Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

здобувача Вакульчик Владислав Олексійович
(ПІБ)

академічної групи 123-21-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)

на тему «Веб-сервіс обслуговування клієнтів інтернет магазину
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Ткаченко С.М			
Спеціальної частини	доц. Ткаченко С.М			

Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"__" _____ 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

здобувача Вакульчик В.О. **академічної групи** 123-21-1
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Веб-сервіс обслуговування клієнтів інтернет-магазину зоотоварів»

затверджену наказом ректора НТУ «Дніпровська політехніка» від 5.05.2025 336-с

Розділ	Зміст	Термін виконання
Розробка веб-сервісу для обробки клієнтських запитів	Сформулювати найменування й призначення веб-сервісу для обробки клієнтських засобів, висунути технічні вимоги до нього. Спроєктувати веб-сервіс. Розробити апаратну частину згідно вимогам.	12.02.2025
Розроба клієнтського сервісу для взаємодії з користувачами через мессенджер телеграмм	Сформулювати найменування й призначення клієнтського сервісу для взаємодії з користувачами через месенджер Telegram, висунути технічні вимоги до нього. Розробити апаратну частину. Організувати взаємодію між суміжними системами.	27.03.2025
Розробка веб-сервісу інтерактивного користувацького інтерфейсу	Сформувати вимоги до програмного забезпечення. Розробити програмне забезпечення для керування апаратом штучної вентиляції легень	30.04.2025

Завдання видано _____ доц. Ткаченко С.М.
(підпис керівника) (прізвище, ініціали)

Дата видачі 25.02.2025

Дата подання до екзаменаційної комісії 17.06.2025

Прийнято до виконання _____ Вакульчик В.О.

РЕФЕРАТ

Пояснювальна записка: 77 с., 12 рис., 7 табл., 1 дод., 29 джерел.

JAVA, TYPESCRIPT, ВЕБ-СЕРВІС, ІНТЕРАКТИВНИЙ ІНТЕРФЕЙС, БАЗА ДАНИХ

Об’єкт розробки: веб-сервіс обслуговування клієнтів інтернет магазину.

Мета: створення веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів. Має буде створено:

- веб сервіс для обробки клієнтських запитів;
- клієнтський сервіс для взаємодії з користувачами через месенджер Telegram;
- веб-сервіс інтерактивного користувацького інтерфейсу.

Ця система потенційно може стати основною платформою торгівлі для середнього бізнесу.

Веб-сервіс розроблений за допомогою мов програмування Java, TypeScript, та баз даних на PostgreSQL. результати тестування наведені у пояснювальній записці та у додатках.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

HTTP	– англ. Hypertext Transfer Protocol, протокол, що використовується для обміну даними між клієнтами та серверами в Інтернеті, особливо для веб-перегляду;
HTTPS	– англ. Hypertext Transfer Protocol Secure, безпечний протокол передачі даних через Інтернет, який забезпечує шифрування інформації між клієнтом та сервером;
REST	– англ. Representational State Transfer, підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів;
CRM	– англ. Customer Relationship Management, система, яка дозволяє компаніям управляти взаємодіями з клієнтами та потенційними клієнтами;
FTP	– англ. File Transfer Protocol, стандартний мережевий протокол прикладного рівня, призначений для пересилання файлів між клієнтом та сервером в комп'ютерній мережі;
SPA	– англ. Single-page Application, тип застосунків, який відкривається в браузері й при цьому не потребує постійного перезавантаження сторінки;
DOM	– англ. Documet Object Model, специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами;
SSR	– англ. Server-Side Rendering, техніка, яка використовується для генерації HTML-контенту на сервері та надсилання попередньо створених сторінок клієнтам;
SEO	– англ. Search Engine Optimization, процес оптимізації сайту для того, щоб він займав вищі позиції в результатах пошуку пошукових систем;
HTML	– англ. Hypertext Markup Language, стандартизована мова розмітки документів для перегляду веб-сторінок у браузері;

CLI – англ. Command Line Interface, текстовий інтерфейс, який дозволяє користувачу взаємодіяти з комп'ютером за допомогою текстових команд;

API – англ. Appliction Programming Interface, спосіб взаємодії комп'ютерних програм між собою

CMS – англ. Content Management System, система керування вмістом, або програмне забезпечення для організації та управління веб-сайтами чи іншими інформаційними ресурсами;

PHP – англ. Personal Home Page, скриптова мова програмуваня;

ORM – англ. Object-Relational Mapping, технологія програмування, яка забезпечує взаємодію між об'єктно-орієнтованими мовами програмування та реляційними базами даних;

CSV – англ. Comma-Separated Values, текстовий формат файлу, який використовує кому як роздільник між значеннями у таблиці;

CRUD – англ. «create, read, update, delete», 4 основні функції управління даними «створення, читання, оновлення, видалення»;

SQL – англ. Structured Query Language, стандартна мова запитів та програмування, яка використовується для роботи з реляційними базами даних;

CSRF – англ. Cross-Site Request Forgety, тип веб-атаки, що призводить до виконання певних дій від імені користувача на веб-сторінці, де останній аутентифікований;

XCC – англ. Cross-Site Scripting, тип вразливості інтерактивних інформаційних систем у вебі;

OWASP – англ. Open Worldwide Application Project, відкритий проєкт по забезпеченню безпеки веб-додатків;

LTS – англ. Long-Term Support, довгострокова підтримка програмного забезпечення;

NPM – англ. Node Package Manager, інструмент, що використовується для управління

	пакетами (модулями) мови програмування JavaScript;
PM2	– менеджер процесів для середовища JavaScript Node.js;
CPU	– англ. Central Processing Unit, центральний процесор;
JVM	– англ. Java Virtual Machine, віртуальна машина для використання байт-коду Java;
VPS	– англ. Virtual Private Server, послуга, в рамках якої, клієнтові надають віртуальний сервер;
LINQ	– англ. Language-Integrated Query потужний інструмент, який спрощує роботу з колекціями та даними;
MVCC	– англ. Multiversion Concurrency Control, метод керування паралельним доступом до бази даних;
JSON	– англ. JavaScrip Object Notation, текстовий формат обміну даних між комп’ютерами;
JSONB	– двійковий формат збереження даних JSON;
SSH	– англ. Secure Shell, мережевий протокол прикладного рівня;
SSL	– англ. Secure Sockets Layer, криптографічний протокол, що забезпечує безпечне з’єднання між клієнтом та сервером;
БД	– база даних;
СУБД	– система управління базами даних;
ПЗ	– програмне забезпечення.

ВСТУП

Коротка характеристика стану проблеми. В сучасному світі сфера інтернет-торгівлі має економічну, та стратегічну важливість для бізнесу. Невід'ємною частиною сучасного успішного підприємства, являється продвинута інтернет-система обслуговування клієнтів. Згідно з прогнозами аналітиків, обсяг світового ринку e-commerce, що вже вимірюється трильйонами доларів США, продовжить своє зростання і, за очікуваннями, перевищить позначку $6.8 трлн у 2025 році, з перспективою досягнення $8 трлн до 2027 року (згідно з прогнозами SOAX та SellersCommerce на 2025-2027 роки). Ця тенденція свідчить про фундаментальний зсув у споживацьких звичках та необхідність для бізнесу адаптуватися до нових реалій.

Також потрібно знати, що попри складні економічні та соціальні умови, вітчизняний ринок електронної комерції показує неабияку стійкість та динаміку зростання. За підсумками 2024 року, українці витратили на придбання товарів та послуг в інтернеті вражаючу суму – 239 мільярдів гривень, що на 25% перевищує показники попереднього року (згідно з даними Promodo та dev.ua за 2024 рік). Це свідчить про зростаючу довіру споживачів до онлайн-покупок та адаптацію бізнесу до цифрових каналів продажів. Через це, все більше компаній охоче вкладають гроші в розробку інтернет магазинів.

Мета та завдання роботи створення веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів. Має буде створено:

- веб сервіс для обробки клієнтських запитів;
- клієнтський сервіс для взаємодії з користувачами через месенджер Telegram;
- веб-сервіс інтерактивного користувацького інтерфейсу.

Ця система потенційно може стати основною платформою торгівлі для середнього бізнесу.

Актуальність вирішення полягає в створенні відказостійкої масштабуїмої веб-системи, за допомогою сучасних та новітніх технологій. Використання Java в комбінації з Spring Framework, дозволяють створювати

високонавантажені Back-end сервери, для обробки запитів клієнтів не тільки з ближчих областей, а й з усієї країни, з перспективою розвитку бізнесу на міжнародній арені. Для побудови системи комунікацій між різними компонентами - буде використана архітектура REST, по протоколу HTTP та HTTPS, що гарантує універсальність та гарну масштабованість кожного з структурних компонентів системи. Front-end буде розроблено на технології Angular, яка вже багато років являється конкурентним варіантом в світі веб-розробки. Використання PostgreSQL зумовлене швидкістю роботи, та великим розповсюдженням на ринку баз даних. Така структура майбутньої системи являється максимально актуальною, яка може повністю задовільнити всі потреби замовника та користувачів.

З концептуальної та функціональної точки зору система повинна відповідати усім вимогам до сучасного веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів. Він повинен реалізовувати та відповідати на наступні виклики, до сучасної ритейлингової системи:

1. Наявність актуальної інформації (користувач повинен мати інформацію про наявні товари в магазині);
2. Швидкість та зручність оформлення замовлень (при оформленні замовлення, користувач повинен мати можливість оплати прямо на сайті, та організувати доставку товарів до себе);
3. Фільтрування підходящої продукції (пошук в великих інтернет магазинах, серед тисяц найменувань товарів повинен реалізовуватися за допомогою системи фільтрації, та пошуку по ключовим словам, облегшуючи та прискорюючи пошук підходящих позицій);
4. Достатній розгорнутий опис товару (під час аналізу користувач повинен мати змогу отримати повноцінний опис товару, яким торгує магазин).
5. Система фідбеку (користувачі повинні мати доступ до можливості публікації своїх коментарів).

Такий високий рівень функціональності системи зумовлений високою конкуренцією. В умовах високої конкуренції в e-commerce ключовим

фактором диференціації та побудови довгострокових відносин зі споживачами стає якість обслуговування.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Роль і значення об’єкту професійної діяльності, а також актуальність вирішуваної задачі для об’єкту впровадження у сфері застосування

В контексті роздрібної торгівлі, при збільшенні бізнесу до розміру від 10 до 20 магазинів, стають актуальними такі питання, як організація централізованої бази даних, де фіксуються транзакції. Зачасту це реалізується під якусь CRM систему, на кшталт Poster, який дозволяє комфортно працювати з БД навіть некваліфікованому працівнику. Також, велика сітка магазинів активно починає шукати нові ринки збуту, а також можливість збільшити свої обсяги торгівлі.

В контексті конкурентної боротьби, інтернет-магазин розширює географію продажів за межі фізичної присутності магазинів, дозволяючи охопити нових клієнтів та ефективніше конкурувати з суто онлайн-гравцями. Він також надає потужний інструмент для збору даних про попит, що допомагає формувати більш точні замовлення для офлайн-магазинів. Можливість онлайн-замовлення з опцією самовивозу ("click and collect") з найближчого магазину підвищує зручність для клієнтів та інтегрує онлайн і офлайн досвід. Крім того, онлайн-платформа може стати централізованим джерелом детальної інформації про товари, частково компенсуючи потребу в глибокій експертизі кожного продавця у фізичних магазинах та забезпечуючи єдиний стандарт інформаційного супроводу. Таким чином, інтернет-магазин не просто додає новий канал збуту, а й допомагає вирішити низку операційних та конкурентних проблем, перетворюючи мережу на більш гнучку та стійку омніканальну структуру.

Омнікальність - це інтеграція різних каналів взаємодії з клієнтом (онлайн та офлайн) в єдину, злагоджену систему для забезпечення послідовного та цілісного досвіду. Це означає, що клієнт отримує узгоджену

інформацію та обслуговування через будь-який канал, будь то сайт, соціальні мережі, магазин або колл-центр.

Омнікальність бізнесу досягається чітким налаштуванням великої системи, яка буде поєднувати між собою користувачів, працівників, та адміністрацію. При розробці такої системи дуже важливо звернути увагу на правильне налаштування баз даних, в яких буде зберігатися вся потрібна інформація. Бази даних повинні зберігати інформацію без дублікатів, чітко структуровано, з можливістю максимально швидкого пошуку.

Створення такої веб-системи може збільшити прибуток, та рентабельність бізнесу, налагодити новий рівень комунікації з користувачами, дати більш чітку аналітику ринкової ситуації, а також збільшити медійність продукту, надавши змогу дізнатися про магазин, навіть користувачам з областей країни, де ще не було відкрито фізичну точку збуту продукції.

1.2 Характеристика, структура, особливості, умови роботи об’єкту впровадження з розкриттям недоліків і проблем, які має вирішити об’єкт професійної діяльності

Сітка фізичних магазинів являє собою етап розвитку бізнес-діяльності коли власники потребують розширення ринку збуту, та розширення географії покупців. Для цього, вони починають збільшувати кількість своїх точок продажів, відкриваючі нові магазини

Умови роботи сітки фізичних магазинів характеризуються високим рівнем конкуренції, та швидкозмінними трендами торгівлі.

Структура такого виду бізнесу полягає в створенні напів-автономних робочих магазинів, кожен з яких може вести свій облік, та навіть мати локального адміністратора, який керує працівниками всередині, але підпорядковується власнику сітки магазинів.

Проблематика такого підходу заключається в наступних пунктах:

- кожна окрема точка збуту має свої обмеження щодо географії охоплених покупців. В умовах великої віддаленості покупці втрачають бажання приходити до магазину;

- відкриття нової точки збуту вимагає аренди приміщення в підходящому для продажів приміщенні, яке повино розташовуватися в зручному для покупців місці, що підвищує собівартість аренди;
- відсутність можливості продемонструвати в повному обсязі спект товарів в магазині, та їх собівартість;
- мала степінь комунікації з користувачами;
- втрата можливостей CEO оптимізації та зниження можливостей в рекламі;
- відсутність візитівки в інтернет-просторі;

Розгалужена сітка магазинів, яка планує розвиватися, та займатися торгівлею в веб-просторі, для успішної та комфортної роботи повинна задовольняти даним потребам: актуальність інформації для кожного вузла, можливість отримання сповіщень, відсутність аномалій баз даних, можливість швидкого та простого процесу підготовки нового працівника.

Створення централізованого веб-сервісу, з адмін панеллю для керування наповненням сайту - найкращий вибір для середнього та великого бізнесу. Це зможе вирішити наступні проблеми:

- географічно веб-сервіс інтернет-магазину не обмежений по степені свого впливу;
- веб-сервіс інтернет-магазину має значну перевагу в собівартості утримання та відкриття;
- відкриває нові можливості для проведення рекламних кампаній;
- надає можливість користувачам в будь який час отримати доступ до списку актуальних товарів;
- аренда приміщень тільки для працівників;

Далі на Рис.1.1 буде представлена діаграма взаємодії користувача та об’єкта впровадження після додавання об’єкту професійної діяльності.

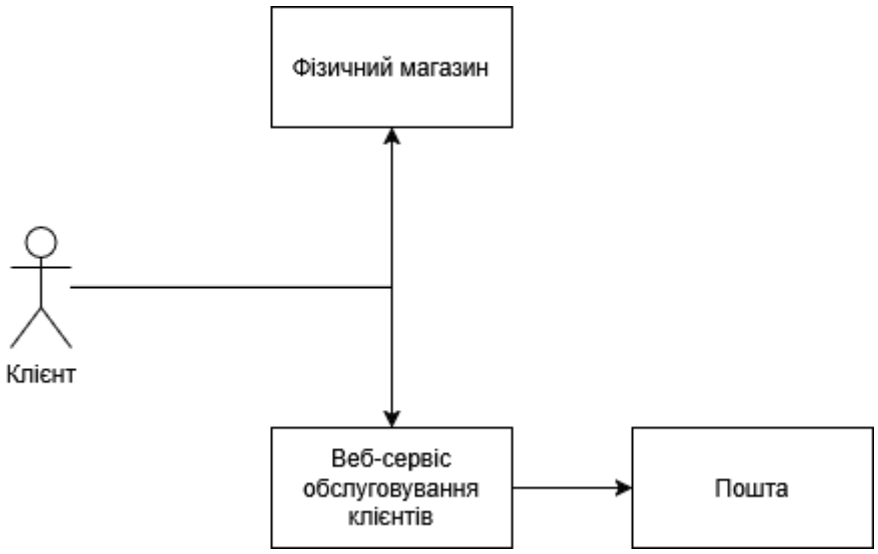


Рисунок 1.1 – Взаємодія користувача з об’єктом впровадження після додавання об’єкту професійної діяльності

1.3 Огляд існуючих аналогів КС, технологій, архітектур та програмних рішень

Спираючись на архітектуру запланованої системи ми повинні розглянути альтернативні технології для різних частин програми. Ми повинні проаналізувати конкурентні технології та впевнитися в актуальності обраного підходу.

Спершу потрібно розглянути технології, які виконують роль створення і підтримки візуального інтерфейсу користувача. Ці технології знаходяться в сектор Front-end і потрібні нам для реалізації нашої системи. Для розробки цієї частини було обрано Angular, тому далі будуть описані аналоги даної технології.

Щодо технології React, розробленої компанією Facebook, варто зазначити її домінуючі позиції на ринку. React є JavaScript-бібліотекою для створення користувацьких інтерфейсів. Її ключовими перевагами є використання віртуального DOM, що забезпечує високу продуктивність та швидке оновлення інтерфейсу, а також компонентно-орієнтований підхід, який сприяє модульній розробці та повторному використанню коду. Велика та активна спільнота, багата екосистема готових рішень та інструментів, а також відносно нижчий поріг входження для розробників, знайомих з JavaScript,

роблять React привабливим вибором для багатьох проектів. Гнучкість бібліотеки дозволяє команді самостійно обирати інструменти для маршрутизації, управління станом та інших аспектів розробки, що може бути перевагою для специфічних або менших за обсягом завдань.

Однак, для запланованої системи, що орієнтована на середній бізнес з перспективами зростання та має такі вимоги, як складний інтерфейс користувача, висока інтерактивність, робота в реальному часі, SEO-оптимізація, швидкість завантаження та легкість масштабування, підхід React може виявити певні слабкі сторони порівняно з Angular. По-перше, згадана гнучкість React, що вимагає інтеграції численних сторонніх бібліотек для реалізації повноцінного функціоналу (наприклад, для маршрутизації, управління глобальним станом, валідації форм), може призвести до ускладнення конфігурації проекту та його підтримки, особливо при масштабуванні та зростанні команди. Для забезпечення ефективної SEO-оптимізації в React-додатках часто потрібні додаткові зусилля та інструменти, такі як рендеринг на стороні сервера (SSR) за допомогою фреймворків типу Next.js, що додає ще один шар складності. Хоча команда має досвід, Angular пропонує більш комплексну та структуровану платформу "з коробки", що включає TypeScript для підвищення надійності коду, вбудовані рішення для маршрутизації, HTTP-запитів, а також потужні інструменти на кшталт RxJS для ефективної роботи в реальному часі та управління асинхронними потоками даних. Така уніфікація може бути більш вигідною для розробки складного інтерфейсу та забезпечення довгострокової підтримки системи з перспективами зростання, зменшуючи "різнорідність" технологічного стеку.

Vue.js є прогресивним фреймворком для побудови інтерфейсів користувача, що поєднує в собі простоту бібліотеки з потужністю повноцінного фреймворка. Його реактивна система на основі спостережуваних властивостей робить оновлення DOM максимально ефективним, а шаблонний синтаксис, подібний до HTML, полегшує розробку складних форм та відображення даних зі змінним станом. Завдяки легкій вазі базового ядра та модульній архітектурі можна поступово додавати необхідні

пакети — Vue Router для маршрутизації, Vuex (або Pinia) для менеджменту стану, а також офіційні рішення для інтернаціоналізації та роботи з формами. CLI-інструмент Vue CLI або Vite забезпечують швидке створення проєкту зі стандартною структурою файлів, шаблонами компонентів та інтеграцією з TypeScript за бажанням, що підходить для інтернет-магазину з REST API та гібридними термінами розробки. Завдяки низькому порогу входження і добре документованим API Vue підходить як для невеликих команд, так і для проєктів середнього масштабу, де важливо швидко налаштувати основу та поступово вводити складніші патерни розробки.

Проте в порівнянні з Angular, Vue поступається за кількома ключовими аспектами, критичними для довгострокових і масштабованих проєктів. По-перше, хоча Vue пропонує офіційні рішення, вони не є частиною єдиного ядра, і команда має окремо приймати рішення щодо версій та сумісності між Vue Router, Vuex/Pinia та іншими плагінами. Angular натомість уже «із коробки» пропонує скоординовану екосистему модулів, сервісів та інструментів, що забезпечує єдиний підхід до Reactive Forms, HttpClient, AOT-компіляції та строгого TypeScript на всіх рівнях. По-друге, жорстка структуризація файлів і конвенції Angular CLI гарантують єдиний стиль коду та полегшують підтримку великими командами, чого може бракувати у Vue, де організація директорій і архітектурні патерни більш гнучкі, але менш стандартизовані. Нарешті, корпоративна підтримка та прийняття Angular у великих enterprise-проєктах зазвичай вища, оскільки фреймворк надає комплексні рішення для безпечного впровадження та масштабування, а сувора типізація на базі TypeScript знижує кількість помилок у рантаймі без потреби додаткових налаштувань, що в умовах довгострокового обслуговування інтернет-магазину є вагомим аргументом на користь Angular.

WordPress – це найпопулярніша CMS у світі, що дає змогу швидко створювати та запускати вебсайти без глибоких навичок програмування. Завдяки гігантській екосистемі тем, плагінів і хостингових рішень, розробник може за лічені години налаштувати інтернет-магазин із базовим каталогом товарів, системою оплат і навіть модулем розсилки. Інтуїтивна панель

адміністратора, вбудована підтримка SEO та велика спільнота користувачів забезпечують низьку вартість впровадження і швидке вирішення технічних питань. WordPress безкоштовний, а комерційні плагіни пропонують готові рішення для інтеграції з платіжними системами та популярними маркетплейсами. Для невеликих і середніх проєктів, які потребують мінімальної кастомізації і де критична швидкість виходу на ринок, ця CMS залишається незамінним інструментом.

Проте в порівнянні з Angular WordPress виявляє суттєві обмеження при побудові складних, добре структурованих SPA-інтерфейсів із глибокою інтеграцією з REST API. По-перше, архітектура WordPress побудована на PHP і серверних рендерах, що призводить до додаткових затримок і меншої продуктивності під час роботи з великим масивом даних та складними фільтрами товарів. По-друге, кастомізація front-end у WordPress часто потребує безладу із змішаних тем і плагінів, що ускладнює підтримку єдиного стилю коду та уніфікованої архітектури, особливо коли команда звикла до чітких модулів і сервісів. Angular же пропонує «із коробки» готовий інструментарій для побудови reactive forms, маршрутизації, HTTP-клієнта та сильно-типізованого середовища на базі TypeScript, що спрощує розробку складних SPA-магазинів і гарантує високу якість коду та передбачувану продуктивність при масшабуванні проєкту.

Надалі буде розглянута частина Backend серверної частини. Ключовими факторами в цій розробці є стабільність системи, перспектива її розвитку та масштабування, швидкість виконання операцій, можливість витримувати великі навантаження, та можливість створення високої степіні захисту даних користувача. Мій вибір падає на Backend технологію: Java Spring.

Python із фреймворком Django пропонує стрімкий шлях до реалізації повнофункціонального бекенду завдяки своєму «батарейкам включено» підходу: вбудований ORM спрощує роботу з реляційними БД (PostgreSQL) і форматом CSV завдяки потужним механізмам міграцій та серіалізації даних, а готова панель адміністратора дозволяє миттєво отримати інтерфейс для CRUD-операцій. Django підтримує чітку структуру проєкту з розділенням

шарів «моделі–відображення–контролер», що полегшує підтримку та розширення коду в умовах гібридної архітектури (моноліт із виділеними сервісами). Висока продуктивність при типових CRUD-запитах до PostgreSQL досягається за рахунок оптимізованого SQL, що генерується ORM, а можливість кешування на рівні ORM і шаблонів допомагає витримувати суттєві навантаження навіть без складних DevOps-рішень. Керування безпекою реалізовано через вбудовані засоби захисту від CSRF, XSS та SQL-ін’єкцій, а велика спільнота й регулярні релізи гарантують своєчасні патчі та оновлення.

Проте в порівнянні з Java Spring + JPA Django має декілька обмежень для проєкту з максимальними запитами до продуктивності, безпеки та довгострокової підтримки. По-перше, відсутність статичної типізації збільшує ризик помилок у рантаймі й ускладнює масштабування великого кодового базису без суворих засобів статичного аналізу — тоді як Java з JPA спирається на компіляцію, що відсіює помилки на ранніх стадіях. По-друге, для тонкого налаштування виконання SQL-запитів Django ORM не завжди дозволяє досягти оптимального плану запиту, тоді як Hibernate дає більше контролю над fetch-стратегіями й кешуванням другого рівня. По-третє, у проєктах із суворими вимогами до enterprise-безпеки та відповідності стандартам OWASP набір вбудованих механізмів у Spring Security виявляється ширшим і глибшим, забезпечуючи комплексну політику аутентифікації, авторизації та аудиту. Нарешті, довгострокова підтримка й гарантія LTS-релізів у екосистемі Java дають упевненість у стабільності й масштабованості рішення на роки вперед, тоді як цикл релізів Django може вимагати частішого оновлення та міграцій під час еволюції платформи.

JavaScript із Node.js відкриває фронтендерам можливість працювати в єдиному мовному середовищі як на клієнті, так і на сервері, що значно спрощує передачу знань і знижує бар’єр входження для команд із різнорівневим досвідом. Неблокуюча, подієво-орієнтована модель вводу-виводу дозволяє одночасно обслуговувати тисячі запитів без потреби у створенні окремого потоку під кожне підключення, що особливо корисно для

інтернет-магазину з великим числом одночасних переглядів каталогу, запитів до кошика та оформлення замовлень. Екосистема NPM налічує понад 1 млн пакетів, серед яких можна знайти рішення для ORM (Sequelize, TypeORM), валідації даних, роботи з CSV і інтеграції з PostgreSQL, а фреймворки Express і NestJS пропонують структуровані підходи до побудови REST API, модульного коду та підтримки TypeScript. Використання кластеризації через модуль cluster або зовнішні інструменти на кшталт PM2 дозволяє масштабувати Node.js-додаток в багатоядерних середовищах стандартного VPS, а підтримка WebSocket (через Socket.IO, WS) нативно інтегрується з тим самим середовищем без додаткової інфраструктури. Завдяки легкості налаштування, мінімальній конфігурації та можливості "гарячої" перезагрузки сервера розробка MVP відбувається швидко, а велика спільнота забезпечує швидкі оновлення і велику кількість прикладів реалізацій. Нарешті, активний розвиток TypeScript у світі Node.js дозволяє поступово вводити статичну типізацію, зберігаючи гнучкість динамічної мови на ранніх етапах проєкту.

Проте у порівнянні з Java Spring + JPA платформа Node.js має декілька обмежень, що стають критичними при високих навантаженнях, жорстких вимогах безпеки та довгостроковій підтримці. По-перше, хоча TypeScript значно знижує ризики, сама природа JavaScript як динамічної мови породжує вищий ризик помилок у рантаймі та потребує додаткових шарів статичного аналізу (ESLint, TSLint), у той час як Java забезпечує виявлення багатьох помилок ще на етапі компіляції. По-друге, однопоточна модель Node.js добре справляється з I/O-інтенсивними завданнями, але створює вузькі місця під час виконання CPU-відповідальних обчислень або складних SQL-запитів без зовнішніх інструментів розподілу навантаження; Spring Boot із Hibernate натомість використовує багатопотоковість JVM і кеш другого рівня для оптимізації доступу до реляційної БД. По-третє, екосистема Node.js характеризується високим рівнем фрагментації пакетів і частими змінами в API, що може призвести до несумісностей та незрозумілості залежностей у великому проєкті; у Spring всі основні модулі (Spring Data JPA, Spring Security, Spring MVC) розробляються синхронно і ретельно тестуються командою

Spring, забезпечуючи єдину узгоджену платформу. Крім того, вбудовані засоби Spring Security надають комплексні механізми аутентифікації, авторизації, захисту від CSRF та аудит-відповідність стандартам OWASP без потреби в сторонніх пакетах, а довгострокова підтримка LTS-релізів Java гарантує стабільність і прогнозовані оновлення для підприємств із високими вимогами до безпеки й надійності. В підсумку, для проєктів із максимальними вимогами до продуктивності, безпеки та підтримки на роки вперед Java Spring + JPA залишається більш передбачуваним і масштабованим вибором.

PHP із фреймворком Laravel надає розробникам «батарейками включено» середовище для швидкого створення бекенд-додатків завдяки вбудованим механізмам ORM (Eloquent), системі міграцій, мок-інжекції та багатофункціональному CLI (Artisan). Використання Eloquent дозволяє інтуїтивно описувати взаємозв’язки між моделями та працювати з PostgreSQL і CSV-файлами через зручні методи серіалізації й імпорту/експорту даних. Очистка кешу, побудова черг та робота з job-системою через Redis чи database driver спрощують розподіл навантаження і реалізацію асинхронних задач навіть на звичайному VPS без складної інфраструктури. Система пакетів Composer і модульна архітектура дозволяють швидко підключати сторонні рішення для аутентифікації, валідації, файлових завантажень та API-документування. Завдяки хорошій документації й активній спільноті Laravel забезпечує високу продуктивність у типових CRUD-операціях і швидкий випуск MVP, а механізми валідації запитів і захист від CSRF, XSS та SQL-ін’єкцій із коробки гарантують базовий рівень безпеки.

Проте в порівнянні з Java Spring + JPA Laravel має кілька суттєвих обмежень для проєктів із максимальними вимогами до продуктивності, безпеки та довгострокового супроводу. По-перше, динамічна типізація PHP, хоч і доповнена типами в останніх версіях, не забезпечує такого рівня статичної перевірки коду на етапі компіляції, як Java, де помилки відсіюються ще до виконання програми. По-друге, однопоточна обробка запитів у PHP-середовищі підвищує час відгуку під навантаженнями, тоді як Spring Boot з Hibernate ефективно розподіляє запити між потоками JVM і використовує кеш

другого рівня для оптимізації доступу до реляційної БД. По-третє, хоча Laravel пропонує готові пакети для безпеки, для повноцінної відповідності корпоративним стандартам OWASP і аудиту потрібна інтеграція декількох сторонніх рішень, у той час як Spring Security забезпечує «із коробки» комплексну систему авторизації, аутентифікації та логування подій. Нарешті, екосистема Laravel не має такої довгострокової LTS-підтримки, як у Java, а часті оновлення можуть вимагати додаткових зусиль із міграцій і тестування, що підвищує ризики при масштабуванні і супроводі проєкту впродовж багатьох років.

C# із фреймворком ASP.NET Core забезпечує дуже високу продуктивність завдяки своїй легковажній веб-серверній реалізації Kestrel, оптимізованій під асинхронну обробку запитів. Статично типізована мова C# у поєднанні з .NET Core дає змогу відловлювати значну частину помилок на етапі компіляції, а потужна система залежностей і вбудований механізм інверсії керування (Dependency Injection) сприяють чіткій архітектурі моноліту або гібридного рішення. Для роботи з реляційними БД і CSV-файлами використовують Entity Framework Core, який підтримує міграції, LINQ-запити і відкладене (lazy) завантаження, а також інтеграцію зі сторонніми бібліотеками для пакетного імпорту та кешування. Розробка бекенду на стандартному VPS не вимагає складного DevOps-конфігурування, достатньо розгорнути. NET-додаток під Linux чи Windows, а вбудовані шаблони проектів і CLI-інструменти (dotnet CLI) пришвидшують створення MVP. Безпека реалізована через ASP.NET Identity, політики авторизації та захист від CSRF і XSS «із коробки», а регулярні LTS-релізи платформи гарантують тривалу підтримку. Завдяки широкій спільноті та сильній підтримці Microsoft цей стек ідеально підходить для швидкої розробки надійних веб-сервісів із чіткою структурою коду та високою продуктивністю.

Однак у порівнянні з Java Spring + JPA ASP.NET Core має кілька обмежень для проєктів із максимальними вимогами до масштабованості, безпеки та довгострокового розвитку. По-перше, хоча EF Core прогресує, він досі не досягає рівня функціональності Hibernate у плані другого рівня

кешування, стратегій вибірки й оптимізації складних SQL-запитів, які в Spring Data JPA можна тонко налаштовувати через конфігурацію fetch plan і кешу. По-друге, екосистема Spring Security пропонує ширший спектр перевірених у production рішень для аутентифікації, авторизації за ролями та політиками, а також сполучення з OAuth2/OIDC, ніж стандартні компоненти ASP.NET Identity, які часто доводиться доповнювати зовнішніми бібліотеками. По-третє, у великих enterprise-проєктах архітектурні патерни Spring (AOP, event-driven design, declarative transaction management) більш розвинені й краще документовані, що полегшує підтримку та розширення системи через багато років. Нарешті, довші LTS-цикли Java та ширша спільнота корисних плагінів і стартерів Spring Boot дають додаткову впевненість у стабільності та передбачуваності оновлень для критично важливих систем, у той час як ASP.NET Core, хоч і швидко розвивається, все ще вимушений адаптувати частину своїх компонентів до змін у .NET-платформі.

Далі буде йти мова про вибір СУБД. Зберігання даних потрібно оформити чітко структуровано, тому буде розглянуто тільки конкурентні системи з ринку реляційних баз даних. Головними вимогами до бази даних в цьому проєкті являються швидкий доступ до даних, можливість написання складних запитів в БД, швидка робота на середніх та великих об’ємах даних, можливість розширення та вдосконаленняЮ захист від аномалій при читанні. Для цього проєкту було обрано PostgreSQL. Альтернативами для цієї СУБД являються: MySQL, OracleDB, та MariaDB.

MySQL – одна з найвідоміших систем керування реляційними базами даних із відкритим вихідним кодом, яка вже тривалий час використовується у багатьох веб-додатках і комерційних продуктах. Її популярність зумовлена простотою налаштування, великою кількістю навчальних матеріалів, підтримкою широкого спектру мов програмування та хорошою інтеграцією з популярними CMS і фреймворками, такими як WordPress, Laravel, Django тощо. MySQL демонструє стабільну продуктивність при обробці типових CRUD-операцій, має підтримку транзакцій, зовнішніх ключів, індексів і реплікації, що дозволяє масштабувати систему вертикально або

горизонтально. Вона добре працює на середніх об’ємах даних, має зрозумілий синтаксис SQL і вбудовані засоби безпеки, такі як контроль доступу та шифрування з'єднань. Також варто відзначити, що MySQL має два основні рушії зберігання — InnoDB і MyISAM, де перший гарантує підтримку транзакцій і забезпечує цілісність даних.

Однак, порівнюючи з PostgreSQL, MySQL втрачає перевагу при роботі з великими обсягами даних, складними запитами та аналітичними операціями. PostgreSQL має значно кращу підтримку складної логіки в SQL, зокрема — CTE (WITH-запити), window-функції, повноцінну підтримку JSONB для гібридних моделей даних, типи масивів та користувацькі типи даних. Це дозволяє PostgreSQL бути не тільки класичною реляційною СУБД, а й гнучкою платформою для зберігання напівструктурованих даних. Крім того, механізми контролю конкурентності в PostgreSQL, зокрема MVCC (мультиверсійний контроль транзакцій), реалізовані точніше, забезпечуючи кращий захист від аномалій читання, таких як phantom reads чи non-repeatable reads. PostgreSQL також виграє в гнучкості розширення: вона дозволяє створювати власні функції на різних мовах, підключати сторонні модулі (наприклад, PostGIS) та створювати індекси на основі виразів. Таким чином, у контексті цього проєкту, який потребує масштабованості, гнучкості запитів і високого ступеня надійності даних, PostgreSQL є кращим вибором.

Oracle Database — це одна з найпотужніших і найфункціональніших реляційних СУБД на ринку, орієнтована передусім на корпоративний сегмент і критично важливі системи з високими вимогами до доступності, масштабованості та безпеки. OracleDB підтримує розподілені транзакції, реплікацію на рівні бази даних, гнучке шардінгування, механізми автоматичного балансування навантаження та відновлення після збоїв. Система має власні механізми обробки запитів, які дозволяють досягати оптимальної продуктивності навіть у випадках надскладної бізнес-логіки, з великою кількістю приєднань (JOIN), вкладених запитів або матеріалізованих представлень (materialized views). Oracle активно використовується у фінансових установах, телекомі, авіації — скрізь, де критично важлива

стабільність і відповідність стандартам безпеки, зокрема — підтримка стандартів PCI DSS та ISO 27001. Також важливо, що Oracle пропонує надбудови типу APEX для швидкої побудови UI, а також глибоку інтеграцію з хмарною інфраструктурою Oracle Cloud.

Проте у порівнянні з PostgreSQL OracleDB часто виявляється надмірною та недоцільною для середніх або стартап-проєктів, як у випадку з даною системою. По-перше, Oracle є пропрієтарною СУБД з комерційною ліцензією, що тягне за собою високу вартість підтримки, оновлень і масштабування. PostgreSQL, як система з відкритим кодом, не потребує жодних витрат на ліцензування, що є критично важливим для проєктів, орієнтованих на MVP або обмежений бюджет. По-друге, складність налаштування та адміністрування Oracle є вищою, що потребує окремих сертифікованих спеціалістів. У PostgreSQL адміністрування виконується простіше, а функціональність розширюється через модулі та розширення, які легко інтегрувати без втрати стабільності. По-третє, PostgreSQL має сучасну реалізацію MVCC, повноцінну підтримку ACID і більш зручний для розробників механізм роботи з JSON, що робить її гнучкішою в умовах швидко змінюваних вимог. Отже, попри всю потужність OracleDB, вона не виправдовує себе в межах цього проєкту, де ключовими є швидкість розробки, відкритість, аналітична гнучкість і стабільність при помірних витратах.

MariaDB – це реляційна система управління базами даних, яка з’явилася як форк MySQL після придбання останньої корпорацією Oracle. Вона зберегла повну сумісність з MySQL на рівні синтаксису, API та конфігурацій, що дозволяє легко мігрувати проєкти між цими двома системами. У порівнянні зі своїм "прабатьком", MariaDB активно розвивається спільнотою з відкритим кодом, включаючи покращення продуктивності, підтримку нових рушіїв зберігання (як-от Aria, ColumnStore для аналітики, MyRocks для високої швидкодії), а також розширені можливості реплікації — наприклад, multi-source replication, що дозволяє об'єднувати дані з кількох джерел. Система зручна для інтеграції у невеликі та середні проєкти, де потрібен швидкий доступ до даних, транзакційна цілісність, підтримка SQL-стандарту та

зниження залежності від комерційних рішень. MariaDB має гарну підтримку в хмарних провайдерах, а також хорошу документацію, що пришвидшує розгортання проєктів і налаштування бази на стандартному VPS.

Однак, попри численні переваги, MariaDB поступається PostgreSQL за рядом ключових характеристик, що є критично важливими в рамках цього проєкту. Насамперед, PostgreSQL забезпечує глибшу підтримку складних SQL-конструкцій — таких як рекурсивні запити (WITH RECURSIVE), віконні функції, CTE, і складні транзакційні ізоляції з гарантією захисту від аномалій читання. PostgreSQL також має повноцінну підтримку JSONB з можливістю індексації, що дозволяє зручно працювати з напівструктурованими даними в межах однієї бази. Крім того, PostgreSQL демонструє кращу продуктивність на великих обсягах даних, де потрібна аналітика в реальному часі або паралельна обробка запитів, завдяки продуманому планувальнику запитів і підтримці паралельного виконання. Ще однією перевагою є екосистема розширень — таких як PostGIS, pg_partman, TimescaleDB — які дозволяють PostgreSQL адаптувати під різноманітні задачі, від геоінформаційних систем до time-series баз. У контексті даного проєкту, де важливі масштабованість, складна бізнес-логіка, продуктивність і безпека, PostgreSQL перевершує MariaDB як у гнучкості, так і в перспективності розвитку.

В якості комунікації з користувачем і працівниками, а також для можливості підтвердження номеру телефону потрібно створити систему комунікації в месенджері. Для проєкту було обрано месенджер Telegram. Конкурентами на ринку месенджерів являються WhatsUp, Viber.

WhatsApp — один із найпопулярніших месенджерів у світі, який має величезну аудиторію, особливо в країнах Азії, Латинської Америки та частково Європи. Його ключовими перевагами є простота використання, надійне end-to-end шифрування за замовчуванням, глибока інтеграція з мобільною адресною книгою, а також підтримка мультимедійних повідомлень і групових чатів. Для бізнесу WhatsApp пропонує WhatsApp Business API, що дозволяє надсилати повідомлення користувачам, проводити авторизацію та організовувати базову взаємодію з ботами або операторами. У деяких

випадках також можливо використовувати повідомлення як підтвердження номера телефону. Компанія Meta, якій належить месенджер, підтримує стабільну інфраструктуру та забезпечує високу доступність сервісу в усьому світі.

Однак, у контексті створення бота для стартап-проєкту, WhatsApp має низку обмежень, які роблять його менш привабливим у порівнянні з Telegram. По-перше, WhatsApp Business API є платним — стягується плата за кожну ініційовану розмову (як з боку бізнесу, так і користувача), що призводить до постійних витрат і ускладнює масштабування. По-друге, функціональність ботів у WhatsApp дуже обмежена — бот не може працювати повністю автономно: потрібна складна інфраструктура, зовнішні інтеграції (через сторонні сервіси на кшталт Twilio), і неможливо створювати повноцінні меню, inline-кнопки, або кастомні інтерфейси без обмежень. По-третє, реєстрація номера для API вимагає перевірки бізнесу, часу на активацію та схвалення Meta, що не відповідає потребі швидкого запуску MVP. Натомість Telegram дозволяє безкоштовно створити і запустити бота за лічені хвилини, має повноцінну API, розширені UI-можливості та значно гнучкішу логіку взаємодії з користувачем. Це робить Telegram значно більш привабливою платформою для реалізації бота на етапі запуску проєкту або при обмеженому бюджеті.

Viber — популярний месенджер, особливо поширений у країнах Східної Європи, Близького Сходу та частково в Азії. Його основними перевагами є простота використання, підтримка як текстових, так і голосових повідомлень, наскрізне шифрування для приватних чатів, а також наявність офіційного API для ботів, який дозволяє створювати базову логіку взаємодії з користувачем. Боти у Viber можуть відображати кнопки, меню, зображення, опитування, і реагувати на повідомлення користувача. Крім того, як і WhatsApp, Viber підтримує валідацію номера телефону, що може бути корисно для верифікації користувачів у системі. Розробка ботів у Viber є безкоштовною, а для підключення офіційного аккаунта не потрібно складної процедури схвалення

(як у WhatsApp). Також перевагою є активна українська аудиторія — зокрема в категорії малого бізнесу та старшого покоління.

Проте при порівнянні з Telegram Viber виявляється менш зручним для реалізації технічно складного або кастомізованого бота. По-перше, API Viber значно обмеженіший, ніж Telegram Bot API — не підтримує inline-режим, callback-кнопки, повноцінні сценарії на основі станів або глибоку інтеграцію з повідомленнями (неможливість редагувати вже відправлені повідомлення, складно реалізувати багаторівневі меню). По-друге, документація Viber менш детальна, спільнота розробників менша, а кількість прикладів коду та бібліотек обмежена, що ускладнює розробку. Крім того, Telegram дозволяє швидко запускати, тестувати та масштабувати бота без прив'язки до публічної сторінки або обов'язкового брендування, тоді як у Viber потрібен офіційний паблік-акаунт для повноцінної роботи. Нарешті, Telegram надає більше простору для кастомізації — від використання inline web apps, rich media до налаштувань приватності і адміністрування груп. У підсумку, попри деякі переваги Viber на локальному ринку, Telegram є значно більш гнучкою, потужною та економічно вигідною платформою для створення інтерактивного бота в межах даного проєкту.

1.4 Обґрунтування вибраного напрямку вирішення задачі

У сучасних умовах цифрової трансформації бізнесу ефективна робота онлайн-комерційної платформи вимагає комплексного та технологічно збалансованого підходу до проєктування системи. Для забезпечення високої продуктивності, безпеки, масштабованості та зручної взаємодії з користувачами необхідно обирати перевірені, стабільні та водночас інноваційні рішення, що вже зарекомендували себе у розробці великих комерційних систем. У випадку створення інтернет-магазину з підтримкою інтегрованої логістики, розгалуженої клієнтської бази, автоматизованого управління каталогом товарів та обробкою замовлень, доцільним є застосування технологічного стеку, до складу якого входять: Angular для фронтенду, Java Spring для бекенду, PostgreSQL як основна реляційна система

керування базами даних, а також Telegram API як інтерактивний канал комунікації з клієнтами.

Angular — це потужний клієнтський фреймворк, розроблений Google, який забезпечує структуровану побудову динамічних односторінкових додатків (SPA). Завдяки суворій архітектурі, підтримці модульності, декларативному підходу до інтерфейсів і широкому набору вбудованих інструментів, Angular дозволяє створювати масштабовані та легкі в супроводі інтерфейси. Це особливо важливо для інтернет-магазину, де інтерфейс взаємодії має залишатися швидким та зрозумілим навіть за великого обсягу даних — численних категорій, фільтрів, кошиків, сторінок замовлення. Інтеграція з RxJS для реактивного програмування та можливість централізованого управління станом (наприклад, через NgRx) дають змогу ефективно обробляти складні сценарії користувацької поведінки. При цьому офіційна підтримка, оновлення, активна спільнота та велика кількість готових рішень роблять Angular безпечним вибором для комерційної розробки.

На рівні бекенду обрано Java Spring — один із найпотужніших і найпопулярніших фреймворків для створення корпоративних додатків. Його модульна структура (Spring Boot, Spring Security, Spring Data тощо) дозволяє швидко створювати надійні RESTful API, реалізовувати автентифікацію та авторизацію, працювати з транзакціями та захищати критичні точки доступу. У зв’язці з JPA (Hibernate), Spring дає можливість ефективної взаємодії з базою даних, гнучкого побудування запитів та реалізації складної логіки обробки даних. Завдяки своїй зрілості, Spring є перевіреним вибором для розробки високонавантажених систем — включаючи системи обліку, електронної комерції, фінансові платформи. Додатково, використання Java забезпечує високу продуктивність при паралельному виконанні завдань, можливість оптимізації пам’яті та доступ до багатої екосистеми корпоративних рішень.

Що стосується зберігання даних, то основним вибором стала PostgreSQL — одна з найпотужніших відкритих систем управління реляційними базами даних. Її основними перевагами є підтримка складних SQL-запитів, транзакційності, розширюваність типів даних, а також розгалужена система

прав доступу, що є критично важливою для захисту чутливої інформації. PostgreSQL підтримує роботу з великими обсягами даних, паралельні запити, індексацію по різних критеріях, зокрема повнотекстовий пошук, та може бути масштабована горизонтально й вертикально. У порівнянні з іншими рішеннями, PostgreSQL демонструє відмінну продуктивність, чудово підходить як для OLTP-, так і для аналітичних сценаріїв (OLAP), та активно розвивається за підтримки широкої спільноти і великих корпорацій.

Особливу роль у системі комунікації відіграє Telegram API, який забезпечує миттєвий зворотний зв'язок з користувачами, верифікацію телефонів та обслуговування клієнтів безпосередньо в месенджері. На відміну від багатьох конкурентів, Telegram дозволяє безкоштовне створення ботів з повним API-доступом, підтримує розширений UI з inline-кнопками, меню, веб-додатками, а також не потребує тривалих етапів погодження або сертифікації. Це забезпечує високу швидкість розгортання MVP та дає змогу впроваджувати інноваційні сценарії роботи з клієнтом — сповіщення про статус замовлення, підтвердження оплати, push-кампанії тощо. Telegram, з його відкритістю та гнучкістю, ідеально підходить для інтеграції з іншими компонентами системи через вебхуки та REST API.

Таким чином, поєднання Angular, Java Spring, PostgreSQL та Telegram API створює цілісний, масштабований, безпечний і ефективний технологічний стек для реалізації інтернет-магазину. Такий вибір забезпечує надійну роботу як у пілотному режимі, так і при зростанні навантаження та масштабів проєкту, зберігаючи гнучкість та відповідність сучасним стандартам розробки.

1.5 Мета і задачі роботи

Полягає в створенні Веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів. Має буде створено:

- веб сервіс для обробки клієнтських запитів;
- клієнтський сервіс для взаємодії з користувачами через месенджер Telegram;
- веб-сервіс інтерактивного користувацького інтерфейсу.

Ця система потенційно може стати основною платформою торгівлі для середнього бізнесу.

.

2 РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ОБРОБКИ КЛІЄНТСЬКИХ ЗАПИТІВ

2.1 Технічні вимоги до об’єкту професійної діяльності

2.1.1 Найменування і призначення об’єкта професійної діяльності

Веб-сервіс для обробки клієнтських запитів, який включає серверну частину інформаційної системи інтернет зоомагазину, призначений для надання REST API для клієнтських застосунків для керування контентом на сайті.

Система забезпечує обробку запитів, взаємодію з базою даних, телеграм ботом, роботу зі сторонніми сервісами еквайрінгу, перевірку прав доступу, а також відповідає за захист та збереження даних.

2.1.2 Вимоги до функціонування об’єкта професійної діяльності

Система забезпечує наступні функціональні можливості:

- створення замовлень;
- реєстрацію аккаунтів;
- синхронізацію з системою обліку товарів магазину;
- сервіс має повертати список продуктів в вигляді сторінок;
- сервіс має повертати список відфільтрованих продуктів;
- сервіс має надавати список можливих фільтрів, генеруючи їх динамічно, в залежності від стану бази даних;
- сервіс має повертати повну інформацію про товар, по унікальному ідентифікатору;
- сервіс має реалізовувати можливість створення та перегляду коментарів.

2.1.3 Структурні вимоги до об’єкту професійної діяльності

Система забезпечує наступні функціональні можливості:

- модуль керування персональним кабінетом (відповідальний за реєстрацію облікових записів, та контроль персональної інформації користувача);
- модуль захисту (забезпечує захист закритого API від несанкціонованого доступу);
- модуль валідації (забезпучує зберігання та використання тільки коректної інформації, перевіряючи вхідні дані)
- модуль фільтрації (формує фільтра, за якими можна фільтрувати товари в базі даних);
- модуль пагінації (збирає інформацію в сторінки певного розміру та порядку, повертаючи на клієнтську сторону тільки певну частину потрібної інформації, для економії трафіку, та підвищення продуктивності системи);
- модуль транспортування файлів (забезпечує надійне підключення та швидке з’єднання для отримання всіх необхідних файлів по мережі інтернет);
- модуль банківських операцій (підтримує створення реквізитів для оплати, та саму оплату);
- модуль фільтрації (дозволяє фільтрувати потрібні ресурси, при виконанні пошуку товарів в базі даних).

2.1.4 Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами Системи

Обмін інформацією між внутрішніми компонентами системи реалізується через імплементацію архітектурного патерну MVC. Також організація правильного потоку даних організована за допомогою влаштованої в Spring Framework функції Dependency Injection.

2.1.5 Вимоги до характеристик взаємозв'язків створюваної Системи із суміжними Системами

Для обміну інформаціїєю із суміжними системами використовуються протоколи:

- HTTP (використовується для створення REST API, в форматі обміну інформацією “запит-відповідь”);
- FTP (протокол для передачі файлів по мережі).

2.1.6 Межі розвитку

Збільшення масштабів може призвести до потреби в створенні нових структурних модулів:

- модуль підбору схожих товарів;
- модуль аналізу попиту користувачів;
- модуль таргетованої реклами;
- модуль оптимізації пошукових запитів.

Такі модулі можуть збільшити трафік сервіса, збільшити відсоток успішних замовлень та покращити сервіс магазину, надавши користувачам більш персоналізований інтерфейс пошуку підходящих товарів.

2.1.7 Показники призначення

До показників роботи об’єкту професійної діяльності відноситься:

- швидкість запуску програми до двох хвилин;
- швидкість оновлення БД до 1 хв;
- швидкість обробки запиту на отримання 1 сторінки відфільтрованих товарів до 3с.

2.2 Розробка апаратної частини

2.2.1 Обґрунтування технічних характеристик програми

2.2.1.1 Опис призначення вхідних і вихідних даних

Для комфортної роботи програми - вхідні дані будуть приходити по декільком каналам.

Запити по мережі, з використанням протоколу HTTP. Такі запити мають потрапляти до маршрутизатора, та делегуватися в виконанні різним методам, передаючі в ці методи, всю доступну інформацію з запиту. Комунікація з використанням HTTP, використовує PathVariables, RequestParams, JSON bodies, headers, credentials.

Вхідні дані передані по HTTP протоколу будуть мати всю необхідну інформацію для виконання, здебільшого пошукових операцій (інформація про номер переглядаємої сторінки, примінених фільтрів, розмір сторінки, категорію товарів, іменування, тощо), або операцій створення нової сутності, наприклад оформлення замовлення (Ім’я людини, номер телефону, адреса доставки, список замовлених товарів, тип доставки, тощо). Також для коректної роботи сервісу, клієнтська сторона повинна пам’ятати та зберігати закріплений за собою JSESSIONID, який використовується Spring Security, для ідентифікації кожного унікального користувача. Цей токен буде повертатися сервісом, після аутентифікації на сайті, і буде використовуватися для подальшої авторизації при кожному новому запиті.

Передача даних про наявність товарів в магазинах, а також інформація про оформлення нового замовлення, реалізована через передачу файлів до FTP серверу, являється оптимальним варіантом. Вхідними даними - будуть являтися CSV таблиці, які зберігають інформацію про актуальний стан бази даних. Вихідними даними, при передачі файлів через FTP будуть являтися JSON файли, які будуть в собі зберігати кожне окреме замовлення. Їх іменування повинно бути унікальним, тому воно буде формуватися з дати в мілісекунах, та випадково згенерованого числа, зводячи шанс колізії імен майже до нуля. Синхронізація файлів з певною періодичністю дозволить тримати дані в сервісі актуальними, і надавати користувачам тільки справжню інформацію, яка не вступає в колізію з реальним станом системи. Спираючись на показники, середньостатистичного магазину зоотоварів схожого розміру -

можна зробити висновки, що в день формується від 5 до 10 замовлень, через інтернет магазин. Таким чином, розраховуючи навантаження магазину навіть на 10 замовлень за годину - оптимальним періодом синхронізації буде 1 година. Вірогідність колізії в вигляді оформлення замовлення, вже неіснуючого товару - прийнятно низька, та близиться до нуля. Чіткий показник порахувати майже неможливо, але якщо взяти показник в 12 000 позицій товарів в магазині, а також факт того, що їх кількість складає хоча б 5 штук на 1 позицію, то шанс того, що за 1 годину розпродадуть всі 5 товарів, в умовах ідеально рівномірного попиту становить 10^{-18}.

2.2.1.2 Опис і обґрунтування вибору складу технічних і програмних засобів, що використовує програма.

Для коректного виконання цієї частини системи - потрібно використання додаткових бібліотек, які входять до складу стандартного складу Spring Framework, або мають сторонній ресурс. Для підключення таких бібліотек, як: spring-boot-starter-data-jpa, spring-boot-starter-data-rest, spring-boot-starter-security, spring-boot-starter-web, jackson-dataformat-xml, slugify, spring-boot-starter-webflux, spring-boot-devtools, postgresql, lombok, spring-boot-starter-test, commons-net, spring-boot-starter-thymeleaf, liqpay-sdk, jaxb-api - потрібна система підвантаження залежностей. Основними опціями в такому випадку являються - Maven, Gradle. Для виконання цього проєкту було обрано саме Maven. Він має високу популярність серед розробників, та вже давно знаходиться на ринку, що підкреслює його стабільність, довготривалість та актуальність використання в сучасних проєктах. Він стабільний та стандартизований, що стало основною причиною для вибору саме цієї технології.

Серед бібліотек які були підключені знаходиться Spring Security. Ця офіційна частина екосистеми Spring Framework, дозволяє зручно та надійно налаштовувати захист API, від несанкціонованого доступу. Чутлива інформація вимагає високої степені захищеності. Проєкт буде напряму взаємодіяти з персональними даними користувачів, тому має буде захищеним

від основних типів атак. Spring Security одразу надає високий рівень захисту, тому він був обраний для цього проєкту. Окрім цього, слід зазначити, що він налаштовується за двома основними паттернами проєктування: O2Auth, UserDetailsService. Через потребу в високій контрольованості аккаунтів користувачів - було обрано варіант створення захисту через стандартний UserDetailsService.

База даних має бути налаштована з урахуванням потреби в поступовому розростанні інформації. Алгоритми пошуку та додавання товарів - повинні бути максимально швидкими. Тому для збереження інформації було обрано PostgreSQL. Ця технологія за останній час зайняла дуже міцне положення на ринку IT технологій, через свою легку масштабованість, безкоштовність, швидкість та розширений функціонал.

Для комфортної взаємодії з базою даних, потрібно обрати технологію надсилання запитів. Найкращим варіантом на даний момент являється ORM підхід. В Java Spring це місце традиційно займає Hibernate, який повністю реалізує специфікацію JPA. Такий підхід допоможе усунути проблему можливих атак за допомогою SQL-Injection, який є однією з найбільших загроз в сучасній веб мережі. Hibernate автоматично екранує всі запити, не дозволяючи шкідливим запитам потрапити до SQL запиту, та виконатися на базі даних.

Для роботи з запитами по протоколу HTTP, зачасту використовуються такі формати як XML та JSON. На даний момент, використання XML - являється дуже персоналізованим та екзотичним налаштуванням, яке не має популярності серед інтернет-спільноти. Для легкого підтримання коду, та розширення API було обрано саме JSON формат, в якому будуть передаватися дані по мережі. Для їх коретного парсингу - буде використана бібліотека Jackson. Вона є влаштованою в Spring Framework. Окрім цього, було прийнято рішення - додати її додаткове розширення, яке дозволяє вручну формувати JSON об’єкти, та парсити їх назад. Підхід розробки API з використанням Jackson - являється оптимальним балансом між гнучкістю та швидкістю розробки.

Інтернет еквайрінг - дуже важлива частина будь-якого інтернет магазину. Тому було обрано саме LiqPay. Цей український продукт дає можливість вільно налаштовувати систему оплати, в режимі Sandbox, не ризикуючи невдалими фінансовими операціями, та надходженнями грошей на неправильні рахунки. Також LiqPay – є продуктом банку ПриватБанк, який являється надійним українським продуктом, який функціонує вже багато років, і на даний момент являється національним банком України.

Для обробки CSV файлів часто використовуються вже готові бібліотеки, але для цього випадку було обрано створити власний клас, та підхід до аналізу CSV таблиць. Для такої бібліотеки потрібно розуміти принципи Generic класів, та методів. Що являється основою Java Core, та не може викликати якісь довгострокові проблеми.

Також в проєкті буде використана технологія NGINX. Він дозволяє зручно кастомізувати проксі, та перенаправляти запити, по портам. Таким чином, на одному сервері можна запустити Frontend, Backend та Телеграм бота, без ризику конфлікту маршрутизації.

2.2.2 Опис розробленої програми

2.2.2.1 Програмне забезпечення й мова програмування, необхідні для функціонування програми

Для коректної роботи веб-сервісу слід правильно налаштувати оточення, в якому воно запускається. Спочатку потрібно підібрати коректне технічне обладнання. (Таблиця 2.1 - 2.2)

Таблиця 2.1 – Мінімальні технічні вимови

Мінімальні технічні вимоги	
Компонент	**Параметри**
CPU	1 vCPU
RAM	2 ГБ RAM

Диск	SSD 15 ГБ
Мережа	100 Мбіт/с або більше

Таблиці 2.2 – Рекомендовані технічні вимоги

Рекомендовані технічні вимоги	
Компонент	**Параметри**
CPU	2 vCPU
RAM	4 ГБ RAM
Диск	SSD 30 ГБ (щоб було місце для резервних копій)
Мережа	100 Мбіт/с або більше

Програмне забезпечення яке повинно бути встановлене:

- JDK 21 (або будь-якої іншої сумісної версії);
- PostgreSQL (з налаштуванням створеного користувача для підключення до БД);
- Git (контроль версій);
- NGINX (проксі, який потрібен для маршрутизації запитів);
- Certbot (генерування SSH ключів, для захищеного каналу передачі даних).

2.2.2.2 Опис логічної структури програми, зв'язок з іншими програмами

Логічна структура програми являється важливою частиною. Вона дає змогу спроєктувати систему, та розібратися з її використанням. Графічна схема протоколів, за допомогою яких різні частини сервісу - спілкуються поміж собою. (Рисунок 2.1)

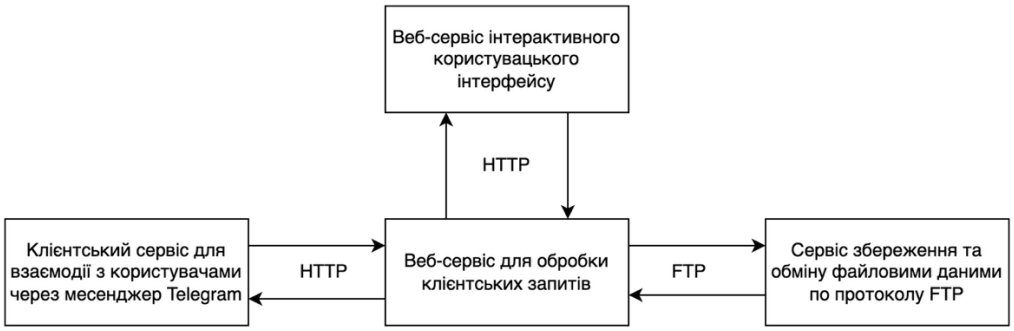


Рисунок. 2.1 – Схема протоколів спілкування

Обмін повідомленнями HTTP буде оформлений за допомогою REST принципів, для того, щоб зробити адреси більш стандартизованими, та облегшити подальшу розробку.

FTP протокол буде обмінюватися файлами в форматі JSON, CSV, PNG, JPEG, JPG. Всі вони потрібні для коректної роботи програми.

Діаграма інформаційних ресурсів, в якій наочно продемонстровані всі ресурси, якими може користуватися Backend для отримання інформації, по запиту користувача. В ресурси використання Backend входять: База даних, FTP сервер (файли картинок, знімки БД, та замовлення), а також сторонні сервіси, такі як LiqPay в якому є інформація з приводу виконаних транзакцій, та їх результату, або такі як Telegram Bot, який може відправляти повідомлення користувачам, в зручній для них формі. (Рисунок 2.2)

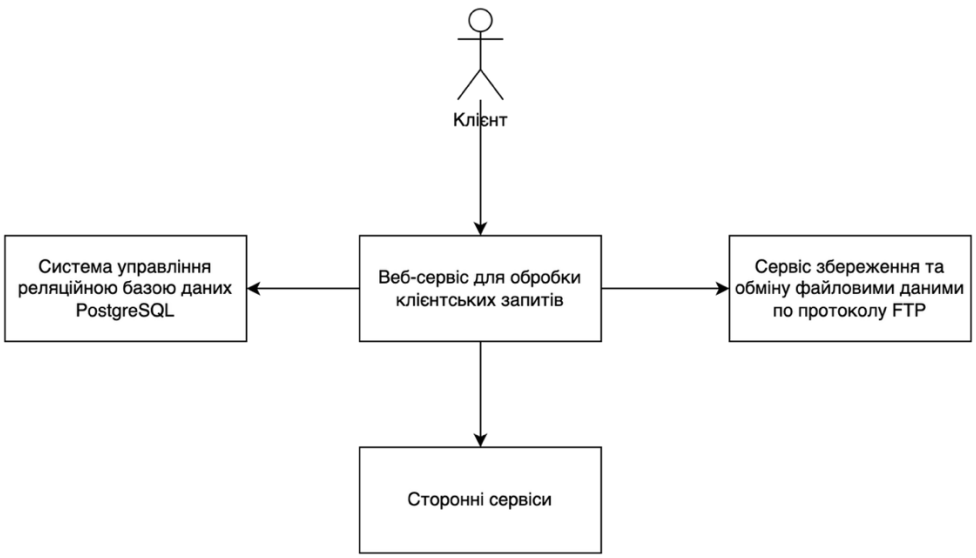


Рисунок. 2.2 – Діаграма інформаційних ресурсів

Представлений підход до організації коду сервісу, з використанням популярного патерну проєктування MVC (Рисунок 2.3)

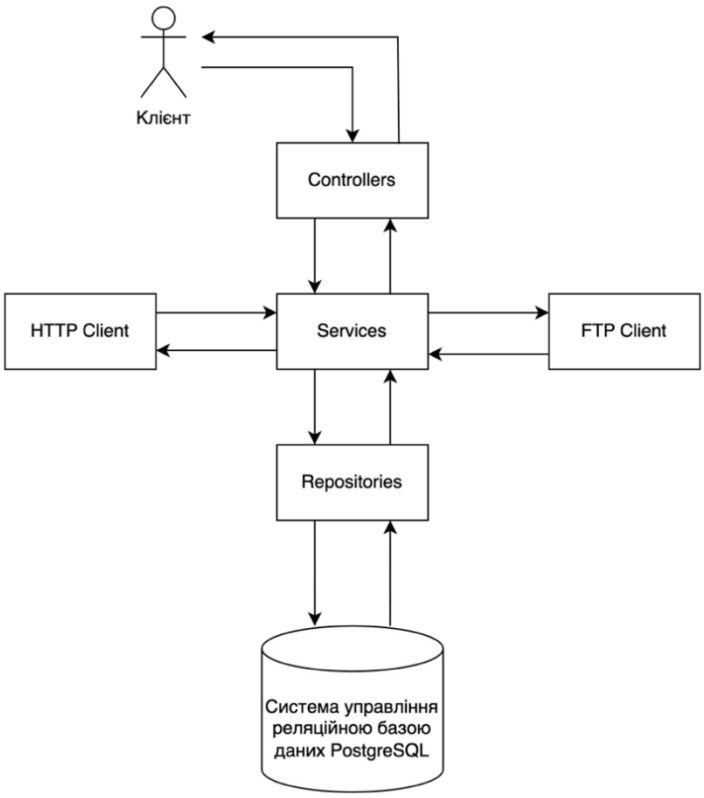


Рисунок. 2.3 – MVC архітектура сервіса (взаємодія між прошарками)

Схема бази даних (Рисунок 2.4). В ній наведена архітектура по якій в базі даних будуть зберігатися таблиці. Також в діаграмі наведені їх зв’язки.

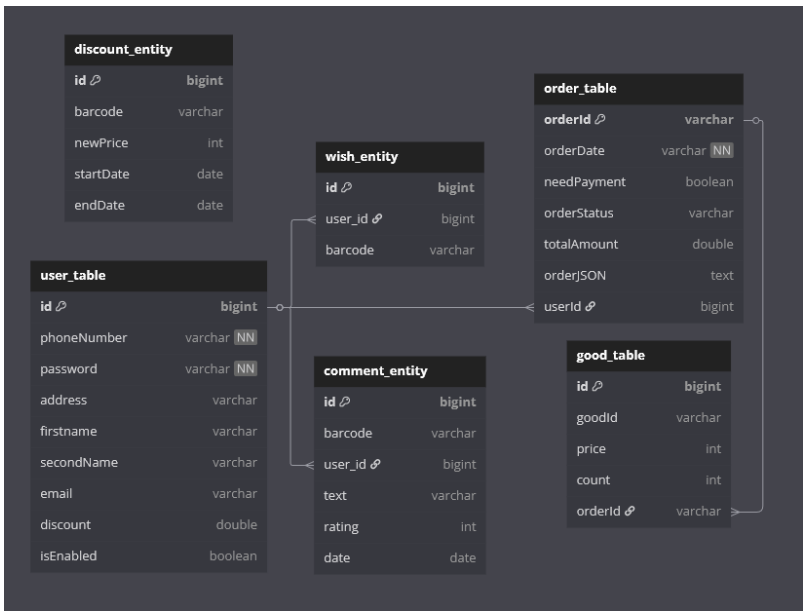


Рис. 2.4 – Діаграма таблиць в Базі Даних

2.2.2.3 Виклик і завантаження програми (спосіб виклику програми з відповідного носія даних, вхідні точки в програму)

Всі пояснення які я надаватиму далі, будуть відповідати алгоритму налаштування на операційній системі UBUNTU. При використанні іншої операційної системи, команди та алгоритм дій може відрізнятися.

Для запуску - потрібно мати JAR файл, який має бути доставленим на сервер. Методи того, як це можна зробити досить різноманітні, будуть наведені тільки декілька з них:

– фізичний носій інформації;
– передача файлу через GITHUB репозиторій;
– передача файлу через FTP;
– передача файлу через SSH.

Після передачі на сервер, потрібно перейти за допомогою терміналу в теку з файлом, використовуючи команду cd в термінальному рядку. Після потрапляння до потрібної теки - ми повинні впевнитись що файл лежить саме в цій директорії, тому використовуємо команду ls, для перегляду внутрішнього наповнення теки. Якщо файл дійсно знаходиться там, то ми запускаємо його, за допомогою команди “nohup java -jar name_of_jar.jar &”. Ця команда дозволить виконуватись програмі в фоновому режимі, зберігаючи всі

логи в окремий файл nohup. Якщо всі тести пройшли успішно, а також успішно пройшло підключення до бази даних, то наш сервер буде працювати в фоновому режимі.

При умовах закритих портів серверу - ми можемо також окремо налаштувати nginx, встановивши його, а потім відредагувавши файл по шляху /etc/nginx/sites-available/default. В цьому файлі потрібно вказати новий сервер:

```
server {
    listen 8080;
    location / {
        proxy_pass http://127.0.0.1:8080;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Такі налаштування дозволять слухати зовнішній порт 8080 та перенаправляти його за допомогою proxy на внутрішній порт 8080, передаючі всі особливі й потрібні хедери.

2.2.2.4 Вхідні та вихідні дані у вигляді таблиці з наведенням типів, адрес, дозволів та ін. з необхідними коментарями (Таблиця 2.3 - 2.4)

Таблиця 2.3 – Опис вхідних даних

Метод	**URI**	**Опис**	**Параметри**	**Тіло**	**Аутентифі**

					кація
GET	/api/comments	Отримати коментарі за баркодом товару	String barcode	CommentDTO commentDT	ні
POST	/api/comments	Отримати коментарі за баркодом товару	-	-	так
POST	/api/update/	Оновлення бази даних	-	-	ні
GET	/api/filters/{category}	Отримати фільтри за категорією	-	-	ні
GET	/api/filters/byName	Отримати фільтри за назвою	String search, String category	-	Hі
GET	/api/images/{imageName}	Отримати зображення за назвою	-	-	ні
POST	/api/orders	Створити замовлення	-	OrderPostDTO orderPostDTO	так

Продовження таблиці 2.3

GET	/api/orders	Отримати замовлення користувача	-	-	так

POST	/api/payments/{id}	Отримати посилання на оплату	-	-	ні
POST	/api/payments/callback	Обробити callback від платіжної системи	String data, String signature	-	ні
GET	/api/products/v2/	Отримати всі товари	int page, int pageSize	-	ні
GET	/api/products/v2/bySlug	Отримати товар за slug	String slug	-	ні
GET	/api/products/v2/filter	Отримати відфільтровані товари	int page, int pageSize, String attributes, String category, int minPrice, int maxPrice, String sortBy, String search	-	ні

Продовження таблиці 2.3

GET	/api/products/v2/search	Отримати відфільтровані товари з пошуком	int page, int pageSize, String attributes, String category, int	-	ні

			minPrice, int maxPrice, String sortBy, String search		
GET	/api/products/v2/getDiscountedProducts	Отримати товари зі знижкою	int page, int pageSize	-	ні
GET	/api/products/v2/byName	Отримати товари за назвою	String name, int pageSize, int page	-	ні
POST	/api/users/	Реєстрація користувача	-	UserPostDTO userPostDTO	ні
GET	/api/users/phoneNumberExist	Перевірити чи існує номер телефону	String phoneNumber	-	ні
PUT	/api/users/updatePassword	Оновити пароль за номером телефону	String phoneNumber	-	ні

Продовження таблиці 2.3

PUT	/api/users/updateUserInfo	Оновити персональні дані користувача	String name, String surname, String email	-	так

GET	/api/users /current	Отримати поточного користувача	-	-	так
GET	/api/users /current/orders	Отримати замовлення поточного користувача	-	-	так
POST	/api/users /current/ wishes	Додати товар до обраного	String barcode	-	так
DELETE	/api/users /current/ wishes/{barcode}	Видалити товар з обраного	-	-	так
GET	/api/users /current/ wishes	Отримати список обраного користувача	-	-	так
POST	/api/admin/images/ upload	Завантажити банер	MultipartFile image, String link, String bannerId	-	так

Закінчення таблиці 2.3

GET	/api/admin/get/{bannerName}	Отримати зображення банера	-	-	ні

GET	/api/admin/get/link/{bannerName}	Отримати посилання на банер	-	-	ні
POST	/api/admin/discounts	Створити знижку	String barcode, int newPrice, Date startDate, Date endDate	-	так
DELETE	/api/admin/discounts	Видалити знижку	Long discountId	-	так
POST	/api/admin/login	Увійти до адмін-панелі	String username, String password	-	ні
POST	/api/admin/blacklist/ban	Забанити користувача за номером телефону	String phoneNumber	-	так
POST	/api/admin/blacklist/unban	Розблокувати користувача за номером телефону	String phoneNumber	-	так

Таблиця 2.4 – Опис вихідних даних

Метод	**URI**	**Тип повертаємого значення**
GET	api/comments	ResponseEntity<ListDTO<CommentDTO>>
POST	/api/comments	ResponseEntity<CommentDTO>
GET	/api/comments	ResponseEntity<ListDTO<CommentDTO>>
POST	/api/comments	ResponseEntity<CommentDTO>
POST	/api/update/	ResponseEntity<Void>
GET	/api/filters/{category}	ResponseEntity<ProductFiltersDTO>
GET	/api/filters/byName	ResponseEntity<ProductFiltersDTO>
GET	/api/images/{imageName}	byte[]
POST	/api/orders	ResponseEntity<String>
GET	/api/orders	ResponseEntity<List<OrderEntity>>
POST	/api/payments/{id}	ResponseEntity<LinkDTO>
POST	/api/payments/callback	ResponseEntity<Void>
GET	/api/products/v2/filter	ResponseEntity<ListDTO<ProductDTO>>

Продовження таблиці 2.4

GET	/api/products/v2/search	ResponseEntity<ListDTO<ProductDTO>>
GET	/api/products/v2/	ResponseEntity<ListDTO<ProductDTO>>
GET	/api/products/v2/getDiscountedProducts	ResponseEntity<ListDTO<ProductDTO>>
GET	/api/products/v2/bySlug	ResponseEntity<ProductDTO>
GET	/api/products/v2/byName	ResponseEntity<ListDTO<ProductDTO>>
POST	/api/users/	ResponseEntity<String>
GET	/api/users/phoneNumberExist	ResponseEntity<Boolean>
PUT	/api/users/updatePassword	ResponseEntity<Void>
PUT	/api/users/updateUserInfo	ResponseEntity<Void>
GET	/api/users/current	ResponseEntity<UserGetDTO>

Закінчення таблиці 2.4

GET	/api/users/current/orders	ResponseEntity<ListDTO<OrderDTO>>
POST	/api/users/current/wishes	ResponseEntity<Void>
DELETE	/api/users/current/wishes/{barcode}	ResponseEntity<Void>
GET	/api/users/current/wishes	ResponseEntity<ListDTO<ProductDTO>>
POST	/api/admin/images/upload	redirect:/api/admin/imagesUpdate
GET	/api/admin/get/{bannerName}	byte[]
GET	/api/admin/get/link/{bannerName}	ResponseEntity<Map<String, String>>
POST	/api/admin/discounts	redirect:/api/admin/discounts
DELETE	/api/admin/discounts	redirect:/api/admin/discounts
POST	/api/admin/login	redirect:/api/admin/home
POST	/api/admin/blacklist/ban	redirect:/api/admin/blacklist
POST	/api/admin/blacklist/unban	redirect:/api/admin/blacklist
GET	/api/admin/home	html
GET	/api/admin/imagesUpdate	html
GET	/api/admin/discounts	html
GET	/api/admin/blacklist	html
GET	/api/admin/login	html

3 РОЗРОБКА КЛІЄНТСЬКОГО СЕРВІСУ ДЛЯ ВЗАЄМОДІЇ З КОРИСТУВАЧАМИ ЧЕРЕЗ МЕСЕНДЖЕР TELEGRAM

3.1 Технічні вимоги

3.1.1 Найменування і призначення об’єкта професійної діяльності

Клієнтський сервіс для взаємодії з користувачами через месенджен Telegram, який включає зберігання інформації про зареєстрованих користувачів та груп. Призначений для надання інформації веб-сервісу обробки клієнтських запитів, при обробці запитиів на реєстрації, а також для оповіщення зареєстрованих користуваів.

Підтримуючи постійний зв’язок з іншими компонентами системи - він надає актуальну інформацію про стан користувача, а також може інформувати користувача про повідомлення з серверу.

3.1.2 Вимоги до функціонування об’єкта професійної діяльності

Вимоги до функціонування об’єкта професійної діяльності наведені далі:

- оповіщення груп в яких знаходяться працівники;
- відправка актуального паролю користувачу;
- форма реєстрації;
- оповіщення Backend про створення нового аккаунту;
- використання текстової розмітки MD для покращення читабельності системних сповіщень;
- валідація інформації при реєстрації;
- створення комфортного інтерфейсу для взаємодії з користувачем.

3.1.3 Структурні вимоги до об’єкту професійної діяльності

Далі будуть наведені структурні вимоги до об’єкту професійної діяльності:

- модуль валідації даних (відповідає за перевірку введеної користувачем інформації);

- модуль реєстрації користувача (відповідає за поетапний збір персональної інформації про користувача);
- модуль веріфікації телефоного номеру (відповідає за перевірку власності вказаного контакту, він має обов’язково належати поточному користувачу);
- модуль комунікації з суміжними компонентами системи (повинен забезпечувати швидку передачу інформації по мережі інтернет);
- модуль прослуховування повідомлень (відповідає за постійне спілкування з клієнтом, відповідаючи на його повідомлення);
- модуль форматування повідомлень (відповідає за графічне відображення повідомлень, їх форматування та наповнення текстом).

3.1.4 Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами Системи

Далі будуть наведені вимоги до способів і засобів зв’язку для інформаційного обміну між компонентами системи:

- використання Dependency Injection для виклику коректних методів;
- використання SQL для комунікацію з системою управління реляційною базою даних PostgreSQL;
- зчитування даних про повідомлення використовуючи Telegram API.

3.1.5 Вимоги до характеристик взаємозв'язків створюваної Системи із суміжними Системами

Далі будуть наведені вимоги до характеристик взаємозв’язків створюваної системи із суміжними системами:

- передача даних по HTTP, з обов’язковим SSL, через виский рівень чутливості зберігаємої та обробляємої інформації;
- використання форматів JSON для універсальної передачі об’єктів поміж суміжними системами;

- використання ORM системи для коректної взаємодії з Системою управління реляційною базою даних PostgreSQL, унеможливлюючи шанси на SQL ін’єкцію.

3.1.6 Межі розвитку

Перспектива розвитку включає в себе можливість створення нових модулів:

- модуль таргетованих сповіщень (може відповідати за оповіщення клієнтів, використовуючи їх персональну інформацію для підбору товарів);
- модуль акційних сповіщень (може відповідати за оповіщення клієнтів про актуальні акції та пропозиції магазину);
- модуль геолокації (може відповідати за навігацію користувача до найближчого магазину цієї мережі).

3.1.7 Показники призначення

Далі наведені показники призначення:

- цілодобове обслуговування;
- безкоштовні повідомлення;
- швидкість доставки повідомлення до користувача - до 5 с.

3.2 Розробка апаратної частини

3.2.1 Обґрунтування технічних характеристик програми

3.2.1.1 Опис призначення вхідних і вихідних даних

Telegram Bot є невід’ємною частиною архітектури веб-сервісу обслуговування клієнтів інтернет-магазину, що працює з зоотоварами. Його функціональне навантаження зосереджене на забезпеченні взаємодії з користувачами поза межами веб-інтерфейсу, виконанні верифікаційних процедур, надсиланні службових повідомлень та підвищенні безпеки системи. Telegram API дозволяє реалізувати асинхронну взаємодію з користувачем у зручному месенджері, що спрощує досвід використання та пришвидшує обробку подій.

Вхідні дані для Telegram Bot`a – це інформація, що надходить від користувачів або співробітників зоомагазину у вигляді повідомлень або подій. Основними видами вхідних даних є:

1. Команди користувачів – це шлях взаємодії з користувачем, коли він вводить спеціальну команду через чат-інтерфейс, який ініціює запуск відповідних сценаріїв
2. Реєстраційна форма – це ланцюг з послідовних повідомлень від користувача, які він відправляє в якості відповіді на інтерактивні запити бота, в них він вказує:
   - прізвище;
   - ім’я;
   - електрону пошту;
   - номер телефону;
   - події додавання бота в групу – бот запам’ятовує ті групи, куди його додали, для подальшої можливості оповіщення цих груп, про оформлені замовлення.

Вихідні дані, у відповідь на дії користувачів або системні події - діляться на наступні типи:

1. Оповіщення – приходять персоналу, який повинен підготуватися до видачі, або відправки замовлення. Сповіщаються групи, а також адміністрація.
2. Персональні повідомлення - бот має можливість оповіщувати конкретних користувачів, використовуючи їх номер телефону, який вони вказали при реєстрації.
3. Відправка повідомлень з тимчасовим паролем для входу в персональний кабінет. Такі повідомлення присилає бот, при запиті на Логін форму на сайті. В цьому випадку він сповіщає користувача про те, який саме пароль йому було надано для входу в систему.

3.2.1.2 Опис і обґрунтування вибору складу технічних і програмних засобів, що використовує програма

Для створення телеграм бота, була обрана бібліотека Telegram Bots, яка знаходиться в вільному доступі в maver-repository. Доцільність її використання зумовлена високим рівнем абстракції, що дозволяє спростити розробку. Стандартний варіант створення телеграм бота полягає в підключення до відкритого API, який в режими комунікації по протоколу HTTP повертає всю актуальну інформацію, а також приймає команди, які виконує. Бібліотека дозволяє абстрагуватися від конкретних HTTP запитів, та зосередитись на функціональній та архітектурній частині програми.

Підключення бібліотек делеговано технології Maven, а Spring Framework закриває всі технологічні питання пов'язані з розробкою звичайного серверу, який здатен приймати, обробляти, та відповідати на запити клієнтської частини.

База даних реалізована на PostgresSQL. Вибір цієї бази даних зумовлений спільним підбором Backend і TelegramBot. Таким чином, при запуску цих сервісів на одній машині - можна буде легче проводити процес інсталяції та запуску проєкту, не дублюючи Database Servers.

3.2.2. Опис розробленої програми

3.2.2.1 Програмне забезпечення й мова програмування, необхідні для функціонування програми

Мінімальні вимоги до технічного оснащеня працюючої системи дуже схожі на вимоги до Backend частини, але мають на порядок менші вимоги, і при вертикальному підвищені обчислювальних потужностей, ця система не стане набагато швидше працювати, тому що вона напряму залежить від офіційних серверів сервісу Telegram. В цьому контексті вона напряму залежить від швидкості інтернет з'єднання, тому інтернет з'єднання тут відіграє ключову роль (Таблиця 3.1)

Таблиця 3.1 – Мінімальні технічні вимоги

Мінімальні технічні вимоги	
Компонент	**Параметри**
CPU	1 vCPU

RAM	1 ГБ RAM
Диск	SSD 10 ГБ
Мережа	100 Мбіт/с або більше

Список встановленого ПО, яке потрібно для запуску серверу:

- JDK 21 (або будь-якої іншої сумісної версії);
- PostgreSQL (з налаштуванням створеного користувача для підключення до БД);
- Git (контроль версій);
- NGINX (проксі, який потрібен для маршрутизації запитів);
- Certbot (генерування SSH ключів, для захищеного каналу передачі даних).

Операційна система, в вибраному варіанті має бути Ubuntu - актуальної версії. На цій системі сервіс точно буде працювати стабільно, та матиме можливість встановити все додаткове необхідне ПЗ. Також окремою рекомендацією являється запуск цієї системи на одній машині, разом з Backend. Такий варіант являється оптимальним з точки зору економії та оптимізації системи.

3.2.2.2 Опис логічної структури програми, зв'язок з іншими програмами. Надати графічне представлення алгоритму програми

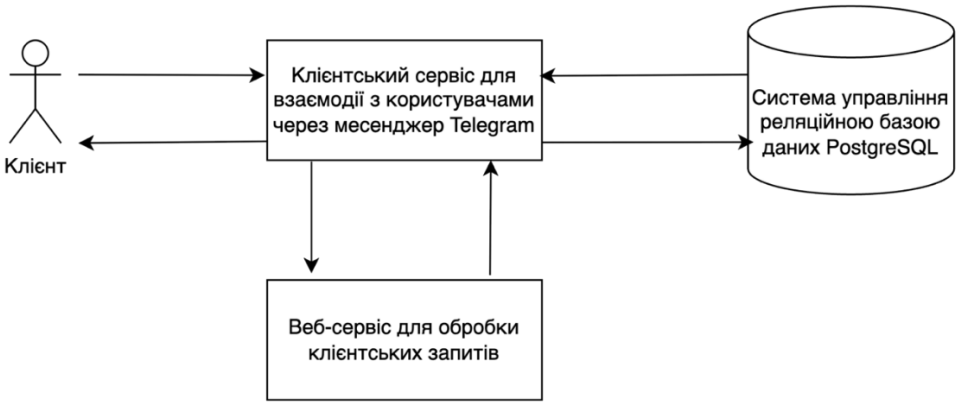


Рисунок. 3.1 – Логічна діаграма взаємодії сервісів з точки зору Telegram Bot

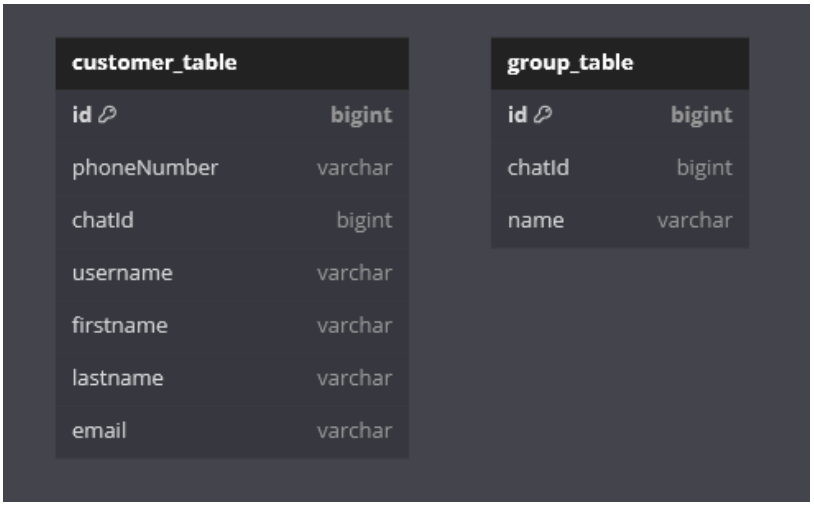


Рисунок. 3.2 – Схема бази даних TelegramBot

3.2.2.3 Виклик і завантаження програми (спосіб виклику програми з відповідного носія даних, вхідні точки в програму

Для запуску - потрібно мати JAR файл, який можна сформувати за допомогою maven. Після передачі файлу на сервер, потрібно перейти в теку з файлом. Якщо файл дійсно знаходиться там, то ми запускаємо його, за допомогою команди “nohup java -jar name_of_bot_jar.jar &”. Ця команда дозволить виконуватись програмі у фоновому режимі, зберігаючи всі логи в окремий файл nohup. Якщо всі тести пройшли успішно, а також успішно пройшло підключення до бази даних, то наш сервер буде працювати в фоновому режимі.

При умовах закритих портів серверу – ми можемо також окремо налаштувати nginx, встановивши його, а потім відредагувавши файл по шляху /etc/nginx/sites-available/default. В цьому файлі потрібно вказати новий сервер:

```
server {
   listen 8081;
   location / {
      proxy_pass http://127.0.0.1:8081;

      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
   }
}
```

Такі налаштування дозволять слухати зовнішній порт 8081 та перенаправляти його за допомогою proxy на внутрішній порт 8081, передаючі всі особливі й потрібні хедери.

3.2.2.4 Вхідні та вихідні дані у вигляді таблиці з наведенням типів, адрес, дозволів та ін. з необхідними коментарями (Таблиця 3.2)

Таблиця 3.2 – Опис вхідних даних

Метод	URI	Опис	Параметри	Тіло	Аутентифікація
POST	/checkPhoneNumber	Перевірити, чи пройшов реєстрацію користувач з вказаним phoneNumber	-	PhoneNumberDTO phoneNumberDTO	ні
POST	/sendCode	Відправити новий тимчасовий пароль доступу до аккаунту користувачу	-	CodeUpdateDTO codeUpdateDTO	так
POST	/sendMessage	Відправити повідомлення користувачу	-	Map<String, String> message	ні

Закінчення таблиці 3.2

POST	/sendMessageInGroup	Відправити повідомлення в відповідну групу	-	Map<String, String> message	ні

Таблиця 3.2 – Опис вихідних даних

Метод	**URI**	**Тип повертаємого значення**
POST	/checkPhoneNumber	ResponseEntity<Boolean>
POST	/sendCode	ResponseEntity<Void>
POST	/sendMessage	ResponseEntity<Void>
POST	/sendMessageInGroup	ResponseEntity<Void>

4 РОЗРОБКА ВЕБ-СЕРВІСУ ІНТЕРАКТИВНОГО КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

4.1 Технічні вимоги

4.1.1 Найменування і призначення об’єкту професійної діяльності.

Веб-сервіс інтерактивного користувацького інтерфейсу, який реалізує взаємодію з користувачем. В своїй роботі використовує веб-сервіс обробки клієнтських запитів, для отримання актуальної інформації щодо стану товарного складу, та надає користувачам можливість взаємодіяти з контентом на сайті.

Основне функціональне призначення полягає в забезпеченні зручного візуального доступу до операцій інтернет-магазину.

4.1.2 Вимоги до функціонування об’єкта професійної діяльності.

Далі наведені вимоги до функціонування об’єкта професійної діяльності:

- головна сторінка (сторінка з найважливішим контентом, рекламою та банерами);
- навігаційна панель (відповідає за зручне переміщеня поміж сторінками)
- сторінка пошуку товарів за назвою (дозволяє знаходити проудкти по запиту);
- сторінка персонального кабінету (відображення персональної інформації опираючись на аккаунт користувача);
- сторінка опису (опис сайту на якому знаходиться користувач).

4.1.3 Структурні вимоги до об’єкту професійної діяльності.

Далі наведені структурні вимоги до об’єкту професійної діяльності:

- модуль навігації (відповідає за переміщення користувача між сторінками);
- модуль валідації (відповідає за перевірку значень які вводить користувач);

- модуль комунікації з сторонніми сервісами (відповідальний за отримання актуальної інформації, з різноманітних ресурсів);
- модуль фільтрації (відповідає за графічну та функціональну роботу фільтрів);
- модуль рекламний (відповідає за відображення акційних пропозицій та рекламних банерів);
- модуль оформлення замовлень (відповідає за збір інформації для оформлення замовлення);
- модуль персонального кабінету (відповідає за відображення персональної сторінки користувача).

4.1.4 Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами Системи.

Обмін між внутрішніми компонентами системи повинен проводитись, з зберіганням інформації в local storage, а також в cookie файлах. Окрім цього існує система передачі інформації за допомогою вкладеності компонентів в Virtual-DOM дереві.

4.1.5 Вимоги до характеристик взаємозв'язків створюваної Системи із суміжними Системами.

Спілкування з суміжними системами - відбувається виключно за допомогою HTTP протоколу з покриттям SSL, для шифрування та захисту інформації. Обмін даними виконується в форматі JSON.

4.1.6 Межі розвитку.

При розширенні системи, є можливість додавання необмеженної кількості графічних та функціональних модулів. Обмеженням виступає функціональність браузеру, швидкість інтернет з’єднання, об’єм оперативної пам’яті пристрою з якого здійснюється перегляд сторінки.

В перспективі розширення, рекомендовано додати:

- модуль стеження (відповідальний за відображення актуального місцезнаходження посилки);
- модуль рекомендаційної системи (система рекомендацій, яка пропонує товари на основі минулих замовлень);
- модуль вакансій (можна розміщувати актуальні вакансії, та запрошення на роботу нового персоналу).

4.1.7 Показники призначення

- завантаження головної сторінки до 3с;
- пошук доступних відділень Пошти до 2с;
- заванатаження сторінки з продуктами до 5с;
- завантаження персональної сторінки до 5с.

4.2 Розробка апаратної частини

4.2.1 Обгрунтування технічних характеристик програми

4.2.1.1 Опис призначення вхідних і вихідних даних

Вхідні дані поступаючі в веб-сервіс інтерактивного користувацького інтерфейсу мають інформаційний характер та слугують для подальшого графічного відображення на сторінці сайту. Таким чином, веб-сервіс інтерактивного користувацького інтерфейсу перетворює сухі дані з JSON формата, які мають нечитабельний формат для пересічних користувачів, в графіку, яка відображається в браузері.

Вихідними даними - являються запити на суміжні сервіси для виконання на них відповідних встановлених дій. Перед відправкою запита потрібно впевнитись в правильному форматі повідомлення, щоб серверна сторона могла коректно обробити ці дані.

4.2.1.2 Опис і обґрунтування вибору складу технічних і програмних засобів, що використовує програма.

В основі програмної реалізації клієнтської сторони лежить фреймворк Angular, рішення на користь якого було прийнято з огляду на низку його стратегічних переваг. Його архітектура, що базується на модульному компонентному підході, дозволяє структурувати додаток у вигляді незалежних та перевикористовуваних блоків, що суттєво спрощує подальший супровід і розширення функціонала проєкту. Застосування мови TypeScript, яка є надбудовою над JavaScript, вносить у процес розробки статичну типізацію, що допомагає виявляти потенційні помилки ще на етапі кодування та значно підвищує загальну стабільність кодової бази. Окрім цього, розвинена екосистема Angular надає розробникам готовий набір інструментів, включно з розвиненою системою навігації та засобами для керування даними, що оптимізує створення складних односторінкових додатків, типових для платформ електронної комерції, та забезпечує високу швидкість відгуку для кінцевого користувача.

Для управління стилями візуального інтерфейсу було прийнято рішення використовувати препроцесор SCSS, що є синтаксично досконалішим розширенням стандартного CSS. Такий вибір продиктований необхідністю створення масштабованої та легко підтримуваної архітектури стилів, що є критичним для великих проєктів. SCSS надає ключові інструменти, відсутні в нативному CSS, зокрема змінні, які дозволяють централізовано зберігати й керувати дизайн-токенами, гарантуючи стилістичну цілісність усього додатку. Можливість вкладеності селекторів значно покращує читабельність коду, оскільки його структура логічно повторює ієрархію HTML-документа. Нарешті, використання міксинів (mixins) та функція імпорту окремих файлів сприяють модульності та дозволяють ефективно перевикористовувати блоки стилів, що мінімізує дублювання коду та спрощує його подальшу модифікацію.

4.2.2 Опис розробленої програми

4.2.2.1 Програмне забезпечення й мова програмування, необхідні для функціонування програми

Для запуску програми потрібно завчасно встановити певний список ПО. Далі наведено список програмного забезбечення яке потрібно встановити:

- Node.js (кросплатформенне середовище виконання JavaScript, що базується на рушії V8 від Google Chrome. Його наявність є критично важливою, оскільки він забезпечує виконання JavaScript-коду на стороні сервера, що є фундаментальним для роботи інструментів, що підтримують розробку Angular-додатків. Зокрема, Node.js є необхідною передумовою для функціонування менеджера пакетів npm);
- Npm або Yarn (пакетні менеджери для JavaScript, які інтегровані з Node.js. Вони виконують функцію централізованого управління залежностями проекту, дозволяючи автоматизувати процес завантаження, встановлення, оновлення та видалення зовнішніх бібліотек та модулів, які необхідні для коректної роботи Angular-додатку. Ці інструменти забезпечують цілісність та консистентність залежностей проекту);
- Angular CLI (офіційний інструмент командного рядка, розроблений командою Angular для автоматизації типових завдань розробки. Він надає набір команд для ініціалізації нових проектів, генерації компонентів, сервісів, модулів та інших елементів Angular, а також для компіляції (build) проекту для розгортання та виконання юніт-тестів. Angular CLI значно підвищує продуктивність розробника, стандартизуючи робочі процеси та зменшуючи кількість ручних операцій).

4.2.2.2 Опис логічної структури програми, зв'язок з іншими програмами. Надати графічне представлення алгоритму програми

Фронтенд-частина веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів реалізована за допомогою Angular — сучасного фреймворку для створення односторінкових веб-застосунків. В основі структури клієнтської частини лежить компонентний підхід, тобто застосунок поділено на незалежні функціональні блоки, кожен з яких відповідає за певний сегмент взаємодії з користувачем (наприклад, каталог товарів, кошик, персональний кабінет).

Уся інформація, з якою працює клієнтський застосунок, зберігається у спеціальних сервісах, які відповідають за зберігання, обробку та передачу даних між частинами інтерфейсу та зовнішніми джерелами. Користувач взаємодіє з інтерфейсом: виконує пошук товарів, додає товари в кошик, заповнює дані замовлення, реєструється, переглядає історію покупок. Кожна з цих дій активує звернення до відповідного сервісу.

Зв'язок фронтенду з серверною частиною (Backend) забезпечується через мережеві запити. Angular-застосунок надсилає HTTP-запити на REST API, реалізоване в сервері, й у свою чергу отримує у відповідь дані у форматі «JSON». Це можуть бути списки товарів, інформація про сам товар чи також повідомлення про помилки. Обмін даними відбувається виключно через захищений протокол HTTPS.

Клієнтська частина також опосередковано взаємодіє з Telegram-ботом. Після введення номера телефону у боті, користувач отримує одноразовий код доступу з 6 символів. Цей код вводиться у відповідне поле на веб-сайті. Angular-застосунок пересилає його на сервер для перевірки, після чого, у разі успішної верифікації, користувачу дозволяється доступ до особистого кабінету. Таким чином, Telegram-бот виконує функцію безпечної аутентифікації, а Angular забезпечує передачу коду та реакцію на результат перевірки.

Кожен функціональний блок на клієнтській стороні логічно ізольований, але має змогу взаємодіяти з іншими через спільні сервіси та механізм маршрутизації. Такий підхід дозволяє реалізувати плавну та логічну взаємодію між усіма частинами інтерфейсу без дублювання даних.

4.2.2.3 Виклик і завантаження програми (спосіб виклику програми з відповідного носія даних, вхідні точки в програму

Для запуску сервісу потрібно виконати натсупний алгоритм дій:

- перехід до директорії проєкту (За допомогою команди cd перейдіть у кореневу папку вашого Angular-проєкту. Це та папка, де знаходяться файли package.json, angular.json та папка src);

- запустіть сервер розробки Angular CLI (команда “ng serve”, Вона компілює ваш Angular-код та Запускає локальний сервер).

4.2.2.4 Вхідні та вихідні дані у вигляді таблиці з наведенням типів, адрес, дозволів та ін. з необхідними коментарями

Вхідні та вихідні дані - співпадають з вхідними та вихідними даними розробка веб-сервісу для обробки клієнтських запитів. Структура запитів підходить саме для використання розробленого в веб-сервісі для обробки клієнтських запитів REST API. Та представляє собі його клієнтську сторону.

4.2.2.5 Опис інтерфейсу програми

Інтерфейс програми структурно складається з декількох основних сторінок, та проміжних між ними.

Головна сторінка програми є центральним елементом інтерфейсу програми та першою точкою взаємодії користувача із сайтом. Тому вона повинна бути зрозумілою, зручною у користуванні та водночас інформативною. На головній сторінці розміщуються головні події на сайті, актуальна реклама, акції а також інформаційні банери. Окрім цього головна сторінка має створювати зручний інтерфейс для навігації по сайту, який повинен бути інтуїтивно зрозумілим навіть людям з поганим зором (Рисунок. 4.1 – 4.2).

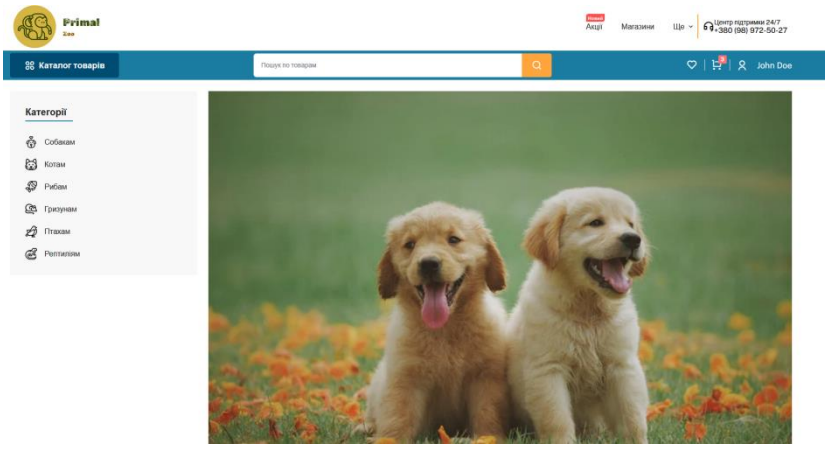


Рисунок. 4.1 – Головна сторінка веб-сайту

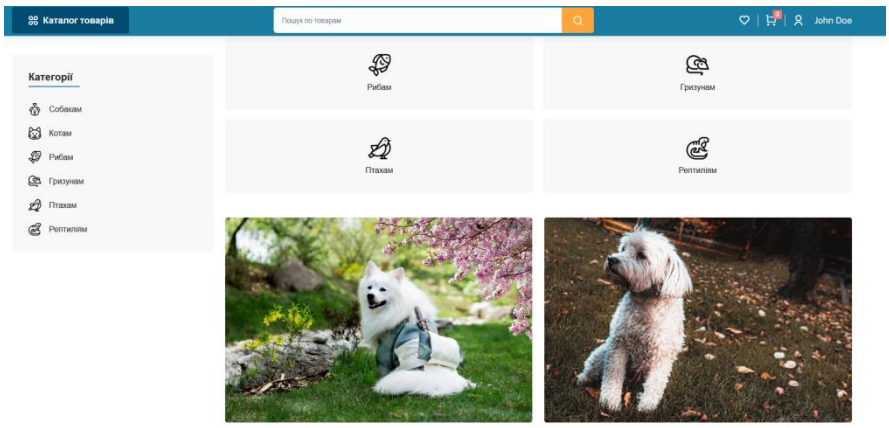


Рисунок. 4.2 – Головна сторінку веб-сайту з можливістю обрати товар для певної тварини

Сторінка з товарами є однією з ключових частин програми, адже саме тут користувачі можуть ознайомитися з доступними позиціями, порівняти їх, а також обрати потрібні товари відповідно від своїх уподобань. Основна мета – забезпечити швидкий, зручний та ефективний пошук товару серед усього асортименту. Головна функціональність саме цієї сторінки полягає у можливості фільтрування товарів, за наявними критеріями в базі даних (Рисунок. 4.3).

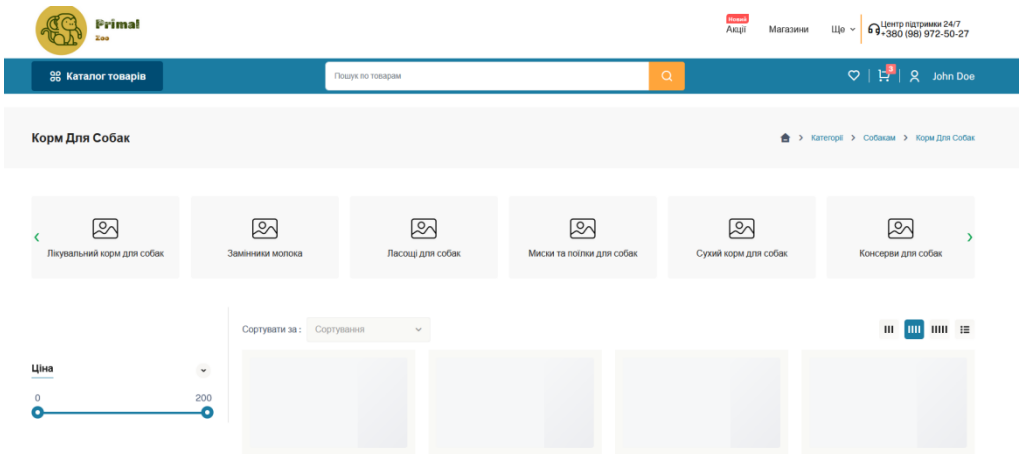


Рисунок. 4.3 – Сторінка з доступними товарами та категоріями

Ще один із найважливіших елементів зручного користування веб-додатком – це завжди доступний кошик, що розміщений у верхній навігаційній панелі. Його головна функція – це забезпечити швидкий доступ

до списку товарів, які користувач додав до покупки, незалежно від того, на якій сторінці сайту зараз користувач знаходиться (Рисунок. 4.4).

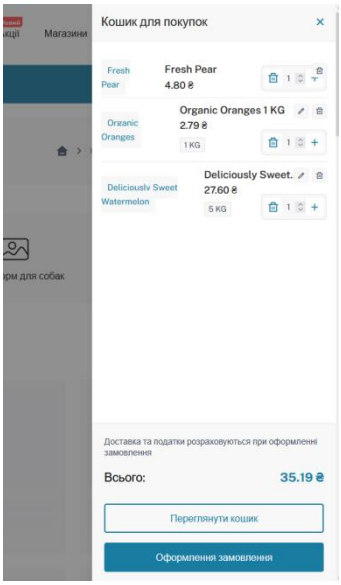


Рисунок. 4.4 – Доступний кошик у верхній навігаційній панелі

Персональна сторінка з панеллю користувача – це індивідуальний розділ сайту, доступний після входу в систему. Вона є ключовим моментом взаємодії користувача із сайтом, оскільки зручно дозволяє керувати своїми замовленнями, налаштуваннями та переглядати особисту інформацію. Головна мета цієї сторінки – забезпечити повний контроль над замовленнями та персональними даними, а також гнучкий моніторинг власної активності на веб-сайті (Рисунок. 4.5).

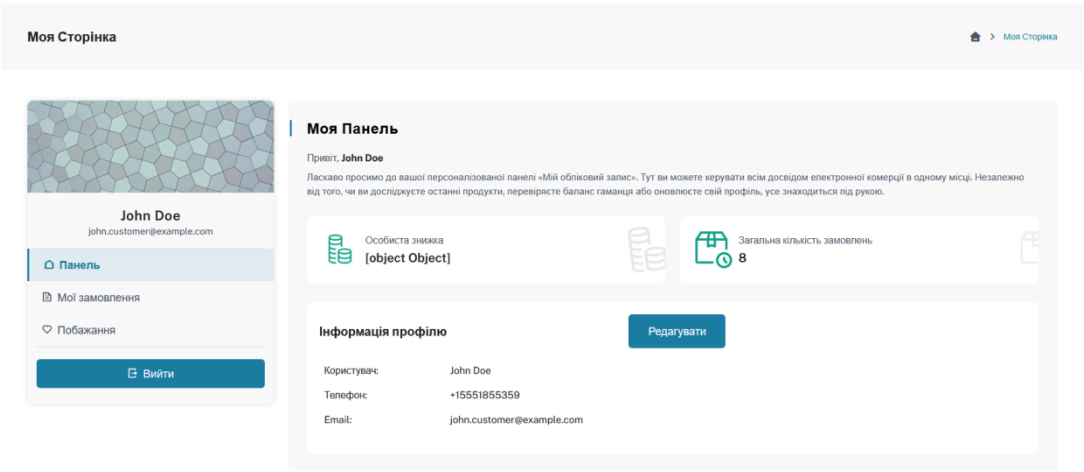


Рисунок. 4.5 – Персональна сторінка

Сторінка з детальним відображеням інформації про продукт – являється дуже важливою при більш детальному розгляданні товару, покупцем (Рисунок 4.6)

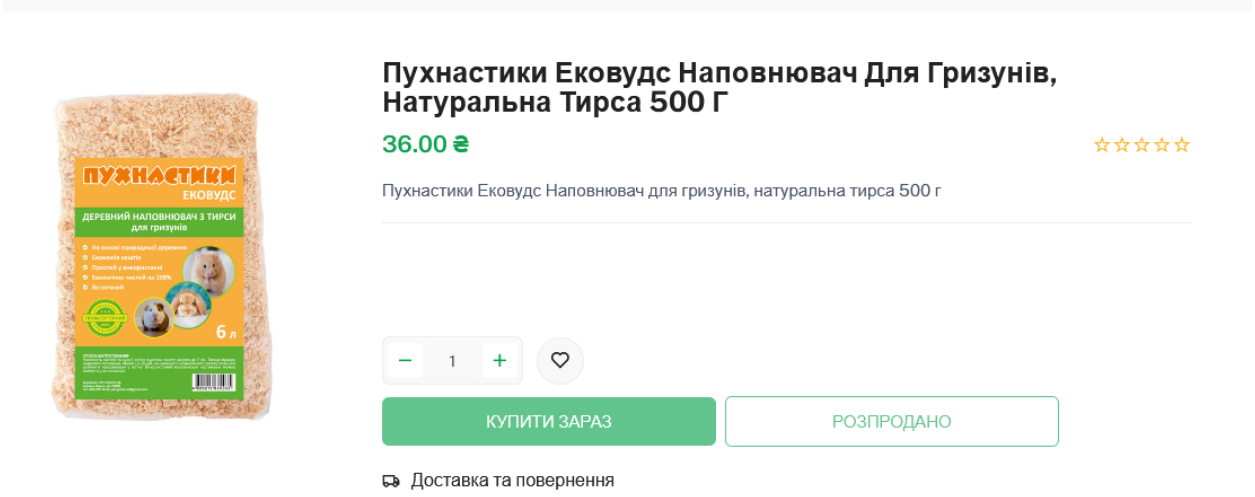


Рисунок. 4.6 – Сторінка продукту

При підборі товарів – користувач має отримати змогу передивлятися весь широкий асортимент товарів всередині магазину. Наведені блоки мають бути візуально зрозумілими, та реагувати на натискання користувача, переносячи його на сторінку товару (Рисунок 4.7)

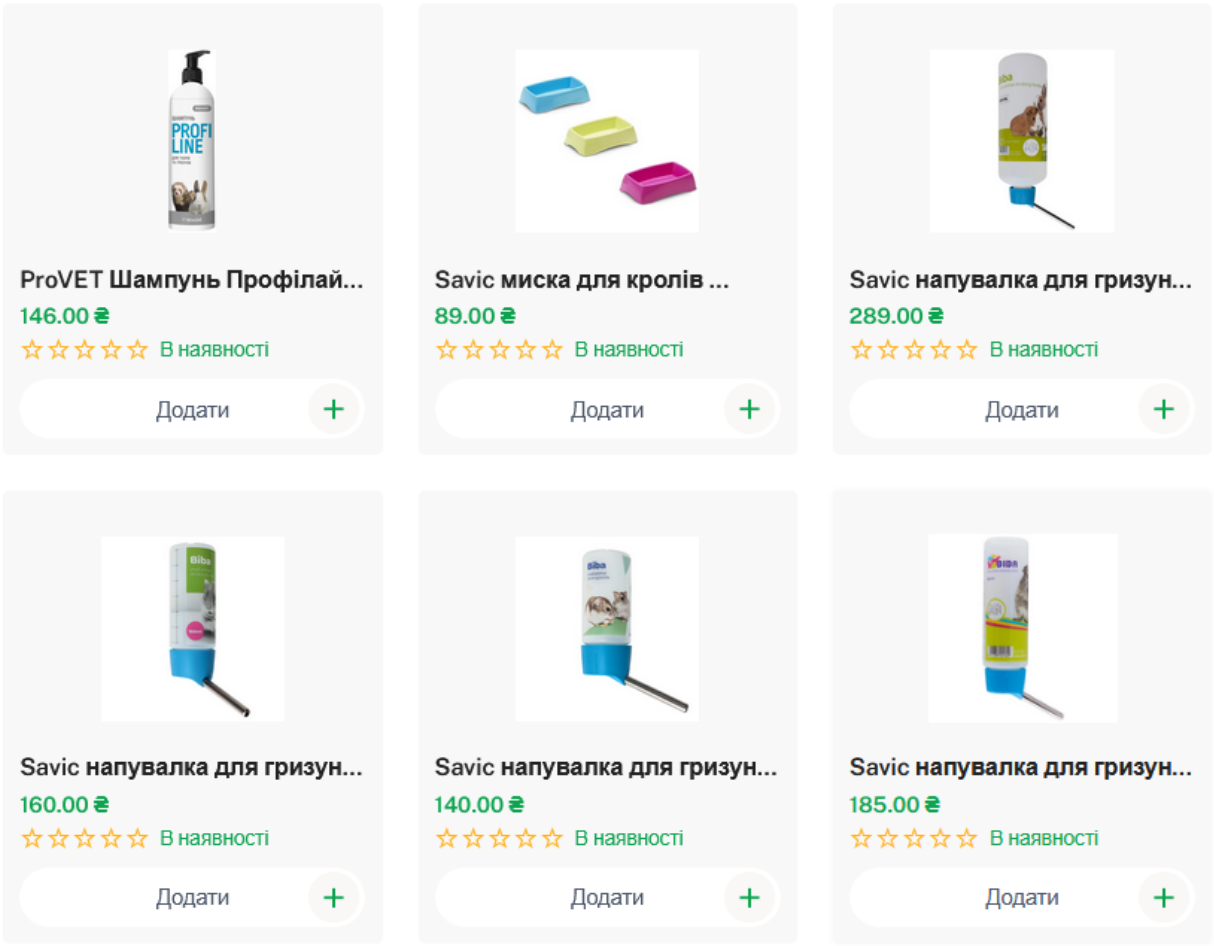


Рисунок 4.7 – Сторінка списку продуктів

ВИСНОВКИ

Було реазлізовано створення веб-сервісу обслуговування клієнтів інтернет-магазину зоотоварів. Було створено:

- веб сервіс для обробки клієнтських запитів;
- клієнтський сервіс для взаємодії з користувачами через месенджер Telegram;
- веб-сервіс інтерактивного користувацького інтерфейсу.

Ця система потенційно може стати основною платформою торгівлі для середнього бізнесу. Було створено відмовостійку систему для взаємодії з користувачами за допомогою веб-інтерфейсів.

Кожен веб-сервіс має свої можливості комунікації між суміжними системами, разом реалізовуючи мікросервісну екосистему. Розгортання цієї системи на пристроях які співпадають по технічним характеристикам з вимогами системи - гарантує швидку та надійну роботи всієї системи.

Було покращено роботу об’єкту впроваждення, шляхом модернізації взаємодії з користувачами, за допомогою веб комунікації.

Розширення цієї системи може призвести до створення додаткових функціональних модулей, що позитивно відобразиться на функціонуванні об’єкту впроваждення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. AChernega. React vs Vue vs Angular. *Хабр*. URL: https://habr.com/ru/companies/auriga/articles/703836/ (дата звернення: 22.02.2025).

2. Що таке rest api: основні принципи і як воно працює. *FoxmindEd*. URL: https://foxminded.ua/ru/chto-takoe-rest-api/ (дата звернення: 01.02.2025).

3. Що таке Wordpress: огляд найбільш популярної CMS | HOSTiQ Wiki. *HOSTiQ Wiki*. URL: https://hostiq.ua/wiki/ukr/wordpress-review/?gad_source=1&gad_campaignid=22221308613&gclid=CjwKCAjwgb_CBhBMEiwA0p3oOFgfrBLPcehnIBYHdWAqSELfbJgrhmXGMgyfCNw7oBzydQRVrl6VqxoCPhUQAvD_BwE (дата звернення: 04.02.2025).

4. Dudka M. Spring JDBC проти Spring Data JPA. *DOU*. URL: https://dou.ua/forums/topic/46082/ (дата звернення: 01.02.2025).

5. Guarana. What Are the Back-End Languages? Python, Java, PHP. *iOS & Android Mobile App Developers Toronto & Montréal | Guaraná*. URL: https://guarana-technologies.com/blog/what-are-the-backend-languages (дата звернення: 02.02.2025).

6. HTTP і HTTPs: що це таке і в чому різниця | HOSTiQ Wiki. *HOSTiQ Wiki*. URL: https://hostiq.ua/wiki/ukr/http-https/?gad_source=1&gad_campaignid=22221308613&gclid=CjwKCAjwgb_CBhBMEiwA0p3oODMHryxcHdiFOpnbICHhe1fcWErW3ExTEJdI2-yQaAG06MukCx2osBoCSPgQAvD_BwE (дата звернення: 05.02.2025).

7. OWASP Top Ten | OWASP Foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-top-ten/ (date of access: 02.02.2025).

8. PostgreSQL и MySQL – різниця між системами управління реляційними базами даних (RDBMS) – AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/ru/compare/the-difference-between-mysql-vs-postgresql/ (дата звернення: 03.02.2025).

9. Burd B. Java For Dummies (Java for Dummies). For Dummies, 2006. 384 с.

10. Walls C. Spring in Action, Sixth Edition. Manning Publications Co. LLC, 2022.

11. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 432 с.

12. Java Platform, Standard Edition Documentation - Releases. *Oracle Help Center*. URL: https://docs.oracle.com/en/java/javase/ (дата звернення: 02.02.2025).

13. Spring Framework Documentation : Spring Framework. *Spring | Home*. URL: https://docs.spring.io/spring-framework/reference/index.html (дата звернення: 07.02.2025).

14. Telegram Bot API. *Telegram APIs*. URL: https://core.telegram.org/bots/api (дата звернення: 09.02.2025).

15. Angular. *Home • Angular*. URL: https://angular.dev/overview (дата звернення: 05.02.2025).

16. 42+ E-commerce Growth Statistics 2025 (Key Data Revealed). *Yaguara*. URL: https://www.yaguara.co/e-commerce-growth-statistics/ (дата звернення: 05.02.2025).

17. 51 ECommerce Statistics In 2025 (Global And U.S. Data) | SellersCommerce. *SellersCommerce*. URL: https://www.sellerscommerce.com/blog/ecommerce-statistics/ (дата звернення: 02.02.2025).

18. Geuens R. E-commerce growth: How much has e-commerce grown over the years?. *SOAX*. URL: https://soax.com/research/growth-ecommerce (дата звернення: 02.02.2025).

19. Global Ecommerce Statistics: Trends to Guide Your Store in 2025 - Shopify Canada. *Shopify*. URL: https://www.shopify.com/ca/enterprise/blog/global-ecommerce-statistics (дата звернення: 05.02.2025).

20. 40 Essential Ecommerce Statistics to Know in [2025 Edition]. *Digital Minds BPO*. URL: https://digitalmindsbpo.com/blog/ecommerce-statistics/ (дата звернення: 01.02.2025).

21. Ukraine eCommerce Trends: Market Insights & Forecast. *Promodo | Digital Marketing Agency*. URL: https://www.promodo.com/blog/research-of-the-ukrainian-ecommerce-market (дата звернення: 01.02.2025).

22. frwhCe. In 2024, Ukrainians spent UAH 239 billion on online purchases. This is 25% more than last year. *dev.ua*. URL: https://dev.ua/en/news/u-2024-mu-rotsi-ukraintsi-vytratyly-na-onlain-pokupky-na-25-bilshe-nizh-torik-1734683726 (дата звернення: 02.02.2025).

23. 10 Key Figures About Ukrainian eCommerce in 2024 | Promodo Research. *Promodo | Digital Marketing Agency*. URL: https://www.promodo.com/blog/10-key-figures-about-ukrainian-ecommerce (дата звернення: 02.02.2025).

24. ROMI 2024: Дослідження українського ринку e-commerce. Що чекає на ринок?. *newage*. URL: https://newage.agency/uk/blog-uk/romi-2024-doslidzhennia-ukrainskoho-rynku-e-commerce-shcho-chekaie-na-rynok/ (дата звернення: 02.02.2025).

25. 80+ Customer Service Statistics You Need to Know in 2025 | AmplifAI. *The Contact Center AI Platform | CX and Performance Management | QA and Coaching | AmplifAI*. URL: https://www.amplifai.com/blog/customer-service-statistics (дата звернення: 16.06.2025).

26. Importance Of Customer Service For Ecommerce: Meteor Space. *Meteor Space | Warehousing & Order Fulfillment Services in Ireland & Europe: Meteor Space*. URL: https://www.meteorspace.com/importance-of-customer-service-for-ecommerce-statistics-you-should-know/ (дата звернення: 02.02.2025).

27. Watts A. 40+ Statistics That Highlight the Importance of Customer Experience. *Training that's built for your frontline | eduMe*. URL: https://www.edume.com/blog/customer-experience-statistics (дата звернення: 02.02.2025).

28. Importance of Customer Service in eCommerce Business - weDevs. *weDevs*. URL: https://wedevs.com/blog/286250/importance-of-customer-service-in-ecommerce/ (дата звернення: 01.02.2025).

29. Brown E. J. 10 Retail Customer Service Stats You Need To Know | CM.com. *CM.com*. URL: https://www.cm.com/en-gb/blog/10-retail-customer-service-stats-you-need-to-know/ (дата звернення: 04.02.2025).

ДОДАТОК А

```java
@RestController
@RequestMapping("/api/orders")
public class OrderController {

    @Autowired
    OrderService orderService;

    @Autowired
    UserEntityService userEntityService;



    @PostMapping
    public ResponseEntity<String> createOrder(@RequestBody OrderPostDTO orderPostDTO,
                                              @AuthenticationPrincipal UserEntity userEntity){

        OrderEntity createdOrder = orderService.createOrder(orderPostDTO, userEntity);


        return ResponseEntity.status(HttpStatus.CREATED)
                .body(createdOrder.getOrderId());
    }

    @GetMapping
    public ResponseEntity<List<OrderEntity>> getOrders(@AuthenticationPrincipal UserEntity userEntity){
        return ResponseEntity.ok(orderService.getByUser(userEntity));
    }

}
```

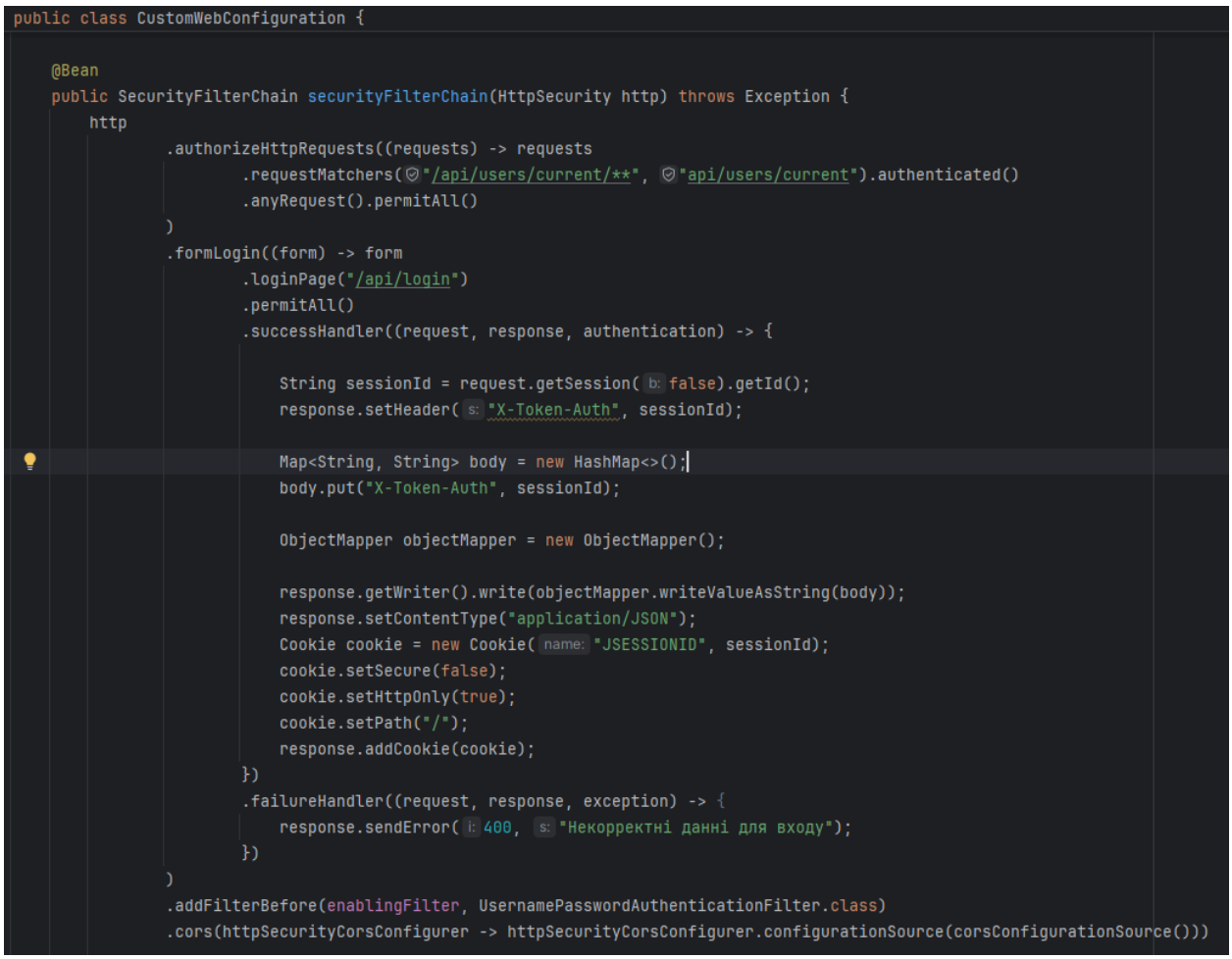
Рисунок 1 – Код класу OrderController

```java
public class ProductPageController {

    @GetMapping("/search")
    public ResponseEntity<ListDTO<ProductDTO>> getFilteredProductsInSearch(@RequestParam(value = "page", defaultValue = "1") int page,
                                                                           @RequestParam(value = "pageSize", defaultValue = "20") int pageSize,
                                                                           @RequestParam(value = "attributes", defaultValue = "") String attributes,
                                                                           @RequestParam(value = "category", defaultValue = "") String category,
                                                                           @RequestParam(value = "minPrice", defaultValue = "-1") int minPrice,
                                                                           @RequestParam(value = "maxPrice", defaultValue = "" + Integer.MAX_VALUE) int maxPrice,
                                                                           @RequestParam(value = "sortBy", defaultValue = "") String sortType,
                                                                           @RequestParam(value = "search", defaultValue = "") String search){
        List<Product> productsCsv = productPageService.getFilteredProducts(attributes, minPrice, maxPrice, category, sortType, search);

        int lastPage = (productsCsv.size() / pageSize) + 1;
        int plusPage = lastPage - page;

        plusPage = Math.min(plusPage, 2);

        page = Math.max(page, 0);

        Map<String, CategoryDTO> categoryDTOS = new HashMap<>();

        for (Product product : productsCsv){
            String cata = product.getUnformattedCategories().getFirst();
            if (!categoryDTOS.containsKey(cata)){
                categoryDTOS.put(cata, new CategoryDTO(cata));
            }
        }

        return ResponseEntity.ok(new ListDTO<>(productsCsv
                .stream() Stream<Product>
                .skip(Math.min(((long) (page - 1) * pageSize), productsCsv.size()))
                .limit(pageSize)
                .map(product -> {
                    List<CommentEntity> comments = commentService.getCommentsByBarcode(product.getBarcode());
                    return new ProductDTO(product, comments);
                }) Stream<ProductDTO>
                .toList(),
                plusPage,
                lastPage)
                .setCategoryDTOS(categoryDTOS.values().stream().toList()));
    }
```

Рисунок 2 – Приклад фільтрації ProductPageController

```
public class CSVParser {

    private static final String FILE_NAME = "snapshot_db.csv";
    //Самая первая страничка - 1 нумерация идет с 1

    public static  <T extends Mappable> T mapObject(String line, T objectToMap, String[] tableHeaders){

        String values[] = line.split( regex: ";"); // Получаем каждое поле в формате "Значение 1"
        removeDoubleQuotes(values);

        for(int i = 0; i < tableHeaders.length; i++) {
            if (values[i].trim().isEmpty()) continue;

            objectToMap.mapField(tableHeaders[i], values[i]);
        }

        return objectToMap;
    }

    public static String[] getHeaders(){
        try (BufferedReader bufferedReader = new BufferedReader(new FileReader(FILE_NAME))){
            String firstLine = bufferedReader.readLine().substring( beginIndex: 1);

            String[] headers = firstLine.split( regex: ";");

            removeDoubleQuotes(headers);

            System.out.println("Headers of CSV: ");
            for (String header : headers){
                System.out.print(header + ", ");
            }
            System.out.println();

            return headers;
        } catch (Exception e){
            System.out.println("Error in reading file!");
        }
        return null;
    }
```

Рисунок 3 – Приклад методів для читання CSV файлів



```
public class CustomWebConfiguration {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .authorizeHttpRequests((requests) -> requests
                        .requestMatchers("/api/users/current/**", "api/users/current").authenticated()
                        .anyRequest().permitAll()
                )
                .formLogin((form) -> form
                        .loginPage("/api/login")
                        .permitAll()
                        .successHandler((request, response, authentication) -> {

                            String sessionId = request.getSession( b: false).getId();
                            response.setHeader( s: "X-Token-Auth", sessionId);

                            Map<String, String> body = new HashMap<>();
                            body.put("X-Token-Auth", sessionId);

                            ObjectMapper objectMapper = new ObjectMapper();

                            response.getWriter().write(objectMapper.writeValueAsString(body));
                            response.setContentType("application/JSON");
                            Cookie cookie = new Cookie( name: "JSESSIONID", sessionId);
                            cookie.setSecure(false);
                            cookie.setHttpOnly(true);
                            cookie.setPath("/");
                            response.addCookie(cookie);
                        })
                        .failureHandler((request, response, exception) -> {
                            response.sendError( i: 400,  s: "Некорректні данні для входу");
                        })
                )
                .addFilterBefore(enablingFilter, UsernamePasswordAuthenticationFilter.class)
                .cors(httpSecurityCorsConfigurer -> httpSecurityCorsConfigurer.configurationSource(corsConfigurationSource()))
```

Рисунок 4.1 – Приклад налаштування WebSecurity

```
                .addFilterBefore(enablingFilter, UsernamePasswordAuthenticationFilter.class)
                .cors(httpSecurityCorsConfigurer -> httpSecurityCorsConfigurer.configurationSource(corsConfigurationSource()))

                .csrf(AbstractHttpConfigurer::disable)
                .logout(logout -> logout
                        .logoutUrl("/api/logout")
                        .invalidateHttpSession(true)
                        .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
                        .logoutSuccessHandler(new LogoutSuccessHandler() {
                            @Override  no usages
                            public void onLogoutSuccess(HttpServletRequest request, HttpServletResponse response, Authentication authentication) throws IOException, ServletExcep
                                response.setStatus(HttpStatus.NO_CONTENT.value());
                            }
                        })
                        .permitAll()
                ).exceptionHandling(httpSecurityExceptionHandlingConfigurer -> {
                    httpSecurityExceptionHandlingConfigurer.authenticationEntryPoint( (request, response, authException) -> {
                        response.setStatus(403);
                        response.getWriter().write( s: "Authorize first!");
                    });
                    httpSecurityExceptionHandlingConfigurer.accessDeniedHandler((request, response, accessDeniedException) -> {
                        response.setStatus(403);
                        response.getWriter().write( s: "Authorize first!");
                    });
                });

        return http.build();
    }
```

Рисунок 4.2 – Приклад налаштування WebSecurity

```java
@Entity  24 usages
public class CommentEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    String barcode;  3 usages
    @ManyToOne  3 usages
    @JoinColumn(name = "user_id")
    UserEntity userEntity;

    String text;  3 usages
    int rating;  3 usages

    Date date = new Date();  2 usages


    public CommentEntity(String barcode, UserEntity userEntity, String text, int rating) {  1 usage
        this.barcode = barcode;
        this.userEntity = userEntity;
        this.text = text;
        this.rating = rating;
    }

    public CommentEntity() {
    }
```

Рисунок 5 – Приклад сутності для збереження в БД

```java
@Override  no usages
public void onUpdateReceived(Update update) {
    System.out.println("UPDATED");
    if (update.hasMyChatMember()){

        ChatMemberUpdated chatMemberUpdated = update.getMyChatMember();
        Chat chat = chatMemberUpdated.getChat();
        String newStatus = chatMemberUpdated.getNewChatMember().getStatus();

        if ("member".equals(newStatus)) {
            // Бота добавили в группу
            Group group = new Group(chat.getId(), chat.getTitle());
            groupRepo.save(group);

        } else if ("kicked".equals(newStatus) || "left".equals(newStatus)) {
            // Бота удалили из группы
            groupRepo.deleteByChatId(chat.getId());
        }

    }
    else if (update.hasMessage()){
        if (!customerService.customerExist(update.getMessage().getChatId())){
            // Если он еще не начал процесс регистрации
            System.out.println(update.getMessage().getChatId() + " Unregistrated");
            processUnregistrated(update);
        }
        else if (!customerService.customerIsFullfield(update.getMessage().getChatId())){
            // Если в процессе
            System.out.println(update.getMessage().getChatId() + " not fullfieled");
            processRegistrateNow(update, customerService.getCustomerByChatId(update.getMessage().getChatId()));
        }
        else{
            // Если он уже зарегестрирован
            System.out.println(update.getMessage().getChatId() + " Registrated");
            processRegistrated(update);
        }
    }

}
```

Рисунок 6 – Приклад обробки повідомлень від користувачів Telegram сервісу

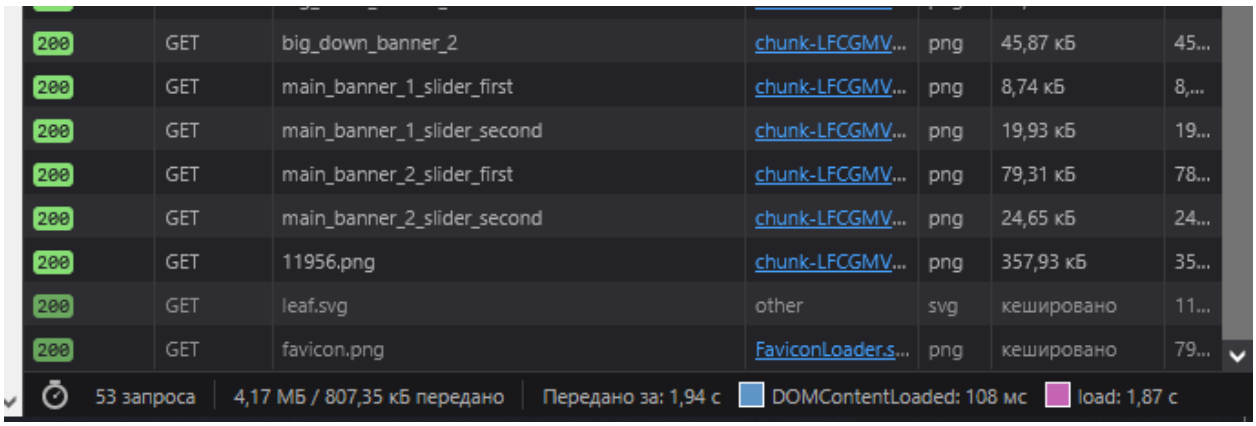


Рисунок 7 – Показники швидкості завантаження веб-інтерфейсу користувача

Рисунок 8 – Показники швидкості завантаження інформації з веб-сервісу з обслуговування клієнтських запитів

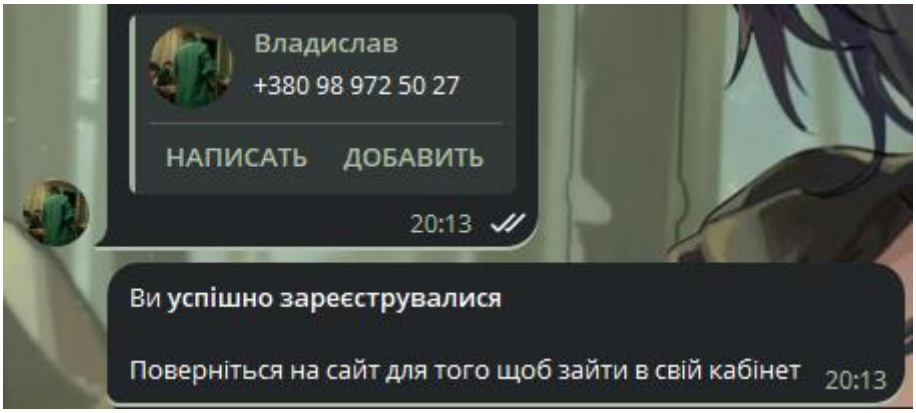


Рисунок 9 – Показники швидкості відповіді бота на повідомлення.

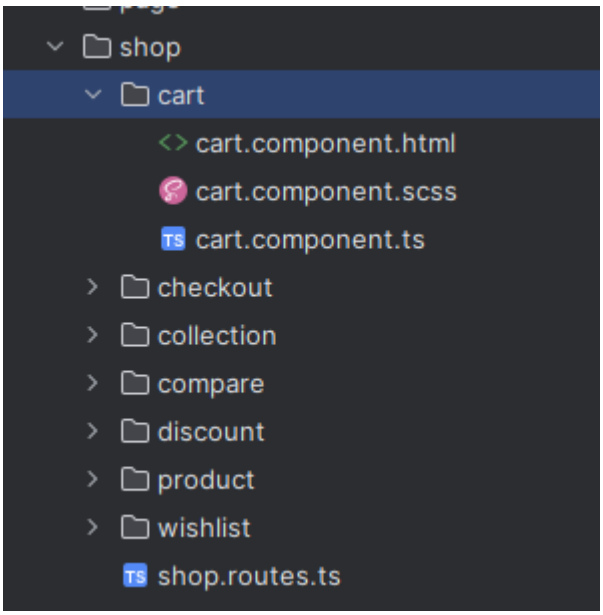
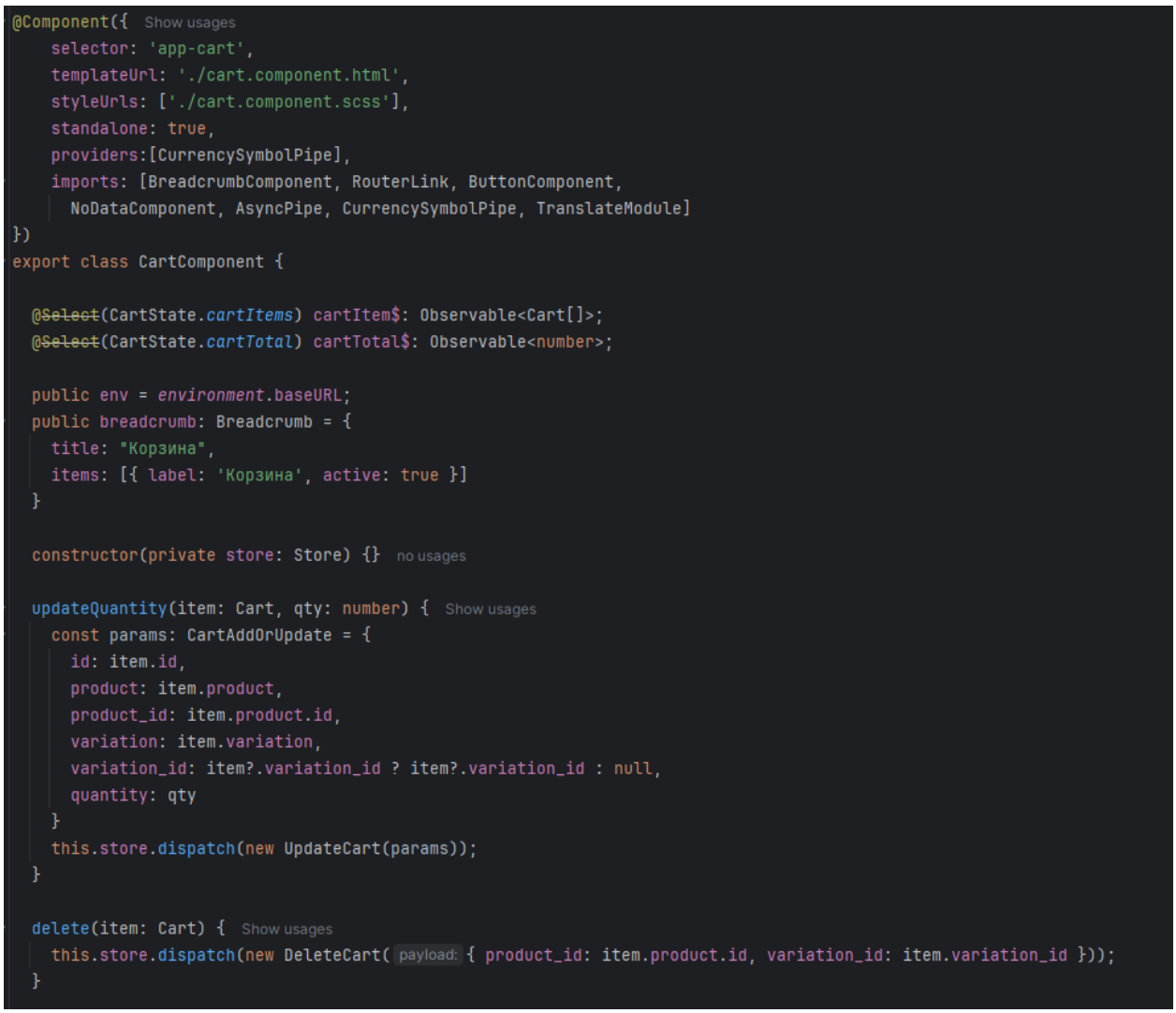


Рисунок 10 – Приклад файлової структури Angular

```
<app-breadcrumb [breadcrumb]="breadcrumb"></app-breadcrumb>
<section class="cart-section section-b-space">
  <div class="container-fluid-lg">
    @if ((cartItem$ | async)?.length) {
      <div class="row g-xl-5 g-sm-4 g-3">
        <div class="col-xxl-9 col-xl-8">
          <div class="cart-table">
            <div class="table-responsive desktop">
              <table class="table">
                <tbody>
                  @for (item of cartItem$ | async; track item) {
                    <tr class="product-box-contain">
                      <td class="product-detail">
                        <div class="product border-0">
                          <a [routerLink]="['/product/', item.product.slug]" class="product-image">
                                                <img [src]="
                                                        item.product.product_thumbnail
                                                        ? env + 'api/images/' + item.product.product_thumbnail
                                                        : 'assets/images/product.png'" class="img-fluid" alt="product">
                          </a>
                          <div class="product-detail">
                            <ul>
                              <li class="name">
                                <a [routerLink]="['/product/', item.product.slug]">
                                  {{item.variation ? item.variation.name : item.product.name}}
                                </a>
                              </li>
                              <li class="text-content">
                                <h5>
                                  {{ (item?.variation ? item.variation.sale_price : item.product.sale_price) | currencySymbol }}
                                  @if (item.product.is_sale_enable) {
                                    <del class="text-content">{{ (item.variation ? item.variation.price : item.product.price) | currencySymbol }}</del>
                                  }
                                </h5>
                              </li>
                              @if (item.product && item.product.unit) {
                                <li class="text-content">
                                  @if (item.product.is_sale_enable) {
                                    <h6 class="theme-color">
                                      {{ 'you_save' | translate }} : {{ ((item.variation ? item.variation.price
                                      : item.product.price) - (item.variation
```

Рисунок 11 – Приклад html файла компонентної системи Angular



```
@Component({  Show usages
    selector: 'app-cart',
    templateUrl: './cart.component.html',
    styleUrls: ['./cart.component.scss'],
    standalone: true,
    providers:[CurrencySymbolPipe],
    imports: [BreadcrumbComponent, RouterLink, ButtonComponent,
      NoDataComponent, AsyncPipe, CurrencySymbolPipe, TranslateModule]
})
export class CartComponent {

  @Select(CartState.cartItems) cartItem$: Observable<Cart[]>;
  @Select(CartState.cartTotal) cartTotal$: Observable<number>;

  public env = environment.baseURL;
  public breadcrumb: Breadcrumb = {
    title: "Корзина",
    items: [{ label: 'Корзина', active: true }]
  }

  constructor(private store: Store) {}  no usages

  updateQuantity(item: Cart, qty: number) {  Show usages
    const params: CartAddOrUpdate = {
      id: item.id,
      product: item.product,
      product_id: item.product.id,
      variation: item.variation,
      variation_id: item?.variation_id ? item?.variation_id : null,
      quantity: qty
    }
    this.store.dispatch(new UpdateCart(params));
  }

  delete(item: Cart) {  Show usages
    this.store.dispatch(new DeleteCart( payload: { product_id: item.product.id, variation_id: item.variation_id }));
  }
```

Рисунок 12 – Приклад TypeScript файлу для реалізації компонентного функціоналу