

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ
АВТОМАТИЗАЦІЇ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE
TOOLS FOR AUTOMATED DECISION-MAKING IN E-COMMERCE

Виконав(ла): студент(ка) 2 курсу, групи КНТ-123м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДЯЧЕНКО М.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДЯЧЕНКО Максим Олександрович
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація засобів автоматизації прийняття рішень для електронної комерції. Research and Software Implementation of the Tools for Automated Decision-making in E-commerce

керівник проєкту (роботи) д-р філософ., ЛЕОЩЕНКО Сергій Дмитрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “07” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 02 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЕОЩЕНКО С.Д., доцент		
Нормоконтролер	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “ 09 ” вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис)

Максим ДЯЧЕНКО

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Сергій ЛЕОЩЕНКО

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
88 с., 3 табл., 42 рис., 3 дод., 15 джерел.

АВТОМАТИЗАЦІЯ, БІЗНЕС, ПІДПРИЄМСТВО, КОМЕРЦІЯ,
КЛІЄНТ, TYPESCRIPT, REACT, БРАУЗЕР, NODE.JS, HTTP, DATABASE,
REST API, SQL, VITE, TYPEORM, POSTGESQL, CRM.

Об'єкт дослідження – програмні засоби автоматизації прийняття рішень для електронної комерції.

Предмет дослідження – веб-застосунок, який допомагає автоматизувати та покращити ефективність прийняття рішень для електронної комерції.

Мета роботи – дослідження сучасних front-end веб-технологій та їх застосування на прикладі створення веб-сайту для автоматизації роботи, з фокусом на поліпшенні прийняття рішень для електронної комерції.

Матеріали, методи та технічні засоби: метод автоматизації прийняття рішень для електронної комерції, фреймворк Nest.js, мова програмування Node.js, мова розмітки HTML, таблиця каскадних стилів Tailwind CSS, база даних PostgreSQL, середовище розробки Visual Studio Code.

Результати. Було розроблено програмне забезпечення, спрямоване на поліпшення прийняття рішень для електронної комерції.

Висновки. Реалізовано дослідження та програмна реалізація прийняття рішень для електронної комерції. Проект буде поєднувати в собі інтуїтивний інтерфейс, базу даних клієнтів та можливість зручного управління структурованими даними, що оптимізуватиме бізнес-процеси та підвищить її ефективність.

Галузь використання – приватні підприємства, бізнеси, власна комерція.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 88 pages, 3 tables, 42 pictures, 3 appendixes, 15 sources.

AUTOMATION, BUSINESS, ENTERPRISE, COMMERCE, CLIENT, TYPESCRIPT, REACT, BROWSER, VISUAL STUDIO CODE, GITHUB, NODE.JS, HTTP, DATABASE, REST API, SQL, VITE, TYPEORM, POSTGESQL, CRM.

Research subject – decision automation software for e-commerce.

The subject of the research is a web application that helps automate and improve decision-making efficiency for e-commerce.

The purpose of the work is to study of modern front-end web technologies and their application on the example of creating a website to automate project work, with a focus on improving decision-making for e-commerce.

Materials, methods and technical means: decision automation method for e-commerce, Next.js framework, Node.js programming language, HTML markup language, Tailwind CSS cascading style sheet, PostgreSQL database, Visual Studio Code development environment.

Results. Software was developed to improve decision-making for e-commerce.

Conclusions. Research and software implementation of decision-making for e-commerce was carried out. The project will combine an intuitive interface, a customer database, and the ability to manage structured data in a highly transparent way, which will optimize business processes and increasde efficiency.

Field of application - private enterprises, businesses, personal commerce.

ЗМІСТ

	С.
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Основні поняття і актуальність автоматизації в електронній комерції	10
1.2 Огляд існуючих програмних рішень для автоматизації прийняття рішень	11
1.3 Висновки за розділом 1	16
2 Матеріали та методи	17
2.1 Вибір мови програмування та інших технологій розробки.....	17
2.2 Вибір середовища розробки для створення програмного забезпечення	23
2.3 Використані моделі даних.....	25
2.4. Висновки за розділом 2	27
3 Опис програми.....	28
3.1 Структура програмного забезпечення автоматизації прийняття рішень для електронної комерції.....	28
3.2 Функціонування програмного забезпечення для автоматизації прийняття рішень для електронної комерції.....	29
3.3 Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для автоматизації прийняття рішень для електронної комерції	34
3.4 Розширення функціональності програмного забезпечення для автоматизації прийняття рішень для електронної комерції за допомогою модулів.....	39
3.5 Висновки за розділом 3	41
4 Експлуатація, тестування та експериментальне дослідження програми	42
4.1 Призначення й умови застосування програми.....	42

4.2	Характеристики програмного забезпечення для автоматизації прийняття рішень для електронної комерції.....	44
4.3	Інструкція по експлуатації програми.....	45
4.3.1	Звернення до програми.....	45
4.3.2	Вхідні й вихідні дані.....	45
4.3.3	Повідомлення	46
4.4	Виконання програмного забезпечення для автоматизації прийняття рішень для електронної комерції.....	46
4.5	Тестування програмного забезпечення для автоматизації прийняття рішень для електронної комерції.....	51
4.6	Висновки за розділом 4	51
	Висновки	53
	Перелік джерел посилання.....	55
	Додаток А Технічне завдання	57
	Додаток Б Фрагмент тексту програми	61
	Додаток В Слайди презентації.....	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CSS – Cascading Style Sheets;

HTML – HyperText Markup Language;

HTTPS – HyperText Transfer Protocol Secure;

JS – JavaScript;

OS – Operating System;

DB – Data Base;

CRM – Customer Relationship Management;

VS – Visual Studio;

БД – база даних;

СКБД – система керування базами даних.

ВСТУП

Сучасний світ відзначається стрімким розвитком інформаційних технологій, які значно впливають на всі аспекти нашого життя, включаючи бізнес.

Управління великим обсягом інформації, що стосується замовлень, клієнтів, витрат і залишків продукції, є нелегким завданням для власників підприємств.

Із збільшенням обсягів виробництва питання зберігання та використання інформації стає більш важким та важливим. На допомогу приходять підручні блокноти і нотатки, але і вони не надають належної ефективності та можуть призвести до витрати часу і ресурсів.

Час - це найцінніший ресурс, яким ми володіємо, і його справжня цінність стає очевидною, коли ми його втрачаємо. Суть цього виразу полягає в тому, що не правильне використання часу має вирішальне значення для успіху компанії. Маючи купу роботи з продукцією, власник додатково повинен вести підрахунок наявних інструментів, замовлень і клієнтів. Це на пряму впливає на оптимізацію роботи і ефективності підприємства. Саме тому і постає питання в корисному електронному помічнику.

Без сумніву, кожен в дитинстві, маючи великі об'єми однотонної роботи замислювався про те, як би це зробити легшим для виконання. Цю роль виконує розроблена програма. Сучасні програми пов'язані з електронною комерцією дозволяють власникам зосередитися більше на стратегічних, а не на рутинних завданнях.

Метою роботи є дослідження та програмна реалізація прийняття рішень для електронної комерції. Проект буде поєднувати в собі інтуїтивний інтерфейс, базу даних клієнтів та можливість гучного управління структурованими даними, що оптимізуватиме бізнес-процеси та підвищить її ефективність.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Основні поняття і актуальність автоматизації в електронній комерції

Автоматизація в електронній комерції – це процес впровадження технологічних рішень для керування рутинними та повторюваними завданнями, такими як: обробка замовлень, управління запасами, обслуговування бази даних клієнтів і аналіз продажів. Це дозволяє компаніям зменшувати витрати часу та ресурсів, забезпечуючи при цьому точність та швидкість операцій. Завдяки автоматизації бізнес має можливість оптимізувати внутрішні процеси та покращити якість обслуговування клієнтів.

Зростання електронної комерції викликає потребу в автоматизації, оскільки обсяг даних, які потрібно обробити, значно перевищує можливості людського керування. Перед сучасним бізнесом стоїть завдання підтримувати високу швидкість і персоналізацію обслуговування, які є важливими складовими конкурентоспроможності. У таких ситуаціях автоматизація стає не тільки корисною, але й в край необхідною.

Крім того, автоматизація дозволяє знизити ризики, пов'язані з людським фактором. Помилки в обробці замовлень, несвоєчасне поповнення запасів або некоректна взаємодія з клієнтами можуть мати значний вплив на репутацію компанії.

На допомогу приходять автоматизовані системи, такі як CRM для управління клієнтами або ERP для інтеграції бізнес-процесів, що забезпечують надійний контроль і прописані операції.

Актуальність автоматизації також зумовлена стрімкими змінами на ринку. Підприємства повинні швидко адаптуватися до нових ситуацій, таких як зростання попиту на продукцію, розширення сфери діяльності або вихід на міжнародний ринок. Автоматизовані системи дозволяють бізнесу

розширювати її кордони без збільшення витрат, що є ключовою перевагою у великих масштабах.

Тому автоматизація в електронній комерції є основним інструментом підвищення ефективності, точності та конкурентоспроможності бізнесу. Це відкриває нові можливості для зростання та адаптації до сучасного динамічного ринкового середовища.

1.2 Огляд існуючих програмних рішень для автоматизації прийняття рішень

В сучасному світі дуже багато різних платформ та систем, які слугують для планування автоматизації взаємодії між замовниками, задля покращення рівня продажів, зберігання інформації про клієнтів та оптимізації маркетингу.

Наприклад системи автоматизації прийняття рішень KeyCRM, Prom.ua, Microsoft Dynamics 365.

Всі ці системи об'єднує спільна мета – автоматизація бізнес-процесів і підтримка рішень. Деякі з вищеперерахованих систем були створені українськими розробниками, деякі закордонними. Не дивлячись на це, усі вони виконують однакові задачі: підрахунок продажів та прибутку, роботу з персоналом, менеджмент проектів, адміністрування сайту, оформлення замовлень, тощо.

KeyCRM – це багатофункціональна українська CRM-система, яка зосереджена на автоматизації бізнес-процесів, особливо для електронної комерції, маркетингів та невеликих бізнесів, логотип якої зображено на рисунку 1.1 [1].

KeyCRM вирішує такі задачі:

- автоматично збирає ліди з Viber, Telegram, Instagram, Facebook (включаючи Lead Ads), телефонії та онлайн-чатів;

- уся комунікація з клієнтами (месенджери, email, IP-телефонія, чати) об'єднана в одному інтерфейсі;
- замовлення з усіх каналів продажу (інтернет-магазинів, маркетплейсів, Instagram) інтегруються в одне вікно;
- генерація ТТН, відстеження посилок та інші процеси відправки автоматизовані;
- контроль залишків, автосписання, синхронізація складу з магазинами, а також підтримка сканерів і штрихкодів для швидкого обліку;
- постановка задач із пріоритетами, контролем дедлайнів та статусів виконання;
- система пропонує детальну аналітику продажів, товарів, менеджерів, фінансів та витрат.

На рисунку 1.1 продемонстровано інтерфейс системи KeyCRM.

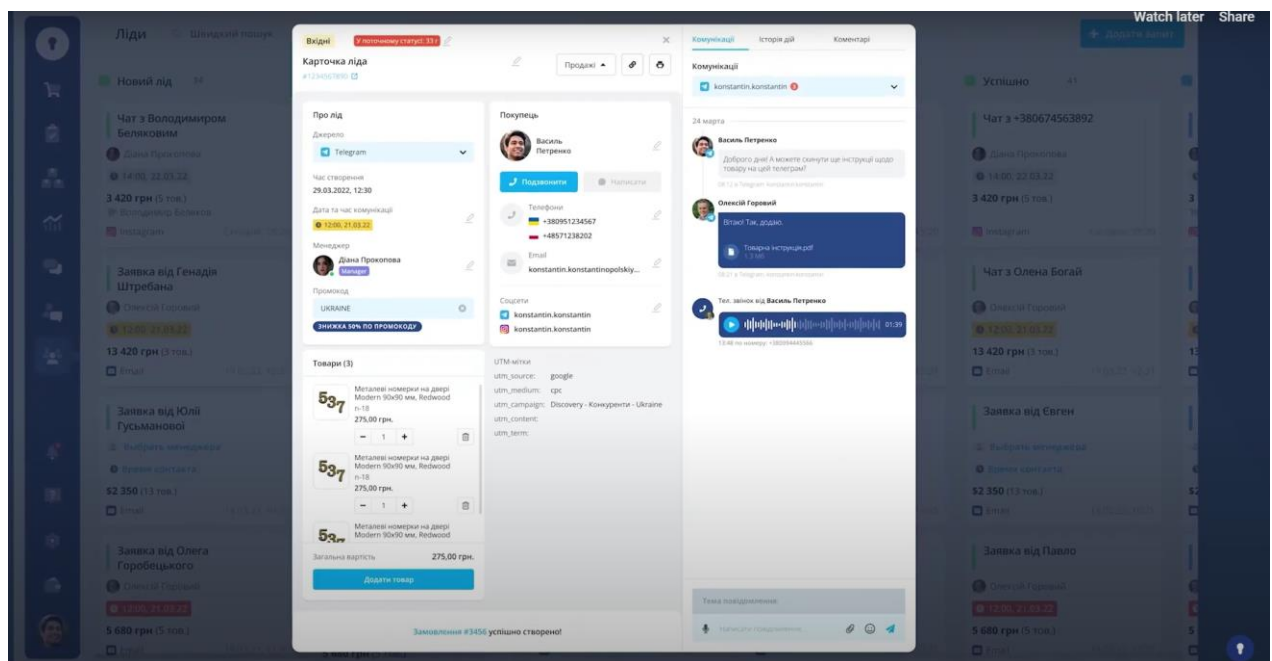


Рисунок 1.1 – Інтерфейс системи управління навчання KeyCRM [1]

Prom.ua – це український маркетплейс, у якому продавці самостійно створюють інтернет-магазини і розміщують товари в загальному каталозі. Для покупців на Prom.ua зібрано більше 100 мільйонів товарів. Наразі Prom.ua пропонує три тарифних планів, в залежності від кількості товарів в асортименті продавця.

На рисунку 1.2 продемонстровано інтерфейс маркетплейсу Prom.ua.

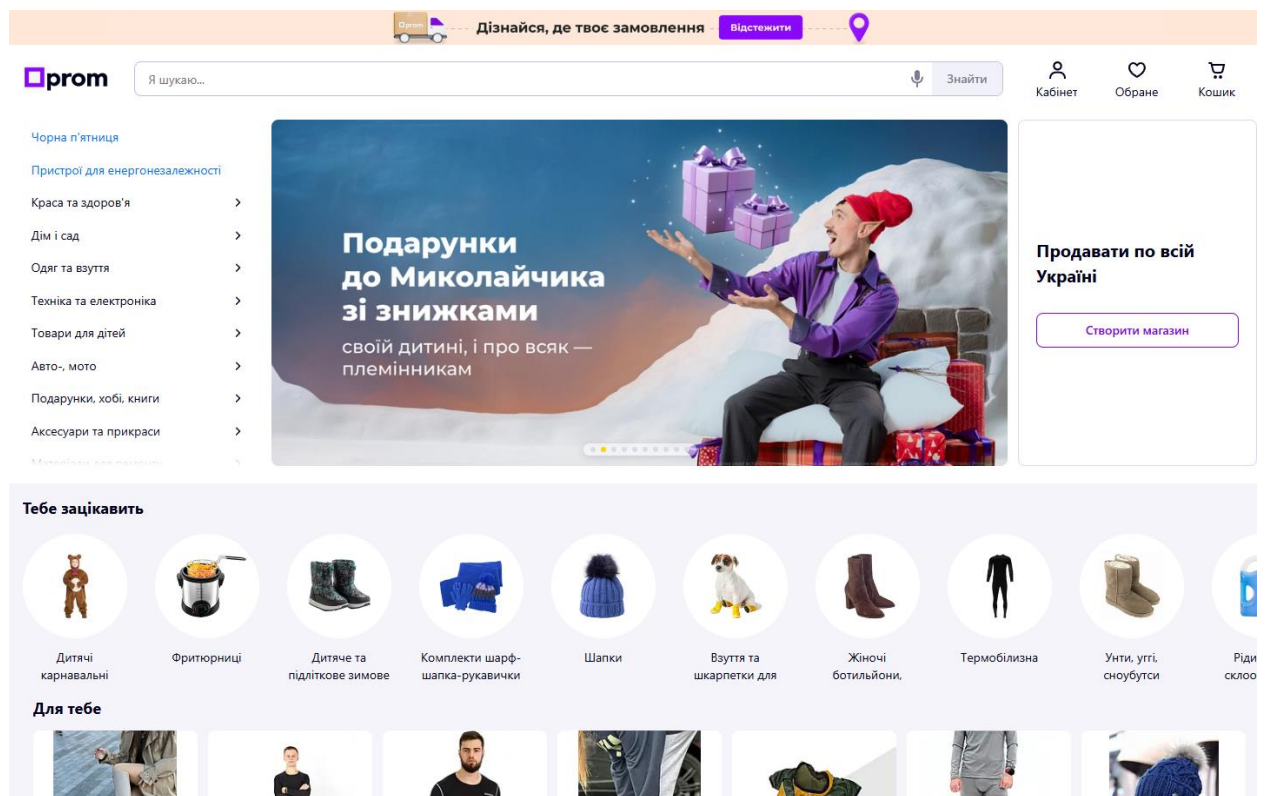


Рисунок 1.2 – Інтерфейс системи маркетплейсу Prom.ua [2]

Microsoft Dynamics 365 – це система, робота якої залежить від того, наскільки точно їм вдається прогнозувати потреби клієнта. Цей продукт виготовлений компанією Microsoft. Покликана для автоматизації відділу продажів. Вибір користувача припадає саме на цю систему через автоматизовану комунікацію в одній системі.

На рисунку 1.3 продемонстровано інтерфейс системи Microsoft Dynamics 365.

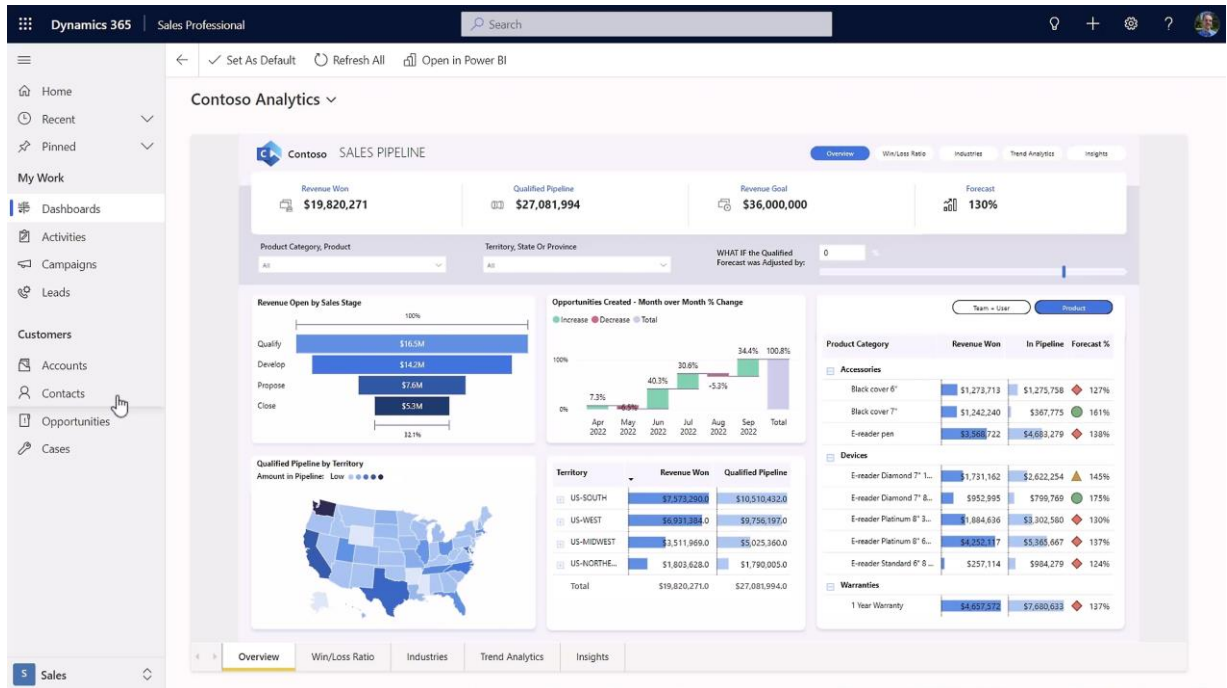


Рисунок 1.3 – Інтерфейс системи Microsoft Dynamics 365 [3]

Порівняння наведених систем автоматизації наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння систем автоматизації прийняття рішень

Характеристика	KeyCRM	Prom	Microsoft Dynamics 365
Основна мета	Управління взаємовідносинами з клієнтами	Платформа для автоматизації продажів	Комплексне управління бізнесом і CRM
Основні переваги	Легкість у впровадженні, зручний інтерфейс	Поглиблене управління ресурсами і ланцюгами постачання	Потужна аналітика, інтеграція з Microsoft

Кінець таблиці 1.1

Характеристика	KeyCRM	Prom	Microsoft Dynamics 365
Основні можливості	Робота з клієнтами, автоматизація продажів і мультиканальна підтримка.	Управління процесами, автоматизація складу та звітність.	Інтеграція з продуктами Microsoft, аналіз і прогнозування даних, управління фінансами та персоналом.
Підтримка аналітики та звітності	Основна аналітика для продажів і маркетингу	Поглиблена аналітика для планування	Потужні аналітичні інструменти
Масштабованість	Підходить для малих і середніх підприємств	Підходить для середніх і великих підприємств	Підходить для великих підприємств
Ціна	Доступна для малих і середніх бізнесів	Зазвичай вища за KeyCRM, залежно від функціональності	Висока, особливо для великих компаній
Гнучкість налаштування	Висока для адаптації до потреб бізнесу	Гнучка, але з обмеженнями для специфічних випадків	Висока, з багатьма параметрами налаштування

Можна зробити висновки, що KeyCRM підходить для малих підприємств, через її доступності та простоті впроваджені. Prom підійде

більше для середнього бізнесу з його можливостями в управлінні ресурсами та постачанням. Якщо ви керівник великого підприємства то кращим варіантом буде обрати Microsoft Dynamics 365, щоб максимально оптимально керувати бізнесом та легко аналізувати прогрес.

1.3 Висновки за розділом 1

Відповідно до проблеми, що розглядається, результатами виконання магістерської роботи необхідно забезпечити виконання наступних завдань:

- проаналізувати принципи роботи систем автоматизації в електронній комерції;
- дослідити методи підтримки роботи систем автоматизації прийняття рішень;
- виконати проектування програмної системи підтримки роботи систем автоматизації в електронній комерції;
- реалізувати програмну систему підтримки роботи системи управління автоматизації прийняття рішень для електронної комерції з інтуїтивним та зрозумілим інтерфейсом.

У першому розділі, перше завдання було вирішене таким чином: спочатку було досліджено основні поняття та актуальність автоматизації в електронній комерції. Після цього проведено огляд і аналіз програмних аналогів, серед яких для детального розгляду було виділено KeyCRM, Prom та Microsoft Dynamics 365. Саме ці системи автоматизації прийняття рішень було визначено як основний прототип розроблюваної програмної системи.

2 МАТЕРІАЛИ ТА МЕТОДИ

2.1 Вибір мови програмування та інших технологій розробки

Вибір мови програмування та інших технологій є важливим етапом у процесі розробки програмного продукту. У цьому проекті для створення front-end було використано бібліотеку React.js та Typescript. Для побудови та запуску проекту було обрано Vite. Для стилізації користувацького інтерфейсу було використано фреймворк Tailwind CSS.

Для back-end розробки було використано прогресивний фреймворк для побудови серверних додатків на базі TypeScript Nest.JS. TypeORM - це інструмент для роботи з базами даних, який використовує об'єкти та класи замість SQL-запитів.

Для зберігання даних було використано реляційну базу даних PostgreSQL. Одним з переваг роботи з даною БД є її підтримка стандарт SQL та JSON.

Також проект базується на архітектурі REST API. Це стиль розробки веб-сервісів, який використовує HTTP-протокол для взаємодії між клієнтом і сервером.

React.js - це перша бібліотека, що почала використовувати Virtual DOM для оптимізації оновлення DOM дерева. Суть оптимізації в тому, щоб оновлювати лише ті дані, які дійсно змінились, а для пошуку відмінностей використовувати полегшену віртуальну структуру. Однак це йому не заважає бути найпопулярнішою бібліотекою серед фреймворків. На рисунку 2.1 зображено рейтинг найпопулярніших фреймворків, де React.js займає кращі позиції.

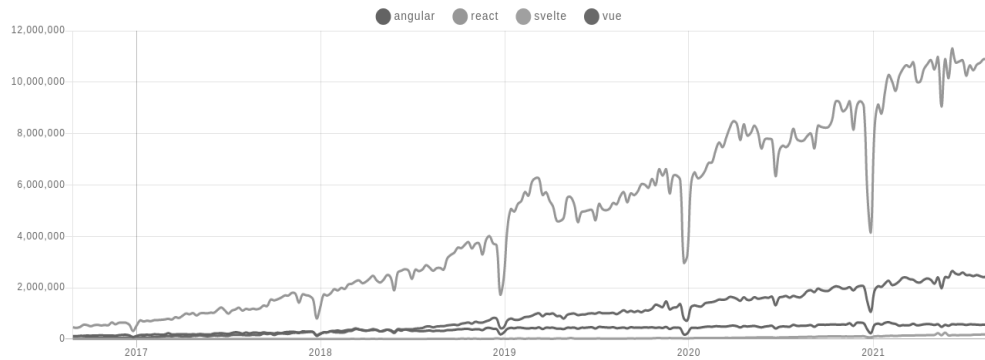


Рисунок 2.1 – Рейтинг найпопулярніших фреймворків [4]

Його рідко використовують окремо, тому одразу підключають такі додаткові бібліотеки, як React Router, React DOM або їхні аналоги.

React зручний у розумінні, велику гнучкість і свободу, що є перевагою для невеликих проєктів, але й не менш потужний, щоб використовувати його для великих.

Основною проблемою є його перевага — потрібно збирати фреймворк самому, з великої кількості бібліотек. Типовий проєкт на React виглядає так: React, React Router, React DOM, Redux, Redux React. Якщо все це упакувати в коробку, вийшов би фреймворк, але це все потрібно робити самотужки.

Але світ IT не був би собою, якби не були знайдені рішення. На просторах GitHub можна знайти безліч шаблонів, які генерують повністю готовий для роботи проєкт для будь-яких потреб, з будь-якими препроцесорами CSS, бібліотеками для роутерів чи управління станом, тому хвилюватись за відсутність повноцінної CLI не варто [4].

TypeScript – це безкоштовна високорівнева мова програмування з відкритим вихідним кодом, розроблена Microsoft, яка додає до JavaScript статичний тип із додатковими анотаціями типу. Він розроблений для розробки великих додатків і транспілюється на JavaScript [5].

TypeScript виник через недоліки JavaScript для розробки великомасштабних програм як у Microsoft, так і серед їхніх зовнішніх

клієнтів. Проблеми роботи зі складним кодом JavaScript призвели до попиту на спеціальні інструменти для полегшення розробки компонентів мови [6].

TypeScript — це строго типізована мова програмування, яка базується на JavaScript і надає вам кращі інструменти будь-якого масштабу.

Її головні аспекти:

- TypeScript додає до JavaScript додатковий синтаксис для підтримки більш тісної інтеграції з редактором;
- код TypeScript перетворюється на JavaScript, який виконується будь-де, де виконується JavaScript: у браузері, на Node.js, Deno, Bun і у ваших програмах;
- TypeScript розуміє JavaScript і використовує визначення типу, щоб надати чудові інструменти без додаткового коду. [7]

Vite - це сучасний інструмент для розробки front-end, який користується величезною популярністю.

Перш ніж зануритися в процес міграції з Webpack на Vite, варто зазначити, що середовище front-end розробки постійно розвивається. Порівняння цих двох пакувальників, наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння Webpack та Vite

Характеристика	Webpack	Vite
Підхід до збірки	Збирає весь проєкт у один або декілька бандлів перед запуском	Використовує ES для прямого завантаження файлів у браузер
Швидкість першої збірки	Повільна, особливо у великих проєктах	Дуже швидка завдяки відсутності необхідності у попередньому бандлінгу
Підтримка старих браузерів	Підтримує через використання поліфілів та транспіляції	Для старих браузерів потрібна додаткова конфігурація

Кінець таблиці 2.1

Характеристика	Webpack	Vite
Гнучкість налаштувань	Дуже гнучкий, але потребує більше часу для налаштування	Простіший у налаштуванні, більшість функцій «з коробки»
Призначення	Підходить для великих проєктів із багатьма залежностями	Ідеальний для невеликих або середніх проєктів
Підтримка TypeScript	Підтримує через інтеграцію з Babel або ts-loader.	Вбудована підтримка через плагіни Vite
Популярність	Більш відомий та використовується у великих корпоративних проєктах	Швидко набирає популярність, особливо серед розробників сучасних front-end фреймворків

Бібліотека Tailwind досить молода, перші бета-версії з'явилися в 2017 році. Вписуються короткі імена заздалегідь спроектованих класів: кольори, відступи, позиціонування. Вистачає класів і для сайзингу, анімації бекграундів та багато іншого. Код реалізації зображено на рисунку 2.2 [9].

```
import { TELEGRAM_LINK, SUPPORT_EMAIL } from '@/constants';

export const Contacts = memo(() => {
  return (
    <div className='t-tiny-regular text-center opacity-50'>
      <Trans i18nKey='common:signin--news'>
        <a className='underline' target='_blank' rel='noreferrer'
          href={TELEGRAM_LINK} />
      </Trans>
      <br />
      <Trans i18nKey='common:signin--support'>
        <a className='underline' href={`mailto:${SUPPORT_EMAIL}`} />
      </Trans>
    </div>
  );
});
```

Alexander Vodolazskikh, 10 months ago • Auth page fix ...

Рисунок 2.2 – Реалізація Tailwind на React JS

Головною та легкопомітною відмінністю Tailwind від інших бібліотек є те, що у строку з розміткою вписуються сформовані короткі назви класів, які мають свої стилі.

У цьому проєкті REST API став ключовим елементом. Його простота, гнучкість та широкі можливості дозволяють легко інтегрувати функціональність сайту з іншими системами та додатками. REST API відкриває шлях до ефективного обміну даними та взаємодії з сервером, що важливо для створення зручного та функціонального веб-додатку. REST API — один з найпопулярніших підходів для проєктування API ваших сервісів. Ця технологія здається простою, але, на жаль, дуже часто розробники неправильно розуміють або застосовують її принципи. Тим більше, що REST (на відміну від того ж HTTP) не є стандартом. Для проєкту використовується архітектура RESTful API, схему якої показано на рисунку 2.3.

REST – Architecture

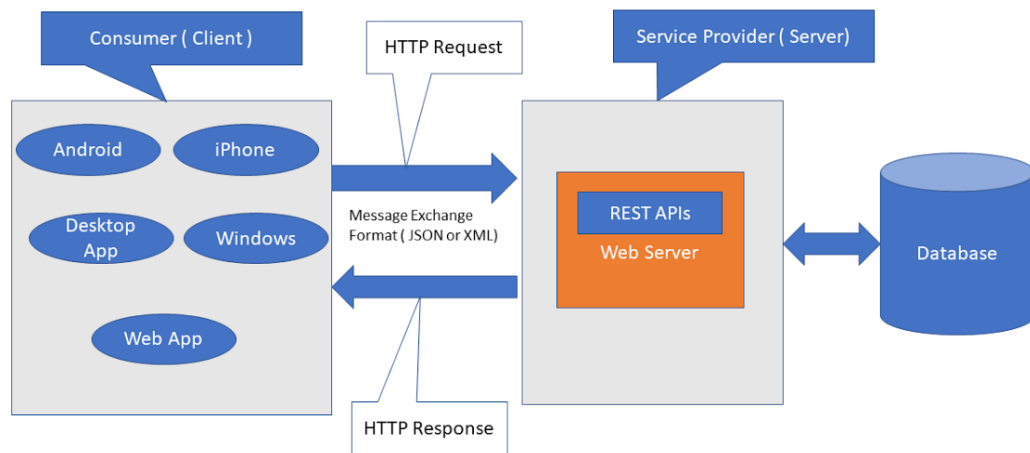


Рисунок 2.3 – Схема RESTful архітектури

Відповідно, якщо ви створюєте сервіси в своєму застосунку і вони відповідають цим обмеженням, то ви маєте усі підстави назвати свої сервіси RESTful. Ви можете використовувати будь-який протокол, формат даних або фреймворк для розробки своїх сервісів. Але традиційно саме протокол HTTP і формат JSON найбільш часто використовуються завдяки своїй простоті та популярності в IT. Тоді ваші сервіси матимуть назву RESTful веб-сервіси [3].

Що ж таке REST? Це не протокол, не формат даних і не фреймворк, це архітектурний стиль, який вводить 5 обмежень або принципів (constraints) для комунікації в розподіленій архітектурі:

- наявність клієнта і сервера;
- принцип stateless;
- єдині інтерфейси (Uniform interface) для ресурсів;
- layered-архітектура;
- можливості кешування відповіді сервера.

RESTful веб-сервіси використовують HTTP, а цей протокол вже є стандартом зі своєю специфікацією. Тому загальноживані конвенції потрібні для того, щоб правильно використовувати HTTP і щоб полегшити процес розробки. Приклад роботи відображено на рисунку 2.4.

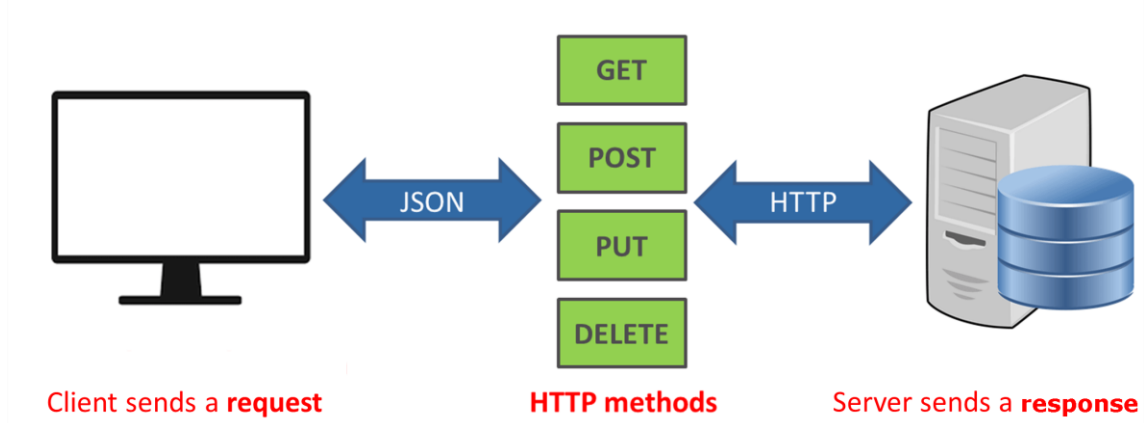


Рисунок 2.4 – Приклад роботи з HTTP методом

NestJS — це сучасний фреймворк для розробки серверних застосунків на TypeScript або JavaScript, створений на базі Express.js. NestJS, фреймворк, який повністю написаний на TypeScript, він легко тестується і містить все необхідне. NestJS побудований на Express.js. Будь-які розширення для Express.js, легко впровадити в NestJs.

Структура NestJS організована у вигляді модулів, задля швидкого та легкого масштабування. Цей фреймворк заснований на архітектурі MVC (Model-View-Controller), що дозволяє легко керувати залежностями між компонентами. Також NestJS може інтегруватися з будь-якими базами даних, зокрема такими як TypeORM.

2.2 Вибір середовища розробки для створення програмного забезпечення

Visual Studio Code (VS Code) — це спрощений, але потужний редактор вихідного коду, який запускається на комп'ютері (рисунок 2.5) й доступний для Windows, macOS і Linux. Вона підтримує JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов (наприклад, C++, C#, Java, Python, PHP і Go) і середовищ виконання (наприклад, .NET і Unity).[10]

Для професійного програмування часто користуються інструментами, які розуміють код краще, ніж прості блоки тексту. Visual Studio Code включає в себе вбудовану підтримку автодоповнення коду IntelliSense[11], багатфункціональне семантичне розуміння коду та навігацію, а також редакторинг коду.

IntelliSense - технологія автодоповнення Microsoft, найбільш відома в Microsoft Visual Studio. Допишує назву функції при введенні початкових букв. Крім прямого призначення IntelliSense використовується для доступу до документації та для усунення неоднозначності в іменах змінних, функцій і методів, використовуючи рефлексію [11].

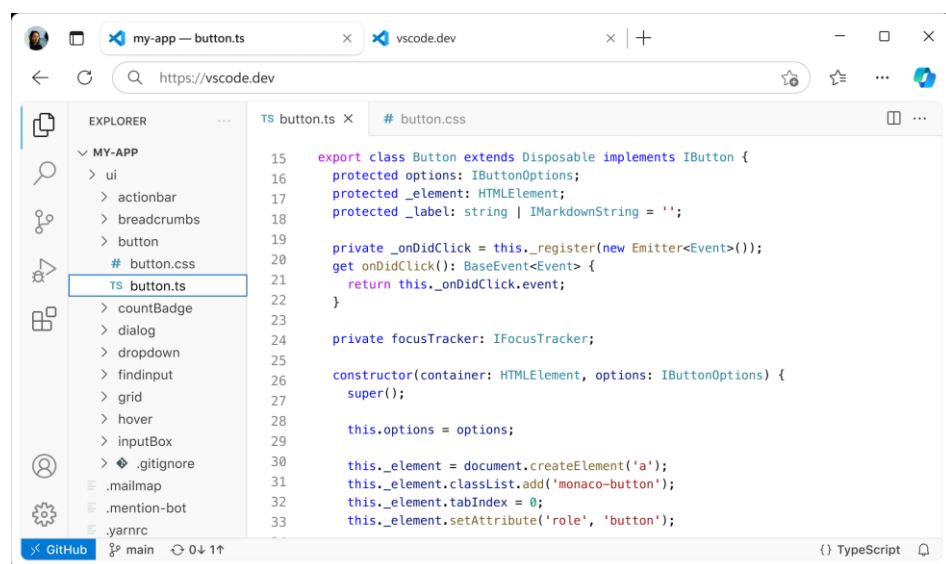


Рисунок 2.5 – Застосунок Visual Studio Code [12]

Як відомо, VS Code — це найпопулярніший редактор у світі. Причому як серед програмістів, так і серед фахівців усіх спеціальностей загалом, свідчать дані Stackoverflow [13].

Обґрунтування вибору Visual Studio Code як середовища розробки для створення програмного забезпечення:

- не вимагає багато ресурсів системи, що робить його швидким навіть на середніх конфігураціях;
- має вбудовану підтримку для багатьох мов програмування;

- легке додавання плагінів для будь-якої мови програмування;
- інтеграція з Git дозволяє виконувати команди безпосередньо в редакторі;
- наявність терміналу прямо в середовищі дозволяє запускати команди, компілювати код і працювати з іншими інструментами, не залишаючи редактора;
- велика спільнота розробників забезпечує постійне оновлення та швидкий розвиток;
- VS Code є абсолютно безкоштовним і з відкритим вихідним кодом, що робить його доступним для всіх;
- можливість інтеграції з Azure, Docker, і навіть розробка в хмарі через Codespaces.

2.3 Використані моделі даних

Розділяють два способи організації даних:

- реляційна база даних (SQL);
- нереляційна база даних (NoSQL).

SQL – це мова запитів для отримання з бази даних необхідної інформації. Використовується в реляційних базах даних, де дані знаходяться в вигляді таблиць, вони є структурованими та розкладені по рядках і стовпцях.

NoSQL – це загальна назва нереляційних баз даних. В NoSQL-базах може не бути таблиць, замість них є свої моделі даних, кожна з яких підходить під свої задачі. Порівняння типів баз даних наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння характеристик NoSQL та SQL

Характеристика	SQL	NoSQL
Тип Даних	Структуровані дані	Неструктуровані або поліструктуровані дані
Схема	Фіксована схема	Гнучка або відсутня схема
Мова запитів	SQL	Залежить від конкретного типу NoSQL бази даних
Масштабованість	Вертикальна та горизонтальна масштабованість	Горизонтальна масштабованість
Запити JOIN	Підтримка JOIN операцій	Зазвичай обмежена підтримка JOIN операцій
Функції	Реляційні бази даних надають широкі функціональні можливості, включаючи складні запити та агрегації	Зазвичай бази даних NoSQL надають обмежені функціональні можливості, фокусуючись на простоті та ефективності

PostgreSQL — це сучасна реляційна база даних із відкритим вихідним кодом. Вона широко використовується завдяки своїй потужності, надійності та розширюваності.

Також у проекті використовується TypeORM - це інструмент для роботи з базами даних на основі Object-Relational Mapping (ORM). Це означає, що можна працювати з базою даних, використовуючи об'єкти, а не безпосередньо SQL-запити.

Завдяки цьому таблиці в базі даних представляються у вигляді класів у коді. Рядки таблиці – це екземпляри, а поля – властивості класів. Замість написання SQL-запитів використовуються методи й властивості об'єктів.

У проєкті ці дві технології добре взаємодіють. PostgreSQL - це «місце», де зберігаються дані. TypeORM - це інструмент, що дозволяє взаємодіяти з PostgreSQL через об'єкти в коді.

2.4. Висновки за розділом 2

Для розробки клієнтської частини обрано бібліотеку React.js із використанням TypeScript з метою забезпечення типізації та зручності роботи з кодом. Для побудови та запуску проєкту було використано інструмент Vite, який забезпечує швидке створення та обслуговування застосунку.

Серверна частина створена на базі Nest.js — сучасного фреймворку, що забезпечує модульність і зручну організацію коду.

Для зберігання даних було обрано реляційну базу даних PostgreSQL. Її інтеграцію з back-end спрощує TypeORM, що дозволяє працювати з базою через об'єкти в коді, мінімізуючи потребу в написанні SQL-запитів вручну.

Використовується RESTful API для обміну даними через протоколи HTTP, що забезпечує надійний зв'язок між клієнтською та серверною частинами.

Для розробки додатку було обрано середовище Visual Studio Code. Воно забезпечує розширену підтримку таких ключових технологій, як React.js і Nest.js, що використовуються в проєкті, а також є безкоштовним, відкритим та потужним інструментом для розробників.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення автоматизації прийняття рішень для електронної комерції

Структура програмного забезпечення автоматизації прийняття рішень для електронної комерції наведено на рис. 3.1.

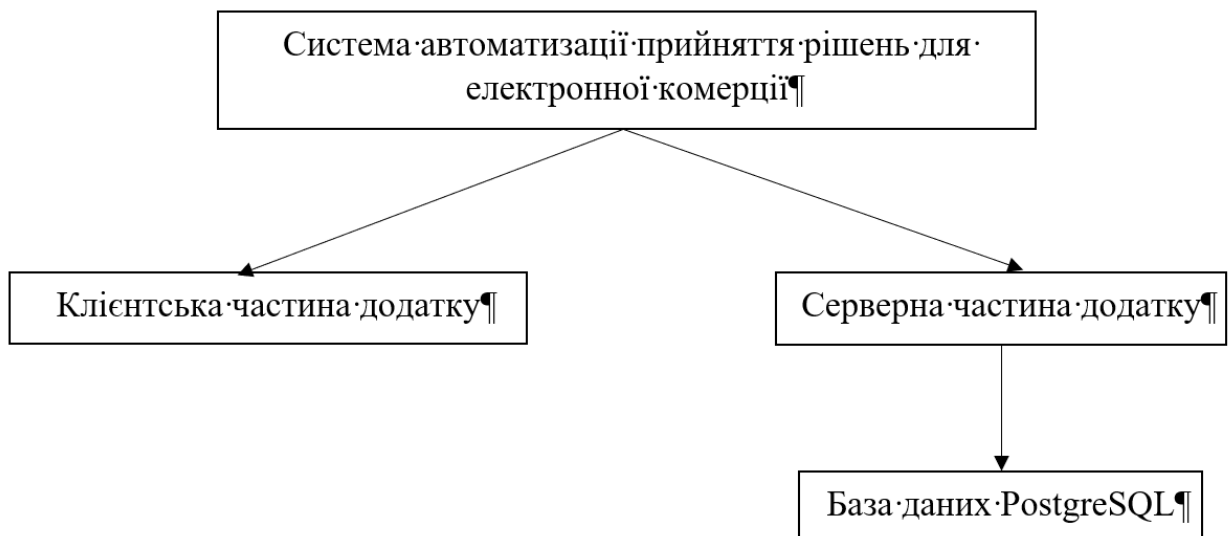


Рисунок 3.1 – Структура програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Як видно, основними складовими програмного забезпечення є такі:

- клієнтська частина додатку;
- серверна частина додатку;
- база даних PostgreSQL.

Виконаємо аналіз функціонального призначення ключових компонентів програмного забезпечення додатка для автоматизації прийняття рішень для електронної комерції.

Клієнтська частина додатку реалізована у вигляді веб-сторінок, створених із використанням фреймворку React.js, строго типізовані за

допомогою Typescript, мови розмітки HTML5 та каскадних стилів CSS, оформлених за допомогою фреймворку Tailwind CSS.

Серверну частину додатку створено на основі фреймворку Nest.js із використанням строгої типізації, забезпеченої мовою TypeScript.

Система автоматизації прийняття рішень для електронної комерції є кросплатформним веб-застосунком, доступ до якого здійснюється через Інтернет за протоколом HTTPS.

Для зберігання даних системи було обрано реляційну базу даних PostgreSQL, яка забезпечує безперебійний і безпечний доступ до інформації. Для взаємодії з базою даних використовується TypeORM, що дозволяє ефективно працювати з даними, забезпечуючи зручність управління та гнучкість завдяки підтримці об'єктно-реляційного підходу.

3.2 Функціонування програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Загальну схему функціонування програмного забезпечення, призначеного для автоматизації прийняття рішень у сфері електронної комерції, представлено на рисунку 3.2. Користувач взаємодіє з програмою шляхом створення запитів через елементи інтерфейсу такі як: кнопки та форми. На клієнтській стороні здійснюється попередня перевірка введених даних на коректність, після чого ці дані передаються на сервер.

Серверна частина отримує та обробляє дані, надіслані користувачем. У разі потреби сервер взаємодіє з базою даних для збереження або отримання необхідної інформації. Згенеровані сервером дані надсилаються назад на клієнтську сторону через веб-інтерфейс, де вони відображаються у зручному для користувача форматі.

Схему функціонування програмного забезпечення для автоматизації прийняття рішень у сфері електронної комерції також можна подати у вигляді діаграми станів, наведеної на рисунку 3.3.

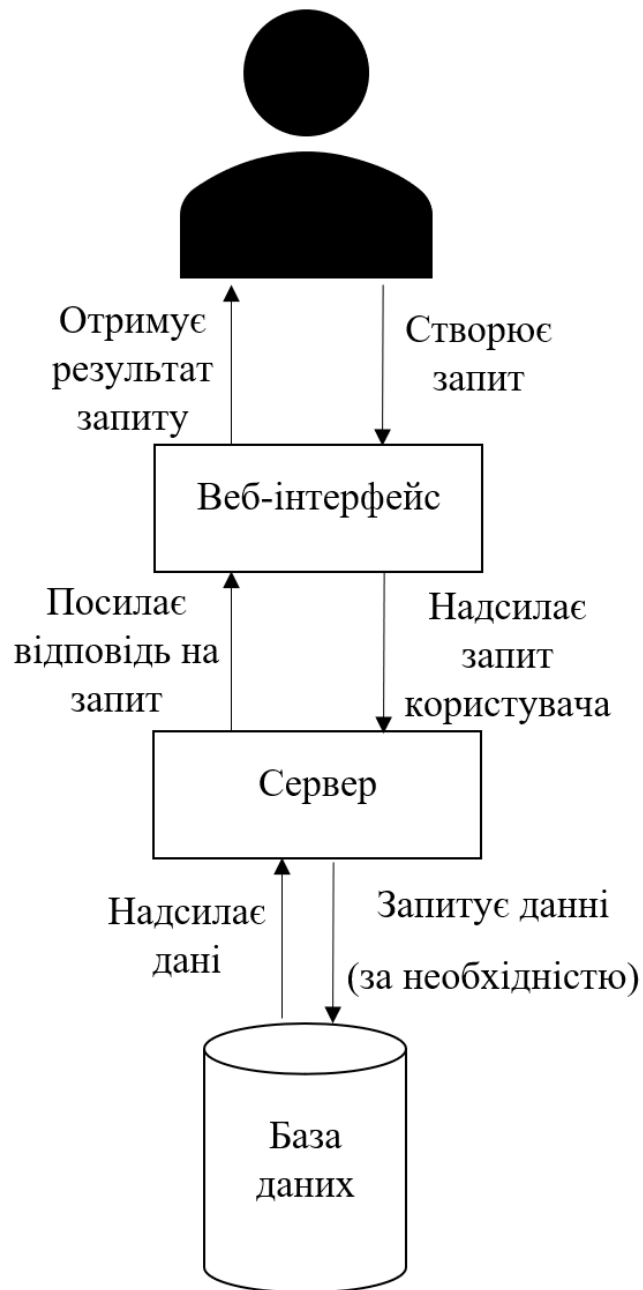


Рисунок 3.2 – Загальна схема функціонування програмного забезпечення для автоматизації прийняття рішень для електронної комерції

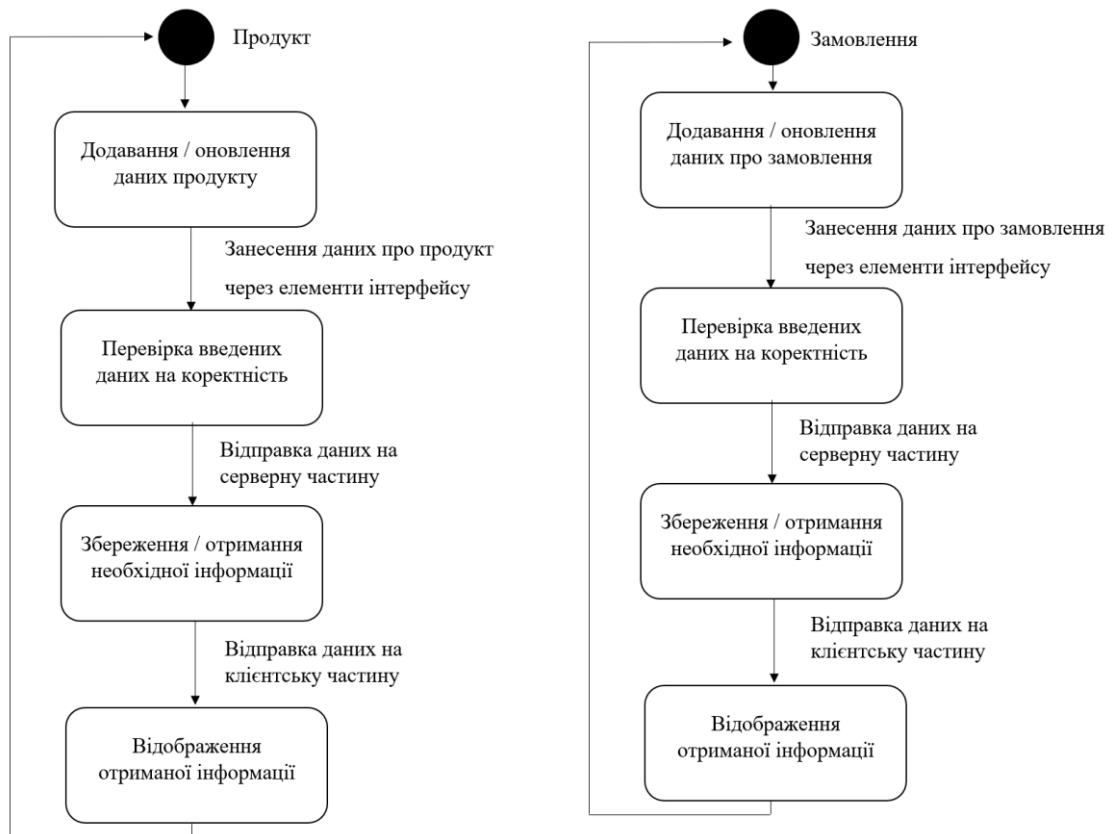


Рисунок 3.3 – Схема функціонування програмного забезпечення для автоматизації прийняття рішень для електронної комерції у вигляді діаграми станів

Оскільки основною метою розробки програми є автоматизація прийняття рішень для електронної комерції, було побудовано діаграму станів, яка відображає основні процеси програми: процес існування продуктів та процес обробки замовлень (рисунок 3.3).

Початковим станом для продукту є «Додавання або оновлення даних», де користувач через веб-інтерфейс вводить або змінює інформацію про продукт. Після цього продукт переходить у стан «Перевірка введених даних на коректність», який виконується на клієнтській стороні. На цьому етапі перевіряються:

- мінімальна довжина назви продукту;
- відсутність від'ємних або некоректних значень для параметрів, таких як ціна, об'єм, залишок.

Лише після успішного проходження цього етапу дані надсилаються на сервер для подальшої обробки.

На сервері продукт потрапляє у стан «Збереження або оновлення необхідної інформації», де:

- перевіряється існування продукту у разі оновлення даних;
- виконується конвертація даних у єдиний формат для збереження у базі даних.

Після успішної обробки сервер повертає дані на клієнтську сторону. Продукт переходить у стан «Відображення отриманої інформації», де дані виводяться у зручному для користувача форматі через веб-інтерфейс.

Процес оформлення замовлення починається зі стану «Додавання або оновлення даних», під час якого користувач через веб-інтерфейс заповнює всі необхідні параметри для замовлення. Після цього замовлення переходить у стан «Перевірка введених даних на коректність», який включає перевірку правильності та повноти заповнених полів.

У разі успішного проходження перевірки введені дані надсилаються на сервер, де замовлення переходить у стан «Збереження або оновлення необхідної інформації». На цьому етапі сервер:

- за необхідності звертається до бази даних для отримання додаткової інформації;
- виконує потрібні обчислення, необхідні для обробки конкретного замовлення.

Після завершення серверної обробки замовлення повертається на клієнтську сторону, де переходить у стан «Відображення отриманої інформації». Тут результати обчислень серверної частини відображаються у зручному для користувача форматі через веб-інтерфейс.

Ця діаграма станів дозволяє наочно продемонструвати послідовність операцій для двох ключових процесів програми, підкреслюючи важливість кожного етапу обробки даних.

Діаграма діяльності програмного забезпечення для автоматизації прийняття рішень для електронної комерції зображена на рисунку 3.4. Наявні такі види діяльності програмного забезпечення для автоматизації прийняття рішень для електронної комерції як: початок роботи з продуктом, додавання даних продукту, оновлення даних продукту, перевірка введених даних на коректність, відправка даних на сервер, збереження інформації, відображення отриманої інформації, початок роботи з замовленням, додавання нового замовлення, оновлення існуючого замовлення, отримання необхідної інформації.

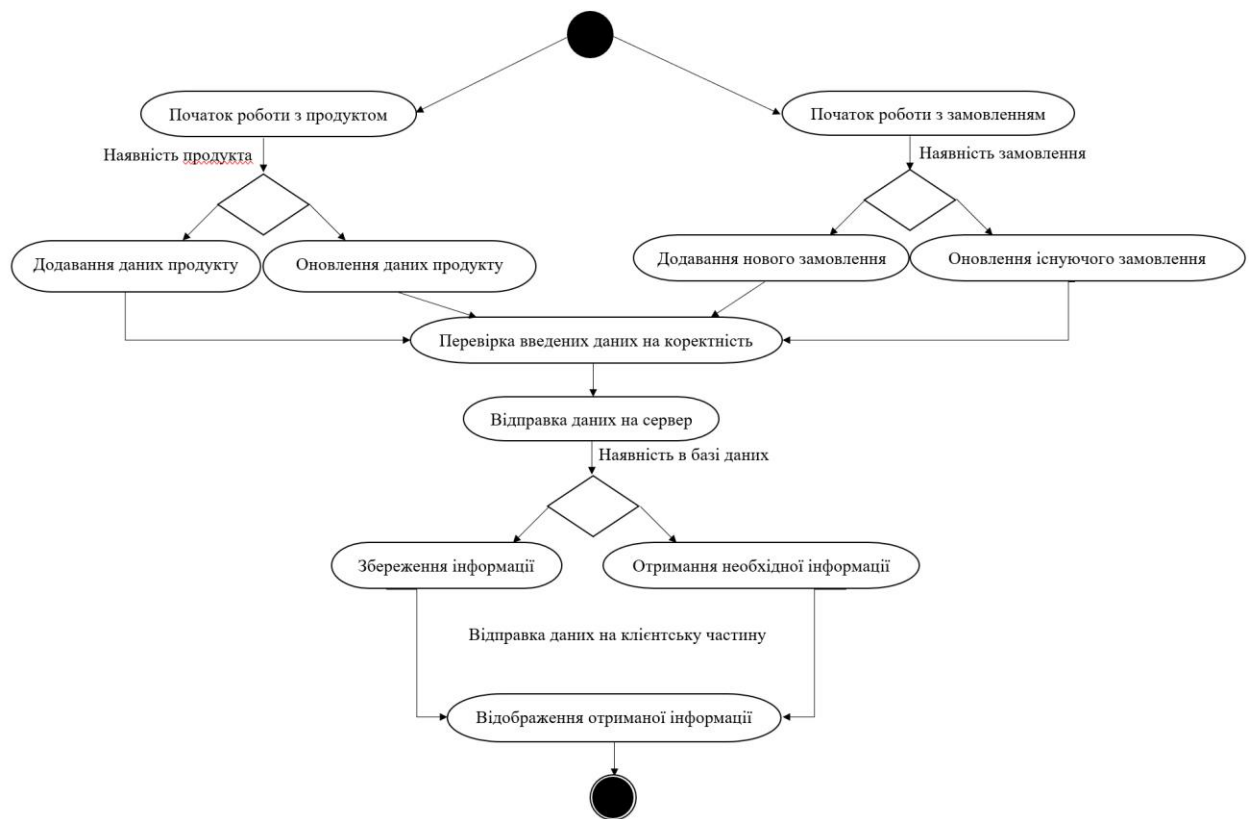


Рисунок 3.4 – Діаграма діяльності програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Діаграму кооперацій програмного забезпечення для автоматизації прийняття рішень для електронної комерції зображено на рисунку 3.5.



Рисунок 3.5 – Діаграма кооперацій програмного забезпечення для автоматизації прийняття рішень для електронної комерції

3.3 Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для автоматизації прийняття рішень для електронної комерції

Під час розробки інтерфейсу користувача для програмного забезпечення, призначеного для автоматизації прийняття рішень у сфері електронної комерції, були враховані всі вимоги технічного завдання. Приклади створених прототипів інтерфейсних форм представлені на рисунках 3.6 – 3.12.

Прототип головної сторінки програмного забезпечення наведено на рисунку 3.6. На цій сторінці реалізовані такі елементи інтерфейсу: логотип, навігаційне меню, кнопки управління, а також таблицю, яка відображає інформацію про залишки продуктів.

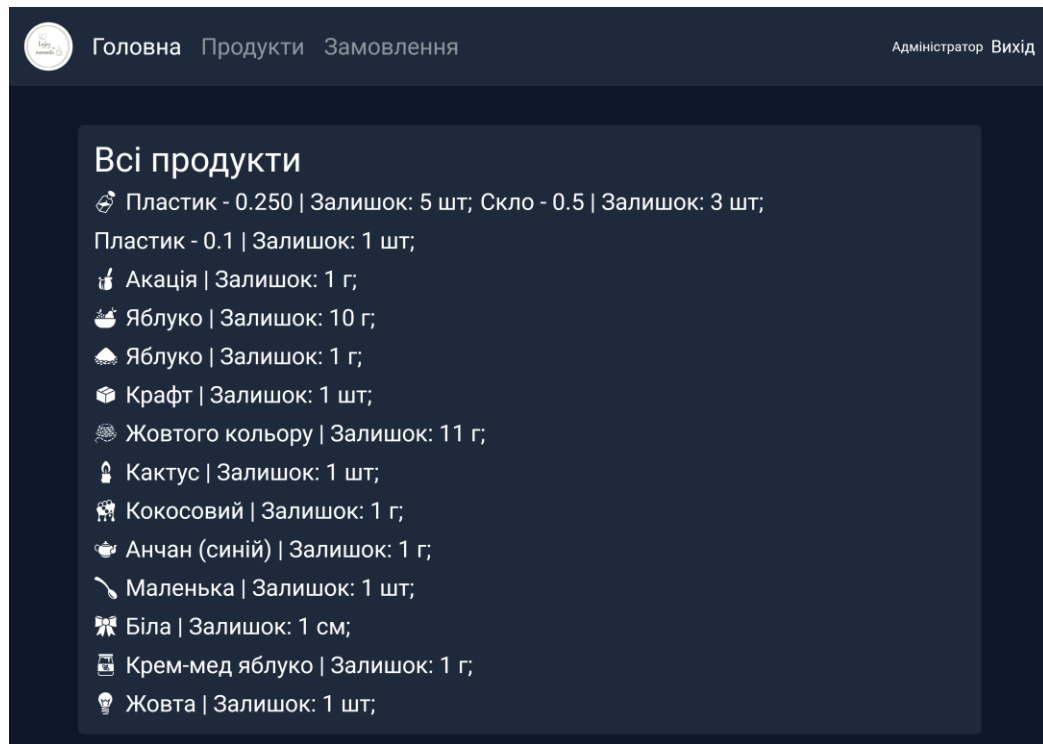


Рисунок 3.6 – Прототип головної сторінки програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Розроблено прототип сторінки авторизації користувача в системі, який представлено на рисунку 3.7. Для авторизації користувачу необхідно ввести логін, пароль і натиснути кнопку «Увійти». Прототип сторінки включає текстові підказки, поля для введення логіну та паролю, а також кнопку управління, що виконує функцію авторизації.

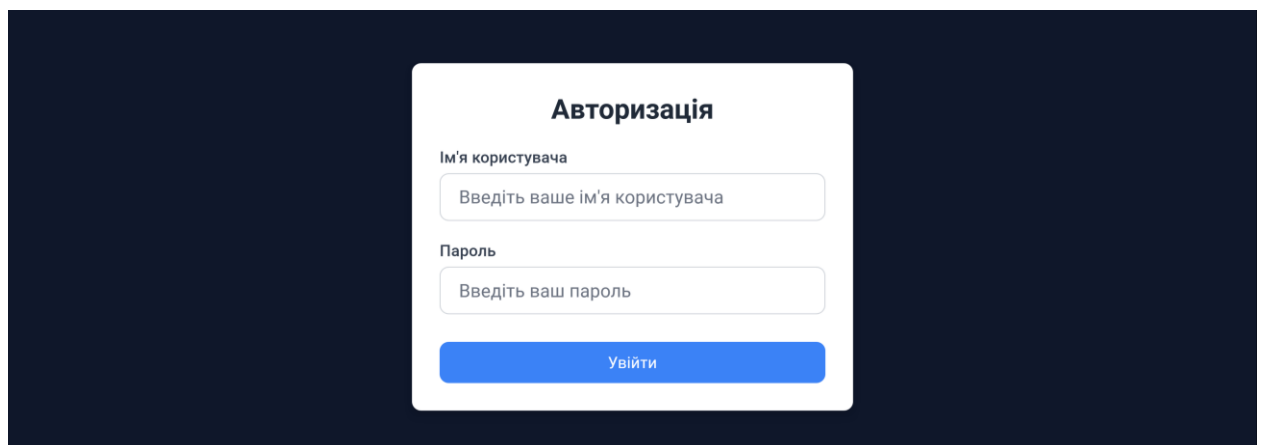


Рисунок 3.7 – Прототип сторінки авторизації

Прототип сторінки продукту представлено на рисунку 3.8. Ця сторінка відкривається, коли користувач на головній сторінці натискає на іконку поруч із відповідним пунктом у таблиці залишків або обирає продукт у випадаючому списку. Сторінка відображає всю інформацію про вибраний продукт і надає можливість редагувати, видаляти чи додавати новий запис.

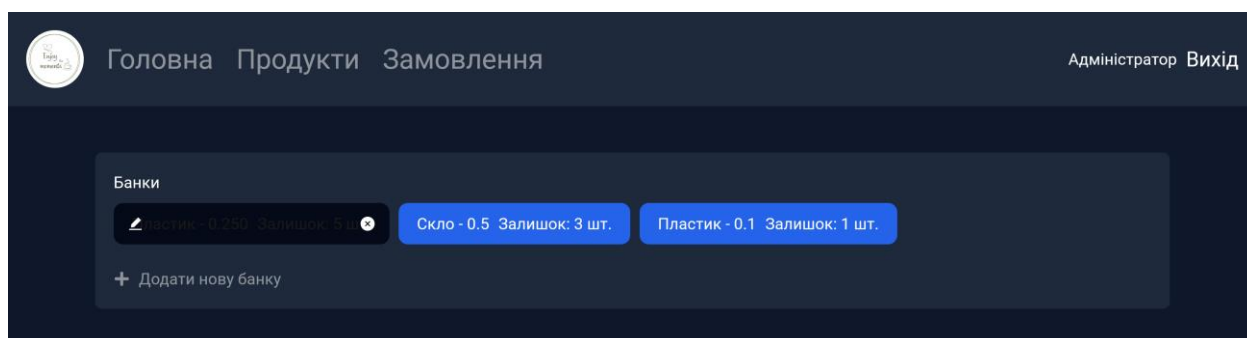


Рисунок 3.8 – Прототип сторінки продукту

При наведенні курсору на товар з'являється можливість його оновлення або видалення. Для процесів оновлення та додавання використовується модальне вікно, представлене на рисунку 3.9. У цьому вікні передбачені поля для редагування таких характеристик товару, як об'єм, собівартість, ціна продажу, залишок, а також поле вибору «Матеріал». Для керування діями передбачено кнопки «Зберегти», що відповідає за збереження внесених змін, та «Закрити», яка дозволяє закрити форму без збереження.

Прототип сторінки замовлення відображено на рисунку 3.10. Ця сторінка відкривається після натискання користувачем відповідного пункту меню в шапці сторінки. Відображає інформацію про всі замовлення і надає можливість редагувати, видаляти та додавати новий.

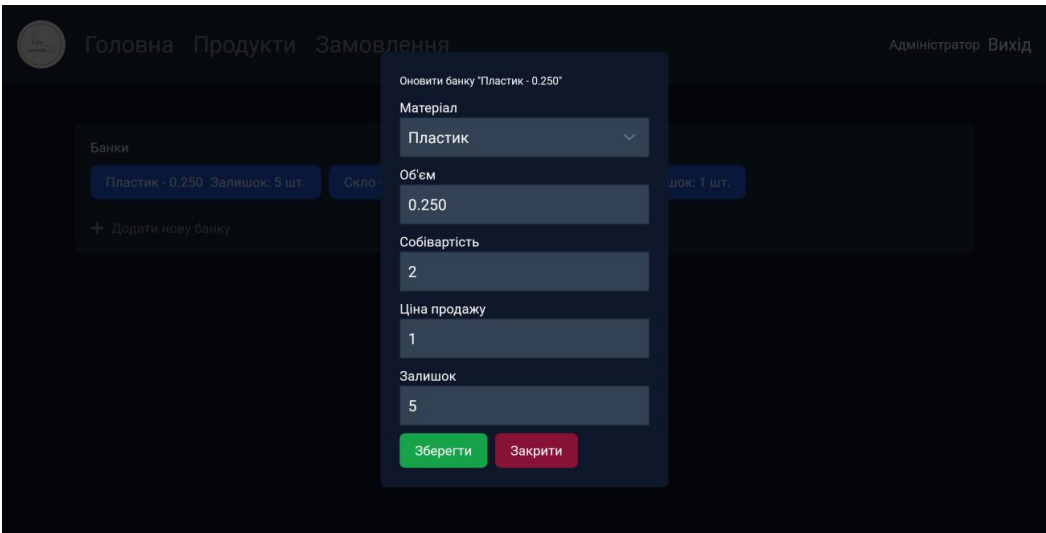


Рисунок 3.9 – Модальне вікно оновлення / додавання продукту

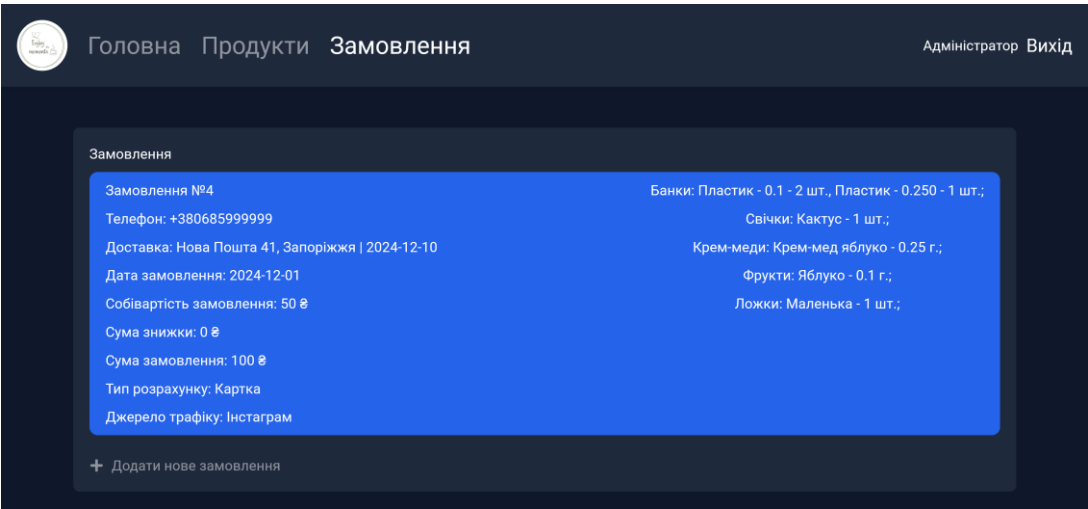


Рисунок 3.10 – Прототип сторінки замовлення

При наведенні на замовлення з’являється можливість його оновлення або видалення. Для виконання цих дій використовується модальне вікно, зображене на рисунках 3.11 та 3.12. У першому вікні (рисунок 3.11) представлені поля вибору: джерело трафіку, тип розрахунку, тип доставки та дата доставки. Також передбачені поля для редагування, зокрема: номер телефону для доставки та сума знижки. Інтерфейс включає кнопки керування: «Створити» для збереження внесених змін, «Закрити» для закриття форми без збереження, а також «Далі» для переходу до другого

вікна. У нижній частині форми розміщено контейнер для кожного виду продукту, який дозволяє деталізувати інформацію про склад замовлення.

Головна Продукти Замовлення

Додати нове замовлення

Джерело трафіку
Інстаграм

Тип розрахунку
Готівка

Тип доставки
Самовивіз

Номер телефону для доставки
+380660000000

Дата доставки
12 / 01 / 2024

Створити Закрити Далі →

Рисунок 3.11- Прототип модального вікна оновлення / додавання замовлення

Додати нове замовлення

Банки Мед Фрукти Коробки

Соломинки Свічки Чай Ложки

Стрічки Крем-меди Гірлянди

Сума знижки
0

Створити Закрити ← Назад

Рисунок 3.12 – Прототип модального вікна оновлення / додавання замовлення

3.4 Розширення функціональності програмного забезпечення для автоматизації прийняття рішень для електронної комерції за допомогою модулів

Для реалізації функціональних вимог, визначених у технічному завданні, під час розробки програмного забезпечення для автоматизації прийняття рішень в електронній комерції було застосовано спеціалізовані модулі. Зокрема, для забезпечення валідації даних використано бібліотеку Yup, яка стала основним інструментом для перевірки коректності введених даних на сторінках реєстрації та аутентифікації користувачів.

У процесі розробки використовувалися модулі, такі як `email()`, `required()`, `min()`, `matches()`, які дозволяють задавати обмеження для полів без необхідності ручного опису правил валідації. Наприклад, ці модулі автоматизують перевірку кількості символів у логіні, складності пароля тощо.

Застосування цих модулів забезпечило інтерактивне попередження користувачів про некоректно введені дані, а також запобігло обробці невірної інформації до моменту її повної відповідності заданим вимогам клієнтської частини. Такий підхід сприяв підвищенню зручності користування та надійності роботи системи.

На рисунку 3.13 продемонстровано дію механізму обмеження на поле введення, що реалізовано у програмному забезпеченні. Зокрема, наведено приклад, коли поле залишається незаповненим, і система генерує повідомлення про помилку, що свідчить про неприпустимість залишення поля порожнім. Такий функціонал є частиною процесу валідації даних, забезпечуючи контроль правильності введення інформації користувачем.

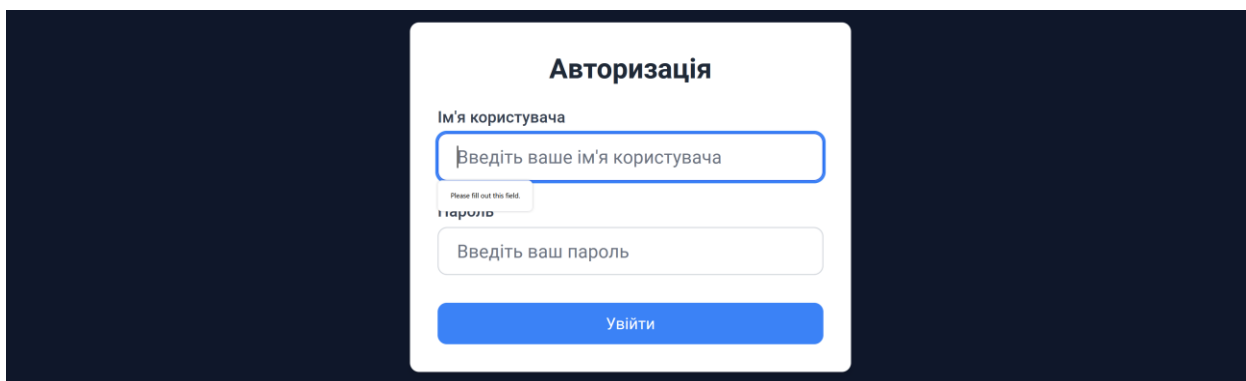


Рисунок 3.13 – Приклад обробки порожнього поля

На рисунку 3.14 представлено приклад обробки ситуації, коли користувач вводить некоректні дані у поле форми. Система реагує на помилку, відображаючи відповідне повідомлення про неправильність введеної інформації. Це є частиною функціоналу валідації даних, що сприяє забезпеченню коректності інформації, яка надходить до системи, та запобігає виникненню можливих помилок у подальшій обробці.

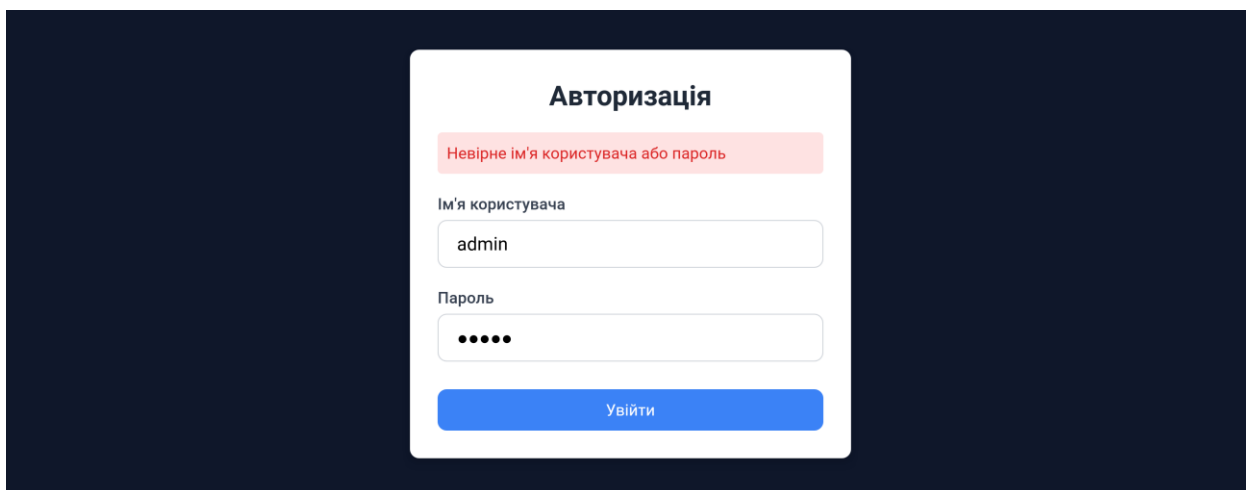


Рисунок 3.14 – Приклад обробки ситуації, пов'язаної з некоректним введенням даних

3.5 Висновки за розділом 3

У розділі описано структуру, функціонал і інтерфейс програмного забезпечення для автоматизації прийняття рішень у сфері електронної комерції. Розроблено трирівневу архітектуру, що складається з клієнтської частини, серверної частини та бази даних PostgreSQL, із використанням сучасних технологій, таких як React.js, Nest.js та TypeORM.

Функціонування програми охоплює перевірку даних на клієнтській стороні, їхню обробку сервером і взаємодію з базою даних. Розроблено схеми, які демонструють процеси створення, перевірки та збереження продуктів і замовлень.

Інтерфейс системи спроектовано з урахуванням вимог зручності для користувачів. Представлено прототипи сторінок і модальних вікон, які забезпечують легке управління інформацією про продукти й замовлення.

Для перевірки коректності даних використано бібліотеку Yup із заздалегідь визначеними модулями. Це спростило процес валідації та виключило можливість обробки некоректної інформації.

Розроблене програмне забезпечення відповідає вимогам технічного завдання та може ефективно використовуватися для автоматизації прийняття рішень для електронної комерції.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Ефективний діалог між учасниками процесу прийняття рішень у сфері електронної комерції є ключовим фактором для підвищення продуктивності та досягнення високих результатів. Автоматизація цього процесу, яка є основною метою розробленої програми, реалізована за допомогою сучасного програмного забезпечення, що забезпечує точність і швидкість прийняття рішень.

Клієнтська частина програми створена за допомогою JavaScript-бібліотеки React, яка має відкритий вихідний код, забезпечує високу продуктивність, підтримує компонентний підхід і є популярною серед розробників завдяки зручності у використанні. Для серверної частини використовувалася платформа Node.js разом із фреймворком Nest.js, що дозволяє створювати масштабовані та структуровані додатки. Розробка здійснювалася в середовищі Visual Studio Code, яке є зручним, універсальним і підтримує безліч інструментів для ефективної роботи. Програмне забезпечення оптимізоване для роботи на будь-якій операційній системі, забезпечуючи кросплатформеність та стабільність.

Програма для автоматизації прийняття рішень для електронної комерції забезпечує виконання таких функцій:

- можливість додавання нового продукту до системи;
- можливість обчислення доходу від замовлення;
- можливість додавання нового замовлення до системи;
- можливість оновлення наявних продуктів;
- можливість обчислення залишку продуктів;
- можливість оновлення наявних замовлень;
- можливість видалення продуктів із системи;

- можливість видалення замовлень із системи.

Система має розподілену архітектуру, яка включає клієнтську та серверну частини, між якими йде обмін даними. Клієнтська частина забезпечує зручний інтерфейс із мінімалістичним дизайном і зрозумілою навігацією, тоді як сервер обробляє запити та підтримує функціональність клієнта.

Для роботи клієнтської частини необхідний персональний комп'ютер з доступом до мережі Інтернет. Апаратна платформа повинна відповідати мінімальним системним вимогам веб-браузера, який використовується для роботи із системою.

Програмне забезпечення є кросплатформним і стабільно працює на різних операційних системах.

Мінімальні вимоги для операційної системи Windows:

- Windows XP (2 пакет оновлень) або більш пізньої версії;
- процесор Intel i3 та аналогічні з підтримкою SSE2 [14].

Рекомендовані вимоги для Windows:

- Windows 7, Windows 8, Windows 8.1, Windows 10 або більш пізньої версії;
- процесор Core i5 2-го покоління (від 2 ГГц), Core i5 3-го або 4-го покоління чи аналогічні.

Мінімальні вимоги для операційної системи macOS:

- macOS X Yosemite 10.09 або більш пізньої версії;
- процесор Intel Core 2 Duo або аналогічний із підтримкою SSE2.

Рекомендовані вимоги для macOS:

- macOS X 10.14 (Mojave) або більш нові версії;
- Процесор Intel Core i5 2-го покоління або новіший.

Мінімальні вимоги для операційної системи Linux:

- Ubuntu 14.04 (64-розрядної версії) або старше, Debian 8 або старше, openSUSE 13.3 або старше, Fedora Linux 24 або старше;

- процесор Intel Pentium 4 або старше з підтримкою SSE2 [15].

Для коректної роботи серверної частини необхідно дотримуватися таких мінімальних умов:

- CPU: Intel Core i5;
- RAM: 2 Gb;
- HDD/SSD: 1Gb
- WEB-сервер: Apache 2 або Nginx;
- база даних: PostgreSQL;
- порти: 80 (HTTP), 5432 (стандартний порт PostgreSQL);
- програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn.

Рекомендовані апаратні характеристики покращують продуктивність системи та дозволяють обробляти більші обсяги даних у реальному часі.

Функціональні характеристики розробленого програмного забезпечення для автоматизації прийняття рішень у сфері електронної комерції детально описані в технічному завданні (Додаток А). У додатку Б наведено фрагменти коду програмного забезпечення, що реалізує ключові функції автоматизації прийняття рішень.

4.2 Характеристики програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Програмне забезпечення для автоматизації прийняття рішень у сфері електронної комерції здатне працювати безперервно, відповідно до потреб користувача. Для його функціонування необхідна встановлена операційна система, а також фреймворки React і Nest.js. Для забезпечення стабільної роботи необхідно розгорнути сервер, налаштувати програму та базу даних застосунку.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Користувачі взаємодіють із програмним забезпеченням для автоматизації прийняття рішень у сфері електронної комерції через стандартні засоби операційної системи.

Для запуску клієнтської частини проєкту необхідно:

- відкрити термінал і перейти до каталогу клієнтського проєкту, де розташований файл `package.json`;

- виконати команду `npm run dev`;

Щоб запустити серверну частину проєкту слід виконати наступні дії:

- за допомогою терміналу перейти до каталогу серверного проєкту, в якому міститься файл `package.json`;

- ввести команду `npm run start:dev`;

Після цього клієнтська та серверна частини будуть готові до роботи.

4.3.2 Вхідні й вихідні дані

Вхідними даними програмного забезпечення для автоматизації прийняття рішень для електронної комерції є інформація про продукти і замовлення.

Вихідними даними програмного забезпечення для автоматизації прийняття рішень для електронної комерції є створені, оновлені або відредаговані записи про продукти та замовлення.

4.3.3 Повідомлення

Користувач програмного забезпечення для автоматизації прийняття рішень у сфері електронної комерції може отримувати різноманітні повідомлення, серед яких:

- повідомлення про необхідність заповнення обов'язкового поля, якщо введення даних у це поле є обов'язковим для завершення операції;
- повідомлення про некоректність введених даних, якщо введені значення не відповідають встановленим вимогам або форматам;
- повідомлення про успішне виконання операції, наприклад, успішне створення нового продукту або замовлення.

Ці повідомлення забезпечують зручність у роботі з програмою та допомагають користувачеві виправляти помилки під час введення даних.

4.4 Виконання програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Для використання сайту необхідно перейти за відповідним посиланням, при цьому з'явиться сторінка авторизації, зображена на рисунку 4.1.

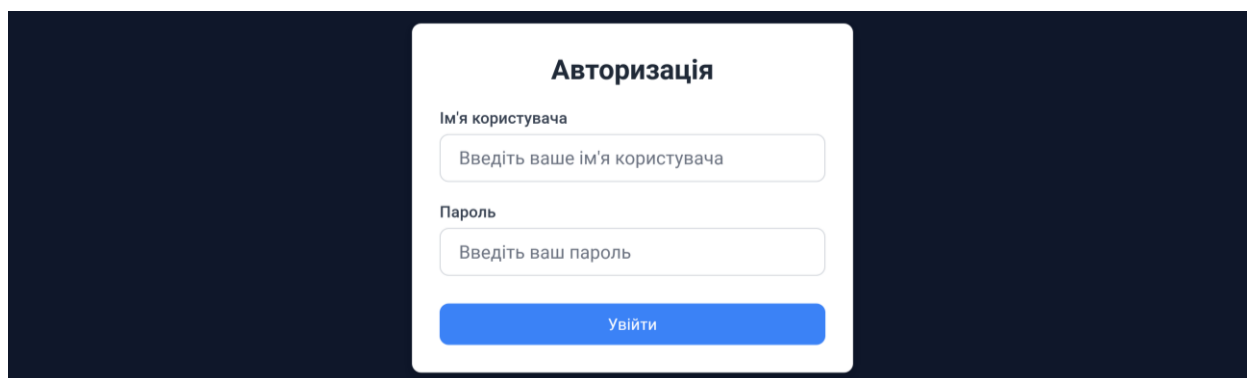


Рисунок 4.1 – Інтерфейс форми для входу у програмне забезпечення для автоматизації прийняття рішень для електронної комерції

Після успішної авторизації відбувається перехід на головну сторінку сайту, де розміщені усі продукти. Сторінку зображено на рисунку 4.2.

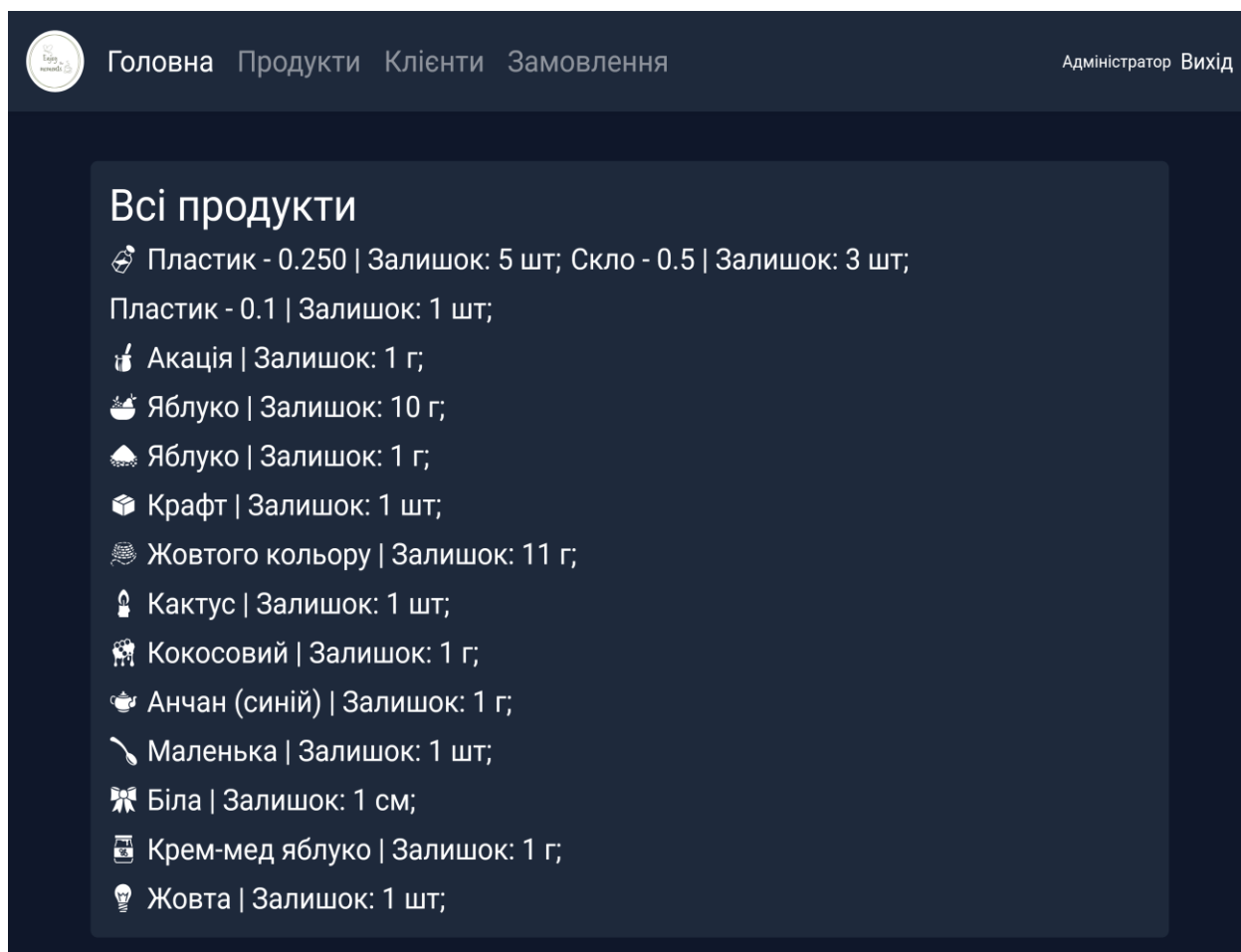


Рисунок 4.2 – Головна сторінка програмного забезпечення для автоматизації прийняття рішень для електронної комерції

На головній сторінці користувачі мають можливість переходити до перегляду деталей конкретного продукту або замовлення за допомогою відповідних кнопок, розміщених у верхній частині сторінки. Окрім цього, на сторінці представлено таблицю із залишками всіх доступних продуктів, де для кожного товару передбачено окрему кнопку у вигляді іконки, яка дозволяє перейти до сторінки з детальною інформацією про цей продукт.

На рисунку 4.3 представлено приклад сторінки конкретного продукту, а також показано ефект, який виникає при наведенні на продукт із метою його оновлення або видалення.

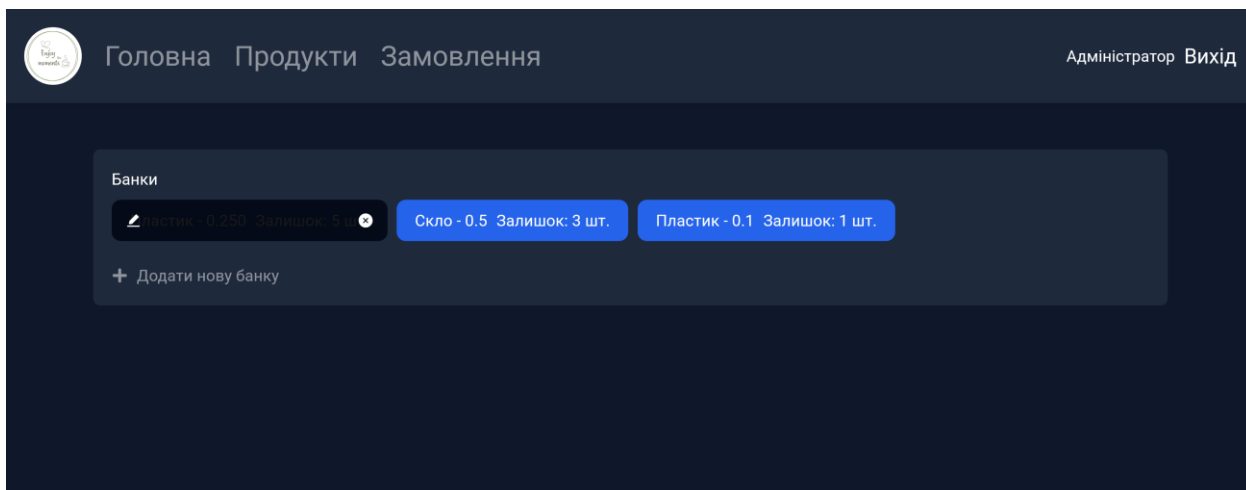


Рисунок 4.3 – Сторінка окремого продукту

Приклад модального вікна, яке використовується для оновлення або додавання продукту, наведено на рисунку 4.4.

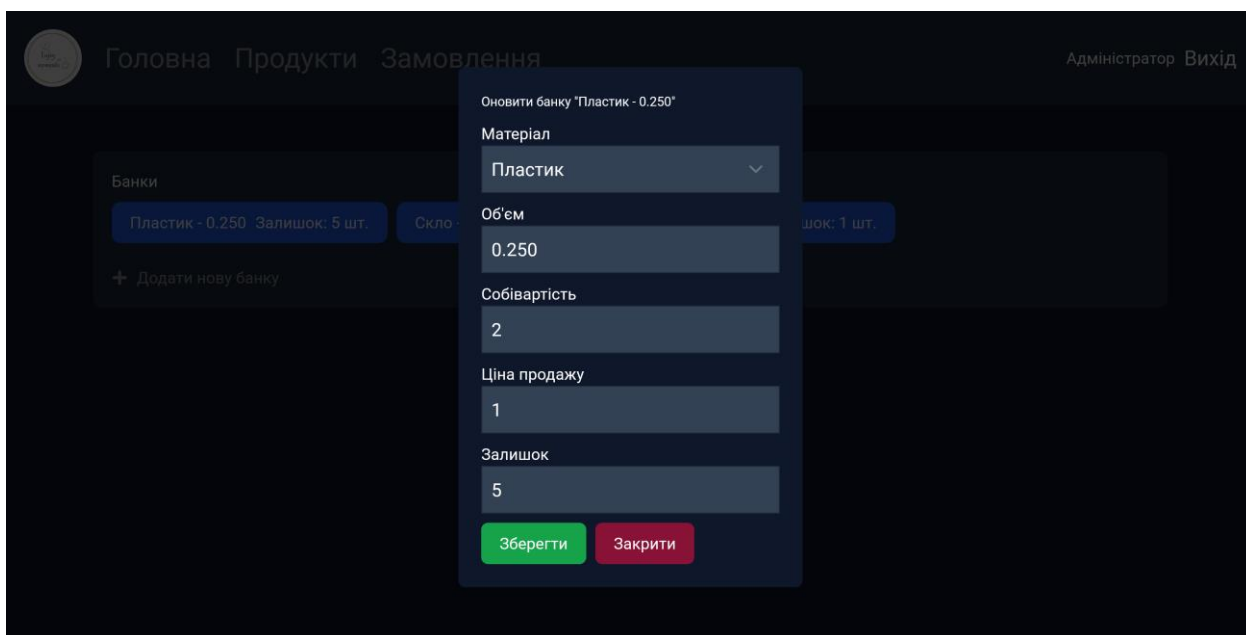


Рисунок 4.4 – Модальне вікно для оновлення або додавання продукту

На рисунку 4.5 представлено приклад сторінки, яка містить перелік усіх замовлень із відображенням необхідної інформації про кожне з них. За потреби користувач може навести на конкретне замовлення, щоб оновити його дані або видалити.

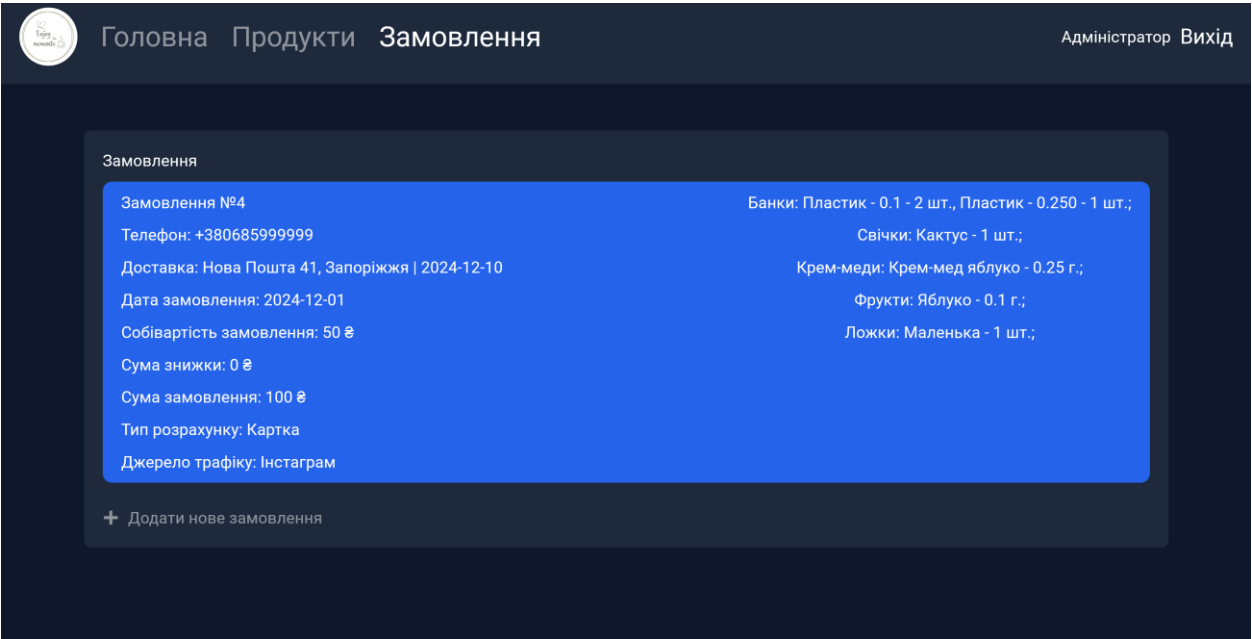


Рисунок 4.5 – Сторінка всіх замовлень

На рисунку 4.6 показано приклад модального вікна для додавання нового замовлення. Процес додавання складається з двох етапів. На першому етапі користувач заповнює інформацію про клієнта, обирає тип доставки, спосіб оплати, джерело трафіку та дату доставки (рисунок 4.6).

Головна Продукти Замовлення

Адміністратор Вихід

Додати нове замовлення

Джерело трафіку
Instagram

Тип розрахунку
Картка

Тип доставки
Самовивіз

Номер телефону для доставки
+380660000000

Дата доставки
11 / 25 / 2024

Створити Закрити Далі →

Рисунок 4.6 – 1 етап додавання нового замовлення

На другому етапі користувач обирає необхідні продукти для додавання до замовлення та, за потреби, заповнює поле «Сума знижки». Після цього, натисканням кнопки «Створити», відправляється запит на сервер для створення нового замовлення. На сервері під час обробки запиту автоматично обчислюються собівартість і дохід замовлення, а також оновлюється залишок кожного товару, включеного в замовлення.

Додати нове замовлення

Банки Мед Фрукти Коробки

Соломинки Свічки Чай Ложки

Стрічки Крем-меди Гірлянди

Сума знижки
0

Створити Закрити ← Назад

Рисунок 4.7 – 2 етап додавання нового замовлення

4.5 Тестування програмного забезпечення для автоматизації прийняття рішень для електронної комерції

Розроблене програмне забезпечення для автоматизації прийняття рішень у сфері електронної комерції створене на основі кросплатформених технологій, що дозволило перевірити його функціонування на різних операційних системах.

У ході тестування було проведено перевірку відповідності функціональних вимог, зокрема:

- додавання нового продукту до системи;
- розрахунок доходу від замовлення;
- додавання нового замовлення до системи;
- оновлення інформації про наявні продукти;
- розрахунок залишків продуктів;
- оновлення даних існуючих замовлень;
- видалення продуктів із системи;
- видалення замовлень із системи.

Окрім цього, було протестовано інші аспекти відповідно до технічного завдання.

Результати тестування підтвердили, що програмне забезпечення повністю відповідає вимогам технічного завдання.

4.6 Висновки за розділом 4

У роботі описано програмне забезпечення для автоматизації прийняття рішень у сфері електронної комерції. Проведено тестування розробленої системи, під час якого перевірено її функціональність та відповідність технічному завданню. Результати тестування підтвердили, що

програма відповідає заявленим вимогам і може успішно використовуватися за призначенням.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи було проведено дослідження і розробку програмного забезпечення, орієнтованого на автоматизацію прийняття рішень у сфері електронної комерції. Досліджені існуючі програмні рішення, такі як KeyCRM, Prom та Microsoft Dynamics 365, підтвердили актуальність впровадження систем автоматизації для покращення бізнес-процесів, зокрема оптимізації замовлень, інвентаризації, обробки клієнтських даних і розрахунків прибутковості.

Аналіз проблеми показав, що сучасні підприємства, які працюють у сфері електронної комерції, стикаються зі значними обсягами інформації, котрі потребують своєчасної та точної обробки. Застосування інструментів автоматизації дає змогу мінімізувати вплив людського фактора, знижувати ризик помилок, підвищувати швидкість і точність операцій. Це, у свою чергу, підвищує конкурентоспроможність бізнесу на сучасному динамічному ринку.

Створене програмне забезпечення базується на стеку сучасних технологій. Для клієнтської частини використано React.js, TypeScript та Tailwind CSS, що забезпечує високу продуктивність, гнучкість і легкість у розробці інтерфейсів. Серверна частина реалізована за допомогою Nest.js та TypeORM, а для зберігання даних застосовано реляційну базу даних PostgreSQL. Застосування архітектури RESTful API спростило інтеграцію з іншими сервісами і надало системі гнучкості при подальшому масштабуванні функціональності. Типізація коду та ретельна валідація даних сприяли підвищенню надійності, а модульність і компонентність забезпечили легкість у підтримці та розвитку продукту.

Інтерфейс користувача було розроблено з фокусом на доступність та інтуїтивність. Проектування прототипів сторінок і модальних вікон, впровадження механізмів валідації та повідомлень про помилки покращили

взаємодію користувача з системою та зменшили кількість некоректних операцій.

Проведене тестування програмного забезпечення підтвердило відповідність функціоналу технічному завданню. Система коректно виконує додавання, оновлення та видалення продуктів і замовлень, розрахунок прибутковості, аналіз залишків, а також забезпечує необхідний рівень безпеки й стабільності. Перевірка на різних платформах і дотримання апаратних вимог засвідчили кросплатформеність та масштабованість рішення.

Таким чином, у межах дипломної роботи створено ефективне програмне забезпечення, що може бути використане у реальних умовах електронної комерції. Завдяки комплексному підходу – від аналізу предметної області до тестування та вдосконалення функціональності – отримана система сприятиме підвищенню ефективності бізнес-процесів, зниженню витрат часу і ресурсів, покращенню якості обслуговування клієнтів і, зрештою, підвищенню конкурентоспроможності підприємств на сучасному ринку електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. KeyCRM. Опис продукту [Електронний ресурс]. – Режим доступу: <https://ua.keycrm.app/product>.
2. Prom.ua. Інформація [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Prom.ua>.
3. Microsoft Dynamics 365. Опис функцій [Електронний ресурс]. – Режим доступу: <https://progresia.online/uk/microsoft-dynamics-365/sales>.
4. JavaScript та його екосистема [Електронний ресурс]. – Режим доступу: <https://dou.ua/forums/topic/34739/>.
5. HTML & CSS: Design and Build Web Sites / J. Daket. – New York: Spring, 2011. – 55 с.
6. TypeScript [Електронний ресурс]. – Режим доступу: <https://en.wikipedia.org/wiki/TypeScript>.
7. Офіційна документація TypeScript [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/>.
8. Міграція з Webpack на Vite [Електронний ресурс]. – Режим доступу: <https://devzone.org.ua/post/posibnyk-po-mihratsiyi-z-webpack-na-vite>.
9. Використання Frontend у проєктах [Електронний ресурс]. – Режим доступу: <https://habr.com/companies/kuper/articles/737474/>.
10. Як вивчити JavaScript [Електронний ресурс]. – Режим доступу: <https://dou.ua/lenta/articles/how-to-learn-javascript/?hl=ru>.
11. IntelliSense. Інформація [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/IntelliSense>.
12. Visual Studio Code. Офіційна сторінка [Електронний ресурс]. – Режим доступу: <https://code.visualstudio.com/>.
13. Обговорення IntelliSense [Електронний ресурс]. – Режим доступу: <https://dou.ua/forums/topic/35210/>.
14. Таня Раська. Налаштування середовища React у Windows

[Електронний ресурс]. – Режим доступу: <https://www.taniarascia.com/getting-started-with-react/>

15. Олександр Шевченко. Налаштування середовища React на Ubuntu [Електронний ресурс]. – Режим доступу: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу ведення та обробки електронної звітності.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація засобів автоматизації прийняття рішень для електронної комерції», затверджене наказом Національного університету «Запорізька політехніка» № 398 від 07 жовтня 2024 р.

A.2 Призначення розробки

Програмний продукт призначений для автоматизації прийняття рішень для електронної комерції.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для автоматизації прийняття рішень для електронної комерції забезпечує виконання таких функцій:

- можливість додавання нового продукту до системи;
- можливість обчислення доходу від замовлення;
- можливість додавання нового замовлення до системи;
- можливість оновлення наявних продуктів;
- можливість обчислення залишку продуктів;
- можливість оновлення наявних замовлень;

- можливість видалення продуктів із системи;
- можливість видалення замовлень із системи;

A.3.2 Вимоги до надійності

Інтерфейс програмного забезпечення для автоматизації прийняття рішень для електронної комерції повинен бути зручним для користувачів.

A.3.3 Вимоги до надійності

Програмне забезпечення для автоматизації прийняття рішень для електронної комерції повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для автоматизації прийняття рішень для електронної комерції необхідна наявність персонального комп'ютера у користувача.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи клієнтської частини необхідний персональний комп'ютер з доступом до мережі Інтернет. Апаратна платформа повинна відповідати мінімальним системним вимогам веб-браузера, який використовується для роботи із системою.

Мінімальні вимоги для Windows:

- Windows XP (2 пакет оновлень) або більш пізньої версії;
- процесор Intel i3 та аналогічні з підтримкою SSE2 [14].

Мінімальні вимоги для macOS:

- OS X Yosemite 10.09 або більш пізньої версії.

Мінімальні вимоги для Linux:

- Ubuntu 14.04, Debian 8, openSUSE 13.3, Fedora Linux 24;
- процесор Intel Pentium 4 або старше з підтримкою SSE2 [15].

Для коректної роботи серверної частини необхідно дотримуватися таких мінімальних умов:

- CPU: Intel Core i5;
- RAM: 2 Gb;
- HDD/SSD: 1Gb
- WEB-сервер: Apache 2 або Nginx;
- база даних: PostgreSQL;
- порти: 80 (HTTP), 5432 (стандартний порт PostgreSQL);
- програмне забезпечення: середовище Node.js.

A.3.6 Вимоги до маркування і пакування

Програма для автоматизації прийняття рішень для електронної комерції може бути записана на будь-якому носії інформації. На пакуванні повинна бути назва програми – «Програмне забезпечення для автоматизації прийняття рішень для електронної комерції».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для автоматизації прийняття рішень для електронної комерції є інформація про продукти і замовлення.

Вихідними даними програмного забезпечення для автоматизації прийняття рішень для електронної комерції є створені, оновлені або відредаговані записи про продукти та замовлення.

ДОДАТОК Б
Фрагмент тексту програми

Компонент для відображення даних про продукт

```
import { FC, useState } from "react";
import { AiFillCloseCircle, AiFillEdit } from "react-icons/ai";
import { FaPlus } from "react-icons/fa";
import { Form, useLoaderData } from "react-router-dom";
import { instance } from "../api/axios.api";
import { IBank } from "../types/types";
import BankModal from "../components/BankModal";

export const banksAction = async ({ request }: any) => {
  switch (request.method) {
    case "POST": {
      const formData = await request.formData();
      const newBank = {
        volume: formData.get("volume"),
        costPrice: formData.get("costPrice"),
        sellingPrice: formData.get("sellingPrice"),
        material: formData.get("material"),
        remainder: formData.get("remainder"),
      };

      await instance.post("/banks", newBank);
      return null;
    }
    case "PATCH": {
      const formData = await request.formData();
      const bank = {
        id: formData.get("id"),
        volume: formData.get("volume"),
```

```

        costPrice: formData.get("costPrice"),
        sellingPrice: formData.get("sellingPrice"),
        material: formData.get("material"),
        remainder: formData.get("remainder"),
    };

    await instance.patch(`/banks/${bank.id}`, bank);
    return null;
}

case "DELETE": {
    const formData = await request.formData();
    const bankId = formData.get("id");
    await instance.delete(`/banks/${bankId}`);

    return null;
}

};

export const bankLoader = async () => {
    const { data } = await instance.get<IBank[]>("/banks");
    return data;
};

const Banks: FC = () => {
    const banks = useLoaderData() as IBank[];
    const [bank, setBank] = useState<IBank | null>(null);
    const [isEdit, setIsEdit] = useState<boolean>(false);

```

```

const [visibleModal, setVisibleModal] = useState<boolean>(false);

return (
  <>
    <div className="mt-10 rounded-md bg-slate-800 p-4">
      <h1>Банки</h1>
      <div className="mt-2 flex flex-wrap items-center gap-2">
        {banks.length > 0
          ? banks.map((bank, idx) => (
              <div
                key={`${bank.volume}-${bank.id}-${idx}`}
                className="group relative flex items-center gap-2 rounded-lg
bg-blue-600 px-4 py-2"
              >
                {`${bank.material} - ${bank.volume}`}
                <span>{'\nЗалишок: ${bank.remainder} шт.'}</span>
                <div className="absolute bottom-0 left-0 right-0 top-0 hidden
items-center justify-between rounded-lg bg-black/90 px-3 group-hover:flex">
                  <button
                    onClick={() => {
                      setVisibleModal(true);
                      setBank(bank);
                      setIsEdit(true);
                    }}
                  >
                    <AiFillEdit />
                  </button>

                  <Form className="flex" method="delete" action="/banks">

```

```

        <input type="hidden" name="id" value={bank.id} />
        <button type="submit">
            <AiFillCloseCircle />
        </button>
    </Form>
</div>
</div>
))
: "Жодної банки не додано"}
</div>
<button
    onClick={() => {
        setVisibleModal(true);
    }}
    className="max-w-fit flex items-center gap-2 text-white/50 mt-5
hover:text-white"
>
    <FaPlus />
    <span>Додати нову банку</span>
</button>
</div>

{visibleModal && (
    <BankModal
        type="post"
        setVisibleModal={setVisibleModal}
        setProduct={setBank}
    />
)}

```

```

    {visibleModal && isEdit && bank !== null && (
      <BankModal
        type="patch"
        product={bank}
        setVisibleModal={setVisibleModal}
        setIsEdit={setIsEdit}
        setProduct={setBank}
      />
    )}
  </>
);
};

export default Banks;

```

Компонент модального вікна для додавання та оновлення продуктів

```

import { FC, useState, useEffect } from "react";
import { Form } from "react-router-dom";
import { IBank } from "../types/types";

interface IBankModal {
  type: "post" | "patch";
  product?: IBank;
  setVisibleModal: (visible: boolean) => void;
  setIsEdit?: (isEdit: boolean) => void;
  setProduct: (product: IBank | null) => void;
}

```

```
const materialOptions = [
  { value: "", label: "Оберіть матеріал" },
  { value: "Скло", label: "Скло" },
  { value: "Пластик", label: "Пластик" },
];
```

```
const BankModal: FC<IBankModal> = ({
  type,
  product,
  setVisibleModal,
  setIsEdit,
  setProduct,
}) => {
  const [volume, setVolume] = useState<number | string>(product?.volume
|| "");

  const [costPrice, setCostPrice] = useState(product?.costPrice || "");
  const [sellingPrice, setSellingPrice] = useState(product?.sellingPrice || "");
  const [material, setMaterial] = useState(product?.material || "");
  const [remainder, setRemainder] = useState<string | number>(
    product?.remainder || ""
  );

  const handleInputChange = (
    event: any,
    setter: React.Dispatch<React.SetStateAction<string | number>>
  ) => {
    let value = event.target.value;
```

```
value = value.replace(/,/g, ".");
```

```
const dotCount = (value.match(/\./g) || []).length;
if (dotCount > 1 && event.target.selectionStart) {
  value =
    value.slice(0, event.target.selectionStart - 1) +
    value.slice(event.target.selectionStart);
}
```

```
value = value.replace(/^[^0-9.]/g, "");
```

```
setter(value);
};
```

```
useEffect(() => {
  if (product) {
    setVolume(product.volume || "");
    setCostPrice(product.costPrice || "");
    setSellingPrice(product.sellingPrice || "");
    setMaterial(product.material || "");
    setRemainder(product.remainder || "");
  }
}, [product]);
```

```
return (
  <div className="fixed bottom-0 left-0 right-0 top-0 flex h-full w-full
items-center justify-center bg-black/50">
    <Form
      action="/banks"
```

```

method={type}
className="grid w-[300px] gap-2 rounded-md bg-slate-900 p-5"
onSubmit={() => {
  setVisibleModal(false);
  setProduct(null);
}}
>
<small>
  {product
    ? `Оновити банку "${product.material} - ${product.volume}"`
    : "Додати нову банку"}
</small>

<label htmlFor="material">
  Матеріал
  <select
    className="input w-full"
    name="material"
    value={material}
    onChange={(e) => setMaterial(e.target.value)}
  >
    {materialOptions.map((option) => (
      <option key={option.value} value={option.value}>
        {option.label}
      </option>
    ))}
  </select>
</label>

```

```
<label htmlFor="volume">
  Об'єм
  <input
    className="input w-full"
    type="text"
    name="volume"
    placeholder="Введіть об'єм"
    value={volume}
    onChange={e => handleInputChange(e, setVolume)}
  />
</label>
```

```
<label htmlFor="costPrice">
  Собівартість
  <input
    className="input w-full"
    type="text"
    name="costPrice"
    placeholder="Введіть собівартість"
    value={costPrice}
    onChange={e => handleInputChange(e, setCostPrice)}
  />
</label>
```

```
<label htmlFor="sellingPrice">
  Ціна продажу
  <input
    className="input w-full"
    type="text"
```

```

      name="sellingPrice"
      placeholder="Введіть ціну продажу"
      value={sellingPrice}
      onInput={(e) => handleInputChange(e, setSellingPrice)}
    />
  </label>

```

```

  <label htmlFor="remainder">
    Залишок
    <input
      className="input w-full"
      type="text"
      name="remainder"
      placeholder="Введіть залишок"
      value={remainder}
      onInput={(e) => handleInputChange(e, setRemainder)}
    />
  </label>

```

```

  <input type="hidden" value={product?.id} name="id" />

```

```

  <div className="flex items-center gap-2">
    <button type="submit" className="btn btn-green">
      {type === "patch" ? "Зберегти" : "Створити"}
    </button>
    <button
      className="btn btn-red"
      onClick={() => {
        setVisibleModal(false);
        if (product !== null && setIsEdit) {

```

```

        setIsEdit(false);
    }
  }}
  >
    Закрити
  </button>
</div>
</Form>
</div>
);
};

```

```
export default BankModal;
```

Компонент головної сторінки

```

import { FC } from "react";
import { instance } from "../api/axios.api";
import { NavLink, useLoaderData } from "react-router-dom";
import { IBank, ICream, IProduct } from "../types/types";
import {
  GiSquareBottle,
  GiHoneypot,
  GiFruitBowl,
  GiPowder,
  GiCardboardBoxClosed,
  GiRopeCoil,
  GiCandleLight,
  GiDrippingHoney,
  GiTeapot,

```

```

    GiSpoon,
    GiBowTieRibbon,
    GiHoneyJar,
    GiLightBulb,
  } from "react-icons/gi";

```

```

export const homeLoader = async () => {
  const banks = await instance.get("/banks");
  const honey = await instance.get("/honey");
  const fruit = await instance.get("/fruit");
  const powder = await instance.get("/powder");
  const box = await instance.get("/box");
  const straw = await instance.get("/straw");
  const candle = await instance.get("/candle");
  const wax = await instance.get("/wax");
  const tea = await instance.get("/tea");
  const spoon = await instance.get("/spoon");
  const ribbon = await instance.get("/ribbon");
  const cream = await instance.get("/cream");
  const garland = await instance.get("/garland");

  return {
    banks: banks.data,
    honey: honey.data,
    fruit: fruit.data,
    powder: powder.data,
    box: box.data,
    straw: straw.data,
    candle: candle.data,

```

```

    wax: wax.data,
    tea: tea.data,
    spoon: spoon.data,
    ribbon: ribbon.data,
    cream: cream.data,
    garland: garland.data,
  };
};

```

```

const Home: FC = () => {
  const products = useLoaderData() as {
    banks: IBank[];
    honey: IProduct[];
    fruit: IProduct[];
    powder: IProduct[];
    box: IProduct[];
    straw: IProduct[];
    candle: IProduct[];
    wax: IProduct[];
    tea: IProduct[];
    spoon: IProduct[];
    ribbon: IProduct[];
    cream: ICream[];
    garland: IProduct[];
  };
  return (
    <div className="mt-10 rounded-md bg-slate-800 p-4">
      <h1 className="text-4xl">Всі продукти</h1>
      <div className="mt-2 flex flex-wrap items-center gap-2">

```

```

<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/banks">
    <GiSquareBottle size={25} title="Banks" />
  </NavLink>
  {products.banks.length > 0
    ? products.banks.map((product, idx) => (
      <div
        key={` ${product.volume}-${product.id}-${idx}`}
        >{` ${product.material} - ${product.volume} | Залишок:
${product.remainder} шт;`} </div>
      ))
    : "Жодної банки не додано"}
</div>

<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/honey">
    <GiHoneyPot size={25} title="Honey" />
  </NavLink>
  {products.honey.length > 0
    ? products.honey.map((product, idx) => (
      <div
        key={` ${product.name}-${product.id}-${idx}`}
        >{` ${product.name} | Залишок: ${product.remainder}
г;`} </div>
      ))
    : "Жодного меду не додано"}
</div>

<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/fruit">
    <GiFruitBowl size={25} title="Fruits" />

```

```

</NavLink>
{products.fruit.length > 0
  ? products.fruit.map((product, idx) => (
    <div
      key={`${product.name}-${product.id}-${idx}`}
      >{`${product.name} | Залишок: ${product.remainder}
r;`}</div>
    ))
  : "Жодного фрукту не додано"}
</div>
<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/powder">
    <GiPowder size={25} title="Powders" />
  </NavLink>
  {products.powder.length > 0
    ? products.powder.map((product, idx) => (
      <div
        key={`${product.name}-${product.id}-${idx}`}
        >{`${product.name} | Залишок: ${product.remainder}
r;`}</div>
      ))
    : "Жодного порошку не додано"}
</div>
<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/box">
    <GiCardboardBoxClosed size={25} title="Boxes" />
  </NavLink>
  {products.box.length > 0
    ? products.box.map((product, idx) => (

```

```

    <div
      key={`${product.name}-${product.id}-${idx}`}
      >{`${product.name} | Залишок: ${product.remainder}
шт;`}</div>

  ))
  : "Жодної коробки не додано"}
</div>

<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/straw">
    <GiRopeCoil size={25} title="Straws" />
  </NavLink>
  {products.straw.length > 0
    ? products.straw.map((product, idx) => (
      <div
        key={`${product.name}-${product.id}-${idx}`}
        >{`${product.name} | Залишок: ${product.remainder}
р;`}</div>

      ))
      : "Жодної соломинки не додано"}
</div>

<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/candle">
    <GiCandleLight size={25} title="Candles" />
  </NavLink>
  {products.candle.length > 0
    ? products.candle.map((product, idx) => (
      <div
        key={`${product.name}-${product.id}-${idx}`}

```

```

        >{\${product.name} | Залишок: \${product.remainder}
шт;`}</div>

    ))
    : "Жодної свічки не додано"}
</div>
<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/wax">
    <GiDrippingHoney size={25} title="Wax" />
  </NavLink>
  {products.wax.length > 0
    ? products.wax.map((product, idx) => (
      <div
        key={`\${product.name}-\${product.id}-\${idx}`}
        >{\${product.name} | Залишок: \${product.remainder}
р;`}</div>

      ))
      : "Жодного воску не додано"}
</div>
<div className="w-full items-center flex flex-wrap gap-2 text-2xl">
  <NavLink to="/tea">
    <GiTeapot size={25} title="Teas" />
  </NavLink>
  {products.tea.length > 0
    ? products.tea.map((product, idx) => (
      <div
        key={`\${product.name}-\${product.id}-\${idx}`}
        >{\${product.name} | Залишок: \${product.remainder}
р;`}</div>

      ))

```

```

      : "Жодного чаю не додано"}
    </div>
    <div className="w-full items-center flex flex-wrap gap-2 text-2xl">
      <NavLink to="/spoon">
        <GiSpoon size={25} title="Spoons" />
      </NavLink>
      {products.spoon.length > 0
        ? products.spoon.map((product, idx) => (
          <div
            key={`${product.name}-${product.id}-${idx}`}
            >{`${product.name} | Залишок: ${product.remainder}
шт;`} </div>
          ))
        : "Жодної ложки не додано"}
    </div>
    <div className="w-full items-center flex flex-wrap gap-2 text-2xl">
      <NavLink to="/ribbon">
        <GiBowTieRibbon size={25} title="Ribbons" />
      </NavLink>
      {products.ribbon.length > 0
        ? products.ribbon.map((product, idx) => (
          <div
            key={`${product.name}-${product.id}-${idx}`}
            >{`${product.name} | Залишок: ${product.remainder}
см;`} </div>
          ))
        : "Жодної стрічки не додано"}
    </div>
    <div className="w-full items-center flex flex-wrap gap-2 text-2xl">

```

```

    <NavLink to="/cream">
      <GiHoneyJar size={25} title="Cream" />
    </NavLink>
    {products.cream.length > 0
      ? products.cream.map((product, idx) => (
        <div
          key={`${product.name}-${product.id}-${idx}`}
          >{`${product.name} | Залишок: ${product.remainder}
г;`}</div>))
        : "Жодного крем-меду не додано"}
    </div>
    <div className="w-full items-center flex flex-wrap gap-2 text-2xl">
      <NavLink to="/garland">
        <GiLightBulb size={25} title="Garland" />
      </NavLink>
      {products.garland.length > 0
        ? products.garland.map((product, idx) => (
          <div
            key={`${product.name}-${product.id}-${idx}`}
            >{`${product.name} | Залишок: ${product.remainder}
шт;`}</div>
          ))
        : "Жодної гірлянди не додано"}
    </div>
  </div>
);
};
export default Home;

```

ДОДАТОК В
Слайди презентації

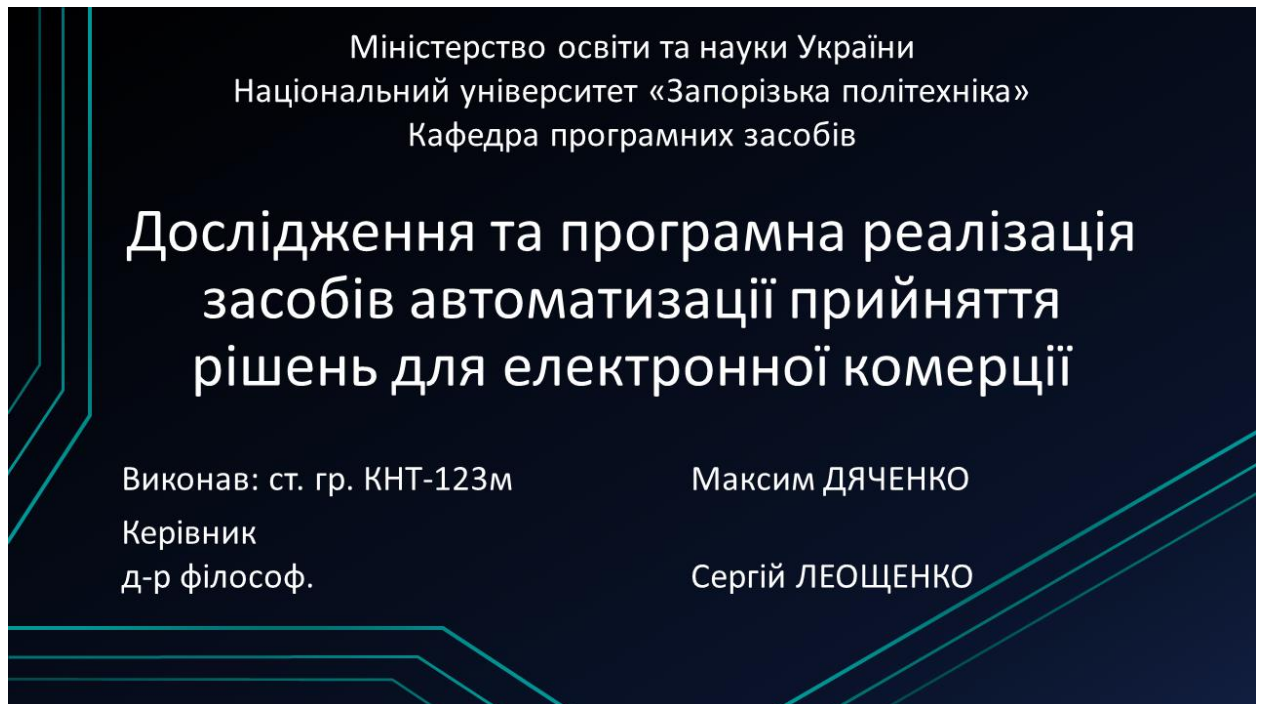


Рисунок В.1 – Слайд 1

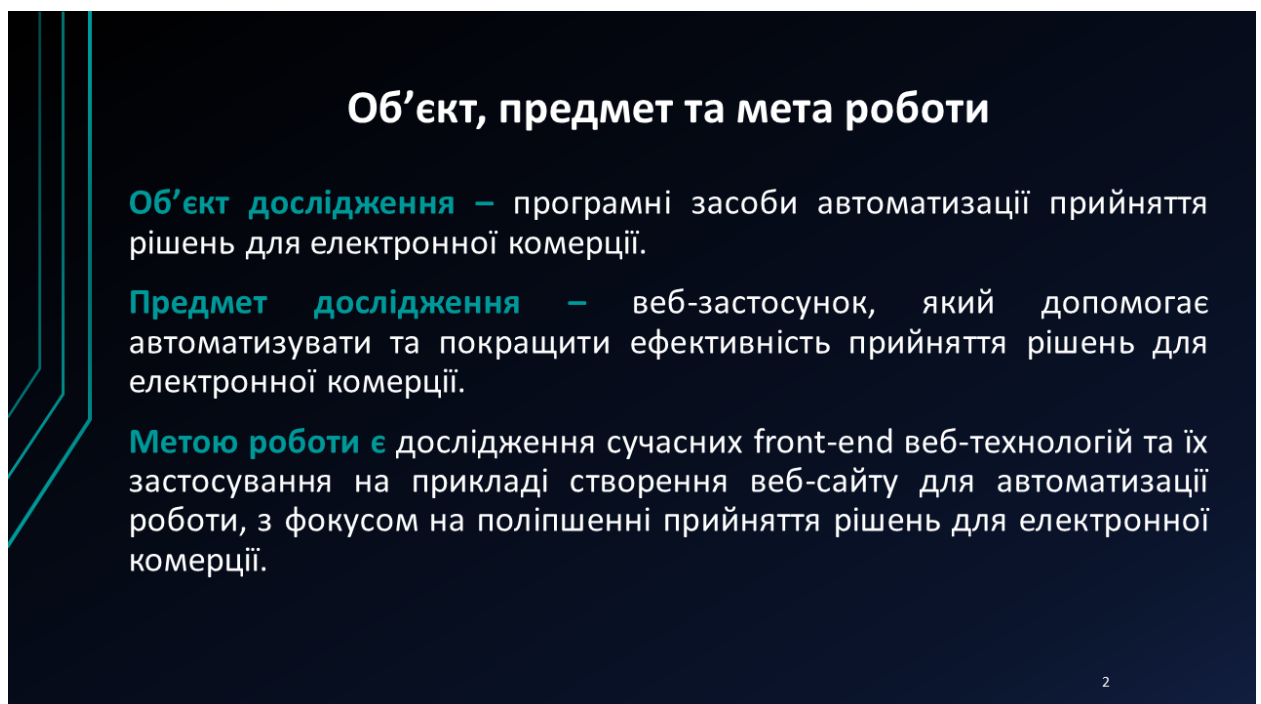


Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у дипломній кваліфікаційній роботі магістра необхідно вирішити такі завдання:

- провести аналіз предметної області та програмних засобів для автоматизації прийняття рішень у сфері електронної комерції;
- розробити модель взаємодії користувача з програмним забезпеченням для автоматизації прийняття рішень;
- здійснити проектування програмного забезпечення для вирішення завдань автоматизації прийняття рішень;
- створити програмне забезпечення для вирішення завдань автоматизації прийняття рішень;
- провести тестування розробленого програмного забезпечення для автоматизації прийняття рішень у сфері електронної комерції.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих програмних рішень

Назва	Основна мета	Ціна	Масштабованість
KeyCRM	Управління взаємовідносинами з клієнтами	Доступна для малих і середніх бізнесів	Підходить для малих і середніх підприємств
Prom.ua	Платформа для автоматизації продажів	Зазвичай вища за KeyCRM, залежно від функціональності	Підходить для середніх і великих підприємств
Microsoft Dynamics 365	Комплексне управління бізнесом і CRM	Висока, особливо для великих компаній	Підходить для великих підприємств

4

Рисунок В.4 – Слайд 4

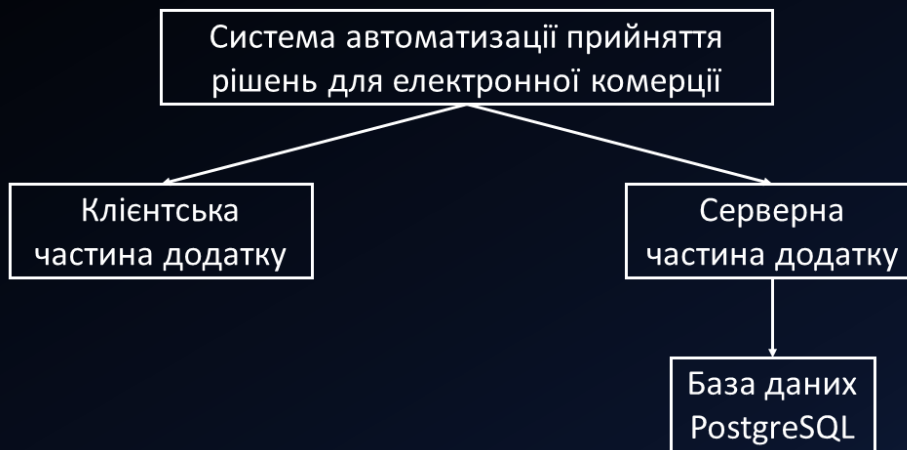
Вибір фреймворку розробки

Критерій	React	Angular
Більше свободи в виборі сторонніх бібліотек	Так	Ні
Тестування	Гірше	Краще
Підтримує легку композицію компонентів	Так	Ні
Підтримує хуки для управління життєвим циклом компонентів	Так	Ні
Має вбудовані засоби для формування анімацій	Ні	Так

5

Рисунок В.5 – Слайд 5

Структура програмного забезпечення автоматизації прийняття рішень для електронної комерції



6

Рисунок В.6 – лайд 6

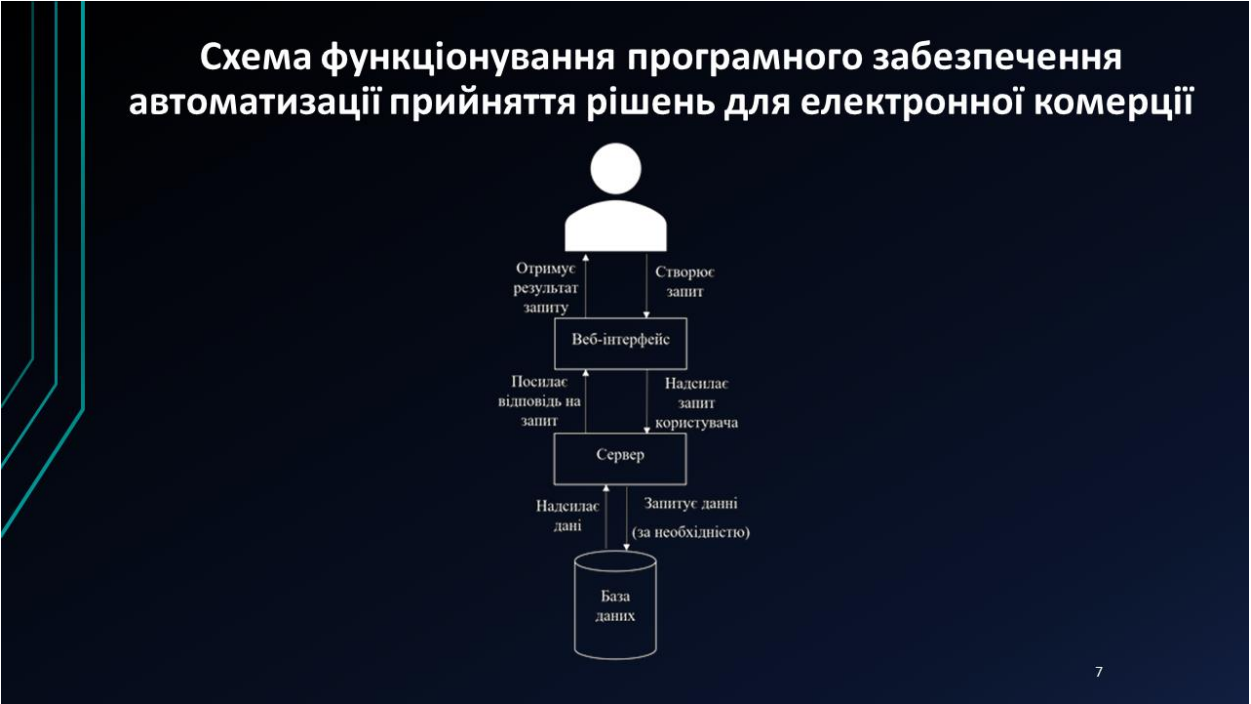


Рисунок В.7 – Слайд 7



Рисунок В.8 – Слайд 8

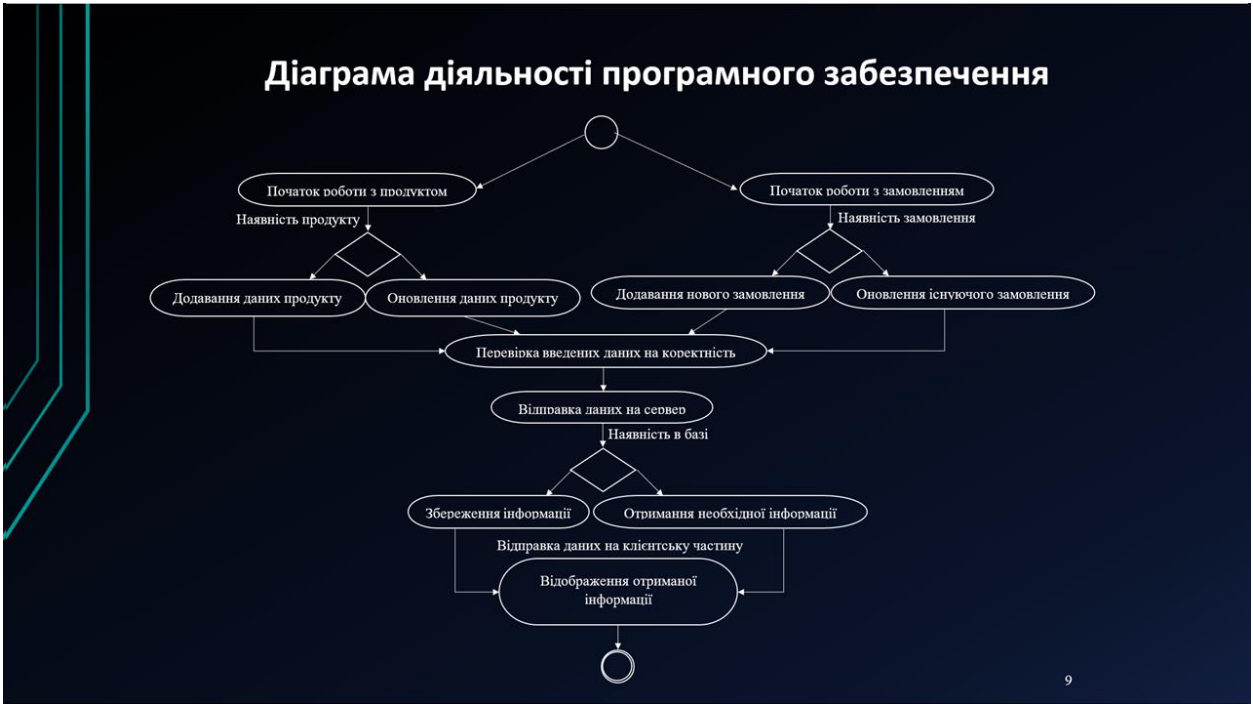


Рисунок В.9 – Слайд 9

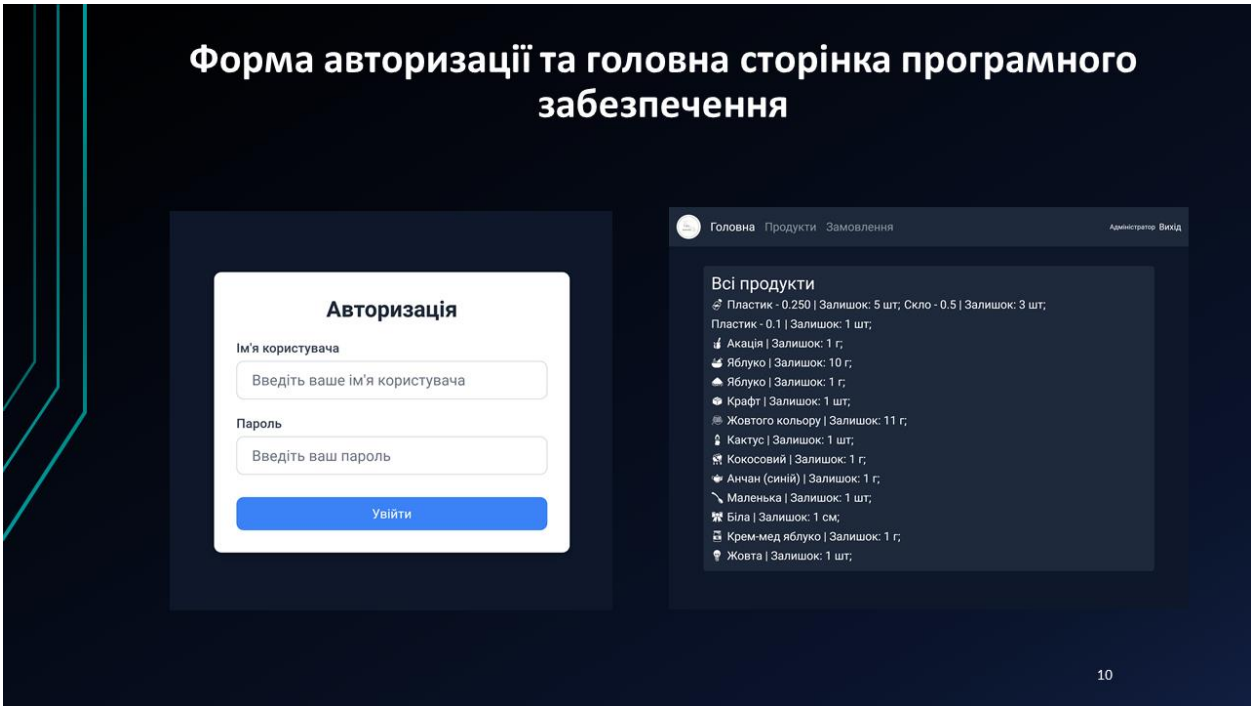


Рисунок В.10 – Слайд 10

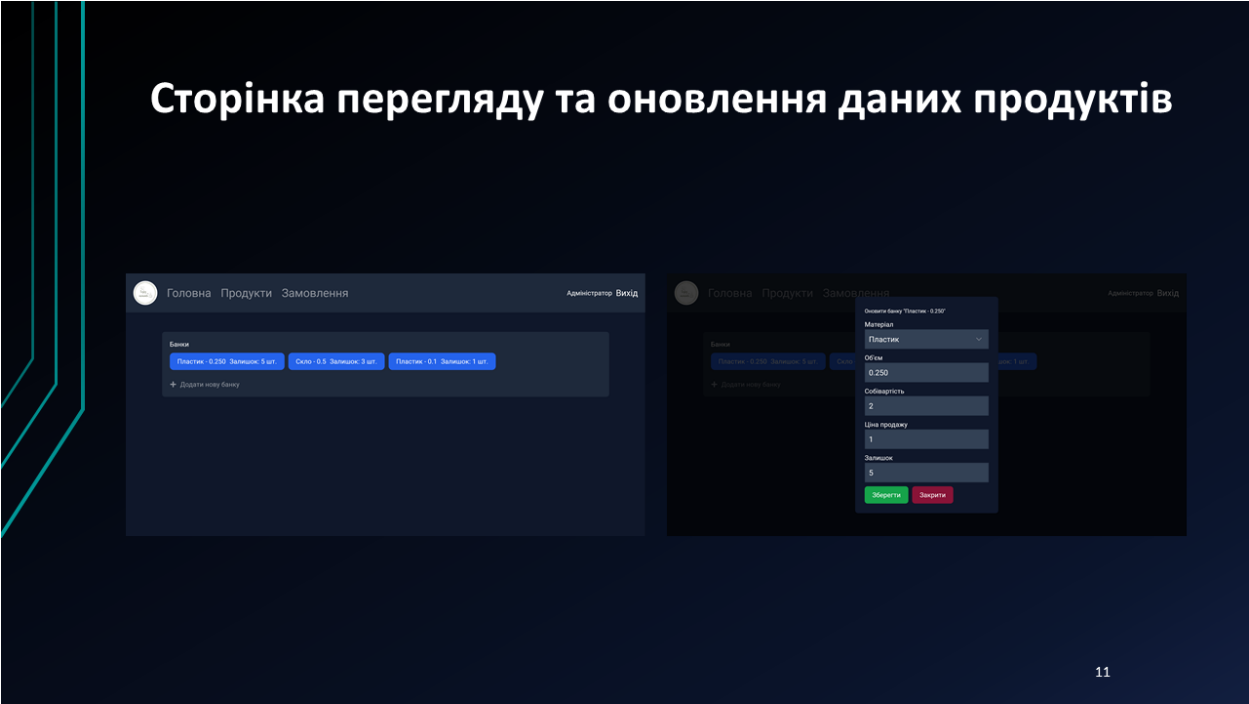


Рисунок В.11 – Слайд 11

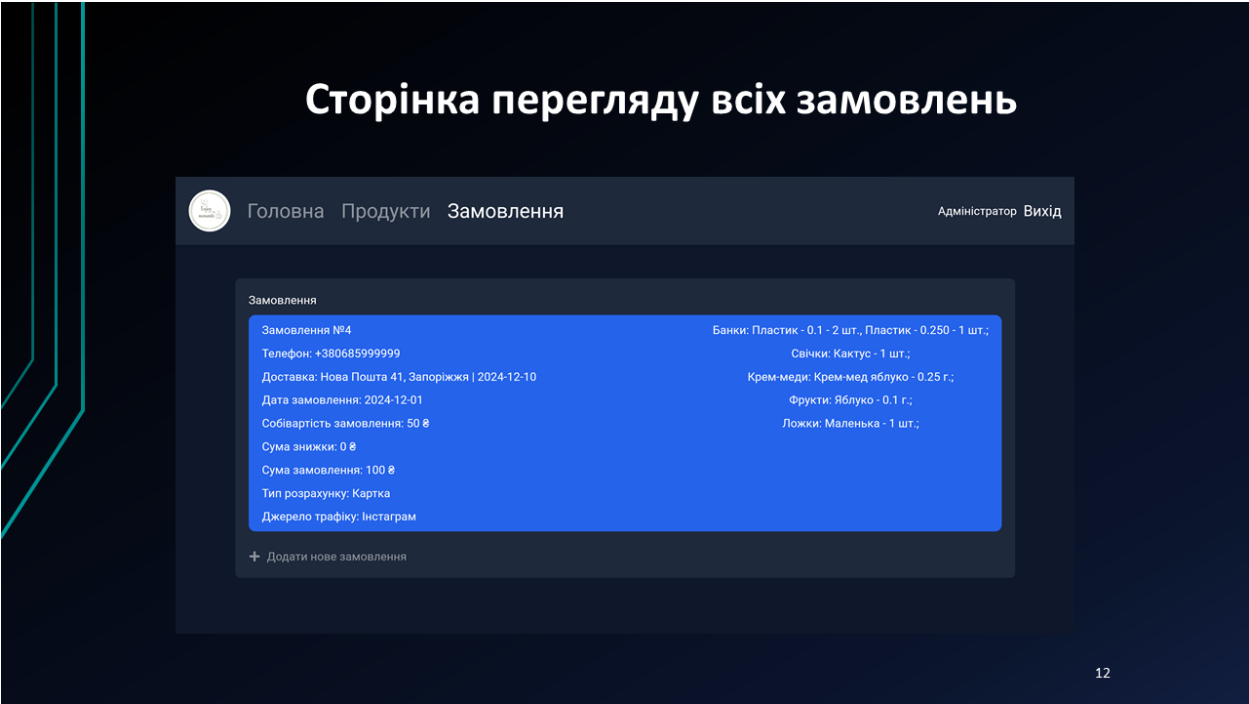


Рисунок В.12 – Слайд 12

Висновки

У ході виконання дипломної кваліфікаційної роботи було проведено дослідження і розробку програмного забезпечення, орієнтованого на автоматизацію прийняття рішень у сфері електронної комерції.

Досліджені існуючі програмні рішення, такі як KeyCRM, Prom та Microsoft Dynamics 365, підтвердили актуальність впровадження систем автоматизації для покращення бізнес-процесів, зокрема оптимізації замовлень, інвентаризації, обробки клієнтських даних і розрахунків прибутковості.

Усі завдання дипломної роботи магістра виконано повністю.

13

Рисунок В.13 – Слайд 13