Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Кафедра інформаційних технологій та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: «Розроблення вебзастосунку "Бакалійний Дім" з використанням фреймворку Spring»

Виконав студент 2 курсу, групи ІСТС-21

спеціальності:

126 „Інформаційні системи та технології"
(шифр і назва напрямку підготовки спеціальності)

Вільгельм А.М.
(прізвище та ініціали)

Керівники: ***Івасюк Р.В.***
Павлюк У.В.
(прізвище та ініціали)

Рецензент Процик Ю. С.
(прізвище та ініціали)

Львів - 2024

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ Комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та комп'ютерного моделювання

Рівень вищої освіти перший (бакалаврський)

Спеціальність 126 „Інформаційні системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри ІСКМ

_____ Сторожук О.Л.

"06" 02 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Вільгельму Артуру Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення вебзастосунку "Бакалійний Дім" з використанням фреймворку Spring»

Керівники проекту: Івасюк Руслан Васильович

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

Павлюк Уляна Володимирівна, к.е.н

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

Затверджені наказом вищого навчального закладу від 6 лютого 2024р. № С-87

2. Термін подання студентом роботи 10 червня 2024р.

3. Вихідні дані до роботи *Розробити вебзастосунок для онлайн-замовлення продуктових товарів, який буде простим та функціональним. Розробити Front-end частину для інтерфейсу застосунку та розробити Back-end частину для обробки даних з базою даних.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1)Стан проблемної області

2)Інформаційне забезпечення

3)Програмне забезпечення

4)Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка презентаційного матеріалу до доповіді (слайди для доповіді загальним обсягом 5-12 слайдів)

6. Дата видачі завдання 7 лютого 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	08.02.2024-21.02.2024	Виконано
2.	Постановка задачі і її формалізація	21.02.2024-25.02.2024	Виконано
3.	Виконання вхідного етапу технології	26.02.2024-10.03.2024	Виконано
4.	Реалізація головних алгоритмів проекту	10.03.2024-07.04.2024	Виконано
5.	Виконання етапу відлагодження проекту	07.04.2024-06.05.2024	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	08.05.2024-31.05.2024	Виконано
7.	Оформлення записки до дипломного проекту.	01.06.2024-10.06.2024	Виконано

Студент _____ Вільгельм А.М
(підпис) (прізвище та ініціали)

Керівники роботи _____ Івасюк Р.В.
(підпис) (прізвище та ініціали)

_____ Павлюк У.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 48 сторінок пояснювальної записки, 55 рисунків, 3 додатки, 15 джерел. Дипломна робота розповідає про розробку вебзастосуноку для онлайн-замовлення продуктових товарів. Під час розробки вебзастосунку було використано декілька фреймворків: React, Spring Boot, Spring Security, Spring Data JPA.Spring Security потрібен для авторизації та аутентифікації, також Spring Data Jpa для з'єднання з базою даних, щоб сервер міг обробляти дані, React для створення інтерфейсу сайта.

Ключові слова: Java, Spring Boot, Spring Security, React, MySQL, Tailwind CSS, MUI.

ABSTRACT

The thesis contains 48 pages of explanatory note, 55 figures, 3 appendices, 15 sources. This thesis is about the development of a web application for online grocery ordering. Several frameworks were used to develop the web application: React, Spring Boot, Spring Security, Spring Data JPA.Spring Security is needed for authorisation and authentication, also Spring Data Jpa for connecting to the database so that the server can process the data, React for creating the site interface.

Keywords: Java, Spring Boot, Spring Security, React, MySQL, Tailwind CSS, MUI.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити вебзастосунок для онлайн замовлення продуктових товарів, який буде простим та функціональним, забезпечити зручність для користувачів. Створити базу даних з інформацію про покупця, продуктів, замовлень. Реалізувати реєстрацію та авторизацію користувачів.

Використати стеки: Java, Spring Boot, JPA Hibernate для роботи з базою даних MySQL. Javascript, React, MUI, Tailwind CSS для розроблення клієнтського інтерфейсу. Інтерфейс користувача мусить бути такі компоненти:

- Реєстрація та авторизація користувачів.
- Перегляд каталогу товарів.
- Пошук товарів.
- Додавання товарів до кошика.
- Оформлення замовлення.
- Перегляд історії замовлень.

Використати Spring Security для авторизації та аутентифікації користувачів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

JPA(Java Persistence API) - стандарт Java для роботи з об'єктно-реляційним відображенням даних в програмах

DTO(Data Transfer Object) - шаблон проектування в програмуванні, для передачі даних

JWT (Json Web Token) - використовується для автентифікації та передачі даних

MUI(Material UI) – бібліотека React

JCE – (Java Cryptography Extension) – стардартне розширення для платформи Java

ВСТУП

Інформаційні технології все більше проникають у різні аспекти нашого життя, пропонуючи нові способи покращення якості обслуговування у різних сферах, включаючи роздрібну торгівлю. Однією з найважливіших галузей, що активно розвивається завдяки IT-рішенням, є роздрібна торгівля продуктами харчування. Вебзастосунки для управління бакалійними магазинами стають дедалі популярнішими, оскільки вони забезпечують зручність для споживачів та ефективність для бізнесу. Актуальність теми. Актуальність теми обумовлена зростанням попиту на онлайн-сервіси для здійснення покупок, особливо в умовах глобальної пандемії, яка значно змінила поведінку споживачів та бізнес-процеси. Вебзастосунки, що дозволяють швидко та безпечно замовляти продукти харчування, забезпечують не лише зручність для клієнтів, але й сприяють оптимізації бізнес-процесів та зменшенню витрат для підприємств. Об'єктом дослідження є створення вебзастосунку "Бакалійний дім" на базі Spring Boot. Акцент роботи робиться на вивченні та реалізації сучасних підходів до розробки вебзастосунків, зокрема використання принципів архітектури REST та інтеграції з фронтенд технологіями. Предметом дослідження є технічна реалізація вебзастосунку для онлайн продажу бакалійних товарів з використанням мови програмування Java та фреймворку Spring Boot, а також фронтенд технологій React, Tailwind та MUI. Робота передбачає використання таких інструментів, як Spring Security для забезпечення безпеки, Hibernate/JPA для роботи з базою даних та інтеграції з MySQL.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

Проблеми офлайн продажу продуктами харчування:

Нестача кваліфікованих персоналу

- Не усіх офлайн магазинах є можливість знайти індивідуальний підхід до покупців через нестачу інформації про їхні вподобання.
- Не завжди є можливість вчасно та якісно реагувати на змінні інтересу покупців.

Ефективність ресурсів

- Керування запасами,опрацювання замовлень, оплата з покупцями потребують великих людських ресурсів та часу.
- Є великий ризик виникнення людських помилок під час обробки замовлення та керування складські запаси.

Обмеження в часі та просторі

- Звичайні магазині обмежані робочами часу, що виникає незручності для клієнтів, які не мають можливості прийти в магазин їхній зручний час.
- Локаційне місцезнаходження магазину може бути далеко для клієнтів особливо для тих, хто проживає в селі.

ми під час розроблення вебзастосунків для продуктових товарів:

вторення зручного клієнтський інтерфейсу

- Розроблення простий та зручного інтефейсу для користовачів з різними рівнеи

ключення зовнішній сервіси

- Необхідність підключити платіжні системи для обробки онлайн-платіжів.

3. Технічні проблеми

- Вимоги до швидкості завантаження сторінок
- Забезпечення безпеки даних користувачів та захист від кібератак.

Сувора безпека вебдодатків має вирішальне значення для успіху в Інтернеті, тому безпеці вебдодатків також приділяється велика увага. вразливості включають недостатній захист транспортного рівня, витік інформації, міжсайтовий скриптинг та SQL-ін'єкції. Локальний стан клієнта - це масив повідомлень. Компонент містить логіку для оновлення повідомлень за допомогою API чату через HTTP. Після успішного оновлення для повідомлень буде встановлено нове значення. Старий стан повідомлень не має значення. Цікавою частиною є спосіб надсилання повідомлень. Ми не керуємо його успішним результатом. Маршрут GET, відповідає за пошук у базі даних і повернення результату. Маршрут POST для нас найбільш цікавий. Він оновлює базу даних новим повідомленням і в разі успіху повертає клієнту порожній json. Однак одразу після вирішення відповіді сервер також транслює повідомлення MESSAGE_EVENT усім підписникам. Повертаючись до коду клієнта, він містить екземпляр клієнта WS, який очікує того самого MESSAGE_EVENT. Після отримання подія ініціює оновлення місцевого стану. повідомлення WS не має корисного навантаження. Його єдина мета – інформувати клієнта про зміни в стані бази даних. За отримання оновленого стану відповідає сам клієнт.

1. Клієнт створює дії

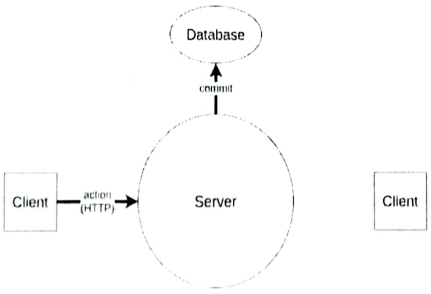


Рисунок 1.1 – Клієнт створює дії

У нашому випадку це нове повідомлення. Використовується протокол HTTP. Сервер вносить зміни до бази даних. Клієнт отримує відповідь без будь-якого корисного навантаження. Повідомлення відправлено.

2. Сервер транслює подію

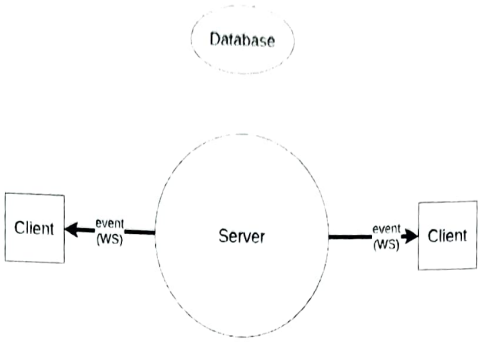


Рисунок 1.2 – Сервер транслює подію

Зміну прийнято. Наступним кроком сервер транслює подію про нове повідомлення всім абонентам. У цьому випадку за допомогою протоколу WS. На цьому етапі клієнти знову не отримують корисного навантаження.

3. Клієнти синхронізують стан

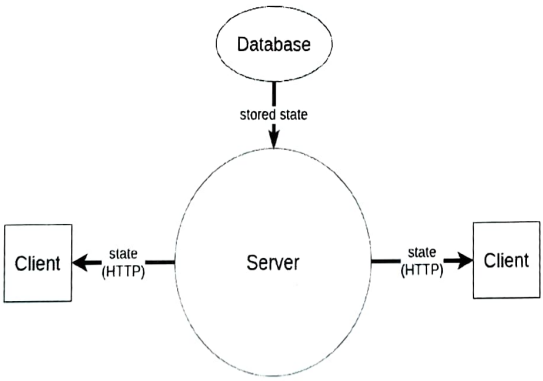


Рисунок 1.3 – Клієнти синхронізують стан

Підписані клієнти, викликані подією нового повідомлення, оновлюють свої локальні стани за допомогою HTTP. На цьому кроці передається оновлений стан.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Java

Рисунок 2.1 – Логотип Java

Java – це строго типізована мова програмування, розроблена в 1995 року компанією «Sun Microsystems» . Засновником мови програмування Java був Джеймс Артур Гослінг. Гослінг почав розробляти проект мови програмування Java в липні 1991 року, для використання його в одному із своїх багаточисельних проектів set-top box. Мова спочатку називалась Oak «Дуб» на честь дуба, який ріс перед офісом Гослінга, але в кінці вибір був зупинений на Java, назва була вибрана із списку випадковим чином. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». Java має подібний синтаксис з C та C++, адже взято за основу об'єктну модель C++, але її модифіковано.

Різниця між Java від C++

1. Парадигма програмування:

- **C++:** Багатопарадигмова мова, що підтримує як процедурне, так і об'єктно-орієнтоване програмування (ООП).
- Java: Чисто об'єктно-орієнтована мова, де все описується за допомогою класів та об'єктів.

2. Компіляція:

- **C++:** Компілюється в машинний код, який має можливість бути запущеним навіть на процесорі.

- Java: Компілюється в байт-код, який виконується віртуальною машиною Java (JVM) на будь-якій платформі, що має JVM. Це означає що мова Java є кросплатформною мовою [4].

Чому Java є хороший вибір у написаній коду?

1. По-перше Java славиться універсальністю, це значить що її можна використовувати для широкого спектру програм:
   - створення вебзастосунків
   - створення програми для ком`ютерів
   - створення комп'ютерні ігри прикладом є гра «Minecraft»
   - створення мобільних додатків
2. По-друге Java є надійною мовою. Java славиться своєю надійністю та стійкістю до помилок. Компілятор у Java проводить ретельну перевірку коду, це означає що допомогає виявити помилки в стадії розробки програми. Java також має вбудовані функції захисту JCE. JCE це технологія, яка допомагає захистити програми від хакерських атак.
3. По-третє Java є високопродуктивна мова програмування, яка має можливість ефективно обробляти дані. Компанія «Oracle» постійно оптимізує віртуальну машину JVM в кожних оновлення, що має можливість забезпечити максимальну продуктивність.
4. По-четверте Java є легка мова. Порівняно з іншими мовами програмування Java є легка мова. Її синтаксис є легкий і чіткий, є багато безкоштовних навчальних ресурсів.
5. По-п'яте у Java є багато бібліотек та інструментів. Для Java має багато бібліотек та інструментів, які мають можливість облегшувати програмістів у розробці програм.

Java в веброзробці

Мова програмування Java часто використовують у веброзробці в Back-end частині, розробляють багато різноманітних типів веб-застосунків наприклад:

- Вебсайти електронної комерції
- Вебсайти соціальних мереж
- Вебсервіси
- Вебпортали

Основним фреймворком для розроблення вебзастосунку на мові Java це Spring framework.

Java у створенні мобільних додатків

Java вже багато років є однією найпопулярноших мов програмування для розробки мобільних додатків і ось причини цьому:

1. Поширеність на Android:

- Java є офіційною мовою програмування для операційною системою Android, що робить її природним вибором для розробки програм під цю ОС.
- Більше 93% мобільних додатків для Android розроблено на мові Java, що значить про її зрілість та надійність

2. Продуктивність та масштабованість:

- Java відома своєю високою продуктивністю та масштабованістю, що робить її ідеальною для створення складних мобільних додатків.
- Її віртуальна машина (JVM) забезпечує ефективне виконання коду на різних пристроях.

3. Багатий набір бібліотек та фреймворків:

- Існує безліч бібліотек та фреймворків Java, які допомагають розробникам швидко та легко створювати мобільні додатки.
- Серед популярних фреймворків - Android SDK, Spring Mobile та Apache Cordova.

4.Універсальність:

Java не обмежується лише розробкою Android. Її можна використовувати для створення мобільних додатків для інших платформ, таких як iOS, BlackBerry та Windows Phone.Це робить Java чудовим вибором для розробників, які хочуть створювати кросплатформні мобільні додатки.

Java у розробці десктопних додатків

Java використовують не лише для розробки мобільних додатків та у веброзробці, але й для створення десктопних програм. Її потужність, гнучкість та кросплатформність роблять її популярним вибором серед розробників. Популярні фреймворки Java для розроблення десктопних додатків:

- SWT
- JavaFX
- Swing
- Eclipse RCP

2.2 Поняття про веброзробку

Рисунок 2.2 – Логотипи сервісів для вебзастосунку

Вебпрограмування – це створення сайтів та програм у мережі. Веброзробка є 2 напрямки це Front-end розробка і Back-end розробка. Front-end – це графічна частина

вебсайтів з якою користувач може взаємодіяти. Це те, що користувач бачить та з чим можете взаємодіяти на екрані, наприклад:

- Інтерфейс користувача (UI): елементи, з якими користувач безпосередньо взаємодіє, такі як кнопки, меню, текстові поля та зображення.
- Візуальний дизайн: кольори, шрифти, макет та загальний естетичний вигляд додатку.

Основні технології Front-end включають:

- HTML
- CSS
- JS

HTML(агнл. HyperText Markup Language) – це мова гіпер тексту, яка використуються для створення структури вебсторінки.

CSS(англ. *Cascading Style Sheets) - це* мова стилів, яка використовується для управлінні зовнішнього вигляду вебсторінки.

JS(Javascript) - це динамічна, об'єктно-орієнтована, прототипна мова програмування. Javascript часто використовують для створення інтерактивних вебсторінок. Javascript дає можливість додавати динамічні елементи до вебсторінок, це означає що робить більш зручним для користовачів. Наприклад за домомогою Javascript можна розробити:

- Анімації
- Ігри
- Інтерактивні форми
- Реакції на події користувача (наприклад, кліки мишею, натискання клавіш)

Популярні фреймворки для розроблення Front-end є:

JQUERY

JQuery – це бібліотека JavaScript, яка полегшує роботу з DOM, обробку подій, анімацію та AJAX. Вона надає простий у використанні API

React

React - це JavaScript-бібліотека з відкритим кодом,компанією Meta розроблена для створення динамічних інтерфейсів користувача. Її перше публічне використання відбулося у 2013 році, а з того часу вона стала однією з найпопулярніших технологій front-end розробки.

Vue.js

Vue.js – це прогресивний JavaScript-фреймворк з відкритим кодом, створений для створення інтерфейсів користувача. Він був вперше офіційно випущов у 2014 року, але його розробка почалася раніше, у 2013 році, Еваном Ю, колишнім інженером Google. Vue.js доволі швидко набрав популярності завдяки своєму простому синтаксису, гнучкості та інтуїтивно зрозумілому підходу. Подібний до React, але також включає в себе деякі функції Angular, такі як двостороння прив'язка даних.

Angular

ngular- це фреймворк JavaScript з відкритим кодом, розроблений та підтримуваний ogle. Він використовується для створення односторінкових вебдодатків (SPA) та адних вебінтерфейсів. Angular відомий своєю структурованою архітектурою, онентним підходом та широким набором функцій, що робить його популярним ом для розробки масштабних вебдодатків. Angular має багату історію, що сягає року, коли він був вперше представлений як AngularJS. З того часу фреймворк ов значні зміни, еволюціонуючи до Angular, випущеного у 2016 році. Angular використовується у широкому колі проектів, від стартапів до корпоративних . Його популярність обумовлена його здатністю створювати надійні, бовані та динамічні вебдодатки.

ack-end – це частина вебдодатку або вебсайту, якою називають «внутрішнім» прихована від користувачів. Різниця від Front-end, який відповідає на

візуальний стиль інтерфейсу та взаємодію з користувачем, Back-end виконує як сервер і забезпечує механізм сайту.

Основні функції Back-end:

- Обробка даних. Back-end працює ролі сервера сайта і отримує різні запити від користувачів і має можливість обробити їх і після обробки даних може відправити дані у відповідь. Прикладом є як регістрація користувачів, автентифікація, має можливість доступ до бази даних, записувати нові рядки в таблиці або видаляти.
- Логіка сайту. Back-end виконує логіку роботи вебсайту або вебзастосунок. Це входить умови, за якими користувач отримувати доступ до інформації або контенту.
- Безпека. Back-end виконує функцію захисту даних користувачів так і захист сайту від злому. У захист входить шифрування даних, аунтефікацію користувачів та захист від кіберзагроз.
- Робота з базою даних. Back-end зазвичай працює з базою даних для зберігання та отримання даних. Це може включати дані користувачів, продукти, замовлення, або будь-яку іншу інформацію, необхідну для роботи сайту.
- Інтеграція: Back-end може інтегруватися з іншими системами та сервісами, такими як платіжні сервісами, системи доставки або соціальні мережі.

Мови програмування які можуть бути ролі Back-end:

- Python
- Java
- C#
- PHP
- Ruby
- Javascript

Фреймворки для Back-end є:

- Django(Python)
- Spring(Java)
- .NET(C#)
- Laravel(PHP)
- Node.js(Javascript)
- Express.js(Javascript)

Майже завжди Back-end розробці є створення та підключення бази даних. На сьогоднішній час у Back-end набувають популярності використовувати такі мови програмування з бази даних це SQL та NoSQL. SQL(*Structured query language*)- це одна з найпоширеності мов програмування для створення бази даних. Його розробили в 1974 року у компанії «IBM», щоб мати можливість надати легкий та універсальний спосіб взаємодіяти з реляційними базами даних. SQL легко можна вивчити, тому він набрав популярності у програмістів [8]. На сьогоднішній час різні компанії розробили безліч модифіковані версії SQL такі як:

- MySQL
- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- MariaDB
- SQLite
- Firebird

NoSQL – це мови програмування для створення бази даних, щоб мати можливість надати легкий та універсальний спосіб взаємодіяти з нереляційними базами даних. Відрізняється від традиційних мов програмування для бази даних, в не використовуються таблична схема рядків і стовпців. Його почали використовувати в кінцу 90-х роках, а набув популярності в 2010 року. NoSQL немає фіксованої схеми,

тому можна динамічно додавати, видаляти, змінювати поля без шкоди бази даних. Це означає що є більш адаптивним від SQL.

На сьогоднішній час у світі появились безліч NoSQL мов програмування такі як:

- MongoDB
- CouchDB
- OrientDB
- HBase
- Cassandra
- Neo4j
- Memcached
- FlockDB
- TimescaleDB
- CrateDB
- InfluxDB
- VoltDB

2.3 MySQL

Рисунок 2.3 – Логотип MySQL

MySQL розробили у 1984 році шведська компанія «TcX». MySQL це мова рамування для роботи реляйціними базами даних. Реляйціні бази даних вають, де таблиці пов'язані між собою. В 2010 році MySQL став належати иканській компанії «Oracle». MySQL є легким для вивчення і тому викладають нову в коледжах та університетах. MySQL працює за принципом сервер-клієнт, ділиться на 2 частини. Це MySQL Server та MySQL client. MySQL Server

находиться в комп'ютері, який виконує роль сервера, тобто зберігає базу даних та також має можливість обробляти запити що надсилає клієнт. MySQL Server ще має виконувати функцію захист даних. MySQL client може бути будь-яка програма, яка має можливість надсилати запити до MySQL Server та отримує відповідь від сервера [8]. На сьогоднішній час MySQL є топ 1 за популярністю. MySQL має безліч інтерфейси:

- MySQL WorkBench
- phpMyAdmin
- MySQL Shell
- SequelPro
- DBVisualizer

2.4 React

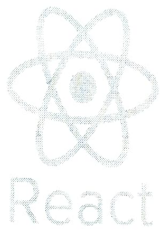


Рисунок 2.4 – Логотип React

React – це фреймворк для Javascript. Джордан Уолк створив її, коли він працював на посаді інженер-програміст в компанії «Facebook». Про неї дізналися в 2011 році, а в 2012 році компанія «Facebook» купила цю технологію. В 2013 році на конференції «JSConf 2013» React було представлено представлена як система з відкритим кодом і приєдналася до великої категорії UI-бібліотек, таких як jQuery, Angular, Dojo, Meteor та інші. Це означає, що компоненти React та React виступали як рівень подання або користувацького інтерфейсу для додатків JavaScript. В 2015 році компанія «Netflix» заявила, що також використовує React в своїх проектах для розробки користувацького інтерфейсу. З того часу у React появилось багато інструментів, таких як Redux, React Router, Mobx та інші. В 2019 році, розробили Хуки(Hooks) - новий спосіб

додавання і спільної використання логіки з відстеженням стану міжкомпонентами. На сьогоднішній день React розробка посідає топ 2 після Vue.js [2].

2.5 Tailwind CSS

Рисунок 2.5 – Логотип Tailwind CSS

Tailwind CSS – це фреймворк CSS, який надає набір готових класів для розробки користувацької інтерфейсу. Суть полягає в тому, щоб прописувати стилі в напряму в тегу class, а не окремому CSS-файлі. В 2017 році Адам Уетен опублікував цей фреймворк на платформі Github, а в 2019 році Уетен заснував власну компанію «Tailwind Labs» для підтримки розробки Tailwind CSS. Кожного року програмісти все більше обирають використовувати Tailwind CSS, а ніж Boostrap, тому що Tailwind CSS легше налаштувати і він може генерувати більш чистий CSS-код і це означає що можливо буде покращення в продуктивності [7].

2.6 MUI

Рисунок 2.6 – Логотип MUI

MUI(Material UI) – це бібліотека React-компонентів. Заснована вона в 2014 році. З 2016 року MUI різко починає набувати популярності все через своїй простоті використання компонентів. Її починають використовувати такі гіганти як: «Netflix», «Amazon», «Nasa», «Unity», «Shutterstock» так і маленькі компанії. Також MUI має підтримку мови програмування Typescript і це робить більш зручною для Typescript-програмістів. В 2024 році MUI продовжує розвиватися та використовувати тисячами розробників у всьому світі [6].

2.7 Node.js

Рисунок 2.7 – Логотип Node.js

Node.js – програмне забезпечення, яке дозволяє запускати програми, які написані на мові програмування Javascript. Node.js є open source і це означає він абсолютно безкоштовним. В 2009 році Райан Даль розробив Node.js, щоб програмісти які пишуть на мові програмування Javascript мали можливість розробляти серверну частину. Спочатку Node.js підтримував лише з операційними системами Mac OS та Linux тоді ще керував лише Даль, а згодом компанія «Joyent» почала фінансувати в цей проект. Node.js застосовується в двох режимах :

- виконання в стилі інтерфейсу командного рядка
- виконання «довгограючих» сценаріїв, таких як вебсервери.

Node.js використовує однопоточне середовище для виконання з подіями і це робить його ідеальним для створення мережевих застосунків, які потребують Високої продуктивності та масштабованості.

2.8 Spring Framework

Рисунок 2.8 – Логотип Spring Framework

Spring – це безкоштовний фреймворк для мови програмування Java. Spring ився на початку 2000-х років і з того часу набрав популярності і швидко вершив свого конкурента EJB за простотою моделі програмування,

різноманітністю, функцій та інтегрованих сторонніх бібліотек. Фреймворк Spring можна розбити на такі модулі:

- Core
- Web
- Data Access
- Boot
- Cloud
- Batch
- Test
- Security
- Messaging

Spring boot – це один найбільш популярний модуль у Spring Framework, він надає такі переваги як автоконфігурація це функція, яка надає скоротити дій, які повинні були робити розробники, тобто він автоматично може налаштовувати саму програму Spring на основі раніше доданих залежностей. Також Spring boot має можливість самостійно визначати набір налаштованих bean-компонентів за замовчування, які можна перевизначити за необхідності [1]. Spring Security – це також одна з найбільш цікавий модуль у Spring Framework, його функція полягає в тому щоб шифрувати та дешифрувати дані тобто виконує функції автетифікацію, авторизацію та захист від кібер-атак [14].

2.9 IntelliJ IDEA

Рисунок 2.9 – Логотип IntelliJ IDEA

IntelliJ IDEA – це програмне середовище для написання різних проектів на мові Java та Kotlin. IntelliJ IDEA не тільки включає Java, а майже всі технології які пов'язені з java розробки наприклад:

- HTML
- CSS
- Javascript
- Typescript
- Docker
- XML

На ній можна розробляти не тільки консольні програми, а вебзастосунки, декстопні програми, мобільні додатки. IntelliJ IDEA був заснованим чеська компанія «JetBrains» в 2001 році в Празі. Розробники сильно хвалять середовище за її зручність та легко находить помилки в коді тобто баги. IntelliJ IDEA є 2 весрії це платна IntelliJ IDEA Ultimate та безкоштовна версія. IntelliJ IDEA Community Edition. Компанія «JetBrains» надає безкоштовну ліцензії IntelliJ IDEA Ultimate для студентів та викладачам. В 2024 році середовище IntelliJ IDEA є топ 1 середовище серед середовищ для розробки для мови програмування Java та Kotlin.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 ПРОЦЕС РОЗРОБКИ

Під час розробки вебзастосунку «Бакалійний дім» нам потрібно інсталювати такі середовище:

- Node.js
- MySQL
- IntelliJ IDEA

Спочатку ми заходим на офіційним сайт node.js та клікаєм на кнопку «Download Node.js»

Рисунок 3.1 – Сайт node.js

Натискаєм на кнопку Next.

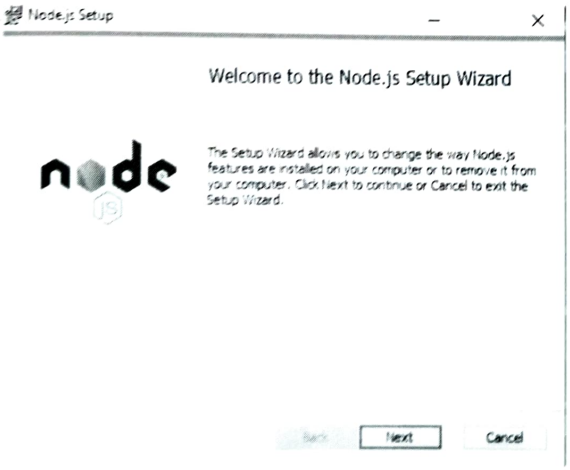


Рисунок 3.2 – Інсталяція node.js

Далі ми вибираєм де ми будем інсталювати node.js, щоб не виникали ніяких помилок під час розробки, розробники рекомендують встановлювати в диск C.

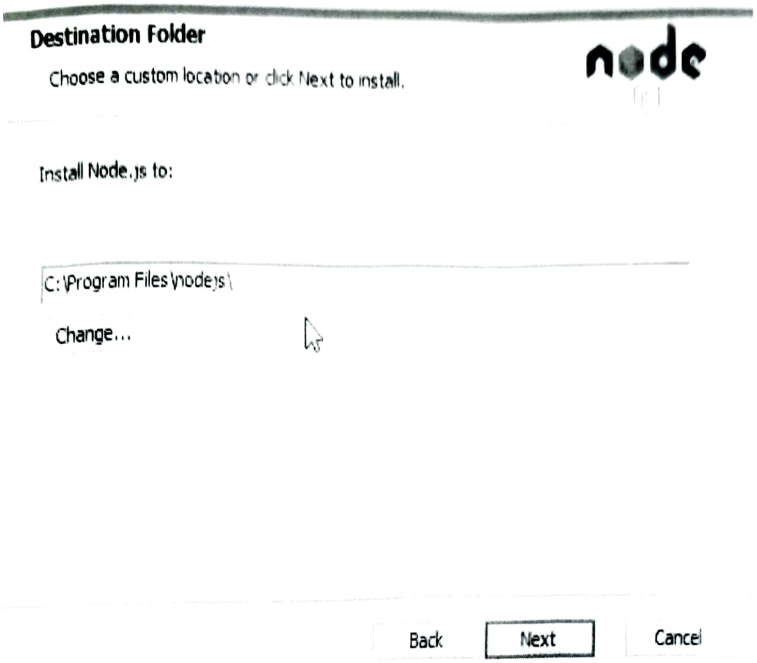


Рисунок 3.3 – Інсталяція node.js

Коли ми успішно встановили node.js, ми можем перейти до інсталяції MySQL.

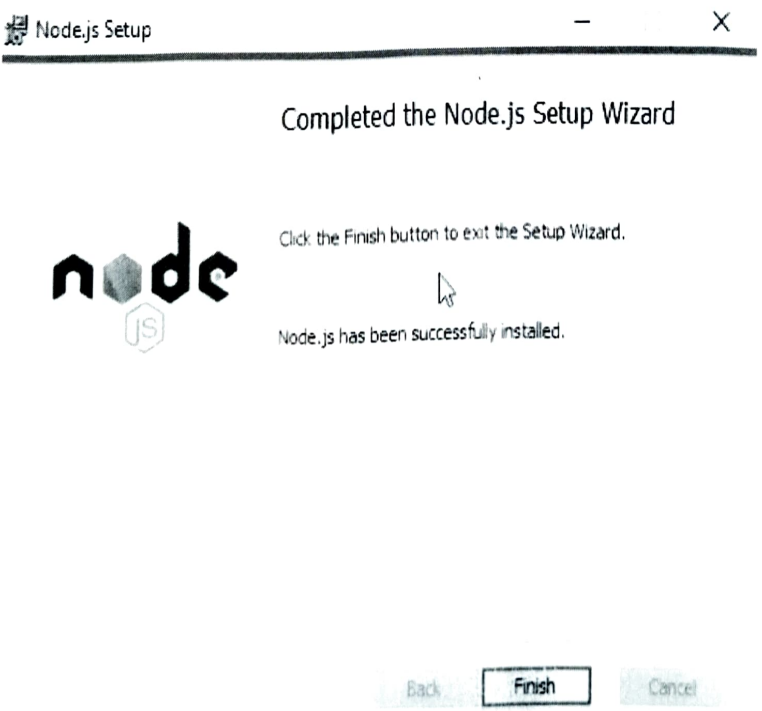


Рисунок 3.4 – Інсталяція завершено

Щоб інсталювати MySQL ми заходим на офіційний сайт MySQL.

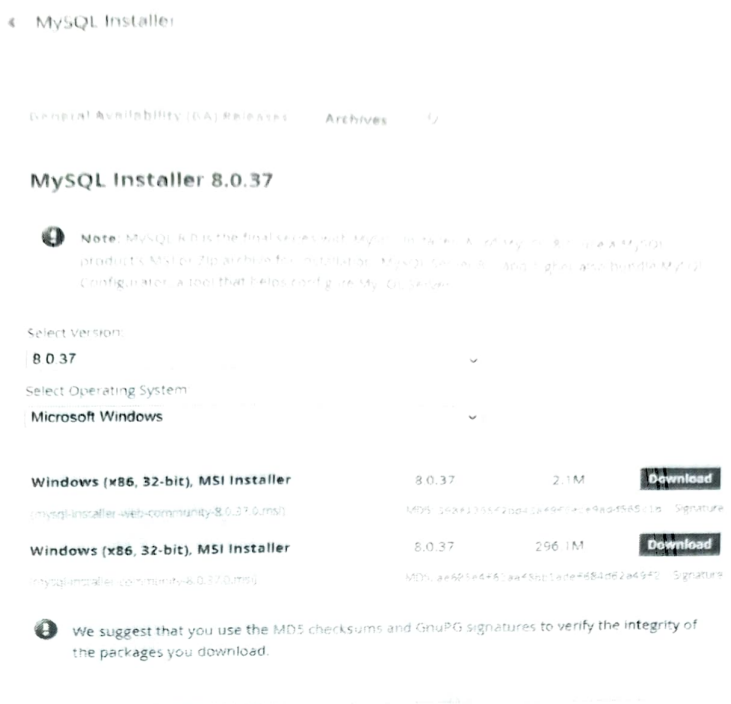


Рисунок 3.5 – Сайт MySQL

Нам перед інсталяції пропонують які частини встановлювати клієнтську чи серверну, але нам потрібно всі частини встановлювати і ставим маркер на Full

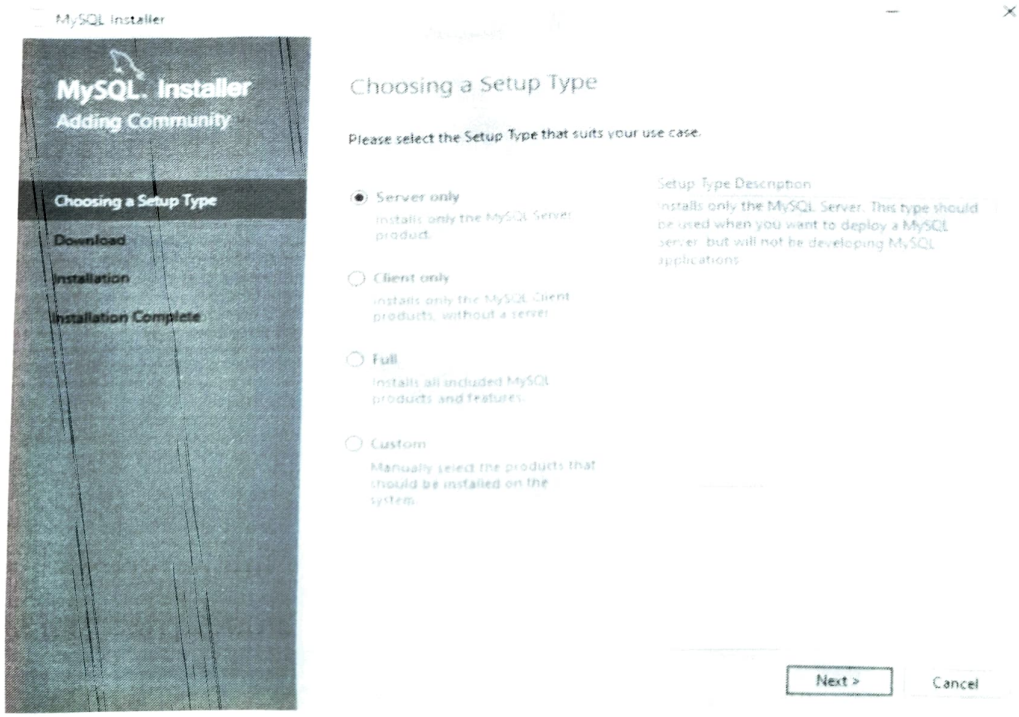


Рисунок 3.6 - Інсталяція MySQL

Коли ми встановили усі компоненти MySQL Нам слід авторизуватись

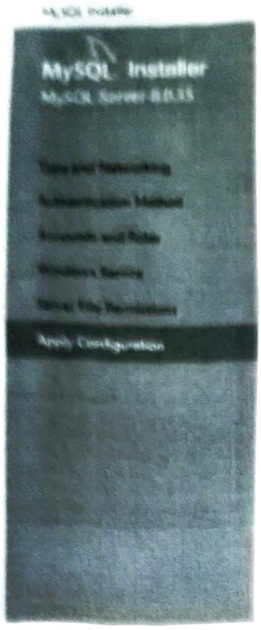


Рисунок 3.7 - Інсталяція MySQL

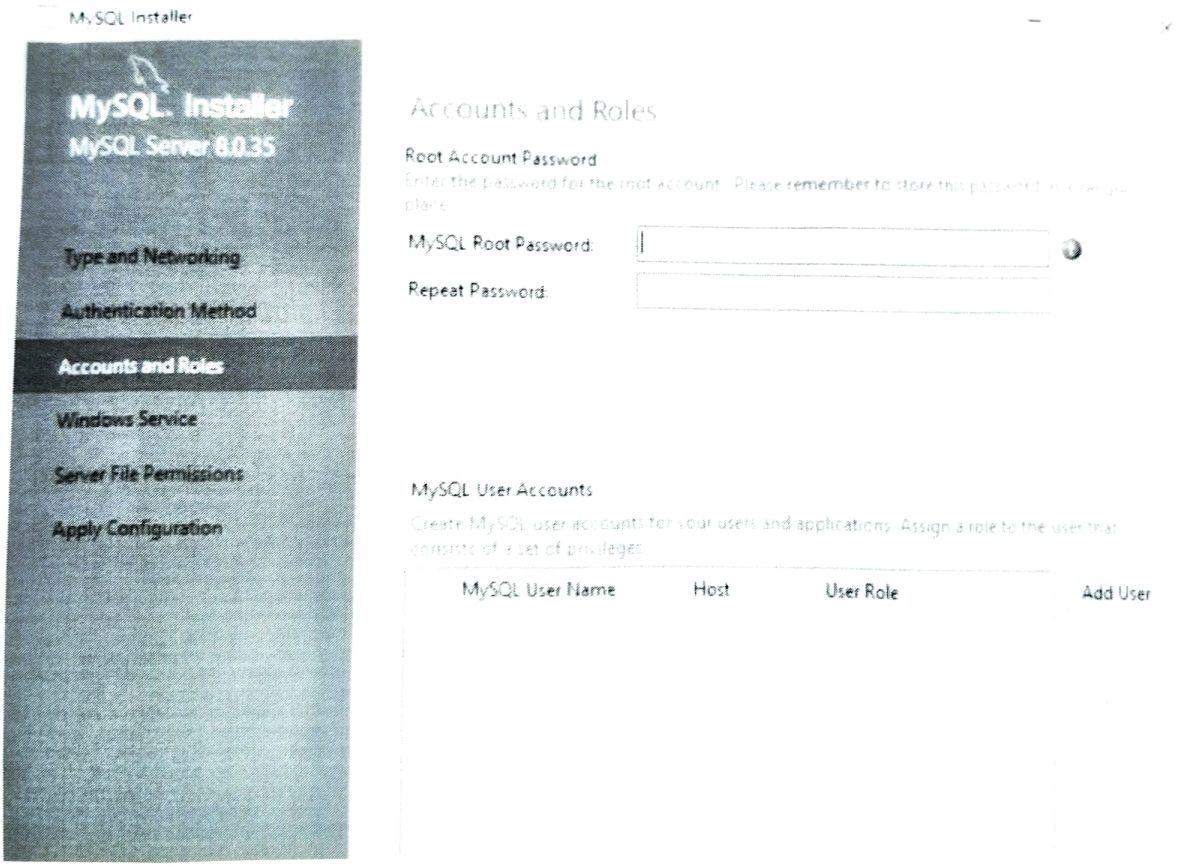


Рисунок 3.8 – Створюємо профіль

Коли ми встановили MySQL ми переходим до встановлення IntelliJ IDEA. Заходим на офіційний сайт Jetbrains і завантажуєм IntelliJ IDEA.

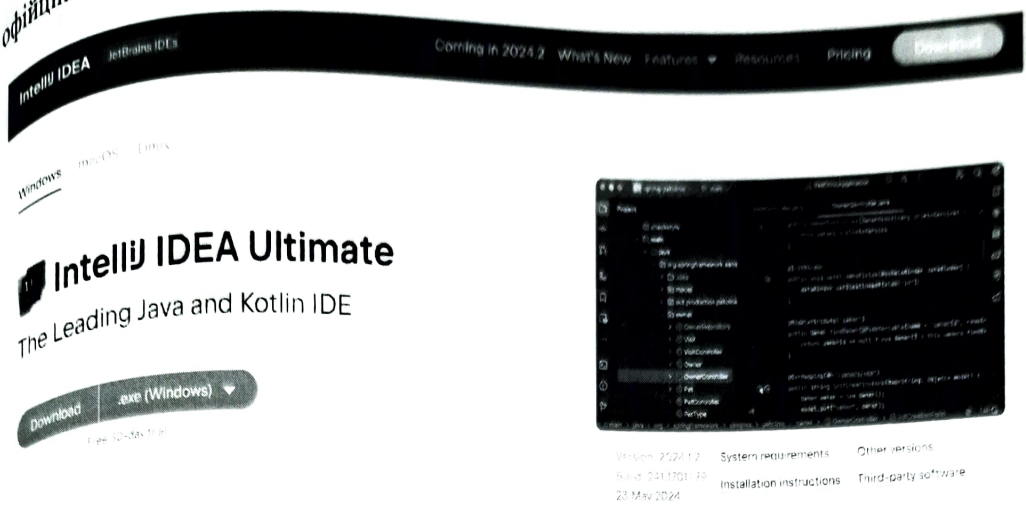


Рисунок 3.9 – Сайт JetBrains

Натискаєм на кнопку Next.

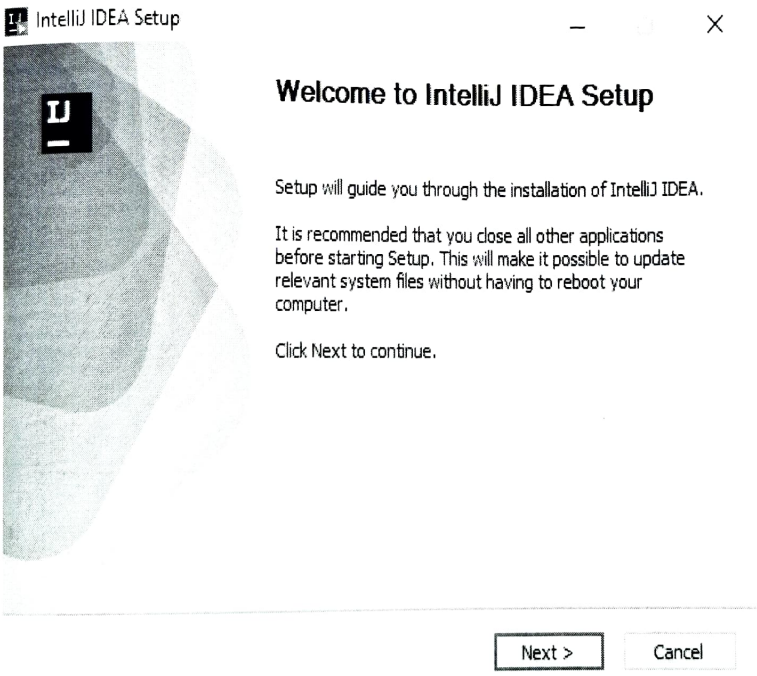


Рисунок 3.10 - Інсталяція IntelliJ IDEA

Ставимо маркери як зображено на рисунку 3.11.

Рисунок 3.11- Інсталяція IntelliJ IDEA

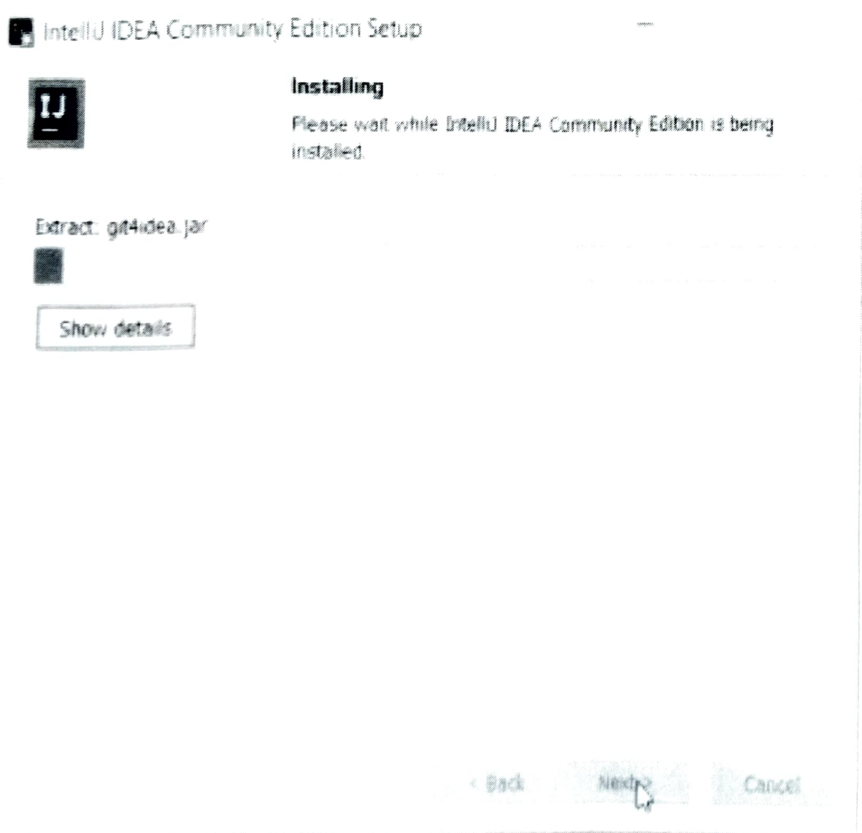


Рисунок 3.12 - Інсталяція IntelliJ IDEA

Рисунок 3.13 - Інсталяція IntelliJ IDEA

Коли ми встановили всі програми ми можем приступати до розробки програми, а саме Back-end частини. На сайті Spring initialzr ми можем ініціалізувати наш Back-end проект.

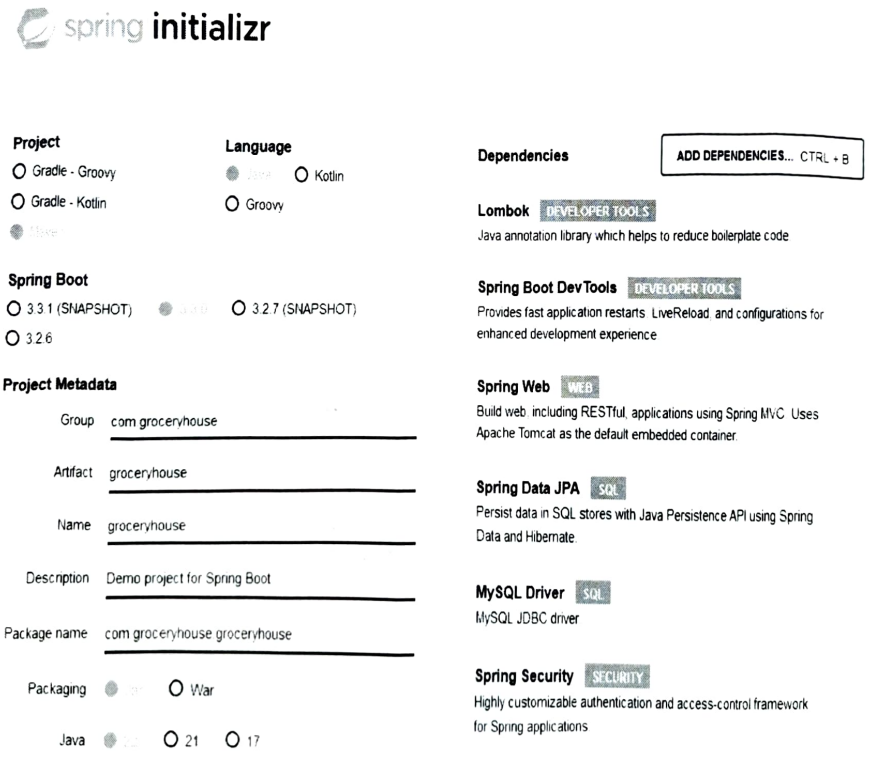


Рисунок 3.14 – Spring initializr

Коли ми відкрили цей проект нам необхідно створити 8 директорій такі як Config, dto, service, request, model, controller.

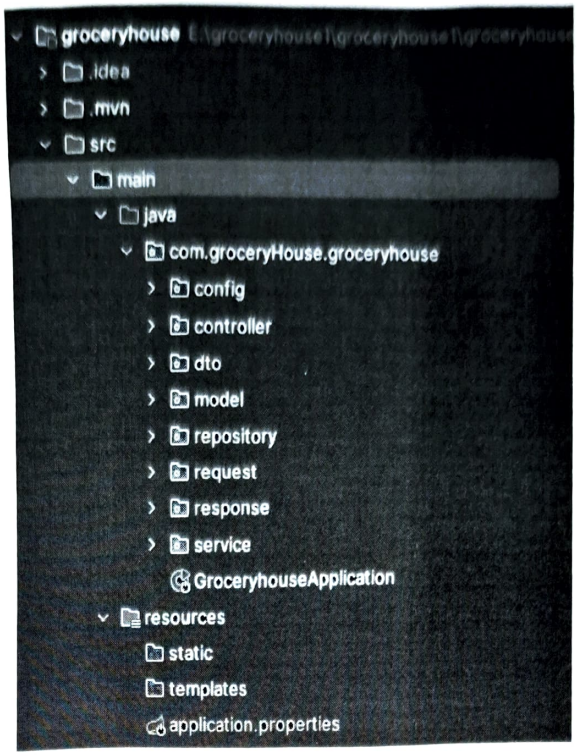


Рисунок 3.15 – Створили 8 директорій

В файлі application.properties ми налаштовуєм зв'язок базою даних з сервером, а також підключим JPA.

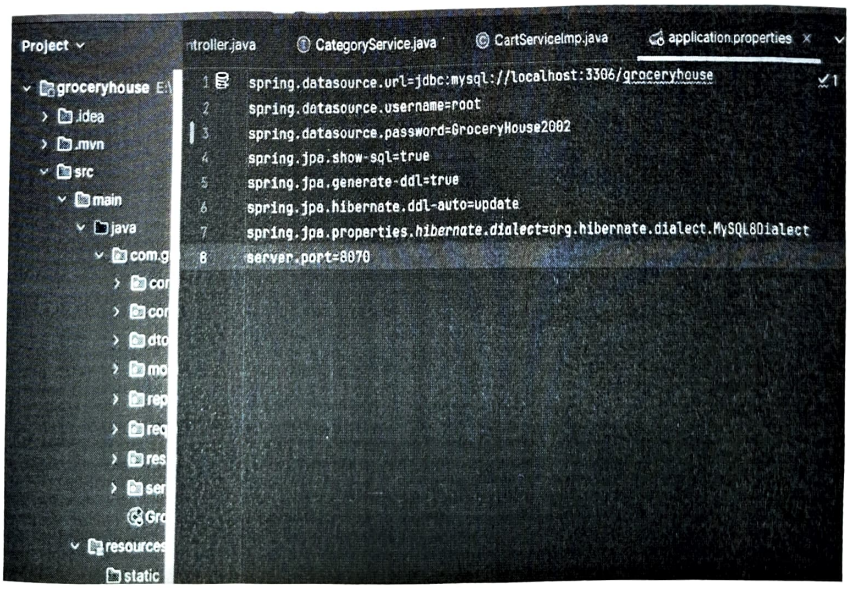


Рисунок 3.16 – Application.properties

Нам слід створити API User, Product, Food, Home, Category, Ingredient, Order Почнемо з User API. Спочатку у папці model створимо декілька класи і класах додаєм атрибути, які ми хочем добавити в таблицях.

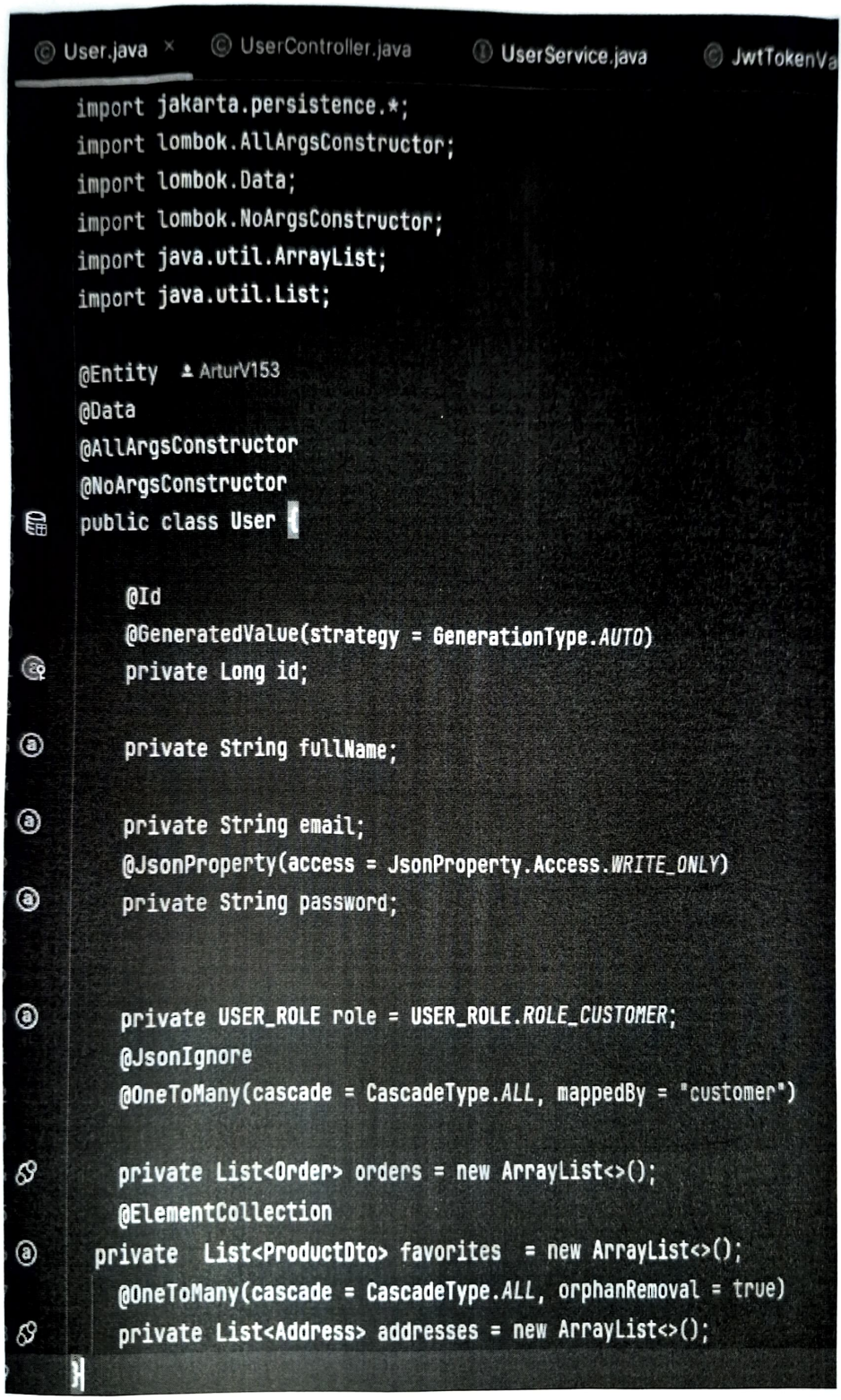


Рисунок 3.17 - User.java

```java
package com.groceryHouse.groceryhouse.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.ArrayList;
import java.util.List;

@Data  ▲ ArturV153 *
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @OneToOne
    private User owner;

    private String name;

    private String description;

    private String cuisineType;
    @OneToOne
    private Address address;


    @OneToMany(mappedBy = "product",cascade = CascadeType.ALL,orphanRemoval = true)
    private List<Order> orders= new ArrayList<>();
    @ElementCollection
    @Column(length = 1000)
    private List<String> images;
```

Рисунок 3.18 - Product.java

```java
@NoArgsConstructor
@Data
@Entity
public class Food {

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

    private String name;

    private String description;

    private Long price;
    @ManyToOne
    private Category foodCategory;

    @Column(length = 1000)
    @ElementCollection
    private List<String> images;

    private boolean available;

    @ManyToOne
    private Product product;

    private boolean isSugarFree;
    private boolean isSeasonal;
    @ManyToMany
    private List<IngredientsItem> Ingredients = new ArrayList<>();

    private Date creationDate;
```

Рисунок 3.19 - Food.java

Після того як створили усі моделі ми можем приступати до написання репозиторій.

```java
package com.groceryHouse.groceryhouse.repository;

import com.groceryHouse.groceryhouse.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,Long> {

    public User findByEmail(String username);

}
```

Рисунок 3.20 - UserRepository.java

```java
package com.groceryHouse.groceryhouse.repository;

import com.groceryHouse.groceryhouse.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;


import java.util.List;


public interface ProductRepository extends JpaRepository<Product,Long> {

    @Query("SELECT r FROM Product r where lower(r.name) LIKE lower(concat('%',:query,'%') ) " +
            "OR lower(r.cuisineType)" +
            "Like lower(concat('%',:query,'%'))")
    List<Product> findByQuery(String query);

    Product findByOwnerId(Long userId);

}
```

Рисунок 3.21 - ProductRepository.java

```java
package com.groceryHouse.groceryhouse.repository;

import com.groceryHouse.groceryhouse.model.Food;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
@Repository
public interface FoodRepository extends JpaRepository<Food,Long> {

    List<Food> findByProductId(Long productId);

    @Query("SELECT f from Food f  WHERE  f.name LIKE %:keyword% OR f.foodCategory.name LIKE %:keyword%")
    List<Food>searchFood(@Param("keyword") String keyword );

}
```

Рисунок 3.22 - FoodRepository.java

Після того як створили усі репозиторії, треба нам налаштувати Spring Security.В папці config необхідно створити файл AppConfig @Configuration – це значить, що цей клас для утворення або налаштування бінів.@EnableWebSecurity - Додає підтримку веб безпеки до вебдодатку Spring Метод securityFilterChain(HttpSecurity http) налаштовує безпеку для HTTP запитів .addFilterBefore(new JwtTokenValidator(), BasicAuthenticationFilter.class) Додає фільтр для валідації JWT токенів перед виконанням базової аутентифікації.

```
public class AppConfig {

    @Bean  ArturV153
    SecurityFilterChain securityFilterChain(HttpSecurity http)throws Exception{

        http.sessionManagement(managment ->managment.sessionCreationPolicy(SessionCreationPolicy.S
            .authorizeHttpRequests(Authorize -> Authorize
                .requestMatchers("/api/admin/**").hasAnyRole( ...roles: "Admin")
                .requestMatchers("/api/**").authenticated()
                .anyRequest().permitAll()
        ).addFilterBefore(new JwtTokenValidator(), BasicAuthenticationFilter.class)
                .csrf(csrf-> csrf.disable())
                .cors(cors-> cors.configurationSource(corsconfigurationSource()));
        return http.build();
    }

    private CorsConfigurationSource corsconfigurationSource() { 1 usage  ArturV153
    return new CorsConfigurationSource() {  ArturV153
        @Override  no usages  ArturV153
        public CorsConfiguration getCorsConfiguration(HttpServletRequest request) {
            CorsConfiguration cfg = new CorsConfiguration();
            cfg.setAllowedOrigins(Arrays.asList(
                "https://bakaliynyy-dim",
                    "http://localhost:3000/"
            ));
            cfg.setAllowedMethods(Collections.singletonList("*") );
            cfg.setAllowCredentials(true);
            cfg.setAllowedHeaders(Collections.singletonList("*"));
            cfg.setExposedHeaders(Arrays.asList("Authorization"));
            cfg.setMaxAge(3600L);

            return cfg;

        }
    };
    }
@Bean  ArturV153
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

}
```

Рисунок 3.23 - AppConfig.java

ля того як налаштували Spring Security нам потрібно перевірити чи все
ажд працює.

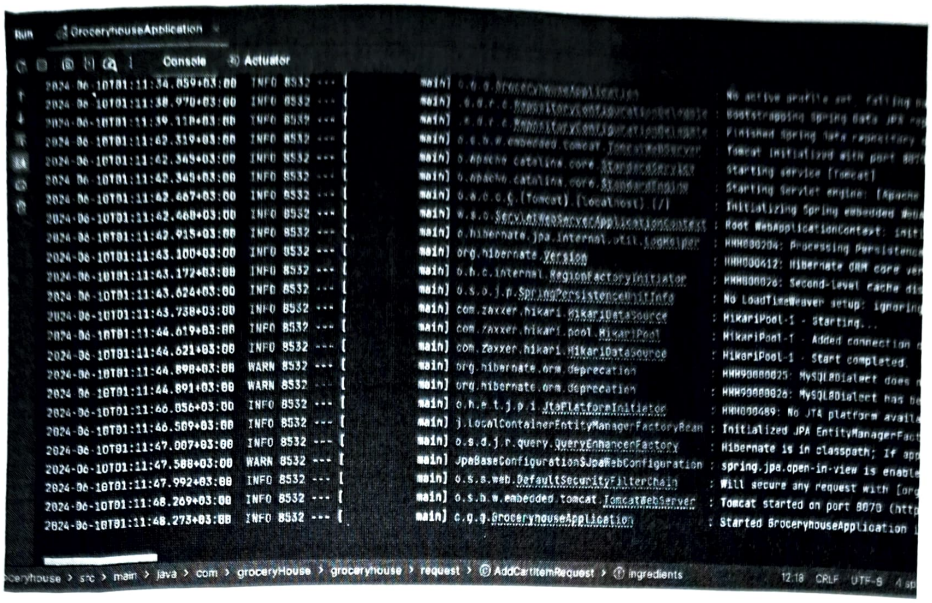
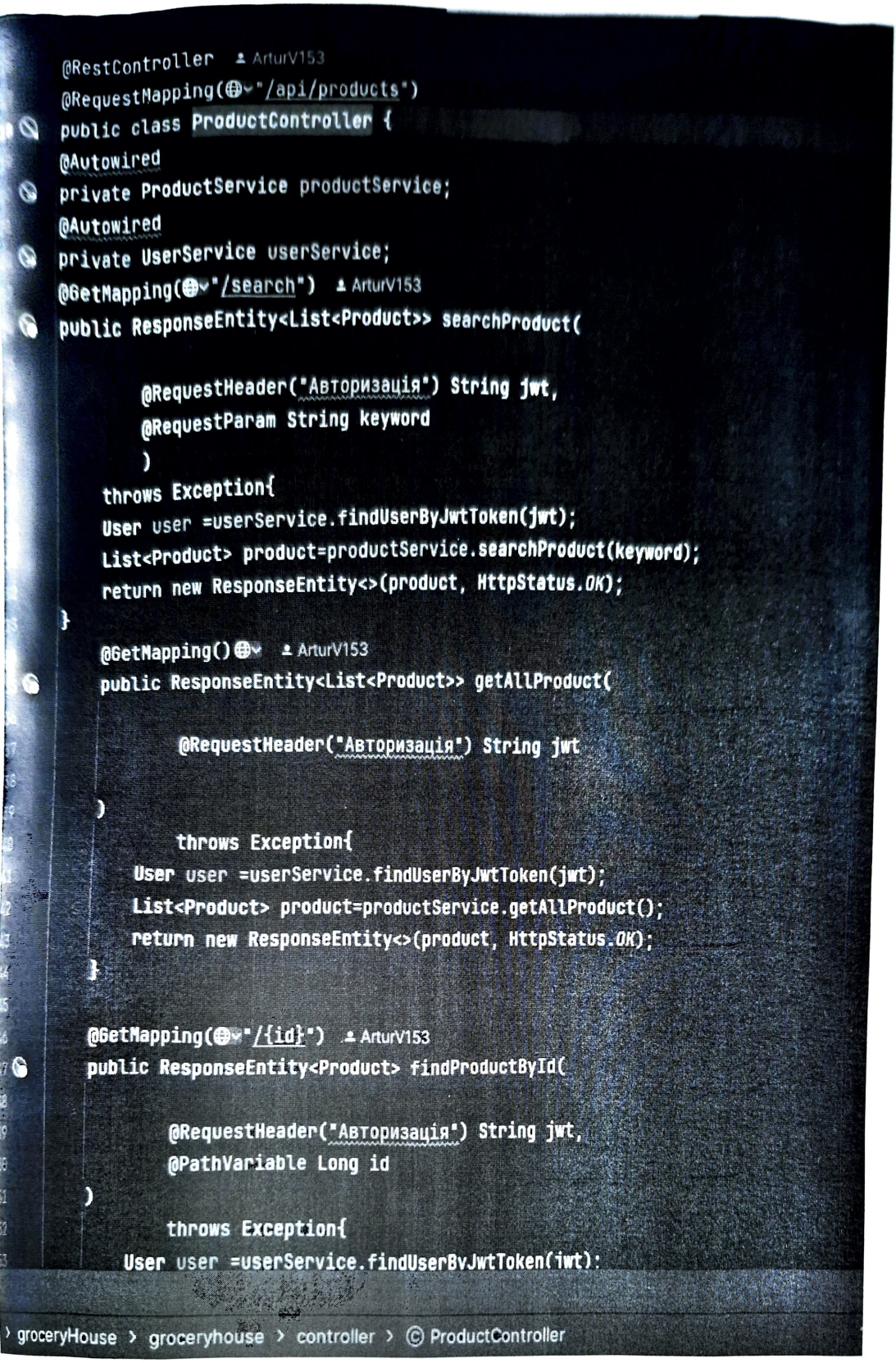


Рисунок 3.24 – Запуск Back-end

Тоді почнемо створювати контролери, щоб клієнт міг відправляти свої
апити.

```
package com.groceryHouse.groceryhouse.controller;

import com.groceryHouse.groceryhouse.model.User;
import com.groceryHouse.groceryhouse.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController  ♟ ArturV153 *

@RequestMapping("/api/users")
public class UserController {
    @Autowired
    private UserService userService;

@GetMapping("/profile")  ♟ ArturV153
    public ResponseEntity<User> findByJwtToken(@RequestHeader("авторизація") String jwt) throws Exception {

    User user=userService.findUserByJwtToken(jwt);
    return new ResponseEntity<>(user, HttpStatus.OK);

    }
```

Рисунок 3.25 - UserController.java



```
@RestController  ArturV153
@RequestMapping("/api/products")
public class ProductController {
@Autowired
private ProductService productService;
@Autowired
private UserService userService;
@GetMapping("/search")  ArturV153
public ResponseEntity<List<Product>> searchProduct(

        @RequestHeader("Авторизація") String jwt,
        @RequestParam String keyword
        )
    throws Exception{
    User user =userService.findUserByJwtToken(jwt);
    List<Product> product=productService.searchProduct(keyword);
    return new ResponseEntity<>(product, HttpStatus.OK);
}
    @GetMapping()  ArturV153
    public ResponseEntity<List<Product>> getAllProduct(

            @RequestHeader("Авторизація") String jwt

    )
            throws Exception{
        User user =userService.findUserByJwtToken(jwt);
        List<Product> product=productService.getAllProduct();
        return new ResponseEntity<>(product, HttpStatus.OK);
    }

    @GetMapping("/{id}")  ArturV153
    public ResponseEntity<Product> findProductById(

            @RequestHeader("Авторизація") String jwt,
            @PathVariable Long id
    )
            throws Exception{
        User user =userService.findUserByJwtToken(jwt);

› groceryHouse › groceryhouse › controller › © ProductController
```

Рисунок 3.26 - ProductController.java

```java
@RestController  ArturV153
@RequestMapping("/api/food")
public class FoodController {
    @Autowired
    private FoodService foodService;
    @Autowired
    private UserService userService;
    @Autowired
    private ProductService productService;
    @GetMapping ("/search")  ArturV153
    public ResponseEntity<List<Food>> searchFood(@RequestParam String name,
                                                 @RequestHeader("Авторизація")String jwt) throws Exception {

        User user = userService.findUserByJwtToken(jwt);

        List<Food> foods = foodService.searchFood(name);


        return new ResponseEntity<>(foods, HttpStatus.CREATED);
    }
    @GetMapping ("/product/{productId}")  ArturV153
    public ResponseEntity<List<Food>> getProductFood(@RequestParam boolean sugarFree,
                                                     @RequestParam boolean seasonal,
                                                     @RequestParam boolean NoSugarFree,
                                                     @RequestParam(required = false) String food_category,
                                                     @PathVariable Long productId,
                                                     @RequestHeader("Авторизація")String jwt) throws Exception {

        User user = userService.findUserByJwtToken(jwt);

        List<Food> foods = foodService.getProductsFood(productId,sugarFree,seasonal,NoSugarFree,food_category);


        return new ResponseEntity<>(foods, HttpStatus.OK);
    }

}
```

Рисунок 3.27 - FoodController.java

Після створень контролерів нам слід перевірити на Postman чи програма обробляє апити.

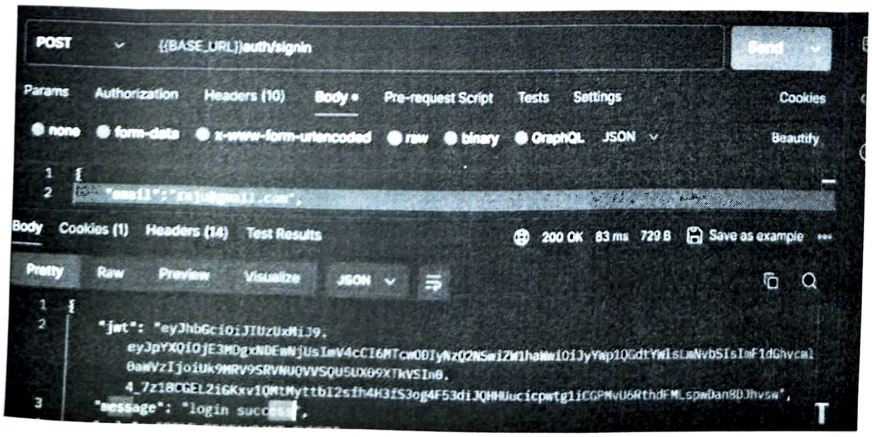


Рисунок 3.28 – Перевірка на авторизацію

Тоді як перевірили чи працюють усі запити ми може приступати до Front-end частини. Створюємо окрему папку та в терміналі вводимо цю команду для того, щоб створити react застосунок.

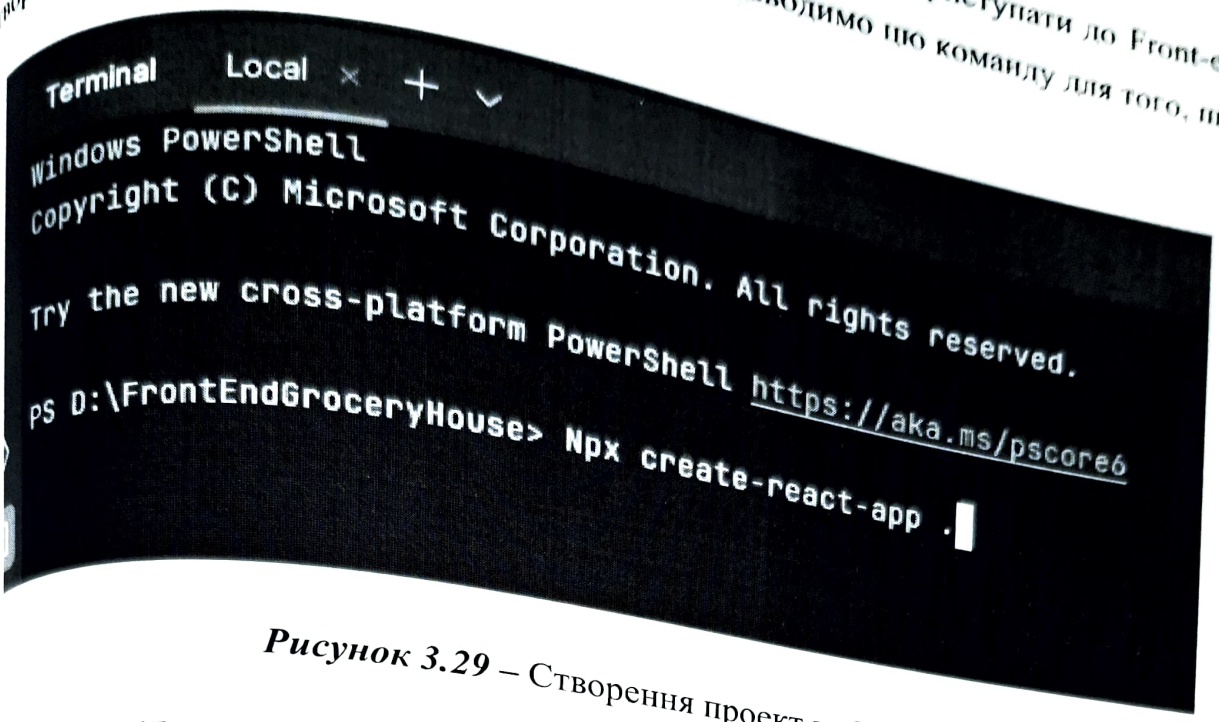


Рисунок 3.29 – Створення проект на React

Далі нам потрібно написати команду, щоб завантажити tailwind css фреймворк.

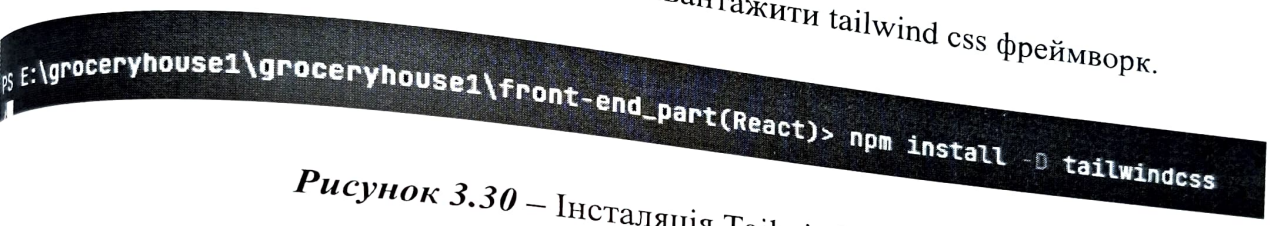


Рисунок 3.30 – Інсталяція Tailwind CSS

Наступним кроком нам слід завантажити formik бібліотеку.

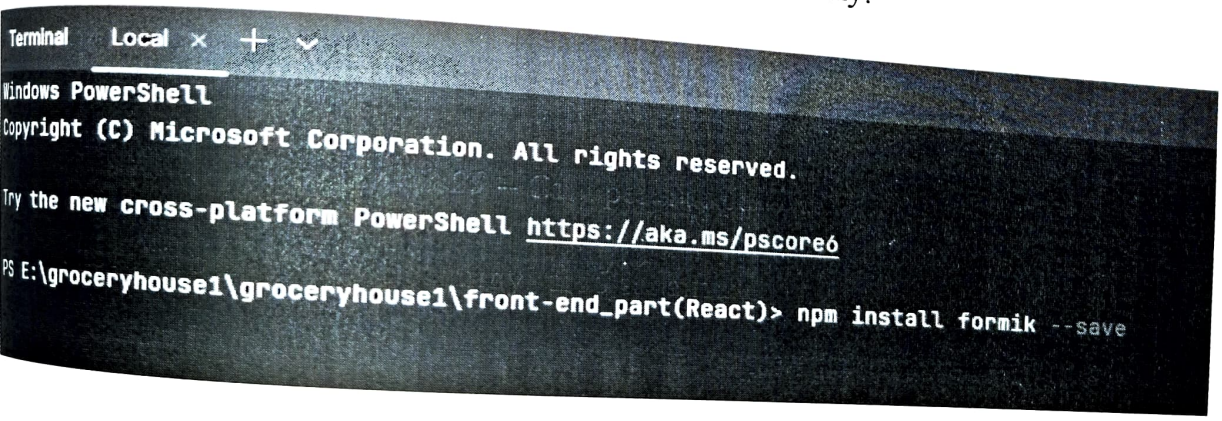


Рисунок 3.31 - Інсталяція Formik

Formik бібліотека дає можливість створювати форми. Наші customer створюємо 4 папки: theme, components, pages, routers. Наші customer ми створимо файл Navbar.jsx.

```jsx
import React from 'react';
import './Navbar.css';
import SearchIcon from "@mui/icons-material/Search";
import {Avatar, Badge, IconButton} from "@mui/material";
import ShoppingCartIcon from "@mui/icons-material/ShoppingCart";
import {green} from "@mui/material/colors";
import {useNavigate} from "react-router-dom";


const Navbar1 = () => {  Show usages
    const navigate : NavigateFunction =useNavigate()
    return (
        <div className='px-5 z-50 py-[.8rem] bg-[#216108] lg:px-20 flex justify-between'>
            <div className='lg:mr-10 cursor-pointer flex items-center space-x-4' onClick={() : void =>navigate("/")}>
                <li className='logo font-semibold text-gray-300 text-2xl' >
                    Бакалійний дім
                </li>
            </div>
            <div className='flex items-center space-x-2 lg:space-x-10'>
                <div className=''>
                    <IconButton>
                        <SearchIcon sx={{fontSize: '1.5rem'}}/>
                    </IconButton>
                </div>
                <div className=''>
                    <Avatar sx={{bgcolor: "white", color: green.A400}}>C</Avatar>
                </div>
                <div className=''>
                    <IconButton  onClick={() : void =>navigate("/cart")}>
                        <Badge color='primary' badgeContent={3}>
                            <ShoppingCartIcon sx={{fontSize: '1.5rem'}}/>
                        </Badge>
                    </IconButton>
                </div>
            </div>
        </div>
    );
};
```

Рисунок 3.32 – Navbar.jsx

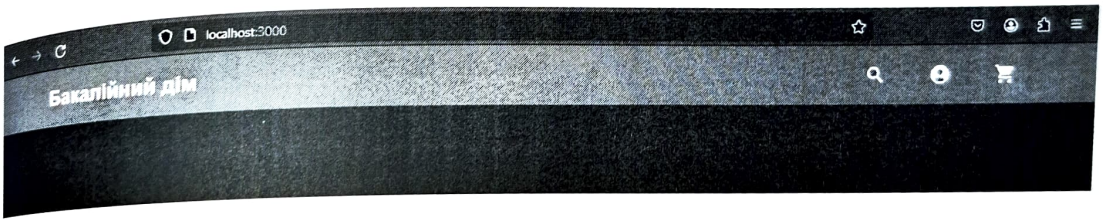
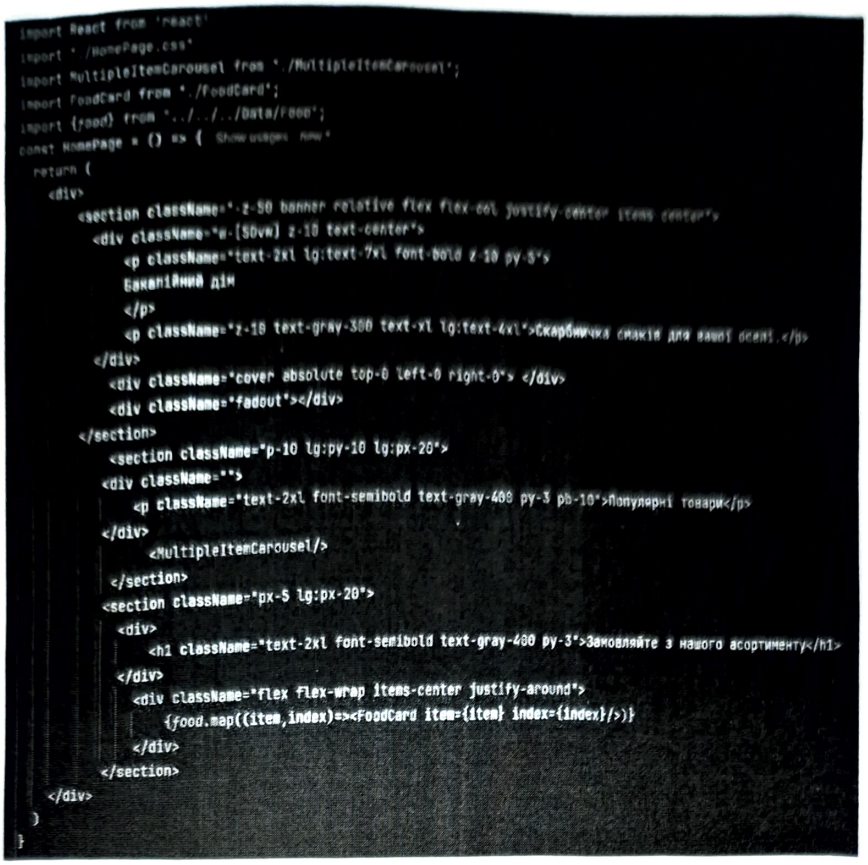


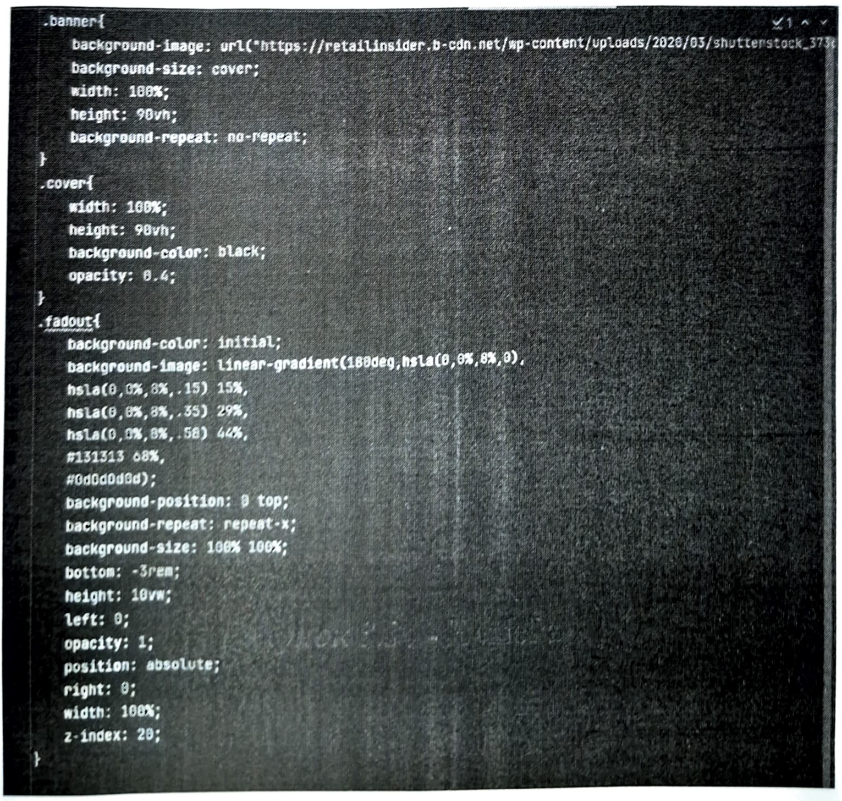
Рисунок 3.33 - Навігаційна панель

Після того ми створили віджет Navbar. Ми переходимо до розробки головної торінки Для головної сторінки головної сторінки потрібно розробити такі елементи «карусель» продуктів, що буде показувати популярний асортимент та картки продуктів.



```
import React from 'react'
import './HomePage.css'
import MultipleItemCarousel from './MultipleItemCarousel';
import FoodCard from './FoodCard';
import {food} from '../../../Data/Food';
const HomePage = () => {
  return (
    <div>
      <section className='z-50 banner relative flex flex-col justify-center items-center'>
        <div className="w-[50vw] z-10 text-center">
          <p className="text-2xl lg:text-7xl font-bold z-10 py-5">
          Бажаний дім
          </p>
          <p className="z-10 text-gray-300 text-xl lg:text-4xl">Скарбничка смаків для вашої осені.</p>
        </div>
        <div className="cover absolute top-0 left-0 right-0"> </div>
        <div className="fadout"></div>
      </section>
        <section className="p-10 lg:py-10 lg:px-20">
        <div classNames="">
          <p classNames="text-2xl font-semibold text-gray-400 py-3 pb-10">Популярні товари</p>
        </div>
            <MultipleItemCarousel/>
        </section>
        <section classNames="px-5 lg:px-20">
          <div>
            <h1 classNames="text-2xl font-semibold text-gray-400 py-3">Замовляйте з нашого асортименту</h1>
          </div>
          <div classNames="flex flex-wrap items-center justify-around">
            {food.map((item,index)=><FoodCard item={item} index={index}/>)}
          </div>
        </section>
    </div>
  )
}
```

Рисунок 3.34 - HomePage.jsx



```
.banner{
    background-image: url("https://retailinsider.b-cdn.net/wp-content/uploads/2020/03/shutterstock_373
    background-size: cover;
    width: 100%;
    height: 90vh;
    background-repeat: no-repeat;
}
.cover{
    width: 100%;
    height: 90vh;
    background-color: black;
    opacity: 0.4;
}
.fadout{
    background-color: initial;
    background-image: linear-gradient(180deg,hsla(0,0%,0%,0),
    hsla(0,0%,8%,.15) 15%,
    hsla(0,0%,8%,.35) 29%,
    hsla(0,0%,8%,.58) 44%,
    #131313 68%,
    #0d0d0d0d);
    background-position: 0 top;
    background-repeat: repeat-x;
    background-size: 100% 100%;
    bottom: -3rem;
    height: 10vw;
    left: 0;
    opacity: 1;
    position: absolute;
    right: 0;
    width: 100%;
    z-index: 20;
}
```

Рисунок 3.35 - HomePage.css

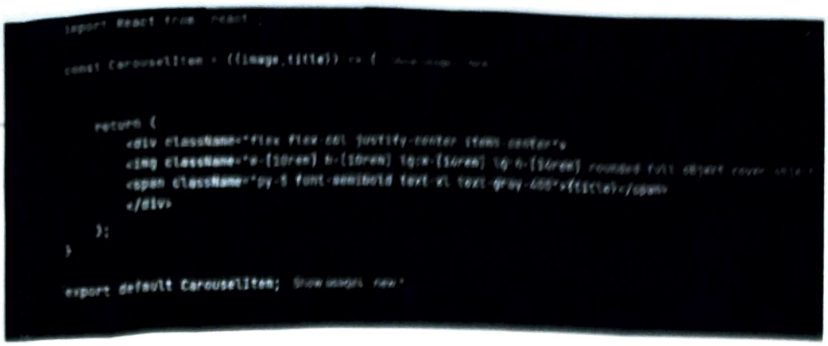


Рисунок 3.36 - CarouselItem.jsx

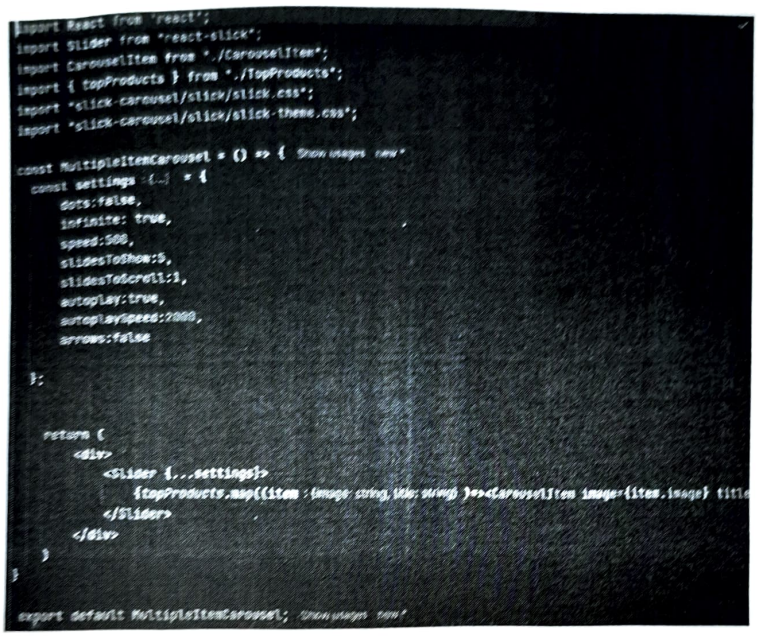


Рисунок 3.37 - MultipleItemCarousel.jsx

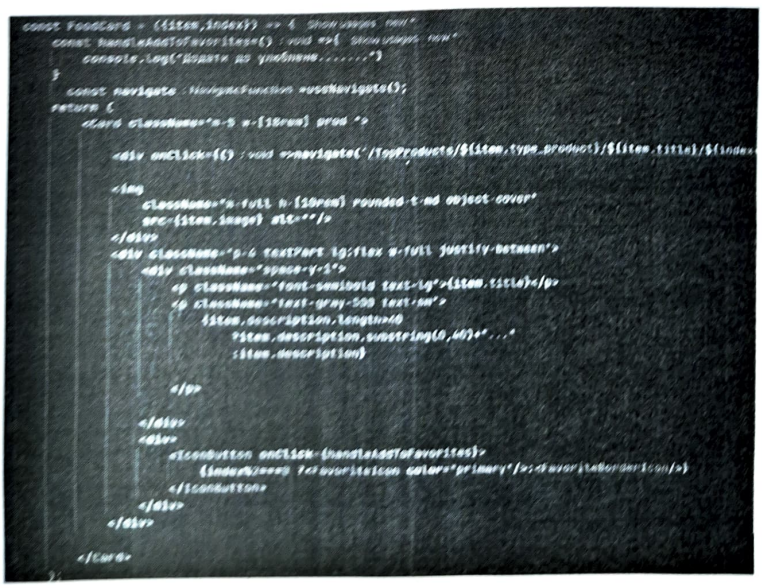


Рисунок 3.38 - FoodCart.jsx

3.2 Інтерфейс клієнтської частини

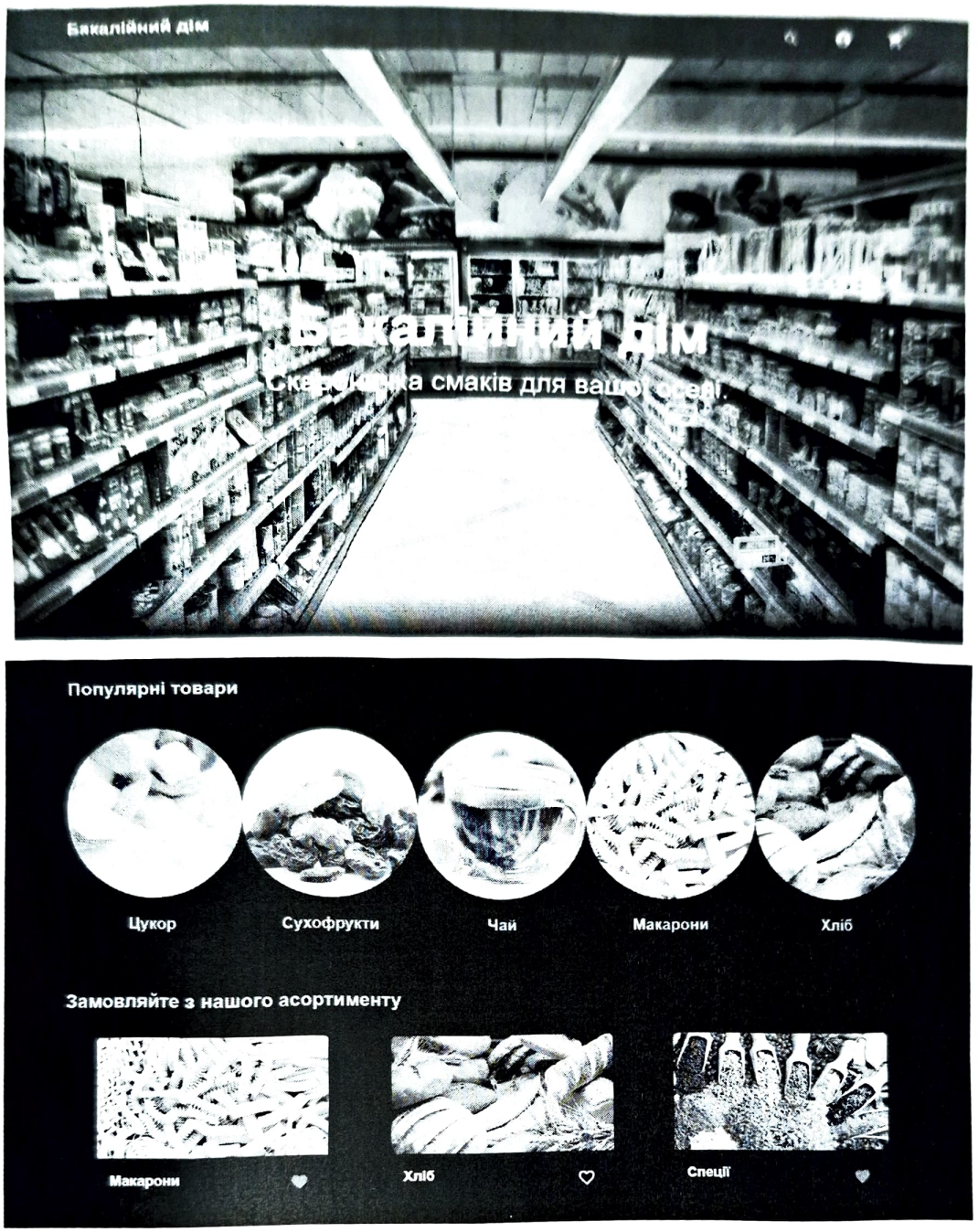


Рисунок 3.39 - Дизайн головної сторінки

Коли ми зробили всі сторінки ми можем приступати до файлу Routers. ей компонент буде відповідати за визначення маршуту для різних сторінок.

```jsx
import React from 'react';
import {Route, Routes} from "react-router-dom";

import CustomerRouters from "./CustomerRouters";

const Routers = () => {  Show usages  new *
    return (
        <Routes>
            <Route path="/*" element={<CustomerRouters/>} />
        </Routes>
    )
};

export default Routers;  Show usages  new *
```

Рисунок 3.40 - Routers.jsx

```jsx
import React from 'react';
import {Route, Routes} from "react-router-dom";
import HomePage from "../customer/pages/HomePage/HomePage";
import Product from "../customer/pages/Product/Product";
import Profile from "../customer/pages/Profile1/Profile";
import Navbar1 from "../customer/components/Navbar/Navbar1";
import Cart1 from "../customer/pages/Cart1/Cart1";


const CustomerRouters = () => {  Show usages  new *
    return (
        <div className="relative">
            <div className="sticky top-0 z-50">
                <Navbar1/>
            </div>
            <Routes>
                <Route path="/" element={<HomePage/>} />
                <Route path="/TopProducts/:type_product/:title/:id" element={<Product/>} />
                <Route path="/cart" element={<Cart1/>} />
                <Route path="/my-profile/*" element={<Profile/>} />
            </Routes>
        </div>
    );
};

export default CustomerRouters;  Show usages  new *
```

Рисунок 3.41 - CustomerRouters.jsx

Щоб перевірити чи сайт працює ми вводимо команду і перевіряєм чи сторінки працюють.

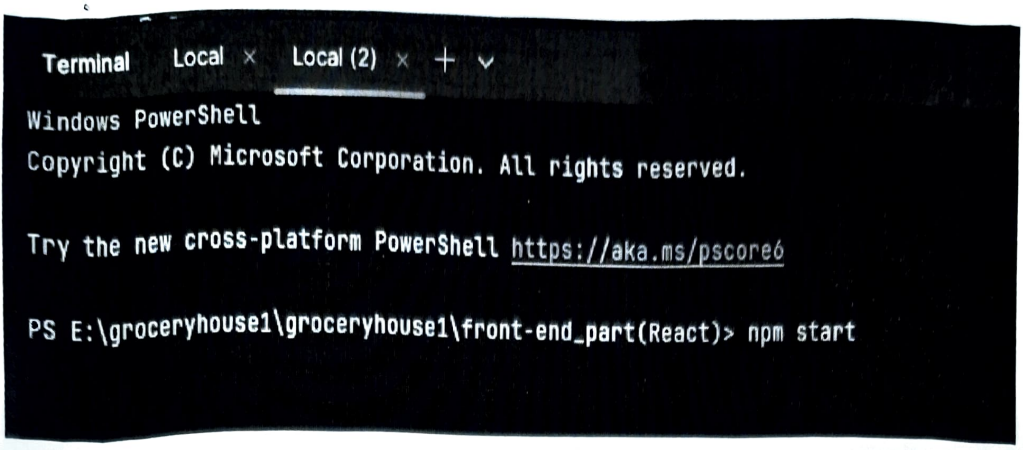


Рисунок 3.42 – Запуск застосунку

Інтерфейс сторінки «мій профіль».

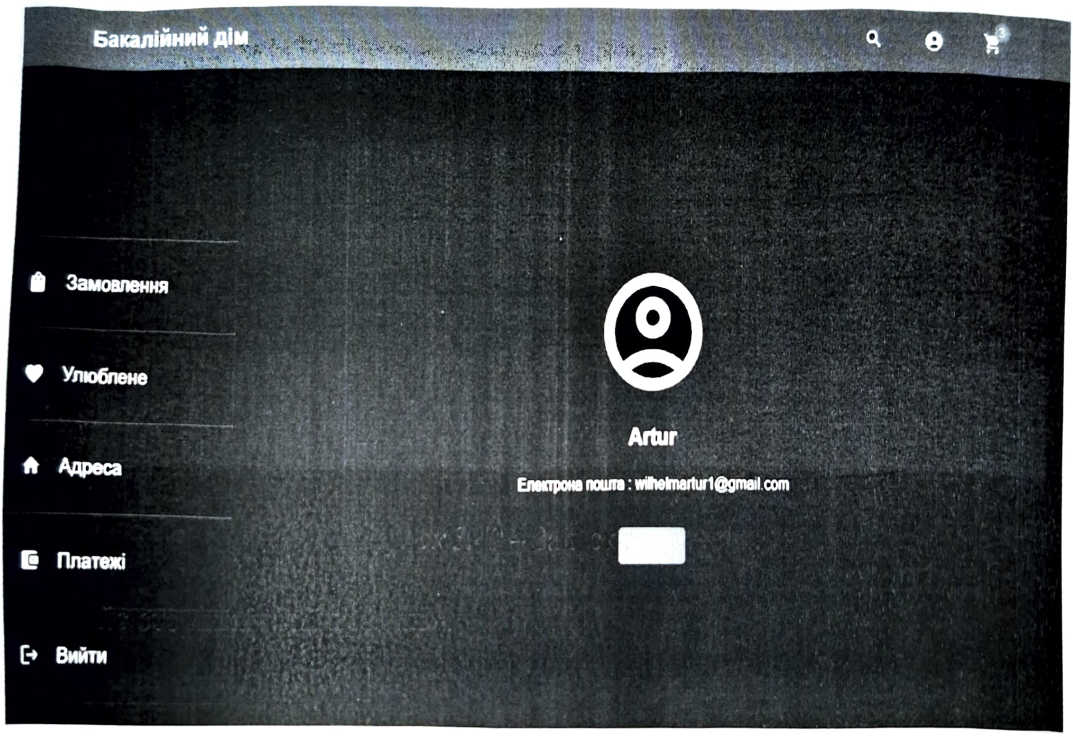


Рисунок 3.43 - Інтерфейс сторінки «Особистий кабінет»

ВИСНОВКИ

В дипломній роботі було розроблено вебзастосунок, який використовує технології: Spring boot, Spring security, MySQL, React. Вебзастосунок дозволяє клієнтам пройти реєстрацію або авторизуватись: користувачі можуть створити новий обліковий запис або увійти до існуючого за допомогою надійної системи авторизації, реалізованої за допомогою Spring Security, перегляд каталогу: товару усі товари представлені в зручному каталозі, де користувачі можуть ознайомитись з описом, цінами та іншими характеристиками, пошук товару: для полегшення навігації та пошуку потрібного товару реалізовано функціонал пошуку, додавання товару до кошику: користувачі можуть додавати вибрані товари до кошику для подальшого замовлення., перегляд до кошика, оформлення замовлення, особистий кабінет: у своєму профілі користувачі можуть переглядати історію замовлень, змінювати особисті дані та налаштування. Під час розробки вебзастосунку було впроваджено ряд технічних рішень, таких як використання REST API для взаємодії з базою даних, реалізація системи авторизації користувачів. Таким чином, розроблений вебзастосунок забезпечує всі необхідні функції для комфортного і безпечного онлайн-шопінгу, використовуючи передові технології і підходи в розробці вебзастосунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Spring Boot in Action: Craig Walls - December 2015
2. React Quickly, Second Edition 2nd ed. Edition. Azat Mardan, Morten Barklund
3. Learning React: Modern Patterns for Developing React Apps 2nd Edition, O'Reilly, Alex Banks, Eve Porcello – 2020
4. Java: The Complete Reference, Twelfth Edition 12th Edition. Herbert Schildt
5. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL (Coding & Programming - QuickStart Guides) Paperback – Illustrated, November 18, 2019
6. Mui [Електронний ресурс]. Режим доступу:https://mui.com/material-ui/getting-started/ (дата звернення: 12.03.2024).
7. Tailwind CSS [Електронний ресурс]. Режим доступу:https://tailwindui.com/documentation (дата звернення: 16.03.2024).
8. SQL Cookbook. by Anthony Molinaro Released December 2005
9. Youtube [Електронний ресурс].Режим доступу: https://youtu.be/8SGI_XS5OPw?si=QGZejb4TLP0FX0Ie (дата звернення: 15.03.2024).
10. Youtube [Електронний ресурс]. Режим доступу:https://youtu.be/L2KHCWJ3gjs?si=F07cRz755leONWnB (дата звернення: 16.03.2024).
11. Youtube [Електронний ресурс]. Режим доступу: https://youtu.be/BHEPVdfBAqE?si=JAljunZe2Ry5U9Hq (дата звернення: 16.03.2024).
12. Udemy [Електронний ресурс]. Режим доступу: https://www.udemy.com/share/109kIa/ (дата звернення: 03.03.2024).
13. Spring Rest [Електронний ресурс]. Режим доступу: https://spring.io/guides/tutorials/rest (дата звернення: 02.04.2024).

14. Spring Security [Електронний ресурс]. Режим доступу: https://spring.io/guides/gs/securing-web (дата звернення: 10.04.2024).

15. MySQL Tutorial [Електронний ресурс].Режим доступу: https://www.w3schools.com/MySQL/default.asp (дата звернення: 12.04.2024).

ДОДАТКИ

Додаток А. Лістинг коду розроблення контролеру авторизації

```
package com.groceryHouse.groceryhouse.controller;
import com.groceryHouse.groceryhouse.config.JwtProvider;
import com.groceryHouse.groceryhouse.model.Cart;
import com.groceryHouse.groceryhouse.model.USER_ROLE;
import com.groceryHouse.groceryhouse.model.User;
import com.groceryHouse.groceryhouse.repository.CartRepository;
import com.groceryHouse.groceryhouse.repository.UserRepository;
import com.groceryHouse.groceryhouse.request.LoginRequest;
import com.groceryHouse.groceryhouse.response.AuthResponse;
import com.groceryHouse.groceryhouse.service.CustomerUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.BadCredentialsException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.Collection;
@RestController
@RequestMapping("/auth")
```

```
public class AuthController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private JwtProvider jwtProvider;
    @Autowired
    private CustomerUserDetailsService customerUserDetailsService;
    @Autowired
    private CartRepository cartRepository;

    @PostMapping("/signup")
    public ResponseEntity<AuthResponse>createUserHandler(@RequestBody User user)
throws Exception {
        User isEmailExist=userRepository.findByEmail(user.getEmail());

        if(isEmailExist!=null){
            throw new Exception("Електронна пошта вже в іншому акаунті");
        }
        User createdUser=new User();
        createdUser.setEmail(user.getEmail());
        createdUser.setFullName(user.getFullName());
        createdUser.setRole(user.getRole());
        createdUser.setPassword(passwordEncoder.encode(user.getPassword()));

        User savedUser = userRepository.save(createdUser);

        Cart cart= new Cart();
        cart.setCustomer(savedUser);
```

```
    cartRepository.save(cart);

    Authentication authentication=new
UsernamePasswordAuthenticationToken(user.getEmail(),user.getPassword());
    SecurityContextHolder.getContext();
    String jwt=jwtProvider.generateToken(authentication);

    AuthResponse authResponse= new AuthResponse();
    authResponse.setJwt(jwt);

    authResponse.setMessage("Реєстрація успішна");

    authResponse.setRole(savedUser.getRole());

    return new ResponseEntity<>(authResponse, HttpStatus.CREATED);
  }

 @PostMapping("/signin")
 public ResponseEntity<AuthResponse> signin(@RequestBody LoginRequest req){

   String username=req.getEmail();
   String password=req.getPassword();
   Authentication authentication= authenticate(username,password);
   Collection<? extends GrantedAuthority>authorities=authentication.getAuthorities();
   String role=authorities.isEmpty()?null:authorities.iterator().next().getAuthority();
   String jwt=jwtProvider.generateToken(authentication);

   AuthResponse authResponse= new AuthResponse();
   authResponse.setJwt(jwt);
```

```
        authResponse.setMessage("Вхід успішний");
        authResponse.setRole(USER_ROLE.valueOf(role));

        return new ResponseEntity<>(authResponse, HttpStatus.OK);
    }

    private Authentication authenticate(String username, String password) {


        UserDetails userDetails=
customerUserDetailsService.loadUserByUsername(username);

        if(username==null){
            throw new BadCredentialsException("невірний логін...");
        }
        if (!passwordEncoder.matches(password,userDetails.getPassword())){
            throw new BadCredentialsException("невірний пароль...");
        }
        return new
UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());
    }
}
```

Додаток Б. Лістинги коду файлу Authenticaion_Action.js

```
import axios from "axios";
import {api, API_URL} from "../../components/config/api";

export const registerUser=(reqData)=>async (dispatch)=>{
    dispatch({type:"REGISTER_REQUEST"})
    try {
        const {data} = await axios.post(`${API_URL}/auth/signup`,reqData.userData)
        if (data.jwt)localStorage.setItem('jwt',data.jwt);
        if(data.role==="ROLE_ADMIN"){
            reqData.navigate("/admin/product")
        }
        else {
            reqData.navigate("/")
        }
        dispatch({type:"REGISTER_SUCCESS",payload:data.jwt})
        console.log("register success",data)
    }
    catch(error){
        console.log("error",error)
    }
}
export const loginUser=(reqData)=>async (dispatch)=>{
    dispatch({type:"LOGIN_REQUEST"})
    try {
        const {data} = await axios.post(`${API_URL}/auth/signin`,reqData.userData)
        if (data.jwt)localStorage.setItem('jwt',data.jwt);
        if(data.role==="ROLE_ADMIN"){
            reqData.navigate("/admin/product")
        }
```

```
    else {
      reqData.navigate("/")
    }
    dispatch({type:"LOGIN_SUCCESS",payload:data.jwt})
    console.log("login success",data)
  }
  catch(error){
    console.log("error",error)
  }
}
export const getUser=(jwt)=>async (dispatch)=>{
  dispatch({type:"GET_USER_REQUEST"})
  try {
    const {data} = await api.get(`/auth/signin`,{
      headers:{
        Authorization: `Bearer ${jwt}`
      }
    })
    dispatch({type:"LOGIN_SUCCESS",payload:data})
    console.log("added to favorite",data)
  }
  catch(error){
    console.log("error",error)
  }
}
export const addToFavorite=({jwt,productId})=>async (dispatch)=>{
  dispatch({type:"ADD_TO_FAVORITE_REQUEST"})
  try {
    const {data} = await api.put(`/api/products/${productId}/add-favorite`,{},{
      headers:{
```

```
            Authorization: `Bearer ${jwt}`
          }
        })
        dispatch({type:"ADD_TO_FAVORITE_SUCCESS",payload:data})
        console.log("user profile",data)
    }
    catch(error){
        console.log("error",error)
    }
}
export const logout=()=>async (dispatch)=>{
    dispatch({type:"ADD_TO_FAVORITE_REQUEST"})
    try {
        dispatch({type:"LOGOUT"})
        console.log("logout success")
    }
    catch(error){
        console.log("error",error)
    }
}
```