Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ЗАМОВЛЕННЯ ТОВАРІВ ІЗ СУПЕРМАРКЕТУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-3
Ругальов В.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст. викл. Кіношенко Д.К.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ругальову Владиславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для замовлення товарів із супермаркету

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, офіційна документація мови Java, офіційна документація мови JavaScript, інтегроване середовище розробки IntelliJ IDEA, програмне забезпечення pgAdmin 4, програмне забезпечення Postman.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих вебзастосунків для замовлення товарів із супермаркету.
2. Аналіз технологій для розробки вебзастосунків.
3. Моделювання роботи вебзастосунку для замовлення товврів із супермаркету.
4. Програмна реалізація вебзастосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми створення вебзастосунків для замовлення товарів із супермаркету. актуальність вибору технологій для розробки сучасних вебзастосунків, постановка задачі, моделювання процесів взаємодії користувача із вебзастосунком, архітектура застосунку, лістинги програмної реалізації, результати тестування.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-15.04.24	
3	Аналіз літератури з досліджуваної проблеми	16.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-25.04.24	
5	Розробка методу	26.04.24-14.05.24	
6	Програмна реалізація	15.05.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	10.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Кіношенко Д.К.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 64 с., 9 табл., 28 рис., 32 джерела.

ВЕБЗАСТОСУНОК, ІНТЕРНЕТ МАГАЗИН, API, МОВА ГІПЕРТЕКСТОВОЇ РОЗМІТКИ HTML, КАСКАДНІ ТАБЛИЦІ СТИЛІВ CSS, МОВА ПРОГРАМУВАННЯ JAVASCRIPT, JAVASCRIPT-БІБЛІОТЕКА REACT, МОВА ПРОГРАМУВАННЯ JAVA, JAVA-ФРЕЙМВОРК SPRING, РЕЛЯЦІЙНА БАЗА ДАНИХ POSTGRESQL.

Об’єктом роботи процес розробки клієнт-серверної архітектури вебзастосунку для замовлення товарів.

Метою роботи є розробка надійного та ефективного інтернет-магазину, який дозволить користувачам замовляти товари з супермаркету.

У процесі роботи над кваліфікаційною роботою, було проведено дослідження існуючих сайтів для замовлення товарів із супермаркету. Створено прикладний програмний інтерфейс серверної частини для взаємодії клієнта з базою даних. Клієнтську програму розроблено у вигляді вебсайту. Побудовано схему бази даних супермаркету.

У результаті роботи створено вебзастосунок з базовим функціоналом інтернет-магазину.

WEB APPLICATION, ONLINE SHOP, API, HYPERTEXT MARKUP LANGUAGE HTML, CASCADE STYLE SHEETS CSS, JAVASCRIPT PROGRAMMING LANGUAGE, REACT JAVASCRIPT LIBRARY, JAVA PROGRAMMING LANGUAGE, SPRING JAVA FRAMEWORK, POSTGRESQL RELATIONAL DATABASE.

The object of the work is the process of developing a client-server architecture for a web site for purchasing goods.

The goal of the work is to develop a reliable and effective online store that will allow merchants to buy goods from the supermarket.

In the process of working on the qualification work, a study of existing sites for ordering goods from a supermarket was conducted. An application software interface of the server part was created for the client's interaction with the database. The client program has been developed in the form of a website. A supermarket database scheme is built.

As a result of the work, a web application with the basic functionality of the online store were created.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД – система керування базами даних

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

DOM – Document Object Model (об’єктна модель документа)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

MVC – Model-View-Controller (модель-вид-контролер)

SPA – Single-Page Application (односторінковий застосунок)

API – Application Programming Interface (прикладний програмний інтерфейс)

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

JVM – Java Virtual Machine (віртуальна машина Java)

JWT – JSON Web Token

SQL – Structured Query Language (мова структурованих запитів)

DDL – Data Definition Language (мова визначення даних)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

UML – Unified Modeling Language (уніфікована мова моделювання)

ВСТУП

Роздрібна торгівля в інтернет-магазині – це один із методів збільшення доходу мережі супермаркетів та ефективний спосіб підвищення конкурентоспроможності. Вебсайт є додатковим джерелом клієнтів, які можуть замовити товари з будь-якої точки країни.

Все більше користувачів віддають перевагу замовленню товарів онлайн, адже це зручно та економить їх час. Тому сучасні компанії все частіше замислюються над створенням власного вебзастосунку для торгівлі в інтернеті. Розробка такого застосунку потребує певних ресурсів, але, як показує практика світових лідерів, ці вкладення згодом окупаються.

Розробка вебзастосунку для замовлення товарів із супермаркету дає можливість розширити ринок збуту продукції, привернути увагу потенційних покупців, підвищити комфорт створення покупок для існуючих клієнтів та забезпечити швидку передачу інформації про товари.

Актуальність роботи також полягає у підвищенні безпеки здійснення покупок під час війни в Україні. Створити онлайн замовлення можна з безпечного місця, потрібно лише мати доступ до мережі інтернет. Невпинно зростає кількість користувачів, які прагнуть зручно та безпечно придбати необхідні товари. Використання сучасних технологій розробки допоможе створити зручний та ефективний вебзастосунок, який зможе вдовольнити потреби покупців.

1 АНАЛІЗ ІСНУЮЧИХ ВЕБЗАСТОСУНКІВ ДЛЯ ЗАМОВЛЕННЯ ТОВАРІВ

1.1 Сучасний стан вебзастосунків для замовлення товарів із супермаркету в Україні

Майже кожна крупна мережа супермаркетів в Україні мають власний сайт, який одразу виконує декілька важливих функцій. Навіть незареєстрований користувач може переглянути каталог товарів, актуальні пропозиції, знижки та новинки.

Власникам супермаркетів більше не потрібно витрачати папір та друкувати буклети зі знижками, достатньо адміністратору сайту лиши помістити певні товари в потрібну категорію. Клієнти у свою чергу одразу розуміють які товари продаються зі знижками, фізично не відвідуючи магазин. Аналогічно до товарів зі знижками, більшість інтернет-магазинів мають розділ із новинками, де відображаються останні додані товари. Приклад, як сучасні супермаркети сповіщають клієнтів про акційні пропозиції наведено на рисунку 1.1.

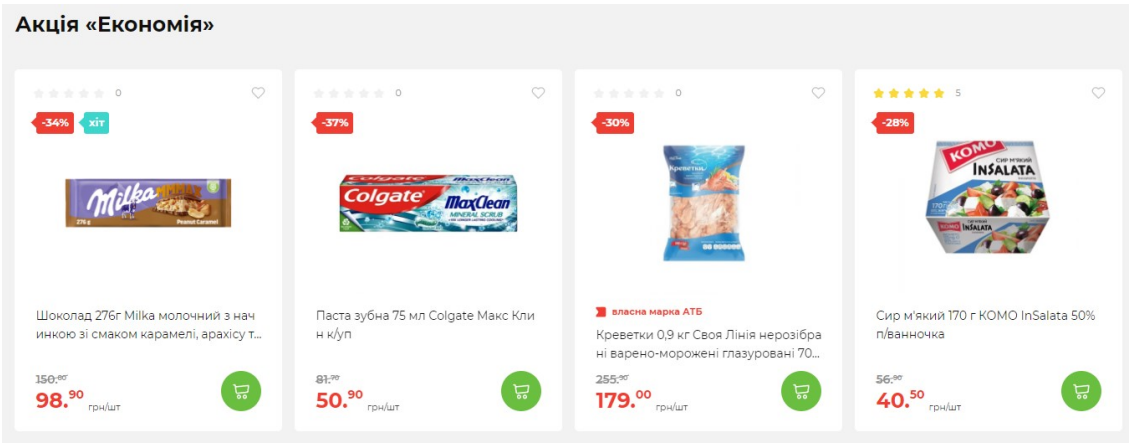

Рисунок 1.1 – Акційні пропозиції [1]

У супермаркеті усі товари розташовані на стелажах у відповідних відділах. Вебзастосунки відомих супермаркетів також класифікують товари

за категоріями (рис. 1.2). Кожен товар може мати лише одну категорію. Деякі супермаркети розділяють категорії ще й на підкатегорії.

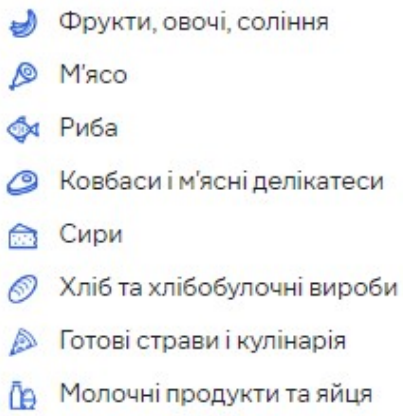

Рисунок 1.2 – Категорії товарів [2]

Кожен товар у каталозі має власну картку на якій розміщено інформацію про продукт (рис. 1.3). Проаналізувавши сайти популярних супермаркетів, можна виділити наступні елементи карток товару:

- зображення товару;
- ціна;
- ціна зі знижкою, якщо акція діє на цей товар;
- відсоток знижки;
- назва товару;
- одиниця виміру;
- кнопка, що додає товар до кошику;
- рейтинг товару.

При натисканні на картку товару, завантажується повна сторінка продукту, яка містить більш детальну інформацію. Деякі застосунки дозволяються на сторінці товару залишити не лише оцінку, а ще й розгорнутий відгук про товар, але це можуть робити лише зареєстровані користувачі. Після реєстрації клієнт може додавати товари до кошику та оформлювати замовлення, попередньо вказавши адресу для доставки.

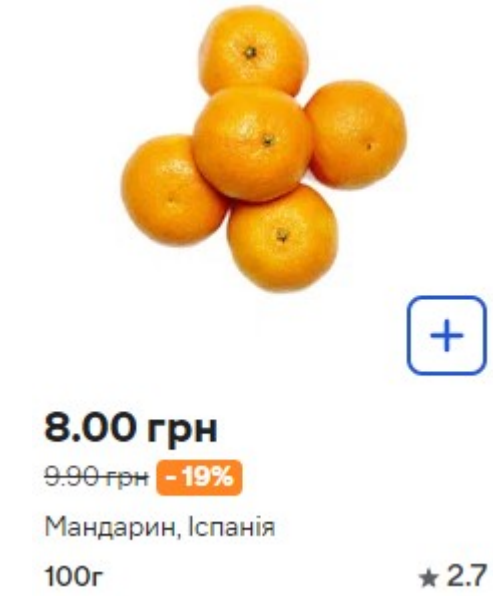

Рисунок 1.3 – Картка товару [2]

Проаналізувавши існуючі можна побачити як власники супермаркетів залучають нових користувачів до онлайн-замовлень, а саме: пропонують знижку на перше замовлення або безкоштовну доставку, якщо замовляєш вперше.

Важливо не лише обробляти онлайн-замовлення, а й отримувати зворотній зв’язок від користувачів, тому більшість відомих інтернет-магазинів розміщують свої контактні дані. Найчастіше така інформація зустрічається у нижній частині сайту. Зазвичай це телефон гарячої лінії, адреса головного офісу магазину, електронна пошта або посилання на соціальні мережі.

1.2 Аналіз технологій для розробки вебзастосунків

На сьогоднішній день розробники мають доступ до багатьох технологій, мов програмування, бібліотек та фреймворків. Усі вони мають свої недоліки та переваги. Зробивши огляд існуючих вебсайтів можна одразу починати аналізувати доступні інструменти для вдалого та ефективного

вирішення задач. Успіх вибору засобів реалізації вебзастосунку залежить не тільки від поставлених задач, але й від рівня знань та навичок розробника. Усі мови програмування для веброзробки можна поділити на дві групи: мови для розробки клієнтської та серверної частини.

1.2.1 Аналіз технологій клієнтської частини

Клієнтська частина може бути представлена у вигляді вебсайту, мобільного додатку рідше окремим програмним забезпеченням для персонального комп'ютера.

Для розробки сайту використовують мову гіпертекстової розмітки HTML. Ця мова дозволяє побудувати розмітку вебсторінки для браузера за допомогою тегів. Відповідно до об'єктної моделі документа DOM, кожен HTML тег є об'єктом. У кожного такого об'єкта є певні атрибути, які зазвичай представлені парою «ключ-значення». Теги HTML – це будівельні блоки, за допомогою яких на сторінці можна розмістити тест, зображення, таблицю, посилання та інші елементи. Теги можуть всередині себе розміщувати інші теги, що дозволяє розробнику створювати комплексну структуру сайту [3].

Актуальною, на момент написання кваліфікаційної роботи, є версія HTML 5, яку підтримують усі популярні браузери. Документ HTML 5 має три частини:

- декларація типу документа;
- голова документа, в якій записані основні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
- тіло документа, яке містить основну інформацію документа.

Хоча гіпертекстова розмітка і є базовим каркасом вебсторінки, але сьогодні лише її недостатньо для розробки сучасного сайту. Як мінімум

необхідна ще каскадна таблиця стилів CSS. Це спеціалізована мова для налаштування стилю HTML документів. Вона дозволяє змінювати шрифти, стилізувати задній фон, задавати відступи між елементами та впливати на інші візуальні аспекти документу. Два однакових HTML документа можуть бути по-різному відображені браузером залежно від заданих CSS стилів.

Каскадне верстання сторінок замінило табличний метод. Основна перевага каскадного верстання – можливість розділити дані сторінки та її візуальне представлення. Але разом з цим можна виділити й інші переваги CSS, завдяки яким ця технологія широко використовується у розробці вебдодатків, а саме:

– стилістичні дані сайту, або його частин можуть розміщуватися в одному файлі, що пришвидшує розробку та зміну дизайну сторінок;

– клієнтоорієтований підхід у створенні дизайну: окремі стилі для мобільних пристроїв або планшетів, збільшений шрифт для людей з вадами зору;

– наявність певних правил застосування, з урахування яких буде побудована сторінка;

– зменшено час завантаження сайтів, обсяг передаваємої інформації та навантаження на сервери, за рахунок кешування даних про стилі браузером.

Для спрощення написання стилів розробники використовують різні CSS бібліотеки та фреймворки. Фреймворк – це сукупність бібліотек, які дозволяють програмістам швидко та ефективно розробляти застосунки за допомого заздалегідь написаних алгоритмів і структур даних. Найпоширенішими CSS фреймворками них є:

– Tailwind CSS;

– Bootstrap;

– Foundation;

– Materialize CSS;

– Semantic UI.

Кожен з цих фреймворків дозволяє створити сучасний та привабливий сайт. Front-end фреймворки містять компоненти для спрощення роботи верстальника, пришвидшення розробки та мінімізації числа помилок (проблеми сумісності різних версій браузерів, пристроїв і т.д.). Більш функціональні фреймворки мають розширений функціонал та додаткові можливості завдяки JavaScript.

JavaScript – це об’єктно-орієнтована мова програмування для реалізації сценаріїв на стороні клієнта, що дають можливість управляти браузером, обмінюватися даними асинхронно з сервером, змінювати DOM-дерево. Прототипування, яке використовується в JavaScript зумовлює відмінності у роботі порівняно з традиційно клас-орієнтованими мовами. Імперативна та частково функціональна парадигми програмування також підтримуються мовою програмування. JavaScript є інтерпретованою, динамічно-типізованою мовою із автоматичним керуванням пам’яті. Як і усі інші перераховані технології, остання версія JavaScript підтримується усіма сучасними браузерами, тому її використання є доволі безпечною з точки зору кросбраузерності.

Для швидкої та ефективної розробки клієнтської частини вебзастосунків було створено багато JavaScript фреймворків. Слід зазначити, що JavaScript можна використовувати не лише для розробки front-end, а й для back-end частини. Взагалі ця мова характеризується великою кількістю написаних фреймворків, що безумовно є плюсом для розробників (рис. 1.4). Серед існуючих фреймворків для розробки клієнтської частини можна виділити:

– Angular – фреймворк розроблений Google з повноцінною підтримкою MVC патерну, але який має недолік у вигляді великого розміру вихідного коду;

– React – бібліотека візуалізації від Meta, але завдяки підключенню інших бібліотек, таких як: React Router, React DOM і т.д. можна зібрати повноцінний фреймворк [4];

– Vue.js – збалансований фреймворк, що дозволяє швидко створювати SPA застосунки, має детальну документацію та низький поріг входження [5];

– Svelte – має власний синтаксис, який завдяки компілятору перетворює написаний код в оптимізований код JavaScript. Це дозволяє не завантажувати повністю увесь фреймворк.

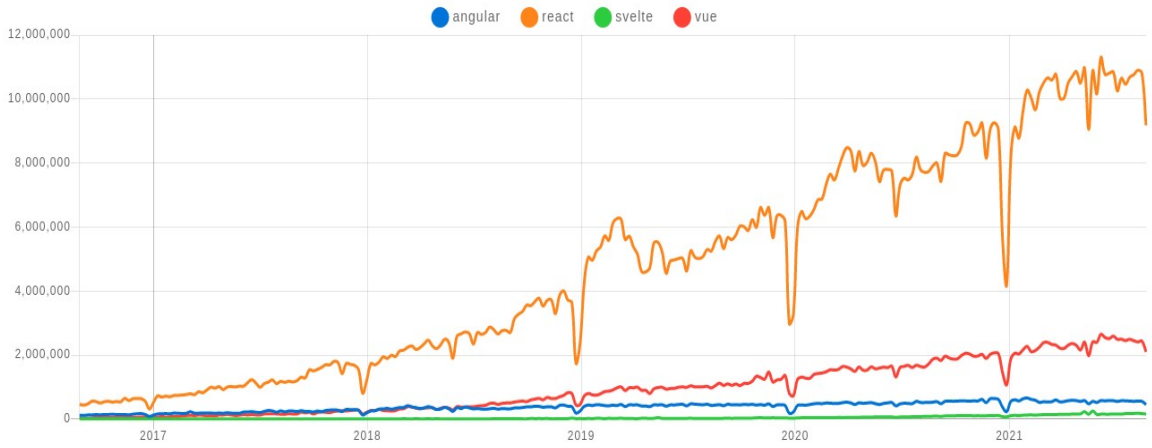

Рисунок 1.4 – Графік популярності JavaScript фреймворків [4]

1.2.2 Аналіз технологій серверної частини

Для розробки серверної частини вебзастосунку існує багато мов програмування, бібліотек та фреймворків.

JavaScript є мовою програмування на якій можна написати і front-end, і back-end. Написання серверної частини на JavaScrript стало можливим після появи у 2008 році Node.js.

Node.js – програмна платформа, що базується на рушії V8, яка перевтілює JavaScript із вузькоспеціалізованої мови в мову широкого використання. Досягається це завдяки надання можливості JavaScript взаємодіяти з пристроями вводу-виводу через власний API Node.js, підключати зовнішні бібліотеки, які навіть написані на інших мовах. Як і у випадку з front-end розробкою, JavaScript має багато фреймворків для back-

end розробки. Усе це у сукупності з швидкістю та ефективністю розробки на JavaScript, робить платформу Node.js привабливою технологією для розробки великих та масштабованих вебзастосунків, які здатні опрацьовувати значні обсяги даних.

Серед популярних JavaScript фреймворків для серверної частини виділяють:

- Express;
- Koa.js;
- Adonis.js;
- Nest.js;
- Fastify.

На відміну від інструментарію для front-end розробки, серверна частина має більший вибір мов програмування. Кожна мова з п’ятірки найпопулярніших за індексом PYPL дозволяє розробникам написати серверну частину вебзастосунку. PYPL (Popularity of Programming Language) – індекс популярності мов програмування, створений завдяки аналізу частоти запитів в Google (рис. 1.5).

Worldwide, Apr 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	28.43 %	+0.7 %
2		Java	16.04 %	-0.1 %
3		JavaScript	8.72 %	-0.8 %
4	↑	C/C++	6.65 %	+0.2 %
5	↓	C#	6.63 %	-0.2 %

Рисунок 1.5 – Рейтинг мов програмування за індексом PYPL [6]

Незважаючи на те що Python є новачком у галузі веб розробки відносно інших мов програмування, це не заважає їй займати перше місце в рейтингу

PYPL. Ця мова має низький поріг входу та чудову документацію для початківців, що робить її лідером у багатьох галузях. Синтаксис є не складним для розуміння, легко та інтуїтивно читається, що тільки додає Python популярності серед розробників.

Python як і інші мови має спеціалізовані фреймворки для веб розробки. Найвідомішим серед них є Django. Він має усі необхідні компоненти для створення повноцінних вебсайтів будь-якої складності. Переваги Django:

- наявність готової системи автентифікації користувачів;
- легка маршрутизація URL-адрес;
- вбудований ORM для роботи з базами даних ні рівні обєктів;
- підтримка різних вебсерверів таких як Apache або Nginx;
- власний рушій шаблонів для HTML сторінок.

Недоліками Django є обширність власних бібліотек які можуть знижувати гнучкість вебзастосунку та ускладнювати інтеграцію з іншими технологіями. Для деяких проектів, які не потребують складної логіки та широкого функціоналу, Django може виявитися надлишковим. Більш легковажною альтернативою Django є мікроферймворк Flask, який використається у розробці вебзастосунків, API та RESTful сервісів.

Ще одним вагомим гравцем на ринку веброзробки є мова програмування Java. Ця мова чудово підходить для розробки великих корпоративних застосунків. Згідно з рейтингом мов програмування DOU, Java – третя за популярністю мова серед українських розробників. Java має велике ком’юніті розробників та розвинену екосистему .

Java – це об’єктно-орієнтована, статично типізована мова програмування, яка має с-подібний синтаксис. Ця мова є незалежною від архітектури, написаний на Java код працюватиме будь-де. Це досягається завдяки компіляції початкового коду у байт-код. Потім цей байт-код виконується на JVM, яка інтерпретує код у зрозумілі команди для конкретної архітектури.

Отже основними перевагами розробки вебзастосунків на Java є:

– кросплатформеність;
– надійність;
– безпека;
– продуктивність.

Spring Framework – це Java фреймворк з відкритим кодом, який підтримує аспектно-орієнтоване програмування та інверсію керування. Spring – це ціла екосистема, яка складається з 21 проєкту. За своєю суттю Spring Framework є контейнером впровадження залежностей із зручними модулями (наприклад, доступ до бази даних, проксі, аспектно-орієнтоване програмування, вебінфраструктура MVC). Усе пришвидшує розробку якісних Java-застосунків.

1.3 Постановка задачі

Таким чином, розробка вебзастосунку для замовлення товарів з супермаркету є актуальним завданням, тому що сучасний темп життя все частіше спонукає людину ефективно розпоряджатися вільним часом та оптимізувати навіть такі процеси як покупка їжі. Тому ставиться завдання розробки вебзастосунку, який дозволить швидко та зручно замовляти товари з супермаркету.

Об’єктом роботи процес розробки клієнт-серверної архітектури вебзастосунку для замовлення товарів.

Метою роботи є розробка надійного та ефективного інтернет-магазину, який дозволить користувачам замовляти товари з супермаркету.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати існуючі інтернет супермаркети;
– проаналізовати сучасні технології для розробки вебзастосунків;
– спроектувати модель бази даних для вебзастосунку;

– реалізувати авторизацію, реєстрацію та автентифікацію користувачів;

– розробити каталог товарів з відповідними категоріями;

– створити кошик користувача та реалізувати можливість оформлювати замовлення;

– розробити привабливий інтерфейс сайту.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ЗАМОВЛЕННЯ ТОВАРІВ З СУПЕРМАРКЕТУ ТА ТЕХНІЧНЕ ОБҐРУНТУВАННЯ ОБРАНИХ ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Моделювання загальної концепції роботи вебзастосунку та системи автентифікації користувачів

Основна концепція вебзастосунку полягає у реалізації клієнт-серверної архітектури сайту, що дозволить користувачам зручно та швидко замовляти товари з супермаркету.

Вебзастосунок розрахований на три ролі користувачів: зареєстрований користувач, адміністратор супермаркету та незареєстрований гість. Кожній ролі надано свої власні можливості та обмеження. Для реалізації даної концепції розмежування функціоналу за ролями, потрібно розробити систему реєстрації, авторизації та автентифікації користувачів на сайті.

Моделюючи загальну концепцію використання застосунку, необхідно урахувати усі можливі сценарії взаємодії користувача і сайту. На основі змодельованої системи розробник зможе спроектувати структуру бази даних користувачів [7].

Адміністратор має вільний доступ до бази даних товарів, користувачів та замовлень. Йому надано можливість змінювати, створювати та видаляти товари. Також він може керувати замовленнями користувачів.

Клієнт супермаркету або користувач, який пройшов реєстрацію має можливість додавати товари до кошика, створювати замовлення та відстежувати їх статус.

Незареєстрований користувач, або гість може лише передивлятися інформацію, перегляд якої не потребує реєстрації на сайті. Для незареєстрованих користувачів доступна сторінка авторизації та реєстрації.

Змоделювавши три ролі користувачів вебзастосунку, побудуємо Use Case діаграму сайту для замовлення товарів з супермаркету (рис. 2.1).

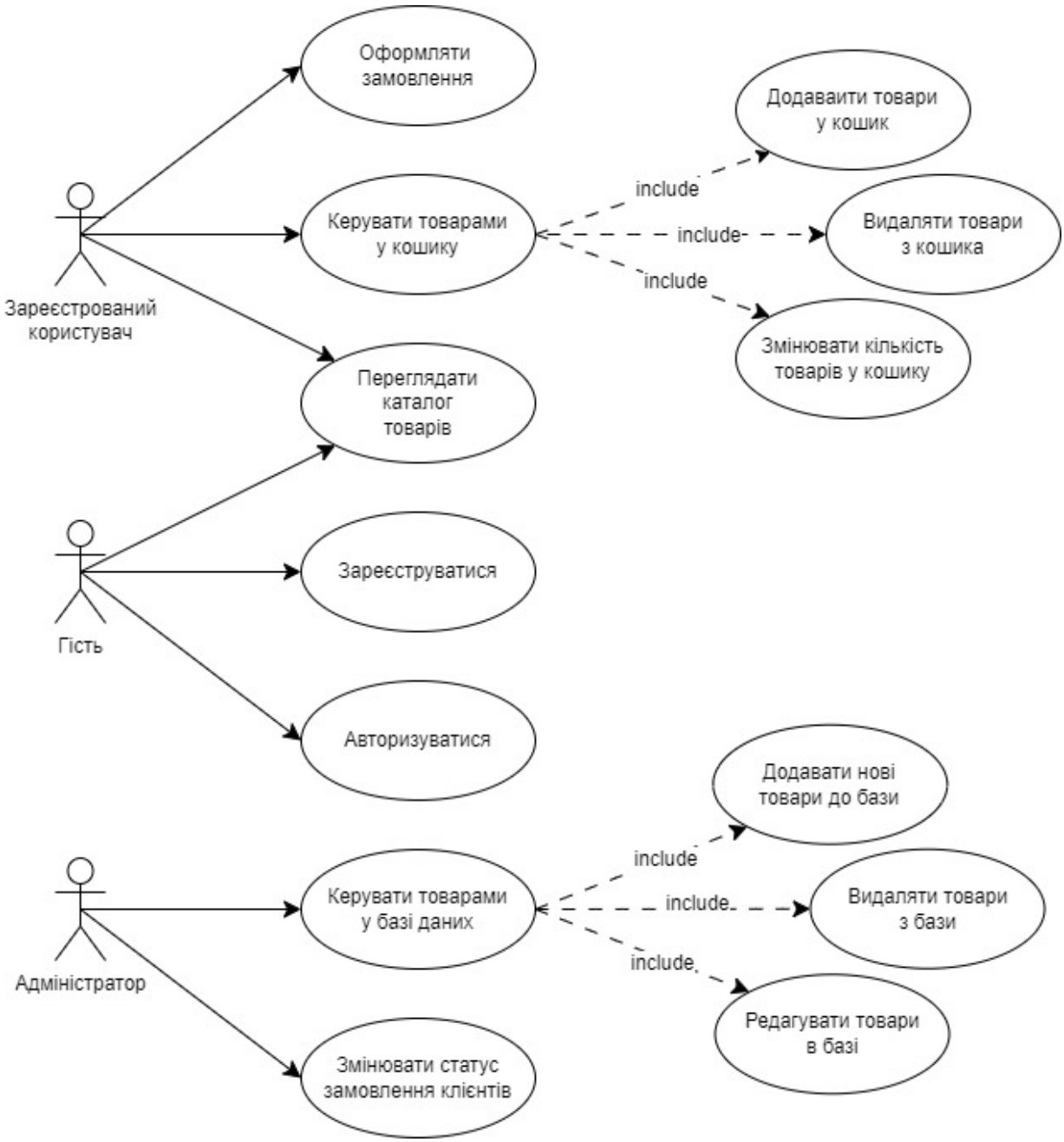

Рисунок 2.1 – Use Case діаграма вебзастосунку

Для того, щоб кожна роль на сайті мала свій функціонал, необхідно розробити систему реєстрації, авторизації та автентифікації користувачів.

Для реалізації даної системи існує декілька стандартів, серед яких можна виділити:

– OAuth – відкритий стандарт авторизації, що дає користувачам можливість ділитися власними приватними даними, такими як: ім’я, фото, список контактів та ін., що зберігаються на іншому сайті, без обов’язкового

введення логіну та паролю. Цей метод має свої переваги, але не підходить для вебзастосунку для замовлення товарів з супермаркету, бо припускається, що аудиторія користувачів супермаркету є доволі різноманітною і деякі з них можуть не мати облікових записів на інших ресурсах;

– SQRL – ще один відкритий стандарт безпечної авторизації та автентифікації користувачів. Замість облікового запису на іншому ресурсі цей стандарт передбачає використання QR-коду. Клієнт ідентифікується анонімно, замість введення логіну та паролю. Цей метод є стійким від перебору паролів та викрадення даних, але теж не підходить для вебзастосунку супермаркету, адже не кожен має доступ до пристрою сканування QR-коду під час користування сайтом;

– JWT – відкритий стандарт для створення JSON токенів доступу. JWT створюється на стороні серверу, підписуються одним із методів шифрування за допомогою ключа, який заборонено розголошувати, далі створений токен передається користувачу, який надалі використовується для автентифікації. Цей метод чудово підходить для розробки системи автентифікації користувачів вебзастосунку для замовлення товарів з супермаркету.

Перевагами використання JWT є швидкість та простота розробки та можливість розвантажити сервер від зберігання сесій користувачів та частих запитів інформації про користувача.

На рисунку 2.2 зображено діаграму послідовності взаємодіє клієнта з сервером для отримання токена. Для цього клієнт надсилає POST запит до серверу з даними про логін та пароль користувача у форматі JSON. Далі сервер перевіряє наявність надісланого логіну в базі даних, якщо такий запис існує, то перевіряється переданий пароль, який хешується та зіставляється з хешем паролю, що зберігається в базі. Якщо хеші паролів однакові, то за допомогою спеціального алгоритму генерується токен і повертається клієнту. Якщо клієнт передає невалідні дані, то сервер повертає відповідь із статус-кодом 401 Unauthorized [8].

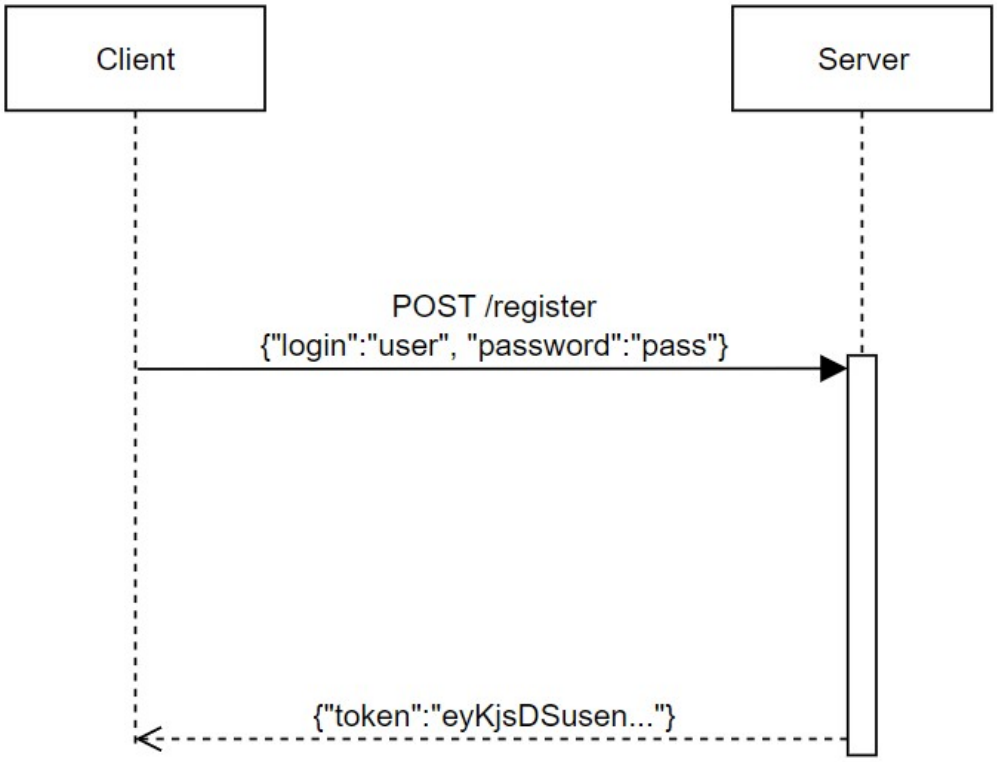

Рисунок 2.2 – Діаграма послідовності отримання токену

Після вдалого отримання токену, користувач є ідентифікований системою і усі запити, які потребують авторизації, надсилаються зі спеціальним заголовком, який розміщується у секції «Header». Стандартом передачі JWT є заголовок авторизації, який починається ключовим словом «Bearer», а далі йде сам згенерований токен. Після отримання токену сервер перевіряє його на валідність та надає доступ до сторінки, якщо час дії токену вийшов, або він недійсний, то клієнт отримує відповідь із кодом помилки 401. Побудуємо діаграму послідовності валідації токену (рис. 2.3).

На стороні клієнта токен може зберігатися або у локальному сховищі «local storage», або у кукі-файлах. Для розробки вебзастосунку для замовлення товарів з супермаркету обрано варіант зберігання у локальному сховищі браузера. Такий метод має перевагу у спрощені процесу розробки, але й має недолік у вигляді вразливості атаки міжсайтового скриптингу (Cross-site scripting). Захиститися від такої атаки можна валідацією вхідних даних та налаштуваннямм політики безпеки контенту (Content Security Policy) вебзастосунку [9].

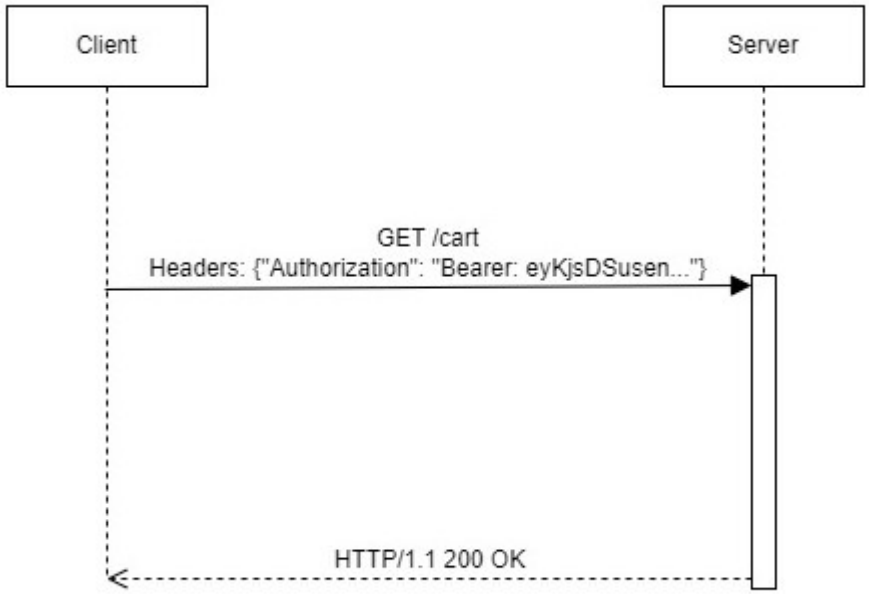

Рисунок 2.3 – Діаграма послідовності валідації токену

Розглянемо структуру JWT (рис. 2.4). Структура кожного JWT складається з трьох частин:

– частина «Header» містить у собі загальну інформацію про токен, зазвичай зазначається тип токену та алгоритм за допомогою якого він підписується. Зазвичай один алгоритм використовується для підпису усіх токенів. Найпопулярнішими алгоритмами підпису JWT є HS256, RS256, ES256;

– частина «Payload» є корисним навантаженням токену. Сюди розробник може покласти будь-яку інформацію про користувача, яка може знадобитися для його подальшої автентифікації. Наприклад, ім’я користувача, перелік його ролей, час створення токену та час коли термін дії токену спливає;

– остання частина «Signature» є підписом усього токену. Вона генерується за допомогою алгоритму зазначеного у першій частині. Підпис генерується на основі даних з частин «Header» та «Payload», а також до них ще додається секретний ключ, який слугує захистом від підробки токену. Тільки знаючи цей ключ можна згенерувати валідний JWT, тому його категорично заборонено розголошувати.

Рисунок 2.4 – Структура JWT [10]

Існує два варіанта розробки системи автентифікації користувачів сайту за допомогою JWT. Перший передбачає використання лише токену доступу «Access token», другий зобов'язує використовувати ще й токен відновлення «Refresh token». Токен доступу дозволяє авторизованому користувачу відправляти запити на отримання сторінок, які дозволено переглядати лише авторизованим користувачам. Оскільки кожен токен має обмежений час дії, то згодом токен доступу стане не дійсним і користувачу потрібно буде заново пройти процедуру авторизації. Токен відновлення є більш довготривалим і за його допомогою користувач може отримати новий токен доступу не вводячи заново логін та пароль. Перевагою використання двох токенів є захист від несанкціонованого доступу до облікового запису користувача. Якщо зловмисник заволодіє токеном доступу, то через

короткий час він все одно втратить доступ, через те що час дії такого токену швидко закінчиться, а для отримання нового потрібно знати ще й токен відновлення. Недоліками такого методу є довша розробка більш складної системи авторизації, більше навантаження на сервер через використання двох токені замість одного [11].

При розробці вебзастосунку для замовлення товарів з супермаркету доречно використати варіант з одним токеном доступу. Проблему з безпекою можна частково вирішити за допомогою встановлення короткого терміну дії для JWT. Отже, визначивши модель авторизації, побудуємо діаграму послідовності авторизації користувачів на сайті (рис. 2.5) [12].

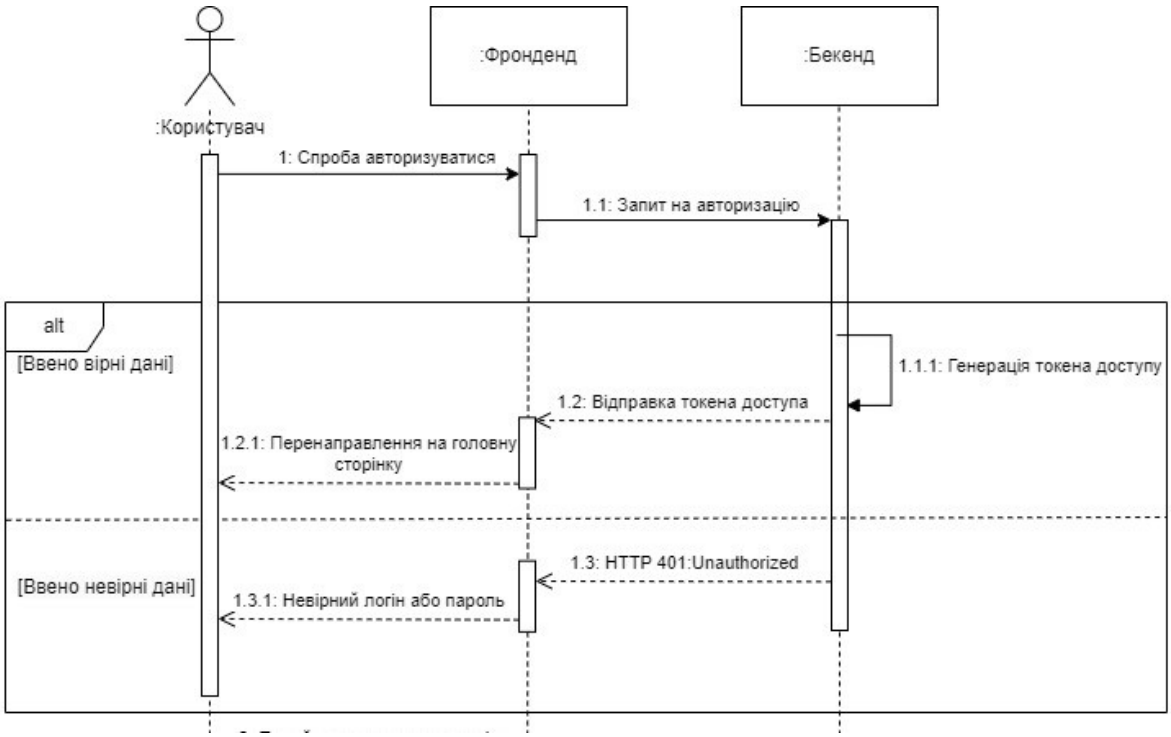

Рисунок 2.5 – Діаграма послідовності авторизації користувача

2.2 Моделювання структури бази даних

База даних – це впорядкована та структурована сукупність даних, що зазвичай розміщуються на електронних носіях. Майже кожен сучасний

вебзастосунок використовує бази даних для зберігання інформації, швидкого та ефективного доступу і оновлення даних. Для управління базами даних розробники використовують СКБД.

СКБД – це спеціальне програмне забезпечення, що дозволяє створювати, змінювати, видаляти та переглядати інформацію у базах даних. СКБД спрощують роботу з базами завдяки графічному інтерфейсу та автоматизації деяких запитів. Окрім зручності та ефективності роботи з інформацією, бази даних також забезпечують безпеку даних, вони дозволяють гнучко налаштовувати доступ до бази та гарантують цілісність даних. Також слід виконувати резервне копіювання даних з бази для захисту від знищення інформації та від несанкційованого редагування даних.

Важливо правильно змоделювати правильну структуру даних, яка дозволить швидко та ефективно організовувати та обробляти дані. Серйозні помилки, допущені на цьому етапі, можуть призвести до непередбачуваних наслідків та некоректної роботи всієї системи в цілому.

Внутрішня організація баз даних представлена записом даних у таблицю, що відповідає переліку рядків і стовпців, які проіндексовані для ефективнішого пошуку інформації. На момент написання кваліфікаційної роботи існує декілька видів моделей баз даних, а саме ієрархічні, реляційні, нереляційні, мережеві. Кожна з цих моделей даних має свої переваги та недоліки, але для розробки вебзастосунків найчастіше використовують реляційні бази даних.

Реляційна база даних представлена набором таблиць, які пов’язані між собою відповідними зв’язками. Зв’язки або відношення між таблицями зазвичай позначають у числовому вираженні. Це можуть бути відношення один до одного, один до багатьох або багато до багатьох. Моделювання складних відносин реалізовано за допомогою зовнішніх ключів, які є посиланням на дані в інших таблицях [13]. Для маніпуляції з реляційними базами даних використовується мова запитів SQL. Для спрощення та автоматизації виконання SQL запитів розробники використовують реляційні

СКБД. Популярними реляційними СКБД є MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Firebird [14].

Нереляційні або NoSQL бази даних мають структуру відмінну від моделі таблиця-відношення. Оскільки відсутня чітка система відношення між даними, то в NoSQL базах можна одночасно зберігати різні типи даних. В таких базах можна організувати зберігання погано структурованих даних або зовсім не структурованих даних. Нереляційні бази даних підходять для роботи з великими даними та вебзастосунками у режимі реального часу. NoSQL бази краще підходять для горизонтального масштабування, ніж реляційні бази даних, завдяки наявності розподіленої архітектури.

Нереляційні бази даних розподіляються на документоорієнтовані бази, бази даних «ключ-значення», графові бази даних та стовпчикові, або колонкові бази даних. Найпопулярнішими нереляційними базами даних є MongoDB, RethinkDB, Redis, Cassandra, HBase.

Розглянувши основні відмінності реляційних і нереляційних баз даних, можна сказати, що реляційні бази даних є привабливішим варіантом для розробки вебзастосунку для замовлення товарів з супермаркету. Однією з надійніших та безпечних реляційних баз даних з обширною екосистемою є PostgreSQL, тому саме її доречно використати для розробки інформаційної системи вебзастосунку. Ще однією перевагою PostgreSQL є сумісність з широким спектром технологій та мов програмування, таких як: C/C++, Go, Java, Python, .NET, Ruby, що у свою чергу не накладає жодних обмежень у виборі технологій для розробки серверної частини [15].

Описавши систему авторизації та автентифікації користувачів сайту у попередньому підрозділі, можна перейти до моделювання структури бази даних користувачів та їх ролей. Для реалізації визначеної бізнес-логіки необхідно створити дві сутності: користувач «User» та роль «Role». Визначимо поля та типи даних таблиці користувачів (табл. 2.1).

Таблиця 2.1 – Характеристики таблиці користувачів

Key	**Name**	**Type**	**Description**
PK	Id	Bigint	Ідентифікатор користувача
	Name	Character varying (20)	Ім’я користувача
	Password	Character varying (255)	Хеш паролю користувача

Таблиця «Users» має мінімальний необхідний набор полів для ідентифікації користувачів. Поле «Id» є унікальним ідентифікатором користувача, воно повинно автоматично генеруватися базою за допомогою визначеної послідовності. Поля «Name» і «Password» є ненульовими і задаються при реєстрації користувача. Слід зазначити, що поле «Name» обмежене по довжині, а пароль збігається не відкрито, а у вигляді хешу з міркувань безпеки.

Щоб користувачі розподілялися по ролям, створимо таблицю «Roles» (табл. 2.2).

Таблиця 2.2 – Характеристики таблиці ролей

Key	**Name**	**Type**	**Description**
PK	Id	Bigint	Ідентифікатор ролі
	Name	Character varying (255)	Назва ролі

Таблиця ролей має схожу структуру з таблицею користувачів. Але оскільки один користувач може мати декілька ролей, то характеристика ролі користувача зазначена не у вигляді поля таблиці «Users», а винесена у окрему таблицю «User_roles», для запобігання повтору даних. При реєстрації нового користувача, йому автоматично присвоюється роль простого клієнта. Змінювати ролі може лише адміністратор бази даних. Розглянемо детальніше структуру таблиці «User_roles» (табл. 2.3).

Таблиця 2.3 – Характеристики таблиці «User_roles»

Key	Name	Type	Description
FK	User_id	Bigint	Ідентифікатор користувача
FK	Role_id	Bigint	Ідентифікатор ролі

Поля «User_id» і «Role_id» є зовнішніми ключами, що посилаються на ідентифікатори таблиць користувачів та ролей. Сутність користувача та ролі між собою відносяться як багато до багатьох, що означає, що один користувач може мати декілька ролей і одна роль може бути призначена багатьом користувачам. Таблиця «User_roles» призначена розвантажити таке відношення, замінивши його двома відношеннями типу один до багатьох.

Описавши модель даних користувачів та їх ролей, побудуємо ER-діаграму на основі визначених таблиць та відношень між ними (рис. 2.6).

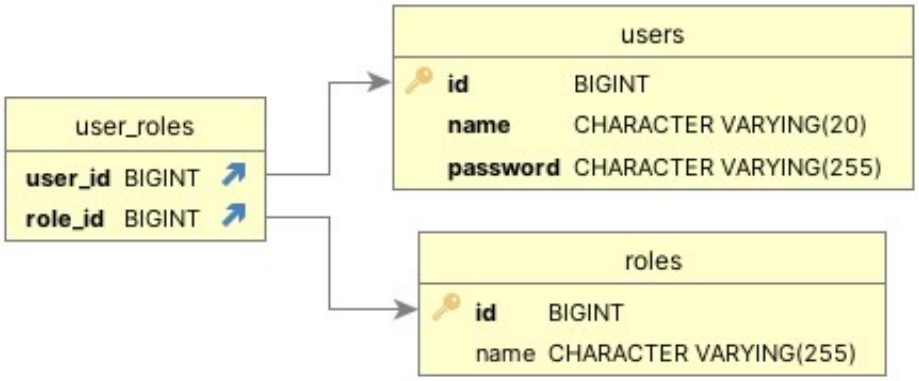

Рисунок 2.6 – ER-діаграма моделі даних для реалізації рольового доступу

Окрім користувачів та ролей у вебзатосунку для замовлення товарів з супермаркету, повинна бути сутність товару. Товар має унікальний ідентифікатор, назву, опис, ціну, посилання на зображення та категорію. Категорія є окремою сутністю. Кожен товар може належати лише до однієї категорії. На основі цих даних створимо таблиці «Products» та «Catagories», які міститимуть інформацію про товари та категорії відповідно (табл. 2.4, 2.5). Зв’язок між цими таблицями визначимо як один до багатьох. Оскільки

кожен товар повинен належати до однієї з категорій, то позначимо поле «category_id» обов'язковим для заповнення.

Таблиця 2.4 – Характеристики таблиці товарів

Key	**Name**	**Type**	**Description**
PK	Id	Bigint	Ідентифікатор товару
	Price	Numeric(38,2)	Ціна товару
	Title	Character varying (255)	Назва товару
	Description	Character varying (255)	Опис товару
FK	Category_id	Bigint	Ідентифікатор категорії
	ImageURL	Character varying (255)	Посилання на зображення товару

Таблиця 2.5 – Характеристики таблиці категорій

Key	**Name**	**Type**	**Description**
PK	Id	Bigint	Ідентифікатор категорії
	Title	Character varying (255)	Назва категорії

Тепер необхідно спроектувати модель даних, яка дозволить користувачам додавати товари в кошик. Для цього створимо таблицю «Carts», яка дозволить зберігати додані в кошик товари на стороні серверу (табл. 2.6). Для того щоб окремо зберігати елементи кошику створимо, іншу таблицю «Cart_items», щоб не дублювати дані (табл. 2.7). В таблиці «Cart_items» будуть зберігатися дані про товар, його кількість у кошику та посилання на окремий кошик користувача. Один кошик може мати багато елементів, тому визначимо зв'язок між таблицями як один до багатьох, але один користувач може мати лише один кошик. Після здійснення замовлення товари з кошику користувача необхідно буде видалити.

Таблиця 2.6 – Характеристики таблиці кошиків

Key	Name	Type	Description
PK	Id	Bigint	Ідентифікатор кошику
FK	User_id	Bigint	Посилання на користувача

Таблиця 2.7 – Характеристики таблиці елементів кошика

Key	Name	Type	Description
PK	Id	Bigint	Ідентифікатор елементу кошика
FK	Product_id	Bigint	Посилання на ідентифікатор товару
FK	Cart_id	Bigint	Посилання на ідентифікатор кошику
	Quantity	Integer	Кількість товару

Аналогічно створимо таблиці «Orders» та «Orders_details» для можливості оформити заказ на доставку товарів доданих у кошик (табл. 2.8, 2.9). Будемо фіксувати ціну товару на момент оформлення замовлення, на випадок якщо ціна на товар зміниться під час доставки. Для зберігання часових місток створення замовлення та зміни статусу замовлення будемо використовувати представлений у PostgreSQL тип даних «Timestamp without time zone», що відповідає міткою дати та часу без збереження часового поясу. Використовуючи такий тип даних зробимо припущення, що усі наші клієнти та сервери будуть розміщенні у одному часовому поясі.

Також змоделюємо окрему таблицю «Order_status», в якій визначимо перелік статусів які можуть бути присвоєні замовленню. Оскільки одне замовлення може містити у собі багато товарів, а отже і багато деталей, то відношення між таблицями визначимо як один до багатьох. Але статус замовлення у певний момент часу може бути лише один.

Таблиця 2.8 – Характеристики таблиці замовлень

Key	Name	Type	Description
PK	Id	Bigint	Ідентифікатор кошику
	Address	Character varying (255)	Адреса замовлення
	Created	Timestamp	Дата замовлення
	Sum	Numeruc (38,2)	Сума замовлення
	Updated	Timestamp	Дата зміни статусу
FK	Status_id	Bigint	Статус замовлення
FK	User_id	Bigint	Користувач

Таблиця 2.9 – Характеристики таблиці деталей замовлення

Key	Name	Type	Description
PK	Id	Bigint	Ідентифікатор деталей
	Price	Numeric (38,2)	Ціна замовленого товару
	Qantity	Integer	Кількість товару
FK	Order_id	Bigint	Посилання на замовлення
FK	Product_id	Bigint	Посилання на замовлений товар

ER-діаграми потрібно буде перетворити в реляційні таблиці за допомогою написання відповідних DDL скриптів мовою SQL. Створюючи такі скрипти потрібно спиратися не лише на спроектовану структуру бази даних, а й на певні обмеження, які наявні у таблицях [16].

Наприклад, потрібно встановити режим каскадного видалення у таблицях «Carts» та «Orders» навпроти зовнішнього ключа, що посилається на ідентифікатор користувача. Це призведе до видалення товарів із кошику та знищенню усіх замовлень які робив користувач до того як його обліковий запис було видалено з бази. Такий прийом захищає таблиці від накопичення зайвої інформації, що у свою чергу пришвидшує роботу усієї системи в цілому.

Наразі усе готово для побудови діаграми послідовності, яка відобразить процес замовлення товарів авторизованим користувачем у застосунку (рис. 2.7).

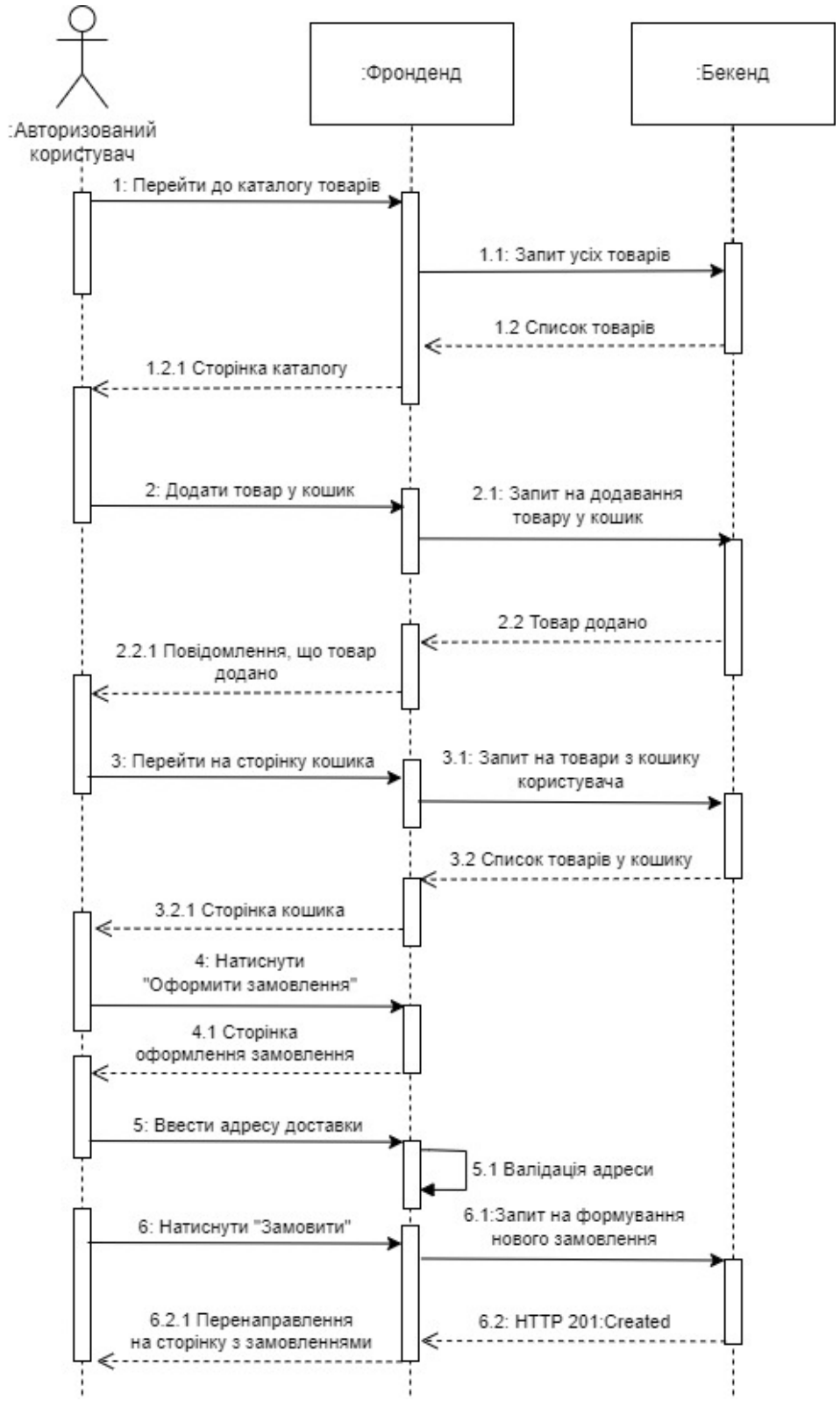

Рисунок 2.7 – Діаграма послідовності замовлення товарів у вебзастосунку

Спроектувавши усі таблиці для зберігання даних вебзастосунку та визначивши відношення між ними, зобразимо загальну ER-діаграму інформаційної системи вебзастосунку для замовлення товарів з супермаркету (рис. 2.8).

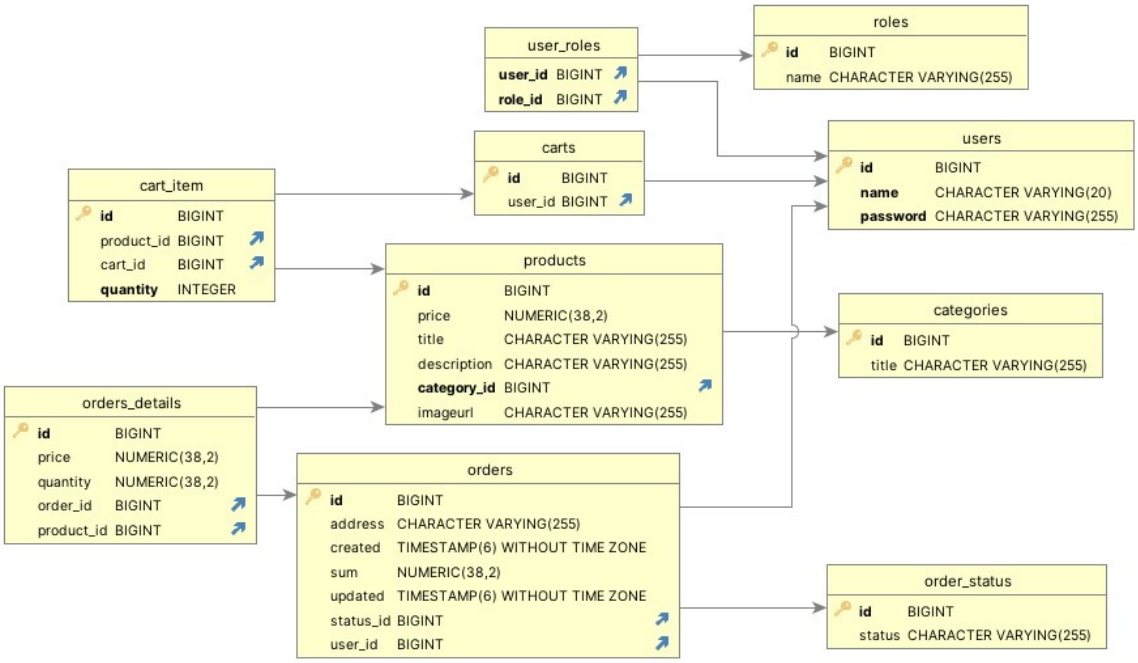

Рисунок 2.8 – ER-діаграма моделі даних вебзастосунку

3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ЗАМОВЛЕННЯ ТОВАРІВ З СУПЕРМАРКЕТУ

3.1 Вибір інструментальних засобів для реалізації поставленої задачі

Важливим етапом розробки вебзастунку для замовлення товарів з супермаркету є вибір інструментальних засобів для досягнення поставленої мети. Проаналізувавши сучасні технології та мови програмування, було прийнято рішення використовувати інтегроване середовище розробки IntelliJ IDEA Community edition для написання серверної частини вебзастосунку мовою Java [17]. Для розробки front-end частини буде використано текстовий редактор Microsoft Visual Studio Code. Основною мовою написання клієнтської частини обрано JavaScript з фреймворком React. Backend розроблено за допомогою фреймворку Java Spring.

IntelliJ IIDEA – це інтегроване середовище розробки від JetBarains, яке має безкоштовну та комерційну версії, Community та Ultimate Edition відповідно. InteliJ IDEA обрано через широкі можливості для розробки та помірні системні вимоги.

Visual Studio Code – це тестовий редактор з підсвічуванням синтаксису розроблений компанією Microsoft. Має певний інструментарій для роботи з системами керування версій та технологію автодоповнення IntelliSense.

Для проектування діаграм послідовності, ER-діаграм та Use Case діаграм було використано вебресурс draw.io. Для тестування GET та POST запитів до сервера прийнято рішення використовувати програмне забезпечення Postman. Щоб спростити процес розробки та адміністрування бази даних Postgres було завантажено графічний клієнт для роботи з сервером pgAdmin.

Саме такий набір інструментів дозволяє швидко та ефективно досягти поставленої мети та розробити вебзастосунок для замовлення товарів з

супермаркету, використавши актуальні методики розробки вебзастосунків та сучасні практики програмування.

3.2 Етапи розроблення вебзастосунку для замовлення товарів з супермаркету

Вебзастосунок складається з двох частин – клієнт та сервер. Доцільніше розпочати розробки з серверної частини. Оскільки основним фреймворком для розробки бекенду є Java Spring Framework, то для ініціалізації проєкту можна скористатися веб застосунком Spring Initializr (рис. 3.1).

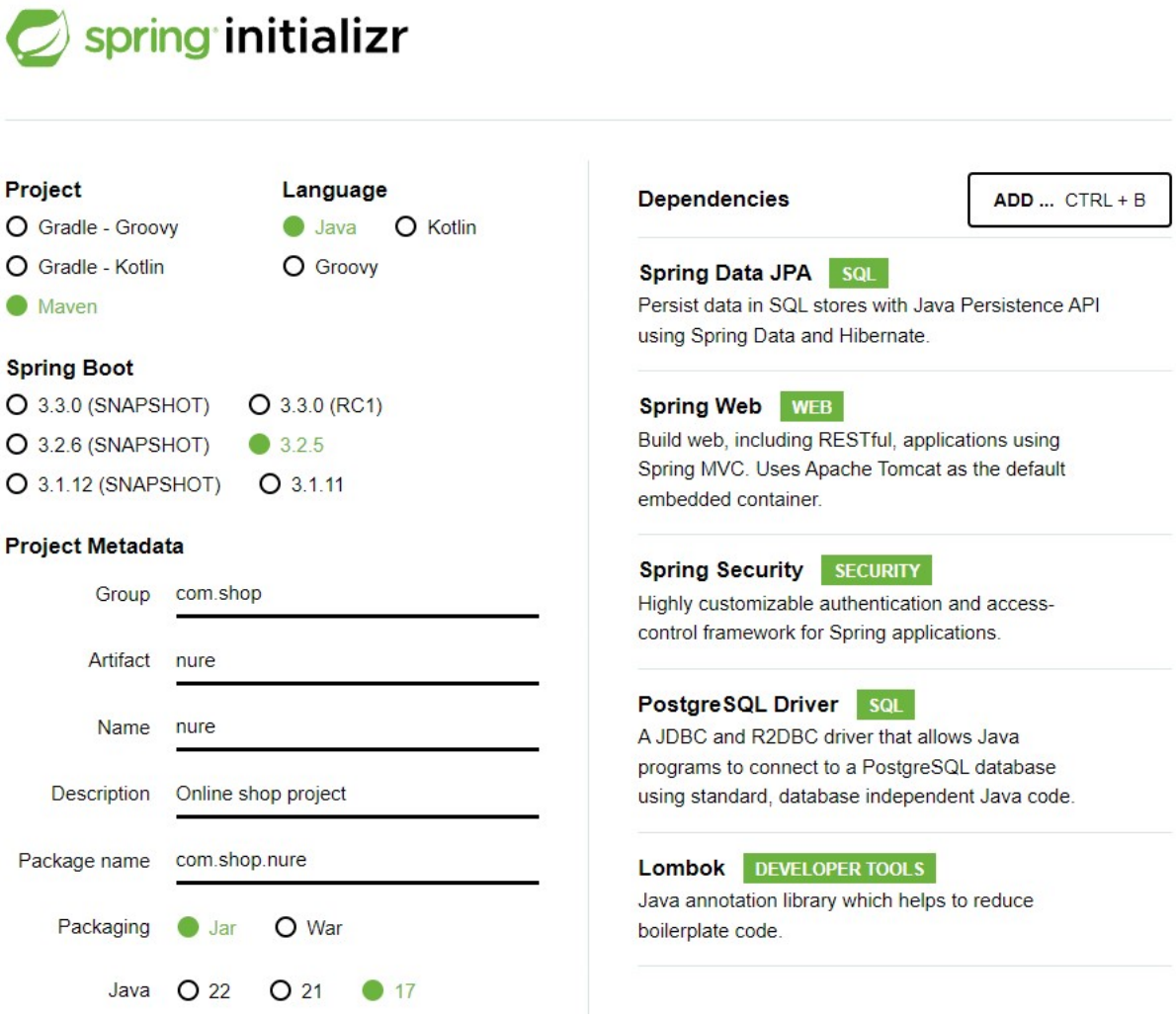

Рисунок 3.1 – Spring Initializr [18]

Спочатку необхідно вибрати мову розробки. Spring Initializr дає на вибір три варіанти мов програмування: Java, Kotlin та Groovy. Обрано Java, так як це було визначено в попередньому підпункті.

Далі необхідно обрати інструмент для автоматизації управління та збирання проєктів. Обрано Maven через те що він має зручне декларативне описання проєкту у POM файлі, а також велику кількість репозиторіїв артефактів, що підтримуються у актуальному стані.

Spring Initializr зручний тим, що дозволяє одразу додати до проєкту залежності, які будуть використані під час розробки. Для розробки вебзастосунку для замовлення товарів з супермаркету необхідно додати наступні залежності: Spring Data JPA, Spring Web, Spring Security, PostgreSQL Driver та Lombok.

Spring Data JPA – бібліотека, що створює додатковий рівень абстрації для роботи програміста з сутностями бази даних та допомогає реалізовувати репозиторії засновані на технології Java Persistence API.

Spring Web – бібліотека, що дозволяє створювати вебзастосунки, включно RESTful сервіси за допомогою Spring MVC. За замовчуванням використовує контейнер сервлетів Apache Tomcat [19].

Spring Security – це потужний фреймворк для реалізації системи автентифікації та контролю доступу [20].

PosgreSQL Driver – реалізація JDBC та R2DBC драйверів для Java вебзастосунків, що дозволяють під’єднуватися до PostgreSQL бази даних та взаємодіяти з нею через стандартні API-інтерфейси.

Lombok – інструментарій розробника заснований на анотаціях, що дозволяє зменшити написання шаблонного Java-коду. В Lombok є певний перелік анотацій, які слугують заміною однотипного коду, написання якого може бути нудним для розробника та займати багато часу.

Також необхідно додати ще одну бібліотеку, яка не представлена у Spring Initializr – це jjwt, яка є інструментарієм для генерації та валідації JWT токенів.

Після ініціалізації проєкту необхідно визначити зовнішні дані конфігурації вебзастосунку. За замовчуванням конфігурація налаштовується в файлі «application» з розширенням «properties», що знаходиться в кореневій директорії проєкту. В такому файлі параметри конфігурації записуються у форматі «ключ-значення», але Spring підтримує й інший варіант налаштування за допомогою «yml(yaml)» формату. Саме таким способом і скористуємося для налаштування вебзастосунку для замовлення товарів з супермаркету (рис. 3.2).

```yaml
spring:
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
  datasource:
    url: jdbc:postgresql://localhost:5433/shop
    username: postgres
    password: admin
jwt:
  secret: emgMArqUjUwmOxKZXatLH1ZScbJZWNRNemgMArqUjUwmOxK
  lifetime: 1d
```

Рисунок 3.2 – Зміст файлу application.yaml

Основними параметрами конфігурації в даному випадку є посилання на сервер з базою даних, ім’я користувача та пароль для доступу до неї. У даному випадку сервер бази даних та сервер бекенду фізично розташовані на одній машині, тому посиланням є «localhost». Також важливо налаштувати параметри генерації JWT токена, а саме секретний ключ для підпису та час дії токена.

Після налаштування конфігурації проєкту, створимо структуру папок, де будуть розміщенні основні класи та інтерфейси вебзастосунку для замовлення товарів із супермаркету (рис. 3.3).

Рисунок 3.3 – Структура папок вебзастосунку

Папка «config» призначена для зберігання класів, які пов’язані з налаштуванням конфігурації Java Web Security. Це включає в себе налаштування менеджера автентифікації, встановлення енкриптору для зберігання паролів, налаштування ланцюжку фільтрів безпеки, створення власних вебфільтрів. Класи конфігурації в Spring позначаються за допомогою анотації «Configuration».

В папці «controller» зберігаються REST-контролери, які обробляють GET та POST запити від клієнта та повертають «Response Entity» у форматі JSON. Контролери в Spring позначаються анотаціями «Controller» та «RestController» в залежності від того який тип вебзастосунку розробляється.

Папка «domain» зберігає усі сутності визначені на етапі моделювання бази даних, описані мовою Java. Згідно з Jakarta Persistence класи, що описують сутності позначаються анотацією «Entity». Зазвичай така анотація використовується разом з анотацією «Table», яка явно вказує Spring з якою таблицею в базі даних розробник хоче зв’язати визначену сутність та явно вказати певні атрибути притаманні таблиці даних.

Папка «dto» призначена для зберігання об’єктів передачі даних «Data-transfer object». Класи таких об’єктів не містять у собі ніякої логіки, вони лише слугують для агрегації даних та передачі їх між різними рівнями вебзастосунку. Запит до серверу є доволі часозатратною дією, тому аби

зменшити кількість запитів, використовують DTO, що є агрегують дані, які без їх використання були отримані за декілька запитів. Єдині процедури які можуть бути реалізовані в таких класах – це процедури серіалізації та десереалізації.

Серіалізація – процедура під час якої вхідні дані перетворюються на бітову послідовність. Дереалізація – процедура, яка є протилежністю серіалізації, під цим терміном розуміють відновлення первісних даних з послідовності бітів.

Папка «exception» зберігає у собі виключення створені користувачем, які можуть бути перехоплені та оброблені за допомогою блоку «try/catch/finally». Такі виключення можуть бути повернуті в якості відповіді сервера, якщо виникає помилка.

Папка «repository» зберігає репозиторії. Репозиторій – це колекція, які містить у собі сутності та може повертати результати запитів в залежності від потреб застосунку. Для створення репозиторіїв у Spring Framework використають анотацію «Repository». Репозиторій є інтерфейсом, що визначає абстрактний механізм зберігання та взаємодії з сутностями бази даних. Зазвичай при роботі з Spring Framework користувацькі репозиторії є нащадками таких інтерфейсів, як «CrudRepository», «PagingAndSortingRepository» та «JpaRepository». Перелічені вище інтерфейси є параметризованими, тому, при їх наслідуванні, у нащадків необхідно вказати тип сутності для якої створюється репозиторій та тип ідентифікатора, який використається в зазначеній сутності.

Папка «service» необхідна для зберігання класів-сервісів, які слугують для того щоб відокремити логіку роботи вебзастосунку. Такі класи не мають полів для зберігання стану. Зазвичай сервіси використовують дані які надходять їм від репозиторії, далі вони їх певним чином обробляють, якщо це потрібно, і повертають клієнту. Для створення сервісу у Spring Framework необхідно використати анотацію «Service». При проектуванні сервісів

необхідно пам’ятати про принцип єдиної відповідальності «Single Responsibility Principle».

Принцип єдиної відповідальності полягає у тому, що один клас (сервіс) повинен вирішувати лише одну задачу. Він може містити декілька методів, які повинні слугувати для вирішення спільної задачі. Якщо клас (сервіс) має декілька призначень, то логіку його роботи слід розділити на окремі класи (сервіси) [19].

Папка «util» зберігає класи які є службовим інструментарієм, що можуть використовуватися іншими класами. Це можуть бути валідатори для перевірки вхідних даних на сервер, або інструментарій для роботи с JWT токенами.

Головним класом застосунку є «NureApplication» він містить функцію «main» з якої починається робота програми. Розробку вебзастосунку для замовлення товарів з супермаркету розпочнемо зі створення класів сутностей в папці «domain» (рис. 3.4).

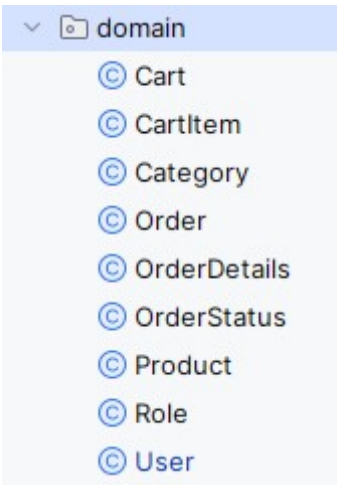

Рисунок 3.4 – Класи сутностей

Усього повинно бути дев’ять сутностей:

– кошик (Cart);
– елемент кошику (Cart Item);
– категорія (Category);

- замовлення (Order);
- деталі замовлення (Order Details);
- статус замовлення (Order Status);
- товар (Product);
- роль (Role);
- користувач (User).

Лістинг 3.1 Реалізація сутності користувача:

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table (name = "users")
public class User {
    private static final String SEQ_NAME = "user_seq";
    @Id
    @GeneratedValue (strategy = GenerationType.SEQUENCE, generator = SEQ_NAME)
    @SequenceGenerator (name = SEQ_NAME, sequenceName = SEQ_NAME, allocationSize = 1)
    private Long id;
    private String name;
    private String password;
    @ManyToMany
    @JoinTable (
        name = "user_roles",
        joinColumns = @JoinColumn (name = "user_id"),
        inverseJoinColumns = @JoinColumn (name = "role_id"))
    private Collection<Role> roles;}
```

Перед об’явленням класу «User» вказуємо наступні анотації:

– «Data» – анотація з бібліотеки Lombok. Це скорочена анотація яка поєднує у собі можливості таких анотацій як «ToString», «EqualsAndHashCode», «Getter», «Setter» та «RequiredArgsConstructor». Така анотація генерує гетери, сетери та конструктор з необхідними аргументами. Також створює методи для конвертації об’єкта в строку, для генерації хеш-функції та перевірки об’єктів на еквівалентність значимих полів [21];

– «NoArgsConstructor» – генерує конструктор без параметрів;

– «AllArgsConstructor» – генерує конструктор з одним параметром для кожного поля класу;

– «Builder» – анотація, що реалізує патерн проектування «Будівельник»;

– «Entity» – відмічає клас як сутність;

– «Table (name = "users")» – вказує на таблицю з якою потрібно асоціювати даний клас. Атрибут «name» вказує на назву таблиці в базі даних.

Далі необхідно створити поля класу «User». Усі поля у сутності користувача є приватними, тому перед визначенням типу даних поля необхідно вказати модифікатор доступу «private». Модифікатор «private» є найбільш строгим модифікатором доступу серед усі наявних у Java. Він обмежує видимість даних у межах одного класу. Цей модифікатор слугує для реалізації одного з принципів об'єктно-орієнтованого програмування, а саме інкапсуляції.

Першим полем визначимо ім’я послідовності для генерації ідентифікатора користувача. Окрім того що це поле є приватним, воно ще є статичним. Це означає, що це поле прив’язано до самого класу, а не його до його об’єкта. Такий прийом дозволяє зменшити використання пам’яті вебзастосунком. Щоб позначите поле як статичне необхідно вказати ключове слово «static» [22].

Оскільки в майбутньому не планується змінювати назву послідовності для генерації ідентифікатора користувача, то таке поле необхідно позначити

як константу. Константа – це змінна, значення якої прив’язується до того, яке їй було задано на етапі ініціалізації, і більше не змінюється. Для об’явлення константи в Java використовують ключове слово «final». Згідно з конвенцією по оформленню коду Java (Java Code Conventions) іменування констант здійснюється великими літерами з підкреслюваннями між словами, якщо назва константи складається більше ніж з одного слова.

Кожна сутність повинна мати унікальний ідентифікатор для всіх об’єктів класу, ключове поле, яке позначається анотацією «Id». Ключові поля можуть бути простими або складеним. Простий ідентифікатор складається з одного поля. У випадку сутності користувача це поле «id» з типом даних «Long». Якщо ідентифікатор складений, то його визначення створюється окремий клас, де перелічуються усі необхідні поля для ідентифікації об’єкта. В Spring Data JPA є чотири стратегії генерації унікальних ідентифікаторів для сутності [23]:

– «auto» – це означає, що JPA провайдер сам визначає оптимальний спосіб для генерації унікальних ідентифікаторів сутності;

– «identity» – при такій стратегії використовується вбудований в базу даних тип даних «identity» для первинного ключа;

– «sequence» – для генерації використовується послідовність, тобто певний об’єкт бази даних для генерації унікальних значень;

– «table» – для генерації унікального значення використовується окрема таблиця, яка емулює певну послідовність. Ця стратегія є найгіршою з точки зору продуктивності, тому її не бажано використовувати для розробки вебзастосунків, робота яких залежить від швидкодії [24].

Для генерації унікального ідентифікатора користувача використовується стратегія «Sequence». Щоб явно вказати стратегію генерації використовується анотація «GeneratedValue», де в параметрах вказується тип стратегії та назва генератора послідовності.

Створимо власний генератор послідовностей за допомогою анотації «SequenceGenerator», де в параметрах вказується назва генератору, стартове значення та шаг послідовності.

Далі необхідно створити поля з типом даних «string» для зберігання ім’я користувача та паролю. Слід стежити за тим щоб ці поля не були пустими, а поле ім’я ще й унікальним, що не дасть можливості зареєструвати у системі двох користувачів з однаковим ім’ям користувача.

Останнім полем у сутності користувача є колекція ролей. У даному випадку поле є колекцією, тому що кожен користувач може мати декілька ролей. А також одна роль може бути призначена двом або більше користувачам, тому щоб відобразити відношення багато до багатьох слід використати анотацію «ManyToMany».

Оскільки відношення багато до багатьох слід уникати, то необхідно перелік ролей користувачів винести в окрему таблицю. Для цього використано анотацію «JoinTable», яка створить нову таблицю куди будуть винесені ідентифікатори користувачів та їх ролей.

Після створення усіх класів для сутностей, необхідно створити інтерфейси репозиторіїв для взаємодії з сутностями бази даних. Перелік усіх необхідних репозиторіїв наведено на рисунку 3.5.

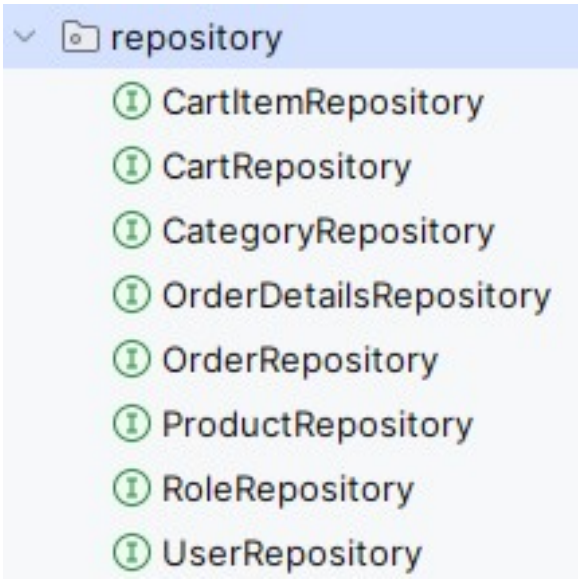

Рисунок 3.5 – Перелік репозиторіїв

Лістинг 3.2 Репозиторій користувача:

```
@Repository
public interface UserRepository extends CrudRepository<User, Long>{
   Optional<User> findByName(String name);
}
```

Інтерфейс репозиторію користувача помічений анотацією «Repository». Він наслідує вбудований в Spring Data JPA репозиторій «CrudRepository». Такий репозиторій по замовчуванню підтримує операції зчитування, створення, оновлення та видалення даних. «CrudRepository» підтримує створення власних методів користувача. Для цього навіть необов'язково писати власну реалізацію визначеного інтерфейсу, достатньо лише правильно вказати назву методу.

Метод «findByName» виконує пошук користувача в репозиторії по його імені, яке передано у метод у вигляді параметру. Результатом роботи такого методу є параметризований тип даних «Optional».«Optional» – це параметризований контейнер, який може містити об'єкт зазначеного типу, або не містити жодного значення. Такий механізм дозволяє уникнути виключення «NullPointerExceeption», яке виникає, коли замість очікуваного об'єкту застосунок отримує значення «null».

Іноді необхідно отримати не один об'єкт з бази даних, а декілька. Для цього можна використати методи «findAll», або «findAllBy», але вони повернуть усі записи, які є в таблиці.

Наприклад, якщо необхідно отримати перелік товарів у супермаркеті, то такий запит оброблятиметься досить довго, тому що кількість товарів може сягати декілька тисяч. Для цього користувачу видають товари по сторінкам, по 15 або 20 товарів, залежно від налаштувань. Відповідно створено необхідний метод у репозиторії товарів, який наведено у лістингу 3.3.

Лістинг 3.3 Репозиторій товару:

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long>
{
   Page<Product> findByCategory (Category category, PageRequest
pageable);
}
```

Метод «findByCategory» повертає товари, категорія яких співпадає з категорією, яку задано в параметрах методу. Другим параметром такого методу є «PageRequest». Це посторінковий запит, який включає в себе два значення: номер сторінки, яку необхідно повернути та кількість товарів на одній сторінці. Результатом роботи такого методу є параметризований об’єкт «Page», який є контейнером, що містить товари розміщенні на необхідній сторінці.

Аналогічно до репозиторіїв користувача та товару визначено репозиторії інших сутностей із необхідними методами. Коли усі репозиторії створено, необхідно створити спеціальні сервіси для роботи з сутностями. Перелік усіх сервісів вебзастосунку для замовлення товарів із супермаркету наведено на рисунку 3.6.

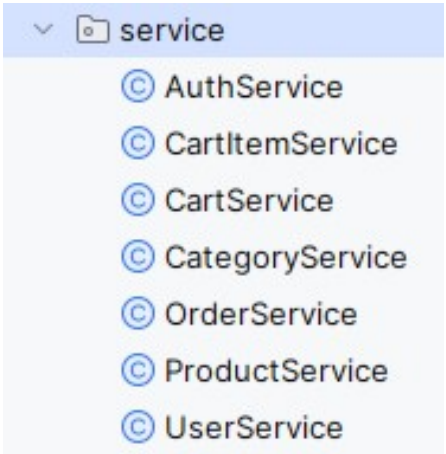

Рисунок 3.6 – Перелік сервісів

Кожен сервіс містить набір функцій для реалізації певного функціоналу вебзастосунку. Деякі сервіси не містять складної логіки, а лише є обгорткою над репозиторієм, але для таких випадків все одно створюється окремий сервіс на перспективу подальшого розвитку та ускладнення проєкту.

Для реалізацій автентифікації користувачів створено сервіс «AuthService», який містить для цього спеціальні методи.

Лістинг 3.4 Метод створення токену автентифікації:

```
    public ResponseEntity<?> createAuthToken(JwtRequest authRequest) {
  try{
     authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(authRequest.getUsername(),
authRequest.getPassword()));
  } catch (BadCredentialsException e) {
     return new ResponseEntity<>(new
AppError(HttpStatus.UNAUTHORIZED.value(),"Invalid login or password!"),
HttpStatus.UNAUTHORIZED);
  }
  UserDetails userDetails =
userService.loadUserByUsername(authRequest.getUsername());
  String token = jwtUtil.generateToken(userDetails);
  return ResponseEntity.ok(new JwtResponse(token));
}
```

Щоб отримати JWT токен авторизації користувач передає серверу дані автентифікації: логін і пароль, які об’єднано в одному об’єкті «authRequest». Далі у блоці «try/catch» відбувається спроба автентифікувати користувача за допомогою менеджера автентифікації. Якщо спроба вдала, то інструментарій для роботи з JWT генерує новий токен, якщо виникає виключення «BadCredentialsException», яке свідчить про те що користувач надав невірні

автентифікаційні дані, то генерується помилка і користувачу повертається HTTP статус із кодом 401.

Аналогічно до сервісу автентифікації створено інші сервіси з відповідними методами для роботи з товарами, замовлення та користувачами.

Зазвичай у вебзастосунках створеними сервісами користуються контролери. Контролери обробляються POST та GET запити від клієнта. Перелік усіх необхідних контролерів для роботи вебзастосунку для замовлення товарів з супермаркету наведено на рисунку 3.7.

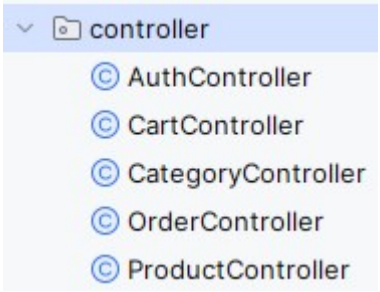

Рисунок 3.7 – Перелік контролерів

«AuthController» приймає від клієнта запити на авторизацію, реєстрацію та автентифікацію користувачів. Для цього він використовує методи сервісу «AuthService». Оскільки серверна частина є RESTful вебсервісом і усі контролери повертають дані у форматі JSON, то для створення контролерів використовується анотація «RestController». Код методу реєстрації нових користувачів наведено в лістингу 3.5.

Лістинг 3.5 Метод контролеру для реєстрації користувачів:

```
@CrossOrigin(origins = "http://localhost:5173/")
@PostMapping("/registration")
    public ResponseEntity<?> register(@RequestBody RegistrationUserDto
registrationUserDto) {
```

```
    return authService.registerNewUser(registrationUserDto);
}
```

Даний метод обробляє POST запит, тому вказано анотацію «PostMapping». Параметром такої анотації є URL-адреса, за якою необхідно здійснювати запит, щоб відпрацював заначений метод. Метод POST використовується для відправки даних на сервер. Метод GET використовується для отримання інформації від сервера.

Також вказано іншу необхідну анотацію «CrossOrigin», яка дозволяє налаштувати сумісне використання ресурсів між різними джерелами. Така анотація дозволить надсилати запити на сервер з клієнтської частини. Це важливо тому що архітектура даного вебзастосунку передбачає розділення клієнта і сервера на два різних вебресурса. В параметрах анотації вказано адресу клієнту з якого дозволяється здійснювати запити. В даному випадку це «localhost:5173», оскільки клієнт і сервер працюють на одному комп’ютері. Анотація «CrossOrigin» підтримується на рівні класу і унаслідується усіма дочірніми методами, тому її можна лише один раз вказати для класу «AuthController».

В параметрах методу реєстрації користувачів вказано анотацію «RequestBody», яка означає що даний метод очікує певні дані в тілі HTTP запиту, а саме об’єкт передачі даних «registrationUserDto». Data-transfer object є шаблоном проєктування, який переносить дані між процесами для зменшення кількості викликів [25].

Для реєстрації користувача серверу необхідно отримати ім’я користувача і пароль, але, щоб передати ці дані за один виклик, вони упаковуються в один об’єкт передачі даних.

Для функціонування вебзастосунку для замовлення товарів із супермаркету створено декілька об’єктів передачі даних. Повний їх перелік зображено на рисунку 3.8.

Рисунок 3.8 – Перелік об’єктів передачі даних

Лістинг 3.6 Код класу «RegistrationUserDto»:

```
@Data
public class RegistrationUserDto {
    private String username;
    private String password;
}
```

Пароль користувача небезпечно зберігати та передавати у відкритому вигляді, тому його слід хешувати. Для цього використовується «BCryptPasswordEncoder» вбудований в бібліотеку Spring Security. Цей механізм хешування використовує однойменний алгоритм bcrypt, він є доволі стійким та безпечним для використання у вебзастосунках. Цей алгоритм є доволі адаптивним, його можна налаштувати, штучно сповільнивши швидкість роботи, для посилення захисту від атаки перебором [26]. Але слід не перестаратися, щоб час оброки запитів залишався комфортним для користувачів. За замовчування клас «BCryptPasswordEncoder» має коефіцієнт складності 10, але цей параметр можна налаштувати за власними потребами.

На даному етапі створено мінімальний життєздатний продукт серверу вебзастосунку для замовлення товарів із супермаркету, отже розпочнемо

розробку клієнтської частини. Для розробки сайту за допомогою фреймворку React необхідно встановити деякі додаткові бібліотеки (рис. 3.9) [27].

```
12    "dependencies": {
13      "axios": "^1.6.8",
14      "bootstrap": "^5.3.3",
15      "jwt-decode": "^4.0.0",
16      "react": "^18.2.0",
17      "react-bootstrap": "^2.10.2",
18      "react-dom": "^18.2.0",
19      "react-hot-toast": "^2.4.1",
20      "react-icons-kit": "^2.0.0",
21      "react-router-dom": "^6.22.3"
22    },
```

Рисунок 3.9 – Бібліотеки необхідні для розробки клієнтської частини

Axios – бібліотека, яка є HTTP клієнтом для асинхронної відправки запитів на сервер. Axios має захист від міжсайтової підробки запитів, проста для вивчення та легка у використанні.

Bootstrap – бібліотека, що дозволяє швидко та ефективно розробляти сучасні та адаптивні сайти. Містить у собі певні набори html та css шаблони, які вбудовуються у код за допомогою визначених тегів.

Jwt-decode – бібліотека для декодування JWT токенів. Важливо розуміти, що ця бібліотека не валідує токени. Її використання передбачає, що сервер заздалегідь генерує і надсилає правильний токен, якщо це дійсно так, то Jwt-decode допоможе отримати з JWT усі публічні дані.

React-dom – бібліотека, що містить інструментарій для роботи з DOM та засоби рендерингу вебсторінки.

React-hot-toast – бібліотека, яку розробники позиціонують як найкраще рішення для створення естетичних сповіщень на сайті, які можна легко кастомізувати за власним смаком.

React-icons-kit – набір значків для покращення візуалу вебзастосунку. Містить у собі 13 тематичних пакетів значків, які можна використовувати не зберігаючи зображення на сервері.

React-router-dom – бібліотека для реалізації навігації між різними частинами сайту. Дозволяє змінювати вміст сторінки без перезавантаження браузера, що є актуальним для розробки SPA.

Перед початком розробки клієнтської частини вебсайту створено новий React проєкт, який сформує наступну структуру папок (рис. 3.10).

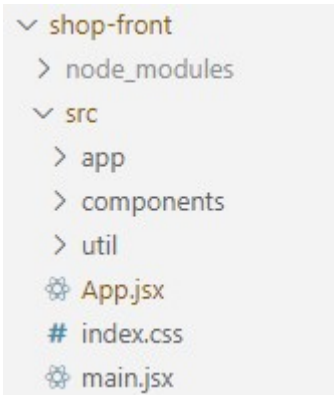

Рисунок 3.10 – Структура папок клієнтської частини

Папка «node_modules» використовується для зберігання модулів, які необхідні для вебзастосунку. Ця папка автоматично створюється з кожним новим проєктом.

Папка «src» містить вихідний код проєкту і включає в себе такі папки як: «app», що містить HTML та CSS код сторінок сайту, «components» – містить react-компоненти, «util» – зберігає JavaScrpt функціональні скріпти.

Файл «main.jsx» – кореневий jsx компонент проєкту. «App.jsx» – містить головну логіку роботи клієнтської частини вебзастосунку. «index.css» – містить каскадні таблиці стилів, які будуть застосовуватися до усіх сторінок сайту.

На основі побудованого каркасу папок створено необхідні сторінки для функціонування вебзастосунку. Створено сторінки реєстрації та авторизації користувачів (рис. 3.11).

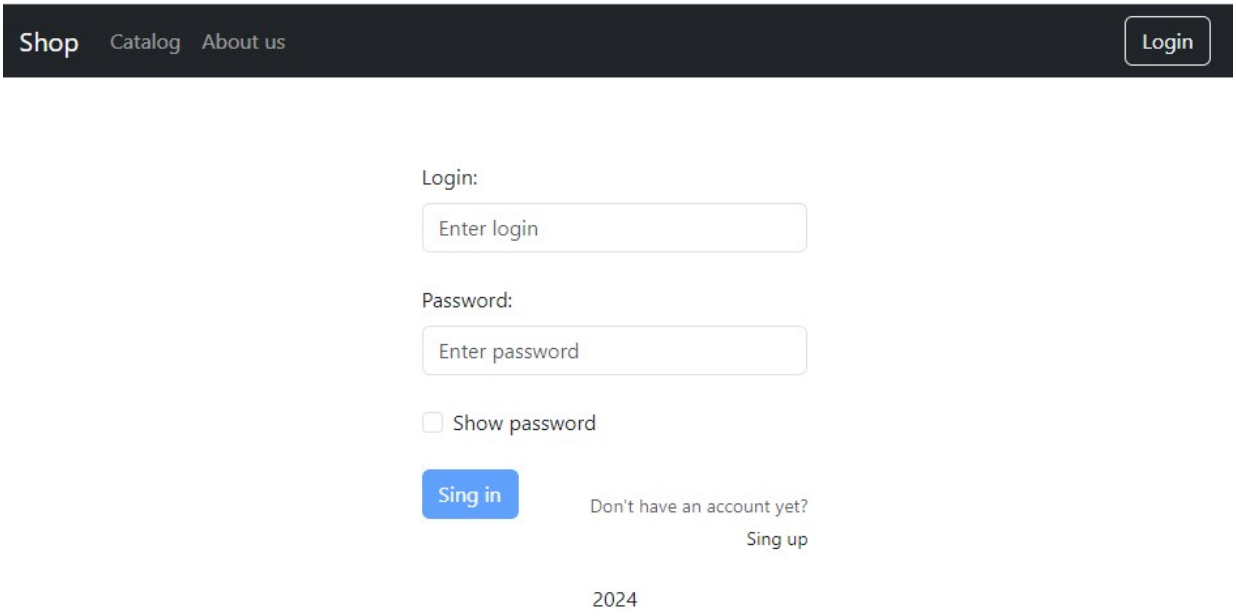

Рисунок 3.11 – Сторінка авторизації користувача

Додатково створено сторінку каталогу товарів, та окремі сторінки для кожного товару. Для авторизованого користувача створено сторінки кошика та заказів, а також інші необідні сторінки. Усі сторінки клієнтської частини взаємодіють з одним сервером, тому визначено базову URL адресу до якої звертається сайт.

Лістинг 3.7 Створення нового екземпляру axios:

```
const $host = axios.create({
    baseURL: "http://localhost:8080/",
});
```

«Axios.create» – це зручна функція бібліотеки, яка використовується для створення нового екземпляру з індивідуальною конфігурацією. За допомогою цієї функції створено клієнт для доступу до попередньо створеного API серверу. Таку конфігурацію можна використовувати для будь-яких запитів, з одного й того самого клієнту, вказавши базову URL адресу серверу. Створений клієнт може надсилати запити GET, POST, PUT, DELETE, PATCH і т.д [28, 29].

На основі створеного HTTP-клієнта розроблено функціонал необхідний для сторінок вебзатосунку. Створено функції, що надсилаються запити на сервер для реєстрації, авторизації, деавторизації користувача, додавання та видалення товарів у кошику, оформлення та скасування замовлення та інші додатково необхідні функції. Детально розглянемо функцію клієнту, що надсилає запит серверу на додавання товару у кошик користувача.

Лістинг 3.8 Функція додавання товару в кошик:

```
const buy = async (id) => {
  await $host.post(
    "/cart/add",
    { id },
    { headers: { Authorization: `Bearer ${auth.token}` } }
  );
};
```

Функція для додавання товару у кошик є стрілковою асинхронною функцією, яку присвоєно константі. Вона має один параметр – ідентифікатор товару, який необхідно додати у кошик. Функція «buy» використовує створений axios клієнт, який надсилає POST запит, на визначену URL адресу. До такого запиту додаються деякі дані, а саме, ідентифікатор товару, який отримується функцією у вигляді параметру, а також набір заголовків запиту, серед яких обов’язковими є токен користувача, бо лише авторизовані користувачі можуть додавати товари у кошик.

На даному етапі, коли створено базу даних, сервер для обробки запитів та клієнтську частину для зручної взаємодії, розробку вебзастосунку для замовлення товарів з супермаркету можна вважати завершеною.

3.3 Тестування реалізованого вебзастосунку та аналіз результатів

Після того як розробка клієнтської та серверної частини завершена, розпочнемо тестування вебзастосунку. Для тестування серверної частини використано програмне забезпечення Postman (рис. 3.12).

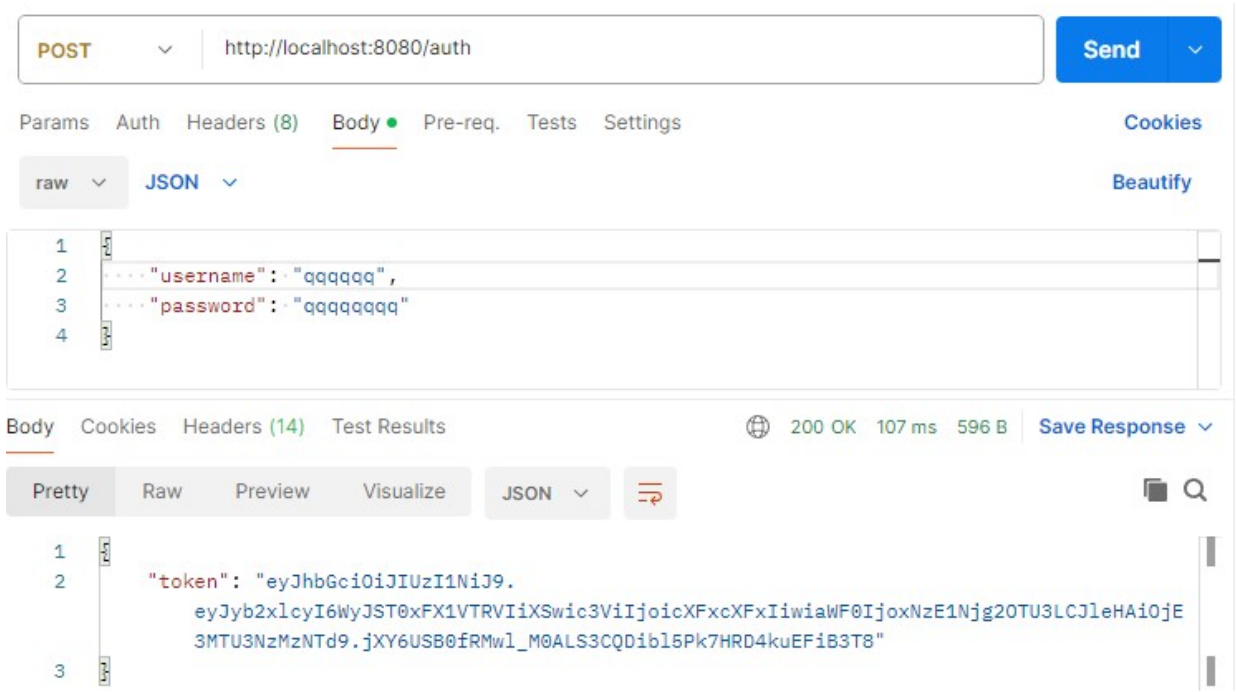

Рисунок 3.12 – Тестування запиту авторизації у Postman

Протестована можливість авторизуватися та отримати токен доступу для користувача, обліковий запис якого було попередньо створено в базі даних. Для цього у програмному забезпечені Postman було вказана URL адреса серверу для авторизації. На вкладці «Body» у форматі JSON передано дані авторизації, а саме ім’я користувача та пароль. Після такої підготовки, натиснута кнопка «Send» і Postman друкує відповідь, яка містить JWT токен також у форматі JSON [30].

Далі протестовано чи валідний токен надсилає сервер. Для цього URL адресу у Postman змінено на отримання даних про товари, які клієнт додав у кошик. Метод запиту змінено з POST на GET. Щоб сервер зрозумів про якого саме користувача необхідно надати інформацію, до запиту додається токен,

який був отримай у попередньому тесті. У вкладці «Authorization» необхідно вказати тип авторизації «Bearer Token» та вказати отриманий раніше токен. Після такої підготовки, натиснута кнопка «Send» і Postman друкує відповідь, яка містить інформацію про товари, які власник JWT токену додав до свого кошику (рис. 3.13).

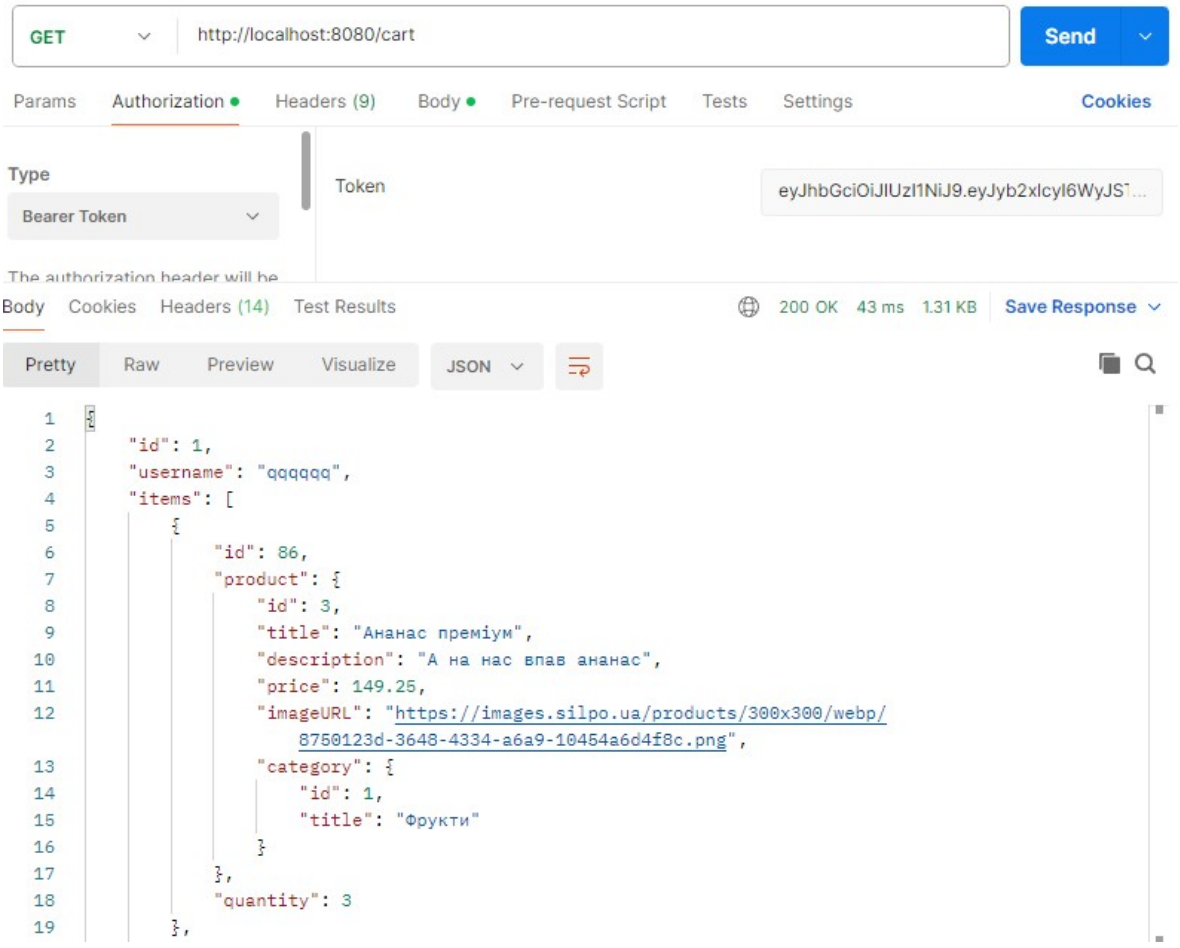

Рисунок 3.13 – Тестування запиту кошику користувача у Postman

Також за допомогою програмного забезпечення Postman було протестовано інші можливотсті серверу повязані не лише з отриманням інформації, а й з її редагуванням, видаленням та створенням [31].

Впевнившись, що сервер працює так як це було задумано та спроектовано, перейдемо до тестування клієнтської частини вебзастосунку.

Відкривши сторінку авторизації користувача, небхідно протестувати поля для вводу імя користувача та паролю. Також, якщо поставити позначку

навпроти напису «Show password», небхідно перевірити, що пароль введений у відповідному полі стає явно видним і більше не замінюється на спецсимволи.

Перейдемо на сторінку кошику користувача та звіримо дані, які відображає клієнтська частина та які надсилає програмне забезпечення Postman (рис. 3.14). Найменування та кількість товарів повинні повністю співпадати. Також протестовано функціонал усіх кнопок сайту. Кількість товарів у кошику змінюється, коли користувач натискає кнопки «+» та «-». Також необхідно перевірити, що товар повністю видаляється з кошику, якщо натиснути червону кнопку «X». Перевірено чи правильно підраховується загальна сума замовлення, а також чи працює кнопка «Оформити доставку», чи перенаправляє вона клієнта на необхідну сторінку.

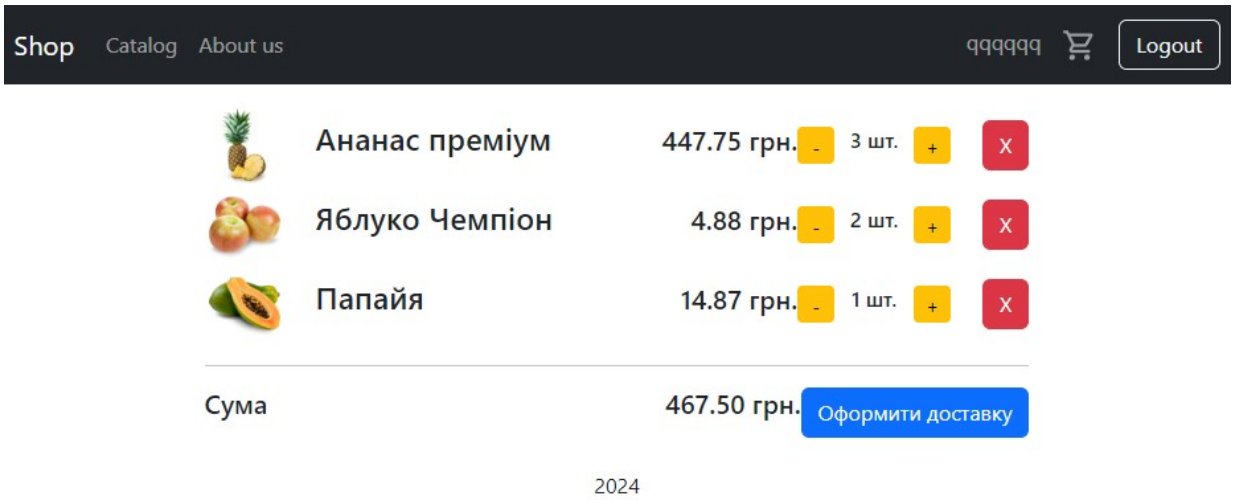

Рисунок 3.14 – Тестування зовнішнього вигляду кошику користувача

Окрім сторінок, які доступні лише для авторизованих користувачів, були протестовані сторінки доступні неавторизованим користувачам. Перевірено правильність відображення сторінки каталогу товарів (рис. 3.15). Кожна карточка товару відповідає своєму продукту. Ціна, назва, опис та зображення співпадають з тим, які зберігаються в базі даних. Протестована робота кнопок «Купити», яка додає обраний товар до кошика. Також усі

товари розподілені по категоріям, протестовано, що сортування товарів за категорією коректно працює.

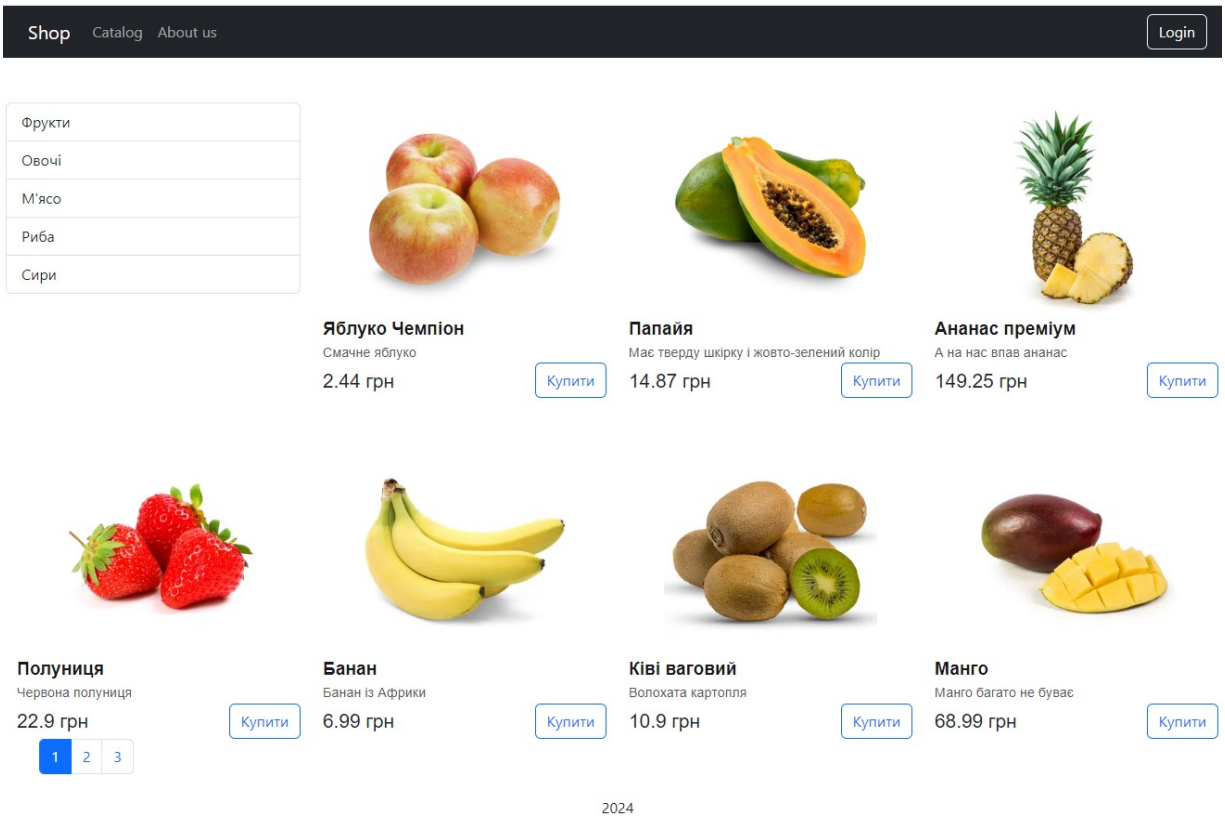

Рисунок 3.15 – Тестування сторінки каталогу

Протестована система відображення товарів по сторінкам. Кожна сторінка відображає по 7 товарів, як це налаштовано по замовченню. Сторінки коректно змінюються і відображаються нові товари [32].

В ході тестування було проведено перевірку функціоналу серверної та клієнтської частини вебзастосунку. Змоделювано детальні сценарії взаємодії користувача із системою, щоб впевнитись чи усі частини програми працюють належним чином. Тестування пройдено успішно, вебзастосунок має увесь наявний перелік функцій, критичних дефектів та недоліків системи виявлено не було. Це означає, шо розроблене програмне забезпечення у подальшому може працювати з реальними клієнтами.

3.4 Перспективи подальшої роботи

Розроблений вебзастосунок має мінімально-необхідний набір функціоналу для коректної роботи інтернет-магазину для замовлення товарів із супермаркету. Розглянута предметна область доволі швидко та ефективно розвивається, а отже має значну перспективу для удосконалення та додавання нового функціоналу.

Наразі можна виділити декілька перспективних напрямків та ідей для подальшого розвитку та удосконалення вебзастосунку:

– адаптація інтерфейсу користувача для смартфонів та планшетів або створення клієнтського застосунку для мобільних платформ;

– розширення функціоналу адміністратора, який дозволить ефективніше керувати товарами, користувачами та замовленнями;

– виділення системи авторизації, автентифікації та реєстрації в окремий мікросервіс для розвантаження основної системи;

– покращити систему реєстрації, надсилаючи перевірочний лист на електронну пошту, яка необхідно вказати при реєстрації. Таке рішення захистить систему від зловмисної реєстрації пустих облікових записів;

– реалізувати систему зміни та паролю через електрону пошту, на випадок, якщо користувач забув пароль, який вказував при реєстрації.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений вебзастосунок для замовлення товарів із супермаркетів.

Перед початком розробки було вивчено та проаналізовано існуючі альтернативи проєкту, а також проведено порівняльну характеристику сучасних технологій розробки вебзастосунків.

В процесі роботи було успішно застосовано накопичені знання про предметну область та використані сучасні методики розробки сайтів. Досягнута головна мета роботи, а отже виконано усі поставлені задачі.

Створений сайт відповідає усім поставленим вимогам та має базовий функціонал інтернет-магазину для замовлення товарів із супермаркету. Система підтримує технологію для користувачів з різними рівнями доступу, дозволяє швидко та зручно створювати замовлення авторизованим користувачам та ефективно керувати замовленнями адміністраторам.

У ході розробки була використана мова програмування JavaScript для клієнтської частини застосунку, а серверна частина реалізована за допомогою мови Java та фреймворку Spring. Було продемонстровано навички та вміння роботи з базою даних PostgreSQL.

На фінальному етапі розробки було проведено ретельне тестування створеної системи на відповідність поставленим вимогам та критеріям якості програмного забезпечення.

Після успішного тестування розглянуто перспективи подальшої роботи та можливості по вдосконаленню проєкту. Виділено основні аспекти роботи застосунку, покращення яких призведе до ефективнішої роботи системи в цілому.

Отже, розроблений вебзастосунок для замовлення товарів із супермаркету є надіним та ефективним рішенням для здійснення роздрібної торгівлі в мережі інтернет, який задовольняє усі потреби користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. АТБ-Маркет. URL: https://www.atbmarket.com/ (дата звернення 27.04.2024).

2. Сільпо. URL: https://silpo.ua/ (дата звернення 27.04.2024).

3. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

4. Огляд фреймворків JavaScript. Що, для чого і коли використовувати. URL: https://dou.ua/forums/topic/34739/ (дата звернення 30.04.2024).

5. Saks, E. (2019). JavaScript Frameworks: Angular vs React vs Vue.

6. PYPL PopularitY of Programming Language index. URL: https://pypl.github.io/PYPL (дата звернення 30.04.2024).

7. Tvoroshenko, I., & Andrieieva, A. (2021). Development of web applications for remote learning of English.

8. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент.

9. Cross-Origin Resource Sharing (CORS) – HTTP | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (дата звернення 02.05.2024).

10. JSON Web Tokens. URL: https://jwt.io/ (дата звернення 30.04.2024).

11. Машталір, С. В., & Гавришева, К. А. (2023, May). ОСОБЛИВОСТІ ПРОЕКТУВАННЯ ЗАСТОСУНКІВ ІЗ ВИКОРИСТАННЯМ ВИСОКОРІВНЕВОЇ МОВИ ТА ЗГІДНО З ОБ'ЄКТНО-ОРІЄНТОВАННОЮ ПАРАДИГМОЮ. In *The 3 rd International scientific and practical conference "Modern problems of science, education and society"(May 22-24, 2023) SPC "Sci-conf. com. ua", Kyiv, Ukraine. 2023. 1522 p.* (p. 427).

12. Kuzomin, O., Tolmachova, T., & Astappiev, O. (2017). Analysis of Web user activity data. International Journal of Information Models and Analyses, 6(2), 108-118.

13. Kornienko, D. V., Mishina, S. V., & Melnikov, M. O. (2021, November). The Single Page Application architecture when developing secure Web services. In Journal of Physics: Conference Series (Vol. 2091, No. 1, p. 012065). IOP Publishing.

14. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, Сучасні інформаційні системи, 7(1), С. 5-13.

15. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

16. Mai, N. (2020). E-commerce Application using MERN stack.

17. Творошенко, І. С. (2021). Технології прийняття рішень в інформаційних системах.

18. Spring initializr. URL: https://start.spring.io/ (дата звернення 02.05.2024).

19. Walls, C. (2022). Spring in action. Simon and Schuster.

20. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

21. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, Telecommunications and Radio Engineering, 79(10), pp. 855-863.

22. Творошенко, І. С. (2018). Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів моделювання складних систем. Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland, 118-121.

23. Kinoshenko, D., Mashtalir, V., Yegorova, E., & Vinarsky, V. (2005). Hierarchical Partitions for Content Image Retrieval from Large-Scale Database. *Machine Learning and Data Mining in Pattern Recognition*, 445.

24. Tvoroshenko, I. (2019). Development of models of spatial analysis of status of interactive processes of complex systems.

25. Fowler, M. (2012). Patterns of enterprise application architecture. Addison-Wesley.

26. Shafronenko, A., Bodyanskiy, Y. (2019). Online algorithm for possibilitic fuzzy clustering based on evolutionary cat swarm optimization.

27. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

28. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

29. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746

30. Види тестування ПЗ. URL: http://qalearning.com.ua/theory/lectures/material/testing-types-functional/ (дата звернення 05.05.2023).

31. Tvoroshenko, I. S., & Maksimenko, H. (2021). To the question of analysis of existing mechanisms of web application testing.

32. Williams, W. C. (2019). Spring and all. Dover Publications.