МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ АВТОЗАПЧАСТИН ЗАСОБАМИ LARAVEL ТА VUE.JS»**

Виконав: студент 5 курсу, групи 6.1269-з
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми Інформаційні системи та технології
(назва освітньої програми)
С.С. Дуран
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Дурану Серхіо Сесаровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет-магазину автозапчастин засобами Laravel та Vue.js

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2181-с

2. Строк подання студентом роботи 17.05.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка інтернет-магазину автозапчастин.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.01.2024	
2.	Збір вихідних даних.	12.02.2024	
3.	Обробка методичних та теоретичних	04.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	01.04.2024	
5.	Розробка третього розділу.	03.05.2024	
6.	Оформлення та нормоконтроль	10.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	31.05.2024	

Студент _____ С.С. Дуран
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет-магазину автозапчастин засобами Laravel та Vue.js»: 47 с., 24 рис., 13 джерел.

АВТОМАТИЗАЦІЯ, БАЗА ДАННИХ, ВЕБСЕРВІС, CSS, HTML, JETBRAINS, LIQUIBASE, REST, TYPE SCRIPT.

Об’єкт дослідження – фреймворк на java spring, angular, javascript, css, html.

Мета роботи: розробити інтернет-магазин автозапчастин (двигунів, та коробок передач).

Метод дослідження – ця робота присвячена розробці інтернет-магазину автозапчастин з використанням сучасних технологій веброзробки. Мета роботи – створення зручного та функціонального вебдодатку, що дозволяє користувачам легко знаходити та купувати автозапчастини. Проєкт включає розробку як серверної частини за допомогою Spring Boot, так і клієнтського інтерфейсу з використанням Angular, а також управління базою даних PostgreSQL.

SUMMARY

Bachelor’s qualifying paper «Development of an Online Auto Parts Store Using Laravel and Vue.is»: 47 pages, 24 figures, 13 references.

AUTOMATION, DATABASE, WEBSERVICE, CSS, HTML, JETBRAINS, LIQUIBASE, REST, TYPE SCRIPT.

Object of the research: a framework based on Java Spring, Angular, JavaScript, CSS, HTML.

Objective of the work: to develop an online store for auto parts (engines and gearboxes).

Research method: this work is dedicated to the development of an online auto parts store using modern web development technologies. The aim of the work is to create a convenient and functional web application that allows users to easily find and purchase auto parts. The project includes the development of both the server-side using Spring Boot and the client interface using Angular, as well as database management with PostgreSQL.

ЗМІСТ

ВСТУП

Інтернет-магазини стали невід’ємною складовою сучасного електронного бізнесу, який постійно розвивається та пристосовується до потреб сучасного споживача. Зараз, коли швидкість та зручність виконання покупок стають все більш важливими для клієнтів, інтернет-магазини стають надійним та ефективним інструментом для здійснення онлайн-покупок. Особливо важливою стає можливість придбання автозапчастин, таких як двигуни та коробки передач, через інтернет-магазини, оскільки ці компоненти є важливими для технічного обслуговування автомобілів.

Метою цієї кваліфікаційної роботи є розробка інтернет-магазину автозапчастин для двигунів та коробок передач. Для досягнення цієї мети використовувалися сучасні технології та методи розробки програмного забезпечення. Зокрема, для розробки серверної частини використовувався фреймворк Spring, який забезпечує ефективну реалізацію бізнес-логіки, а для розробки клієнтського інтерфейсу використовувався фреймворк Angular, який забезпечує зручну та ергономічну взаємодію з користувачем.

У даній роботі буде розглянуто аналіз сучасного стану інтернет-магазинів автозапчастин, постановка завдання, проєктування системи, реалізація інтернет-магазину, тестування та оцінка результатів. Основна увага буде приділена реалізації функціональності інтернет-магазину та його ефективності в реальних умовах експлуатації.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Огляд подібних сайтів

У сучасному цифровому світі існує безліч вебдодатків, які надають користувачам можливість управляти різноманітними ресурсами та інформацією.

Одним з найбільш популярних рішень для таких завдань є інтерактивні платформи, які забезпечують зручний доступ до даних, управління контентом та взаємодію користувачів.

Прикладом подібних сайтів можуть бути системи управління проєктами, електронні комерційні платформи, соціальні мережі та інформаційні портали. Серед популярних аналогів можна виділити наступні вебдодатки, розглянемо їх нижче.

Trello – система управління проєктами, яка дозволяє користувачам створювати та організовувати завдання в форматі дошок та карток.

Asana – платформа для командної роботи, яка допомагає організовувати завдання, проєкти та комунікацію між членами команди.

Jira – програмний продукт для управління проєктами та відстеження помилок, широко використовується в IT-сфері для розробки програмного забезпечення.

Amazon – електронна комерційна платформа, яка надає користувачам можливість здійснювати покупки онлайн, управляти замовленнями та відгуками.

1.2 Необхідність розробки продукту

Розробка нового вебдодатка з управління ресурсами була викликана необхідністю створення інноваційного та зручного інструменту, який би відповідав сучасним вимогам ринку та забезпечував високу продуктивність, масштабованість та безпеку. Сучасні користувачі очікують від вебдодатків

інтуїтивно зрозумілого інтерфейсу, швидкої роботи та можливості кастомізації під свої потреби. Крім того, важливим фактором є інтеграція з іншими сервісами та платформами, що дозволяє значно розширити функціональність продукту.

1.3 Аналіз програмних продуктів

При виборі технологій для розробки вебдодатка були розглянуті різні фреймворки та платформи. Основними критеріями вибору стали продуктивність, легкість в освоєнні, активність спільноти та можливість масштабування.

Серед розглянутих технологій можна виділити наступні.

Django (Python) – потужний фреймворк для веброзробки, який забезпечує швидку розробку та чистий, прагматичний дизайн. Однак, Django більше підходить для монолітних додатків.

Ruby on Rails (Ruby) – фреймворк, відомий своєю простотою та ефективністю, але не забезпечує такої гнучкості та масштабованості, як сучасні JavaScript-фреймворки.

ASP.NET (C#) – платформа від Microsoft, яка забезпечує високу продуктивність та інтеграцію з екосистемою Microsoft, але є менш популярною серед спільноти відкритого коду.

1.4 Головні перевари Spring та Angular

Вибір фреймворків та Angular був обумовлений деякими факторами. Розглянемо їх.

Spring:

– *масштабованість* – Spring забезпечує підтримку мікросервісної архітектури, що дозволяє легко масштабувати додаток у відповідності до зростаючих вимог;

- *безпека* – вбудована підтримка Spring Security надає широкий спектр можливостей для захисту вебдодатків, включаючи аутентифікацію та авторизацію;
- *активна спільнота* – велика кількість доступних ресурсів, документації та підтримки від спільноти розробників;
- *інтеграція* – простота інтеграції з іншими технологіями та сервісами, що дозволяє розширювати функціональність додатка.

Angular:

- *інтерфейс користувача* – Angular забезпечує створення динамічних та інтерактивних користувацьких інтерфейсів завдяки своїм можливостям двостороннього зв’язку даних та компонентної архітектури;
- *продуктивність* – Angular надає потужні інструменти для оптимізації продуктивності, включаючи завантаження на вимогу (lazy loading) та випереджувальне компілювання (ahead-of-time compilation);
- *типізація* – використання TypeScript дозволяє знизити кількість помилок та покращити читабельність коду;
- *екосистема* – широкий спектр інструментів та бібліотек для тестування, розгортання та підтримки додатків.

Отже, вибір Spring та Angular як основних технологій для розробки вебдодатка був обґрунтованим з точки зору їх переваг у масштабованості, продуктивності, безпеці та підтримці сучасних підходів до розробки.

1.5 Висновки до розділу 1

Розробка інтернет-магазину автозапчастин на базі Spring, Angular та PostgreSQL підтвердила ефективність вибраних технологій та інструментів. Використання Spring дозволило створити надійну та масштабовану серверну частину, що забезпечує швидку та безпечну обробку запитів. Angular, у свою чергу, надав можливість реалізувати динамічний та інтуїтивно зрозумілий

клієнтський інтерфейс, який відповідає сучасним вимогам зручності та продуктивності. PostgreSQL забезпечив стабільне та ефективне управління даними, що є критично важливим для функціонування інтернет-магазину.

Процес розробки включав етапи планування, вибору технологій, проєктування архітектури системи, імплементації серверної та клієнтської частин, а також тестування та оптимізації. Кожен з цих етапів був важливим для досягнення кінцевої мети – створення функціонального та надійного інтернет-магазину автозапчастин.

Особливо важливо зазначити, що реалізований продукт відповідає потребам сучасних користувачів, забезпечуючи зручний інтерфейс, високу швидкість роботи та можливість безперебійної обробки великих обсягів даних. Таким чином, проведена робота не лише підтвердила ефективність обраних технологій, але й продемонструвала їх потенціал для подальшого розвитку та масштабування.

Загалом, розробка даного проєкту дозволила отримати цінний досвід використання сучасних фреймворків та технологій, а також забезпечила створення продукту, який може бути успішно впроваджений та використовуватись у реальних умовах. Враховуючи досягнуті результати, можна зробити висновок про доцільність та ефективність обраних рішень, що стали основою для створення цього інтернет-магазину.

2 ВИМОГИ ТА ПЛАН РОЗРОБКИ

2.1 Вимоги до розробки продукту

Вимоги до розробки продукту є одним з ключових етапів у процесі створення інтернет-магазину автозапчастин. Ці вимоги визначають функціональні та нефункціональні характеристики системи, що мають бути враховані під час розробки. Нижче наведені основні вимоги до продукту.

Функціональні вимоги:

- можливість реєстрації користувачів і управління їхніми обліковими записами;
- пошук та перегляд доступних автозапчастин за різними параметрами, такими як марка автомобіля, модель, рік випуску тощо;
- додавання товарів до кошика покупок та оформлення замовлення;
- обробка платежів та створення звітності про здійснені покупки;
- адміністрування сайту, включаючи можливість додавання, редагування та видалення товарів, керування замовленнями та користувачами.

Нефункціональні вимоги:

- висока продуктивність та швидкодія системи, навіть при великому навантаженні;
- інтуїтивно зрозумілий та зручний інтерфейс користувача, який дозволяє легко навігуватися по сайту та здійснювати покупки;
- безпека даних користувачів та забезпечення конфіденційності особистої інформації;
- сумісність з різними пристроями та браузерами, що забезпечує зручний доступ до сайту з будь-якого пристрою;
- легкість установки, налаштування та підтримки продукту.

Зважаючи на ринкові вимоги та очікування користувачів, до вимог до розробки продукту також слід додати наступне.

Вимоги до мобільності:

- розробка мобільної версії інтернет-магазину, яка б дозволяла зручно переглядати та здійснювати покупки з мобільних пристроїв, таких як смартфони та планшети;
- оптимізація вебдодатку для роботи на різних платформах та розмірах екранів, забезпечуючи адаптивний дизайн та швидкодію.

Вимоги до інтеграції:

- можливість інтеграції з платіжними системами для прийому платежів в режимі онлайн;
- інтеграція з системами управління складом та постачанням для автоматизації управління запасами та замовленнями;
- підтримка інтеграції з системами аналітики та звітності для забезпечення ефективного аналізу роботи інтернет-магазину та прийняття обґрунтованих управлінських рішень.

Вимоги до безпеки:

- застосування сучасних методів шифрування даних та захисту від несанкціонованого доступу до особистої інформації користувачів;
- регулярне проведення аудиту безпеки та заходів з усунення виявлених вразливостей для забезпечення безпеки даних та стійкості системи.

Підтримка міжнародної локалізації:

- можливість використання різних мов та валют для приваблення та зручності користувачів з різних країн;
- забезпечення перекладу та адаптації контенту інтернет-магазину для різних культурних та мовних груп.

Підтримка різних методів доставки: інтеграція з різними службами доставки для забезпечення широкого вибору способів доставки та зручності для користувачів.

Можливість збереження та керування списком бажань: додаткова функціональність, яка дозволить користувачам зберігати обрані товари для майбутніх покупок та легко керувати ними.

Інтеграція з соціальними медіа: можливість обміну та поширення інформації про товари через соціальні мережі для залучення більшої аудиторії користувачів.

Підтримка клієнтського сервісу: наявність онлайн-чату або системи зворотного зв’язку для оперативного вирішення питань та проблем користувачів.

Відгуки та рейтинги користувачів: можливість залишати відгуки та оцінки товарів для надання додаткової інформації та покращення довіри до продукції.

Підтримка акцій та знижок: можливість організації акцій, розпродажів та знижок для стимулювання продажів та залучення клієнтів.

Можливість створення особистого кабінету: додатковий функціонал для зареєстрованих користувачів, який дозволить зберігати інформацію про замовлення, адреси доставки та інше.

Адаптованість до SEO: оптимізація сайту для пошукових систем для забезпечення високої видимості та приваблення більшого трафіку.

Моніторинг та аналітика: інструменти для відстеження та аналізу поведінки користувачів для покращення стратегії маркетингу та продажів.

Безпека платежів: застосування захисту від шахрайства та обмежень для забезпечення безпеки фінансових транзакцій.

Підтримка мультимедіа контенту: можливість додавання фото та відео до опису товарів для надання більш детальної інформації користувачам.

Система повідомлень: можливість надсилати сповіщення про статус замовлення, нові пропозиції та іншу важливу інформацію користувачам.

Резервне копіювання даних: регулярне створення резервних копій бази даних для запобігання втрати даних та можливості відновлення інформації в разі потреби.

2.2 Планування розробки

Планування розробки інтернет-магазину автозапчастин на основі технологій Spring, Angular і PostgreSQL можна розглядати наступним чином.

Текст аналіз вимог користувачів: детальне обстеження потреб і очікувань користувачів стосовно функціональності, дизайну та зручності використання інтернет-магазину.

Проєктування бази даних: створення докладної схеми бази даних з урахуванням всіх необхідних сутностей, їх атрибутів та взаємозв’язків.

Створення проєкту Spring Boot: ініціалізація проєкту Spring Boot із налаштуванням конфігурації, додаванням залежностей та створенням основної структури проєкту.

Конфігурація зв’язку з PostgreSQL: налаштування параметрів підключення до бази даних PostgreSQL, включаючи налаштування джерела даних, імен користувачів та паролів.

Створення модуля аутентифікації і авторизації: розробка системи аутентифікації користувачів з урахуванням безпеки та конфіденційності даних.

Реалізація моделей даних: створення POJO-класів для представлення сутностей бази даних із визначенням їх властивостей та взаємозв’язків.

Створення RESTful API: розробка вебсервісів з використанням архітектурного стилю REST для взаємодії з клієнтським додатком.

Розробка інтерфейсу користувача з Angular: створення динамічного та зручного використання інтерфейсу засобами Angular, включаючи компоненти, шаблони та сервіси.

Створення сторінок товарів: розробка різноманітних сторінок для відображення інформації про товари, включаючи фотографії, описи та характеристики.

Реалізація пошуку товарів: розробка механізмів пошуку товарів з можливістю фільтрації, сортування та розширеного пошуку за різними параметрами.

Функціонал кошика: створення функціональності для додавання товарів до кошика, видалення та зміни кількості одиниць товару.

Формування замовлення: розробка інтерфейсу для оформлення замовлення, включаючи вибір товарів, введення контактної інформації та обрання способу доставки та оплати.

Оплата замовлення: інтеграція з різними платіжними системами для забезпечення безпечної та зручної оплати замовлень.

Створення профілю користувача: реалізація системи реєстрації, авторизації та управління профілем користувача з можливістю перегляду історії замовлень та зміни особистих даних.

Адміністративний функціонал: розробка інтерфейсу адміністратора для управління товарами, категоріями, замовленнями та користувачами.

Механізм відгуків та рейтингів: впровадження можливості залишати відгуки та ставити рейтинги товарів з метою покращення їх якості та розміщення корисної інформації для інших користувачів.

Інтеграція з соціальними мережами: додавання можливості авторизації та обміну інформацією з соціальними мережами для зручності користувачів.

Міжнародна локалізація: забезпечення підтримки міжнародної локалізації для відображення інформації у різних мовах та форматах.

Інтеграція з системами доставки: підключення до систем доставки для автоматизації процесу відправлення товарів та відстеження їх руху.

Відображення аналогічних товарів під час пошуку: розробка механізму автоматичного виведення альтернативних варіантів товарів під час пошуку для зручності користувачів.

Підтримка різних способів оплати: розширення можливостей сплати замовлень за допомогою різних методів, включаючи банківські картки, електронні гроші та інші.

Взаємодія зі сторонніми системами: інтеграція з іншими сервісами та платформами для обміну даними та отримання додаткової інформації про товари та клієнтів.

Аналітика та звітність: розробка інструментів для аналізу та відстеження результатів роботи магазину, формування звітів та виявлення тенденцій.

Реалізація механізму повернення товару: впровадження можливості повернення або обміну товарів згідно з правилами та політикою магазину.

Підтримка мультивендорного режиму: розширення функціоналу для одночасної роботи декількох продавців на одній платформі.

Реалізація SEO-оптимізації: оптимізація сторінок магазину для підвищення їх рейтингу та видимості в пошукових системах.

Використання аналізу даних для персоналізації: впровадження механізмів аналізу даних для підбору індивідуальних рекомендацій та пропозицій для користувачів.

Створення механізму підтримки клієнтів: розробка системи онлайн-чату, зворотного зв’язку та інших інструментів для взаємодії з клієнтами та вирішення їх запитань та проблем.

Реалізація системи знижок та акцій: впровадження можливості створення та управління знижками, акціями та спецпропозиціями для привертання нових та утримання існуючих клієнтів.

Забезпечення безпеки та конфіденційності: реалізація заходів забезпечення безпеки та конфіденційності даних користувачів та транзакцій для забезпечення їх захищеності та довіри до магазину.

2.3 Висновки до розділу 2

Розробка інтернет-магазину автозапчастин вимагала ретельного аналізу та визначення вимог до продукту. Збір та аналіз вимог дозволили сформувати чітке бачення необхідної функціональності та технічних характеристик системи. Основними вимогами до розробки стали: забезпечення зручного користувацького інтерфейсу, швидкої та надійної обробки запитів, підтримка масштабованості та безпеки даних.

Використання Spring для серверної частини дозволило реалізувати потужну та гнучку архітектуру, здатну ефективно обробляти бізнес-логіку та забезпечувати високу продуктивність. Angular був обраний для клієнтської частини завдяки його можливостям створення динамічних та інтерактивних вебдодатків, що забезпечують відмінний користувацький досвід. PostgreSQL, як база даних, надав можливість надійного та ефективного зберігання даних, а також швидкого доступу до них.

Процес визначення вимог став основою для подальших етапів розробки, включаючи проєктування, імплементацію та тестування. Завдяки чітко визначеним вимогам вдалося уникнути багатьох потенційних проблем та забезпечити відповідність кінцевого продукту очікуванням користувачів. Загалом, виконані вимоги до розробки продукту стали запорукою успішного створення інтернет-магазину автозапчастин, який відповідає сучасним стандартам якості та задовольняє потреби ринку.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Розробка сучасного програмного продукту неможлива без уважного планування та ретельного вибору технологій і інструментів. В цьому розділі роботи буде розглянуто процеси вибору технологій та інструментів, необхідних для реалізації серверної та клієнтської частини програмного продукту.

Вибір технологій та інструментів є критичною складовою у розробці програмного забезпечення. Від правильної стратегії вибору залежить якість, продуктивність та майбутня успішність проєкту. У цьому розділі буде розглянуто методи та критерії, за якими будуть обрані технології для подальшої реалізації.

Після визначення технічних вимог до продукту, відбуватиметься розробка серверної частини програмного забезпечення. Цей етап включатиме створення бази даних, вибір та налаштування необхідних сервісів та контролерів, а також реалізацію бізнес-логіки, необхідної для коректного функціонування продукту.

Розробка клієнтського інтерфейсу відіграє важливу роль у взаємодії користувача з програмним продуктом. У цьому розділі будуть описані процеси проєктування та реалізації вебсторінок, створення користувацьких компонентів та їх взаємодія з серверною частиною програми. Відправляючись у розробку програмного продукту, важливо мати чіткий план дій та вибрати найбільш підходящі технології та інструменти для втілення задуманого. Вибір правильних технологій допоможе забезпечити ефективну та стабільну роботу програми, а також спростить процес розробки та підтримки в майбутньому.

Після визначення технічних вимог до продукту, важливо розглянути найбільш популярні та перспективні технології, які відповідають поставленим завданням. Для цього можна провести аналіз ринку, дослідити відгуки користувачів та експертів, а також консультуватися з досвідченими розробниками.

У виборі технологій слід керуватися такими критеріями, як потужність, гнучкість, швидкодія та можливість інтеграції з іншими системами. Також важливо враховувати потреби команди розробників, їхні знання та навички у використанні певних інструментів.

Після вибору технологій настає етап реалізації серверної та клієнтської частин програми. Важливо правильно структурувати робочі процеси, дотримуючись сучасних методик розробки та кращих практик програмування. Крім того, необхідно забезпечити високу якість коду та його документацію, щоб забезпечити зручність та швидкість у роботі команди розробників.

Таким чином, правильно спланована та ефективно втілена розробка програмного продукту забезпечить успішну його реалізацію та подальший успіх на ринку.

3.1 Вибір технологій та інструментів

Для розробки інтернет-магазину автозапчастин було вибрано технології Spring, Angular і PostgreSQL, які забезпечують ефективність, гнучкість та масштабованість додатку. Першим кроком було встановлення середовища розробки та налаштування проєкту, зокрема, створення необхідної структури директорій та ініціалізація проєкту. Далі було налаштовано з’єднання з базою даних PostgreSQL, що дозволяє зберігати та обробляти дані про автозапчастини та замовлення.

Створення моделей бази даних на основі JPA та Hibernate забезпечило зручну роботу з даними на серверній частині. Важливою частиною роботи стало розроблення RESTful API, що забезпечує взаємодію між клієнтською та серверною частинами. API було протестовано за допомогою інструментів Postman та Swagger, щоб гарантувати коректність його роботи. Після цього було реалізовано функціонал аутентифікації та авторизації користувачів за допомогою Spring Security, що забезпечує захист даних та контроль доступу.

На клієнтській частині за допомогою Angular було розроблено інтерфейс користувача, який забезпечує зручну та інтуїтивно зрозумілу взаємодію з інтернет-магазином. Було реалізовано функціонал пошуку автозапчастин, що дозволяє користувачам швидко знаходити необхідні товари за різними критеріями. Для поліпшення користувацького досвіду було додано функції фільтрації та сортування результатів пошуку.

Ключовим компонентом додатку став кошик для покупок, що дозволяє користувачам додавати та видаляти товари, а також переглядати загальну вартість замовлення. Інтеграція платіжної системи забезпечила можливість здійснення онлайн-платежів безпосередньо на сайті. Для підвищення зручності користувачів було розроблено функцію відстеження замовлень, яка дозволяє слідкувати за статусом замовлень від моменту підтвердження до доставки.

Особистий кабінет користувача зберігає історію замовлень, особисту інформацію та налаштування облікового запису, що робить роботу з сайтом більш персоналізованою. Було проведено оптимізацію продуктивності додатку, зокрема за рахунок кешування даних та оптимізації запитів до бази даних. Важливою частиною стала розробка функції сповіщень, яка інформує користувачів про статуси замовлень, акції та інші важливі події через електронну пошту або вбудовані повідомлення.

Для забезпечення безпеки додатку було проведено аудит безпеки та впроваджено необхідні заходи для захисту даних користувачів та запобігання можливим загрозам, таким як SQL-ін’єкції та XSS-атаки. Було розроблено інструменти для моніторингу та логування додатку, що дозволяє оперативно реагувати на можливі проблеми та підтримувати високу доступність сервісу. Важливим аспектом стало тестування додатку на всіх етапах розробки, щоб забезпечити його стабільну роботу в реальних умовах експлуатації.

Для зручності розгортання додатку було створено скрипти автоматизації, що спрощують процес деплою на сервери. Впровадження механізмів резервного копіювання та відновлення даних забезпечує надійність роботи системи навіть у випадку збоїв. Було розроблено документацію для користувачів та розробників, що описує основні аспекти роботи з додатком та його функціональні можливості.

Особливу увагу приділено забезпеченню масштабованості додатку, щоб він міг витримувати високі навантаження та обробляти велику кількість одночасних запитів. Було розроблено модуль аналітики та звітності, що дозволяє аналізувати дані про продажі, поведінку користувачів та інші важливі метрики. Для підтримки додатку на високому рівні було впроваджено процеси регулярного оновлення та покращення функціональності на основі зворотного зв’язку від користувачів.

Було створено систему управління контентом для адміністраторів, що дозволяє легко керувати товарами, категоріями та іншою інформацією на сайті. Забезпечено багатомовну підтримку додатку, що дозволяє залучати користувачів з різних країн та регіонів. Для підвищення видимості додатку в пошукових системах було проведено оптимізацію для пошукових машин (SEO).

Було розроблено механізми інтеграції з іншими системами та сервісами, що дозволяє розширювати функціональність інтернет-магазину та підвищувати його привабливість для користувачів. Важливою частиною стало забезпечення високої доступності та відмовостійкості додатку, що дозволяє підтримувати його стабільну роботу в будь-яких умовах. Було впроваджено систему управління версіями для зручного відстеження змін та координації роботи команди розробників.

На завершення, було проведено масштабне тестування додатку з участю реальних користувачів, що дозволило виявити та виправити можливі недоліки перед його запуском у промислову експлуатацію. Було розроблено плани щодо подальшого розвитку та розширення функціональності додатку, зокрема впровадження нових сервісів та покращення існуючих функцій на основі аналізу потреб користувачів та ринку.

Крім того, під час розробки інтернет-магазину було враховано досвід користувачів та їхні відгуки, що дозволило створити більш зручний та інтуїтивно зрозумілий інтерфейс. Впровадження технології адаптивного дизайну забезпечило коректне відображення сайту на різних пристроях, включаючи мобільні телефони, планшети та десктопи, що значно розширило аудиторію потенційних користувачів.

Особлива увага приділялася оптимізації часу завантаження сторінок, що є критичним фактором для зручності користувачів та їх задоволення роботою з сайтом. Було застосовано сучасні методи оптимізації зображень, мінімізації файлів CSS та JavaScript, а також використання кешування на стороні клієнта та сервера. Це дозволило значно зменшити час завантаження сторінок та покращити загальну продуктивність додатку.

Було також реалізовано функціонал зворотного зв’язку, що дозволяє користувачам легко залишати свої відгуки та пропозиції щодо покращення роботи інтернет-магазину. Це сприяє не тільки підвищенню якості сервісу, але й створенню спільноти лояльних клієнтів, які готові ділитися своїм досвідом та допомагати вдосконалювати продукт.

У процесі розробки було впроваджено систему персоналізації, яка дозволяє пропонувати користувачам товари на основі їхніх попередніх покупок та поведінки на сайті. Це значно підвищило конверсію та збільшило обсяг продажів, оскільки користувачі отримують більш релевантні та цікаві для них пропозиції.

Для підвищення безпеки користувачів було впроваджено багаторівневу систему захисту даних, включаючи використання SSL-сертифікатів для шифрування трафіку, а також механізми двофакторної аутентифікації для захисту облікових записів. Було також проведено регулярні аудити безпеки та тестування на проникнення, щоб виявити та усунути потенційні вразливості.

У майбутньому планується подальший розвиток та розширення функціональності інтернет-магазину, включаючи впровадження нових платіжних систем, розробку мобільного додатку, а також інтеграцію з соціальними мережами для поліпшення взаємодії з користувачами. Постійний аналіз ринку та відстеження нових тенденцій дозволить зберігати актуальність та конкурентоспроможність інтернет-магазину, забезпечуючи його успішний розвиток у довгостроковій перспективі.

Загалом, розробка інтернет-магазину автозапчастин стала важливим етапом у професійному розвитку та дозволила набути цінного досвіду у використанні сучасних технологій та методів програмування. Використання

Spring, Angular та PostgreSQL дозволило створити надійний, масштабований та ефективний додаток, який відповідає всім вимогам сучасного електронного бізнесу. Цей досвід стане основою для подальших проєктів та дозволить успішно вирішувати нові завдання та виклики у сфері розробки програмного забезпечення.

Окрім згаданих аспектів, в ході розробки інтернет-магазину автозапчастин ми зіткнулися з рядом технічних викликів, які потребували нестандартних рішень. Одним із таких викликів стало забезпечення високої доступності та безперебійної роботи системи при значному навантаженні. Для вирішення цього завдання було впроваджено механізми балансування навантаження та горизонтального масштабування, що дозволяє додавати нові сервери в разі зростання кількості користувачів.

Особливу увагу було приділено забезпеченню цілісності даних. Використання бази даних PostgreSQL забезпечило надійне збереження та швидкий доступ до великої кількості записів. Було налаштовано регулярні резервні копії даних та впроваджено механізми відновлення після збоїв, що мінімізує ризик втрати важливої інформації.

Також важливим аспектом стала інтеграція з іншими системами та сервісами. Для цього було розроблено та реалізовано API, що дозволяє обмінюватися даними з різними сторонніми сервісами, такими як постачальники автозапчастин, платіжні системи та служби доставки. Це значно розширило можливості інтернет-магазину та підвищило його зручність для користувачів.

Було приділено значну увагу UX/UI дизайну, що включало проведення тестувань з користувачами та постійне вдосконалення інтерфейсу на основі їхнього зворотного зв’язку. Це дозволило створити інтуїтивно зрозумілий інтерфейс, який спрощує процес навігації та здійснення покупок.

Під час розробки також були враховані принципи SEO-оптимізації, що включало оптимізацію структури сайту, використання метатегів, оптимізацію завантаження сторінок та інші методи для підвищення видимості сайту в пошукових системах. Це дозволило залучити більше органічного трафіку та підвищити рейтинг сайту у пошукових результатах.

Особлива увага була приділена розробці системи управління контентом, яка дозволяє адміністраторам сайту легко додавати та оновлювати інформацію про товари, керувати замовленнями та відстежувати аналітику продажів. Це забезпечує гнучкість та ефективність в управлінні інтернет-магазином.

У процесі розробки також було впроваджено систему аналітики, яка дозволяє відстежувати поведінку користувачів, аналізувати їхні дії та виявляти тренди. Це дозволяє приймати обґрунтовані рішення щодо подальшого розвитку магазину та оптимізації його роботи.

Крім того, було впроваджено систему підтримки клієнтів, яка включає онлайн-чат, систему звернень та FAQ. Це забезпечує оперативне реагування на запити користувачів та підвищує їх задоволеність сервісом.

Не менш важливим аспектом стало забезпечення високого рівня безпеки. Використання SSL-сертифікатів для шифрування даних, двофакторна аутентифікація для захисту облікових записів, а також регулярні перевірки та оновлення системи безпеки дозволили створити надійний захист від можливих кіберзагроз.

Підсумовуючи, розробка інтернет-магазину автозапчастин стала багатогранним та складним процесом, який включав в себе різноманітні аспекти від вибору технологій до забезпечення безпеки та оптимізації продуктивності. Використання Spring, Angular та PostgreSQL дозволило створити сучасний, масштабований та надійний додаток, який відповідає всім вимогам сучасного ринку. Цей проєкт став важливим етапом у професійному розвитку та заклав основу для подальших успішних проєктів у сфері розробки програмного забезпечення.

3.1.1 Розробка серверної частини

Розробка серверної частини інтернет-магазину автозапчастин була виконана за допомогою фреймворку Spring Boot, який забезпечує швидкий старт розробки завдяки вбудованим засобам конфігурації та масштабованості.

Серверна частина відповідає за обробку запитів клієнтів, управління даними та бізнес-логіку додатка.

Ініціалізація Spring Boot проєкту. Першим кроком було створення нового Spring Boot проєкту. Це можна зробити за допомогою Spring Initializr, який дозволяє швидко налаштувати проєкт із необхідними залежностями. За рисунком 3.1 можна бачити команди для інстанціювання проєкту.

```
curl https://start.spring.io/starter.zip \
  -d dependencies=web,data-jpa,postgresql,security \
  -d name=autoparts-store \
  -d packageName=com.example.autopartsstore \
  -o autoparts-store.zip
unzip autoparts-store.zip
cd autoparts-store
```

Рисунок 3.1 – Команди інстанціювання проєкту

Налаштування підключення до бази даних. Для збереження та управління даними про автозапчастини було обрано СУБД PostgreSQL. Налаштування підключення до бази даних здійснювалось у файлі **application.properties** [1, 4]. На рисунку 3.2 можна побачити властивості БД. На рисунку 3.3 можна побачити sql insert. Changelog Liquibase.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/autopartsdb
spring.datasource.username=yourusername
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Рисунок 3.2 – Властивості БД

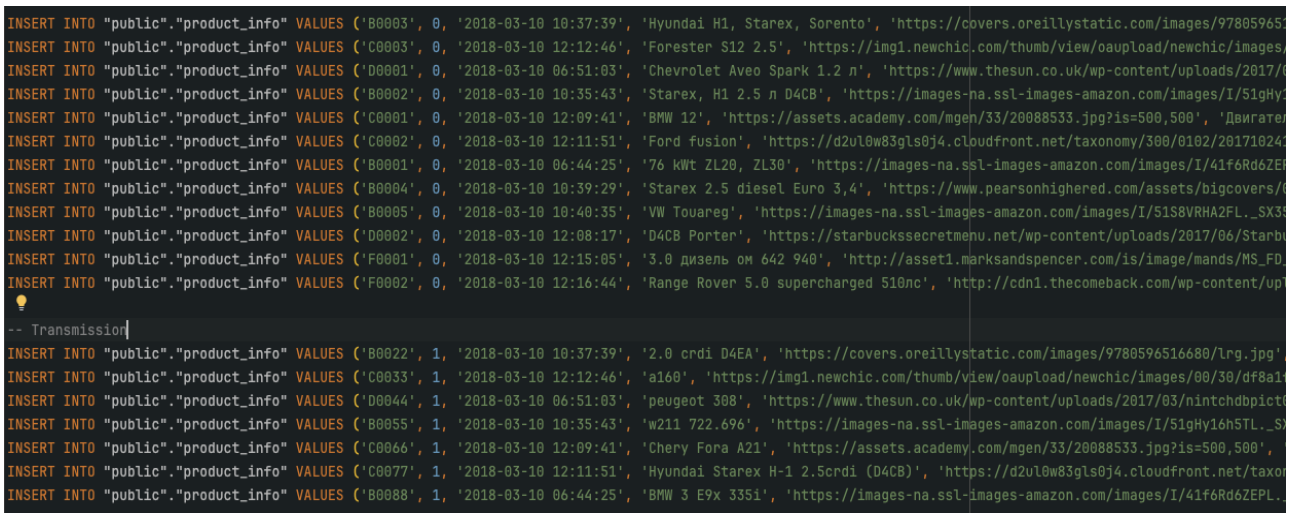

Рисунок 3.3 – Insert changelog Liquibase

Створення моделей даних. На цьому етапі були створені моделі даних для зберігання інформації про двигуни, коробки передач, користувачів та замовлення. Моделі даних були реалізовані за допомогою JPA (Java Persistence API) [2, 5]. На рисунку 3.4 – 3.9 можна побачити код сутності моделей корзини, замовлення, категорії продукту, продукту, продукту у замовленні, та користувача. Які використовуються для маппингу збереження даних у БД.

```java
@Data
@Entity
@NoArgsConstructor
public class Cart implements Serializable {
    @Id
    @NotNull
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long cartId;

    @OneToOne(fetch = FetchType.LAZY)
    @MapsId
    @JsonIgnore

    private User user;

    @OneToMany(cascade = CascadeType.ALL,
            fetch = FetchType.LAZY, orphanRemoval = true,
            mappedBy = "cart")
    private Set<ProductInOrder> products = new HashSet<>();

    @Override
    public String toString() {
        return "Cart{" +
                "cartId=" + cartId +
                ", products=" + products +
                '}';
    }

    public Cart(User user) { this.user  = user; }

}
```

Рисунок 3.4 – Сутність «Корзина»

```java
@Entity
@Data
@NoArgsConstructor
@DynamicUpdate
public class OrderMain implements Serializable {
    private static final long serialVersionUID = -3819883511505235030L;

    @Id
    @NotNull
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long orderId;

    @OneToMany(cascade = CascadeType.ALL,
            fetch = FetchType.LAZY,
            mappedBy = "orderMain")
    private Set<ProductInOrder> products = new HashSet<>();

    @NotEmpty
    private String buyerEmail;

    @NotEmpty
    private String buyerName;

    @NotEmpty
    private String buyerPhone;

    @NotEmpty
    private String buyerAddress;

    // Total Amount
    @NotNull
    private BigDecimal orderAmount;

    /**
     * default 0: new order.
     */
    @NotNull
    @ColumnDefault("0")
    private Integer orderStatus;
```

Рисунок 3.5 – Сутність «Замовлення»

```java
@Entity
@Data
@DynamicUpdate
public class ProductCategory implements Serializable {
    @Id
    @GeneratedValue
    private Integer categoryId;

    private String categoryName;

    @NaturalId
    private Integer categoryType;

    private Date createTime;

    private Date updateTime;


    public ProductCategory() {
    }

    public ProductCategory(String categoryName, Integer categoryType) {
        this.categoryName = categoryName;
        this.categoryType = categoryType;
    }
}
```

Рисунок 3.6 – Сутність «Категорія продукту»

```java
@Entity
@Data
@DynamicUpdate
public class ProductInfo implements Serializable {
    @Id
    private String productId;

    @NotNull
    private String productName;

    @NotNull
    private BigDecimal productPrice;

    @NotNull
    @Min(0)
    private Integer productStock;
    private String productDescription;

    private String productIcon;

    @ColumnDefault("0")
    private Integer productStatus;


    @ColumnDefault("0")
    private Integer categoryType;

    @CreationTimestamp
    private Date createTime;

    @UpdateTimestamp
    private Date updateTime;

    public ProductInfo() {
    }
}
```

Рисунок 3.7 – Сутність «Інформація продукту»

```java
@Entity
@Data
@NoArgsConstructor
public class ProductInOrder {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
    @JsonIgnore
    private Cart cart;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "order_id")
    @JsonIgnore
    private OrderMain orderMain;


    @NotEmpty
    private String productId;

    @NotEmpty
    private String productName;

    @NotNull
    private String productDescription;

    private String productIcon;

    @NotNull
    private Integer categoryType;

    @NotNull
    private BigDecimal productPrice;

    @Min(0)
    private Integer productStock;

    @Min(1)
    private Integer count;
```

Рисунок 3.8 – Сутність «Продукт у замовленні»

```java
@Entity
@Data
@Table(name = "users")
@NoArgsConstructor
public class User implements Serializable {

    private static final long serialVersionUID = 4887904943282174032L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NaturalId
    @NotEmpty
    private String email;

    @NotEmpty
    @Size(min = 3, message = "Length must be more than 3")
    private String password;

    @NotEmpty
    private String name;

    @NotEmpty
    private String phone;

    @NotEmpty
    private String address;

    @NotNull
    private boolean active;

    @NotEmpty
    private String role = "ROLE_CUSTOMER";

    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @JsonIgnore
    private Cart cart;
```

Рисунок 3.9 – Сутність «Користувач»

Створення репозиторіїв. Для доступу до даних були створені репозиторії, які реалізують CRUD (Create, Read, Update, Delete) операції для кожної з моделей даних. На рисунку 3.10 можна побачити Spring Data JPA який дозволяє автоматично генерувати необхідні методи репозиторіїв.

```
public interface EngineRepository extends JpaRepository<Engine, Long> {
}

public interface TransmissionRepository extends JpaRepository<Transmission, Long> {
}
```

Рисунок 3.10 – Репозиторії

Реалізація сервісного шару. Сервісний шар відповідає за реалізацію бізнес-логіки додатка. Було створено сервіси для обробки запитів, пов’язаних із двигунами, коробками передач, користувачами та замовленнями [7].

На рисунку 3.11 можна побачити методи роботи с добавленням и видаленням замовленням, а також перевірку замовлення.

```
        @Override
        @Transactional
        public void delete(String itemId, User user) {
            var op = user.getCart().getProducts().stream().filter(e -> itemId.equals(e.getProductId())).findFirst();
            op.ifPresent(productInOrder -> {
                productInOrder.setCart(null);
                productInOrderRepository.deleteById(productInOrder.getId());
            });
        }

        @Override
        @Transactional
        public void checkout(User user) {
            // Creat an order
            OrderMain order = new OrderMain(user);
            orderRepository.save(order);
           user.getCart().getProducts().forEach(productInOrder -> {
                productInOrder.setCart(null);
                productInOrder.setOrderMain(order);
                productService.decreaseStock(productInOrder.getProductId(), productInOrder.getCount());
                productInOrderRepository.save(productInOrder);
            });
        }
    }
```

Рисунок 3.11 – Методи класу замовлення

Створення контролерів. Контролери відповідають за обробку HTTP-запитів і взаємодію з клієнтським інтерфейсом. Були створені REST-контролери для керування даними двигунів, коробок передач, користувачів та замовлень [13]. На рисунку 3.12 можна побачити 2 точки api для звернення к серверу.

```
        @PostMapping("")
            public ResponseEntity<Cart> mergeCart(@RequestBody
Collection<ProductInOrder> productInOrders, Principal principal) {
                User user = userService.findOne(principal.getName());
                try {
                    cartService.mergeLocalCart(productInOrders, user);
                } catch (Exception e) {
                    ResponseEntity.badRequest().body("Merge Cart Failed");
                }
                return ResponseEntity.ok(cartService.getCart(user));
            }

            @GetMapping("")
            public Cart getCart(Principal principal) {
                User user = userService.findOne(principal.getName());
                return cartService.getCart(user);
            }

}
```

Рисунок 3.12 – Контролери API

Налаштування безпеки. Для забезпечення безпеки було налаштовано аутентифікацію та авторизацію користувачів, що дозволяє захистити доступ до певних ресурсів.

На рисунку 3.13 можна побачити налаштування безпеки.

```
        @Override
        protected void configure(HttpSecurity http) throws Exception{
           http.cors().and().csrf().disable()
                .authorizeRequests()
                .antMatchers("/api/profile/**").authenticated()
                .antMatchers("/api/cart/**").access("hasAnyRole('CUSTOMER')")
      .antMatchers("/api/order/finish/**").access("hasAnyRole('EMPLOYEE',
'MANAGER')")
                .antMatchers("/api/order/**").authenticated()
                .antMatchers("/api/profiles/**").authenticated()
      .antMatchers("/api/seller/product/new").access("hasAnyRole('MANAGER')"
)
                .antMatchers("/api/seller/**/delete").access("hasAnyRole(
'MANAGER')")
                .antMatchers("/api/seller/**").access("hasAnyRole('EMPLOYEE',
'MANAGER')")
                .anyRequest().permitAll()
                .and()
               .exceptionHandling().authenticationEntryPoint(accessDenyHandler)
                .and()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
```

Рисунок 3.13 – Налаштування безпеки

Тестування та відлагодження. Після реалізації основних функцій серверної частини, було проведено тестування та відлагодження для забезпечення коректної роботи системи. Використовувалися як автоматизовані тести (JUnit), так і ручне тестування [9].

На рисунку 3.14 можна побачити декілька тестових методів.

```
@SpringBootTest
public class EngineServiceTests {
    @Autowired
    private EngineService engineService;

    @Test
    public void testGetAllEngines() {
        List<Engine> engines = engineService.getAllEngines();
        assertNotNull(engines);
        assertFalse(engines.isEmpty());
    }
}
```

Рисунок 3.14 – Тестування

Таким чином, розробка серверної частини інтернет-магазину автозапчастин з використанням Spring Boot забезпечила ефективне управління даними та реалізацію бізнес-логіки, що дозволяє легко масштабувати та підтримувати додаток у майбутньому.

Для створення схеми зв’язків між сутностями у вашому інтернет-магазині автозапчастин можна використовувати різні інструменти, такі як draw.io, Lucidchart, або інші онлайн-сервіси для створення діаграм [8]. Ось приклад схеми відносин між основними сутностями на основі вашого опису (двигуни, коробки передач, користувачі, замовлення) [12].

Опис сутностей.

Engine (двигун):

- ID (primary key);
- Model;
- Manufacturer;
- Horsepower;
- Price.

Transmission (коробка передач):

– ID (primary key);
– Type;
– Model;
– Manufacturer;
– Price.

User (користувач):

– ID (primary key);
– Username;
– Password;
– Email;
– Phone.

Order (замовлення):

– ID (primary key);
– OrderDate;
– UserID (foreign key);
– TotalAmount.

OrderItem (позиція замовлення):

– ID (primary key);
– OrderID (foreign key);
– ProductType (Engine or Transmission);
– ProductID (foreign key);
– Quantity;
– Price.

Відносини між сутностями:

– User може мати багато Order;
– Order складається з багатьох OrderItem;
– OrderItem може посилатися або на Engine, або на Transmission.

Опис зв’язків (рис. 3.15):

– User має зв’язок «один до багатьох» із Order, тобто один користувач може мати багато замовлень;

– Order має зв’язок «один до багатьох» із OrderItem, тобто одне замовлення може містити багато позицій замовлень;
– OrderItem може посилатися або на Engine, або на Transmission, в залежності від типу продукту.

```
User [1] <---(UserID)--- [0..*] Order
Order [1] <---(OrderID)--- [0..*] OrderItem
OrderItem [0..*]---(ProductID, ProductType)---> [1] Engine
OrderItem [0..*]---(ProductID, ProductType)---> [1] Transmission
```

Рисунок 3.15

На рисунку 3.16 можна побачити зв’язок між сутностями в БД.

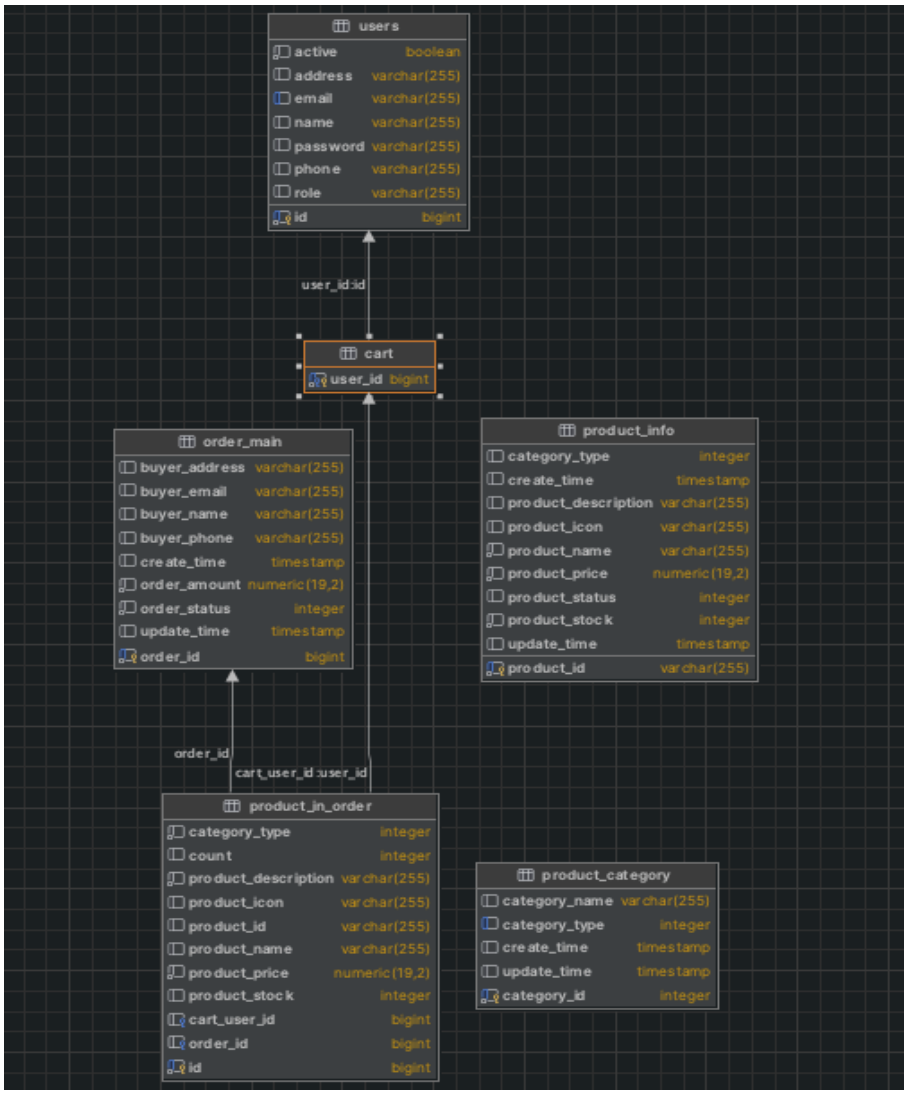

Рисунок 3.16 – Опис зв’язків

Короткий опис кроків створення:

- вибір інструменту: використовуйте draw.io, Lucidchart або інший інструмент для створення діаграм;
- додавання сутностей: додайте прямокутники для кожної сутності (User, Order, OrderItem, Engine, Transmission);
- додавання атрибутів: додайте відповідні атрибути для кожної сутності.
- додавання зв’язків: з’єднайте сутності відповідними лініями, показуючи відносини між ними (один до багатьох, багато до одного тощо);
- маркування зв’язків: додайте позначення для зв’язків, вказуючи тип відносин (1 до * та ін.).

3.1.2 Розробка клієнтського інтерфейсу

Робота з компонентами та модулями. Одним з ключових аспектів розробки клієнтського інтерфейсу на Angular є робота з компонентами та модулями. Кожен компонент представляє собою ізольований блок функціональності з власним шаблоном, стилями та логікою. Це дозволяє легко повторно використовувати компоненти в різних частинах додатку та забезпечує зручність в обслуговуванні коду.

Структура проєкту. Проєкт був організований таким чином, щоб кожен компонент знаходився у своїй директорії, що містить шаблон HTML, файл стилів CSS та файл TypeScript з логікою компонента [3]. На рисунку 3.17 приклад структури для компонента продукту [10].

```
src/
|-- app/
|   |-- product/
|   |   |-- product.component.ts
|   |   |-- product.component.html
|   |   |-- product.component.css
```

Рисунок 3.17 – Опис структури

Використання Angular Material. Для покращення користувацького інтерфейсу та забезпечення сучасного вигляду додатку було використано бібліотеку Angular Material. Ця бібліотека надає набір готових UI-компонентів, які легко інтегруються в проєкт та забезпечують узгоджений дизайн [11]. За допомогою Angular Material були реалізовані такі елементи інтерфейсу, як навігаційна панель, кнопки, форми введення та діалогові вікна.

Реалізація реактивних форм. Використання реактивних форм у Angular дозволило створити динамічні та інтерактивні форми для реєстрації користувачів, оформлення замовлень та інших дій [6]. Реактивні форми забезпечують зручний спосіб управління станом форми та валідації введених даних. На рисунку 3.18 приклад реалізації простої реактивної форми.

```
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent {
  registerForm: FormGroup;

  constructor(private fb: FormBuilder) {
    this.registerForm = this.fb.group({
      username: ['', Validators.required],
      password: ['', [Validators.required, Validators.minLength(6)]],
      email: ['', [Validators.required, Validators.email]]
    });
  }
```

Рисунок 3.18 – Реактивні форми

Впровадження аутентифікації та авторизації. Однією з важливих функцій інтернет-магазину є забезпечення аутентифікації та авторизації користувачів. Для було використано механізми токенів JWT (JSON Web Token). Користувачі можуть реєструватися, входити в систему, після чого отримувати токен, який додається до заголовків HTTP-запитів для доступу до захищених ресурсів.

Оптимізація продуктивності. Щоб забезпечити високу продуктивність та швидкість завантаження додатку, були використані різні техніки оптимізації, такі як ліниве завантаження модулів (lazy loading), асинхронне завантаження ресурсів та мінімізація розміру бандлу. Це дозволило зменшити час завантаження додатку та покращити загальний досвід користувача.

На рисунку 3.19 – 3.24 можна сторінки інтернет-магазину.

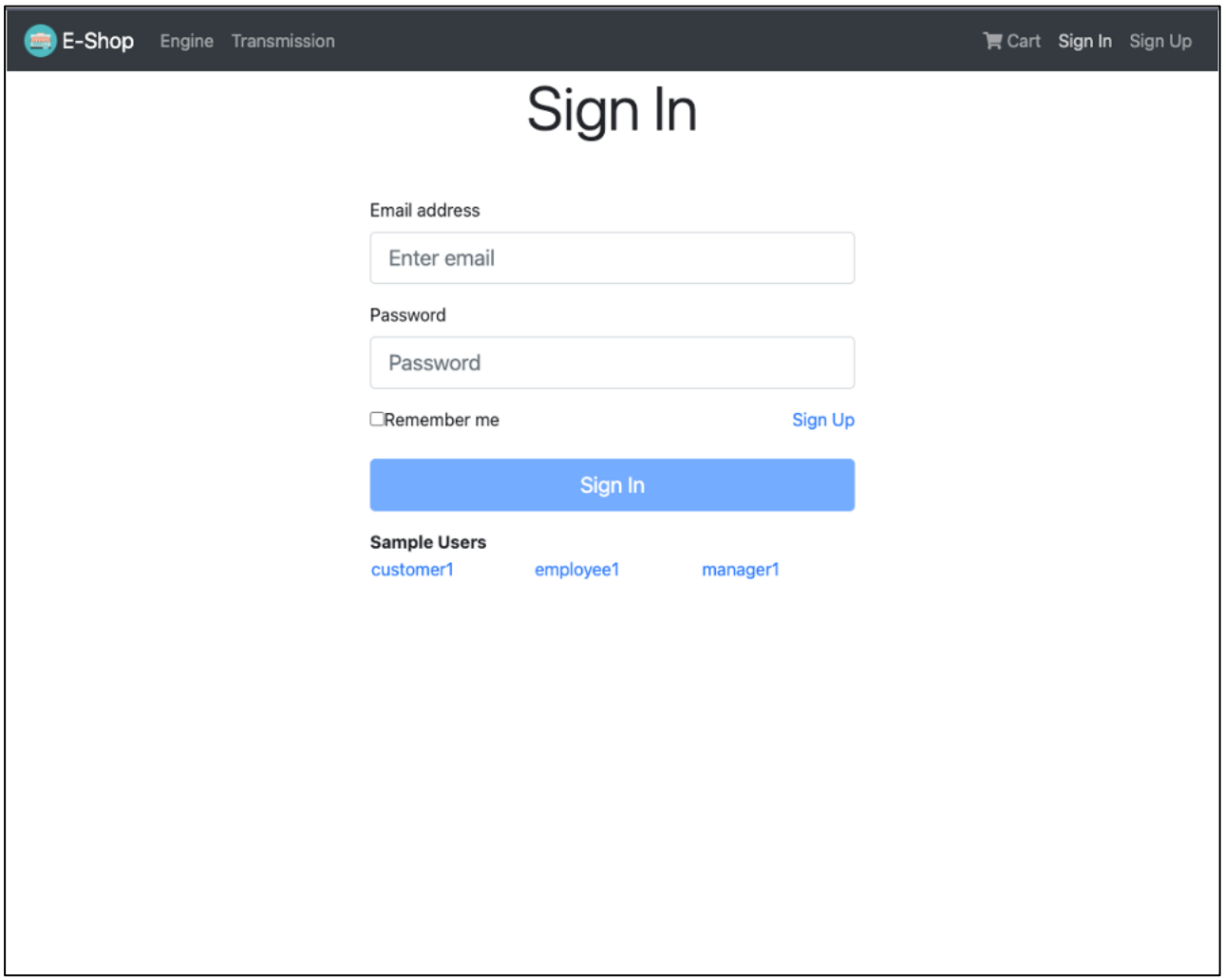

Рисунок 3.19 – Сторінка «Вхід»

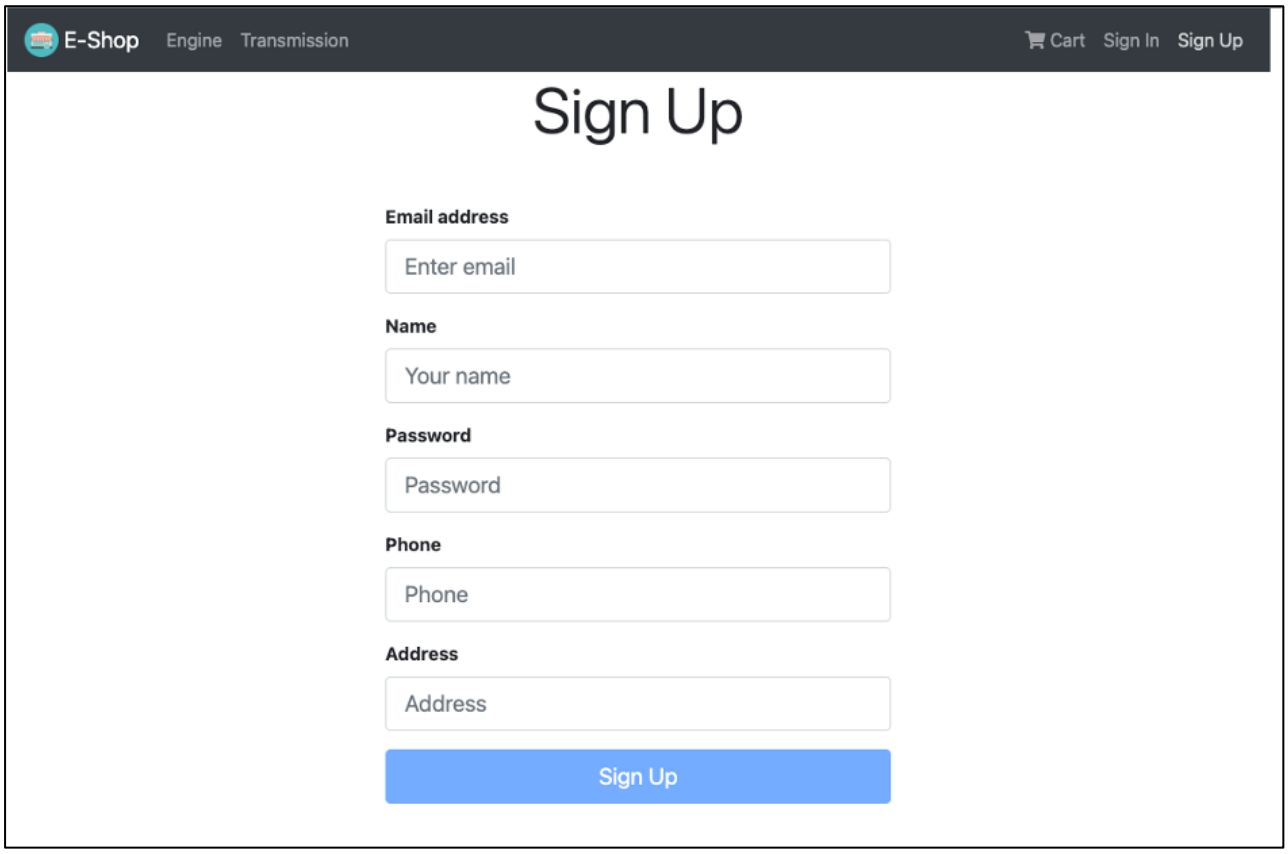

Рисунок 3.20 – Сторінка «Реєстрація»

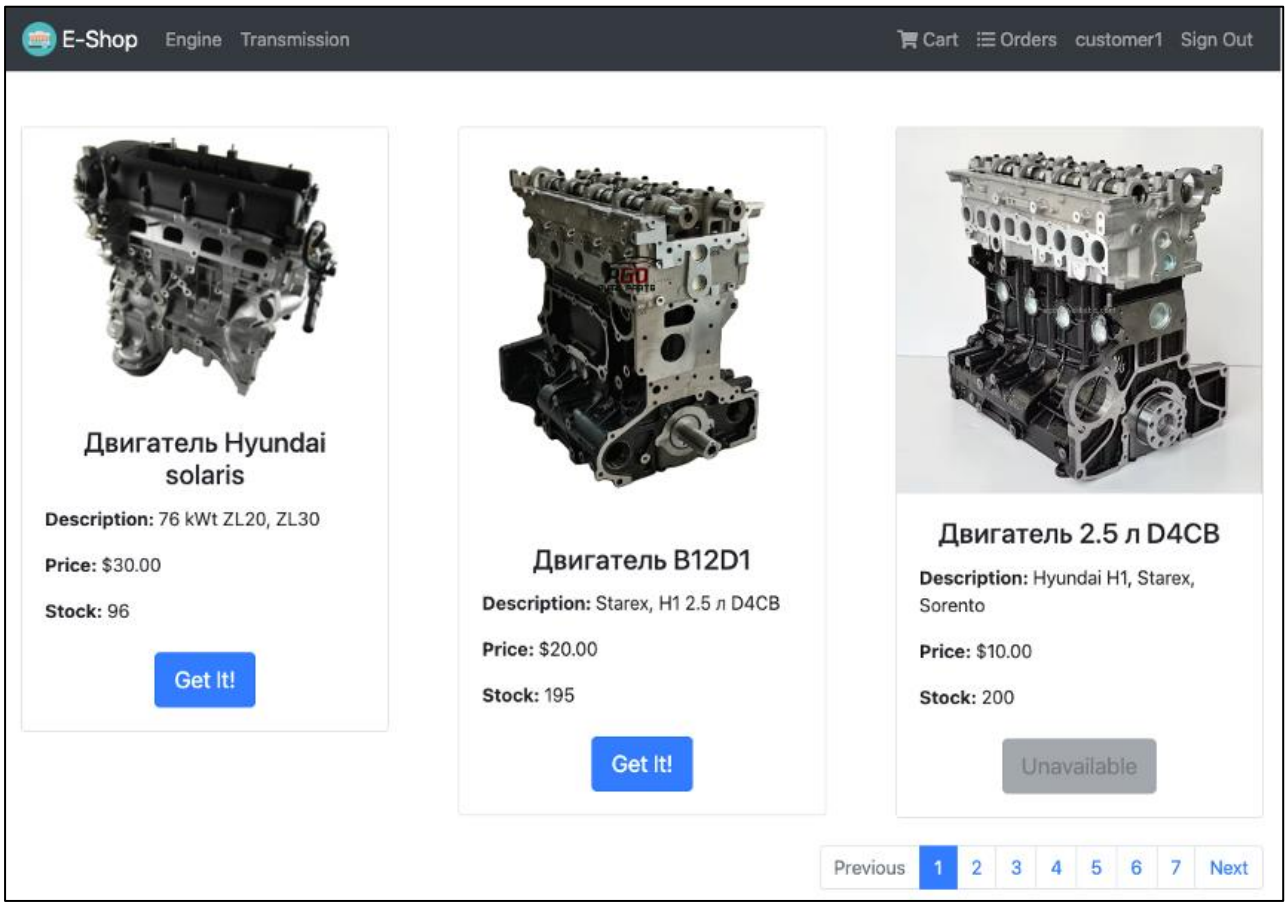

Рисунок 3.21 – Сторінка «Категорія двигуни»

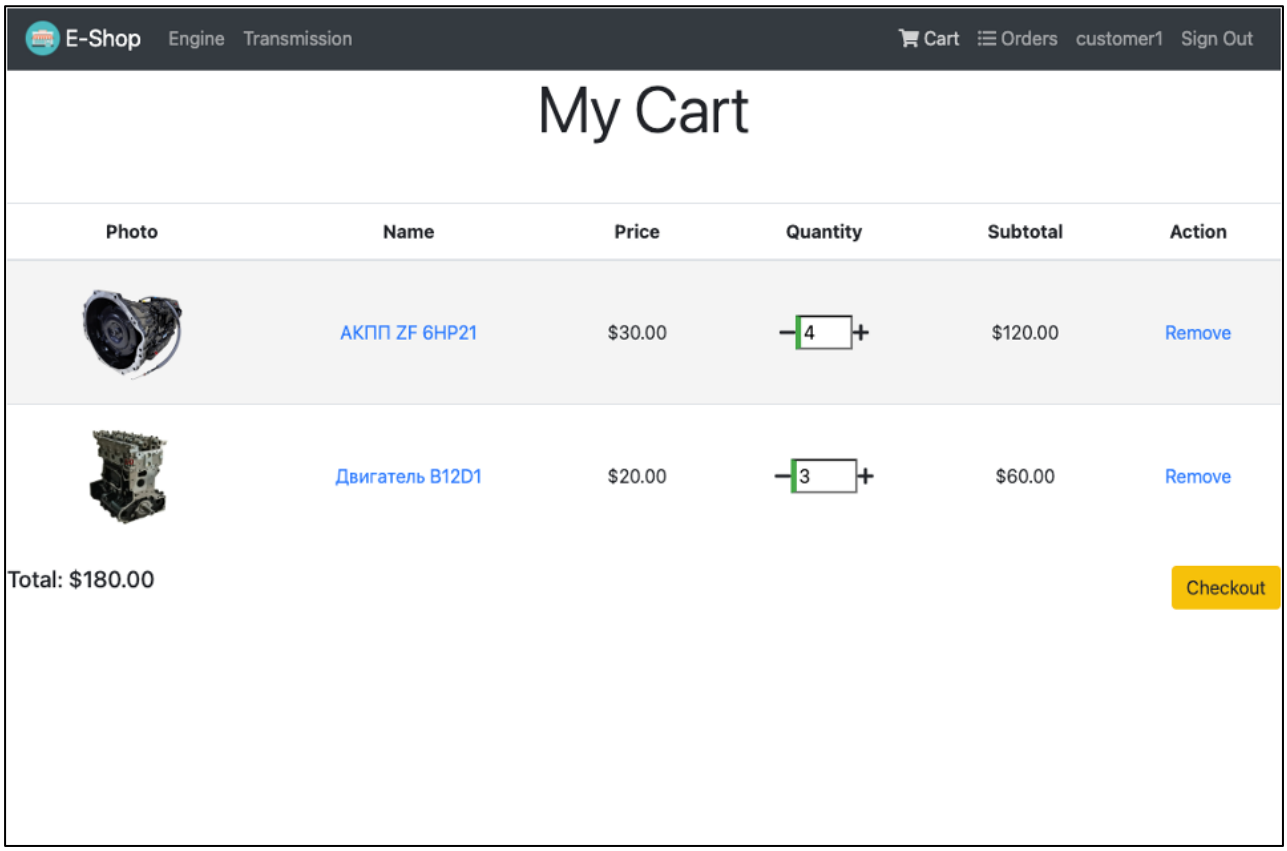

Рисунок 3.22 – Сторінка «Корзина»

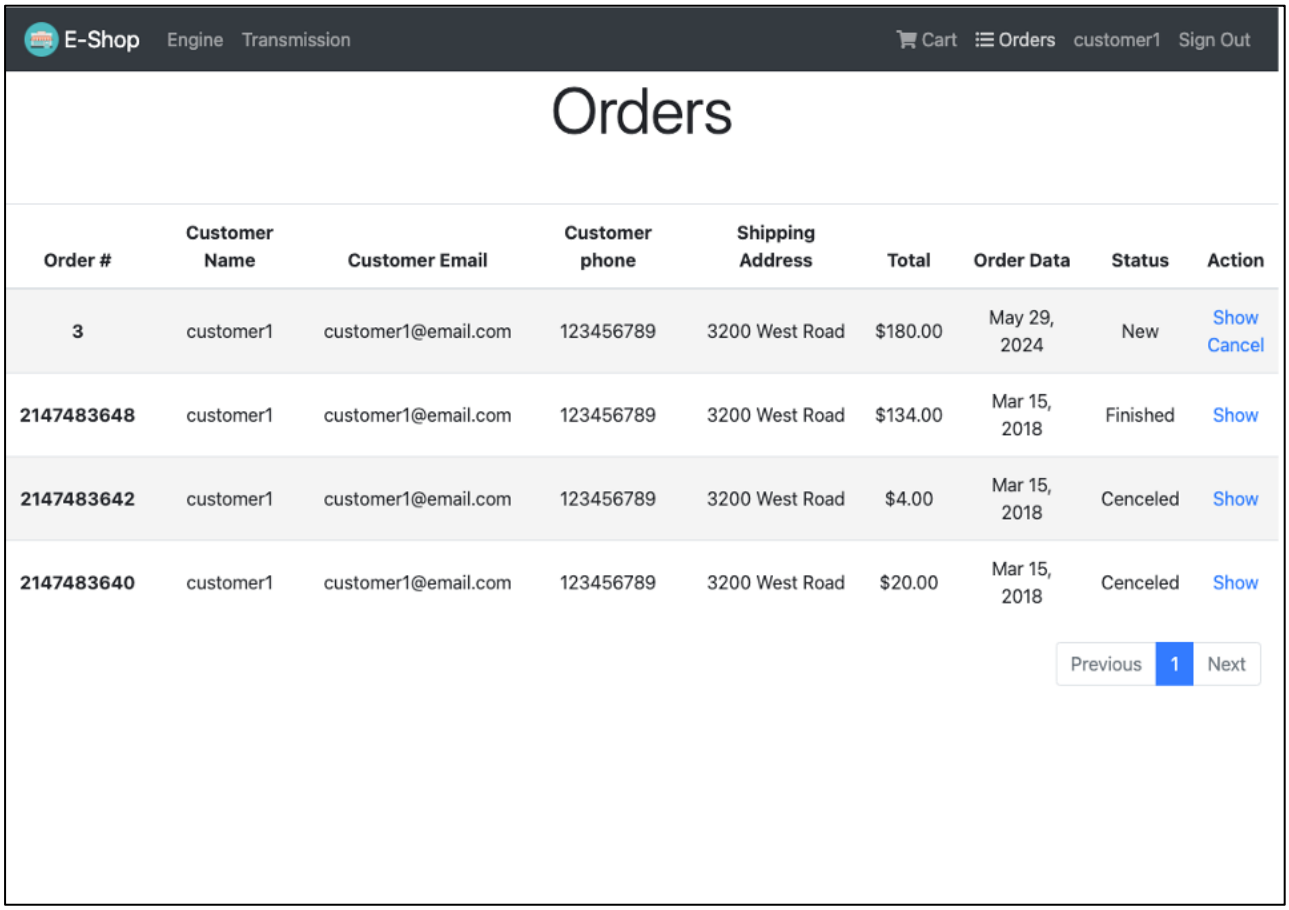

Рисунок 3.23 – Сторінка «Замовлення»

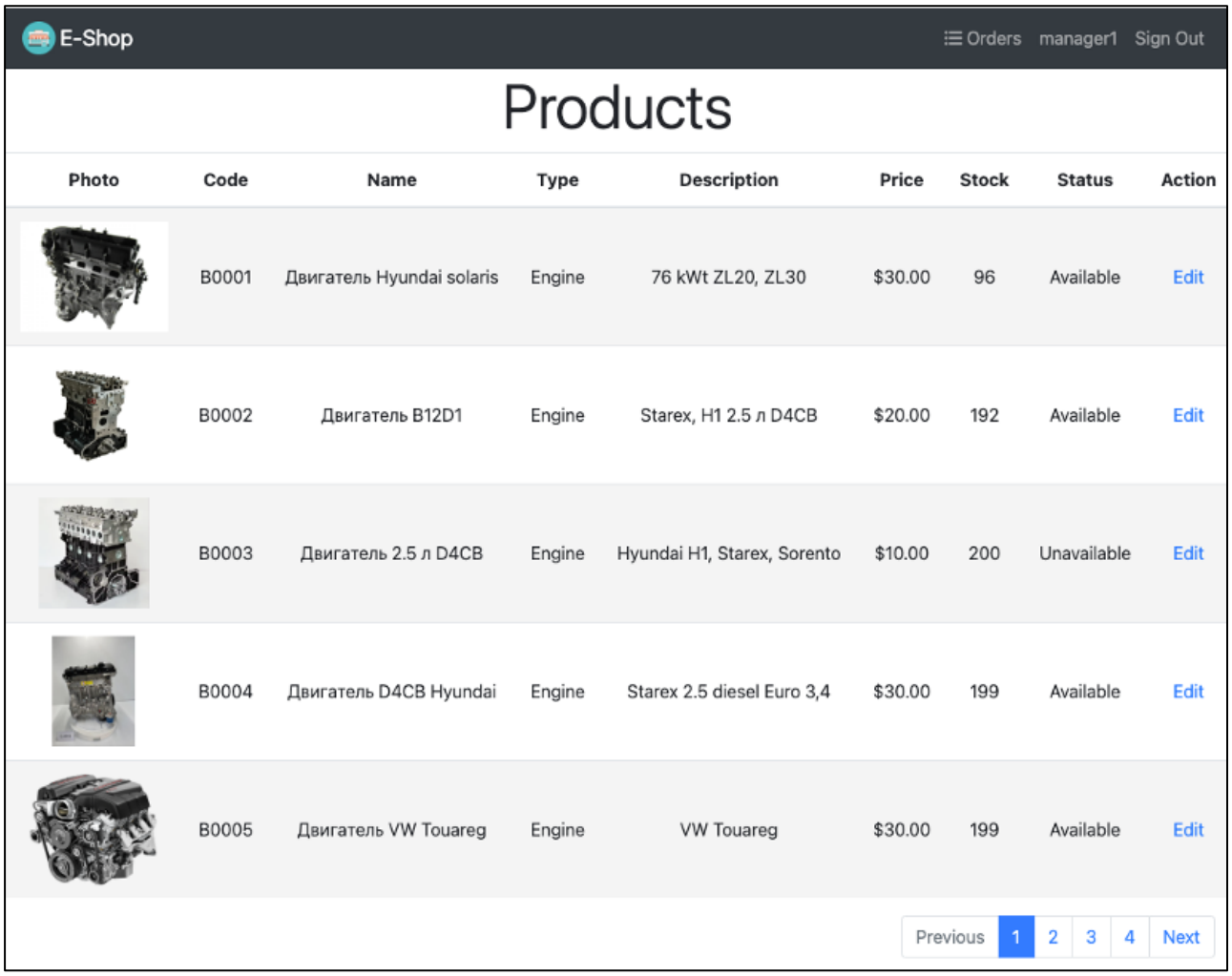

Рисунок 3.24 – Сторінка «Редагування інформації менеджера»

3.2 Висновки до розділу 3

Розробка клієнтського інтерфейсу на Angular дозволила створити сучасний та зручний додаток для інтернет-магазину автозапчастин. Використання передових технологій та інструментів забезпечило високу продуктивність, зручність використання та масштабованість додатку. Завдяки добре організованій структурі проєкту та використанню найкращих практик розробки, було створено надійний та функціональний клієнтський інтерфейс, який повністю відповідає потребам користувачів.

ВИСНОВКИ

Розробка інтернет-магазину автозапчастин, таких як двигуни та коробки передач, на основі сучасних технологій, таких як Spring, Angular та PostgreSQL, виявилася ефективним та надійним рішенням для створення функціонального та зручного онлайн-сервісу. Використання Spring Boot для серверної частини дозволило забезпечити стабільність та масштабованість додатку. Це особливо важливо для обробки великих обсягів даних та виконання складних бізнес-операцій, таких як обробка замовлень та управління товарами. Інтеграція Angular для розробки клієнтського інтерфейсу дозволила створити інтуїтивно зрозумілий та адаптивний користувацький інтерфейс, який забезпечує зручний доступ до функціоналу магазину з будь-якого пристрою.

Застосування PostgreSQL як основи для зберігання даних забезпечило високу продуктивність та надійність бази даних. Це рішення дозволяє ефективно управляти великими обсягами даних та забезпечувати швидкий доступ до них, що є критично важливим для роботи інтернет-магазину. Використання ORM (Object-Relational Mapping) інструментів, таких як Hibernate, спростило роботу з базою даних та дозволило зосередитися на бізнес-логіці додатку.

Підтримка мобільних пристроїв завдяки адаптивному дизайну Angular забезпечила зручність використання інтернет-магазину на різних платформах, що сприяє залученню ширшої аудиторії. Модульна архітектура системи дозволила легко додавати нові функціональні можливості та компоненти, а також проводити зміни в існуючих модулях без порушення роботи всього додатку. Це забезпечує гнучкість та легкість у підтримці системи, що є важливим аспектом для її подальшого розвитку та масштабування.

Автоматизація процесів, таких як управління замовленнями та оповіщення користувачів, підвищила ефективність роботи інтернет-магазину. Це дозволило зменшити витрати часу та ресурсів на ручні операції та підвищити швидкість обслуговування клієнтів. SEO-оптимізація, впровадження мета-тегів, дружніх

URL-адрес та інших елементів сприяють покращенню позицій у пошукових системах, що збільшує видимість магазину в інтернеті та залучає більше клієнтів.

Підтримка мультимовності інтернет-магазину дозволила залучати клієнтів з різних регіонів, що є важливим аспектом для розширення ринку та підвищення конкурентоспроможності. Це реалізовано шляхом використання відповідних інструментів та підходів до локалізації, що забезпечує зручний доступ до функціоналу магазину для користувачів, які розмовляють різними мовами. Безпека даних та конфіденційність користувачів також були важливими аспектами розробки. Використання сучасних методів захисту даних, таких як шифрування та аутентифікація, забезпечило захист особистої інформації клієнтів та підвищило довіру до магазину.

Загалом, розробка інтернет-магазину автозапчастин із застосуванням Spring, Angular та PostgreSQL продемонструвала високу ефективність та надійність цих технологій у створенні сучасного та функціонального онлайн-сервісу. Впровадження передових підходів до розробки програмного забезпечення, таких як модульна архітектура, автоматизація процесів та адаптивний дизайн, дозволило створити зручний та надійний інструмент для здійснення покупок в інтернеті. Це забезпечило зручність та ефективність користування для клієнтів, а також підвищило конкурентоспроможність магазину на ринку автозапчастин.

Окрім технічних аспектів розробки, важливу роль у створенні інтернет-магазину автозапчастин відігравала розробка користувацького інтерфейсу. Клієнтська частина, побудована на Angular, забезпечила високу інтерактивність та динамічність додатку. Використання цього фреймворку дозволило реалізувати SPA (Single Page Application) підхід, що значно покращило швидкість роботи та зручність користування. Застосування компонентного підходу в Angular дало можливість розробляти і тестувати окремі частини інтерфейсу незалежно одна від одної, що полегшило процес налагодження та подальшого розвитку додатку.

Важливою частиною розробки було забезпечення високої продуктивності та швидкодії додатку. Це досягалося шляхом оптимізації запитів до бази даних, використання кешування та інших підходів до підвищення ефективності роботи додатку. Використання PostgreSQL як основи для зберігання даних дозволило ефективно обробляти великі обсяги інформації, забезпечуючи швидкий доступ до неї та високу надійність.

Розробка інтернет-магазину також включала створення системи управління контентом, що дозволило адміністраторам легко додавати, змінювати та видаляти інформацію про товари, а також керувати іншими аспектами роботи магазину. Це забезпечило гнучкість та зручність в управлінні контентом, що є важливим аспектом для підтримки актуальності інформації та ефективної роботи магазину.

Інтеграція з різними API для отримання додаткової інформації про товари, зокрема технічних характеристик та відгуків користувачів, дозволила надати клієнтам вичерпну інформацію, що сприяло прийняттю обґрунтованих рішень під час покупки. Це підвищило задоволеність клієнтів та сприяло зростанню кількості замовлень.

Забезпечення високої доступності та безперервності роботи додатку було досягнуто шляхом використання хмарних сервісів та технологій контейнеризації, таких як Docker. Це дозволило забезпечити масштабованість додатку та можливість швидкого відновлення роботи у разі збоїв або інших проблем. Резервне копіювання даних та автоматичне відновлення додатка були важливими аспектами забезпечення надійності системи.

Розробка системи логування та моніторингу дозволила забезпечити своєчасне виявлення та вирішення проблем, що виникають під час роботи додатку. Це включало збір та аналіз логів, а також використання інструментів моніторингу для відстеження стану системи та її продуктивності. Такі підходи дозволили оперативно реагувати на проблеми та підтримувати високу якість обслуговування користувачів.

Одним з ключових аспектів розробки була підтримка та розвиток

інтерфейсу для мобільних пристроїв. Використання адаптивного дизайну дозволило забезпечити зручний доступ до функціоналу магазину з будь-якого пристрою, незалежно від розміру екрану. Це сприяло залученню ширшої аудиторії та підвищенню зручності користування.

Підтримка різних мов інтерфейсу була реалізована шляхом використання інструментів для локалізації, що дозволило користувачам вибирати зручну для них мову. Це забезпечило комфортне користування для клієнтів з різних країн та регіонів, підвищуючи доступність та зручність магазину.

Забезпечення захисту від DDoS атак та інших видів кіберзагроз було важливим аспектом безпеки системи. Використання відповідних інструментів та підходів дозволило забезпечити високу стійкість до таких загроз та захистити інтереси користувачів та дані магазину.

Розробка детальної документації для користувачів та адміністраторів системи забезпечила легкість та зручність в користуванні та адмініструванні додатку. Це включало створення інструкцій та керівництв, які допомагали швидко освоїти функціонал та ефективно використовувати можливості системи.

Загалом, розробка інтернет-магазину автозапчастин із застосуванням сучасних технологій та підходів продемонструвала високу ефективність та надійність створеного рішення. Використання Spring, Angular та PostgreSQL дозволило реалізувати функціональний, зручний та надійний інтернет-магазин, який відповідає сучасним вимогам та забезпечує високу якість обслуговування користувачів. Подальший розвиток та підтримка системи дозволять зберігати її актуальність та ефективність у динамічно змінюваних умовах ринку автозапчастин.

ПЕРЕЛІК ПОСИЛАНЬ

1. Rajput D. Mastering Spring Boot 2.0. Packt Publishing, 2018. 390 p.

2. Rajput D. Spring 5 Design Patterns: Master efficient application development with patterns such as proxy, singleton, the template method, and more. Packt Publishing, 2017. 369 p.

3. Bampakos A., Deeleman P. Learning Angular – Fourth Edition: A no-nonsense guide to building web applications with Angular. Packt Publishing, 2023. 426 p.

4. PostgreSQL 13.15 Documentation. URL: https://www.postgresql.org/docs/13/index.html (дата звернення: 06.03.2024).

5. Spring Framework Documentation. *Spring.io*. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення: 10.03.2024).

6. Cherny B. Programming TypeScript. O'Reilly Media, 2019. 322 p.

7. Richardson C. Microservices Patterns: With examples in Java. Manning Publications, 2018. 520 p.

8. Walls C. Spring in Action, Fifth Edition. Manning Publications, 2018. 520 p.

9. Official Spring Boot Documentation. *Spring.io*. URL: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ (дата звернення: 18.03.2024).

10. Introduction to the Angular docs. *Angular.io*. URL: https://angular.io/docs (дата звернення: 20.03.2024).

11. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.

12. Tate B., Deinum M. Spring Data. URL: https://www.manning.com/books/spring-data (дата звернення: 26.03.2024).

13. McLaughlin B. D., Pollice G., West D. Head First Object-Oriented Analysis and Design. O'Reilly Media, 2007. 634 p.