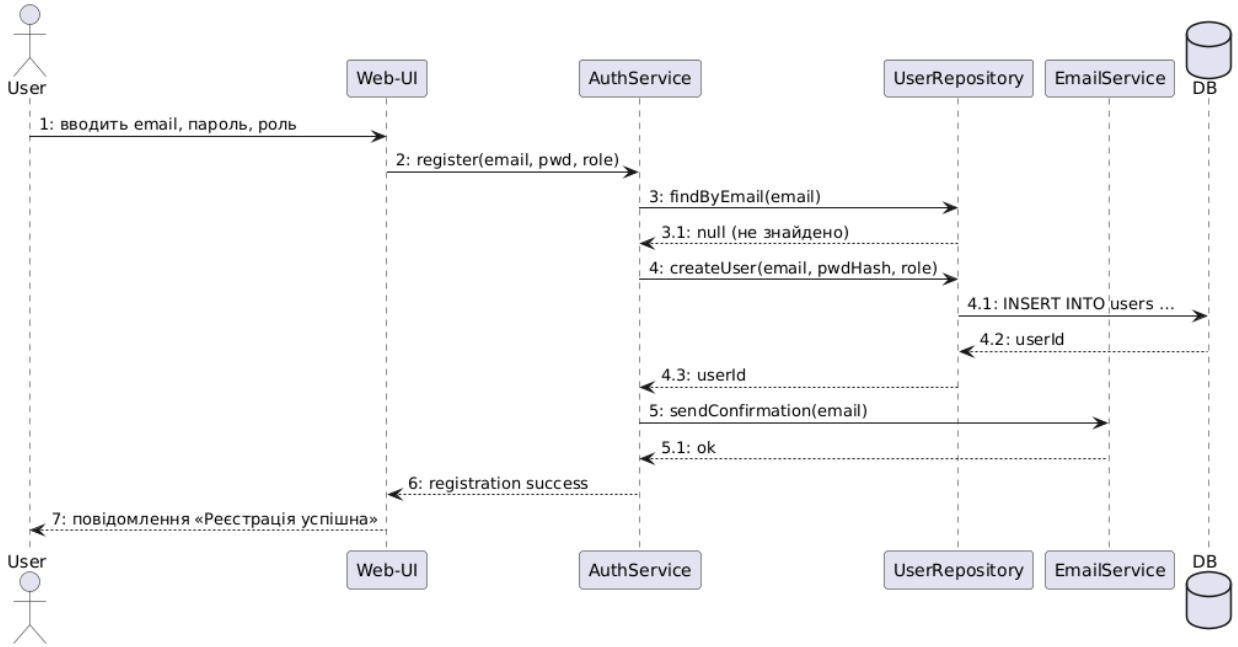

Рис. 2.6 – Діаграма послідовності реєстрації нового користувача

Наступна діаграма (рис. 2.7) деталізує процес авторизації та розподілу прав доступу. Користувач вводить облікові дані, які передаються у метод login(). AuthService робить вибірку з бази даних. Отримавши запис користувача, система зчитує його роль. На основі ролі створюється сесія (createSession) і генерується токен доступу. Ключовим моментом є блок alt (альтернативне виконання):

- Якщо роль ідентифіковано як "buyer", система виконує перенаправлення (redirect) на дашборд покупця (BuyerDashboard).
- Якщо роль "supplier", користувач перенаправляється до панелі управління постачальника (SupplierDashboard). Завершується процес встановленням файлу cookie з токеном сесії на стороні клієнта.

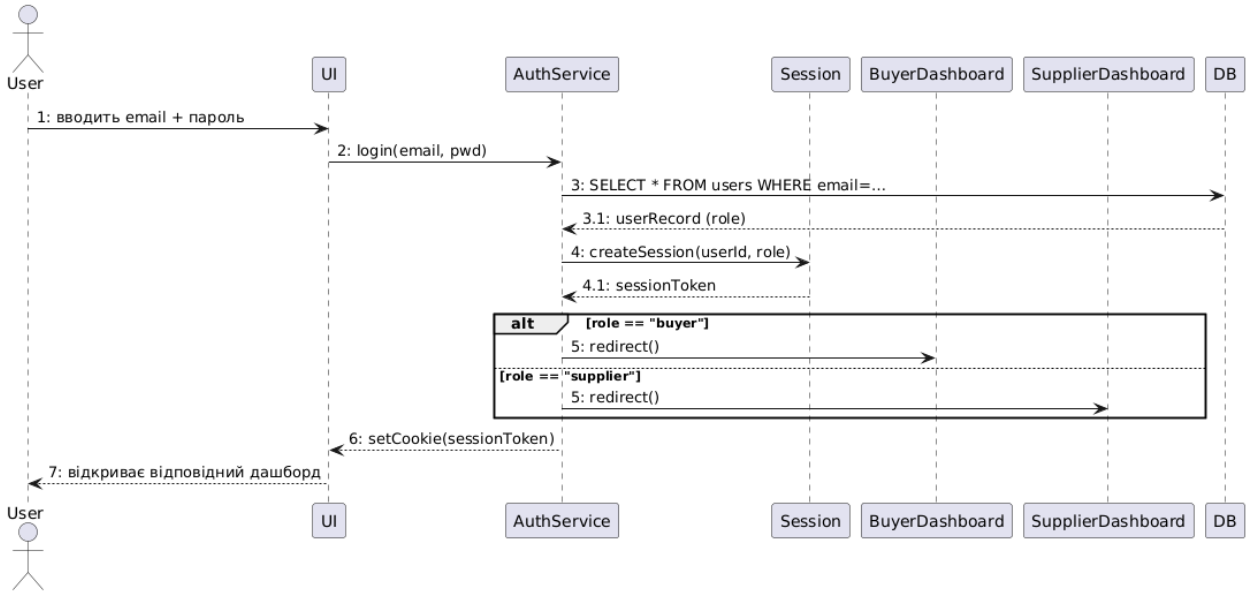

Рис. 2.7 – Діаграма послідовності авторизації користувача та перенаправлення за його роллю

Діаграма послідовності оформлення замовлення (рис. 2.8) демонструє складну взаємодію між об’єктами Buyer, Vehicle, Part, Order та Inventory. Сценарій починається з перевірки сумісності запчастини з автомобілем (getCompatibility). Далі покупець отримує актуальні ціни на товари (getPrice), порівнює їх та ініціює створення замовлення (createOrder) із зазначенням списку товарів та стратегії доставки (наприклад, "NovaPoshta"). Об’єкт Order надсилає сповіщення постачальнику (notifySupplier) і, отримавши підтвердження, звертається до інвентарю для резервування товарів (reserveParts). Тільки після успішного резервування замовлення зберігається в базі даних, а користувачу повертається ID замовлення та статус.

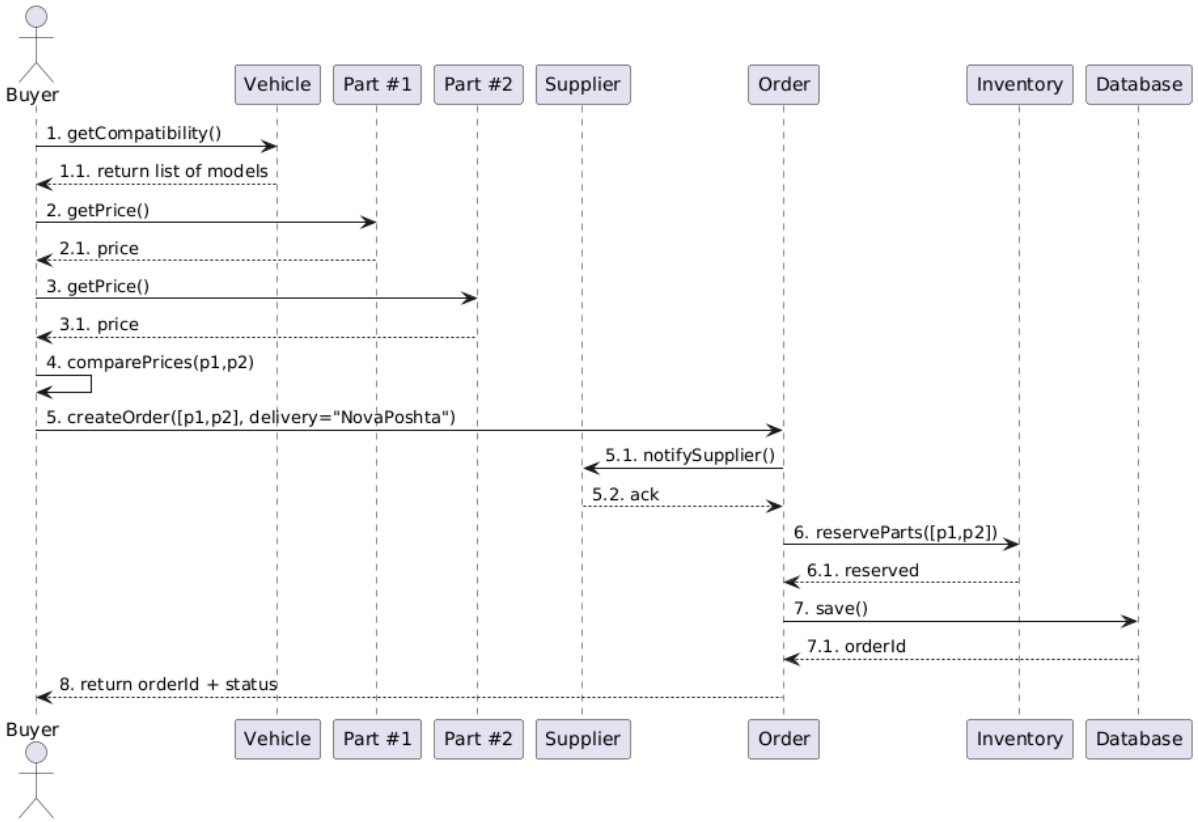

Рис. 2.8 – Діаграма послідовності оформлення замовлення

Фінальна діаграма (рис. 2.9) описує сценарій обробки замовлення постачальником. Постачальник через свій інтерфейс відкриває список замовлень, які мають статус "pending" (очікує). OrderService завантажує ці дані з БД. Коли постачальник обирає замовлення та натискає "Відправити", викликається метод updateStatus зі значенням "shipped". Ця дія запускає ланцюжок подій:

1. Знімається тимчасове резервування товару та остаточно списується кількість зі складу (releaseReservation -> UPDATE stock).
2. Сервіс сповіщень (NotificationService) надсилає покупцеві email про зміну статусу.
3. Оновлений статус відображається в інтерфейсі постачальника.

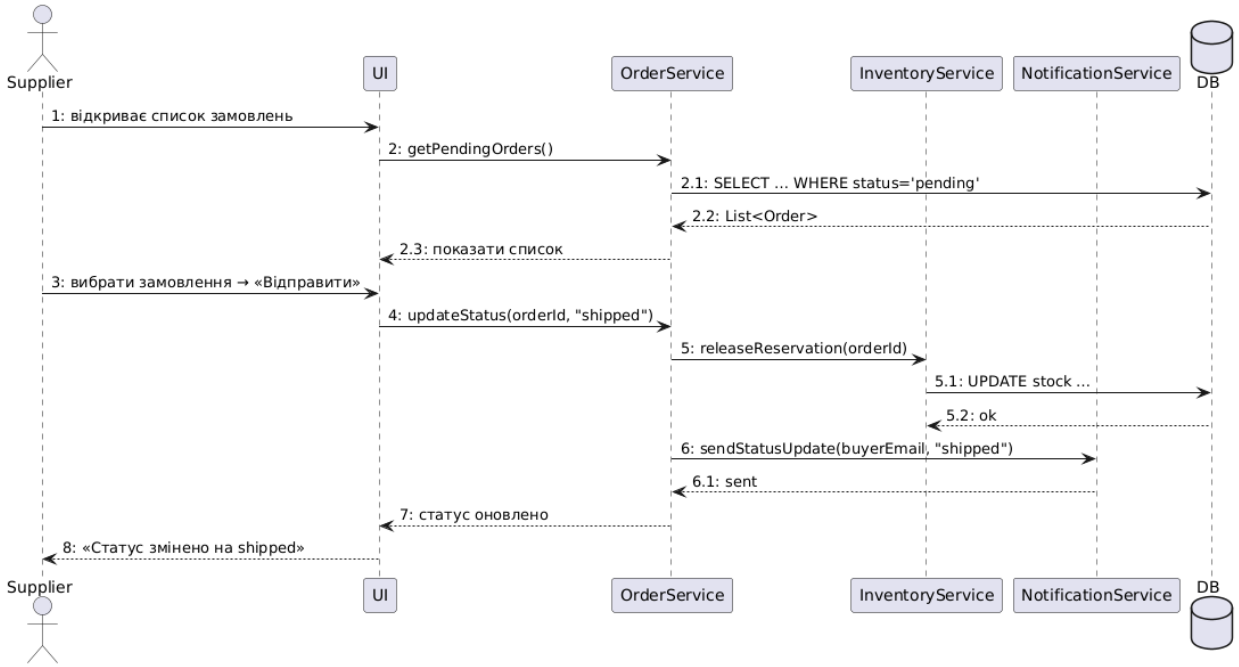

Рис. 2.9 – Діаграма послідовності обробки замовлення постачальником

Висновки до другого розділу

У другому розділі було розроблено детальну модель логічного рівня системи «Онлайн-сервіс підбору автозапчастин».

Побудовано діаграми активності, які формалізують алгоритми реєстрації користувачів, оформлення замовлення та адміністрування контенту. Розроблено об’єктно-орієнтовану модель (діаграму класів), яка відображає структуру сутностей системи (User, Part, Order, Inventory) та типи зв’язків між ними (асоціація, композиція, агрегація). Для забезпечення гнучкості архітектури імплементовано патерни проектування: Factory Method (для створення користувачів), Strategy (для розрахунку доставки) та Observer (для сповіщення про зміну статусу). Створено діаграми послідовності, які описують динамічну взаємодію об’єктів у часі для ключових прецедентів: реєстрації, авторизації з розподілом ролей, створення замовлення покупцем та його обробки постачальником.

Розроблена логічна модель є достатньою для переходу до етапу фізичного моделювання та програмної реалізації.

РОЗДІЛ 3. ФІЗИЧНА МОДЕЛЬ ТА ПРОТОТИП ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Взаємодія компонентів системи

При проектуванні фізичної моделі системи «Онлайн-сервіс підбору автозапчастин» було обрано архітектурний підхід, що базується на декомпозиції системи на незалежні функціональні компоненти. Така структура забезпечує гнучкість, можливість незалежного масштабування окремих модулів та високу відмовостійкість.

Загальна схема взаємодії компонентів зображена на рис. 3.1. Архітектура системи розділена на п'ять логічних рівнів: клієнтська сторона, шлюз API, сервіси бізнес-логіки, зовнішні інтеграції та рівень даних.

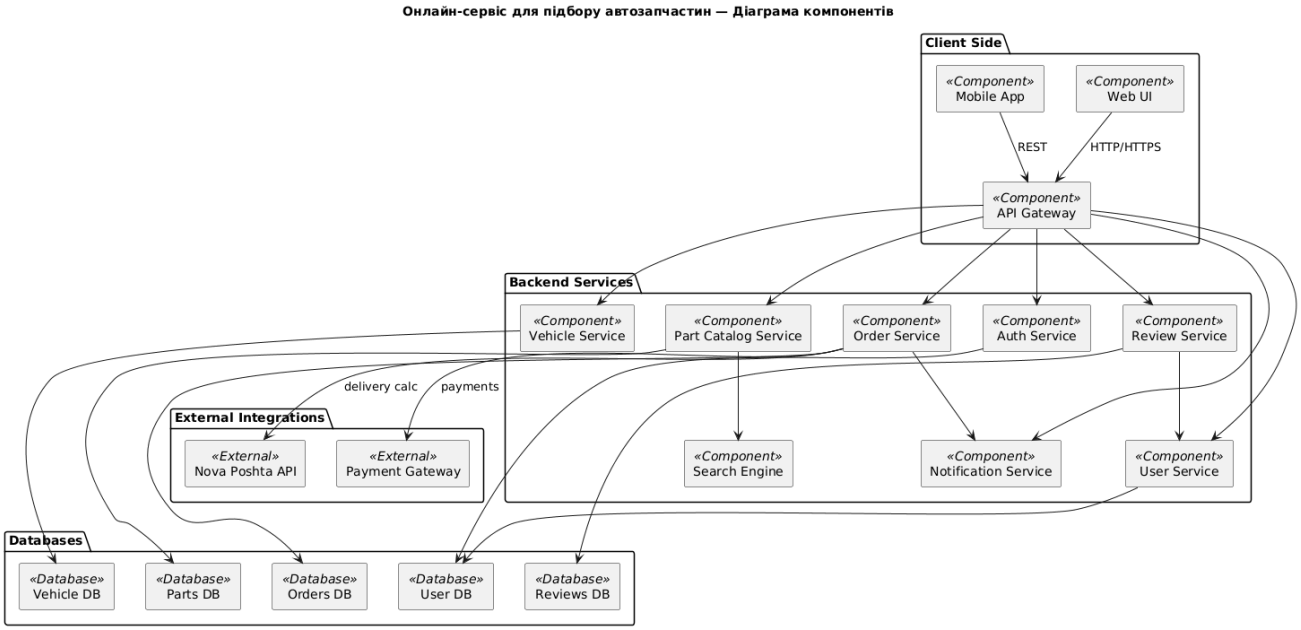

Рис. 3.1 – Діаграма компонентів системи

1. Client Side (Клієнтська сторона) Цей рівень відповідає за взаємодію з кінцевим користувачем. Він представлений двома компонентами:

- Web UI: Веб-інтерфейс, реалізований за допомогою сучасних фронтенд-технологій (наприклад, React або Blazor), який взаємодіє з сервером через протокол HTTP/HTTPS.
- Mobile App: Мобільний додаток, що надає альтернативний доступ до функціонала системи. Обидва клієнти не спілкується з сервісами

напряму, а надсилають REST-запити до єдиної точки входу — API Gateway.

2. API Gateway (Шлюз API) Компонент API Gateway виступає як фасад для всієї системи. Його головна мета — приховати внутрішню структуру бекенду від клієнта. Він виконує маршрутизацію запитів до відповідних мікросервісів, забезпечує балансування навантаження та може виконувати попередню аутентифікацію запитів. Це дозволяє клієнтам звертатися до єдиної адреси, незалежно від того, скільки сервісів працює «під капотом».

3. Backend Services (Сервіси бізнес-логіки) Ядро системи розбите на окремі компоненти, кожен з яких відповідає за свій обмежений контекст (Bounded Context):

- Auth Service: Відповідає за реєстрацію, авторизацію користувачів та видачу токенів доступу.
- Vehicle Service: Керує даними про автомобілі (марки, моделі, роки випуску) та логікою «гаража» користувача.
- Part Catalog Service: Центральний компонент для роботи з номенклатурою запчастин. Він взаємодіє з пошуковим рушієм (Search Engine) для забезпечення швидкого повнотекстового пошуку товарів.
- Order Service: Обробляє життєвий цикл замовлення: від створення кошика до зміни статусу доставки. Цей сервіс є «оркестратором» процесу покупки, оскільки він взаємодіє із зовнішніми платіжними та логістичними системами.
- Review Service: Відповідає за збір та відображення відгуків і рейтингів товарів.
- Notification Service: Забезпечує асинхронну відправку сповіщень (email, push) користувачам про статус їхніх операцій.

4. External Integrations (Зовнішні інтеграції) Для забезпечення повноцінної електронної комерції система інтегрується зі сторонніми API:

- Nova Poshta API: Використовується сервісом замовлень для розрахунку вартості доставки та генерації накладних.

- Payment Gateway: Забезпечує безпечну обробку платежів картками.

5. Databases (Рівень даних) На відміну від монолітних систем, у спроектованій архітектурі використано підхід «Database per Service» (окрема база даних для кожного сервісу). Це забезпечує слабку зв'язність (loose coupling) — зміни в структурі бази даних замовлень не вплинуть на роботу каталогу запчастин.

- Vehicle DB: Зберігає довідники авто.
- Parts DB: Містить інформацію про товари, ціни та залишки.
- Orders DB: Зберігає історію транзакцій та статусів.
- User DB: Зберігає облікові дані та профілі клієнтів.
- Reviews DB: Зберігає коментарі та рейтинги.

Така архітектура дозволяє команді розробників вносити зміни в окремі компоненти без необхідності зупинки всієї системи, що є критично важливим для онлайн-сервісу з високою доступністю.

3.2. Архітектура програмного комплексу та його розгортання

Діаграма розгортання призначена для аналізу апаратної частини системи. За допомогою даної діаграми можна провести аналіз необхідної апаратної конфігурації, на якій працюватимуть окремі процеси системи, і описати їх взаємодію між собою та іншими апаратними пристроями.

Вузол (англ. node) – це фізичний або віртуальний ресурс системи, який має обчислювальні потужності. Наприклад, вузлом може бути сервер, персональний комп’ютер, мобільний пристрій або середовище виконання.

Крім вузлів, на діаграмі розгортання вказуються відносини між ними. У якості відносин виступають фізичні з’єднання та протоколи комунікації. З’єднання є різновидом асоціації і зображуються лініями. Наявність такої лінії вказує на наявність каналу для обміну інформацією між відповідними вузлами.

Система «Онлайн-сервіс підбору автозапчастин» побудована за триланковою архітектурою (Client-Server-Database). Фізична структура системи зображена на рис. 3.2.

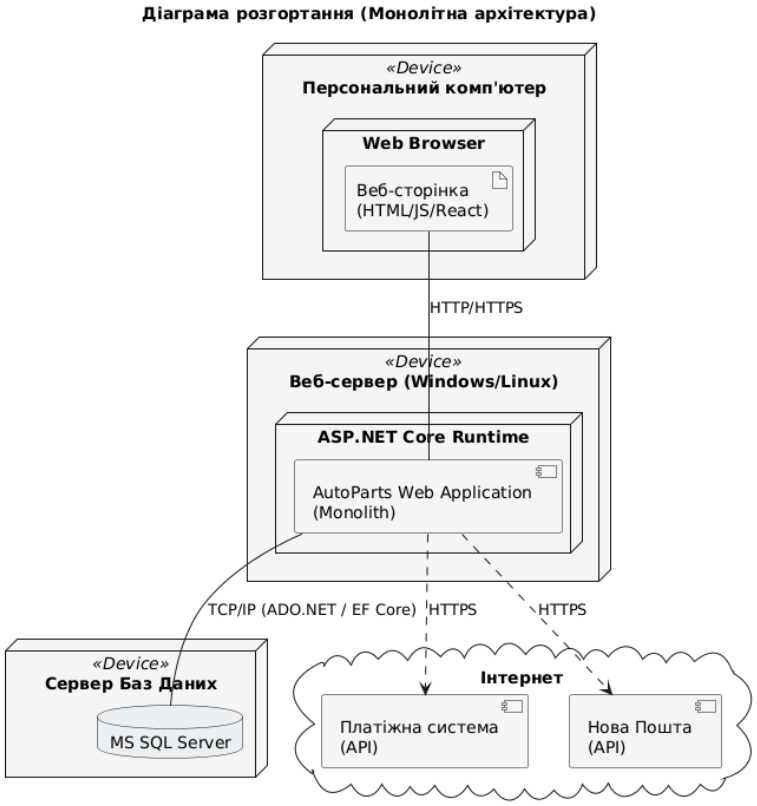

Рис. 3.2 – Діаграма розгортання

Система складається з чотирьох основних типів вузлів (див. рис. 3.2):

1. Персональний комп'ютер (User's Device) – пристрій клієнта (ноутбук, ПК), на якому виконується компонент представлення. Основним середовищем виконання тут є веб-браузер (Web Browser), який завантажує та відображає веб-сторінки інтерфейсу (HTML/CSS/JS/React).

2. Веб-сервер (Web Server) – центральний обчислювальний вузол, що працює під управлінням ОС Windows або Linux. На ньому розгорнуто середовище виконання ASP.NET Core Runtime, яке забезпечує роботу основного артефакту системи – веб-додатку AutoParts Web Application. Оскільки обрано монолітну архітектуру, вся бізнес-логіка (обробка замовлень, каталог, робота з користувачами) зосереджена в цьому єдиному компоненті.

3. Сервер баз даних (Database Server) – виділений сервер, призначений для зберігання та обробки даних. На ньому розгорнуто систему управління базами даних (СКБД) MS SQL Server. Виділення БД на окремий вузол забезпечує високу продуктивність транзакцій та безпеку даних.

4. Зовнішні сервіси (Internet Environment) – хмарні API сторонніх систем, з якими інтегрується додаток. До них належать платіжні шлюзи (для еквайрингу) та API логістичних операторів (наприклад, «Нова Пошта») для розрахунку вартості доставки.

Взаємодія вузлів відбувається наступним чином:

Користувач взаємодіє із системою через веб-браузер. Браузер надсилає HTTP/HTTPS запити до Веб-сервера. Сервер обробляє ці запити, виконуючи відповідні методи контролерів ASP.NET Core.

Для виконання операцій з даними (пошук запчастин, збереження замовлення), веб-додаток звертається до Сервера баз даних через протокол TCP/IP. Взаємодія реалізована за допомогою технології ORM (Entity Framework Core) або ADO.NET.

Для специфічних операцій, таких як оплата або формування накладної, веб-сервер ініціює захищені HTTPS-запити до вузлів Зовнішніх сервісів у мережі Інтернет.

Така схема розгортання є оптимальною для початкового етапу запуску сервісу, оскільки вона спрощує адміністрування, знижує витрати на інфраструктуру та забезпечує достатній рівень продуктивності.

3.3. Генерування програмного коду для прототипу програмного комплексу

На основі розробленої у другому розділі об'єктно-орієнтованої моделі (діаграми класів) та діаграми компонентів, було виконано етап фізичної реалізації прототипу системи «Онлайн-сервіс підбору автозапчастин».

В якості інструментального засобу розробки обрано інтегроване середовище Microsoft Visual Studio 2022, яке забезпечує повну підтримку мови програмування C# та платформи .NET Core. Цей вибір обумовлений суворою типізацією мови, що дозволяє уникнути багатьох помилок на етапі компіляції, та нативною підтримкою принципів об'єктно-орієнтованого програмування.

Для реалізації прототипу було застосовано підхід розділення відповідальності (Separation of Concerns). Структура проекту організована

таким чином, щоб відділити бізнес-логіку від моделей даних та інтерфейсів взаємодії.

На рисунку 3.3 зображено фінальну структуру файлів проекту у вікні Solution Explorer.

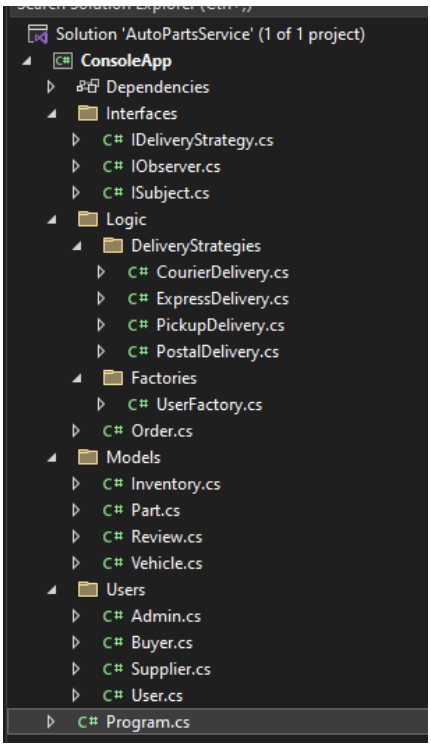

Рис. 3.3 – Структура проекту

Як видно з рисунка, архітектура додатку поділена на логічні каталоги (Namespaces):

1. Interfaces/ — містить контракти, що забезпечують гнучкість системи та реалізацію патернів проектування:
    - IDeliveryStrategy.cs — інтерфейс для стратегій доставки;
    - IObserver.cs та ISubject.cs — інтерфейси для реалізації патерну Спостерігач.
2. Models/ — містить POCO-класи (Plain Old CLR Objects), що описують сутності предметної області:
    - Part.cs (Запчастина), Vehicle.cs (Автомобіль), Inventory.cs (Склад), Review.cs (Відгук).
3. Users/ — реалізує ієрархію користувачів, спроектовану на діаграмі класів:

- Базовий клас User.cs та його спадкоємці Admin.cs, Buyer.cs, Supplier.cs.

4. Logic/ — містить основну бізнес-логіку додатку:

- Order.cs — клас замовлення, який керує статусами та розрахунком ціни;
- Factories/UserFactory.cs — реалізація патерну Factory Method для створення користувачів;
- DeliveryStrategies/ — конкретні алгоритми доставки (ExpressDelivery, PostalDelivery тощо).

Нижче наведено фрагмент програмного коду класу UserFactory, який демонструє генерацію об'єктів користувачів залежно від обраної ролі:

```
namespace AutoPartsService.Logic.Factories
{
    public class UserFactory
    {
        public static User CreateUser(string role)
        {
            switch (role.ToLower())
            {
                case "buyer": return new Buyer();
                case "admin": return new Admin();
                case "supplier": return new Supplier();
                default: throw new ArgumentException("Невідома роль користувача");
            }
        }
    }
}
```

Також наведено реалізацію класу Order, який використовує патерн Стратегія для розрахунку вартості доставки та патерн Спостерігач для сповіщення покупця:

```
namespace AutoPartsService.Logic
{
    public class Order : ISubject
    {
        public string Status { get; private set; }
        public double TotalPrice { get; private set; }

        private IDeliveryStrategy _deliveryStrategy;
        private List<IObserver> _observers = new List<IObserver>();

        public void SetDeliveryStrategy(IDeliveryStrategy strategy)
        {
            _deliveryStrategy = strategy;
            Console.WriteLine($"[Order]        Обрано        стратегію        доставки:
{strategy.GetType().Name}");
        }

        public void CalculateTotal(double partsPrice)
        {
            if (_deliveryStrategy == null)
            {
```

```
                Console.WriteLine("[Order] Помилка: Спосіб доставки не обрано!");
                return;
            }
            double deliveryCost = _deliveryStrategy.CalculateCost(partsPrice);
            TotalPrice = partsPrice + deliveryCost;
            Console.WriteLine($"[Order] Розрахунок: Товар ({partsPrice}) + Доставка
({deliveryCost}) = {TotalPrice}");
        }

        public void UpdateStatus(string newStatus)
        {
            Status = newStatus;
            Console.WriteLine($"[Order] Статус змінено на: {Status}");
            NotifyObservers(); // Сповіщаємо покупця
        }

        // --- ISubject Implementation ---
        public void AddObserver(IObserver observer) => _observers.Add(observer);
        public void RemoveObserver(IObserver observer) => _observers.Remove(observer);

        public void NotifyObservers()
        {
            foreach (var observer in _observers)
            {
                observer.Update($"Ваше замовлення перейшло у статус: {Status}");
            }
        }
    }
}
```

В результаті виконання програмного коду було отримано робочий прототип консольного додатку, який успішно емулює процеси реєстрації, наповнення складу, оформлення замовлення та зміни його статусу.

3.4. Огляд інтерфейсу прототипу

У рамках розробки фізичної моделі системи було створено два рівні представлення продукту: програмна реалізація бізнес-логіки (консольний додаток) та візуальний концепт веб-інтерфейсу (UI/UX Design).

Для перевірки коректності роботи спроектованих алгоритмів та патернів проектування було здійснено запуск розробленого консольного додатку. Результат виконання сценарію «Повний цикл замовлення» наведено на рис. 3.4.

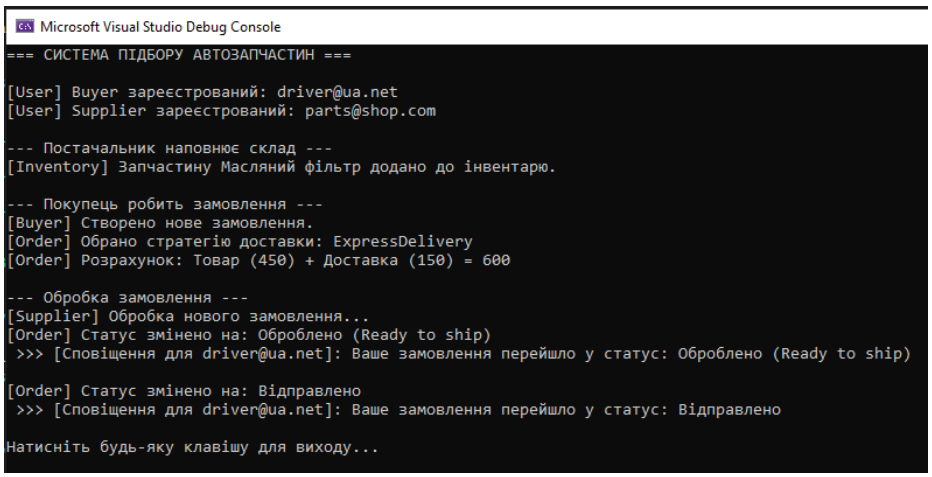

Рис. 3.4 – Лог виконання програми (демонстрація сценарію замовлення)

Як видно з лістингу консолі, система успішно відпрацювала ключові етапи:

1. Реєстрація: Спрацював патерн *Factory Method*, створивши об'єкти Buyer та Supplier з різними ролями.
2. Наповнення складу: Постачальник додав товар («Масляний фільтр») до інвентарю.
3. Розрахунок: Під час створення замовлення спрацював патерн *Strategy* («ExpressDelivery»), який додав до вартості товару (450 грн) фіксовану вартість доставки (150 грн), сформувавши загальну суму 600 грн.
4. Сповіщення: При зміні статусу замовлення постачальником (на «Оброблено» та «Відправлено») спрацював патерн *Observer*. Система автоматично згенерувала повідомлення для клієнта driver@ua.net, що підтверджує наявність реактивного зв'язку між компонентами.

Для забезпечення високого рівня юзабіліті (Usability) та зручності взаємодії користувача з системою, було розроблено дизайн-макет головної сторінки сервісу. Проектування інтерфейсу виконувалося у графічному редакторі Figma.

Обрана колірна гама (темний фон з яскравими червоними акцентами) підкреслює автомобільну тематику та фокусує увагу користувача на ключових елементах (товар, кнопка «Купити»). Структура головної сторінки зображена на рис. 3.5.

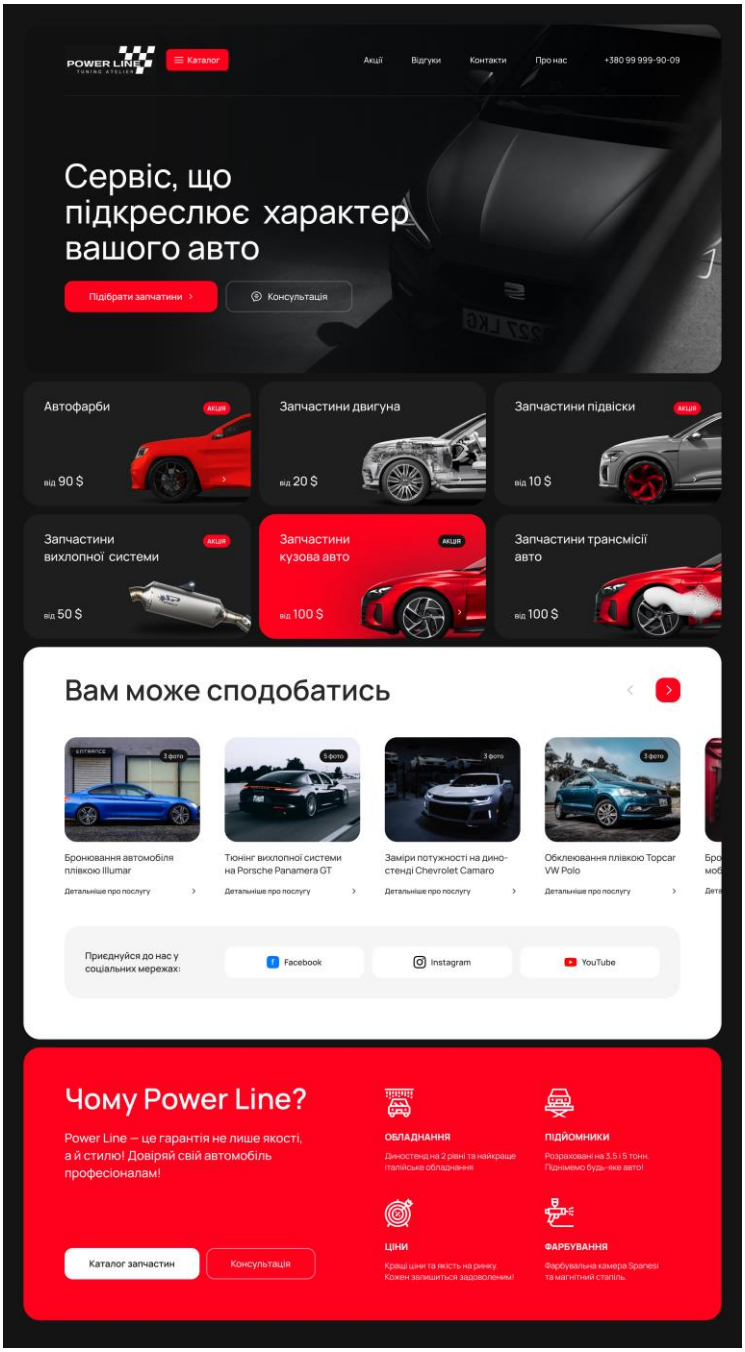

Рис. 3.5 – Макет головної сторінки веб-сервісу

Інтерфейс сторінки поділено на логічні функціональні блоки:

1. Хедер (Header): Верхня панель навігації містить логотип сервісу, контактні дані (+380...) для швидкого зв'язку та кнопку «Каталог» (Call-to-Action), яка дозволяє миттєво перейти до пошуку запчастин.
2. Головний екран (Hero Section): Містить якісне зображення автомобіля, мотиваційний заголовок та дві цільові кнопки: «Підібрати запчастини» (основна дія) та «Консультація» (допоміжна дія). Це

дозволяє направити користувача за потрібним сценарієм одразу після входу на сайт.

3. Блок категорій (Categories Grid): Для спрощення навігації каталог розбито на візуальні картки.
4. Блок рекомендацій: Секція «Вам може сподобатись» реалізує механізм крос-продажів, пропонуючи супутні послуги (бронювання плівкою, тюнінг, заміри потужності).
5. Інформаційний блок (Advantages): Секція «Чому ми?» (на червоному фоні) розкриває конкурентні переваги сервісу: професійне обладнання, підйомники, фарбування.
6. Футер (Footer): Нижня частина сайту містить посилання на соціальні мережі (Facebook, Instagram, YouTube) для комунікації з клієнтами та дублює кнопки доступу до каталогу.

Розроблений інтерфейс є адаптивним та інтуїтивно зрозумілим, що відповідає вимогам до сучасних веб-додатків у сфері e-commerce.

Висновки до третього розділу

У третьому розділі було розроблено фізичну модель програмного комплексу. Визначено схему взаємодії компонентів системи, яка базується на триланковій архітектурі, та побудовано діаграму розгортання, що описує апаратну частину рішення.

Виконано програмну реалізацію прототипу системи мовою C# в середовищі Visual Studio. Структура проекту повністю відповідає спроектованій у другому розділі об'єктно-орієнтованій моделі та забезпечує виконання основних функціональних вимог до онлайн-сервісу підбору автозапчастин.

Розроблено концепт графічного інтерфейсу користувача (UI/UX) у середовищі Figma, який демонструє візуальний стиль, зручну структуру сторінок та інтуїтивно зрозумілу навігацію для майбутніх клієнтів сервісу.

ВИСНОВКИ

У ході виконання курсової роботи було спроєктовано та розроблено прототип програмного комплексу «Онлайн-сервіс підбору автозапчастин».

На етапі аналізу вимог було досліджено предметну область електронної комерції в автоіндустрії. Сформовано технічне завдання, яке визначає ключові функціональні вимоги до системи. Виявлено трьох основних акторів (Покупець, Постачальник, Адміністратор) та побудовано діаграму варіантів використання, що формалізує їхню взаємодію із системою.

Під час логічного моделювання було розроблено детальну структуру та поведінку системи. Діаграми активності дозволили алгоритмізувати процеси реєстрації, оформлення замовлення та управління інвентарем. Об'єктно-орієнтована модель (діаграма класів) визначила основні сутності та зв'язки між ними. Для забезпечення гнучкості архітектури було успішно імплементовано патерни проєктування. Діаграми послідовності деталізували динаміку обміну даними між об'єктами у часі.

На етапі фізичного моделювання спроєктовано компонентну структуру та схему розгортання системи, що базується на триланковій архітектурі (Клієнт – Веб-сервер – Сервер БД). Визначено необхідні апаратні вузли та зовнішні інтеграції (платіжні шлюзи, служби доставки).

Практична частина роботи полягала у створенні прототипу програмного комплексу мовою C# у середовищі Visual Studio. Реалізована структура проєкту повністю відповідає спроєктованій моделі. Додатково розроблено візуальний концепт інтерфейсу (UI/UX) у середовищі Figma, що демонструє зручність та естетичність майбутнього продукту.

Отже, поставлена мета курсової роботи досягнута: пройдено повний цикл проєктування програмного забезпечення — від аналізу вимог до генерації програмного коду та створення робочого прототипу. Розроблена система готова до подальшого масштабування та впровадження.

ДЖЕРЕЛА ІНФОРМАЦІЇ

1. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th Edition. — Addison-Wesley Professional, 2021. — 464 p.
2. Dennis A., Wixom B. H., Tegarden D. Systems Analysis and Design: An Object-Oriented Approach with UML. 6th Edition. — Wiley, 2020. — 544 p.
3. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Prentice Hall, 2018. — 432 p.
4. Object Management Group. UML 2.5.1 Specification [Electronic resource]. — 2018. — Mode of access: https://www.omg.org/spec/UML/
5. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. — O'Reilly Media, 2020. — 432 p.
6. Sommerville I. Engineering Software Products: An Introduction to Modern Software Engineering. — Pearson, 2020. — 360 p.
7. StarUML Documentation. User Guide [Electronic resource]. — 2024. — Mode of access: https://docs.staruml.io/
8. Wiegers K., Beatty J. Software Requirements. 3rd Edition. — Microsoft Press, 2019. — 672 p.
9. Близнюк М. М. Проектування програмних систем: навчальний посібник. — Чернівці: Чернівецький нац. ун-т, 2019. — 152 с.
10. Гамзаєв Р. О. Інженерія програмного забезпечення: навчальний посібник. — Івано-Франківськ: Фоліант, 2019. — 164 с.
11. Грицюк Ю. І. Аналіз вимог до програмного забезпечення: навчальний посібник. — Львів: Видавництво Львівської політехніки, 2018. — 240 с.
12. Катющева О. В., Дьоменко В. Ф. Проектування інформаційних систем: навчальний посібник. — Київ: КПІ ім. Ігоря Сікорського, 2020. — 176 с.
13. Литвин В. В. Технології розробки програмного забезпечення: підручник. — Львів: «Новий Світ-2000», 2019. — 464 с.

14. Мельник К. В. Моделювання програмного забезпечення UML: практикум. — Київ: КПІ ім. Ігоря Сікорського, 2020. — 95 с.

15. Швець А. Занурення у патерни проектування. — Refactoring.Guru, 2019. — 406 с. (Електронне видання).

ДОДАТКИ

Додаток А

Каталог вимог

Бізнес вимоги

1. Основні цілі: система розробляється з метою автоматизації процесу пошуку, підбору та придбання автомобільних запчастин, а також для забезпечення взаємодії між покупцями та постачальниками на єдиній платформі.

2. Представлення проєкту: проєкт буде реалізовано у вигляді веб-сервісу (маркетплейсу), що міститиме електронний каталог товарів, систему фільтрації за параметрами авто та модуль обробки замовлень.

Вимоги користувачів

1. Можливість реєстрації та авторизації користувачів із розподілом ролей (Покупець, Постачальник, Адміністратор).

2. Можливість швидкого пошуку запчастин за різними критеріями: назва, артикул, VIN-код автомобіля або категорія.

3. Можливість ведення особистого кабінету «Гараж» для покупців, де зберігаються дані про їхні автомобілі для автоматичної фільтрації сумісних деталей.

4. Можливість оформлення замовлення з вибором товарів від різних постачальників та визначенням способу доставки (кур’єр, пошта, самовивіз).

5. Можливість для постачальників керувати власним інвентарем (додавати нові товари, змінювати ціни та залишки на складі).

6. Можливість відстеження статусу виконання замовлення в режимі реального часу.

7. Можливість залишати відгуки та оцінки про товари після здійснення покупки.

Функціональні вимоги

1. Управління доступом: Система повинна забезпечувати розмежування прав доступу до функціоналу залежно від ролі користувача (наприклад, тільки постачальник може змінювати статус замовлення на «Відправлено»).

2. Пошукова система: Система повинна забезпечувати релевантний пошук товарів та перевірку їх сумісності з транспортним засобом користувача.

3. Обробка замовлень: Система повинна автоматично розраховувати загальну вартість замовлення з урахуванням ціни товарів та обраної стратегії доставки.

4. Сповіщення: Система повинна реалізовувати механізм автоматичного відправлення повідомлень (email або push) користувачам при зміні статусу їх замовлення.

Нефункціональні вимоги

1. Сприйняття:
   - час відгуку системи при пошуковому запиті – не більше 2 секунд;
   - інтерфейс має бути адаптивним (коректно відображатися на ПК, планшетах та смартфонах);
   - дизайн має відповідати сучасним стандартам UX для e-commerce
2. Надійність:
   - доступність – система повинна працювати у режимі 24/7;
   - цілісність даних – система повинна забезпечувати транзакційність операцій (гроші не списуються, якщо товару немає на складі).
3. Безпека:
   - паролі користувачів повинні зберігатися у захешованому вигляді;
   - захист від SQL-ін’єкцій та XSS-атак при введенні даних у форми.
4. Масштабованість:
   - архітектура системи повинна дозволяти легке додавання нових категорій товарів та підключення нових служб доставки без зупинки сервісу.

Додаток Б

Глосарій

VIN-код (Vehicle Identification Number) – унікальний ідентифікаційний номер транспортного засобу, що складається з 17 символів. У системі використовується для точної ідентифікації автомобіля та перевірки сумісності запчастин (уникнення помилок при підборі).

Гараж (Virtual Garage) – розділ особистого кабінету користувача (Покупця), де зберігається список його транспортних засобів. Додавання авто в «Гараж» дозволяє системі автоматично фільтрувати каталог, показуючи лише ті деталі, що підходять до конкретної машини.

Замовлення (Order) – сутність системи, що представляє намір покупця придбати певний перелік товарів. Замовлення має унікальний номер, список позицій, загальну вартість, обраний спосіб доставки та поточний статус виконання.

Інвентар (Inventory) – віртуальний склад постачальника в системі. Це сукупність записів про товари, які постачальник пропонує до продажу, включаючи інформацію про їх кількість (залишок), ціну та опис.

Кошик – тимчасове сховище обраних товарів перед оформленням замовлення. Дозволяє користувачеві накопичувати товари, змінювати їх кількість або видаляти перед фінальною оплатою.

Покупець (Buyer) – зареєстрований користувач системи, який використовує сервіс для пошуку, підбору та придбання автозапчастин для особистого використання.

Постачальник (Supplier) – бізнес-користувач системи (магазин, склад, дистриб’ютор), який розміщує свої товари в каталозі, керує цінами та залишками, а також відповідає за фізичне комплектування та відправку замовлень.

Статус замовлення – поточний стан життєвого циклу замовлення. Основні статуси в системі:

- *Pending (Очікує)* – замовлення створено, але не оброблено;

- *Processing (Обробляється)* – постачальник підтвердив наявність і комплектує товар;
- *Shipped (Відправлено)* – товар передано службі доставки;
- *Delivered (Доставлено)* – покупець отримав товар.

Стратегія доставки – алгоритм розрахунку вартості та термінів доставки. У системі реалізовано різні стратегії (Самовивіз, Кур'єр, Поштова служба), які впливають на кінцеву ціну замовлення.

Сумісність (Compatibility) – технічна характеристика запчастини, яка визначає перелік марок, моделей та років випуску автомобілів, на які може бути встановлена дана деталь.