Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Бакалаврська дипломна робота

на тему: «Web-платформа для управління та автоматизації з використанням API»

Виконав: студент 4 курсу, групи 2СП-20б
спеціальності 123 – Комп'ютерна інженерія
освітня програма «Системне програмування»
Березовський Олександр

Керівник :
к.т.н., доц., каф. ОТ Муращенко О.Г.
«14» 06 2024 р.

Рецензент:
К.т.н., доц., каф. ПЗ Ткаченко О.М.
«17» 06 2024 р.

Допущено до захисту
Зав. кафедри ОТ
д.т.н., проф. Азаров О. Д.
«18» червня 2024 р.

м. ВІННИЦЯ – 2024 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень бакалавр

Галузь знань – 12 «Інформаційні технології»

Спеціальність – 123 «Комп'ютерна інженерія» Освітньо-професійна програма «Комп'ютерна інженерія. Системне програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри ОТ, проф., д.т.н Азаров О.Д.

06.02.2024 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Березовському Олександру Сергійовичу

1 Тема роботи – «WEB-платформа для управління та автоматизації». Керівник роботи: Муращенко, д.т.н., професор кафедри ОТ, затверджені наказом вищого навчального закладу від "11" березня 2024 року №80.

2 Строк подання студентом роботи "13" червня 2024 рок.

3 Вихідні дані до роботи: модель розробки – ітеративна; вхідні дані – назва платформи, дані для реєстрації облікового запису, дані продуктів; вихідні дані – продукти, статистика продажів та заробітку, діаграма суми продажів; середовище розробки – VisualStudio Code; мова програмування – JavaScript.

4 Зміст розрахунково–пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задачі дослідження; розробка методів додавання та продажів товарів; розробка структури і програмних компонентів; тестування програми; висновки; список використаних джерел; додатки.

5 Консультанти розділів роботи приведені в таблиці 1.

6 Дата видачі завдання "12" березня 2024 року. Календарний план виконання роботи наведено к таблиці 2.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Р1-Р4	к.т.н., доц., каф. ОТ Муращенко О.Г.		

Таблиця 2 — Календарний план

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз завдання та вибір методу вирішення поставленої задачі	10.03.2024-17.03.2024	Вик.
2	Розробка алгоритму додавання, видалення товару в систему	18.03.2024-01.04.2024	Вик.
3	Розробка методу та алгоритму продажу товару	02.04.2024-12.04.2024	Вик.
4	Розробка методу та алгоритму генерації звіту на основі отриманих данних	13.04.2024-22.04.2024	Вик.
5	Аналіз та вибір мови програмування та середовища розробки	23.04.2024-03.05.2024	Вик.
6	Програмна реалізація	04.05.2024-12.05.2024	Вик.
7	Інструкція користувача	13.05.2024-15.05.2024	Вик.
8	Оформлення матеріалів до захисту БДР	16.05.2024 20.05.2024	Вик.

Студент Березовський О.С. _____

Керівник бакалаврської дипломної роботи Муращенго О.Г. _____

АНОТАЦІЯ

УДК 621.3

Березовський О. С.. Web-платформа для управління та автоматизації з використанням API. Бакалаврська дипломна робота зі спеціальності 123 «Комп`ютерна інженерія», освітня програма «Комп’ютерна інженерія. Системне програмування». Вінниця: ВНТУ, 2024. 102 с.

На укр. мові. Бібліогр.: 30 назв; рис.: 29; табл. 3.

У бакалаврській дипломна роботі розроблено web-платформу для управління та автоматизації товарообігу з використанням API. Платформа дозволяє покращити ефективність управління товарами та обробки замовлень. У загальній частині роботи розглянуто сучасні технології для створення web-додатків, такі як Vue.js, Vuetify та Firebase, а також обґрунтовано доцільність їх використання. У розрахунково-конструкторській частині виконано розробку архітектури системи, проектування бази даних та реалізацію основних функціональних модулів. У технологічній частині виконано налаштування системи, проведено тестування та оптимізацію продуктивності. Також наведено методику вирішення типових проблем.

Ключові слова: web-платформа, API, автоматизація, управління товарами, Vue.js, Vuetify, Firebase, база даних.

ABSTRACT

Berezovsky O. S. Web platform for management and automation using API. Bachelor's qualification work in the specialty 123 "Computer Engineering", educational program " Computer Engineering. System Programming". Vinnytsia: VNTU, 2024. 102 p.

In Ukrainian language. Bibliography: 30 names; 29 illustrations; 3 tables.

In the bachelor's qualification work, a web platform for managing and automating trade using API has been developed. The platform improves the efficiency of inventory management and order processing. The general part of the work reviews modern technologies for creating web applications such as Vue.js, Vuetify, and Firebase, and substantiates the feasibility of their use. The design and engineering part includes the development of the system architecture, database design, and implementation of the main functional modules. The technological part covers system setup, testing, and performance optimization. It also provides methods for solving common problems.

Keywords: web platform, API, automation, inventory management, Vue.js, Vuetify, Firebase, database.

ЗМІСТ

ВСТУП

Актуальність технології web-розробки та використання API відкривають широкі можливості для створення інноваційних рішень у сфері управління та автоматизації. Відтак, вибір теми "Web-платформа для управління та автоматизації з використанням API" є обґрунтованим з кількох причин.

По-перше, ця тема відповідає актуальним тенденціям у сфері програмної розробки та бізнес-автоматизації. Сучасні підприємства та бізнеси все більше використовують web-платформи для автоматизації своїх процесів та управління різними аспектами діяльності.

По-друге, web-платформи наразі є більш популярними аніж інші аналоги. Вони надають можливість інтеграції з різноманітними сервісами та додатками. Використання хмарних технологій дозволяє отримувати доступ до різних даних та функціоналу, практично з будь-якого пристрою, в якому є Інтернет.

По-третє, створення web-платформи для управління та автоматизації з використанням API відповідає потребам бізнесу в оптимізації та автоматизації рутинних операцій. Вона дозволить ефективно керувати процесами, включаючи управління товарами, облік фінансів та аналіз продажів, що сприятиме підвищенню ефективності та прибутковості бізнесу.

Дослідження цієї теми надасть можливість розвинути навички у сфері web-розробки та програмування, а також отримати цінний досвід у розробці програмного забезпечення для підприємств. Крім того, цей проект відкриває нові можливості для подальшого особистого і професійного зростання, дозволяючи реалізувати свій потенціал у сфері інформаційних технологій.

Мета і завдання дослідження полягає в створенні web-платформи, яка дозволить підприємствам ефективно керувати та автоматизувати свої бізнес-процеси. Цей проект націлено на покращення роботи з вхідними повідомленнями та даними шляхом централізації та автоматизації процесу їх отримання та аналізу. Відповідно, потрібно вирішити такі завдання.

Аналіз потреб користувачів та вимог до функціоналу платформи. Це включає визначення основних функцій, які повинна виконувати платформа, таких як

можливість реєстрації користувачів, вхід у систему, додавання та видалення товарів, аналіз фінансових показників та генерація звітів. Вибір технологій та інструментів для реалізації проекту. Включає вибір мови програмування, бібліотек для розробки web-додатків, а також API для взаємодії з базою даних;

Проектування архітектури платформи, де визначаються основні компоненти системи та їх взаємодія. Це включає розробку бази даних, розробку інтерфейсу та інші аспекти проекту;

Реалізація функціоналу платформи передбачає програмування різноманітних модулів, які відповідають вимогам користувачів. Цей етап включає в себе налаштування бази даних;

Документування процесу розробки включає у себе створення технічної документації, яка описує архітектуру системи, процеси розробки та використання платформи. Це важливий етап, який забезпечує зрозумілість та доступність для майбутніх розробників та адміністраторів системи.

Об'єкт дослідження — процеси отримання, обробки та зберігання інформації в базі даних.

Предмет дослідження — методи та засоби централізованого збору даних.

Методи дослідження включають в себе аналіз сучасних технологій та інструментів для розробки веб-платформи, включаючи вивчення можливостей API різних сервісів. Проектування архітектури системи з урахуванням потреб управління та автоматизації процесів. Реалізація функціональності платформи для автоматизації рутинних операцій за допомогою API. Тестування продуктивності та надійності розробленої системи. Експериментальне впровадження та апробація розроблених рішень для перевірки їх ефективності та можливостей впровадження. Ці методи дозволяють систематично вирішувати завдання з автоматизації та управління процесами через веб-платформу з використанням сучасних API.

Практична цінність отриманих результатів в бакалаврській дипломній роботі для різних аспектів розробки та використання web-платформи:

— розроблені алгоритми для управління та обробки даних на платформі дозволяють оптимізувати роботу користувачів та забезпечити швидке та ефективне виконання завдань;

— оптимізована структура сайту дозволяє забезпечити зручну та легку навігацію для користувачів та сприяє покращенню користувацького досвіду та забезпечує ефективну взаємодію з платформою;

— методи розробки використання сучасних технологій та модульного підходу до розробки дозволяє забезпечити зручну та ефективну розробку та підтримку платформи.

Особистий внесок здобувача — визначення потреб користувачів, аналізі вимог до функціоналу платформи та виборі оптимальних методів розробки – виконано автором самостійно. Автору належить розробка усіх алгоритмів у цій роботі.

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ, ЗАСОБУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз потреб користувачів та вимог до функціоналу

Під час аналізу потреб користувачів було проведено ґрунтовне опитування цільової аудиторії [1] — власників малого бізнесу, які мають досвід ведення бізнесу в інтернеті. Мета опитування полягала у збиранні відгуків, виявленні основних проблем та потреб користувачів, а також визначенні ключових вимог до функціоналу майбутньої web-платформи для управління та автоматизації.

Було розроблено опитування, яке охоплювало різні аспекти управління товарами, аналітики, користувацького інтерфейсу та загальних вимог до платформи. Опитування було ретельно перевірено на зрозумілість та відповідність питань перед розповсюдженням.

Опитування розповсюджувалось через підприємців. Це забезпечило достятнє для роботи охоплення цільової аудиторії.

У процесі опитування були зібрані детальні відповіді на такі ключові питання:

1) проблеми при управлінні товарами;

— складність додавання, редагування та видалення товарів на існуючій платформі;

— обмежені можливості налаштування варіантів товарів (розмірів, кольорів тощо);

— відсутність зручних інструментів імпорту/експорту даних про товари;

2) необхідні функції для управління товарами;

— інтуїтивний та простий інтерфейс для додавання, редагування, видалення товарів;

— просте налаштування товарів із зазначенням різних властивостей;

3) важливі дані про товари;

— назва;

— ціна закупівлі та ціна продажу;

4) необхідні аналітичні дані та звіти;

— звіти про продажі;

— аналіз популярності та рентабельності окремих товарів/категорій;

5) вимоги до інтерфейсу користувача;

— інтуїтивно зрозумілий дизайн;

— швидка навігація та пошук необхідних функцій;

— чітке та логічне розташування елементів управління;

6) загальні вимоги до платформи;

— масштабованість та продуктивність системи;

— сумісність з різними web-браузерами та пристроями;

— можливість високого навантаження.

Результати опитування були ретельно проаналізовані та систематизовані. Це дозволило виявити найбільш критичні потреби користувачів, такі як ефективне управління товарами [2], доступ до аналітичних звітів для вивчення продажів, а також забезпечення простого та зручного користувацького інтерфейсу.

Отримані дані стали основою для формування детальних вимог до функціоналу майбутньої web-платформи, забезпечуючи її максимальну відповідність реальним потребам цільової аудиторії.

1.2 Порівняльний аналіз аналогів

Існує не так багато платформ для управління та автоматизації, але у кожної є недоліки. Для створення конкурентного ресурсу [3] потрібно проаналізувати найпопулярніші аналоги. Для кращого аналізу порівняння проведено серед трьох найвідоміших в Україні платформ та створюваним ресурсом.

Найбільш відомими ресурсами є:

— Quincefin;

— Dilovod;

— Облік-онлайн.

Розглянемо детально обрані аналоги.

Quincefin — це web-платформа для управління та автоматизації бізнесу, яка пропонує комплексне рішення для малих та середніх підприємств (рисунок 1.1). Можна виділити такі переваги, як:

— продуктивність Quincefin відзначається високою швидкістю роботи та ефективністю, забезпечуючи безперебійну та плавну взаємодію з системою;

— спільнота користувачів навколо Quincefin сформувалась велика та активна спільнота користувачів, які діляться досвідом, надають поради та підтримку, також, спільнота є цінним ресурсом для вирішення проблем та навчання;

— платформи Quincefin пропонує гнучкі можливості налаштування, дозволяючи адаптувати її під специфічні потреби бізнесу, що забезпечує високий рівень персоналізації та зручності використання.

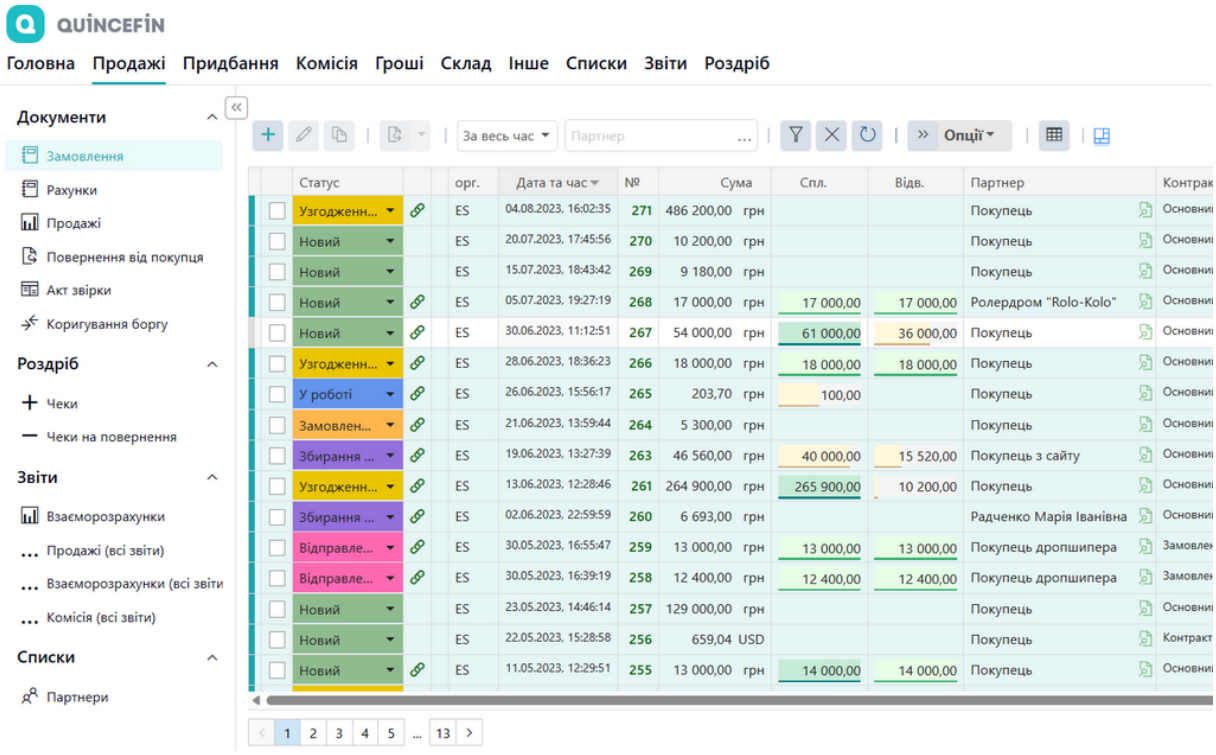


Рисунок 1.1 — інтерфейс web-ресурсу Quincefin

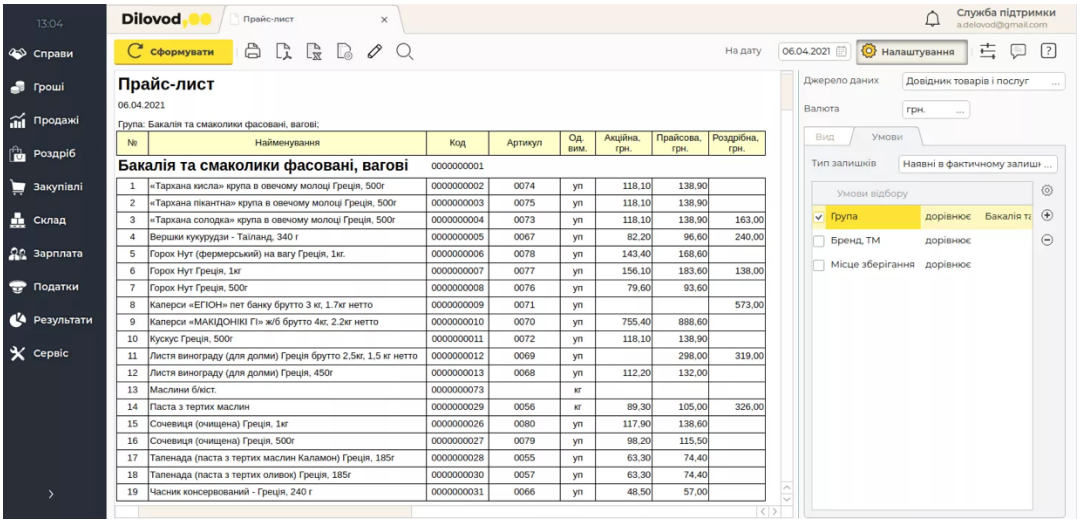
Quincefin, хоча і є потужним інструментом для управління бізнесом, все ж має деякі недоліки, які варто врахувати перед його використанням:

— платформа не є дешевим рішенням на ринку, її цінова політика вважається високою для більшості малих та середніх підприємств;

— деякі користувачі відзначають, що інтерфейс ресурсу може бути складним для освоєння, що потребує додаткового часу на навчання персоналу;

— іноді користувачі скаржаться на недостатню підтримку та рідкість оновлень функцій, що може вплинути на стабільність та розвиток програми у майбутньому.

Dilovod — це ще один варіант програмного забезпечення для управління бізнесом (рисунок 1.2), який має свої переваги та недоліки.



№	Найменування	Код	Артикул	Од. вим.	Акційна, грн.	Прайсова, грн.	Роздрібна, грн.
	Бакалія та смаколики фасовані, вагові	0000000001					
1	«Тархана кисла» крупа в овечому молоці Греція, 500г	0000000002	0074	уп	118,10	138,90	
2	«Тархана пікантна» крупа в овечому молоці Греція, 500г	0000000003	0075	уп	118,10	138,90	
3	«Тархана солодка» крупа в овечому молоці Греція, 500г	0000000004	0073	уп	118,10	138,90	163,00
4	Вершки кукурудзи - Таїланд, 340 г	0000000005	0067	уп	82,20	96,60	240,00
5	Горох Нут (фермерський) на вагу Греція, 1кг.	0000000006	0078	уп	143,40	168,60	
6	Горох Нут Греція, 1кг	0000000007	0077	уп	156,10	183,60	138,00
7	Горох Нут Греція, 500г	0000000008	0076	уп	79,60	93,60	
8	Каперси «ЕГЮН» пет банку брутто 3 кг, 1.7кг нетто	0000000009	0071	уп			573,00
9	Каперси «МАКІДОНІКІ Г!» ж/б брутто 4кг, 2.2кг нетто	0000000010	0070	уп	755,40	888,60	
10	Кускус Греція, 500г	0000000011	0072	уп	118,10	138,90	
11	Листя винограду (для долми) Греція брутто 2,5кг, 1,5 кг нетто	0000000012	0069	уп		298,00	319,00
12	Листя винограду (для долми) Греція, 450г	0000000013	0068	уп	112,20	132,00	
13	Маслини б/кіст.	0000000073		кг			
14	Паста з тертих маслин	0000000029	0056	кг	89,30	105,00	326,00
15	Сочевиця (очищена) Греція, 1кг	0000000026	0080	уп	117,90	138,60	
16	Сочевиця (очищена) Греція, 500г	0000000027	0079	уп	98,20	115,50	
17	Тапенада (паста з тертих маслин Каламон) Греція, 185г	0000000028	0055	уп	63,30	74,40	
18	Тапенада (паста з тертих оливок) Греція, 185г	0000000030	0057	уп	63,30	74,40	
19	Часник консервований - Греція, 240 г	0000000031	0066	уп	48,50	57,00	

Рисунок 1.2 — Інтерфейс web-ресурсу Dilovod

Переваги Dilovod:

— Dilovod відомий своєю простотою та інтуїтивним інтерфейсом, що дозволяє легко освоювати систему навіть користувачам з мінімальним досвідом;

— платформа має зручний інтерфейс, що дозволяє легко орієнтуватися та використовувати різноманітні функції без значних зусиль;

— платформа пропонує різноманітні функції для управління бізнесом, включаючи облік, фінанси, збут та інші аспекти, що дозволяє здійснювати повний контроль над діяльністю підприємства;

— Dilovod надає можливість налаштування під конкретні потреби бізнесу, що дозволяє адаптувати систему до унікальних вимог кожного клієнта.

Недоліки Dilovod:

— хоча Dilovod може бути ефективним інструментом для управління бізнесом, вартість використання платформи може бути високою, особливо для малих підприємств або стартапів з обмеженим бюджетом;

— деякі користувачі вказують на обмежені можливості масштабування системи Dilovod, особливо у випадку збільшення обсягів діяльності або розширення бізнесу;

— деякі користувачі скаржаться на неефективну підтримку та відсутність швидких відповідей з боку команди розробників Dilovod.

"Облік-онлайн" - це платформа для управління бізнесом, яка пропонує широкий спектр функціональності для підтримки різноманітних видів діяльності. Деякі переваги та недоліки цієї платформи наведено нижче.

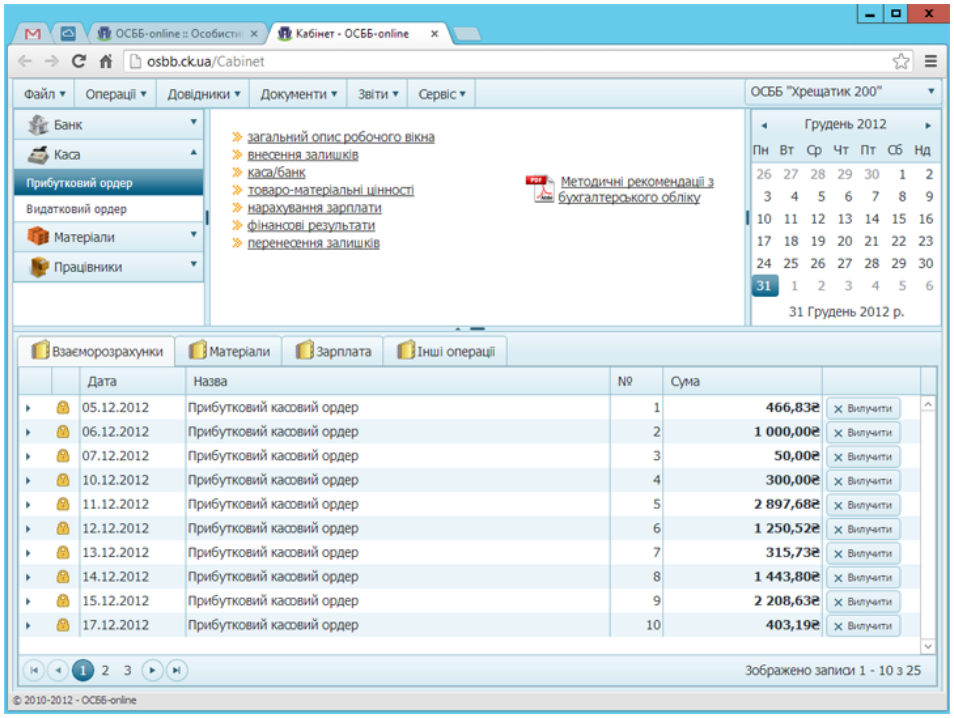


Рисунок 1.3 — Інтерфейс web-ресурсу «Облік-онлайн»

Переваги платформи:

— "Облік-онлайн" пропонує багато функціональних можливостей, включаючи облік, аналітику, звітність, управління клієнтами та інше, що робить її універсальним рішенням для багатьох видів бізнесу;

— надає можливості для ефективного управління клієнтами, включаючи базу даних клієнтів, зв'язок з ними та аналіз [4] їхнього поведінки;

— вартість вважається прийнятною для багатьох бізнесів;

— гнучкість та налаштування: платформа дозволяє налаштовувати різноманітні параметри та адаптувати її до конкретних потреб кожного бізнесу.

Недоліки "Облік-онлайн":

— для деяких великих підприємств може виникнути проблема масштабованості системи, особливо якщо потрібні великі обсяги даних чи висока продуктивність;

— платформа має не зовсім зручний та застарілий інтерфейс, на який скаржаться багато клієнтів.

За допомогою дослідження аналогів ви визначили недоліки та переваги в порівняні з створюваним ресурсом «Skladodrom». Після аналізу отримали такі результати (таблиця 1.1).

Таблиця 1.1 — Порівняльний аналіз аналогів

Критерії	Quincefin	Dilovod	Облік-онлайн	Skaldodrom
Вартість	5/10	5/10	8/10	10/10
Простота	5/10	7/10	5/10	9/10
Масштабованість	7/10	7/10	5/10	6/10
Функціонал	8/10	8/10	7/10	6/10
Підтримка	7/10	7/10	5/10	9/10

Після дослідження було доведено актуальність створення власного сервісу з автоматизації.

1.3 Методи управління товарами

Функціонал для управління [5] товарами може включати в себе різноманітні можливості, спрямовані на ефективне керування і контроль за товарним асортиментом. Розглянемо деякі ключові функції для управління товарами.

Додавання товарів — цей метод дозволяє користувачам додавати нові товари до системи. При додаванні товару користувач зазвичай вказує його назву, ціну закупки, ціну продажу та інші характеристики, які можуть включати опис, категорію, зображення тощо.

Редагування товарів — цей метод дозволяє користувачам вносити зміни до існуючих товарів у системі. Вони можуть змінювати будь-які характеристики товару, такі як ціна, опис, категорія тощо;

Видалення товарів — цей метод дозволяє користувачам видаляти товари з системи, які більше не потрібні або є застарілими;

Пошук товарів — цей метод дозволяє користувачам шукати товари за різними критеріями, такими як назва, категорія, ціна тощо. Пошук може бути здійснений за допомогою текстового запиту або фільтрів;

Перегляд інформації про товар — цей метод дозволяє користувачам переглядати детальну інформацію про конкретний товар, таку як його характеристики, наявність на складі, історію продажів тощо;

Генерація звітів про товари — цей метод дозволяє користувачам генерувати різноманітні звіти про товари, такі як список всіх товарів, звіт про залишки на складі, звіт про продажі тощо. Генерація звітів допомагає аналізувати ефективність управління товарним асортиментом і приймати стратегічні рішення.

Ці методи разом надають користувачам повний контроль [6] над товарними операціями у системі.

1.4 Постановка задач дослідження

Після ретельного аналізу переваг і недоліків наявних web-ресурсів для автоматизації та управління, а також дослідження технічних особливостей [7] різних варіантів та поточного стану хмарних технологій, були визначені наступні завдання для розробки власного web-продукту:

— провести огляд сучасних технологій для створення web-платформи;

— провести огляд методів збереження даних;

— порівняти та обрати систему для збереження даних;

— розробити алгоритм додавання даних з бази даних за допомогою API;

— розробити алгоритм зчитування даних з бази даних;

— проаналізувати дані для звіту та створити алгоритм для звіту;

— створити дизайн та розробити зручний інтерфейс;

— створити компоненти програми для управління та автоматизації;

— створити інструкцію користувача.

Функціональні вимоги:

— користувач може додавати будь-які товари з можливістю введеня назви, ціни закупівлі, ціни продажі, кількості та інше;

— користувач може здійснити продаж товару;

— користувач може переглядати статистику продажів та рентабельність ;

— користувач може переглядати усі додані товари;

— користувач може вилучати товари.

Отже, в процесі аналізу та обґрунтування вибору методу розробки та постановки задач дослідження для нашого проекту були ретельно вивчені потреби користувачів та вимоги до функціоналу, що включає управління товарами. Визначення оптимального методу розробки та формулювання задач дослідження є критично важливим етапом у розробці будь-якого програмного продукту, оскільки вони визначають подальші кроки і напрямок роботи.

У процесі аналізу були визначені основні потреби користувачів нашого програмного продукту. Основні функціональні вимоги включають можливість додавання, редагування та видалення товарів, управління категоріями, пошук та фільтрацію товарів, генерацію звітів та імпорт/експорт даних [8]. Важливо також було врахувати нефункціональні вимоги, такі як підтримка різних операційних систем та захищена передача даних.

У зв'язку з потребою у гнучкості та можливості швидко адаптуватися до змін вимог, було вирішено використовувати ітеративний підхід до розробки програмного забезпечення. Цей метод дозволить нам ефективно взаємодіяти з клієнтом, швидко внести зміни та вдосконалювати продукт на основі зворотного зв'язку.

Для ефективного управління товарами в системі будуть присутні такі методи, як зазначено в таблиці 1.2.

Таблиця 1.2 – Методи управління

Метод управління	Опис
Додавання	Дозволяє користувачам додавати нові товари до бази даних
Редагування	Надає можливість змінювати існуючі дані про товари, такі як назва, опис, ціна, кількість на складі тощо.
Видалення	Дозволяє користувачам видаляти товари з бази даних, які більше не потрібні або не доступні для продажу.
Пошук	Забезпечує можливість здійснювати пошук товарів за різними критеріями, такими як назва, категорія, ціна тощо.
Фільтрація	Дозволяє фільтрувати товари за певними параметрами для швидкого знаходження необхідних товарів.
Перегляд	Надає можливість перегляду детальної інформації про товар, включаючи всі доступні дані та зображення.
Генерація звітів	Дозволяє автоматично генерувати різноманітні звіти на основі зібраних даних про товари, такі як звіт про продажі, звіт про залишки тощо.

Ці методи дозволять нам забезпечити повний контроль та ефективне управління асортиментом товарів.

Для досягнення успішного розроблення програмного продукту ми сформулювали ряд конкретних завдань дослідження. Ці завдання включають аналіз сучасних технологій безсерверних розрахунків, огляд існуючих методів отримання оповіщень, розробку методів та алгоритмів отримання оповіщень за допомогою webхуків та публічних API, а також за допомогою парсингу web-ресурсів, розробку методів та алгоритмів генерації звітів на основі отриманих оповіщень, створення структури та графічного інтерфейсу користувача та

програмних компонентів для централізованого збору оповіщень, розробку модульних тестів для автоматизованого тестування, ручне тестування розробленого програмного забезпечення та розробку інструкції користувача.

У висновку, аналіз та обґрунтування вибору методу розробки та постановка задач дослідження є ключовими етапами у розробці програмного продукту. Вони дозволяють чітко визначити напрямок роботи та забезпечити відповідність платформи потребам користувачів та вимогам функціоналу.

2 РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ ФУНКЦІЙ

2.1 Опис функцій

Опис функцій є важливим етапом у процесі розробки будь-якої програмної системи. Він забезпечує чітке визначення завдань, які має виконувати кожна функція в системі, та оператив вона повинна здійснювати для досягнення цих завдань. Цей детальний опис допомагає розробникам зрозуміти вимоги до системи та усвідомити, який функціонал необхідно реалізувати для успішного функціонування системи.

Кожна функція [9] має свої основні завдання та операції, які необхідно виконати для їх реалізації. Наприклад, у випадку системи управління товарами, функції можуть включати додавання нових товарів, редагування існуючих, видалення та перегляд деталей товарів. Для кожної з цих функцій потрібно чітко визначити, які дані необхідно ввести чи змінити, які операції необхідно виконати для збереження цих змін у базі даних, та як користувачі будуть взаємодіяти з системою для виконання цих завдань.

Опис функцій також допомагає уникнути недопорозумінь між розробниками та замовниками проекту. Чітке визначення завдань кожної функції дозволяє замовнику краще розуміти, яким чином система буде працювати та які можливості вона буде надавати. Водночас, це допомагає розробникам уникнути непорозумінь у процесі реалізації функціоналу, що може призвести до неправильностей у роботі системи.

Функції, які будуть у системі:

— додавання нових товарів;

— редагування існуючих товарів;

— видалення товарів;

— перегляд деталей товарів;

— створення звіту.

Розглянемо детальніше кожну функцію.

Додавання нових товарів — це ключова функція у системі управління товарами, яка забезпечує можливість користувачам додавати нові товари до бази даних системи. Ця функція включає в себе кілька етапів, які дозволяють користувачам ввести всю необхідну інформацію про товар та зберегти її у системі.

Першим кроком є надання користувачу можливості ввести дані про новий товар. Ця інформація може включати такі поля, як назва товару, ціна, кількість на складі тощо. Користувач може ввести ці дані через спеціальну форму, яка забезпечує зручний інтерфейс для введення даних.

Після введення всіх необхідних даних користувачем, інформація про товар передається до системи, де вона обробляється та зберігається у базі даних. У цьому етапі важливо здійснити перевірку коректності введених даних, щоб уникнути помилок чи некоректних записів у базі.

Після успішного збереження інформації про товар у базі даних, він стає доступним для подальшого використання у системі. Користувачі можуть переглядати доданий товар, використовуючи різні функції системи, такі як пошук, фільтрація, або вони можуть просто переглядати повний список товарів, які доступні у системі.

Редагування існуючих товарів — ця функція надає користувачам можливість вносити зміни до вже існуючої інформації про товари. Ця функція дозволяє підтримувати актуальність даних та забезпечує зручний інструмент для коригування інформації про товари в системі.

Першим етапом редагування є пошук існуючого товару, який користувач бажає відредагувати. Це може бути зроблено за допомогою унікального ідентифікатора товару або за допомогою пошуку за іншими параметрами, такими як назва товару, категорія, або будь-який інший атрибут.

Після того, як товар знайдено, користувач має змогу внести зміни до будь-якої інформації про товар. Це може включати зміну назви товару, опису, ціни, категорії, виробника, або будь-якого іншого поля, що пов'язане з товаром. Користувач може використовувати спеціальну форму для редагування даних, яка надає зручний інтерфейс для введення нової інформації.

Після внесення всіх необхідних змін користувачем, нові дані зберігаються в базі даних, замінюючи старі дані про товар. Це забезпечує актуальність інформації в системі та дозволяє користувачам отримувати доступ до оновленої інформації про товари.

Функція видалення товарів надає можливість користувачам очищати базу від застарілих або непотрібних записів про товари. Під час видалення товару важливо забезпечити правильність та безпеку цього процесу, щоб уникнути втрати даних або видалення неправильних записів.

Першим етапом процесу видалення є вибір конкретного товару для видалення. Користувач може вибрати товар для видалення за допомогою унікального ідентифікатора, назви товару, категорії або будь-якого іншого атрибуту, за яким можна ідентифікувати товар. Після вибору товару користувач підтверджує свою дію, щоб уникнути випадкового видалення.

Після підтвердження дії система видаляє обраний товар з бази даних. Під час цього процесу важливо перевірити, чи немає пов'язаних з товаром даних, таких як замовлення, інвентаризація або історія продажів. Якщо такі дані існують, вони також повинні бути видалені або оновлені, щоб уникнути неузгодженостей в базі даних.

Функція перегляду деталей товарів є ключовою для забезпечення ефективного управління інвентарем та надання користувачам доступу до необхідної інформації про товари в системі [10].

Під час використання цієї функції користувач може переглядати різноманітну інформацію про кожен товар, який знаходиться в базі даних. Основні елементи інформації, які зазвичай включаються до перегляду, включають: назву, ціну закупки, ціну продажі, кількість тощо.

Функціонал перегляду повина бути зручною та інтуїтивно зрозумілою для користувачів. Це може включати можливість сортування, фільтрації та пошуку товарів за різними параметрами, а також використання зручного інтерфейсу для навігації та перегляду інформації.

2.2 Вибір методу розробки

На основі аналізу вимог та обставин проекту проводиться вибір методу розробки [11]. У даному випадку можна вибрати ітеративний або каскадний підхід до розробки програмного забезпечення.

Ітеративна модель – це методологія, що передбачає поділ розробки на невеликі ітерації або цикли (рисунок 2.1). Кожен цикл включає в себе фази аналізу, проектування, реалізації, тестування та впровадження, і завершується виходом наступної версії програмного продукту.

Рисунок 2.1 — Ітеративна модель

Основна перевага ітеративного підходу полягає в тому, що він дозволяє розробникам швидко реагувати на зміни вимог та впроваджувати нові функції на ранніх етапах розробки. Наприклад, якщо під час аналізу виявляється, що потрібно додати нову функціональність або змінити існуючу, це може бути зроблено без великих перерв у процесі розробки.

Крім того, ітеративний підхід дозволяє залучити клієнта до процесу розробки, що сприяє кращому зрозумінню його потреб та очікувань. Клієнт може бачити проміжні результати роботи на кожному етапі і вносити свої пропозиції та корективи.

Прикладом ітеративного підходу може бути розробка web-платформи для даної бакалаврської дипломної роботи. Перша ітерація може включати базовий функціонал, такий як реєстрація користувачів та додавання товарів. Наступні

ітерації можуть додавати нові функції, такі як обробка замовлень, звітність або інтеграція з іншими сервісами. Кожна ітерація дозволяє покращувати продукт на основі отриманого зворотного зв'язку та змінювати напрямок розробки залежно від потреб користувачів та ринкових умов.

Такий підхід дозволяє забезпечити швидку реакцію на зміни вимог та виходити на ринок з високоякісним та конкурентоспроможним продуктом.

Каскадна модель [12] — це методологія, яка передбачає послідовне виконання етапів розробки від аналізу вимог до впровадження продукту (рисунок 2.2). Кожен етап розробки ретельно планується і завершується перед переходом до наступного.

Рисунок 2.2 — Каскадна модель

Перший етап — аналіз вимог. Він включає в себе збір і аналіз вимог користувачів, визначення функціональності та основних характеристик продукту. На цьому етапі формується специфікація вимог, яка буде використовуватися на всіх наступних етапах.

Другий етап — проектування. Передбачає розробку архітектури та дизайну продукту на основі вимог і специфікації. Це включає в себе визначення структури бази даних, інтерфейсу користувача та алгоритмів роботи програми.

Третій етап — реалізація. Передбачає написання коду та реалізацію розробленого дизайну. Розробники виконують програмування на основі визначеної архітектури та специфікації, дотримуючись кращих практик програмування.

Четвертий етап — тестування. Включає в себе тестування програмного продукту для переконання в його правильному функціонуванні та відповідності вимогам.

П'ятий етап — впровадження. Передбачає випуск програмного продукту в експлуатацію. Це може включати встановлення програмного забезпечення на сервері або дистрибуцію клієнтам.

Прикладом каскадного підходу до розробки може бути створення web-платформи для електронної комерції. Спочатку проводиться детальний аналіз вимог користувачів щодо функціональності та інтерфейсу платформи. Потім розробляється архітектура системи, проектується інтерфейс користувача та реалізується логіка роботи системи. Після цього проводяться тестування для перевірки правильності роботи програми, і лише після успішного завершення всіх етапів web-платформа готова до впровадження.

У даному випадку, враховуючи те, що вимоги до web-платформи можуть змінюватися протягом розробки, а також потребу в швидкому впровадженні та реагуванні на зміни, ітеративний підхід до розробки є більш привабливим. Він дозволить ефективно взаємодіяти з клієнтом, вдосконалювати продукт на основі зворотного зв'язку та швидко адаптуватися до змін вимог.

Таким чином, рекомендовано використовувати ітеративний підхід до розробки програмного забезпечення для даного проекту. Це дозволить забезпечити більшу гнучкість, ефективність та відповідність потребам користувачів у процесі розробки web-платформи.

2.3 Розробка методів і алгоритмів функцій

Щоб чітко визначити кроки, які потрібно зробити для досягнення потрібного результату, потрібно створити детальний алгоритм [13].

Для додавання, було розроблено наступний алгоритм (рисунок 2.3).

Крок 1 — введення даних. На цьому кроці користувач вводить дані про продукт, які запитує система (назва, ціна, кількість тощо).

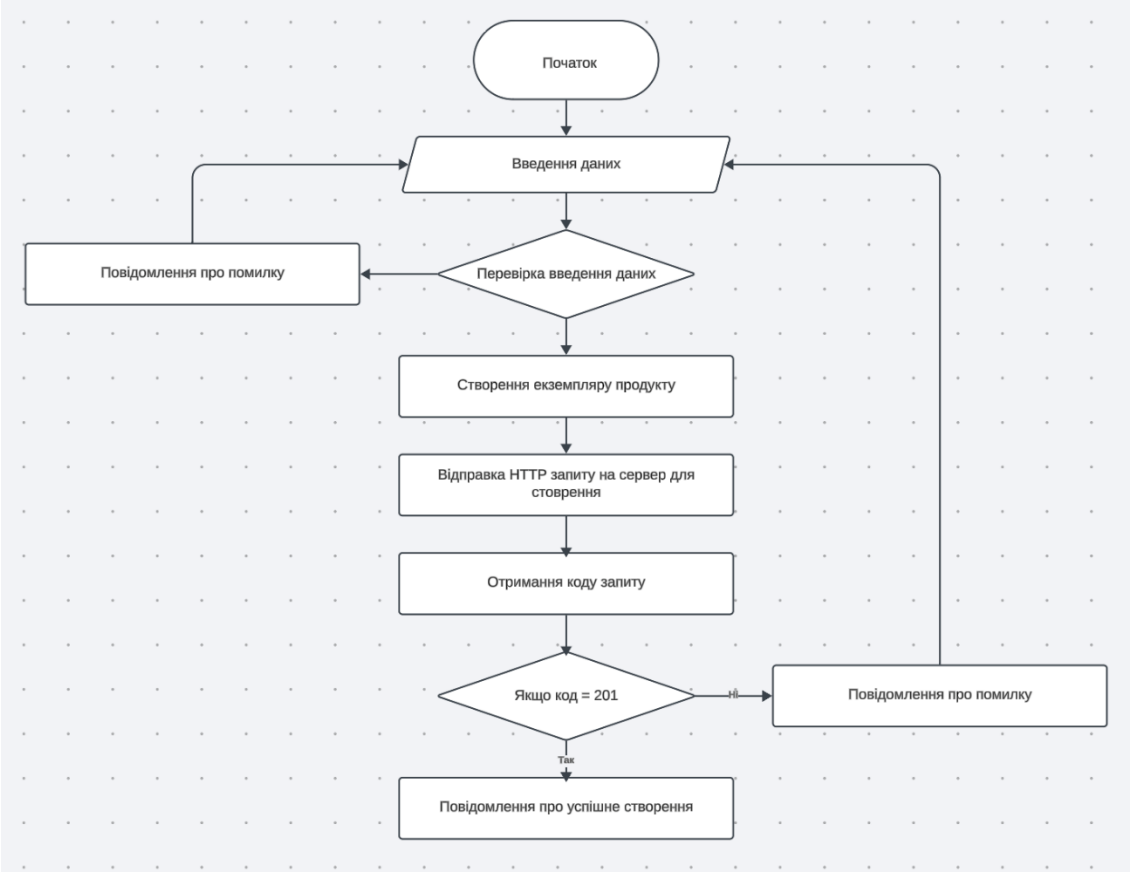


Рисунок 2.3 — Алгоритм додавання

Крок 2 — перевірка введених даних. Виконуються різні перевірки, такі як: обов`язковість введення, мінімальна довжина назви та опису, мінімальна кількість тощо. Якщо успішно йдемо до наступного кроку, якщо ні – виводимо на екран повідомлення про помилку та просимо ввести дані ще раз.

Крок 3 — створення екземпляру продукту. Перед відправкою даних на сервер потрібно створити екземпляр товару, щоб відправити його HTTP запитом.

Крок 4 — відправка HTTP запиту на сервер для створення. Відправляємо запит на сервер командою «POST» [14] з параметрами продукту для створення в базі даних.

Крок 5 — отримання коду запиту. Якщо код 201, виводимо на екран повідомлення про успішне створення продукту, якщо ні – повідомлення про

помилку та прохання ввести дані ще раз.

Для видалення, було розроблено наступний алгоритм (рисунок 2.4).

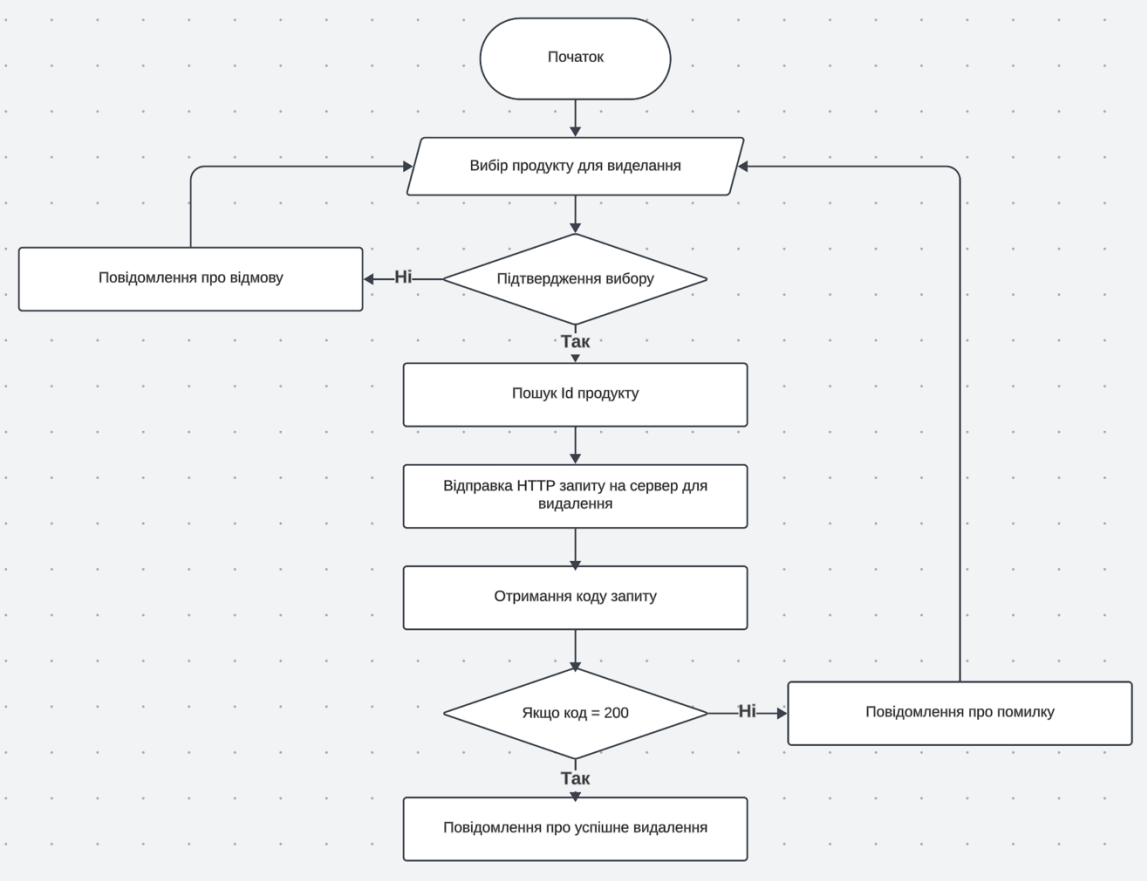


Рисунок 2.4 — Алгоритм видалення

Крок 1 — вибір продукту для видалення. Користувач вибирає продукт, який потрібно видалити та натискає відповідну кнопку.

Крок 2 — підтвердження вибору. Відкривається діалогове вікно підтвердження вибору з кнопками «Так» та «Ні» Якщо користувач обирає «Так» продовжуємо далі алгоритм, якщо «Ні» – виводимо на екран повідомлення про відмову та просимо вибрати продукт ще раз.

Крок 3 — пошук id продукту. Перед відправкою даних на сервер потрібно визначити який продукт було вибрано та взнати його id.

Крок 4 — відправка HTTP запиту на сервер для видалення. Відправляємо запит на сервер командою «DELETE» з параметром id для видалення продукту в базі даних.

Крок 5 — отримання коду запиту. Якщо код 200, виводимо на екран повідомлення про успішне видалення продукту, якщо ні – повідомлення про помилку та прохання вибрати продукт ще раз.

Для редагування, було розроблено наступний алгоритм (рисунок 2.5).

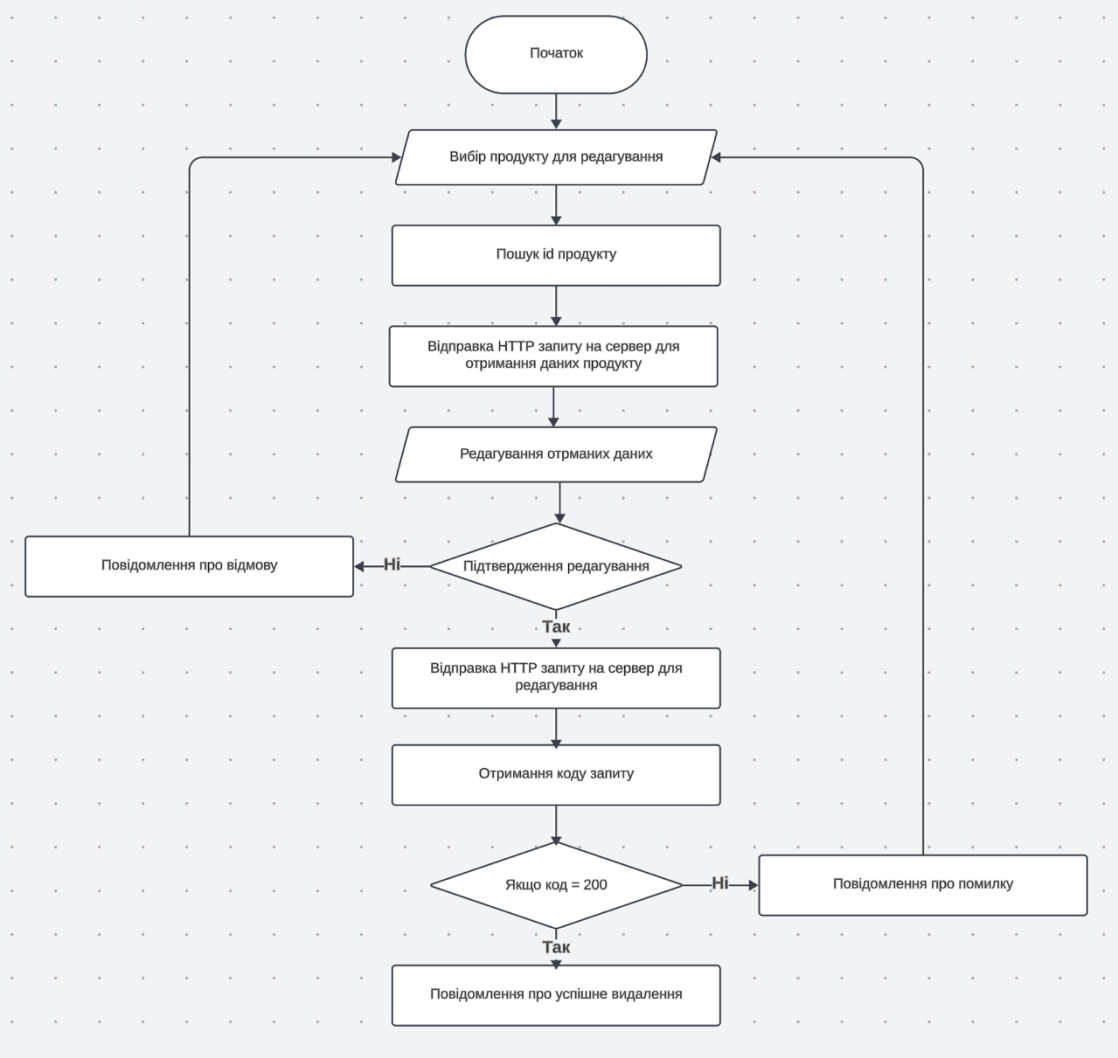


Рисунок 2.5 — Алгоритм редагування

Крок 1 — вибір продукту для редагування. Користувач вибирає продукт, який потрібно видалити та натискає відповідну кнопку.

Крок 2 — пошук id продукту. Щоб відредагувати дані потрібно визначити який продукт було вибрано та взнати його id.

Крок 3 — відправка HTTP запиту на сервер для отримання даних продукту. Відправляємо запит на сервер командою «GET» з параметром id для отримання

інформації про продукту в базі даних.

Крок 4 — редагування отриманих даних. Користувач редагує дані, які було отримано з бази даних.

Крок 5 — підтвердження редагування. Відкривається діалогове вікно підтвердженням редагування з кнопками «Так» та «Ні» Якщо користувач обирає «Так» продовжуємо далі алгоритм, якщо «Ні» — виводимо на екран повідомлення про відмову та просимо вибрати продукт для редагування ще раз.

Крок 6 — відправка HTTP запиту на сервер для редагування. Відправляємо запит на сервер командою «PUT» з відредагованими параметрами продукту для редагування інформації про продукт в базі даних.

Крок 7 — отримання коду запиту. Якщо код 200, виводимо на екран повідомлення про успішне видалення продукту, якщо ні — повідомлення про помилку та прохання вибрати продукт ще раз.

Для отримання детальної інформації, було розроблено наступний алгоритм (рисунок 2.6).

Крок 1 — вибір продукту для перегляду інформації. Користувач вибирає продукт, інформацію якого потрібно отримати.

Крок 2 — пошук id продукту. Щоб отримати інформацію потрібно визначити який продукт було вибрано та взнати його id.

Крок 3 — відправка HTTP запиту на сервер для отримання інформації. Відправляємо запит на сервер командою «GET» з параметром id для отримання детальної інформації про продукт.

Крок 4 — отримання коду запиту. Якщо код 200, виводимо на екран інформацію про продукт, якщо ні — повідомлення про помилку та прохання вибрати продукт для перегляду інформації ще раз.

Розробивши такі детальні алгоритми, було визначено, які дії потрібно зробити для отримання правильного результату виконання різних функцій. Ці алгоритми допоможуть при розробці web платформи.

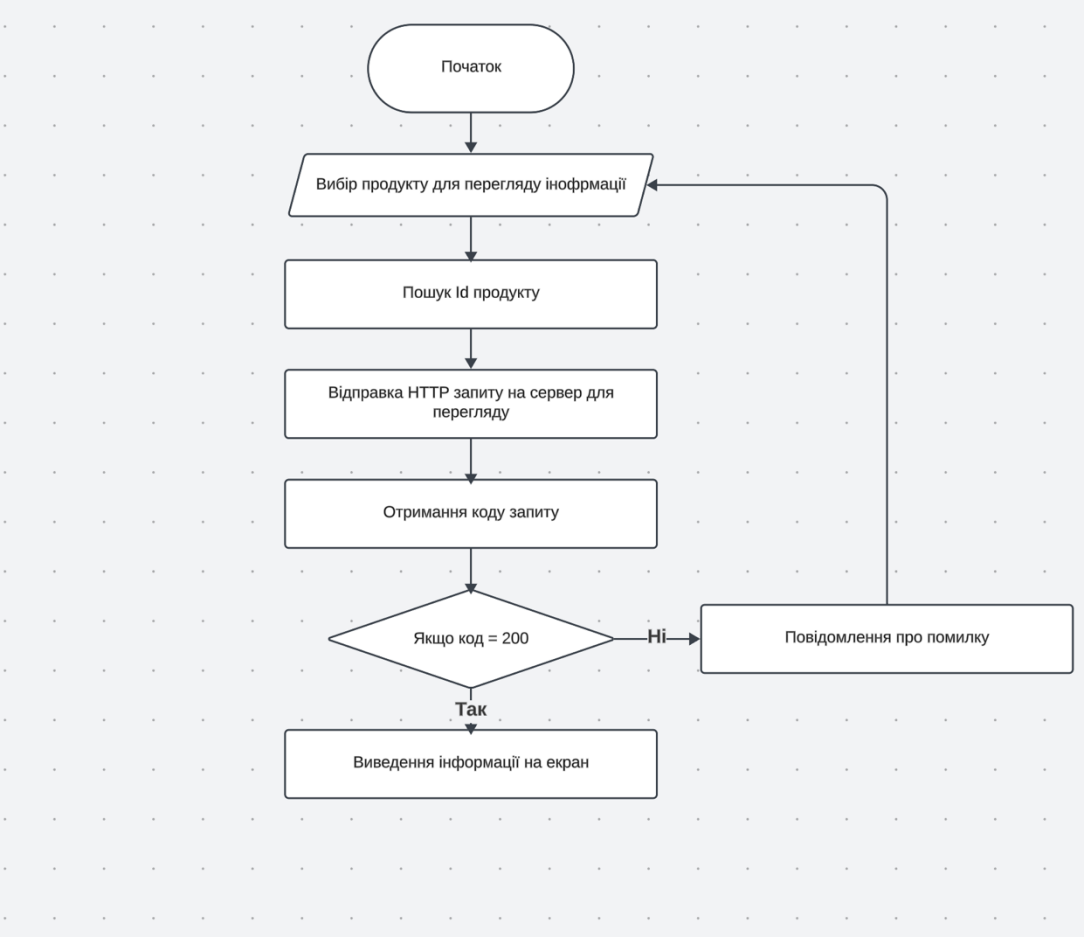


Рисунок 2.6 — Алгоритм отримання детальної інформації

2.4 Розробка методу та алгоритму генерації звіту

Звіти є важливим інструментом для аналізу даних і прийняття рішень, тому їх правильна розробка має велике значення. Правильно сформовані звіти дозволяють бізнесу отримувати точну та актуальну інформацію про стан товарів, продажі, залишки на складі та інші важливі показники, що сприяє ефективному управлінню і стратегічному плануванню. Даний процес включає в себе визначення вимог до звітів, структурування даних, розробку алгоритму генерації звіту, реалізацію цих алгоритмів у коді та їх подальше тестування. Розглянемо детально процес визначення таких вимог, як: вимоги до звітів, структурування даних та розробку алгоритму генерації звіту. Реалізацію алгоритму та тестування [15] буде виконано далі у роботі.

Визначаємо вимоги до звіту. Звіти поділяються на такі типи :

— звіт про продажі за весь час;

— звіт про продажі за сьогодні;

— звіт про залишки товарів на складі;

Структура та взаємозв’язки баз даних:

— таблиця Products (товари): містить поля product_id, name, description, price, category_id, stock_quantity;

— таблиця Sales (продажі): містить поля sale_id, product_id, quantity, sale_date.

— таблиця Products пов'язана з Sales через product_id.

Алгоритм генерації звіту (рисунок 2.7).

Крок 1 — ініціалізація параметрів. Отримання параметрів звіту від користувача (наприклад, період, категорії, фільтри), перевірка введених параметрів на коректність.

Крок 2 — відправка HTTP запиту на сервер для отримання даних. Відправка запиту на сервер командою «GET» з параметром таблиці Sales для отримання інформації про продажі товарів в базі даних.

Крок 3 — отримання коду запиту. Якщо код 200, переходимо до наступного кроку, якщо ні – виводим повідомлення про помилку та прохання ввести параметри ще раз.

Крок 4 — обробка отриманих даних. Обробка отриманих даних для формування звіту, розрахунок додаткових метрик, якщо необхідно (наприклад, середня ціна продажу, кількість залишків);

Крок 5 — форматування звіту. Створення таблиці звіту, форматування звіту для зручності.

Таким чином, розробка методу та алгоритму генерації звіту включає визначення вимог, структурування даних [16], створення алгоритмів обробки інформації та їх подальшу реалізацію і тестування. Це дозволяє створити ефективний інструмент для аналізу даних і прийняття рішень.

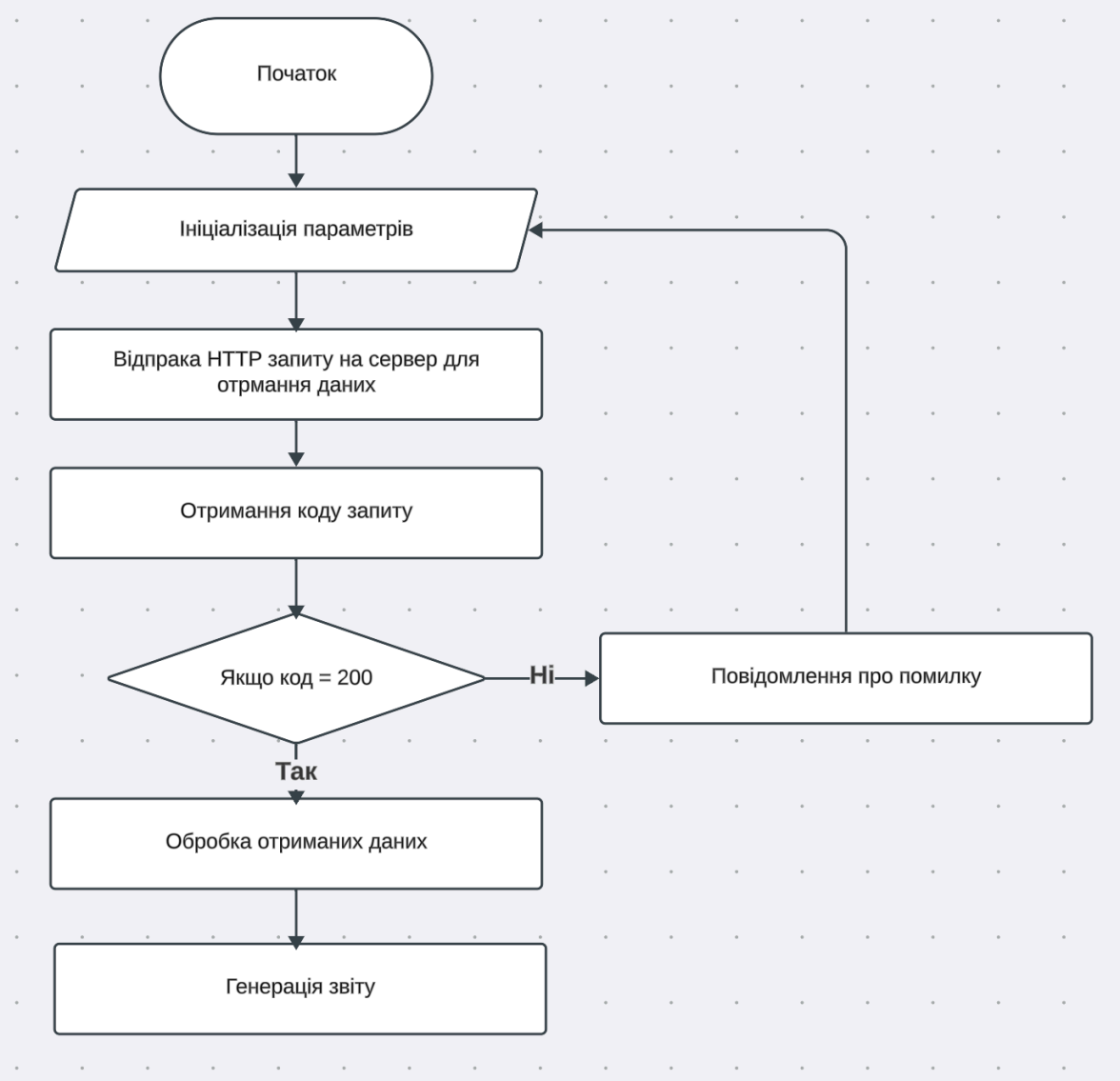


Рисунок 2.7 — Алгоритм генерації звіту

3 РОЗРОБКА СТРУКТУРИ ТА ПРОГРАМНИХ КОМПОНЕНТІВ

3.1 Варіативний аналіз та обґрунтування вибору технологій

У процесі розробки сучасних web-додатків важливим аспектом є вибір правильних технологій [17], які забезпечать високу продуктивність, гнучкість та зручність використання як для розробників, так і для кінцевих користувачів. У даному проекті, зважаючи на специфіку та вимоги до системи, було прийнято рішення використовувати базу даних Firebase, фреймворк Vue.js та бібліотеку Vuetify. Розглянемо детально обґрунтування цього вибору, а також порівняння з іншими можливими варіантами.

Firebase є платформою для розробки web- та мобільних додатків, яка пропонує цілий ряд сервісів, включаючи базу даних в реальному часі, аутентифікацію користувачів, хостинг та інші. Розглянемо основні причини вибору Firebase.

Безсерверна архітектура Firebase дозволяє створювати додатки без необхідності налаштування серверної частини, що значно спрощує розробку та знижує витрати на обслуговування.

Реальна часова база даних Firebase пропонує базу даних в реальному часі, що забезпечує миттєве оновлення даних без потреби в додаткових запитах до сервера.

Firebase автоматично масштабується відповідно до потреб додатку, що дозволяє зосередитись на розробці без турботи про інфраструктуру. Інтеграція з іншими сервісами Google — Firebase легко інтегрується з іншими сервісами Google, такими як Google Analytics, що полегшує збір та аналіз даних.

Vue.js — це прогресивний JavaScript-фреймворк для створення користувацьких інтерфейсів.

Вибір Vue.js для даного проекту базується на наступних перевагах:

— Vue.js має невеликий поріг входу, що робить його ідеальним для швидкого освоєння та використання в проекті;

— Vue.js підтримує компонентний підхід до розробки, що дозволяє легко створювати, повторно використовувати та підтримувати окремі компоненти;

— Vue.js забезпечує високу продуктивність завдяки оптимізації роботи з віртуальним DOM;

— Vue.js має велику і активну спільноту розробників, що забезпечує швидку підтримку та великий вибір готових рішень та бібліотек;

Vuetify — це компонентна бібліотека для Vue.js, яка реалізує Material Design.

Вибір Vuetify обґрунтований наступними причинами:

— Vuetify надає великий набір готових до використання компонентів, що відповідають принципам Material Design, що значно прискорює розробку інтерфейсу;

— використання Vuetify забезпечує високу якість та консистентність дизайну додатку, що робить інтерфейс користувача привабливим та інтуїтивно зрозумілим;

— Vuetify має чудову документацію та активну спільноту, що спрощує процес навчання та впровадження.

Альтернативою Firebase могли б бути традиційні серверні бази даних, такі як MySQL або PostgreSQL. Проте, використання таких рішень вимагало б додаткових витрат на налаштування та обслуговування серверної частини, що ускладнило б процес розробки та збільшило б витрати на інфраструктуру.

Альтернативами Vue.js могли б бути React або Angular.

React має велику популярність і підтримується Facebook. Він також є компонентним, але потребує додаткових бібліотек для повноцінного функціонування (наприклад, для управління станом або роутингом). React складніше освоїти, ніж Vue.js, особливо для новачків.

Angular є фреймворком від Google, що пропонує комплексне рішення для розробки великих додатків. Проте, він складніший у навчанні та використанні в порівнянні з Vue.js, і його повний функціонал може бути надмірним для більш простих проектів.

Альтернативами Vuetify могли б бути інші бібліотеки, такі як Bootstrap-Vue або Element.

Bootstrap-Vue — це бібліотека, що надає компоненти Bootstrap для Vue.js. Вона забезпечує добре відомий та перевірений дизайн, але не підтримує принципи Material Design, що може бути важливим для деяких проектів.

Element — це ще одна компонентна бібліотека для Vue.js, яка пропонує красиві та зручні компоненти. Проте, Vuetify надає більше можливостей для реалізації складних інтерфейсів, що відповідають принципам Material Design.

Вибір Firebase, Vue.js та Vuetify для розробки web-системи управління товарами базується на їхніх перевагах у простоті, гнучкості, продуктивності та підтримці. Ці технології забезпечують швидку та ефективну розробку, що дозволяє зосередитися на створенні функціональності та якості кінцевого продукту. Порівняння з іншими варіантами показує, що обрані рішення є оптимальними для даного проекту, враховуючи вимоги та цілі системи [18].

3.2 Вибір середовища розробки

Вибір середовища розробки є важливим кроком у процесі створення програмного продукту. Це рішення впливає на ефективність роботи розробників, зручність налагодження коду, можливості інтеграції з іншими інструментами та технологіями, а також на загальну продуктивність команди. Існує багато варіантів середовищ розробки, кожне з яких має свої переваги та недоліки. У цьому розділі розглянемо кілька популярних середовищ розробки та обґрунтуємо вибір оптимального для нашого проекту.

Visual Studio Code (VS Code) [19] — це легке та потужне середовище розробки, яке підтримує широкий спектр мов програмування та технологій. Воно є безкоштовним і відкритим вихідним кодом, що робить його доступним для всіх розробників. Графічний інтерфейс наведено на рисунку 3.1.

Основні переваги:

— інтуїтивно зрозумілий інтерфейс та зручність у використанні;

— підтримує велику кількість розширень, які додають нові можливості, такі як дебагінг, інтеграція з базами даних, управління задачами тощо;

— великий каталог розширень, що дозволяє налаштовувати середовище під конкретні потреби проекту;

— вбудована підтримка Git дозволяє легко здійснювати контроль версій безпосередньо з середовища розробки;

— підтримка мов програмування, що робить його універсальним інструментом;

— висока продуктивність навіть на середньому апаратному забезпеченні;

— працює на Windows, macOS та Linux;

безкоштовний, що є великим плюсом для невеликих команд та індивідуальних розробників.

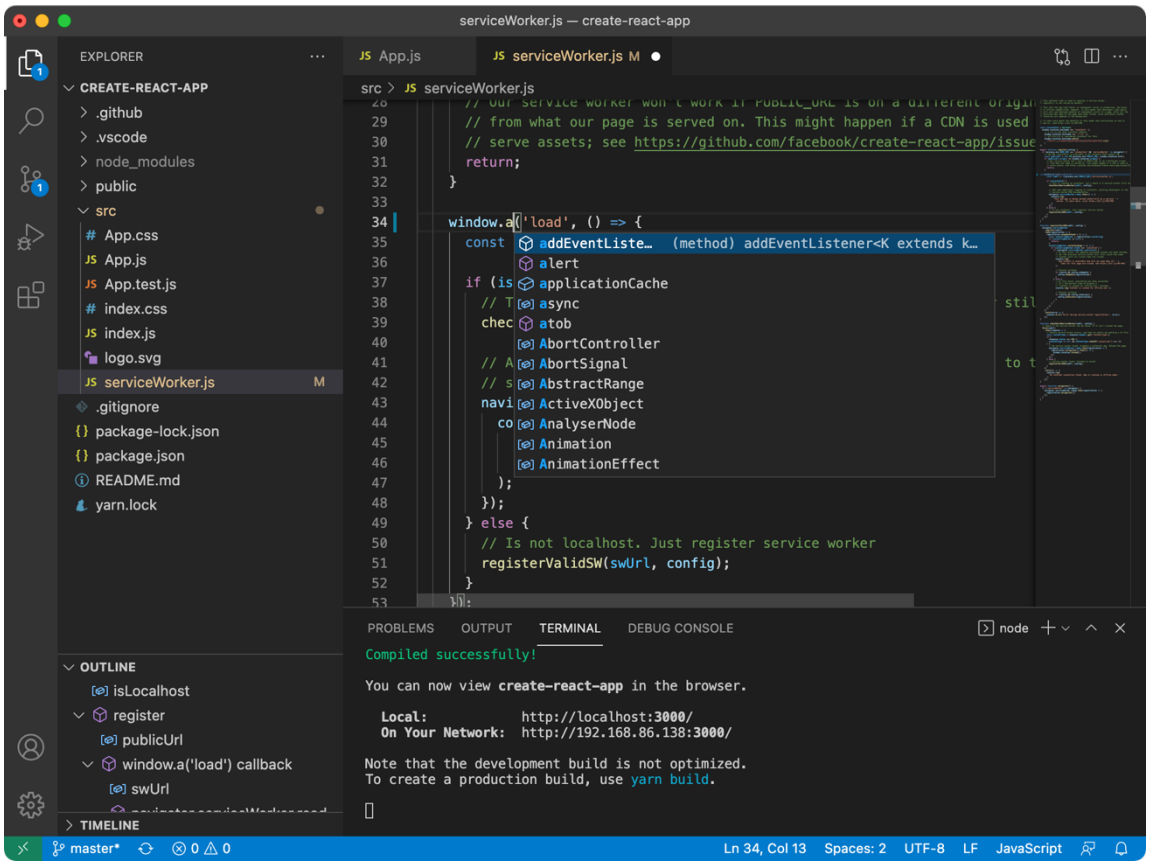


Рисунок 3.1 — Графічний інтерфейс Visual Studio Code

Atom — це текстовий редактор з відкритим вихідним кодом, створений GitHub. Він позиціонується як "хакерський" текстовий редактор, який можна налаштовувати до найменших дрібниць. Графічний інтерфейс наведено на рисунку 3.2.

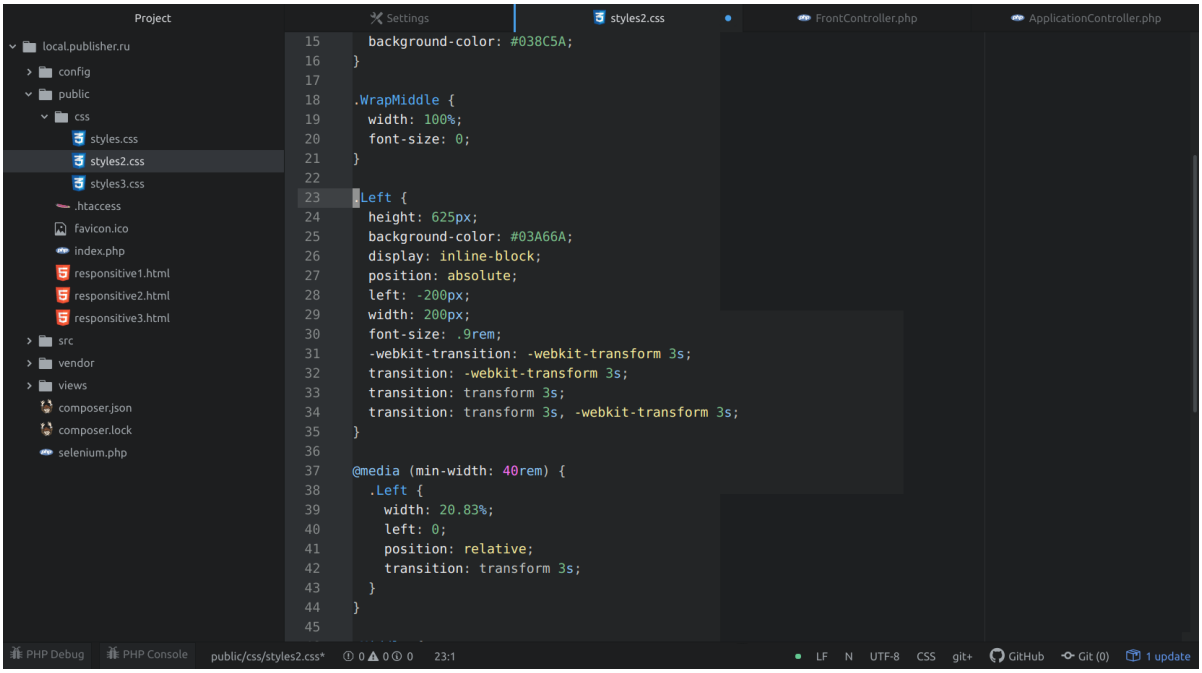


Рисунок 3.2 — Графічний інтерфейс Atom

Основні переваги:

— зручний інтерфейс, але може вимагати часу для налаштування;

— підтримує розширення, але не такий потужний, як спеціалізовані IDE;

— широкий вибір плагінів для додавання нових функцій;

— вбудована підтримка Git, але не настільки зручна, як у VS Code;

— підтримка багатьох мов програмування через плагіни;

— може бути повільнішим на великому коді або на слабкому апаратному забезпеченні;

— працює на Windows, macOS та Linux.

— безкоштовний.

WebStorm [20] — це комерційне середовище розробки від JetBrains, яке спеціалізується на web-розробці. Воно пропонує безліч інструментів для розробки на JavaScript, TypeScript та інших web-технологіях. Графічний інтерфейс наведено на рисунку 3.3.

Основні переваги:

— інтерфейс може бути складним для новачків, але дуже функціональний для досвідчених розробників;

— багато вбудованих інструментів для web-розробки;

— підтримує велику кількість плагінів;

— вбудована підтримка Git та інших систем контролю версій;

— спеціалізується на JavaScript, TypeScript, але підтримує й інші мови;

— висока продуктивність, але може вимагати потужнішого апаратного забезпечення;

— працює на Windows, macOS та Linux;

— платний продукт.

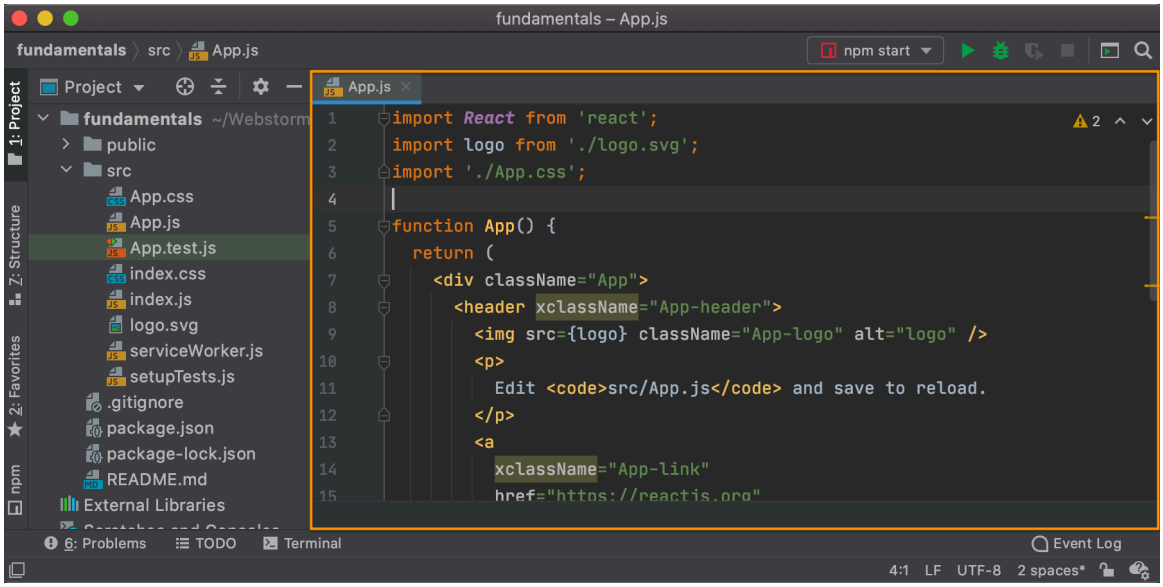


Рисунок 3.3 – графічний інтерфейс WebStorm

Таблиця 3.1 — Порівняння середовищ розробки

Характеристика	Visual Studio Code	Atom	WebStorm
Простота використання	9/10	8/10	8/10
Функціональність	9/10	7/10	9/10
Підтримка плагінів	10/10	8/10	9/10
Інтеграція з Git	10/10	7/10	9/10
Підтримка мов програмування	9/10	8/10	8/10
Продуктивність	8/10	7/10	9/10
Кросплатформеність	10/10	9/10	8/10
Віртість	Безкоштовно	Безкоштовно	Платний
Підсумок	65/70	54/70	60/70

Для нашого проекту ми обрали Visual Studio Code з наступних причин:

— інтуїтивно зрозумілий інтерфейс та зручність у використанні дозволяють швидко розпочати роботу навіть новачкам;

— велика кількість доступних розширень дозволяє налаштувати VS Code під будь-які потреби проекту, включаючи інтеграцію з Firebase, що є ключовим для нашої системи управління товарами;

— величезний каталог плагінів забезпечує можливість додавання необхідних функцій без зміни середовища розробки;

— вбудована підтримка Git дозволяє легко контролювати версії коду та співпрацювати в команді;

— підтримка широкого спектру мов програмування робить VS Code універсальним інструментом для нашого проекту, який включає використання JavaScript та інших технологій;

— висока продуктивність навіть на середньому апаратному забезпеченні дозволяє ефективно працювати без затримок;

— підтримка Windows, macOS та Linux забезпечує зручність роботи на будь-якій операційній системі;

— безкоштовність VS Code є великим плюсом, особливо для невеликих команд та окремих розробників.

Таким чином, Visual Studio Code є оптимальним вибором для нашого проекту з точки зору простоти використання, функціональності, підтримки плагінів, інтеграції з Git та кросплатформеності. Це дозволяє ефективно працювати, швидко адаптуватися до нових вимог та технологій, а також забезпечує високу продуктивність на всіх етапах розробки.

3.3 Розробка структури графічного інтерфейсу

Графічний інтерфейс користувача (GUI) є ключовим елементом будь-якої сучасної інформаційної системи [21], оскільки саме через нього користувачі взаємодіють із програмним забезпеченням. Розробка структури графічного

інтерфейсу вимагає ретельного планування та врахування потреб користувачів, щоб забезпечити зручність, ефективність та інтуїтивну зрозумілість системи.

Графічний інтерфейс користувача — це візуальна частина програмного забезпечення, через яку користувачі можуть взаємодіяти з системою. Він складається з різних елементів, таких як кнопки, форми введення, таблиці, меню, вікна повідомлень тощо. Основною метою GUI є забезпечення зручного та ефективного способу виконання завдань, для яких призначена система.

Важливість графічного інтерфейсу [22] для користувача:

— добре спроектований графічний інтерфейс робить систему більш доступною та зручною для користувачів. Це особливо важливо для нових користувачів, які можуть швидко освоїтися та почати ефективно працювати з системою;

— інтуїтивний і логічно побудований інтерфейс дозволяє користувачам швидко знаходити необхідні функції та виконувати завдання з мінімальними зусиллями, що підвищує їхню продуктивність;

— зрозумілий і простий у використанні інтерфейс знижує ймовірність помилок, оскільки користувачі мають чіткі інструкції та інтуїтивно зрозумілі елементи управління;

— користувачі, які працюють з добре спроектованим інтерфейсом, більш задоволені роботою з системою, що може позитивно вплинути на їхню лояльність і ставлення до продукту.

На рисунку 3.4 представлено структурну схему головної сторінки. Головна сторінка включає наступні елементи інтерфейсу:

1) бічне меню;
2) логотип сайту;
3) навігація сайту;
4) іконка користувача;
5) ім`я користувача;
6) кнопка виходу із системи;
7) назва сторінки;

8) сума продажів за день;

9) сума прибутку за день;

10)сума товару на складі;

11)діаграма рентабельності;

12)форма швидкого додавання товару;

13)форма швидкого продажу товару.

Рисунок 3.4 — Структура головної сторінки

На рисунку 3.5 представлено структурну схему сторінки товарів. Сторінка товарів включає наступні елементи інтерфейсу:

1) бічне меню;

2) логотип сайту;

3) навігація сайту;

4) іконка користувача;

5) ім`я користувача;

6) кнопка виходу із системи;

7) назва сторінки;

8) фільтри товарів;

9) список товарів;

10)кнопка додавання товару.

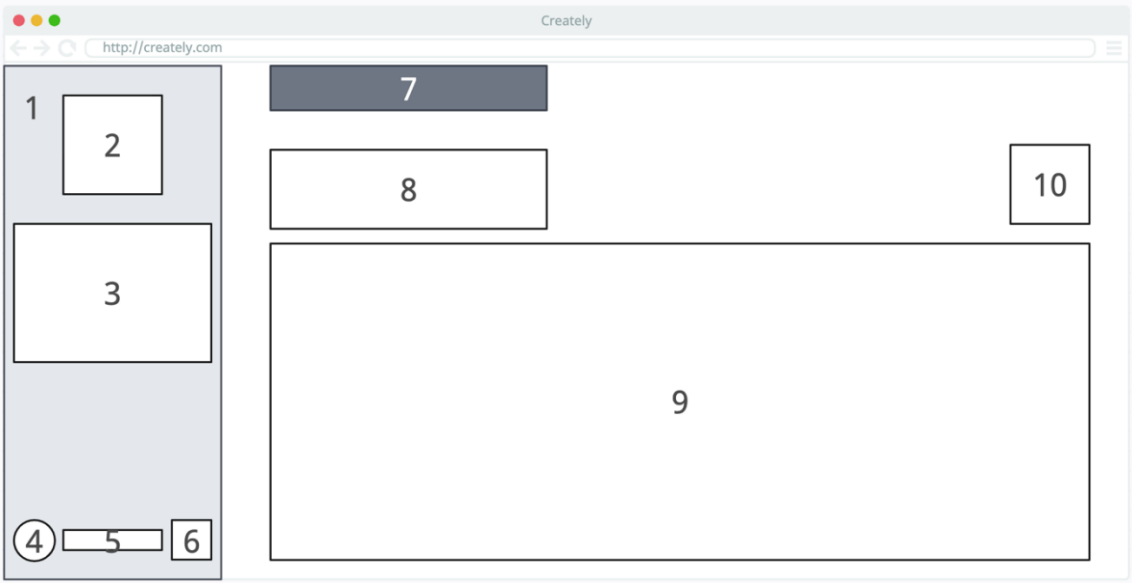


Рисунок 3.5 — Структура сторінки товарів

Кнопка додавання товару, відкриває модальне вікно з формою додавання товару (рисунок 3.6). Модальне вікно включає наступні елементи:

1) назва модального вікна;
2) кнопка закриття модального вікна;
3) форма введення даних товару.

Структурна схема сторінки звіту (рисунок 3.7) ідентична до сторінки товарів, за виключенням кнопки додавання товару.

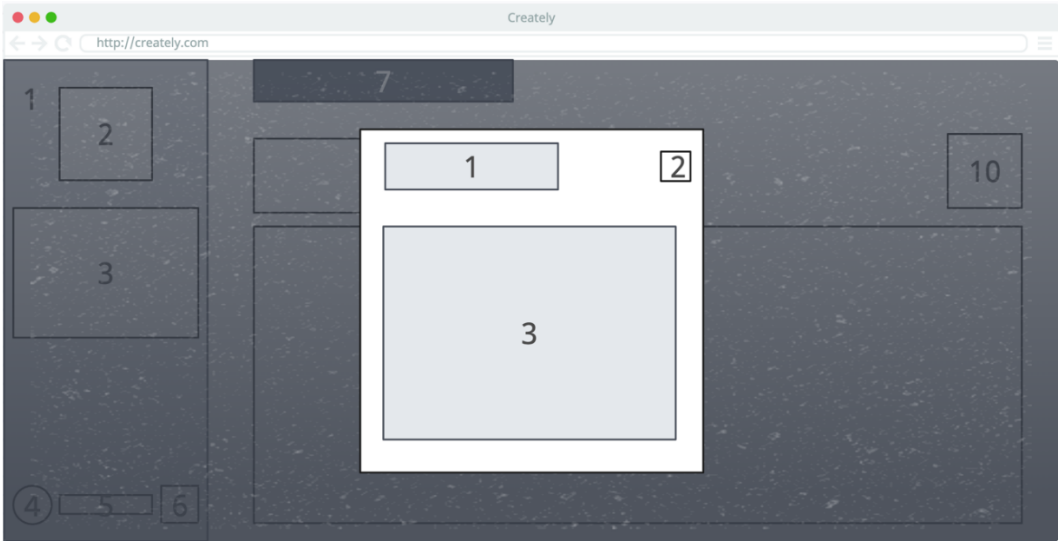


Рисунок 3.6 — Структурна схема модального вікна додавання товару

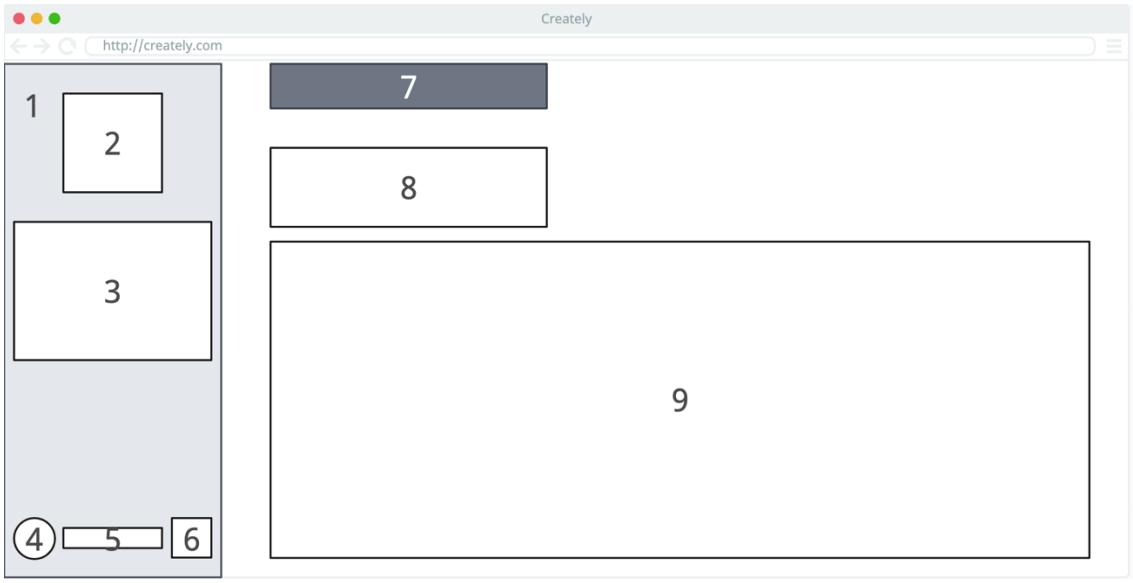


Рисунок 3.7 — Структурна схема сторінки звіту

На рисунку 3.8 зображена структурна схема сторінки авторизації. Вона включає такі елементи:

1) назва сторінки;
2) форма авторизації;
3) кнопка для авторизації.

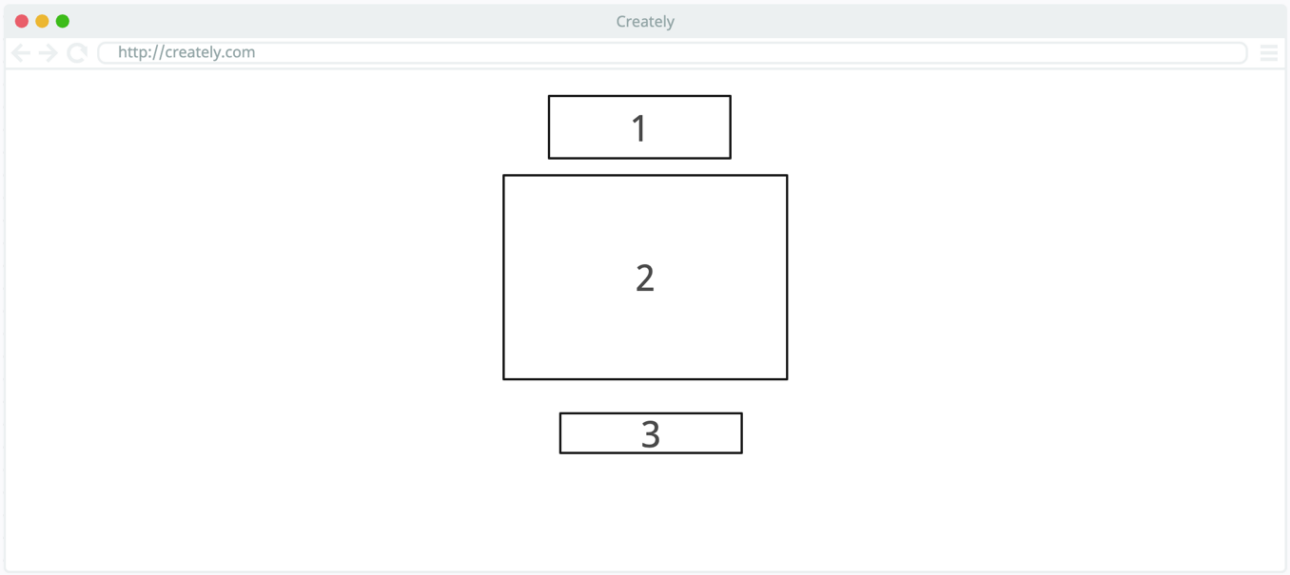


Рисунок 3.8 — Структурна схема сторінки авторизації

Після розробки структури сторінок сайту, було розроблено дизайн сторінок сайту (дизайн сторінок сайту наведено на рисунках Д.1-Д.6).

Отже, добре спроектований інтерфейс забезпечує зручність та ефективність роботи користувачів, підвищує продуктивність та зменшує кількість помилок.

Процес розробки включає визначення вимог користувачів, створення прототипів, розробку макетів, реалізацію інтерфейсу, тестування та вдосконалення. Основні елементи інтерфейсу включають головну панель навігації, форми введення даних, таблиці та списки, кнопки дій та вікна повідомлень.

3.4 Програмна реалізація сайту

На цьому етапі було розроблено функціональний продукт [23]. Програмна реалізація включає в себе розробку, інтеграцію з базою даних, створення логіки додатку. У цьому розділі розглянемо деталі програмної реалізації сайту для управління товарами, включаючи архітектурні рішення та основні етапи розробки.

Перший крок це створенні базової структури проекту Vue.js за допомогою інструменту Vue CLI. Цей крок включає в себе встановлення необхідних бібліотек, налаштування конфігурації та створення заготовок компонентів. Для створення проекту було запущено команду «vue create skladodrom» у командному рядку, де після створення отримали успішне повідомлення про створення (рисунок 3.9).

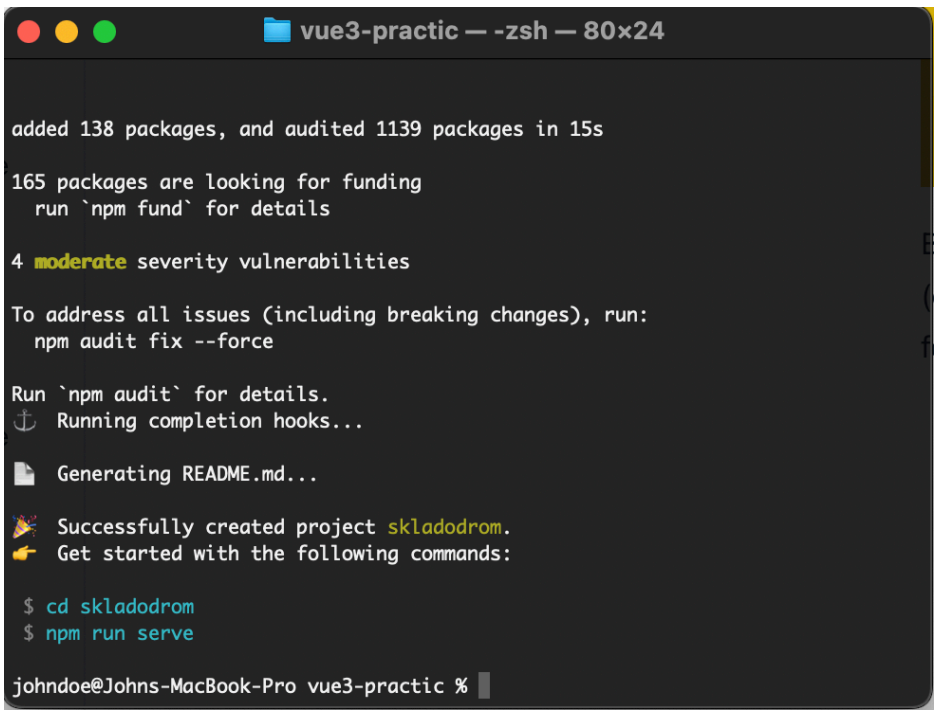


Рисунок 3.9 — Результат виконання команди

Після створення проекту потрібно добавити графічну бібліотеку «Vuetify» за допомогою команди «npm i vuetify» і підключити в проект в файлі main.js (лістинг коду наведено нижче).

Лістинг підключення бібліотеки в проект.

```
import 'vuetify/styles'
import { createVuetify } from 'vuetify'
import * as components from 'vuetify/components'
import * as directives from 'vuetify/directives'
const vuetify = createVuetify({
  components,
  directives,
})
createApp(App).use(vuetify).use(store).use(router).mount("#app");
```

Наступним кроком буде створення структури файлів проекту (рисунок Б.1) та ієрархічної структури сайту (рисунок Г.1).

Далі створено всі потрібні компоненти для сайту (форми, бічна панель та інше).

Після цих дій, можна розпочинати програмну реалізацію функцій системи. Першим кроком було реалізовано вхід у систему [24]. Щоб реалізувати вхід та реєстрацію у систему, потрібно налаштувати базу даних. Після налаштування було надано API ключ для доступу до бази даних і записано його до конфігураційного файлу «.env» під назвою «VITE_FB_KEY». Тепер ми можемо використовувати цю зміну у проекті.

Для авторизації, було розроблено наступну функцію, лістинг коду якої наведено нижче лістинг функції авторизації

```
async login({ commit }, val) {
```

```
        const url =
        `https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?
        key=${import.meta.env.VITE_FB_KEY}`;
        const {data} = await axios.post(url, {...val, returnSecureToken: true });
        commit('setToken', data.idToken);
        console.log(data);
    }
```

В цій функції спочатку сформовано запит до бази даних з API ключем, після чого за допомогою бібліотеки «axios», було відправлено запит методом «POST» з даними, які ввів користувач. Якщо все пройшло успішно, сервер поверне нам об`єкт «data», в якого присутній метод «idToken», за допомогою якого, база даних розуміє, який користувач увійшов в систему. Функція реєстрації майже так, як і входу, за винятком адреси запиту, лістинг функції реєстрації наведено нижче.

```
async register({ commit }, val) {
        const url =
        `https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=${import.
        meta.env.VITE_FB_KEY}`;
        const {data} = await axios.post(url, {...val, returnSecureToken: true });
        commit('setToken', data.idToken);
        console.log(data);
    }
```

Одною з головних функцій системи є можливість додавання товарів до бази даних. Це важлива операція, яка дозволяє користувачам системи додавати нові товари для подальшого управління ними. Процес додавання товарів є ключовим.

Основні кроки для додавання товару до бази даних включають наступне:

Введення інформації про товар — це користувач повинен мати можливість ввести всю необхідну інформацію про товар, таку як назва, опис, ціна, категорія,

виробник тощо. Ця інформація дозволяє системі правильно ідентифікувати та організувати товари у базі даних.

Перевірка коректності даних — перед збереженням нового товару до бази даних, система повинна перевірити коректність введених даних. Це включає перевірку формату email, правильність заповнення обов'язкових полів, валідацію [25] числових значень тощо. Ця перевірка допомагає уникнути помилок та зберегти консистентність даних у базі.

Збереження даних у базі — після успішної перевірки коректності даних, інформація про товар зберігається у базі даних. Це може включати створення нового запису в таблиці бази даних або оновлення існуючого запису.

Повідомлення про успішне додавання —після успішного збереження товару до бази даних, користувачу надсилається повідомлення про успішне додавання. Це може бути підтвердженням у вигляді повідомлення на екрані або електронною поштою.

Оновлення інтерфейсу користувача — щоб користувач міг побачити доданий товар у списку товарів, інтерфейс користувача оновлюється для відображення нового запису. Це може включати оновлення списку товарів або додавання нового товару у відповідну категорію.

Розглянемо реалізацію функції збереження даних у баз, лістинг функції збереження наведено нижче.

```
async create({ commit }, val) {
try {
    const token = store.getters['auth/token'];
const {data} = await axios.post(`https://skladodrom-1-default-
rtdb.firebaseio.com/products.json?auth=${token}`, val);
  } catch (error) {
    alert(error.message);
  }
}
```

Далі було реалізовано функцію продажу товару.

Алгоритм наступний:

— введення назви товару та кількості;

— перевірка введення коректних даних;

— оновлення запису товару в Firebase лістинг функції оновлення запису товару наведено нижче.

```
async sellProduct({ commit }, val, newPcs) {
 try {
        const token = store.getters['auth/token']
        const {data} = await axios.post(`https://skladodrom-1-default-
rtdb.firebaseio.com/products/${val.id}.json?auth=${token}`, {...val, prc: newPcs}
       );
        const req = Object.keys(data).map(id => ({...data[id], id}));
        commit('setLogs', req)
 } catch (error) {
    alert(error.message)
 }
},
```

Виведення товарів на сторінці з товарами. Для реалізації такого функціоналу зробимо наступні дії:

— при переході на сторінку товарів робимо запит до бази даних на отримання всіх товарів (лістинг функції зображено нижче);

— після отримання, виведемо їх у таблиці за допомогою директиви «v-for».

```
async getP({ commit }) {
  const token = store.getters['auth/token']
  const {data} = await axios.get(`https://skladodrom-1-default-
rtdb.firebaseio.com/products.json?auth=${token}`);
  const req = Object.keys(data).map(id => ({...data[id], id}));
```

```
    commit('setRequest', req)
    console.log(req);
},
```

Для видалення товару, було розроблено наступну функцію видалення, лістинг якої наведено нижче..

```
async deleteL({ commit },id) {
  const token = store.getters['auth/token']
  const {data} = await axios.delete(`https://skladodrom-1-default-rtdb.firebaseio.com/logs/${id}.json?auth=${token}`);
  console.log(data);
},
```

Для виходу з системи, було реалізовано функцію виходу із системи, лістинг якої наведено нижче.

```
logout( state ) {
        state.token = null;
        localStorage.removeItem('jwt-token');
}
```

Отже, в процесі програмної реалізації сайту було виконано кілька важливих етапів, які забезпечили створення функціонального і зручного web-додатку для управління та автоматизації товарообігу.

Першим етапом було ініціалізовано проект, що включало встановлення необхідних інструментів і налаштування робочого середовища. Використання сучасних технологій, таких як Vue.js для фронтенду та Firebase для бекенду, дозволило створити міцну основу для розробки. Інтеграція Vuetify забезпечила сучасний та привабливий інтерфейс користувача, що відповідає принципам Material Design.

Наступним кроком було створення структури проекту. Компонентний підхід у Vue.js дозволив розділити додаток на логічні модулі, кожен з яких відповідає за окрему функцію системи. Це значно спростило розробку, тестування та подальше обслуговування проекту. Створення чіткої ієрархії компонентів забезпечило зручність навігації та використання системи.

Основну увагу було приділено реалізації ключових функцій сайту. Зокрема, було створено модулі для управління товарами та інші функції. Кожна функція була ретельно перевірена для забезпечення коректної роботи і відповідності вимогам користувачів. Реалізація CRUD-операцій для роботи з базою даних дозволила забезпечити ефективне управління інформацією про товари та замовлення.

Загалом, програмна реалізація сайту завершена успішно. Створений web-додаток відповідає всім поставленим вимогам, має зручний інтерфейс та забезпечує ефективне управління товарообігом. Реалізація основних функцій дозволила створити потужний інструмент для автоматизації бізнес-процесів, що значно спрощує роботу користувачів та підвищує продуктивність.

Подальші кроки передбачають постійне вдосконалення системи на основі зворотного зв’язку від користувачів, впровадження нових функцій та оптимізацію продуктивності для забезпечення максимальної ефективності і задоволеності користувачів.

4 РОЗРОБКА ІНСТРУКЦІЇ КОРИСТУВАЧА

4.1 Основні вимоги до інструкції користувача

Інструкція користувача є невід'ємною частиною будь-якої програмної системи, оскільки вона допомагає користувачам ефективно використовувати програмне забезпечення, розуміти його функціональні можливості та вирішувати потенційні проблеми. Щоб інструкція була корисною та зручною, вона повинна відповідати кільком основним вимогам [26], таким як зрозумілість, повнота, зручність, ілюстративність та актуальність. Детально розглянемо кожну з цих вимог.

Зрозумілість є однією з найважливіших вимог до інструкції користувача. Текст інструкції повинен бути написаний простою і зрозумілою мовою, щоб користувачі могли легко сприймати і засвоювати інформацію. Використання складної технічної термінології слід уникати, якщо це можливо, або пояснювати такі терміни в тексті інструкції. Кожен термін і поняття, які можуть бути незрозумілими для користувачів, повинні мати чіткі пояснення.

Перш ніж починати написання інструкції, важливо визначити цільову аудиторію. Залежно від рівня знань і досвіду користувачів, текст може бути адаптований під їхні потреби. Наприклад, для новачків інструкція повинна містити більш детальні пояснення та крок за кроком інструкції, тоді як для досвідчених користувачів можна обмежитися короткими описами функцій.

Інструкція користувача повинна охоплювати всі основні функції системи та надавати детальні інструкції з їх використання. Це включає опис кожного розділу та функції програмного забезпечення, крок за кроком інструкції для виконання основних завдань, а також вирішення типових проблем, з якими можуть зіткнутися користувачі. Важливо, щоб користувачі могли знайти в інструкції відповіді на всі свої питання, пов'язані з використанням системи.

Крім того, інструкція повинна включати розділи з описом налаштувань системи [27], безпеки даних та управління обліковим записом. Також варто передбачити розділи з вирішення типових проблем. Зручність використання

інструкції є критично важливою для забезпечення позитивного користувацького досвіду. Інструкція повинна бути добре структурована, з чітким розподілом на розділи та підрозділи. Кожен розділ повинен мати логічну і послідовну структуру, що дозволить користувачам швидко знаходити необхідну інформацію. Зміст інструкції повинен включати посилання на всі основні розділи, що забезпечує легкий доступ до потрібної інформації.

Використання візуальних елементів, таких як скріншоти, діаграми, графіки та інші ілюстрації, значно покращує сприйняття інформації і допомагає користувачам краще зрозуміти текстові інструкції. Скріншоти можуть показувати, як виглядає інтерфейс системи на різних етапах використання, що робить інструкцію більш наочною і зрозумілою.

Крім скріншотів, можна використовувати діаграми для пояснення складних процесів або структури системи. Наприклад, діаграма може показувати взаємодію між різними компонентами системи або процес виконання певної функції. Використання кольорів і графічних елементів допомагає виділити важливу інформацію і звернути увагу користувачів на ключові моменти.

Інструкція користувача повинна бути актуальною і відображати поточний стан системи. З часом програмне забезпечення може змінюватися, додаватися нові функції або змінюватися існуючі. Тому важливо регулярно оновлювати інструкцію, щоб вона відповідала поточній версії програмного забезпечення.

Процес оновлення інструкції повинен бути організований таким чином, щоб зміни в системі оперативно відображалися в тексті інструкції. Це може включати перегляд і редагування існуючих розділів, додавання нових розділів для нових функцій та видалення застарілої інформації. Крім того, слід передбачити механізми для повідомлення користувачів про оновлення інструкції, щоб вони завжди мали доступ до актуальної інформації.

Інтерактивні інструкції стають все більш популярними, оскільки вони забезпечують більш зручний і ефективний спосіб отримання інформації. Інтерактивні інструкції можуть включати відеоуроки, інтерактивні посібники та

демонстрації, які дозволяють користувачам взаємодіяти з інструкцією в режимі реального часу.

Інструкція користувача повинна бути доступною для всіх користувачів, включаючи людей з обмеженими можливостями. Це може включати підтримку різних мов, використання зрозумілих шрифтів і кольорових схем, а також можливість доступу до інструкції через різні пристрої та платформи.

Підтримка різних мов дозволяє користувачам вибирати мову, яку вони найкраще розуміють, що підвищує ефективність використання інструкції. Використання зрозумілих шрифтів і кольорових схем допомагає забезпечити читабельність тексту для всіх користувачів, включаючи людей з вадами зору. Можливість доступу до інструкції через різні пристрої, такі як комп'ютери, планшети та смартфони, забезпечує зручність використання інструкції в будь-який час і в будь-якому місці.

Отже, інструкція користувача повинна відповідати кільком основним вимогам [28], таким як зрозумілість, повнота, зручність, ілюстративність, актуальність, інтерактивність, доступність і контроль якості. Виконання цих вимог забезпечує високу якість інструкції, що допомагає користувачам ефективно використовувати програмне забезпечення, розуміти його функціональні можливості та вирішувати потенційні проблеми. Інструкція користувача є важливим інструментом для забезпечення позитивного користувацького досвіду і задоволення потреб користувачів.

4.2 Інструкція користувача

Для роботи з web-платформою вам знадобиться комп'ютер або мобільний пристрій з доступом до Інтернету та web-браузер, що підтримує сучасні web-технології, такі як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge [29]. Крім того, вам потрібно створити обліковий запис користувача, що можна зробити під час першого входу на платформу.

Для реєстрації на платформі, потрібно виконати наступні кроки:

1) відкрийте web-браузер та перейдіть на сайт;

2) натисніть кнопку «Реєстрація» на сторінці;
3) заповніть форму реєстрації (рисунок 4.1), вказавши свою електрону пошту, пароль та підтвердження паролю;
4) натисніть кнопку «Зареєструватись».

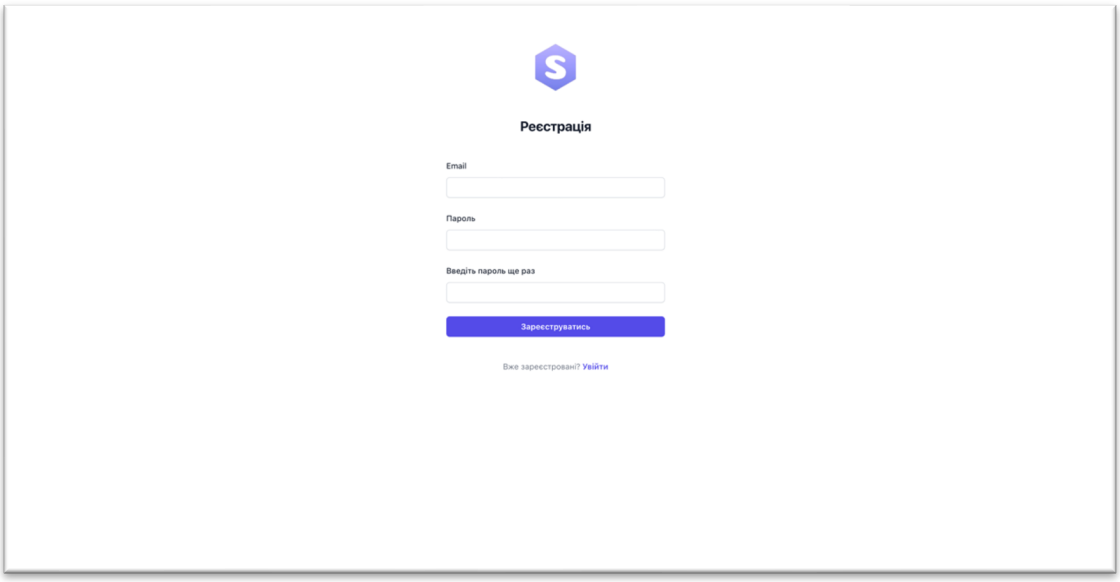


Рисунок 4.1 — Форма реєстрації облікового запису

Для входу в систему, потрібно виконати наступні кроки:

1) відкрийте web-браузер та перейдіть на сайт;
2) натисніть кнопку «Вхід» на сторінці;
3) заповніть форму (рисунок 4.2), вказавши електрону пошту та пароль;
4) натисніть кнопку «Вхід».

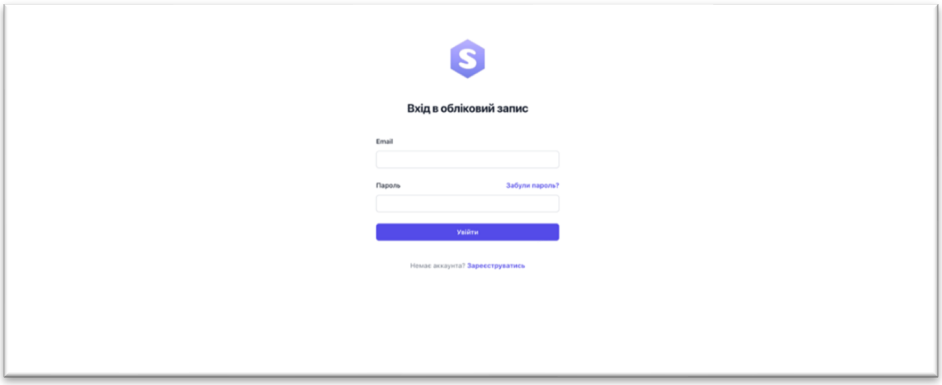


Рисунок 4.2 — Форма входу в систему

Після створення облікового запису, усі функції системи будуть доступні, зокрема, функції управління товарами, розглянемо їх.

Функція додавання нового товару. Є два варіанти додавання, швидке додавання (за допомогою форми на головній сторінці) та розширене додавання. Розглянемо варіант швидкого додавання:

1) увійдіть до свого облікового запису;
2) перейдіть на головну сторінку;
3) за допомогою відповідної форми (форму виділено на рисунку 4.3) ввести обов`язкову інформацію;
4) натиснути кнопку «Додати»;

Для додавання товару з можливістю введення додаткової інформації, потрібно виконати наступні кроки:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Товари»;
3) натисніть кнопку «+»;
4) у відкритому вікні заповніть форму та введіть необхідну інформацію (рисунок 4.4);
5) натисніть кнопку «Додати».

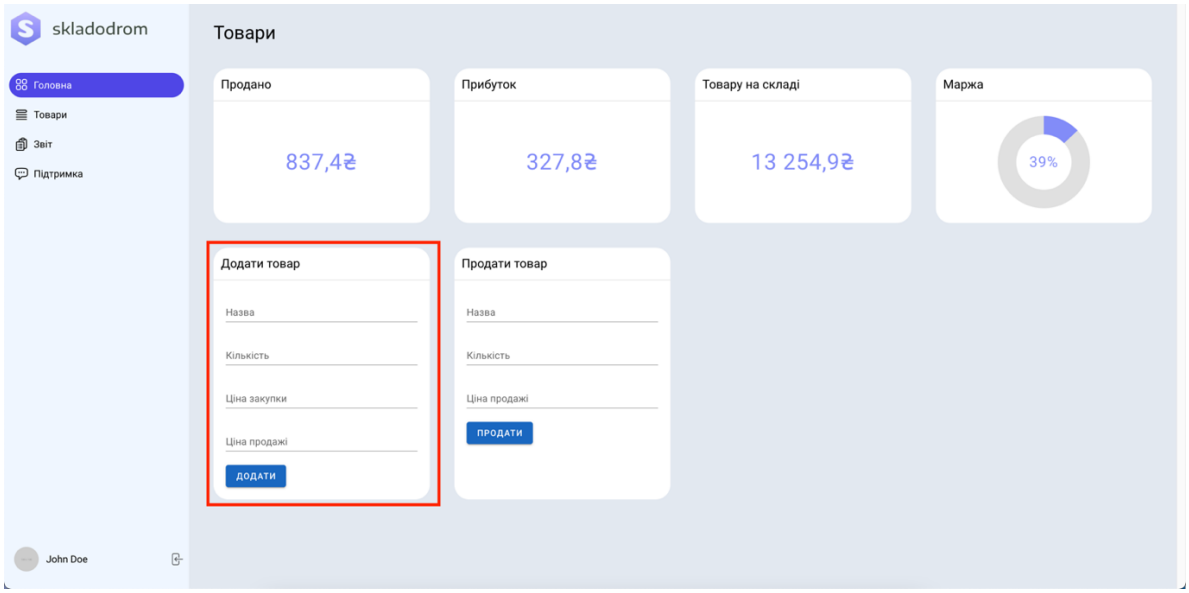


Рисунок 4.3 — Форма швидкого додавання товару

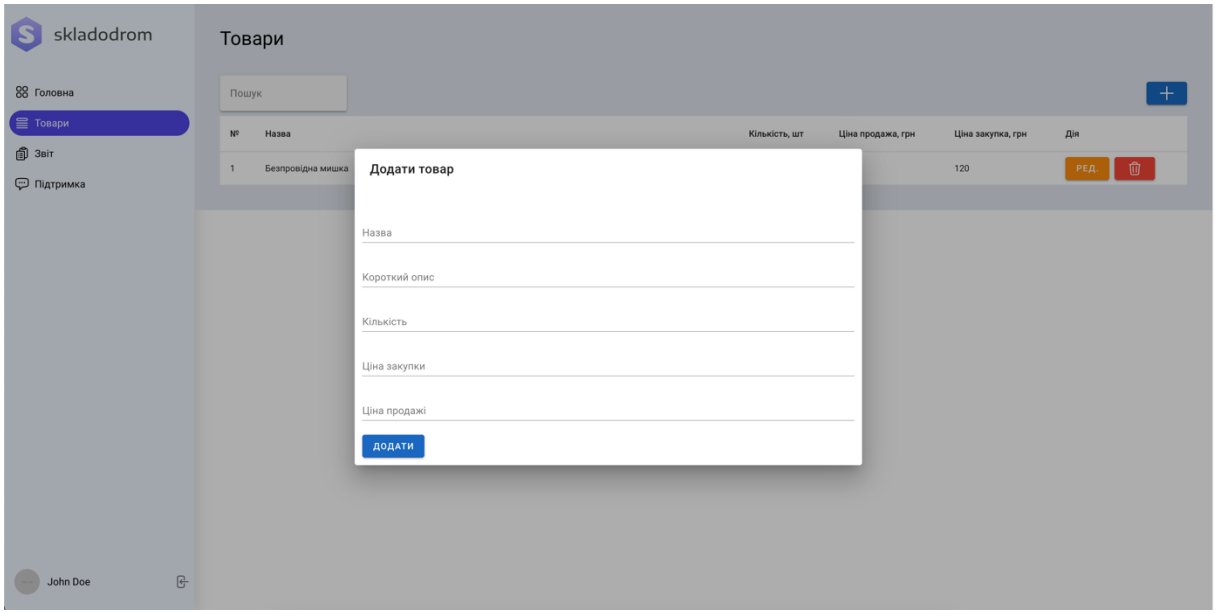


Рисунок 4.4 — Форма додавання товару з введенням детальної інформації

У системі передбачено можливість редагування. Для редагування існуючого товару, потрібно виконати наступні кроки:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Товари»;
3) оберіть товар зі списку, який ви хочете відредагувати, і натисніть кнопку з назвою «Ред.»;
4) введіть необхідні зміни у формі редагування (рисунок 4.5), яка з`явилась;
5) натисніть кнопку «Зберегти».

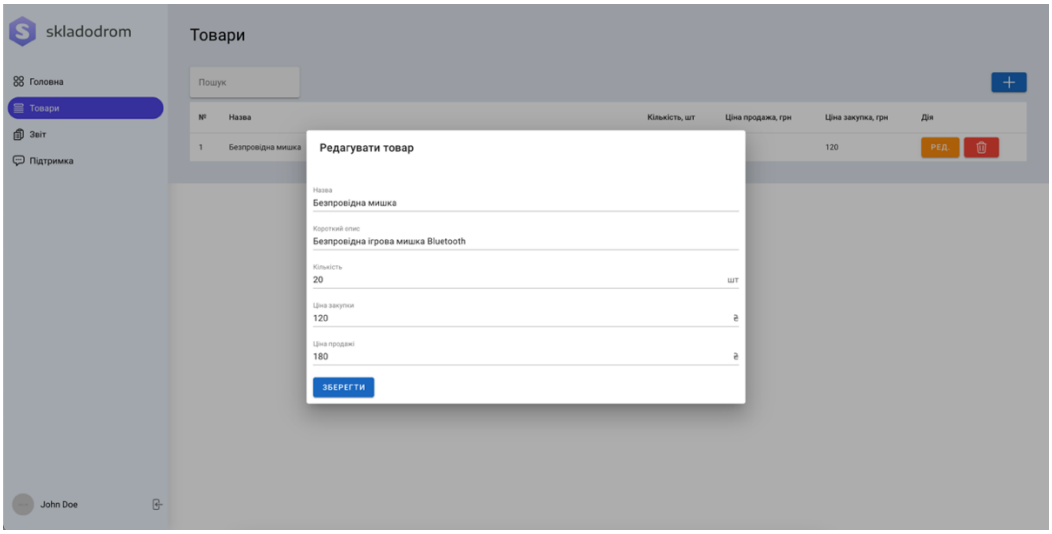


Рисунок 4.5 — Форма редагування товару

Для перегляду деталей товару, потрібно:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Товари»;
3) оберіть та натисніть на товар зі списку, який ви хочете переглянути;
4) в вікні яке відкрилось, ви можете переглянути всю інформацію.

Для видалення існуючого товару, необхідно виконати наступні кроки:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Товари»;
3) виберіть необхідний товар зі списку;
4) натисніть кнопку з зображенням кошику (кнопку виділено на рисунку 4.6);
5) підтвердіть видалення.

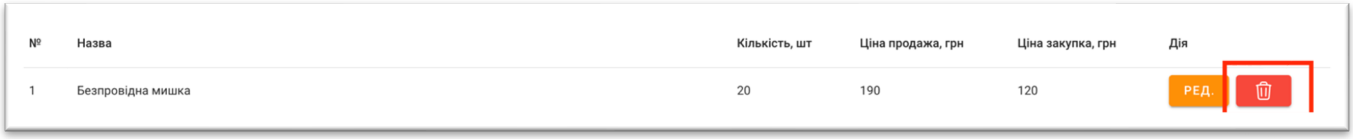


Рисунок 4.6 – кнопка видалення товару

Щоб продати товар, потрібно:

1) увійдіть до системи;
2) перейдіть на головну сторінку;
3) введіть у форму під назвою «Продати» необхідну інформацію (рисунок 4.7), таку як назва, кількість та ціну продажу;
4) натисніть кнопку «Продати».

Для перегляду звіту з продажів, потрібно:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Звіт»;
3) перегляньте необхідну інформацію (рисунок 4.8);

Для звернення до служби підтримки, потрібно виконати наступні кроки:

1) увійдіть до свого облікового запису;
2) перейдіть до розділу «Підтримка»;

3) заповніть форму звернення до служби підтримки вказавши необхідну інформацію (рисунок 4.9);

4) натисніть кнопку «Надіслати»;

5) відповідь на ваше питання буде надіслано на електрону пошту, яку було вказано при реєстрації.

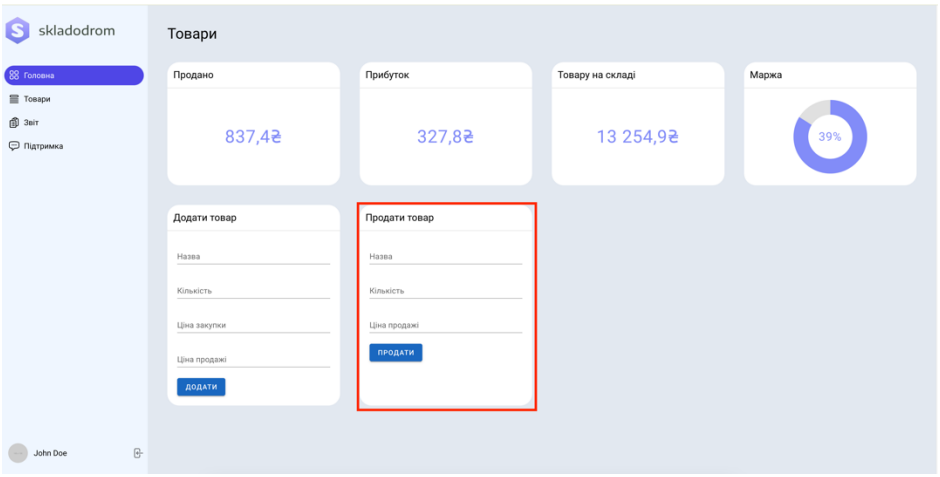


Рисунок 4.7 – форма продажу товару

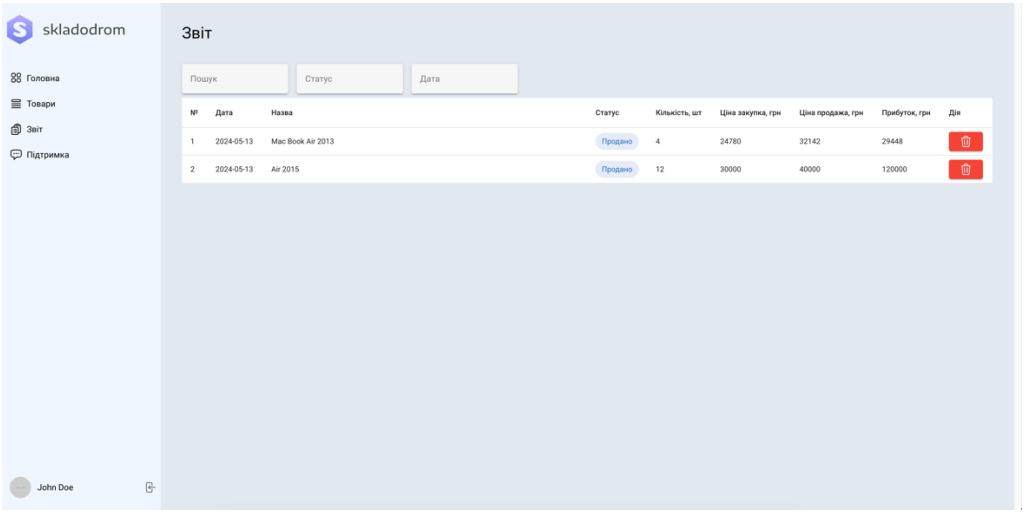


Рисунок 4.8 – перегляд звіту з продажів

Для виходу із системи, потрібно:

1) перейдіть на будь яку сторінку сайту;

2) в бічному меню натисніть кнопку з відповідним зображенням (кнопку виділено на рисунку 4.10).

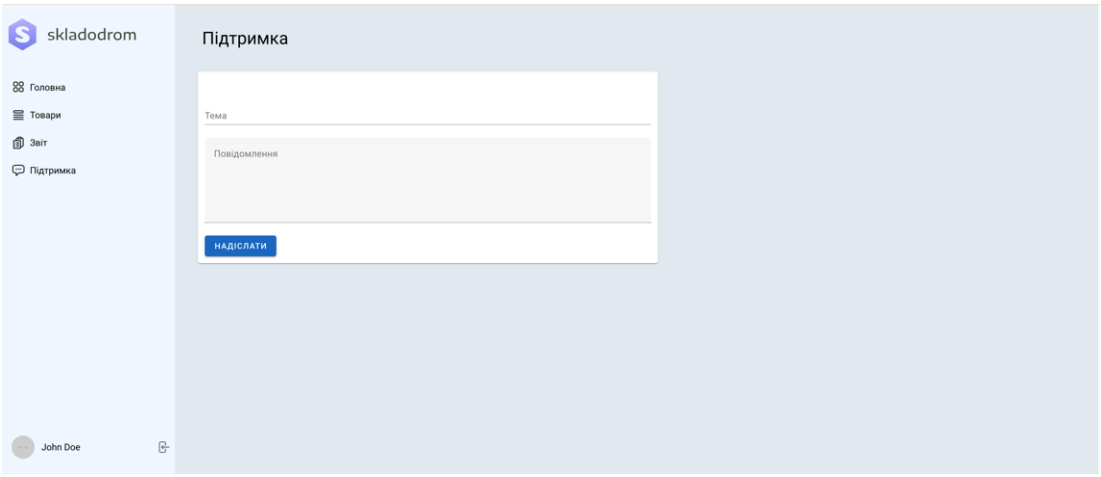


Рисунок 4.9 — Форма звернення до служби підтримки

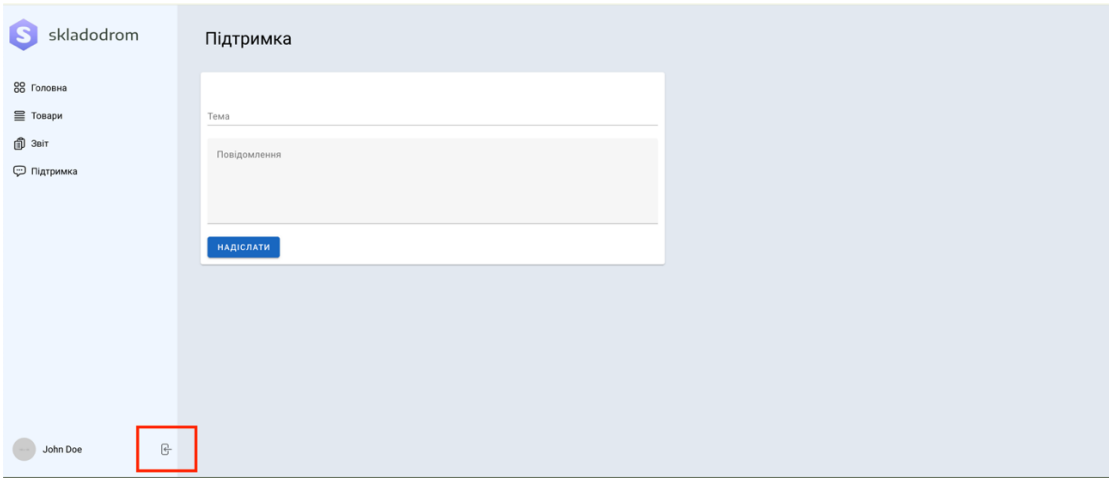


Рисунок 4.10 — Кнопка виходу із системи

Ця інструкція [30] допоможе легко і ефективно використовувати web-платформу для управління та автоматизації товарообігу. Якщо у вас виникнуть питання або проблеми, будь ласка, звертайтеся до служби підтримки.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено web-платформу для управління та автоматизації товарообігу з використанням API під назвою «Trade Management Platform». Розроблене програмне забезпечення призначене для автоматизації процесів управління товарами, обробки замовлень та генерації звітів, що дозволяє підвищити ефективність бізнес-операцій і полегшити роботу кінцевих користувачів.

Проведено аналіз сучасних технологій для створення web-додатків, таких як Vue.js, Vuetify та Firebase. Розглянуто їхні переваги та обґрунтовано доцільність використання цих технологій для розробки платформи. Визначено ключові вимоги до системи та сформульовано мету і завдання дослідження.

Виконано розробку архітектури системи, включаючи проектування бази даних та реалізацію основних функціональних модулів. Система побудована на основі клієнт-серверної архітектури, де клієнтська частина реалізована за допомогою Vue.js і Vuetify, а серверна частина та база даних - за допомогою Firebase. Використання Firebase дозволило забезпечити безсерверну архітектуру, яка спрощує інфраструктуру проекту та забезпечує масштабованість і надійність.

Основні функції системи, такі як додавання, редагування, видалення товарів, управління замовленнями та генерація звітів, були реалізовані та ретельно протестовані. Для кожної функції розроблено алгоритми та блок-схеми, які детально описують процеси взаємодії між компонентами системи та зберігання даних.

Проведено варіантний аналіз і обґрунтування вибору програмних засобів для розробки платформи. Використання Vue.js та Vuetify дозволило створити сучасний та інтуїтивно зрозумілий інтерфейс користувача, що відповідає принципам Material Design. Firebase забезпечив надійне зберігання даних та ефективну роботу з базою даних у режимі реального часу.

На завершення було розроблено інструкцію користувача, яка детально описує процеси реєстрації, входу в систему, управління товарами, замовленнями та

генерації звітів. Інструкція містить покрокові інструкції та візуальні матеріали, що полегшують користувачам розуміння і використання системи.

Таким чином, розроблена web-платформа відповідає вимогам та завданням, поставленим у рамках бакалаврської дипломної роботи. Вона забезпечує ефективне управління товарообігом, автоматизацію ключових бізнес-процесів і зручний інтерфейс для кінцевих користувачів. Платформа готова до подальшого вдосконалення та масштабування на основі зворотного зв’язку від користувачів та впровадження нових функцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nielsen Norman Group. How to Conduct User Research: webсайт. URL: https://www.nngroup.com/articles/how-to-conduct-user-research/.

2. Interaction Design Foundation. User Needs and Requirements: webсайт. URL: https://www.interaction-design.org/literature/topics/user-needs.

3. Gartner. Magic Quadrant for Digital Commerce: webсайт. URL: https://www.gartner.com/en/documents/3983887/magic-quadrant-for-digital-commerce.

4. Forrester. The Forrester Wave: B2B Commerce Suites: webсайт. URL: Cypress Documentation: webсайт. URL: https://docs.cypress.io/guides/overview/why-cypress.

5. Supply Chain Management Review. Inventory Management Best Practices: webсайт. URL: https://www.scmr.com/article/inventory_management_best_practices.

6. APICS. The Essential Guide to Inventory Management: webсайт. URL: https://www.apics.org/apics-for-individuals/publications/apics-magazine-home.

7. Elsevier. How to Write a Research Proposal: webсайт. URL: https://www.elsevier.com/connect/authors-update/how-to-write-a-research-proposal.

8. Springer. Research Design and Methods: webсайт. URL: https://www.springer.com/gp/book/9783030489661.

9. Mozilla Developer Network. Web APIs: webсайт. URL: https://developer.mozilla.org/en-US/docs/Web/API .

10. Microsoft. API Design and Architecture: webсайт. URL: https://docs.microsoft.com/en-us/azure/architecture/api-design.

11. Agile Alliance. What is Agile Software Development?: webсайт. URL: https://www.agilealliance.org/agile101.

12. Scrum.org. What is Scrum?: webсайт. URL: https://www.scrum.org/resources/what-is-scrum.

13. GeeksforGeeks. Introduction to Algorithms: webсайт. URL: https://www.geeksforgeeks.org/fundamentals-of-algorithms/ .

14. MIT OpenCourseWare. Introduction to Algorithms: webсайт. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/.

15. Oracle. Generating Reports Using SQL: webсайт. URL: https://docs.oracle.com/cd/B28359_01/server.111/b31189/rptsql.htm .

16. SAP. Report Generation and Formatting: webсайт. URL: https://help.sap.com/viewer/0fa7020b70f940bda42aeb2020d9fe62/1809.000/en-US/243f7e388b0d4db1a5b8baba0b95c657.html.

17. tack Overflow. Developer Survey Results: webсайт. URL: https://insights.stackoverflow.com/survey/2020 .

18. GitHub. Octoverse: webсайт. URL: https://octoverse.github.com/.

19. JetBrains. IntelliJ IDEA: webсайт. URL: https://www.jetbrains.com/idea/ .

20. Microsoft. Visual Studio Code: webсайт. URL: https://code.visualstudio.com/.

21. Google. Material Design: webсайт. URL: https://material.io/design .

22. Smashing Magazine. Principles of User Interface Design: webсайт. URL: https://www.smashingmagazine.com/2018/03/principles-good-micro-interactions/.

23. Mozilla Developer Network. HTML Basics: webсайт. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics .

24. Mozilla Developer Network. CSS Basics: webсайт. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics .

25. Mozilla Developer Network. JavaScript Basics: webсайт. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics.

26. ISO 9241-11:2018. Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts: webсайт. URL: https://www.iso.org/standard/63500.html.

27. Usability.gov. Developing User Assistance: webсайт. URL: https://www.usability.gov/how-to-and-tools/methods/developing-user-assistance.html.

28. Atlassian. How to Create Great User Documentation: webсайт. URL: https://www.atlassian.com/software/confluence/user-documentation.

29. Google Developers. Writing Technical Documentation: webсайт. URL: https://developers.google.com/tech-writing/overview.

30. GitHub. Documenting Your Projects on GitHub: webсайт. URL: https://guides.github.com/features/wikis/.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.
«___»_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської дипломної роботи
«WEB-платформа для управління та автоматизації з використанням API»
08-54.БДР.017.00.000 ТЗ

Науковий керівник: к.т.н., доц.
каф. ОТ Муращенко О.Г.
Студент групи 2СП-20Б
Березовський О.С.

ВНТУ 2024

1 Підставою для виконання бакалаврської дипломної роботи (БДР)

1.1 Важливість дослідження у напрямку розробки web-платформи для управління та автоматизації товарообігу обумовлена зростаючою потребою у ефективному управлінні запасами та продажами в сучасному бізнес-середовищі. Сучасні підприємства, особливо малого та середнього бізнесу, потребують надійних інструментів для оптимізації своїх операцій та підвищення продуктивності. Створення web-платформи, яка не тільки забезпечує зручне управління товарами та продажами, але й надає можливість генерації звітів та отримання допомоги, має значний потенціал для покращення якості обслуговування клієнтів і оптимізації бізнес-процесів.

1.2 Наказ про затвердження теми БДР.

2 Мета БДР і призначення розробки

2.1 Мета роботи — розробка web-платформи для управління та автоматизації товарообігу з використанням API. Ця платформа забезпечує користувачам можливість реєстрації та входу, управління товарами (додавання, редагування, видалення), обробку продажів, генерацію звітів та отримання допомоги. Проект спрямований на створення інноваційного рішення для малого та середнього бізнесу, що підвищить ефективність управління бізнес-процесами та оптимізує роботу з товарами.

2.2 Web-платформа призначена для використання підприємствами, які займаються продажем товарів, для забезпечення ефективного управління запасами та продажами. Платформа надає зручний інтерфейс для користувачів, що дозволяє легко керувати товарами, обробляти продажі, генерувати звіти та отримувати підтримку. Це дозволить бізнесам швидко реагувати на зміни ринку, покращувати обслуговування клієнтів та підвищувати продуктивність.

3 Вихідні дані для виконання БДР

3.1 Проведення аналізу існуючих рішень та визначення їхніх переваг і

недоліків здійснюється дослідження існуючих web-платформ для управління товарообігом. Аналізуються їхні функціональні можливості, переваги та недоліки, щоб визначити, які функції мають бути реалізовані у власному проекті.

3.2 Розробка архітектури платформи та вибір технологій — проектується архітектура системи, вибираються технології для її реалізації, включаючи Vue.js, Vuetify для створення інтерфейсу та Firebase. Визначаються основні компоненти системи та їх взаємодія.

3.3 Проектування бази даних —розробляється структура бази даних у Firestore, що включає таблиці для зберігання інформації про користувачів, товари, продажі та звіти. Визначаються зв'язки між таблицями та необхідні індекси для оптимізації запитів.

3.4 Розробка програмного забезпечення для управління товарами та продажами — створюються алгоритми для додавання, редагування та видалення товарів, а також для обробки продажів. Реалізуються функції генерації звітів, що дозволяють користувачам отримувати аналітичну інформацію про заробіток, виручку та рентабельність.

3.5 Оцінка перспектив впровадження платформи на ринок —визначається потенційна цільова аудиторія та можливості впровадження платформи на ринок. Аналізуються конкуренти, розробляється стратегія просування та підтримки продукту.

4 Вимоги до виконання БДР

Головна вимога – створення web-платформи, яка забезпечує ефективне управління товарообігом, автоматизацію бізнес-процесів та надає зручний інтерфейс для користувачів. Всі етапи розробки, включаючи проектування, реалізацію та тестування, мають бути виконані з дотриманням відповідних технічних норм і стандартів. Платформа повинна бути масштабованою, надійною та безпечною, забезпечуючи високу продуктивність і зручність у використанні.5 Етапи БДР та очікувані результати

5 Етапи БДР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 – Етапи БДР

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Очікувані результати
1	Аналіз завдання та вибір методу вирішення поставленої задачі	10.03.2024-17.03.2024	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ.
2	Опис основних функцій системи	18.03.2024-01.04.2024	Розділ 2
3	Розробка алгоритму додавання, видалення, редагування та продажу товару	02.04.2024-12.04.2024	Розділ 2
4	Розробка методу та алгоритму генерації звіту на основі отриманих данних	13.04.2024-22.04.2024	Розділ 2
5	Аналіз та вибір мови програмування та середовища розробки	23.04.2024-03.05.2024	Розділ 3
6	Програмна реалізація	04.05.2024-12.05.2024	Розділ 3
7	Інструкція користувача	13.05.2024-15.05.2024	Розділ 4
8	Оформлення матеріалів до захисту БДР	16.05.2024 20.05.2024	ПЗ, графіч. Матеріал, презентація

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКП українською та іноземною мовами, довідка про відповідність оформлення БКП діючим вимогам.

7 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БДР

8.1 При оформлювання БДР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Системне програмуваннння»). Кафедра обчислювальної техніки ВНТУ 2024;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БДР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Структура файлів проекту

```
├── src/
│   ├── assets/                  # Зображення, шрифти та інші ресурси
│   ├── components/              # Компоненти Vue.js
│   │   ├── DashboardWidget.vue  # Компонент віджетів
│   │   ├── TheSidebar.vue       # Компонент бокового меню
│   │   ├── Navbar.vue           # Компонент навігаційної панелі
│   │   └── ...
│   ├── router/                  # Конфігурація маршрутизатора Vue Router
│   │   └── index.js
│   ├── store/                   # Конфігурація та управління станом додатку Vuex
│   │   └── index.js
│   ├── layouts/                 # Макети сторінок
│   │   └── TheMainLayouts.vue
│   │   └── TheAuthLayouts.vue
│   ├── views/                   # Відображення та сторінки додатку
│   │   ├── ViewHome.vue         # Головна сторінка
│   │   ├── ViewProducts.vue     # Сторінка зі списком товарів
│   │   ├── ViewReport.vue       # Сторінка звіту
│   │   └── ...
│   ├── App.vue                  # Головний компонент додатку
│   └── main.js                  # Вхідний файл для налаштування додатку
├── public/                      # Статичні файли
├── .env                         # Файл конфігурації середовища
└── ...
```

Рисунок Б.1 — Структура файлів проекту

ДОДАТОК В

Лістинг

package.json

```
{
 "name": "skaldodrom",
 "private": true,
 "version": "0.0.0",
 "type": "module",
 "scripts": {
  "dev": "vite",
  "build": "vite build",
  "preview": "vite preview"
 },
 "dependencies": {
  "axios": "^1.6.8",
  "vue": "^3.4.21",
  "vue-router": "^4.3.2",
  "vuetify": "^3.6.4",
  "vuex": "^4.0.2"
 },
 "devDependencies": {
  "@vitejs/plugin-vue": "^5.0.4",
  "autoprefixer": "^10.4.19",
  "postcss": "^8.4.38",
  "tailwindcss": "^3.4.3",
  "vite": "^5.2.0"
 }
}
```

Index.html

```
<!doctype html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <link rel="icon" type="image/svg+xml" href="/vite.svg" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Vite + Vue</title>
 </head>
 <body>
  <div id="app"></div>
  <script type="module" src="/src/main.js"></script>
 </body>
</html>
```

src/main.js

```
import { createApp } from "vue";
import './style.css'
```

```
import App from "./App.vue";
import router from "./router";
import store from "./store";
import 'vuetify/styles'
import { createVuetify } from 'vuetify'
import * as components from 'vuetify/components'
import * as directives from 'vuetify/directives'


const vuetify = createVuetify({
  components,
  directives,
})

createApp(App).use(vuetify).use(store).use(router).mount("#app");
```

src/app.vue

```
<template>
  <component :is="'The' + activeLayout + 'Layouts'"></component>
</template>

<script>
import TheMainLayouts from './layouts/TheMainLayouts.vue'
import TheAuthLayouts from './layouts/TheAuthLayouts.vue'

export default {
  computed: {
    activeLayout() {
      return this.$route.meta.layout
    }
  },
  components: {
    TheMainLayouts,
    TheAuthLayouts
  }
}
</script>
```

src/views/ViewAuth.vue

```
<script>

  export default {
    components: {

    },
    data() {
      return {
        val: {
            email: '',
            password: '',
```

```
      },
    }
  },
  methods: {
    async onSubmit() {
      await this.$store.dispatch('auth/login', this.val);
      this.$router.push('/dashboard');
    }
  }
};
</script>

<template>
 <div class="font-sans flex min-h-full flex-col justify-center px-6 py-12 lg:px-8">
 <div class="sm:mx-auto sm:w-full sm:max-w-sm">
  <img class="mx-auto w-[80px]" src="../assets/logo-small.png" alt="Your Company">
  <h2 class="mt-10 text-center text-2xl font-bold leading-9 tracking-tight text-gray-900">Вхід в обліковий запис
</h2>
 </div>

 <div class="mt-10 sm:mx-auto sm:w-full sm:max-w-sm">
  <form @submit.prevent="onSubmit" class="space-y-6" action="" method="POST">
   <div>
    <label for="email" class="block text-sm font-medium leading-6 text-gray-900">Email</label>
    <div class="mt-2">
     <input v-model="val.email" id="email" name="email" type="email" autocomplete="email" required class="block w-full rounded-md border-0 p-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-600 sm:text-sm sm:leading-6">
    </div>
   </div>

   <div>
    <div class="flex items-center justify-between">
     <label for="password" class="block text-sm font-medium leading-6 text-gray-900">Пароль</label>
     <div class="text-sm">
      <a href="#" class="font-semibold text-indigo-600 hover:text-indigo-500">Забули пароль?</a>
     </div>
    </div>
    <div class="mt-2">
     <input v-model="val.password" id="password" name="password" type="password" autocomplete="current-password" required class="block w-full rounded-md border-0 p-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-600 sm:text-sm sm:leading-6">
    </div>
   </div>

   <div>
    <button  type="submit" class="flex w-full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-visible:outline focus-
```

```
visible:outline-2 focus-visible:outline-offset-2 focus-visible:outline-indigo-600
transition">Увійти</button>
     </div>
   </form>

   <p class="mt-10 text-center text-sm text-gray-500">
     Немає аккаунта?
     <router-link to="/reg" class="font-semibold leading-6 text-indigo-600 hover:text-indigo-
500">Зареєструватись</router-link>
   </p>
  </div>
</div>
</template>
```

src/views/ViewDashboard.vue

```
<template>
  <div class="p-10 bg-slate-200 overflow-auto h-full">
    <h2 class="text-3xl">Товари</h2>
    <main class="mt-10 grid grid-cols-4 gap-10 auto-rows-min bg-slate-200">
      <DashboardWidget title="Продано" content="837,4₴"></DashboardWidget>
      <DashboardWidget title="Прибуток" content="327,8₴"></DashboardWidget>
      <DashboardWidget
        title="Товару на складі"
        content="13 254,9₴"
      ></DashboardWidget>
      <DashboardWidget title="Маржа">
        <v-progress-circular
          class="text-2xl"
          color="rgb(129 140 248"
          :model-value="Math.round(Math.random() * 100)"
          :rotate="360"
          :size="150"
          :width="30"
        >
          <template v-slot:default> 39% </template>
        </v-progress-circular>
      </DashboardWidget>

      <DashboardWidgetForm title="Додати товар">
        <form @submit.prevent="add" class="p-3">
          <v-text-field
            label="Назва"
            model-value=""
            v-model="val.title"
            placeholder="Введіть назву товару"
            variant="underlined"
          ></v-text-field>
          <v-text-field
            label="Кількість"
            model-value=""
```

```
        placeholder="Введіть кількість"
        v-model="val.pcs"
        suffix="шт"
        variant="underlined"
      ></v-text-field>
      <v-text-field
        label="Ціна закупки"
        placeholder="Введіть ціну закупки товару"
        model-value=""
        v-model="val.priceZakup"
        suffix="₴"
        variant="underlined"
      ></v-text-field>
      <v-text-field
        label="Ціна продажі"
        placeholder="Введіть ціну продажі товару"
        model-value=""
        v-model="val.priceProdazh"
        suffix="₴"
        variant="underlined"
      ></v-text-field>
      <v-btn type="submit" class="" color="primary">Додати</v-btn>
    </form>
  </DashboardWidgetForm>
  <DashboardWidgetForm title="Продати товар">
    <form @submit.prevent="updateProduct" class="p-3">
      <v-select
        :items="request"
        label="Назва"
        variant="underlined"
      >
        <template v-slot:item="{ props, item }">
          <v-list-item v-bind="props"></v-list-item>
        </template>
      </v-select>
      <v-text-field
        label="Кількість"
        model-value=""
        placeholder="Введіть кількість"
        suffix="шт"
        variant="underlined"
      ></v-text-field>
      <v-text-field
        label="Ціна продажі"
        placeholder="Введіть ціну продажі товару"
        suffix="₴"
        variant="underlined"
      ></v-text-field>
      <v-btn type="submit" class="" color="primary">Продати</v-btn>
    </form>
  </DashboardWidgetForm>
</main>
```

```
  </div>
</template>

<script>
import DashboardWidget from "../components/DashboardWidget.vue";
import DashboardWidgetForm from "../components/DashboardWidgetForm.vue";

export default {
  data() {
    return {
      val: {
        title: "",
        pcs: "",
        priceZakup: "",
        date: new Date().toJSON().slice(0, 10),
        priceProdazh: "",
      }
    };
  },
  methods: {
    async add() {
      await this.$store.dispatch("request/create", this.val);

      this.val.title = "";
      this.val.pcs = "";
      this.val.priceZakup = "";
      this.val.priceProdazh = "";
    },

    async sell() {

      const product = await this.$store.dispatch("request/getProduct", this.val.id);

      await this.$store.dispatch("request/sellProduct", [this.val.id]);
    }
  },
  computed: {
    request() {
      console.log(this.$store.getters["request/getProd"]);
      return this.$store.getters["request/getProd"];
    }
  },
  async mounted() {
    await this.$store.dispatch("request/getP");
  },
  components: {
    DashboardWidget,
    DashboardWidgetForm,
  },
};
</script>
```

src/views/ViewHelp.vue

```
<template>
  <div class="p-10 h-screen bg-slate-200 overflow-auto">
    <h2 class="text-3xl"> Підтримка</h2>

    <main class="mt-10">
      <v-card max-width="800" text="">
        <form @submit.prevent="add" class="p-3">
          <v-text-field
            label="Тема"
            model-value=""
            variant="underlined"
          ></v-text-field>
          <v-textarea label="Повідомлення"></v-textarea>

          <v-btn type="submit" class="" color="primary">Надіслати</v-btn>
        </form>
      </v-card>
    </main>
  </div>
</template>
<script>
export default {
  data() {
    return {
      filterText: "",
      select: null,
      date: null,
      isDataPicker: false,
      inputDate: "",
    };
  },
  methods: {
    clear() {
      this.filterText = "";
      this.select = null;
      this.inputDate = "";
      console.dir(
        `${this.date.getDay()}.${this.date.getUTCMonth()}.${this.date.getFullYear()}`
      );
    },
    pik() {
      this.isDataPicker = false;
      this.inputDate = this.date.toJSON().slice(0, 10);
    },
    del(id, idx) {
      this.$store.dispatch("request/deleteL", id);
      console.log(this.request);
      this.logs.splice(idx, 1);
      console.log(this.request);

      console.log(idx, "_____________");
```

```
  },
 },
 async mounted() {
  await this.$store.dispatch("request/getL");
 },
 computed: {
  logs() {
   return this.$store.getters["request/getLogs"];
  },
 },
}
</script>
```

src/views/ViewProducts.vue

```
<template>
 <div class="p-10 bg-slate-200 overflow-auto">
  <h2 class="text-3xl">Товари</h2>

  <main class="mt-10">
   <div class="flex items-center justify-between mb-2">
    <div class="flex items-center space-x-4">
     <v-text-field
      label="Пошук"
      :model-value="filterText"
      v-model="filterText"
      class="w-[200px]"
      variant="solo-filled"
      hide-details

     >

     </v-text-field>

      <v-btn class="block h-full" @click="filterText = "" v-if="filterText!= "" color="warning"
        ><svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-
width="1.5" stroke="currentColor" class="w-6 h-6">
 <path stroke-linecap="round" stroke-linejoin="round" d="M6 18 18 6M6 6l12 12" />
</svg>
</v-btn>
    </div>
    <v-btn @click="dialog = true" color="primary"
     ><svg
      xmlns="http://www.w3.org/2000/svg"
      fill="none"
      viewBox="0 0 24 24"
      stroke-width="1.5"
      stroke="currentColor"
      class="w-8 h-8"
     >
      <path
       stroke-linecap="round"
       stroke-linejoin="round"
       d="M12 4.5v15m7.5-7.5h-15"
```

```
        />
       </svg>
      </v-btn>
     </div>
     <v-table>
      <thead>
       <tr>
        <th class="w-5text-left">№</th>
        <th class="w-2/4 text-left">Назва</th>
        <th class="text-left">Кількість, шт</th>
        <th class="text-left">Ціна продажа, грн</th>
        <th class="text-left">Ціна закупка, грн</th>
        <th class="text-left">Дія</th>
       </tr>
      </thead>
      <tbody>
       <tr v-for="(item, idx) in filtered(request, filterText)" :key="item.id">
        <td>{{ idx + 1 }}</td>
        <td>{{item.title}}</td>
        <td>{{ item.pcs}}</td>
        <td>{{ item.priceProdazh}}</td>
        <td>{{ item.priceZakup }}</td>
        <td class="space-x-2">
         <v-btn @click="deleteProduct(item.id, idx)" color="warning"
          ><span>
           Ред.
          </span>

         </v-btn>
         <v-btn @click="deleteProduct(item.id, idx)" color="red"
          ><svg
           xmlns="http://www.w3.org/2000/svg"
           fill="none"
           viewBox="0 0 24 24"
           stroke-width="1.5"
           stroke="currentColor"
           class="w-6 h-6"
          >
           <path
            stroke-linecap="round"
            stroke-linejoin="round"
            d="m14.74 9-.346 9m-4.788 0L9.26 9m9.968-3.21c.342.052.682.107 1.022.166m-1.022-.165L18.16 19.673a2.25 2.25 0 0 1-2.244 2.077H8.084a2.25 2.25 0 0 1-2.244-2.077L4.772 5.79m14.456 0a48.108 48.108 0 0 0-3.478-.397m-12 .562c.34-.059.68-.114 1.022-.165m0 0a48.11 48.11 0 0 1 3.478-.397m7.5 0v-.916c0-1.18-.91-2.164-2.09-2.201a51.964 51.964 0 0 0-3.32 0c-1.18.037-2.09 1.022-2.09 2.201v.916m7.5 0a48.667 48.667 0 0 0-7.5 0"
           />
          </svg>

         </v-btn>

        </td>
```

```
        </tr>
      </tbody>
    </v-table>
  </main>
  <v-dialog
    v-model="dialog"
    width="800"
  >
    <v-card
      max-width="800"
      text=""
      title="Редагувати товар"
    >
      <form @submit.prevent="add" class="p-3">
        <v-text-field
          label="Назва"
          model-value=""
          v-model="val.title"
          variant="underlined"
        ></v-text-field>
        <v-text-field
          label="Короткий опис"
          model-value=""
          v-model="val.descr"
          variant="underlined"
        ></v-text-field>
        <v-text-field
          label="Кількість"
          model-value="1"
          v-model="val.pcs"
          suffix="шт"
          variant="underlined"
        ></v-text-field>
        <v-text-field
          label="Ціна закупки"
          model-value=""
          v-model="val.priceZakup"
          suffix="₴"
          variant="underlined"
        ></v-text-field>
        <v-text-field
          label="Ціна продажі"
          model-value=""
          v-model="val.priceProdazh"
          suffix="₴"
          variant="underlined"
        ></v-text-field>
        <v-btn type="submit" class="" color="primary">Зберегти</v-btn>
      </form>
    </v-card>
  </v-dialog>
</div>
```

```
</template>

<script>
export default {
   data() {
   return {
     val: {
       title: "",
       pcs: "",
       priceZakup: "",
       date: new Date().toJSON().slice(0, 10),
       priceProdazh: "",
     },
     filterText: '',
     dialog: false
   };
 },
 computed: {
   request() {
     return this.$store.getters['request/getProd']
   }
 },
 methods: {
  async add() {
     await this.$store.dispatch('request/create', this.val);
     this.request.push(this.val)
   },
   deleteProduct(id, idx) {
     this.$store.dispatch('request/delete', id);
     console.log(this.request);
     this.request.splice(idx, 1);
     console.log(this.request);

     console.log(idx, "___________");
   },
   filtered(r, model) {
     return r.filter(item => {
       if (model) {
         return item.title.includes(model)
       }
       return item
     })
   }
 },
 async mounted() {
   await this.$store.dispatch('request/getP');
 }
}
</script>
```

```
<script>
export default {
  components: {},
  data() {
    return {
      val: {
        email: "",
        password: "",
      },
    };
  },
  methods: {
    async onSubmit() {
      await this.$store.dispatch("auth/register", this.val);
      this.$router.push("/dashboard");
    },
  },
};
</script>

<template>
  <div
    class="font-sans flex min-h-full flex-col justify-center px-6 py-12 lg:px-8"
  >
    <div class="sm:mx-auto sm:w-full sm:max-w-sm">
      <img
        class="mx-auto w-[80px]"
        src="../assets/logo-small.png"
        alt="Your Company"
      />
      <h2
        class="mt-10 text-center text-2xl font-bold leading-9 tracking-tight text-gray-900"
      >
        Реєстрація
      </h2>
    </div>

    <div class="mt-10 sm:mx-auto sm:w-full sm:max-w-sm">
      <form
        @submit.prevent="onSubmit"
        class="space-y-6"
        action=""
        method="POST"
      >


        <div>
          <label
            for="email"
            class="block text-sm font-medium leading-6 text-gray-900"
            >Email</label
          >
```

```
        <div class="mt-2">
          <input
            id="email"
            name="email"
            type="email"
            v-model="val.email"
            autocomplete="email"
            required
            class="block w-full rounded-md border-0 p-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-600 sm:text-sm sm:leading-6"
          />
        </div>
      </div>

      <div>
        <div class="flex items-center justify-between">
          <label
            for="password"
            class="block text-sm font-medium leading-6 text-gray-900"
            >Пароль</label
          >
        </div>
        <div class="mt-2">
          <input
            id="password"
            name="password"
            type="password"
            v-model="val.password"
            autocomplete="current-password"
            required
            class="block w-full rounded-md border-0 p-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-600 sm:text-sm sm:leading-6"
          />
        </div>
      </div>
      <div>
        <div class="flex items-center justify-between">
          <label
            for="password"
            class="block text-sm font-medium leading-6 text-gray-900"
            >Введіть пароль ще раз</label
          >
        </div>
        <div class="mt-2">
          <input
            id="password"
            name="password"
            type="password"
            v-model="val.password"
            autocomplete="current-password"
```

```
            required
            class="block w-full rounded-md border-0 p-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 placeholder:text-gray-400 focus:ring-2 focus:ring-inset focus:ring-indigo-600 sm:text-sm sm:leading-6"
          />
        </div>
      </div>


      <div>
        <button
          type="submit"
          class="flex w-full justify-center rounded-md bg-indigo-600 px-3 py-1.5 text-sm font-semibold leading-6 text-white shadow-sm hover:bg-indigo-500 focus-visible:outline focus-visible:outline-2 focus-visible:outline-offset-2 focus-visible:outline-indigo-600 transition"
        >
          Зареєструватись
        </button>
      </div>
    </form>

    <p class="mt-10 text-center text-sm text-gray-500">
      Вже зареєстровані?
      <router-link
        to="/auth"
        class="font-semibold leading-6 text-indigo-600 hover:text-indigo-500"
        >Увійти</router-link
      >
    </p>
  </div>
 </div>
</template>

<style scoped>
</style>
```

src/views/ViewReport.vue

```
<template>
 <div class="p-10 h-screen bg-slate-200 overflow-auto">
  <h2 class="text-3xl">Звіт</h2>

  <main class="mt-10">
   <div class="flex items-center justify-between mb-2">
    <div class="flex items-center space-x-4">
     <v-text-field
      label="Пошук"
      :model-value="filterText"
      v-model="filterText"
      class="w-[200px]"
      variant="solo-filled"
      hide-details
```

```
  >
  </v-text-field>
  <v-select
    label="Статус"
    :model-value="'123'"
    v-model="select"
    :items="['Продано', 'Видалено', 'Добавлено']"
    class="w-[200px]"
    variant="solo-filled"
    hide-details
  ></v-select>
  <Transition :duration="550">
    <div class="relative">
      <v-text-field
        label="Дата"
        class="w-[200px]"
        variant="solo-filled"
        v-model="inputDate"
        hide-details
        @click="isDataPicker = true"
      >
      </v-text-field>

      <v-date-picker
        v-if="isDataPicker"
        v-model="date"
        @click="pik"
        class="absolute top-0 left-0"
      ></v-date-picker>
    </div>
  </Transition>
  <v-btn
    class="block h-full"
    @click="clear"
    v-if="filterText != '' || select != null || inputDate != ''"
    color="warning"
    ><svg
      xmlns="http://www.w3.org/2000/svg"
      fill="none"
      viewBox="0 0 24 24"
      stroke-width="1.5"
      stroke="currentColor"
      class="w-6 h-6"
    >
      <path
        stroke-linecap="round"
        stroke-linejoin="round"
        d="M6 18 18 6M6 6l12 12"
      />
    </svg>
  </v-btn>
</div>
```

```
    </div>
    <v-table>
     <thead>
      <tr>
       <th class="w-5text-left">№</th>
       <th class="w-30 text-left">Дата</th>
       <th class="w-2/5 text-left">Назва</th>
       <th class="w-28 text-left">Статус</th>
       <th class="text-left">Кількість, шт</th>
       <th class="text-left">Ціна закупка, грн</th>

       <th class="text-left">Ціна продажа, грн</th>
       <th class="text-left">Прибуток, грн</th>
       <th class="text-left">Дія</th>
      </tr>
     </thead>
     <tbody>
      <tr v-for="(item, idx) in logs" :key="idx">
       <td>{{ idx + 1 }}</td>
       <td>{{ item.date }}</td>
       <td>{{ item.title }}</td>
       <td>
        <v-chip color="primary">Продано</v-chip>
       </td>
       <td>{{ item.pcs }}</td>
       <td>{{ item.priceZakup }}</td>
       <td>{{ item.priceProdazh }}</td>
       <td>{{ (item.priceProdazh-item.priceZakup)*item.pcs }}</td>
       <td class="space-x-2">
        <v-btn @click="del(item.id, idx)" color="red"
         ><svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          stroke-width="1.5"
          stroke="currentColor"
          class="w-6 h-6"
         >
          <path
           stroke-linecap="round"
           stroke-linejoin="round"
           d="m14.74 9-.346 9m-4.788 0L9.26 9m9.968-3.21c.342.052.682.107 1.022.166m-1.022-.165L18.16 19.673a2.25 2.25 0 0 1-2.244 2.077H8.084a2.25 2.25 0 0 1-2.244-2.077L4.772 5.79m14.456 0a48.108 48.108 0 0 0-3.478-.397m-12 .562c.34-.059.68-.114 1.022-.165m0 0a48.11 48.11 0 0 1 3.478-.397m7.5 0v-.916c0-1.18-.91-2.164-2.09-2.201a51.964 51.964 0 0 0-3.32 0c-1.18.037-2.09 1.022-2.09 2.201v.916m7.5 0a48.667 48.667 0 0 0-7.5 0"
          />
         </svg>
        </v-btn>
       </td>
      </tr>
     </tbody>
```

```
      </v-table>
    </main>
  </div>
</template>

<script>
export default {
  data() {
    return {
      filterText: "",
      select: null,
      date: null,
      isDataPicker: false,
      inputDate: "",
    };
  },
  methods: {
    clear() {
      this.filterText = "";
      this.select = null;
      this.inputDate = "";
      console.dir(
        `${this.date.getDay()}.${this.date.getUTCMonth()}.${this.date.getFullYear()}`
      );
    },
    pik() {
      this.isDataPicker = false;
      this.inputDate = this.date.toJSON().slice(0, 10);
    },
    del(id, idx) {
      this.$store.dispatch('request/deleteL', id);
      console.log(this.request);
      this.logs.splice(idx, 1);
      console.log(this.request);

      console.log(idx, "_____________");
    },
  },
  async mounted() {
    await this.$store.dispatch("request/getL");
  },
  computed: {
    logs() {
      return this.$store.getters["request/getLogs"];
    },
  },
};
</script>
```

src/store/index.js

```
import { createStore } from "vuex";
```

```
import auth from "./modules/auth.module"
import request from "./modules/request.module"

export default createStore({
  state: {},
  getters: {},
  mutations: {},
  actions: {},
  modules: {auth, request},
});
```

src/store/modules/auth.module.js

```
import axios from "axios"

export default {
    namespaced: true,
    state() {
        return {
            token: localStorage.getItem('jwt-token') || ''
        }
    },
    getters: {
        token(state) {
            return state.token
        },
        isAuth(state) {
            return !!state.token
        }
    },
    actions: {
        async login({ commit }, val) {
            const url = `https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=${import.meta.env.VITE_FB_KEY}`;
            const {data} = await axios.post(url, {...val, returnSecureToken: true });
            commit('setToken', data.idToken);
            console.log(data);

        },
        async register({ commit }, val) {
            const url = `https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=${import.meta.env.VITE_FB_KEY}`;
            const {data} = await axios.post(url, {...val, returnSecureToken: true });
            commit('setToken', data.idToken);
            console.log(data);

        }
    },
    mutations: {
```

```
        setToken(state, token) {
            state.token = token;
            localStorage.setItem('jwt-token', token);
        },

        logout( state ) {
            state.token = null;
            localStorage.removeItem('jwt-token');
        }

    }
}
```

src/store/modules/request.module.js

```
import axios from "axios"
import store from "../../store"

export default {
    namespaced: true,
    state() {
        return {
            requests: [],
            logs: []
        }
    },
    getters: {
        getProd(store) {
            return store.requests;
        },
        getLogs(store) {
            return store.logs;
        },
    },
    actions: {
        async create({ commit }, val) {
            try {
                const token = store.getters['auth/token']
                const {data} = await axios.post(`https://skladodrom-1-default-
rtdb.firebaseio.com/products.json?auth=${token}`, val);
            } catch (error) {
                alert(error.message);
            }
        },

        async delete({ commit },id) {
            try {
                const token = store.getters['auth/token']

                const {data} = await axios.delete(
                    `https://skladodrom-1-default-rtdb.firebaseio.com/products/${id}.json?auth=${token}`
                );
```

```
            console.log(data);
        } catch (error) {
            alert(error.message)
        }

    },

    async deleteL({ commit },id) {
        const token = store.getters['auth/token']
        const {data} = await axios.delete(`https://skladodrom-1-default-
rtdb.firebaseio.com/logs/${id}.json?auth=${token}`);
        console.log(data);
    },

    async getP({ commit }) {
        const token = store.getters['auth/token']
        const {data} = await axios.get(`https://skladodrom-1-default-
rtdb.firebaseio.com/products.json?auth=${token}`);
        const req = Object.keys(data).map(id => ({...data[id], id}));
        commit('setRequest', req)
        console.log(req);
    },

    async getL({ commit }) {
        const token = store.getters['auth/token']
        const {data} = await axios.get(`https://skladodrom-1-default-
rtdb.firebaseio.com/logs.json?auth=${token}`);
        console.log(data);
        const req = Object.keys(data).map(id => ({...data[id], id}));
        commit('setLogs', req)
        console.log(req);
    },
    async sellProduct({ commit }, val, newPcs) {
        try {
            const token = store.getters['auth/token']

            const {data} = await axios.post(
                `https://skladodrom-1-default-
rtdb.firebaseio.com/products/${val.id}.json?auth=${token}`,
                {...val, prc: newPcs}
                );

            const req = Object.keys(data).map(id => ({...data[id], id}));

            commit('setLogs', req)
        } catch (error) {
            alert(error.message)
        }
    },
  },
  mutations: {
```

```
        setRequest(state, requests) {
            state.requests = requests;
        },
        setLogs(state, logs) {
            state.logs = logs;
        },
    }
}
```

src/router/index.js

```
import { createRouter, createWebHistory } from "vue-router";
import ViewDashboard from "../views/ViewDashboard.vue";
import ViewAuth from "../views/ViewAuth.vue";
import ViewProducts from "../views/ViewProducts.vue";
import ViewReport from "../views/ViewReport.vue";
import ViewHelp from "../views/ViewHelp.vue";
import ViewReg from "../views/ViewReg.vue";
import store from "../store"

const routes = [
  {
    path: "/",
    name: "Dashboard",
    component: ViewDashboard,
    meta: {
      layout: 'Main',
      auth: true
    },
    alias: ['/dashboard']
  },
  {
    path: "/",
    name: "Products",
    component: ViewProducts,
    meta: {
      layout: 'Main',
      auth: true
    },
    alias: ['/products']
  },
  {
    path: "/",
    name: "Report",
    component: ViewReport,
    meta: {
      layout: 'Main',
      auth: true
    },
    alias: ['/report']
  },
  {
```

```
    path: "/",
    name: "Help",
    component: ViewHelp,
    meta: {
      layout: 'Main',
      auth: true
    },
    alias: ['/help']
  },
  {
    path: "/auth",
    name: "Auth",
    component: ViewAuth,
    meta: {
      layout: 'Auth'
    }
  },
  {
    path: "/reg",
    name: "Reg",
    component: ViewReg,
    meta: {
      layout: 'Auth'
    }
  }

];

const router = createRouter({
  history: createWebHistory(),
  routes,
});

router.beforeEach((to, from, next) => {
  if (to.meta.auth && store.getters['auth/isAuth']) {
    next()
  } else if (to.meta.auth && !store.getters['auth/isAuth']) {
    next('/auth');
  } else {
    next()
  }
})

export default router;
```

src/layouts/TheAuthLayouts.vue

```
<template>
    <div class="main">
        <router-view></router-view>
    </div>
```

```
</template>
```

src/layouts/TheMainLayouts.vue

```
<template>
   <div class="layout">
      <TheSidebar/>
      <div class="layout-main">
         <router-view></router-view>
      </div>
   </div>
</template>

<script>
import TheSidebar from '../components/TheSidebar.vue'

export default {
   components: {TheSidebar}
}

</script>
<style scoped>
  .layout {
      display: grid;
      grid-template-columns: 300px auto;
      max-height: 100vh;
  }
  .layout-main {
   height: 100vh;
   overflow: auto;
  }
</style>
```

src/components/DashboardWidget.vue

```
<template>
  <div class="flex flex-col bg-white rounded-xl sss">
    <div class="text-xl p-3 border-b">{{ title }}</div>
    <div
      class="p-6 flex items-center justify-center text-indigo-400 tracking-wide text-4xl h-full"
    >
      <div v-if="content">{{ content }}</div>
      <slot />
    </div>
  </div>
</template>

<script>
export default {
  props: {
    title: String,
    content: String,
```

```
  },
};
</script>
```

src/components/DashboardWidgetForm.vue

```
<template>
  <div class="flex flex-col bg-white rounded-xl sss">
    <div class="text-xl p-3 border-b">{{ title }}</div>
    <div
      class="p-2"
    >
      <div v-if="content">{{ content }}</div>
      <slot />
    </div>
  </div>
</template>

<script>
export default {
  props: {
    title: String,
    content: String,
  },
};
</script>
```

src/components/TheSidebar.vue

```
<template>
  <aside
    class="flex flex-col bg-blue-50 p-5 px-2 shadow-[5px_0_20px_rgba(0,0,0,.1)]"
  >
    <img src="../assets/logo.png" alt="" class="w-4/5 mb-10" />
    <ul class="space-y-2 grow mb-3">
      <li class="flex items-center">
        <router-link
        active-class="text-white bg-indigo-600"
          class="text-base p-2 rounded-xl hover:text-white hover:bg-indigo-600 transition duration-75 flex
w-full gap-2 items-center"
          to="/dashboard"
        >
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            stroke-width="1.5"
            stroke="currentColor"
            class="-mt-[2px] w-6 h-6"
          >
```

```
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            d="M3.75 6A2.25 2.25 0 0 1 6 3.75h2.25A2.25 2.25 0 0 1 10.5 6v2.25a2.25 2.25 0 0 1-2.25
2.25H6a2.25 2.25 0 0 1-2.25-2.25V6ZM3.75 15.75A2.25 2.25 0 0 1 6 13.5h2.25a2.25 2.25 0 0 1 2.25
2.25V18a2.25 2.25 0 0 1-2.25 2.25H6A2.25 2.25 0 0 1 3.75 18v-2.25ZM13.5 6a2.25 2.25 0 0 1 2.25-
2.25H18A2.25 2.25 0 0 1 20.25 6v2.25A2.25 2.25 0 0 1 18 10.5h-2.25a2.25 2.25 0 0 1-2.25-
2.25V6ZM13.5 15.75a2.25 2.25 0 0 1 2.25-2.25H18a2.25 2.25 0 0 1 2.25 2.25V18A2.25 2.25 0 0 1 18
20.25h-2.25A2.25 2.25 0 0 1 13.5 18v-2.25Z"
          />
        </svg>
        <span class="leading-6">Головна</span>
      </router-link>
    </li>
    <li class="flex items-center">
      <router-link
      active-class="text-white bg-indigo-600"
        class="text-base p-2 rounded-xl transition duration-75 hover:text-white hover:bg-indigo-600 flex
w-full gap-2 items-center"
        to="/products"
      >
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          stroke-width="1.5"
          stroke="currentColor"
          class="w-6 h-6"
        >
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            d="M3.75 12h16.5m-16.5 3.75h16.5M3.75 19.5h16.5M5.625 4.5h12.75a1.875 1.875 0 0 1 0
3.75H5.625a1.875 1.875 0 0 1 0-3.75Z"
          />
        </svg>

        <span class="leading-6">Товари</span>
      </router-link>
    </li>
    <li class="flex items-center">
      <router-link to="/report"
        class="text-base p-2 rounded-xl transition duration-75 hover:text-white hover:bg-indigo-600 flex
w-full gap-2 items-center"
        href="#"
      >
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          stroke-width="1.5"
          stroke="currentColor"
```

```
          class="w-6 h-6"
        >
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            d="M9 12h3.75M9 15h3.75M9 18h3.75m3 .75H18a2.25 2.25 0 0 0 2.25-2.25V6.108c0-1.135-.845-2.098-1.976-2.192a48.424 48.424 0 0 0-1.123-.08m-5.801 0c-.065.21-.1.433-.1.664 0 .414.336.75.75.75h4.5a.75.75 0 0 0 .75-.75 2.25 2.25 0 0 0-.1-.664m-5.8 0A2.251 2.251 0 0 1 13.5 2.25H15c1.012 0 1.867.668 2.15 1.586m-5.8 0c-.376.023-.75.05-1.124.08C9.095 4.01 8.25 4.973 8.25 6.108V8.25m0 0H4.875c-.621 0-1.125.504-1.125 1.125v11.25c0 .621.504 1.125 1.125 1.125h9.75c.621 0 1.125-.504 1.125-1.125V9.375c0-.621-.504-1.125-1.125-1.125H8.25ZM6.75 12h.008v.008H6.75V12Zm0 3h.008v.008H6.75V15Zm0 3h.008v.008H6.75V18Z"
          />
        </svg>
        <span class="leading-6">Звіт</span>
      </router-link>
    </li>
    <li class="flex items-center">
      <router-link to="/help"
        class="text-base p-2 rounded-xl transition duration-75 hover:text-white hover:bg-indigo-600 flex w-full gap-2 items-center"
        href="#"
      >
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          stroke-width="1.5"
          stroke="currentColor"
          class="w-6 h-6"
        >
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            d="M8.625 9.75a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0H8.25m4.125 0a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0H12m4.125 0a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0h-.375m-13.5 3.01c0 1.6 1.123 2.994 2.707 3.227 1.087.16 2.185.283 3.293.369V21l4.184-4.183a1.14 1.14 0 0 1 .778-.332 48.294 48.294 0 0 0 5.83-.498c1.585-.233 2.708-1.626 2.708-3.228V6.741c0-1.602-1.123-2.995-2.707-3.228A48.394 48.394 0 0 0 12 3c-2.392 0-4.744.175-7.043.513C3.373 3.746 2.25 5.14 2.25 6.741v6.018Z"
          />
        </svg>
        <span class="leading-6">Підтримка</span>
      </router-link>
    </li>
  </ul>
  <div class="flex items-center gap-x-5">
    <a
      class="flex grow leading-6 cursor-pointer text-base p-2 rounded-xl transition duration-75 hover:text-white hover:bg-indigo-400 w-full gap-2 items-center"
    ><img
      class="w-10 h-10 mr-1 rounded-full"
```

```
        src="http://placehold.it/100x100"
        alt=""
       />
       John Doe</a
      >
      <a @click="logout" href="#">
       <svg
       xmlns="http://www.w3.org/2000/svg"
       fill="none"
       viewBox="0 0 24 24"
       stroke-width="1.5"
       stroke="currentColor"
       class="opacity-60 hover:opacity-100 w-6 h-6"
      >
       <path
        stroke-linecap="round"
        stroke-linejoin="round"
        d="M15.75 9V5.25A2.25 2.25 0 0 0 13.5 3h-6a2.25 2.25 0 0 0-2.25 2.25v13.5A2.25 2.25 0 0 0
7.5 21h6a2.25 2.25 0 0 0 2.25-2.25V15M12 9l-3 3m0 0 3 3m-3-3h12.75"
       />
      </svg>
      </a>
    </div>
  </aside>
</template>

<script>
export default {
    methods: {
        logout() {
            this.$store.commit('auth/logout');
            this.$router.push('/auth');
        }
    }
};
</script>
```

ДОДАТОК Г

Ієрархічна структура сайту

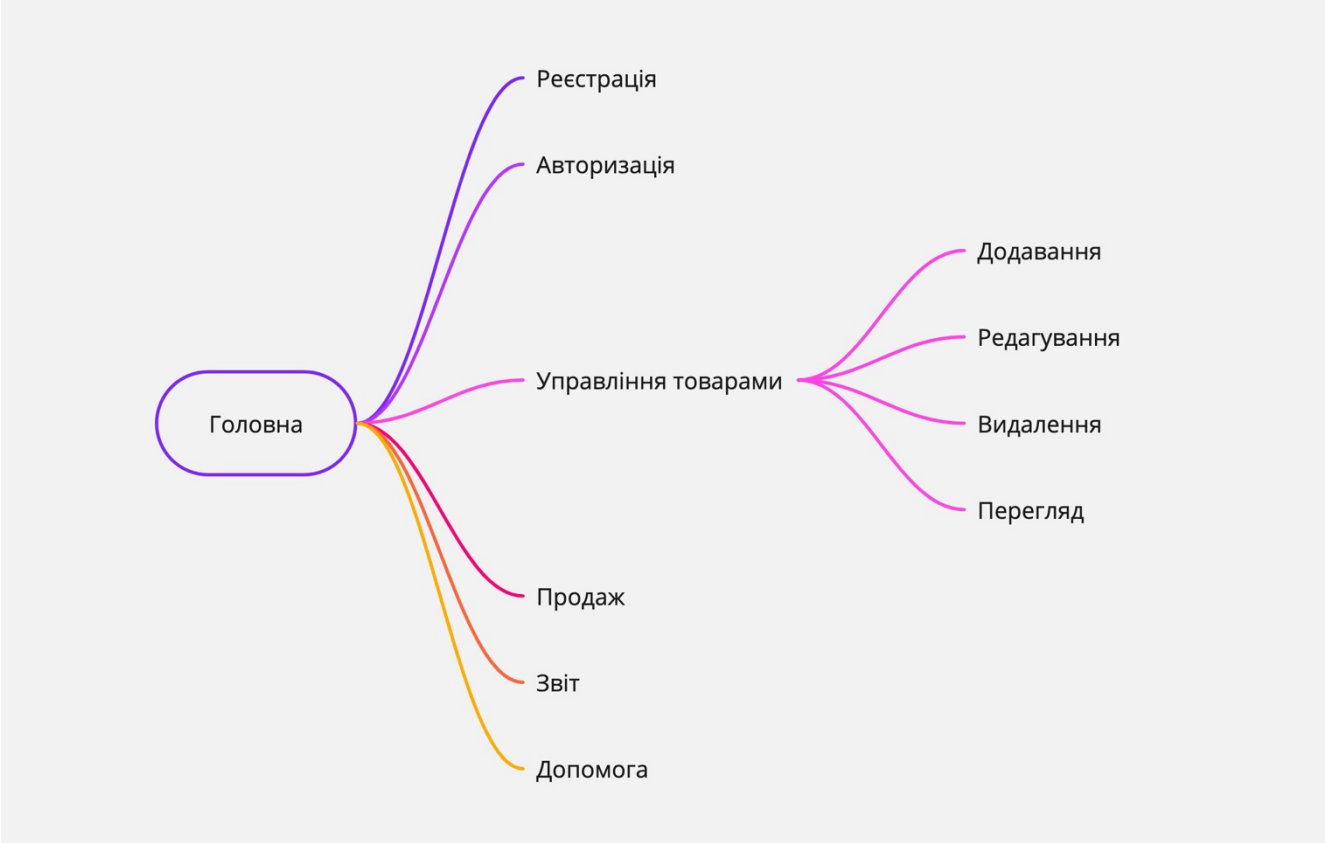


Рисунок Г.1 — Ієрархічна структура сайту

ДОДАТОК Д

Дизайн сторінок сайту

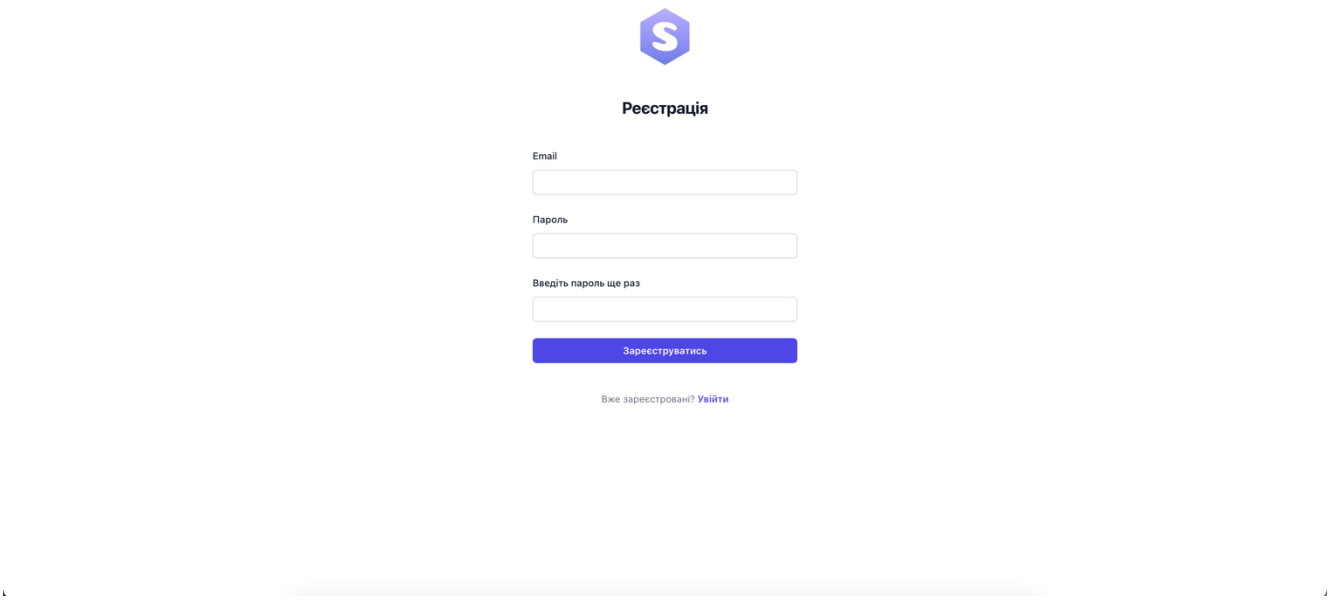


Рисунок Д.1 — Дизайн сторінки реєстрації

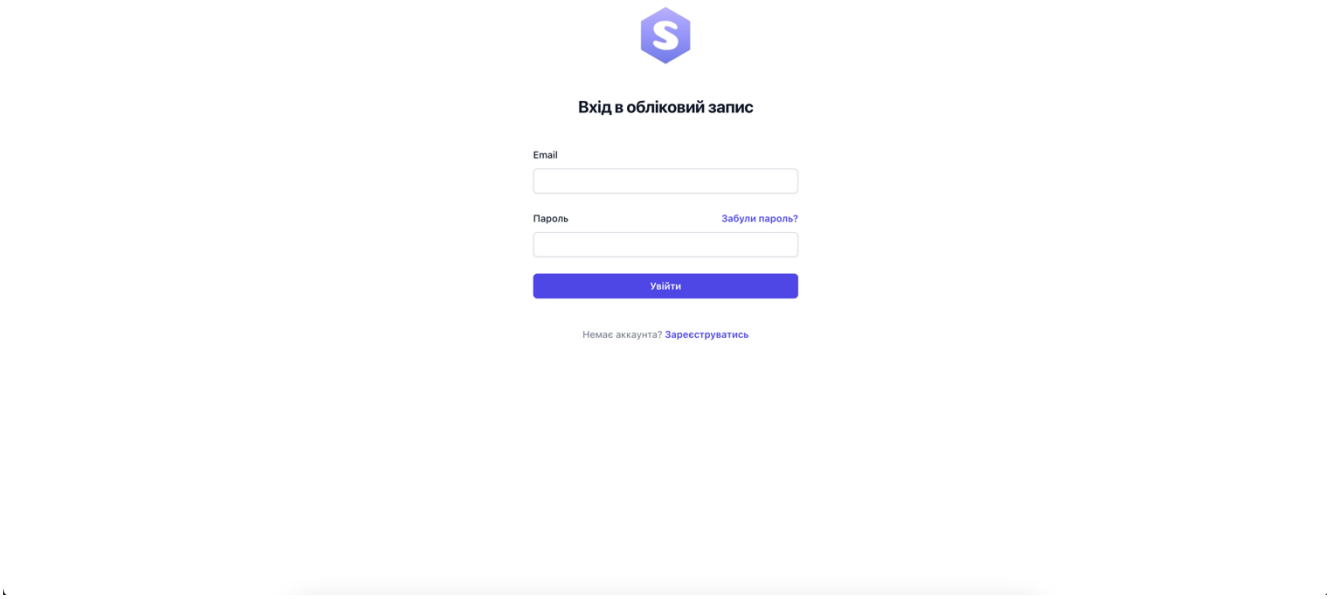


Рисунок Д.2 — Дизайн сторінки входу

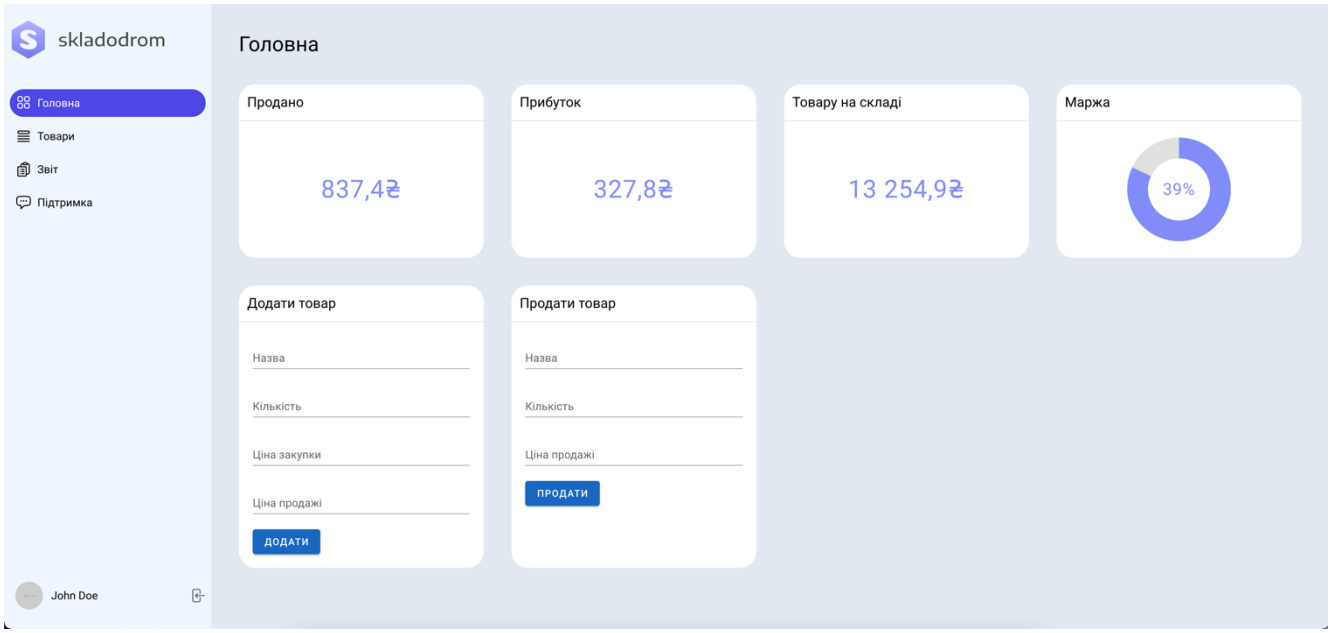


Рисунок Д.3 — Дизайн головної сторінки

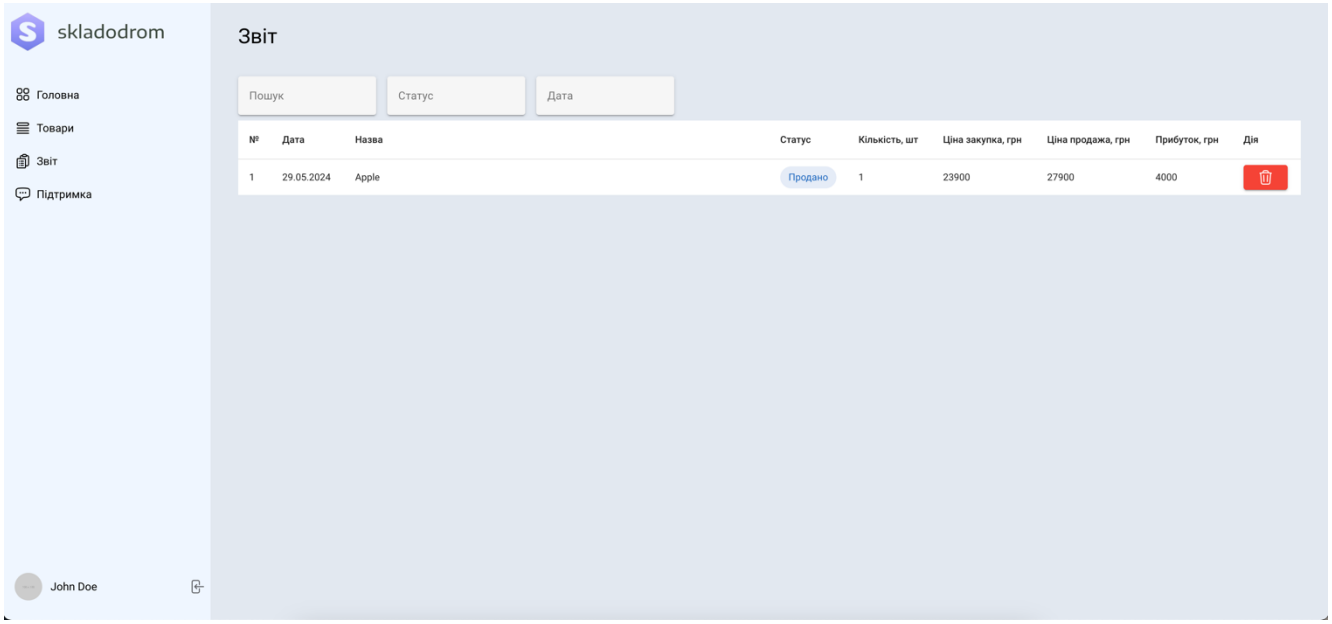


Рисунок Д.4 — Дизайн сторінки звіту

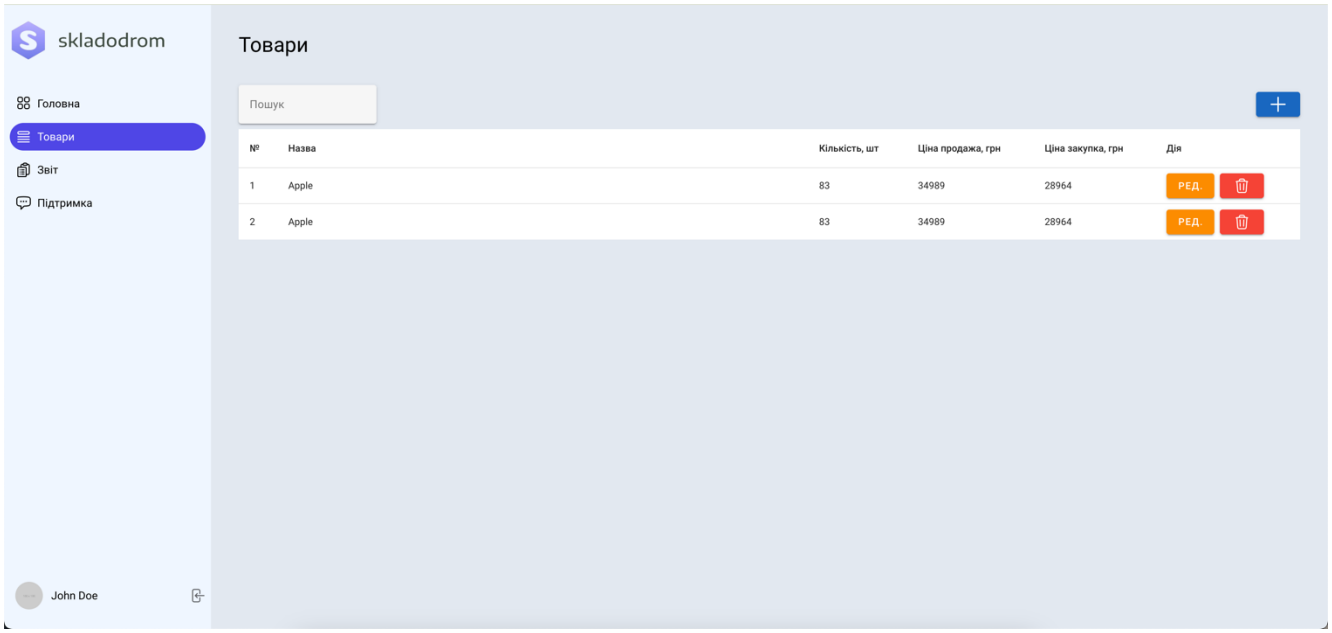


Рисунок Д.5 — Дизайн сторінки товарів

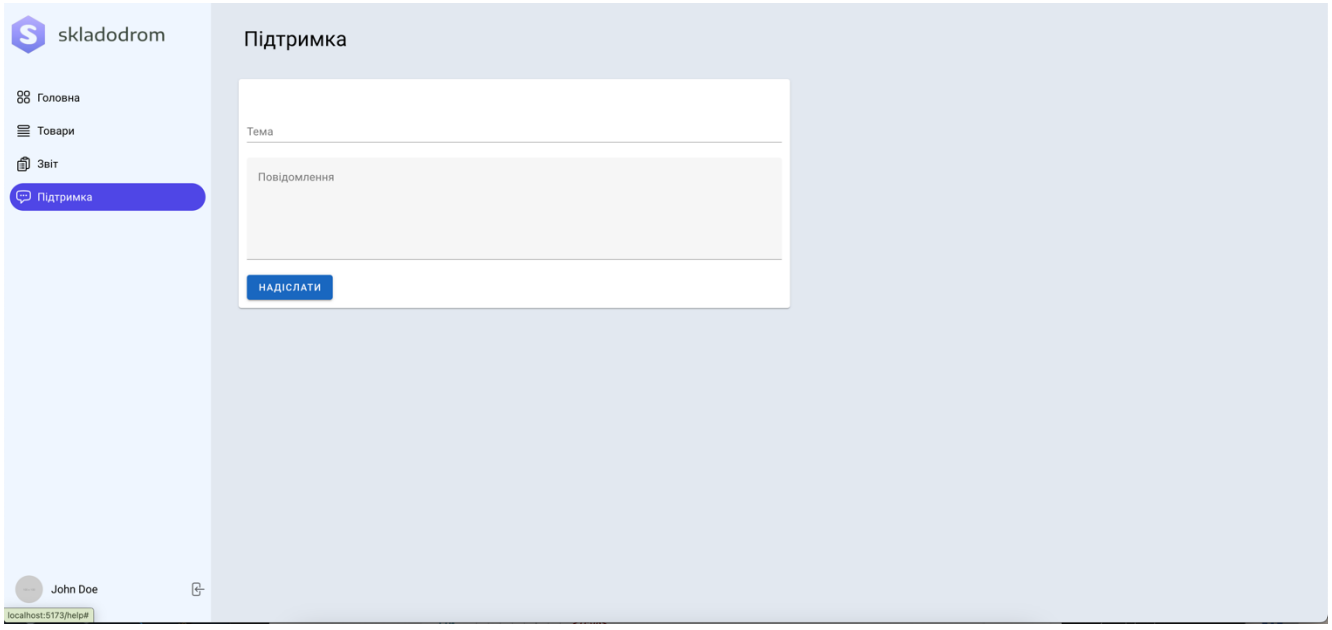


Рисунок Д.6 — Дизайн сторінки підтримки

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «WEB-платформа для управління та автоматизації з використанням API»

Тип роботи: бакалаврська дипломна робота

Підрозділ : кафедра обчислювальної техніки, ФІТКІ

Науковий керівник: Муращенко О.Г.

Показники звіту подібності

Оригінальність	98,2%
Схожість	1,8%

Аналіз звіту подібності (відмінити подібне)

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був ’згенерований Системою щодо роботи (додається)

Автор _____ Березовський О.С.
(підпис) *(прізвище, ініціали)*

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР.

Особа, відповідальна за перевірку _____ Муращенко О.Г.
(підпис) *(прізвище, ініціали)*

Експерт _____ _____
(за потреби) *(підпис)* *(прізвище, ініціали)*