МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет)_Автоматизації і комп'ютерних систем_____
Кафедра _Інформаційних технологій, штучного інтелекту і кібербезпеки_

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ ___Андрій Форсюк___	_____ _Сергій Грибков__
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«__» ____ _____ 2023р.	«__» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності__122 «Комп'ютерні науки»_____
(код та назва спеціальності)
освітньо-професійної програми____Комп'ютерні науки_____
_____
На тему:__Розроблення WEB системи для магазину акваріумістики_____

Виконав: здобувач__4_ курсу, групи__ЗКН-5-2_____

Зданевич Артем Сергійович_____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник _____М'якшило Олена Михайлівна_____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2023р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) __Автоматизації і комп'ютерних систем_____
Кафедра __Інформаційних технологій, штучного інтелекту і кібербезпеки_
Освітній ступінь___бакалавр_____
Спеціальність___122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма ____Комп'ютерні науки_____
(назва)

ЗАТВЕРДЖУЮ

Завідувач

кафедри Інформаційних технологій,

штучного інтелекту і кібербезпеки_

_____Грибков С.В.

"_16_" ____листопада____2022року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Зданевича Артема Сергійовича_____
(прізвище, ім'я, по батькові)

1. Тема роботи: _*« Розроблення WEB системи для магазину акваріумістики»*____
_____

Керівник роботи___*М'якшило Олена Михайлівна, доцент, кандидат технічних наук*_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "16" листопада 2022 року № 815- кс

2. Строк подання здобувачем роботи__*16 січня 2023 року*_____

3. Вихідні дані до роботи *дані про організаційну структуру підприємства ПП «АкваСвіт», інформація прозамовників/працівників/пристрої/постачальників, дані про процес формування замовлень ПП «АкваСвіт», організаційна структура підприємства, схема функціонування відділів.*_____

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)__*системний аналіз діяльності підприємства, моделі функціонування*

підприємства, модель AS-IS та модель TO-BE, розробка інтерфейсу користувача, налаштування функціоналу системи, інструкція користувача, охорона праці та навколишнього середовища.

5. Перелік графічного матеріалу

Контекстна діаграма функціональної моделі, діаграма деталізації, Діаграма декомпозиції блоків, діаграма моделі TO-BE, логічна модель БД, фізична модель БД, схема бази даних у MS SQL Server, скріншоти інтерфейсу системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	М'якшило О.М.		
2	М'якшило О.М.		
3	М'якшило О.М.		
4	М'якшило О.М.		

7. Дата видачі завдання 16 листопада 2022 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз об`єкту автоматизації та постановка задачі на проектування	16.11.22 – 20.12.22	Виконано
2	Реалізація функцій інформаційної системи. Побудова інтерфейсу користувача.	21.12.22 – 30.12.22	Виконано
3	Дослідження питання охорони праці на підприємстві	1.01.23 – 05.01.23	Виконано
4	Оформлення пояснювальної записки та створення презентації	05.01.23 – 28.01.23	Виконано

Здобувач _____ Зданевич А.С.

(підпис) (прізвище та ініціали)

Керівник роботи _____ М’якшило О.М.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота містить у собі 26 зображень, 88 сторінок.

Дана бакалаврська робота на тему «Розроблення WEB системи для магазину акваріумістики» має в собі мету опис процесу створення повноцінної WEB системи для адміністрування онлайн магазином акваріумістики. Розглянені усі аспекти розробки, починаючи від вибору системи розташування майбутньої системи, закінчуючи наданням можливості в майбутньому на зміну певних елементів або системи у цілому.

Ключові слова: інформаційна система, діаграми, дослідження, інтернет-магазин, функціональна модель

ANNOTATION

Qualification work contains 26 images, 88 pages

This bachelor's thesis on the topic "Development of WEB system for aquarium store" aims to describe the process of creating a full-fledged WEB system for the administration of online aquarium store. All aspects of development are considered, starting from the choice of the location system of the future system, ending with the possibility in the future to change certain elements or the system as a whole.

Keywords: information system, diagrams, research, online store, functional model

ЗМІСТ

ВСТУП

У сучасному світі обличчям кожного бізнесу – є його сайт. Задля приваблення клієнта на сайті можуть використовуватись різні елементи, такі як анімація, інтерактивні елементи, яскраві кольори, тощо.

Одним з головних вимог до будь-якого сайту має бути підтримка великої кількості клієнтів на сайті одночасно. Сучасні технології та системи дозволять оптимізувати внутрішню частину сайту різними способами, щоб досягти більше ніж 1 000 000 користувачів одночасно.

Протягом часу змінювались технології, комп'ютери, сервери, але одні із головних вимог зосталися незмінні:

1. Можливість одночасно обробляти більше ніж 1 000 000 запитів
2. Уся система має завантажуватися швидко, аби не змушувати користувача чекати
3. Система мусить бути переносною. Дозволяти на зміну обладнання без додаткового створення компонентів задля конфігурації
4. Система мусить бути захищеною, аби не дозволяти на зовнішнє втручання

Головною метою створення даної бакалаврської роботи є створення системи для магазину акваріумістики, який буде відповідати умовам сучасних веб-систем.

Бакалаврська робота поділена на кілька частин, головні з яких, це:

Створення бекендової частини. Бекенд відповідає за функціонування системи в цілому, що дозволяє на введення додаткових можливостей, або додаткового функціоналу. Головні вимоги до бекенду це можливість швидкої додання функціональності, без критичного рефакторінгу коду, можливість на добру продуктивність для великої кількості користувачів одночасно.

Створення сховища даних. Сховище даних мусить відповідати вимогам безпеки, продуктивності та мінімального рефакторінгу при змінах. Для створення сховища даних обрана система – це MS SQL Server 2022.

Створення фронтенду. Фронтенд – обличчя веб-системи. На фронтенд покладається кілька досить важких умов: швидке завантаження на стороні клієнту

(належить взяти під увагу те, що клієнт може не мати комп’ютеру з доброю продуктивністю), бути легкозмінним, мати можливість швидкої зміни бекенду (навіть без перезавантаження цілої фронтенд-системи), .

Контейнеризація. Задля можливості легкого переносу поміж пристроями належить взяти під увагу контейнеризацію за допомогою інструменту Docker.

Тестова частина. У даній частині будуть описані методи тестування, та результати тестованого матеріалу. Результати повинні відповідати поставленим технічним вимогим для розробленої системи.

Висновками даної бакалаврської роботи є створена та протестована система для продажу акваріумістичного обладнання.

Розділ 1. СИСТЕМНИЙ АНАЛІЗ ОБ`ЄКТУ АВТОМАТИЗАЦІЇ ТА ПОСТАНОВКА ЗАДАЧІ НА ПРОЕКТУВАННЯ

1.1. Характеристика об’єкту автоматизації

Об'єктом автоматизації є надання послуг продажу та доставки акваріумістичного обладнання, зокрема:

- розміщення оголошень на програмній платформі власного інтернет-магазину
- перевірка статусів замовлення і за необхідності їх зміна
- відстеження стану доставки товару

Для здійснення заданих цілей, є необхідним створення власної інформаційної системи, відповідальної за це.

1.1.1. Опис та особливості функціонування компанії

Компанія "АкваСвіт" займається продажем акваріумістичного обладнання на території Європи. У компанії немає свого власного виробництва, вона працює за допомогою закупівель у постачальників товару і продажем за вигідною ціною. Структура компанії базується на наступних відділах (Рис.1.1.):

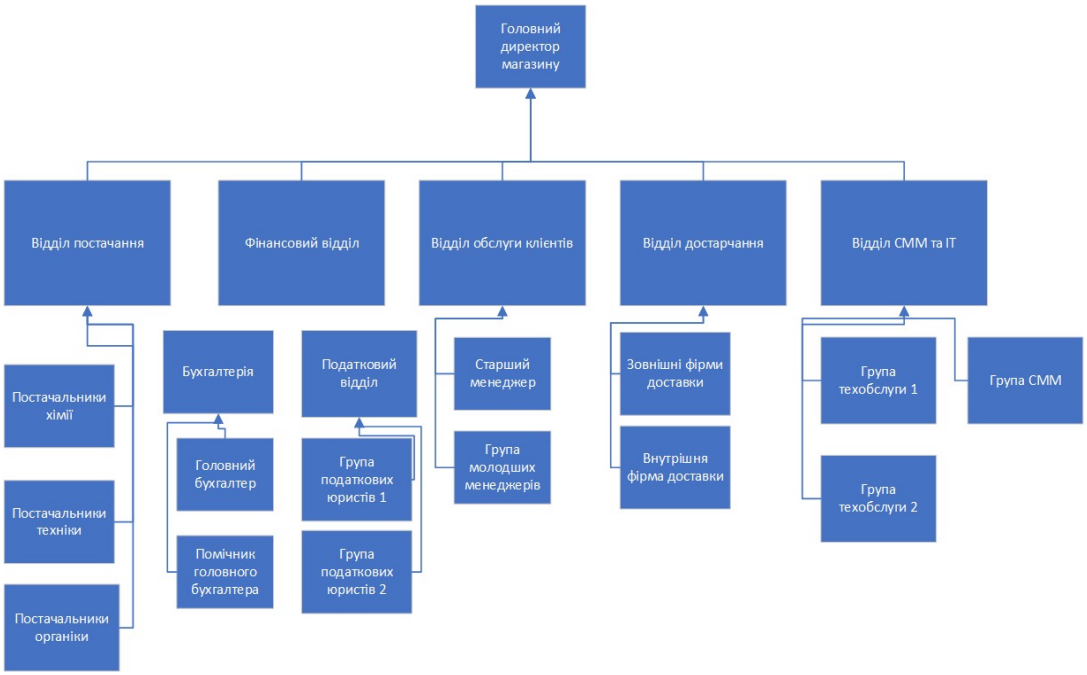


Рис. 1.1. Структура компанії "АкваСвіт"

Кожен відділ має своє завдання:

Відділ постачання	Відділ постачання відповідає за забезпечення підприємства необхідними ресурсами, наприклад, сировиною, обладнанням, послугами та іншими матеріалами, необхідними для продукції або послуг. Він виконує функції пошуку, оцінки та вибору постачальників, переговорів з ними щодо умов постачання, здійснення закупівель та контролю якості та своєчасності постачання.
Фінансовий відділ	Фінансовий відділ відповідає за управління фінансами компанії. Він здійснює планування, аналіз, контролю та репортування фінансової діяльності компанії.
Відділ обслуги клієнтів	Відділ обслуговування клієнтів відповідає за забезпечення високого рівня обслуговування клієнтів і вирішення проблем, які виникають під час роботи з ними.
Відділ достарчання	Відділ доставки відповідає за організацію та координацію доставки товарів або послуг клієнтам. Він здійснює планування та координацію перевезення, забезпечує транспортування товарів в належному стані та в зазначені строки.
Відділ СММ та ІТ	Відділ СММ (Системного Менеджменту Маркетингу) та ІТ (Інформаційних Технологій) відповідає за розробку та впровадження маркетингових стратегій та інформаційних технологій в компанії.

1.1.2. Діаграма діяльності інтернет-магазину «АкваСвіт»

Діаграма діяльності реалізації замовлення демонструє робочий процес створення замовлення і його реалізацію від початку і до кінця. У ній зазначаються постачальники, підрозділи та служби, які беруть участь у процесі створення замовлення.

Ця діаграма також показує, як формується замовлення для відвантаження, і включає перелік документів, необхідних для виконання замовлення.

Діаграма діяльності дає змогу побачити всі процеси, що протікають в організації, з погляду замовника і логічно пов'язати їх між собою.

Вона також допомагає визначити, які процеси і в якій послідовності мають бути реалізовані для забезпечення виконання замовлення.

Діаграма реалізації створеного замовлення представлена на Рис.1.2. Ця модель являє собою діаграму діяльності в стандарті UML.

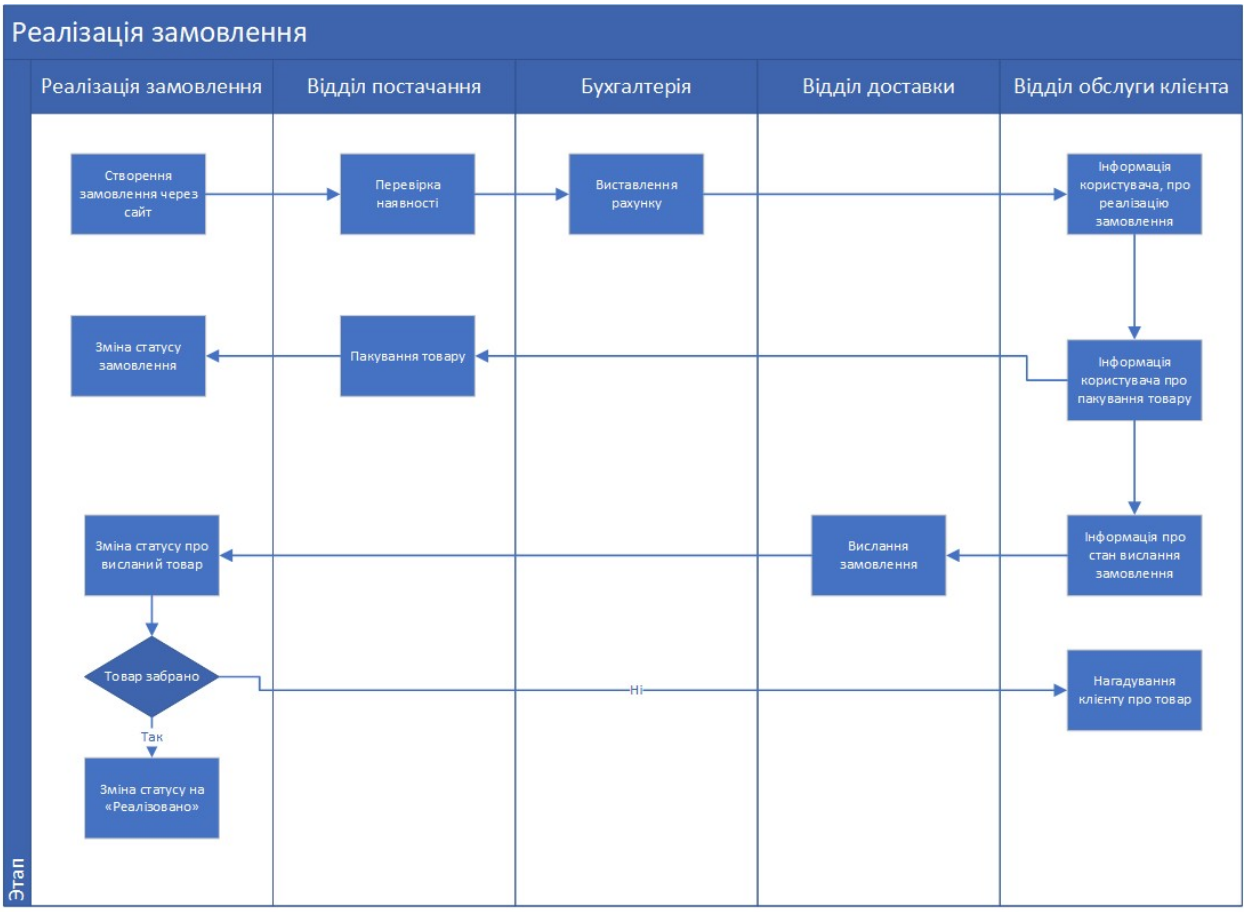


Рис.1.2. Функціональна модель реалізації замовлення

Реалізація замовлення починається з моменту його створення на сайті. Сайт переходить у базу даних, а клієнт отримує повідомлення, що його замовлення прийнято.

На тлі цих процесів, бухгалтерія готує виставлення рахунку, а склад перевіряє точну доступність товару.

Після перевірки та виставлення рахунку, клієнт отримує повідомлення про те, що його замовлення пакується на складі. База даних змінює стан замовлення на "Пакується"

Товар відправляється для замовника через зовнішню або внутрішню службу доставки, і відправляє клієнту повідомлення, що товар був відправлений. Після

цього відбувається перевірка, яка визначає, чи забрав клієнт товар. У разі, якщо клієнт не забрав товар, він продовжує отримувати повідомлення про товар, який надійшов, до моменту повернення товару в магазин або його прийняття клієнтом.

1.1.3. Дослідження та аналіз поточного стану процесів, що підлягають автоматизації

Однією з головних проблем усередині фірми є система комунікацій і відсутність автоматизацій процесів, які можуть бути автоматизовані.

Як завдання автоматизації , варто виділити передавання повідомлень і здійснення комунікацій між відділами. Прикладом такої комунікації може бути безпосередня зв'язок між відділом постачання та відділом доставки, шляхом передачі та організації офіційної частини на віртуальну розподільну систему архівації даних.

Існуючі проблеми комунікацій можуть негативно позначитися на термінах доставки клієнту бажаного товару, а також завдати шкоди фінансовій частині через несвоєчасну сплату податків через низьку координацію роботи відділів усередині фірми.

1.2. Функціональне моделювання

1.2.1. Про програмний засіб CA ERwin Process Modeler

CA ERwin Process Modeler (раніше BPwin) - це відзначений нагородами інструмент моделювання даних, який використовується для пошуку, візуалізації, проектування, розгортання та стандартизації високоякісних корпоративних даних. Знаходьте та документуйте будь-які дані з будь-якого місця для забезпечення узгодженості, ясності та повторного використання артефактів в рамках великомасштабної інтеграції даних, управління основними даними, метаданими, великими даними, бізнес-аналітикою та аналітичними ініціативами - і все це одночасно з підтримкою зусиль з управління даними та аналітикою.[8]

1.2.2. Модель «Аналіз діяльності магазину акваріумістики «АкваСвіт» на рівні А0»

Для аналізу функціонування майбутнього інтернет-магазину було створено модель TO-BE бізнес-процесів під час реалізації замовлення.

Контекстна діаграма моделі в стандарті IDEF0 представлена на рис.1.3. Дана діаграма відповідає нульовому рівню початку процесу, являючись його точкою створення.

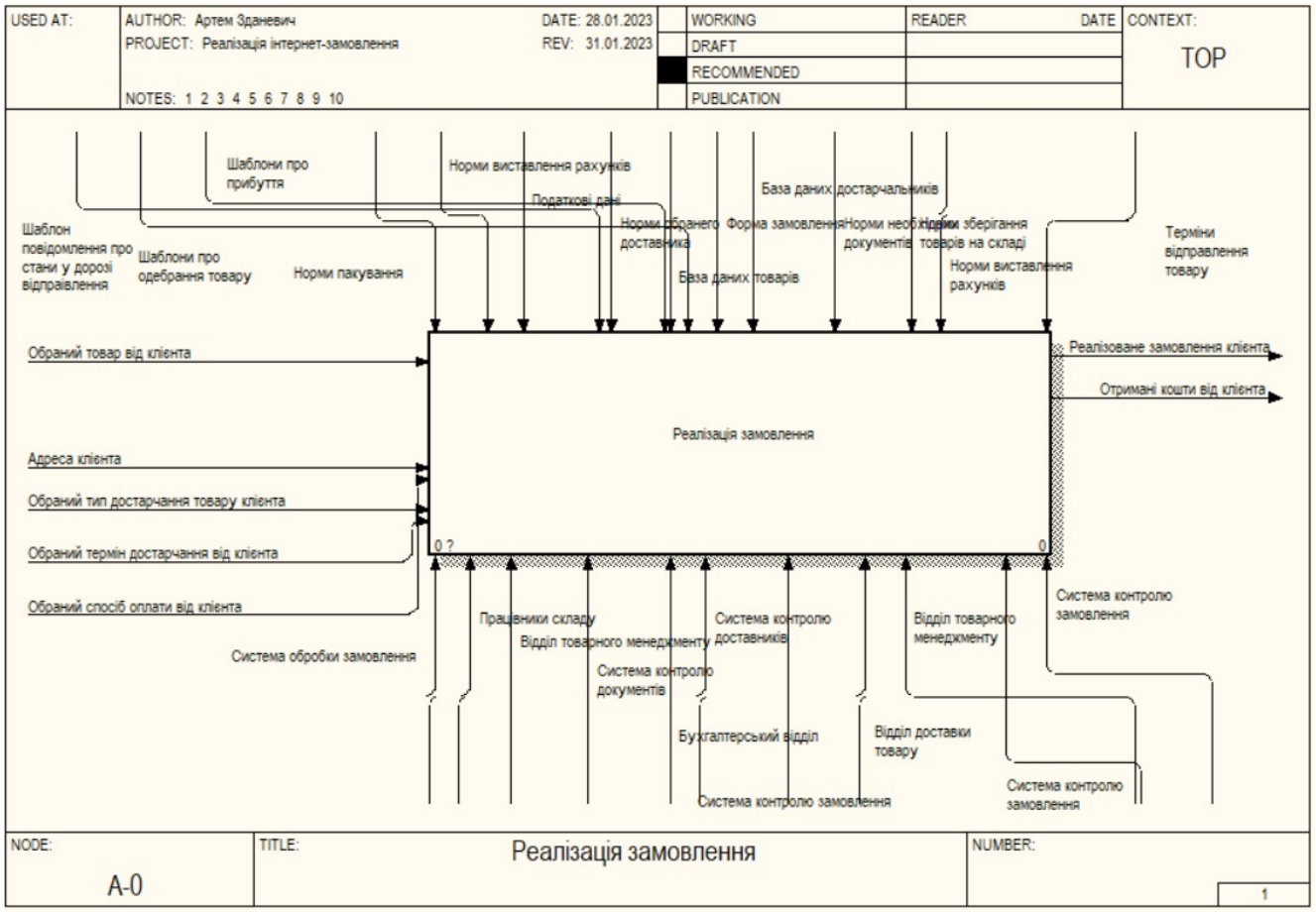


Рис.1.3. Функціональна модель реалізації замовлення магазину акваріумістики "АкваСвіт"

1.2.3. Діаграма декомпозиції 1-го рівня бізнес-процесу « Реалізація замовлення»

Діаграма декомпозиції першого рівня бізнес-процесу « Реалізація замовлення» представлена на рис.1.4:

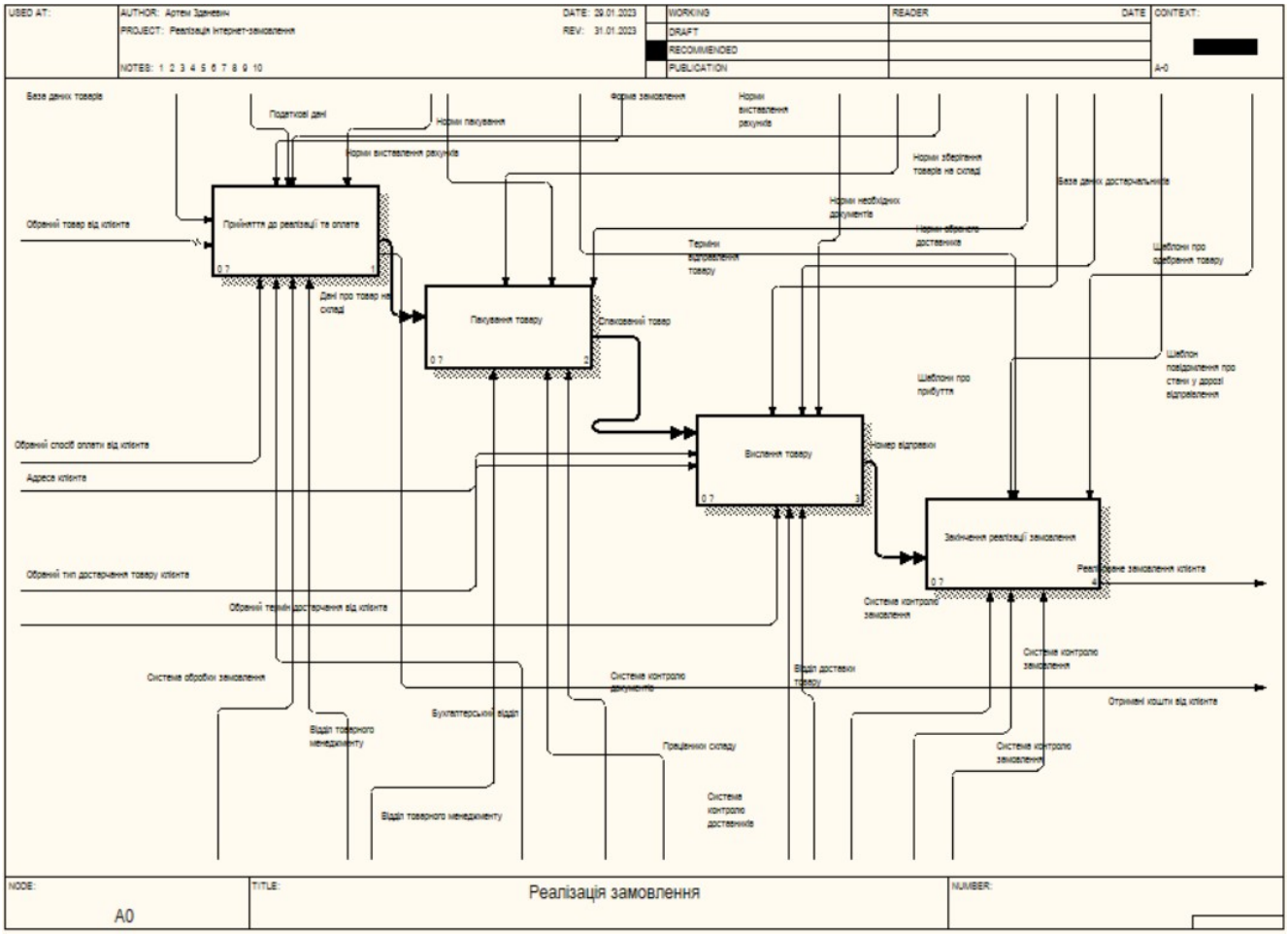


Рис.1.4. Декомпозиція першого рівня «Реалізація замовлення»

Процес реалізації замовлення розділений на 7 частин:

Перша частина це прийняття замовлення і його оплати. У цьому підпроцесі користувач робить замовлення на сайті, залишаючи свої дані. Система ідентифікації замовлень приймає замовлення, у разі якщо користувач вніс передоплату через сторонній сервіс, то додається також інформація про оплату.

Друга частина процесу це пакування товару. З першої частини, в другу передається розташування товару на складі, його стан і кількість. Пакується товар згідно із заздалегідь заданими нормами. На додаток до спакованого товару додається також фінансовий чек про підтвердження замовлення.

Третя частина - частина відправлення товару. З метою підвищення якості доставки товару, перед відправленням товару, згідно з даними покупця, проводять відбір доставщиків, які за якістю ціни і швидкості доставки можуть запропонувати свої послуги. З обраним доставщиком складаються документи на доставку.

Остання частина відповідає за моніторинг доставки замовлення, в разі якщо замовлення не доставлено, відбувається моніторинг на якому етапі процес виявився невдалим.

П'ята частина відповідає за постійне повідомлення користувача про статус його замовлення. Кожен із попередніх підпроцесів відправляє повідомлення в систему моніторингу повідомлень про зміну стану. Система ж своєю чергою надсилає користувачеві повідомлення, що статус його замовлення було змінено.

Шоста частина підпроцесів оглядає продукти на складі та знаходить ті, які найчастіше купували, або ті, у яких підходить термін придатності, і виставляє їх у базу даних із позначкою "на розпродаж".

Сьома частина виставляє товари з бази даних на розпродаж і додатково замовляє нові комплекти товарів у постачальників.

1.2.4. Діаграма декомпозиції 2-го рівня. «Прийняття до реалізації та оплата»

Діаграма декомпозиції другого рівня бізнес-процесу « Реалізація замовлення» представлена у додатку Г 2-го рівня:

Під час обробки товару, згідно зі схемою TO-BE, усі операції, пов'язані зі знаходженням інформації про товар і підготовкою офіційної інформації та контактом із третіми сторонами, відбуваються завдяки системам контролю та комунікацій. Завдяки такому підходу, швидкість комунікації збільшилася в кілька разів. На початковому етапі відбувається перевірка доступності товару, і якщо товар доступний, то надаються його координати на складі для формування замовлення.

Коли дані щодо місцезнаходження товару на складі отримано, відбувається наступний етап складання товару - складання товару.

1.2.5. Діаграма декомпозиції 3-го рівня «Пакування товару»

Функціональну схему пакування товару можна побачити у Додатку Г 3-го рівня:

Одним із головних плюсів інформаційної системи контролю пакування є те, що всі необхідні документи до замовлення дублюються клієнту на пошту, що дає змогу, навіть у разі втрати чека, здійснити повернення завдяки онлайн-чеку.

Після того як товар зібрано в єдине замовлення, його передають у фінансовий відділ на виставлення рахунку і чека на покупку.

1.2.6. Діаграма декомпозиції 4-го рівня «Відправлення товару»

Діаграма декомпозиції 4-го рівня «Відправлення товару» представлена у Додатку Г 4-го рівня

Інформаційна система, завдяки алгоритму аналізу та систематизації доставщиків за критеріями, дає змогу, на основі користувацьких даних, визначити, який доставщик найшвидше впорається зі своїм завданням - доставкою товару. Для покупця

це означає мінімальну кількість часу для реалізації замовлення, а для власника магазину - швидку реалізацію замовлення.

Контакт і договори з доставщиками, виконує інформаційна система за допомогою email листів, або за допомогою відділу комунікацій.

1.2.7. Діаграма декомпозиції 5-го рівня «Закінчення реалізації замовлення»

Діаграма декомпозиції 5-го рівня «Закінчення реалізації замовлення» представлена у додатку Г 5-го рівня.

Протягом усієї доставки інформаційна система моніторить і повідомляє про будь-які зміни стану товару в дорозі відділу доставки. Додатково, після прибуття система надсилає користувачеві сповіщення на електронну пошту та СМС, про те, що товар прибув на вказане користувачем місце.

1.3. Переваги і недоліки онлайн торгівлі порівняно зі стаціонарною

Для надання послуг продажу товару використовують два типи рішень:

- Онлайн рішення. Дозволяє швидко і надійно обслужити величезну частину клієнтів з різних міст, а в деяких випадках навіть і країн
- Стаціонарне рішення, що дає змогу обслужити виключно локальну містність, без можливості взаємодії з іншими постачальниками послуг, приклад: послуги доставки

Стандартне онлайн-рішення не позбавлене певних якостей, серед яких:

Переваги:

- Зручність: Інтернет-магазини дозволяють клієнтам здійснювати покупки, не виходячи з дому, в будь-який час доби.
- Різноманітність: Інтернет-магазини часто мають ширший асортимент товарів, ніж фізичні магазини, оскільки вони не мають таких же обмежень у просторі.
- Ціна: Інтернет-магазини часто можуть пропонувати нижчі ціни, ніж фізичні магазини, оскільки вони мають нижчі накладні витрати.
- Відгуки та рейтинги: Інтернет-магазини, як правило, мають відгуки та рейтинги клієнтів, які можуть допомогти покупцям приймати обґрунтовані рішення.

Недоліки:

- Вартість доставки: Вартість доставки може бути високою, особливо для великих товарів.
- Відсутність тактильного досвіду: Клієнти не мають змоги доторкнутися до товару або побачити його на власні очі перед покупкою.
- Ризик шахрайства: Існує ризик шахрайства при здійсненні покупок онлайн, тому покупці повинні бути обережними при введенні особистої та фінансової інформації.
- Повернення та обмін: Повернення та обмін товарів можуть бути більш складними і трудомісткими при здійсненні покупок в Інтернеті.

1.3.1. Огляд і порівняльний аналіз існуючих інтернет магазинів

Для порівняння інтернет магазинів було прийнято такі критерії:

- Вибір продуктів: широта асортименту, доступність різних видів товарів
- Ціни: рівень цін, відсутність знижок та акцій
- Доставка: час доставки, вартість доставки
- Якість обслуговування: швидкість реакції на запити клієнтів, доступність консультанта
- Доступність інформації: доступність інформації про товари, відгуки клієнтів

Магазин	**Вибір продуктів**	**Ціни**	**Доставка**	**Якість обслуговування**	**Доступність інформації**
Розетка	Великий	Високі	Довгий час	Повільна реакція	Є базова інформація
Рибки.com.ua	Малий	Середньо-низькі	Короткий час	Швидка реакція	Мало
Kakadu	Середній	Занадто високі	Швидко, проте досить дорого	Дуже повільна реакція	Достатньо

Результати порівняння інтернет магазинів за критеріями показують, що "Розетка" має великий вибір продуктів за високими цінами, але доволі довгий час доставки та поверхневий рівень якості обслуговування. "Ribky.com.ua" має невеликий асортимент із середньо-низькими цінами та швидким часом доставки, але недостатньо інформації. "Kakadu" має середній вибір продуктів за занадто високими цінами, швидкий, але дорогий час доставки і дуже повільний рівень якості обслуговування.

Виходячи з цього, магаизин АкваМСвіт матиме такі переваги: великий вибір продуктів з низькими цінами, швидкий час доставки і високий рівень якості обслуговування, а також достатньо інформації для покупців.

З огляду на те, що наявні рішення пропонують досить мізерний вибір акваріумістичного обладнання, вони починають додатково розширювати свої послуги на інші товари. Прикладом такого рішення може бути магазин Розетка. Він мігрував з рішення стандартного онлайн-магазину техніки в товарний майданчик, де кожен може реалізувати свій товар будь-якої категорії, займаючись перепродажем або виставленням своєї фірми.[5]

Додатково будучи одним із головних обмежень на територіальну доставку, наявна проблема може завдати дискомфорту іноземцям, зі створенням нездоланного бар'єру виходу на європейський, або інший ринок. Розглядаючи наявні рішення, існує великий простір для створення платформи, яка сприяла б взаємодії з поширеними проблемами інтернет-магазинів для продажу акваріумістичного обладнання. Для вирішення даних проблем необхідно передбачити наступні помилки:

- Простір продавця або адміністратора. Більшість платформ мають проблему з інтегрованими системами контролю товару, фінансів, поставок і доставок. Спроба охопити все за один раз часто невдала і веде до дискомфорту (який часто переростає у відторгнення до платформи) під час взаємодії з інтернет-системою.
- Низька інформованість покупця про стан його замовлення. Покупець бажає знати, на якій стадії перебуває його товар і чи є прогрес, з моменту початку замовлення. Існуючі лідери в інтернет-продажу, такі як Moyo, Citrus, Rozetka мають близьку до ідеалу систему інформування покупця про стан його замовлення. Однак, мають жахливу складову комунікації в разі проблем з доставкою або товаром, що необхідно врахувати при розробці власної системи обслуговування клієнта.

- Неспішність при введенні нових технологій. Найчастіше інтернет-гіганти ігнорують можливість додаткової інтеграції інноваційної системи (до прикладу оплати Apple Pay, Google Pay), тим самим позбавляючи клієнта простого способу оплати, а себе прискореної можливості отримати оплату за товар.[6]

1.3.2. Оцінка ефективності інтернет торгівлі

Для оцінки економічної ефективності інтернет-магазину можна використовувати різні фінансові метрики, як-от виторг, собівартість товарів (COGS), валовий прибуток, витрати на маркетинг, операційні витрати та чистий прибуток.

Для магазину "AquaSvit" розрахунки можуть виглядати наступним чином:

Revenue = кількість проданих товарів х ціна товару

COGS = кількість проданих товарів х собівартість товару

Gross Profit = Revenue - COGS

Marketing expenses = витрати на рекламу і маркетинг

Operating expenses = витрати на операційну діяльність (наприклад, зарплата співробітників, оренда приміщення)

Net Profit = Gross Profit - Marketing expenses - Operating expenses

Щоб розрахувати економічну ефективність інтернет-магазину "AquaWorld", можна скористатися такою формулою:

Прибуток = Виручка - Операційні витрати

Виручка: 300 позицій * 40 грн = 12,000 грн на добу

Операційні витрати: Щоб визначити операційні витрати, потрібно знати такі витрати, як заробітна плата співробітникам, орендна плата, товарно-матеріальні цінності, доставка, маркетинг та інші витрати. Припустимо, що операційні витрати становлять 6 000 грн на день.

Прибуток = 12 000 грн - 6 000 грн = 6 000 грн на добу

Цей розрахунок показує, що магазин "AquaWorld" приносить прибуток у розмірі 6 000 грн на день. Однак це приблизна оцінка, і для точної оцінки

економічної ефективності магазину необхідний більш комплексний аналіз операційних витрат та інших факторів.

Для подальшої оцінки економічної ефективності магазину "AquaWorld" важливо порівняти його норму прибутку з галузевими стандартами, а також врахувати такі фактори, як ринкова конкуренція, задоволеність покупців і потенціал зростання в майбутньому. Показник рентабельності магазину можна розрахувати наступним чином:

Profit Margin = (Прибуток / Виручка) * 100%

Маржа прибутку = (6 000 грн / 12 000 грн) * 100% = 50%

Показник рентабельності 50% свідчить про те, що на кожні 100 грн отриманої виручки магазин заробляє 50 грн прибутку. Ця інформація може бути корисною в порівнянні з галузевими стандартами для визначення відносної економічної ефективності магазину. Крім того, важливо враховувати операційні витрати магазину, задоволеність клієнтів і потенціал зростання, щоб зробити повну оцінку його економічної ефективності.

Якщо припустити, що стаціонарний магазин продає 300 товарів на день за 4000 грн, то, якщо магазин почне продавати товари онлайн і зможе обробляти більшу кількість замовлень зі скороченим у чотири рази терміном обробки замовлень і доставки за допомогою служб доставки в різні міста, то корисно було б порахувати очікуваний приріст виручки та підвищення ефективності. Це можна зробити за допомогою прогнозування обсягу продажів, аналізу операційних витрат і витрат на доставку, а також розрахунку чистого прибутку. Крім того, моніторинг і аналіз зворотного зв'язку з клієнтами, ринкових тенденцій і конкуренції мають вирішальне значення для доопрацювання стратегії онлайн-продажів, щоб оптимізувати дохід і прибутковість.

Під час розрахунку економічної ефективності інтернет-торгівлі магазин "АкваСвіт" має врахувати збільшення оброблюваних замовлень у 10 разів (до 3000 на день), зменшення терміну реалізації замовлення в 4 рази і можливість доставки товарів у різні міста.

Оцінка економічної ефективності може включати такі розрахунки:

- Збільшення доходу за рахунок збільшення кількості замовлень і збільшення середньої ціни замовлення.
- Зниження витрат на роботу персоналу, зменшення витрат на транспорт і зменшення витрат на оренду приміщення.

Необхідно врахувати витрати на створення та підтримку інтернет-магазину, а також рекламні кампанії для залучення клієнтів. Потрібно провести розрахунки собівартості замовлення і відсотка виручки від кожного замовлення. Оцінити можливі ризики, такі як повернення товару або незадоволені клієнти. Тільки після цього можна зробити висновки про економічну ефективність інтернет-торгівлі для магазину "АкваСвіт".[5]

Період окупності становить ((6000 * 30) / ((3000 * 1000) - (6000 * 30))) = 2 місяці при збільшенні замовлень і скороченні часу доставки. Доставка в різні міста може вплинути на період окупності, але без додаткової інформації, такої як типи доставників, або вплив курсів валют неможливо визначити точний ефект.

1.4. Постановка задачі на проектування

Виходячи з вищесказаного, можна стверджувати, що існує величезний простір з досить міцними аспектами наявних недоліків інтернет-магазинів, на основі яких створена система може стати популярною.

Основний акцент необхідно зробити на:

Модульність. Система має бути взаємозамінною, а значить використовувати мікросервісну архітектуру. Використання цього рішення дасть змогу швидко встановлювати нові зручні компоненти, як-от система контролю фінансів, система контролю доставок, система особистого кабінету користувача або адміністратора.

Хороший комунікаційний відділ. Створення власного відділу може бути гарним економічним плюсом за малої кількості клієнтів, однак при розширенні необхідно задуматися про інтеграцію зовнішньої служби обслуговування клієнтів.

Система повідомлень. Система повідомлень є досить двоякою системою. З одного боку, вона може слугувати плюсом під час інформування клієнта про його замовлення або сезонні знижки на продукти, що його цікавлять. Однак, вона може

стати для клієнта надто настирливою, що створить погане враження про магазин і змусить клієнта знайти йому альтернативу.

Розділ 2. ТЕХНІЧНЕ ЗАВДАННЯ

Загальні положення.

Найменування системи*:* «Інтернет-сервіс продажу акваріумістичного обладнання «АкваСвіт» »

Результати робіт зі створення системи оформлюються згідно з вимогами ДСТУ на відповідні етапи розробки. Порядок оформлення і передачі результатів у даному випадку визначається змістом і календарним планом виконання розробки.

У випадку необхідності на наступних стадіях робіт по створенню системи окремі положення можуть уточнюватися і розвиватися.

Призначення і цілі створення системи.

Інтернет-магазин акваріумістики - це платформа електронної комерції, яка дозволяє клієнтам переглядати та купувати товари, пов'язані з акваріумістикою, такі як риби, водні рослини, акваріуми та обладнання. Метою створення такої системи є надання клієнтам легкого та зручного способу придбання цих товарів без необхідності фізичного відвідування магазину. Система призначена для демонстрації широкого асортименту товарів, надання детальної інформації про кожен товар, а також для того, щоб клієнти могли легко здійснювати покупки та відстежувати свої замовлення. Крім того, система може включати такі функції, як відгуки клієнтів, порівняння товарів та персоналізовані рекомендації для покращення купівельного досвіду клієнтів.

Цілі створення системи.

1. Зручність: Система дозволяє клієнтам купувати товари, пов'язані з акваріумістикою, не виходячи з дому, в будь-який час.
2. Різноманітність товарів: Система пропонує широкий асортимент товарів, що полегшує покупцям пошук потрібних їм товарів.
3. Інформація про товар: Система надає детальну інформацію про кожен товар, таку як технічні характеристики, особливості та відгуки клієнтів, щоб допомогти покупцям прийняти обґрунтоване рішення про покупку.

4. Легка покупка: Система дозволяє клієнтам легко здійснювати покупки та відстежувати свої замовлення, що може покращити загальний досвід покупок.
5. Збільшення продажів: Маючи інтернет-магазин, власник може охопити ширшу аудиторію і потенційно збільшити продажі.
6. Економічна ефективність: Інтернет-магазин може бути економічно вигідним способом для власника продавати свою продукцію, оскільки він усуває необхідність у фізичних вітринах і знижує накладні витрати.
7. Персоналізований досвід: Система може включати в себе такі функції, як рекомендації продуктів на основі історії перегляду та історії покупок, щоб покращити досвід покупок.
8. Задля створення веб-системи, яка відповідає усім технічним вимогим, необхідно їх окреслити на початку.

Характеристика об'єкта автоматизації.

3.1. Короткі відомості про об'єкт автоматизації.

Об'єктом автоматизації є діяльність акваріумістичного магазину «АкваСвіт».

Вимоги до системи

Для загальних вимог приймаємо те, що кількість замовлень щоденно може доходити до 500 000. Для тестування даної кількості замовлень необхідно протестувати як всі компоненти окремо, так і разом. Загалом, тест матиме в собі понад 2 000 000 випадків.

Приймаємо вимогу також на швидкість завантаження сторінки. На початку приймаємо максимальний час очікування 10 секунд. Якщо інтерфейс користувача завантажується довше – тест провалено.

Приймаємо вимогу на завантаження інтерфейсу адміністратора. Прийнятий час – 30 секунд. З уваги на те, що інтерфейс адміністратора має в собі в рази більше елементів, можемо прийняти той факт, що на їх отримання, конвертацію та відображення потрібно більше часу.

Приймаємо також вимогу до бази даних те, що вона має знаходити, виконувати процедури та робити процедури CRUD (Create, Read, Update, Delete) за константний час. В якості константного часу приймаємо 1 хвилину.

Тестування може відбуватися як на окремих компонентах, так і на цілій системі.

Основною вимогою є також її швидке розгортання на сервері. Швидкий зв'язок між компонентами. Приймаємо також те, що система мусить бути відмовостійкою на каскадні помилки з кожної сторони.

Рекомендовані системні вимоги представлено у таблиці 2.1

Таблиця 2.1.Конфігурація пристрою для підтримки інформаційної системи

№ п/п	**Основні характеристики комп'ютера**
Технічне забезпечення для сервера	
1	HP ML115 Intel Xeon Quad Core 2,5 GHz\8 Gb\1 TB RAID5\ LAN 1 Gbit
Технічне забезпечення для клієнта	
1	Athlon QL-65 Dual Core 2,1 GHz; RAM: 2048 Mb; HDD: 250 Gb;
2	Монітор 15"
3	Миша USB
4	Клавіатура USB

Склад і зміст робіт по створенню системи.

Стадії створення системи і терміни виконання робіт наведені в таблиці 2.2.

Таблиця 2.2.

№ п/п	**Найменування робіт**	**Строки виконання робіт**
1	Передпроектне дослідження об'єкта автоматизації	01.01.2023
2	Технічне завдання	15.01.2023
3	Технічний проект	20.01.2023
4	Оформлення документації	28.01.2023

Порядок контролю і приймання системи.

Система вводиться на діючому підприємстві «АкваСвіт». При введенні в дію система повинна пройти приймальні випробування згідно з ДСТУ 3974-2000.

Випробування для визначення працездатності і рішення про можливість приймання системи в дослідну експлуатацію проводять розробники разом із замовником. Програму випробувань складає розробник і затверджує замовник.

Здача в дослідну експлуатацію здійснюється на основі технічного завдання та інструкції користувача. За результатами дослідної експлуатації формується перелік доробок і рекомендовані строки їх виконання.

Введення в дію системи оформлюється актом здачі-прийому.

Вимоги до складу і змісту робіт із підготовки до введення системи в дію.
Для введення в дію замовник виконує ряд робіт із підготовки об’єкта:

- проводить укомплектування технічних засобів;
- організовує навчання користувачів системи роботі на ПК і вивчення інструкції з її експлуатації;
- проводить дослідну експлуатацію і вводить систему в дію.

Вимоги до документації.

На систему розробляється комплекс документації у складі: технічне завдання та технічний проект.

Документація на систему розробляється у відповідності з вимогами Державних стандартів серії 19 «Єдина система програмної документації» та серії 24 «Єдина система стандартів автоматизованих систем управління».

Джерела розробки.

При розробленні технічного завдання на систему використано наступні документи:

- ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;

- ДСТУ 3973–2000 Система розроблення та поставлення продукції на виробництво;
- ДСТУ Б В.2.5–82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом.

Розділ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ

3.1. Розробка бекенду

3.1.1. Розробка інформаційного забезпечення

Вибір СКБД

СКБД має відповідати таким вимогам:

- Надійність: база даних має бути надійною і захищеною від відмов і корупції даних. Резервне копіювання і відновлення мають бути налаштовані для захисту від втрати даних.
- Продуктивність: база даних має бути достатньо потужною та оптимізованою для обробки великого обсягу запитів і транзакцій. Індексація та оптимізація запитів може бути необхідною для підвищення продуктивності.
- Безпека: дані мають бути захищені від несанкціонованого доступу та зміни. Рольова модель безпеки та шифрування даних можуть бути використані для забезпечення безпеки.
- Сумісність: база даних має бути сумісною з іншими системами та додатками, що використовуються в організації.
- Масштабованість: база даних має бути готовою до зростання і зміни навантаження, як зростання кількості даних, так і зростання кількості користувачів. Це може включати в себе використання розподілених систем і кластеризацію для підтримки великого навантаження.
- Адміністрування та моніторинг: база даних повинна мати інструменти для адміністрування та моніторингу, щоб забезпечити безперебійне функціонування та швидке реагування на будь-які проблеми.

SQL Server є реляційною СУБД, яка пропонує безліч можливостей та інструментів для управління й аналізу даних. Вона підтримує різні типи даних, включно з текстом, цілим числом, датами і навіть геоданими. Також вона має розширені можливості для роботи з даними, такі як тригери, збережені процедури та подання.

SQL Server пропонує надійність і високу продуктивність, завдяки технологіям, таким як резервне копіювання та відновлення, індексація та розподілені транзакції. Вона також підтримує масштабування, що дає змогу легко розширювати капасіті та продуктивність залежно від навантаження.

SQL Server пропонує широкий набір інструментів для розробників, включно з SQL Server Management Studio (SSMS) і SQL Server Data Tools (SSDT) для розробки та адміністрування. Вона також має широку підтримку різних мов програмування, таких як C#, Java, Python і PHP, що дає змогу розробникам використовувати їхню улюблену мову для роботи з базою даних.

Загалом, SQL Server є потужною і надійною СКБД, яка може відповідати потребам багатьох різних типів додатків і організацій. Його безліч функцій та інструментів робить його привабливим вибором для багатьох розробників і адміністраторів баз даних.

3.1.1.1. Схеми бази даних

Створення декількох схем у SQL Server може надати кілька переваг:

- Організація даних: Схеми дають змогу організувати об'єкти бази даних, як-от таблиці, подання та збережені процедури в логічні групи, що допомагає спростити навігацію та управління даними.
- Контроль доступу: Схеми можуть використовуватися для контролю доступу до даних, де кожна схема може мати свої власні права доступу та рольова модель безпеки.
- Ізоляція даних: Схеми можуть використовуватися для ізоляції даних між різними додатками або департаментами в організації, щоб уникнути конфліктів і організувати дані в різних середовищах.
- Поділ навантаження: Схеми можуть використовуватися для поділу навантаження між різними серверами або інстансами SQL Server, що може допомогти підвищити продуктивність і забезпечити більш гнучку архітектуру даних.
- Легкість оновлення та міграції: Схеми можна використовувати для оновлення та міграції даних окремо, що допомагає спростити та

прискорити процес оновлення та міграції даних без впливу на інші схеми та об'єкти бази даних.

- Поділ роботи різних команд: Схеми можуть використовуватися для поділу роботи різних команд і розробників над базою даних, що допомагає уникнути конфліктів і забезпечує більш ефективне використання ресурсів і часу.
- Поділ різних бізнес-логік: Схеми можна використовувати для поділу різних бізнес-логік, що допомагає спростити моделювання даних і забезпечує ефективніше використання ресурсів.
- Поділ різних типів даних: Схеми можна використовувати для поділу різних типів даних, наприклад, статичні та динамічні дані, що допомагає спростити моделювання даних і забезпечує більш ефективне використання ресурсів.[4]

Оскільки створена система використовує мікросервісну архітектуру, то необхідно забезпечити кілька схем, які підтримуватимуть стабільну роботу кожного окремого компонента, навіть у разі збою однієї або декількох схем. На основі цього створено такі схеми:

1. aquafront
2. aquaback
3. personal_data

Скрипт для генерації бази даних, а також повнорозмірні моделі бази даних подано в додатку А, Б, В.

3.1.1.2. Aquafront

Схема aquafront (Представлена на Рис.3.1.) зберігає в собі необхідні дані для коректної роботи фронтенда. Основні дані кількома мовами такі:

- Повідомлення для користувача про різні події, що відбуваються на сторінці. Тексти для ярилків на сторінці
- Дані для підключення до мікросервісної архітектури. Порти, посилання, зашифровані паролі та логіни
- Дані для стилю і подій. Кольори, шрифти, посилання на зображення

Для з'єднання між таблицями в схемі використовують референційні ключі. Таблиця Languages має з'єднання з таблицею Messages за колоною LANG_ID. Використання зовнішнього ключа дає змогу застосунку одночасно підтримувати кілька мов, без необхідності змінювати внутрішні системні файли. Достатньо лише додати/змінити/видалити записи в базі даних.

Таблиця Connection_Properties відповість за встановлення з'єднання з внутрішньою мікросервісною архітектурою. Кожен запис у таблиці відповідає (за зовнішнім ключем) таблиці Connection_Types, що дає змогу використовувати однакові імена властивостей для різних з'єднань.[4]

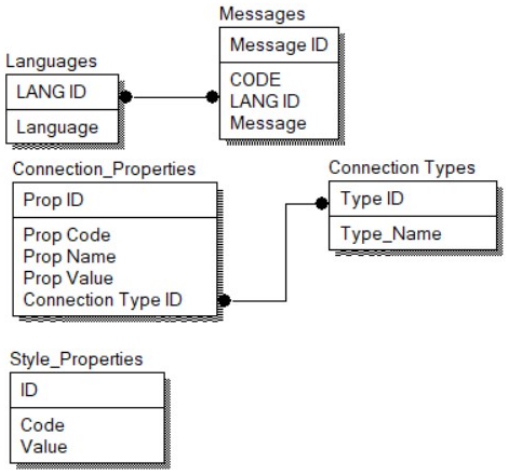


Рис.3.1. Схема таблиць даних у схемі aquafront

Для з'єднання з базою даних, у сервісі фронтенда будуть встановлені константні значення в зашифрованому вигляді, що дає змогу розраховувати на безпеку під час входу і запобігти витоку даних. Сервіс фронтенда працює ізольовано. Він не має доступу до решти схем інших компонентів і не має прямого доступу до даних.

3.1.1.3. Aquaback

Схема aquaback (Представлена на Рис.3.2.) має в собі лише одну головну таблицю, яка відповідає за збереження параметрів роботи сервісу. Кожен параметр має свій код, відповідно до якого додаток зможе його швидко знайти в таблиці.

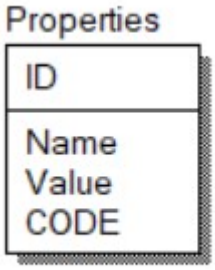


Рис.3. 2. Схема таблиць даних у схемі aquaback

3.1.1.4. Personal_data

Схема personal_data (Представлена на Рис.3.3.) є найголовнішою схемою бази даних. У ній міститься вся інформація про замовлення, покупки, товари, постачальників і фінанси.

Доступ до цієї схеми має відбуватися виключно через процедури, що запобігає виконанню на базі SQL INJECT. У разі, якщо мікросервісна архітектура застосунку виявиться зламаною, база даних має бути ізольована внутрішньою системою доступів, що запобігають прямому зверненню з невідомих і неавторизованих джерел.

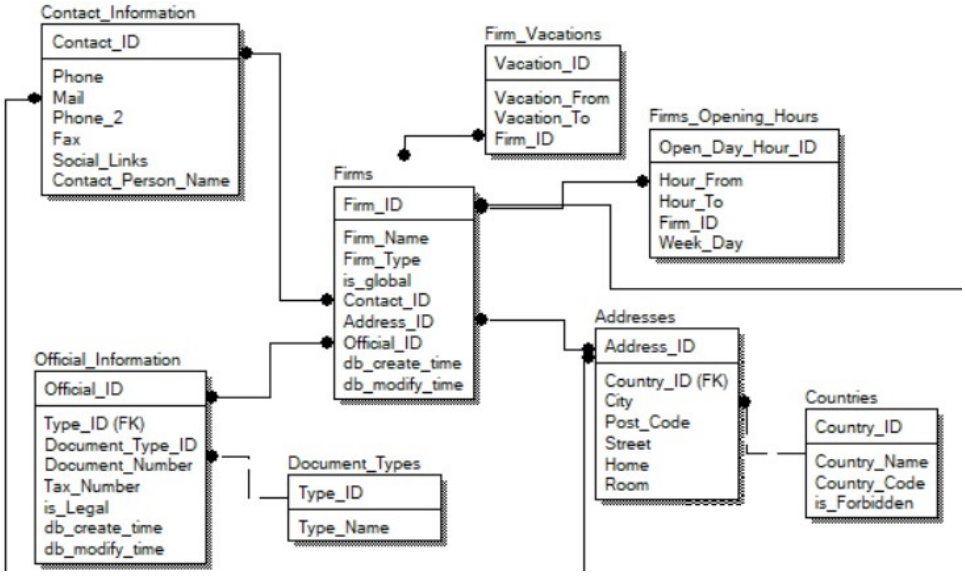


Рис. 1.3. Таблиці даних, які відносяться до постачалників у схемі Personal_Data

Таблиці, що відносяться до головної таблиці постачальників і фірм-постачальників, надають усі дані для контакту, юридичну інформацію та інформацію щодо робочих годин і робочих днів.

Таблиця Addresses. Має в собі адреси для доставок (для клієнтів) і адреси постачальників. Має зовнішнє з'єднання з таблицею Countries. Таблиця Countries містить у собі, крім іншого, колонку is_Forbidden, яка відповідає за те, чи можливий у цю країну експорт товару, або імпорт з неї. Оскільки низка країн є забороненими, з різних причин, зокрема політичних, економічних та інших. То необхідно позначати країни і не допускати запис постачальників або клієнтів із цих країн. У разі, якщо з часом країна стає забороненою (отримує статус is_Forbidden - true), всі постачальники і покупці з цієї країни ігноруються і не приймаються при розрахунку доставок або поставок.

Таблиця Official_Information містить у собі юридичні дані (що стосуються фірм постачальників і фірм постачальників), необхідні для виконання бухгалтерського обліку, і перевірки легальності та можливості доставки/поставки товару. Таблиця з'єднана зовнішнім ключем із таблицею Document_Types, яка містить у собі список дозволених і допущених документів для підтвердження юридичної сили під час реєстрації, виконання закупівель, виконання послуг та інше. Якщо фірма отримує значення в колонці is_Legal - false, через невідповідність юридичних компонентів або з інших причин, усі можливості взаємодії з цією фірмою ігноруються. У разі, якщо такий прапор висить протягом 1 року - фірма видаляється з реєстру фірм, процедурами для очищення бази даних від засмічення. Використання цього атрибута гарантує чесне виконання своїх послуг, і запобігає виникненню конфліктів на ґрунті недобросовісних постачальників/доставщиків.

Таблиця Firm_Vacations містить у собі дані, щодо яких фірма йде у відпустку, або перестає працювати на якийсь проміжок часу. Однак, може трапитися так, що фірма перестає працювати або цілком закривається, у такому разі всі її дані видаляються з усіх таблиць процедурами очищення від засмічення.

Таблиця Firm_Opening_Hours містить у собі дані щодо робочих днів у фірмі. Оскільки фірма може змінити свій розклад, створено процедури, що дають змогу змінювати вже створені робочі плани фірми на певні години або дні тижня. Процедури підбору доставки/поставки стежать за тим, щоб під час вибору

поставки/доставки фірма, яка закрита в потрібний проміжок часу, не була доступна для вибору.

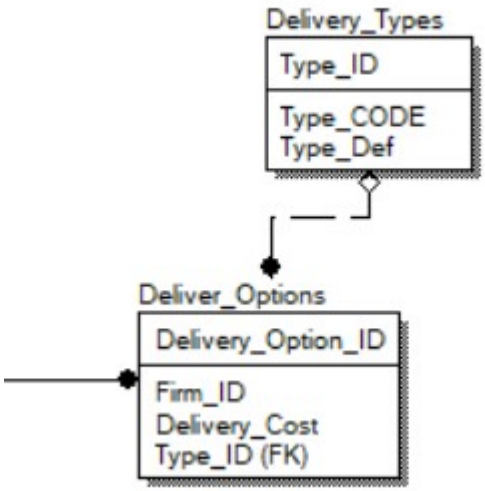


Рис. 3.4. Таблиці даних, які відносяться до фірм-доставників товару у схемі Personal_Data

Для фірм-доставщиків товару створено таблицю Delivery_Options, яка містить у собі інформацію про те, які послуги доставки надає фірма, що має в собі тип "Доставка". Кожна фірма може визначити собі необмежену кількість типів (які записані в таблиці Delivery_Types) доставок, як показано на Рис. 3.4.. Приклад такого типу: "Доставка під двері", "Доставка на роботу" та інше. Кожна фірма додатково може встановити собі константну ціну на доставку товару клієнту.

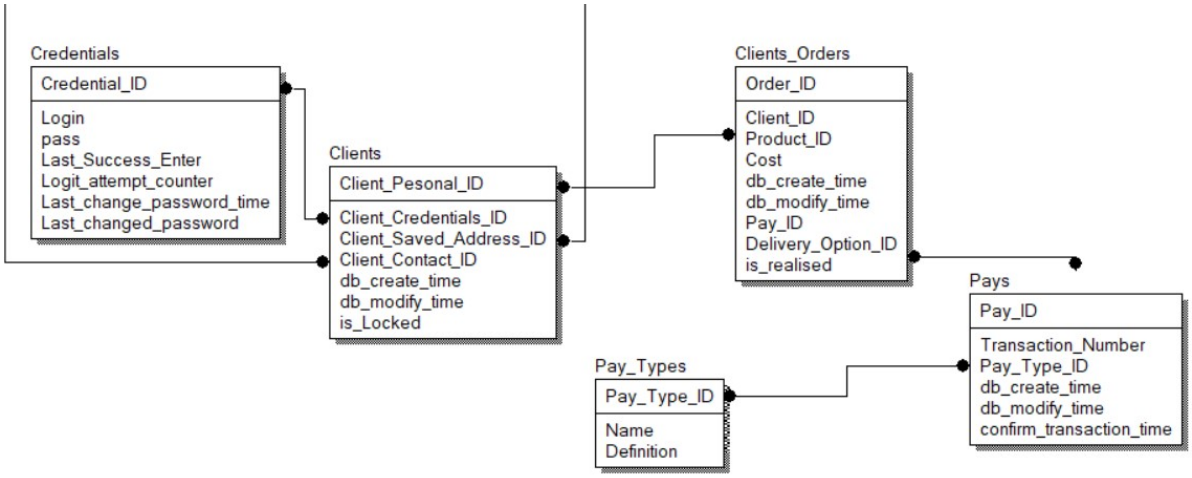


Рис.3.5. Таблиці даних, які відносяться до даних користувачів у схемі Personal_Data

Таблиці, створені для зберігання інформації про користувача та його замовлення, діляться на дві частини (представлено на Рис 3.5.). Перша частина - відповідає за дані клієнта під час його входу або спроб увійти:

Таблиця Credentials відповідає за збереження даних користувача про його спроби увійти в особистий кабінет. Оскільки там містяться особисті дані користувача, то у відкритому вигляді їх не можна надати з міркувань безпеки. Для усунення цієї помилки безпеки, всі паролі будуть зберігатися за допомогою шифрування. Доступ до таблиць буде ізольованим, за допомогою процедур читання і транзакційного методу вставки даних, при повній ізоляції зовнішнього впливу. У таблиці також є колонка, що відповідає за зчитування кількості неправильних входів користувача на сайт. У разі досягнення певного порогу, який можна задати в налаштуваннях сервісу, обліковий запис користувача буде заблоковано, з наданням йому ключа для повернення облікового запису в робочий стан. У разі, якщо акаунт перебуває в режимі офлайн, через блокування, або через відсутність відвідувань з боку користувача, його буде видалено процедурами очищення бази даних від засмічення.

Друга частина ґрунтується на замовленнях користувача. Ділиться вона на такі таблиці:

Client_Orders. Таблиця відповідає за користувацькі замовлення. Зберігає в собі дані, що стосуються оплати, яка з'єднана з таблицею Pays. Таблиця Client_Orders зберігає в собі дані, чи було замовлення реалізовано (завдяки колоні is_realised), у разі, якщо замовлення довго не є реалізованим (від моменту створення замовлення, до зміни прапора is_realised минула встановлена в застосунку максимальна кількість часу), процедури, які відслідковують стани замовлення, надішлють повідомлення-інформоповорот власникові магазину, доставникам та користувачеві, сповіщення:

Користувачеві - що його замовлення продовжує комплектуватися і приноситься вибачення за тривале очікування

Власнику магазину - що замовлення не реалізовано протягом певного часу, що перевищує встановлений час, і що необхідно вжити заходів. Відправляється також інформація щодо фірми, на якому етапі перебуває замовлення, для того, щоб наприкінці кварталу можна було визначити найменш надійні фірми для співпраці.

Фірмі, що відповідає за цей етап замовлення - повідомлення про прострочене замовлення і повідомлення про те, що якщо замовлення не буде виконано, то це загрожує штрафами.[5]

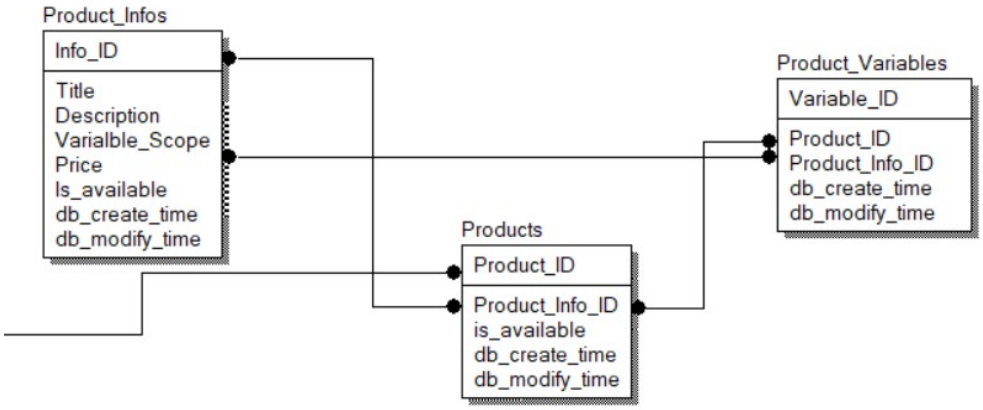


Рис.3.6. Таблиці даних, які відносяться до товарів у схемі Personal_Data

Таблиці, які мають у собі інформацію про товар, діляться на 3 частини, як представлено на Рис 3.6:

1 частина - це основна таблиця з товарами Products. Вона містить у собі посилання на таблицю з даними товару, такими як назва товару, його вартість, його характеристики у форматі JSON і його доступність.

2 частина - таблиця з інформацією про товар, яка може зберігати в собі кілька копій одного товару, з різними порівняльними характеристиками, за колоною Variable_Scope. Використання такого підходу дає змогу детально кастомізувати кожен товар щодо того, якими властивостями він може володіти за різної комплектації.

3 частина - таблиця, що відповідає за збереження в собі посилань на різні варіації товару, залежно від обраної конфігурації.

Таблиця Products має в собі колону is_available, яку встановлюють за допомогою процедур, що відстежують, чи є в наявності хоча б одна версія товару, з прапором is_available - true. У разі, якщо товару довго немає в наявності (відсутність товару триває довше, ніж встановлений у програмі максимум часу відсутності товару на складі), його видаляють із бази процедурами очищення, щоб він не займав місце.

3.1.1.5. Процедури

Використання збережених процедур для читання і запису даних у SQL Server має низку переваг:

- Перевикористання коду: Збережені процедури можна використовувати в різних частинах додатка, що дає змогу істотно скоротити кількість коду і поліпшити його читабельність.
- Поліпшення продуктивності: Збережені процедури компілюються й оптимізуються базою даних, що може збільшити продуктивність програми.
- Безпека: Збережені процедури можуть бути захищені доступом до них і обмеженням прав доступу до даних.
- Управління транзакціями: Збережені процедури можуть бути використані для управління транзакціями, що дає змогу забезпечити цілісність даних.
- Управління конкуренцією: Збережені процедури можна використовувати для управління конкуренцією доступу до даних і забезпечення їхньої цілісності.
- Зручне управління даними: Збережені процедури можуть бути використані для зручного управління даними, наприклад, для виконання масових оновлень або видалення.
- Зручне управління структурою таблиць: Збережені процедури можуть використовуватися для зручного керування структурою таблиць, наприклад, для створення або зміни стовпців.

Для забезпечення надійного рівня безпеки та швидкодії застосунку і бази даних, було створено два типи процедур:

Процедури очищення. Даний тип процедур відноситься до процедур оптималізації використання місця на диску, займаним базою даних. Кожна процедура очищення приймає вхідний параметр - час. Процедура відстежує наявні записи і, якщо запис не виконує вимог до збереження (прикладом тому може слугувати ненадійна фірма доставки/поставки товару або недоступний товар, що перевищив свої строки зберігання в базі даних), його буде видалено. При створенні процедур, основна увага має приділятися оптималізації видалення великої кількості записів > 1 000 000, у зв'язку з тим, що видалення такої кількості записів за один захід спричинить високі витрати продуктивності і час роботи ЦП. У такому разі видалення має відбуватися поетапно, з виділеним параметром batchRemoveSize, який відповідає за те, скільки записів може бути усунуто під час проходження одного циклу. Використання підходу до усунення за допомогою batchSize дає змогу уникнути високих навантажень у тривалому часі, але не дає змоги уникнути високих навантажень загалом. Для запобігання перевантаженню сервера бази даних варто передбачити безвідмовний механізм запуску процедур очищення у визначені терміни (перевірка раз на день є найоптимальнішою) і усунення записів, що невиконують вимоги.

Процедури допуску. До цього типу процедур належать процедури вставки даних і зчитування даних. Через те, що кожна таблиця є прямо чи опосередковано пов'язаною з іншими, процедури мають передбачати можливості відкоту до ранньої версії змін, без шкоди для структури ази даних або її даних. Кожна процедура допуску виконується в транзакції, рівень ізоляції якої найвищий. Кожна процедура, додатково, запитує як параметр персональний ключ додатка, персональний ключ людини, яка робить запит на відображення. Використання методу подвійної автентифікації дасть змогу, насамперед, на запобігання витоку даних, якщо головний сервіс виявиться зламаним. Процедури допуску зобов'язані стежити також за тим, що б вставлені дані, через ці процедури, виконували вимоги актуальних записів і не були простроченими, або неактивними.

Вставлення застарілих або неактивних записів має запобігати ще на моменті валідації даних, видаючи помилку про те, що дані не виконують заздалегідь поставлених критеріїв вставки.

3.1.2 Головний бекенд сервіс

3.1.2.1. Використані технології

Як головну мову програмування бекенд-сервісу було обрано Java, з огляду на її переваги:

Платформонезалежність: Java код може бути запущений на будь-якій платформі, що підтримує Java Virtual Machine (JVM), що робить його ідеальним для розробки крос-платформених додатків.

Багата бібліотека: Java має величезну кількість готових бібліотек і фреймворків, які можна використовувати для розроблення застосунків, що істотно скорочує час розроблення.

Безпека: Java має механізми безпеки, які допомагають захистити додатки від різних загроз, таких як впровадження коду або злом.

Підтримка багатопоточності: Java має вбудовану підтримку багатопоточності, що дає змогу розробникам створювати додатки, які можуть використовувати всі ресурси багатоядерних систем.

Стабільність і довговічність: Java є потужною і надійною мовою, яка використовується в безлічі великих корпоративних проектів і має велику історію використання.

Активна спільнота: Java має дуже активну спільноту розробників, яка постійно розвиває та покращує мову, а також надає документацію та підтримку. Це робить Java мовою з великою ком'юніті, яка дає доступ до безлічі бібліотек, фреймворків та інструментів[3].

Підтримка сучасних технологій: Java є сучасною мовою, яка постійно розвивається і підтримує останні технології, такі як Java 8, 9, 10, 11, 12 і вище. Також має безліч фреймворків і бібліотек, таких як Spring, Hibernate, JavaFX тощо.[3]

Як головний фреймворк проєкту було обрано Spring Boot. Використання Spring у проєкті має низку переваг:

Контейнер бінів: Spring надає контейнер бінів, який автоматично створює, конфігурує та керує об'єктами програми.

Впровадження залежностей: Spring підтримує впровадження залежностей, що дає змогу створювати незалежні компоненти додатка, які легко тестуються.

Управління транзакціями: Spring надає механізм управління транзакціями, що дає змогу забезпечити цілісність даних у застосунку.

Підтримка баз даних: Spring надає інтеграцію з різними СУБД, що дає змогу легко працювати з базами даних у застосунку.

Веб-додатки: Spring надає модулі для розроблення веб-додатків, включно з підтримкою MVC, REST, автентифікацією та аутентифікацією.

Модульність: Spring має модульну архітектуру, яка дає змогу розробникам використовувати тільки ті компоненти, які їм необхідні, що робить додаток легшим і гнучкішим.

Підтримка сучасних технологій: Spring постійно розвивається і підтримує останні технології, як-от Java 8, 9, 10, 11, 12 і вище, а також безліч інших технологій, як-от Spring Boot, Spring Cloud тощо.

Активна спільнота: Spring має дуже активну спільноту, яка постійно розвиває та покращує фреймворк, а також надає документацію та підтримку.

3.1.2.2. Структура контролерів проекту

Для кожного елемента проєкту було створено свій контролер. Для відображення, зміни та створення нового товару створено наступний контролер:

```
@RestController
@RequestMapping("/product")
public class ProductController {
    @Autowired
    private ProductService productService;
    @Autowired
    private UserService userService;

    @PreAuthorize("hasRole('ROLE_ADMIN')")
```

```
    @PutMapping("/edit/{id}")
    public ResponseEntity<Product> editProduct(@PathVariable("id") Long id,
@RequestBody Product product) {
        Product updatedProduct = productService.updateProduct(id, product);
        return new ResponseEntity<Product>(updatedProduct, HttpStatus.OK);
    }

    @PreAuthorize("hasRole('ROLE_ADMIN')")
    @PostMapping("/save")
    public ResponseEntity<Product> saveProduct(@RequestBody Product product) {
        Product savedProduct = productService.saveProduct(product);
        return new ResponseEntity<Product>(savedProduct, HttpStatus.CREATED);
    }

    @GetMapping("/show/{id}")
    public ResponseEntity<Product> showProduct(@PathVariable("id") Long id) {
        Product product = productService.getProductById(id);
        return new ResponseEntity<Product>(product, HttpStatus.OK);
    }
}
```

У цьому контролері використовується анотація @PreAuthorize із зазначенням ролі для доступу до методів редагування та збереження. Замість @Controller використовується @RestController, який дає змогу повертати об'єкти у форматі JSON. Замість @GetMapping використовується @GetMapping для операції show, щоб повертати об'єкт у форматі JSON. Методи edit і save використовують @PutMapping і @PostMapping відповідно, щоб відображати тип запиту HTTP, який вони обробляють. У методі show використовується метод getProductById з ProductService для отримання об'єкта продукту за ідентифікатором і повертається у вигляді відповіді зі статусом OK.

Для входу й автентифікації користувачів було створено такий контролер:

```
@Controller
public class LoginController {
    @Autowired
```

```
    private UserService userService;

    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @PostMapping("/login")
    public String login(@RequestParam("email") String email,
                        @RequestParam("password") String password,
                        Model model) {
        User user = userService.login(email, password);
        if (user != null) {
            model.addAttribute("user", user);
            return "dashboard";
        } else {
            model.addAttribute("error", "Invalid email or password");
            return "login";
        }
    }
}
```

У цьому контролері використовується метод GET для відображення форми входу на сторінці /login, і метод POST для обробки даних форми входу. Метод login з UserService використовується для перевірки введених даних і повернення об'єкта User. Якщо об'єкт User не дорівнює null, користувач перенаправляється на сторінку особистого кабінету, в іншому випадку показується повідомлення про помилку.

userService в даному контролері має в собі логіку збереження сесії, файлів куки і аналізу частоти входу користувача для запобігання атак на сервер.

Під час створення сервісу використовувати структуру ООП дає змогу розділяти логіку бізнес-логіки та логіку доступу до даних. Це дає змогу зробити код більш читабельним і зрозумілим, а також легше модифікувати та масштабувати. Крім того, поділ логіки на шари дає змогу ізолювати шар бізнес-

логіки від деталей реалізації доступу до даних, що робить код надійнішим і легше піддається тестуванню.

3.2. Розробка фронтенду

3.2.1. Створення прототипів

Задля створення інтерфейсу користувача необхідно чітко скласти вимоги та прототип для майбутньої верстки сторінки. Для створення прототипу використовується безкоштовний інструмент Figma.

Прототипування головної сторінки покупця розпочинається з планування головних елементів:

- Кнопки
- Головне меню
- Форми для товару
    - Стан завантажений
    - Стан у завантаженні
    - Стани доступності товару
- Анімації завантажень

Коли користувач заходить у свій особистий кабінет, він бачить сторінку з назвою "Вхід" або "Реєстрація", залежно від того, на якій зі сторінок він перебуває.

На цій сторінці він вводить логін і пароль, які під час реєстрації були введені під час створення облікового запису.

У випадку з логіном йому присвоюється унікальне ім'я користувача, а з паролем - випадковий набір символів.

Коли користувач вводить свої дані, вони в обов'язковому порядку перевіряються на валідність.

Якщо дані вводяться неправильно, то користувачеві видається помилка.

3.2.1.1. Кнопки

Одним з головних інтерактивних елементів кожної програми, сайту, аплікації є кнопки. Під час їх проектування необхідно взяти під увагу такі характеристики, як:

Колір. Для обрання кольорів кнопок, візьмемо за основу стиль Material Design, створений Google. Виходячи з цього, генерування домінуючих кольорів перекладемо на генератора кольорів.

Форма. Переважними формами стилю Material Design є закруглені елементи. В зв'язку з цим, приймаємо, що кожна кнопка мусить мати закруглені краї, та/або бути округлої форми.

Текст. Головної проблемою створення тексту для кнопки є залежність тексту від розмірів кнопки. Створення довгих текстів, або вставлення довгих слів збільшують розмір кнопки, що створює анти-дизайн. Для кінцевого споживача кнопки з таким анти-дизайном будуть занадто «важкими» і не викликатимуть довіри і та бажання натиснути на неї.

Згідно з даними критеріями створюємо кнопки для наступних груп:

Група для додання товару до кошику

Основний колір: **#b2ff59**

Розгорнутий вигляд кнопки для замовлення представлена на Рис.3.7

Рис.3.7. Кнопка для замовлення товару у розгорнутому вигляді

Коротка форма кнопки для замовлення товару представлена на Рис.3.8

Рис.3.8. Кнопка для замовлення товару у скороченому вигляді

Також, після натиснення, користувача необхідно інформувати про те, що товар був доданий до кошику. Для інформування відбуваються кілька подій: кошик змінює свою іконку з пустої на повну, а кнопка мусить дати користувачу інформацію, що його дія була зареєстрована у системі.

Форми кнопки, яка інформує користувача про те, що його замовлення додано до кошику можуть бути наступними: скорочена форма, яка представлена на Рис.3.2.1.1.3., або розгорнута форма кнопки, яка представлена на Рис.3.9 або на Рис.3.10

Рис.3.9. Скорочена форма кнопки для замовлення, яка повідомляє користувачу, що його замовлення було зареєстровано у базі даних

Для кнопки у розгорнутому вигляді можна дати кілька виглядів, в залежності від того, яким чином кнопка мусить повідомляти користувача про його дію.

Першим типом інформування можна прийняти зворотній відклик за допомогою тексту

Рис.3.10. Розгорнута форма кнопки для замовлення, яка повідомляє користувачу, що його замовлення було зареєстровано у базі даних

Другим типом можна прийняти зворотній відклик за допомогою іконок

Рис.3.11 Розгорнута форма кнопки для замовлення, яка повідомляє користувачу, що його замовлення було зареєстровано у базі даних, за допомогою іконки

При натисканні кнопки, для привертання уваги користувача належить застосувати анімацію

Також, необхідно взяти до уваги те, що товар може бути недоступний, або наразі не мати можливості замовлення. Для цього необхідно додатково узгодити кольори з головною кольоровою палітрою.

Для кнопок, стан яких «Наразі замовлення неможливе» пропонується обрати колір **#fbc02d**, який представлено на Рис.3.12

Рис.3.12. Обраний колір #fbc02d для кнопок інформуючи користувача про те, що статус товару - "Товар у дорозі"

Згідно з даним кольором кнопки мусять мати відповідні іконки, або текст для інформування користувача.

Головною іконкою, прийнятою для встановлення на кнопку, є іконка з групи Material Design – „Notification”, яка повідомляє користувача про те, що наразі немає можливості замовлення товару, але користувач може бути повідомлений про терміни доступності товару. Кнопка представлена на Рис.3.13

Рис.3.13. Скорочений вигляд кнопки яка інформує користувача про те, що він може бути повідомлений про терміни доступності товару

Створення кнопки з текстом для інформування користувача, що товар скоро стане доступним має певні складнощі. Однією з яких є необхідність донести до уваги користувача інформацію про те, що товар існує і є у продажі, але на даний момент не є на складі. Існує кілька варіацій інформування користувача текстом: «Замовити», «Інформуй мене», «Дізнатися про дату доступності».

Найкращим вибором для користувача може бути текст «Замовити», оскільки він відповідає головному вимаганню користувача – замовити бажаний товар.

Відповідно до заданих критеріїв кнопка для повідомлення користувача буде мати наступний вигляд, який представлено на Рис.3.14

Рис.3.14. Розгорнутий вигляд кнопки яка інформує користувача про те, що він може бути повідомлений про терміни доступності товару

Група для перемикання опцій товару

Для конфігурації опцій товару належить узгодити також стиль перемикачів опцій. Виходячи з Material Design вони мусять мати закруглені рамки, та мати другорядний колір, без явного виділення елементу на перший графічний план. Також, необхідно за допомогою іконки показати користувачу, що саме мусить виконувати дана кнопка. Отже, перемикач мусить мати текст з обраною опцією та певну іконку, демонструючи те, що при натисканні на неї відкриється ще більше опцій.

Перемикач опцій може мати наступний дизайн, який представлено на Рис.3.15

Рис.3.15. Перемикач оцій товару у складеному стані

При натисканні кнопки розгортання панелі опцій товару належить узгодити також момент, коли товар може бути недоступним, або бути за акцією. Приклад розгорнутої панелі показаний на Рис.3.16

Рис.3.16. Розгорнутий вигляд перемикача опцій товару

Аби не ускладнювати дизайн картки товару, належить додати максимально 2-3 опцій для конфігурації товару (залежить від конфігурування опцій, згідно з виробником).

Група кнопок для головного меню

Для головного меню належить взяти під увагу елементи головного дизайну. До даних елементів можна віднести кнопки наступних типів:

- Перемикачі
- Розгорнуті кнопки
- Кнопки навігації

Кнопки-перемикачі. Стосуються для швидкого перемикання елементу, або стилю відображеного елементу. Використовуються у місцях, де не має потреби переходу на іншу сторінку. Прикладом таких кнопок можуть бути кнопки перемикання тем, перемикання мов, перемикання валют, тощо.

Для створення кнопок перемикачів належить узгодити кілька критеріїв її відображення.

Першим критерієм може бути те, що кнопка не мусить виділятися від інших формами, задовгим текстом, іконками. Головною проблемою даного критерію є те, що кнопка, яка мусить не відрізнятись від інших, мусить привертати увагу користувача, даючи йому можливість зрозуміти, що саме вона має у собі.

Другий критерій стосується того, що меню в середині кнопки не мусить бути а ні завеликим а ні замалим. Користувач мусить розуміти, що саме категорія має у собі. Замалий опис обраного пункту меню призведе до того, що користувач не буде знати чи є необхідний йому товар у даній категорії. З іншої сторони, при завеликому описі, або завеликій кількості підкатегорій, користувач втомиться шукати необхідну йому. У обох випадках, користувач покине сторінку.

Приклад кнопки, яка використовується для показу категорій товарів, у стані необраному, без фокусу на ній представлена на Рис.3.17

Рис.3.17. Кнопка для обрання категорії товарів. Без фокусу на кнопці

Під час наведення комп'ютерної миші на кнопку вона мусить дати зворотній відклик для користувача про те, що відбулась подія на сторінці. Кнопка може змінити свій колір на своїх гранях для цього. Приклад кнопки, на яку наведений курсор представлена на Рис.3.18

Рис.3.18. Кнопка обрання категорії товару, на яку наведений курсор

Під час наведення курсору, кнопка змінює свій сірий колір на колір: **bbdefb** , який є обраним кольором з палітри кольорів стилю Material Design.

Кнопка також мусить повідомити користувача, якщо він натисне на неї, але для створення зворотного відклику належить додати ще один елемент – випадаючий список категорій, який за своїм стилем мусить бути з кнопкою однорідним.

У розгорнутому вигляді кнопка має наступний вигляд, який представлено на Рис.3.19

Рис.3.19. Меню категорій у розгорнотому стані

Кнопки з додатковими опціями. Використовуються, коли необхідно додати додаткові критерії вибору, або категорії. Прикладом таких кнопок можуть бути кнопки, які використовуються для навігації по категоріям товарів.

Стиль такої кнопки може бути як звикле гіперпосилання, при натисненні на яке, користувач переміщався по сайту.

Кнопки для навігації по сторінках сайту. Використовуються, для обрання на яку сторінку сайту може перейти користувач. Звичайна кнопка, яка фарбується у свій колір, якщо користувач знаходиться на сторінці кнопки.

Прикладами таких кнопок можуть бути наступні, зображення, демонструючі кнопки без фокусу на ній, та у фокусі як представлено на Рис 3.20.

Рис.3.20. Кнопки для переміщення по сайту, без фокусу та з фокусом на ній

3.2.1.2. Головне меню

Для прототипування головного меню, приймаємо раніше створені кнопки, та технічні вимоги, задані раніше у пунктах технічних вимог для системи та фронтенду.

Приймаємо, що конфігурація категорій та головних сторінок у магазині перекладена на розробника. З-поміж багатьох варіантів найкращі наступні:

1. Головна. Перехід на головну сторінку. Належить також звернути увагу, що головна сторінка – має бути сторінкою з пропозиціями і мати сенс переходити на неї, в іншому випадку, головна сторінка буде слугувати «Пустою сторінкою» для сайту, що не є доброю практикою
2. Категорії. Кнопка мусить виконувати дві функції. Перша – при наведенні на неї, «випадає» список з можливими категоріями для обрання. Іноді, користувач не знає, яка саме річ йому потрібна, тому друга функція кнопки – відобразити усі доступні товари, з можливістю подальшого його вибору.
3. Доставка. Даний пункт є більш інформаційним для користувача. Він відправляє користувача на сторінку, де він може знайти інформацію про можливості доставки товару. Інформація мусить відповідати на 3 питання:
    a. Чи можуть доставити це в моє місто?
    b. Чи можуть доставити це мені додому?
    c. Скільки часу та грошей візьме придбання та доставка товару?
4. Блог. Для інтернет магазину є важливим підтримувати рівень маркетингу та зацікавленості користувачів. Для виконання даного завдання існують блоги магазинів, які протягом певного часу дають користувачеві корисну інформацію. Наприклад:
    a. «Порівняння фільтрів» - допоможе клієнтові обрати необхідний фільтр, який може бути бестселлером, або товаром з гарними технічними характеристиками, але не мати великої кількості замовлень через низьку відомість фірми – виробника
    b. Конкурси наших клієнтів. Дає можливість надихнутись клієнтові і отримати відчуття того, що даний сайт – не тільки місце до купування товару, а й місце спілкування людей, та натхнення
    c. Тощо.
5. FAQ. Розділ запитання – відповідь. Служить для швидкого пошуку відповіді на конкретне запитання у скороченому вигляді.

Створений концепт меню представлено на Рис.3.21.

Рис.3.21. Головне меню сайту

У випадку зміни стилю, або кольору система має виконувати зміни в усіх елементах автоматично, без залучення розробника до зміни усіх компонентів. Для виконання цієї вимоги, належить взяти під увагу розташування та розширення ресурсів системи.

3.2.1.3. Форми для товару

Для створення форми товару, належить взяти під увагу те, що сторінка може не відкриватися до 10 секунд, у цьому випадку, належить повідомити користувачу те, що сторінка завантажується, а не зависла, чи викинула помилку у консоль.

Цілковита завантажена форма товару представлена на Рис.3.22

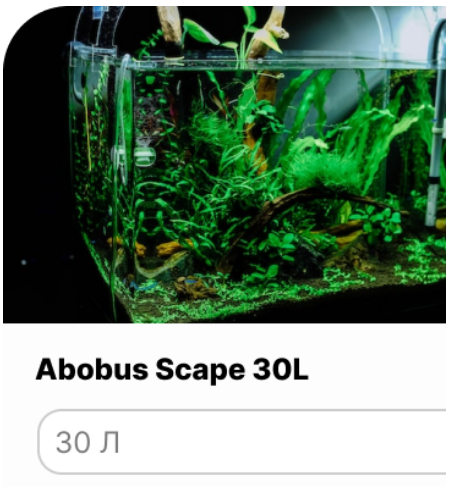


Рис.3.22. Форма завантаженного товару

Вище показана форма демонструє скорочену форму доступного товару товару, з обраною по-замовчуванню опцією товару. Розгорнута опція з вибором форми представлена на Рис.3.23

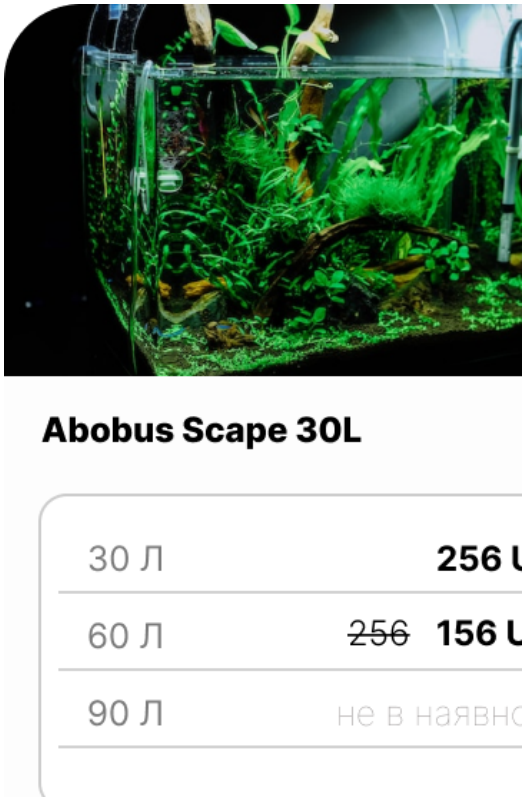


Рис.3.23. Завантажена розгорнута форма товару

Під час обрання опції з розгорнутої форми, користувач може взаємодіяти з усіма доступними компонентами форми, такими як: «Додати до кошика», відкрити сторінку товару, тощо. Під час завантаження сторінки, необхідно взяти також під увагу, щоб усі її елементи (кнопки, зображення, тощо) не були доступні для натискання, щоб не викликати помилку. Створенний дизайн картки, яка завантажується динамічно представлено на Рис 3.24.

Рис.3.24. Форма товару під час завантаження

Використання завчасно відрендерених анімацій дозволяє зменшити навантаження на пристрій клієнта. У такому вигляді, кнопки або інші елементи керування формою недоступні.

У випадку, якщо товар не є в наявності, форма має повідомляти про це користувачу. Наприклад, форма може замалювати кнопку «До кошика» сірим кольором і зробити її недоступною, як це представлено на Рис.3.25.

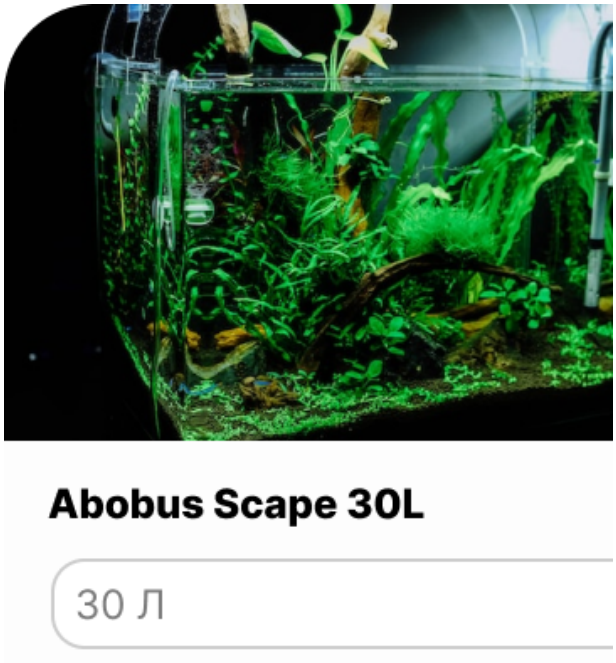


Рис.3.25. Карта товару, який відсутній на складі

3.2.2.Реалізація прототипів

Для реалізації заздалегідь створених прототипів сторінок можна використовувати шаблони або фреймворки для веб-розробки, як-от Bootstrap, Foundation або Materialize. Ви також можете використовувати CSS-бібліотеки для швидкої та зручної верстки, такі як Bulma або Tailwind CSS. Якщо ви використовуєте JavaScript-фреймворк, такий як React або Angular, ви можете створювати компоненти, які можна використовувати кілька разів на різних сторінках сайту.

У нашому випадку, оскільки сторінки мають генеруватися динамічно, під час використання AJAX, нам варто розглянути структуру з динамічно підтримуваними бібліотеками. Сторінки повинні повторюватися і лише видозмінюватися для різних компонентів. За таких вимог сторінки

створюватимуться за допомогою фреймворку React, спільно з бібліотекою Bootstrap 5. Використання цього поєднання призведе до того, що система буде динамічно масштабованою, з низьким порогом входу в технічну частину.

React - це JavaScript-бібліотека, розроблена Facebook, яка використовується для створення інтерактивних користувацьких інтерфейсів. Він дає змогу розробникам створювати компоненти, які можна використовувати кілька разів на різних сторінках сайту та оновлювати їх динамічно, коли дані змінюються. React використовує технологію віртуального DOM, що дає змогу швидко оновлювати тільки частини інтерфейсу, що змінилися, а не всього документа.

React популярний завдяки своїй модульності, яка дає змогу розбивати інтерфейс на безліч маленьких і легко підтримуваних компонентів. React використовує JSX - синтаксис, який дає змогу написати JavaScript код, що має вигляд HTML, роблячи код більш читабельним і зрозумілим. React також має широку і розвинену екосистему, з безліччю додаткових бібліотек та інструментів, як-от React Router для управління маршрутизацією і Redux для управління станом програми.[7]

Верстка сторінки - це процес створення веб-сторінки використовуючи HTML, CSS і JavaScript. Ось загальні етапи верстки сторінки:

- Розробка HTML-коду: написання HTML-коду, який структурує та описує вміст сторінки.
- Розробка CSS-коду: написання CSS-коду, який визначає зовнішній вигляд сторінки, включно зі шрифтами, кольорами, відступами та іншими стилями.
- Розробка JavaScript-коду: написання JavaScript-коду, який додає інтерактивність і функціональність на сторінку.
- Тестування та налагодження: перевірка сторінки на різних пристроях і браузерах, а також виправлення помилок і недоліків.
- Деплоймент: Розміщення веб-сторінки на сервері для доступу до неї з Інтернету.

- Підтримка та оновлення: Після запуску сайту, може знадобитися підтримка, включно з виправленням помилок і оновленням вмісту. Це важливо для збереження його функціональності та безпеки.

Використання фронтенда як частини мікросервісної архітектури означає, що фронтенд є окремим мікросервісом, який спілкується з іншими мікросервісами для отримання даних і оновлення інтерфейсу користувача.

Це дає змогу гнучкіше розвивати та масштабувати фронтенд і бекенд, а також підвищує надійність системи. У такій архітектурі можна використовувати різні мови програмування і технології для різних мікросервісів. Важливо мати на увазі, що це також вимагає наявності добре спроектованого API для взаємодії між мікросервісами.[7]

У мікросервісній архітектурі фронтенд може спілкуватися з іншими мікросервісами за допомогою REST API, gRPC або інших способів взаємодії. Це дає змогу гнучкіше розвивати й масштабувати фронтенд і бекенд, а також підвищує надійність системи, адже кожен мікросервіс може бути оновлений або замінений незалежно від інших.

На додаток до цього, використання фронтенда як мікросервісу також дає змогу ефективніше використовувати ресурси і скоротити витрати на розвиток, оскільки різні команди можуть працювати над різними частинами системи незалежно. Також можна використовувати різні технології та фреймворки для різних частин системи, а не бути зав'язаним на одну технологію або фреймворк для всієї системи.

Загалом використання фронтенда як мікросервісу в мікросервісній архітектурі дає змогу підвищити гнучкість, масштабованість і надійність системи, а також поліпшити ефективність розробки та використання ресурсів.

Використання JavaScript і React для розробки фронтенду в мікросервісній архітектурі може бути корисним варіантом.

- React є популярною JavaScript-бібліотекою, яка використовується для створення інтерактивних і динамічних інтерфейсів. Він має компонентний підхід, який дає змогу розбити інтерфейс на незалежні

частини та використовувати їх багаторазово. Це допомагає зробити код більш читабельним і підтримуваним.

- JavaScript підтримується у всіх сучасних браузерах і використовується для створення інтерактивних і динамічних веб-сайтів і додатків. Він може використовуватися на клієнтській і серверній стороні, що робить його гнучким і потужним інструментом для розробки фронтенду в мікросервісній архітектурі.

Використання JavaScript і React як фронтенду в мікросервісній архітектурі може принести безліч переваг, як-от гнучкість, масштабованість, надійність і ефективність розробки. Однак, важливо мати на увазі, що залежно від конкретного завдання і потреб проєкту, інші мови і технології можуть бути більш придатними.

Докеризація інтерфейсу - це процес створення Docker-образу інтерфейсного додатку та запуск його в Docker-контейнері. Це дозволяє легко розгортати і запускати додаток в різних середовищах без необхідності ручного налаштування або конфігурації. Процес зазвичай включає в себе створення Docker-файлу, який визначає залежності та конфігурацію для програми, створення образу з використанням цього файлу, а потім запуск образу в контейнері. Це можна зробити за допомогою інструментів командного рядка, таких як Docker CLI.

Переваги контейнеризації:

- Переносимість: Контейнери Docker можуть працювати в будь-якому середовищі, яке підтримує Docker, а це означає, що додаток можна легко переміщати між різними середовищами, такими як розробка, тестування і виробництво. Контейнери Docker можуть бути перенесені між різними операційними системами та оточеннями, оскільки вони використовують образи, які містять усе необхідне для роботи програми, включно із залежностями та налаштуваннями. Це означає, що контейнер, запущений на одній машині, може бути перенесений на іншу машину і працюватиме так само, як і на першій.

- Ізоляція: Контейнери забезпечують рівень ізоляції додатка, що може допомогти зменшити проблеми з конфліктуючими залежностями або конфігураціями між різними середовищами.
- Масштабованість: Контейнери Docker можна легко масштабувати відповідно до потреб програми, що може допомогти підвищити продуктивність і знизити витрати.
- Версійність: Образи Docker можна версіонувати і зберігати в реєстрі контейнерів, що дозволяє легко відкотитися до попередньої версії програми в разі виникнення проблем.
- Автоматизація: Докеризація фронтенду дозволяє легко автоматизувати процес розгортання, що може допомогти зменшити кількість людських помилок і підвищити ефективність.

Для запуску створеного на основі React необхідно створити Dockerfile з командами збірки і запуску. Створений Dockerfile представлений нижче:

#Dockerfile

 # Use an official Node.js runtime as the base image

FROM node:14-alpine

 # Set the working directory

WORKDIR /app

 # Copy the package.json and package-lock.json file

COPY package.json ./*

 # Install dependencies

RUN npm install

 # Copy the rest of the application code

COPY . .

 # Build the React application

RUN npm run build

 # Expose the port the application will run on

EXPOSE 3000

 # Start the application

CMD ["npm", "start"]

У цьому Dockerfile ми використовуємо офіційний образ Node.js як базовий образ, встановлюємо необхідні залежності та копіюємо код нашого застосунку в каталог робочої директорії. Потім ми запускаємо команду збірки React і відкриваємо порт 3000, на якому працюватиме застосунок. Нарешті, ми запускаємо команду "npm start" для запуску програми.

Оскільки наша система ґрунтується на контейнеризованій мікросервісній архітектурі, то для запуску всіх контейнерів у визначеному порядку й умовах потрібно створити docker-compose файл, який визначить порядок запуску та додаткові властивості, як-от: порти вводу-виводу, назви сервісів і мережі, до яких їх буде під'єднано. Початково створений docker-compose файл має такий вигляд:

```
version: '3'
services:
  aquafront:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    volumes:
      - .:/app
    environment:
      - NODE_ENV=development
    command: npm start
```

У цьому docker-compose файлі ми визначаємо сервіс aquafront і вказуємо, що докерфайл міститься в папці ./frontend і називається Dockerfile. Потім ми прокидаємо порт 3000 на наш локальний комп'ютер і підключаємо поточну директорію до каталогу /app всередині контейнера. Ми також встановлюємо змінну оточення NODE_ENV в development і запускаємо команду npm start для запуску програми.

Для запуску фронтенд-частини, залишається лише виконання комманди в терміналі

docker-compose run aquafront.

На цьому частина створення фронтенд сервісу завершена.

3.2.3. Інструкція користувача інтернет-магазину

3.2.3.1. Головна сторінка сайту

При завантаженні сторінки користувач буде головну сторінку, яка представлена на Рис.3.26.

Фільтр пошуку:

- Використовуйте фільтр пошуку, щоб знайти конкретний товар за категорією, ціною або виробником.
- Виберіть необхідні критерії та натисніть "Застосувати".

Пошук:

- Використовуйте пошуковий рядок, щоб знайти товар за ключовими словами.
- Введіть слова або фразу і натисніть "Пошук".

Карти товару:

- Використовуйте карти товару, щоб переглянути інформацію про товар у вигляді зображення і короткого опису.
- Натисніть на зображення, щоб побачити більш детальну інформацію.

Вибір товару:

- Виберіть товар з карти товару або списку результатів пошуку.
- Натисніть на кнопку "Детальніше", щоб побачити повну інформацію про товар, включно з описом, характеристиками та ціною.

Додавання товару в кошик:

- Після перегляду докладної інформації про товар ви можете додати його в кошик.
- Виберіть кількість товару і натисніть кнопку "Додати в кошик".

Перехід до кошика:

Щоб переглянути товари у вашому кошику, натисніть на іконку кошика у верхньому правому куті екрана.

Ви можете змінювати кількість товарів у кошику або видаляти товари за необхідності.

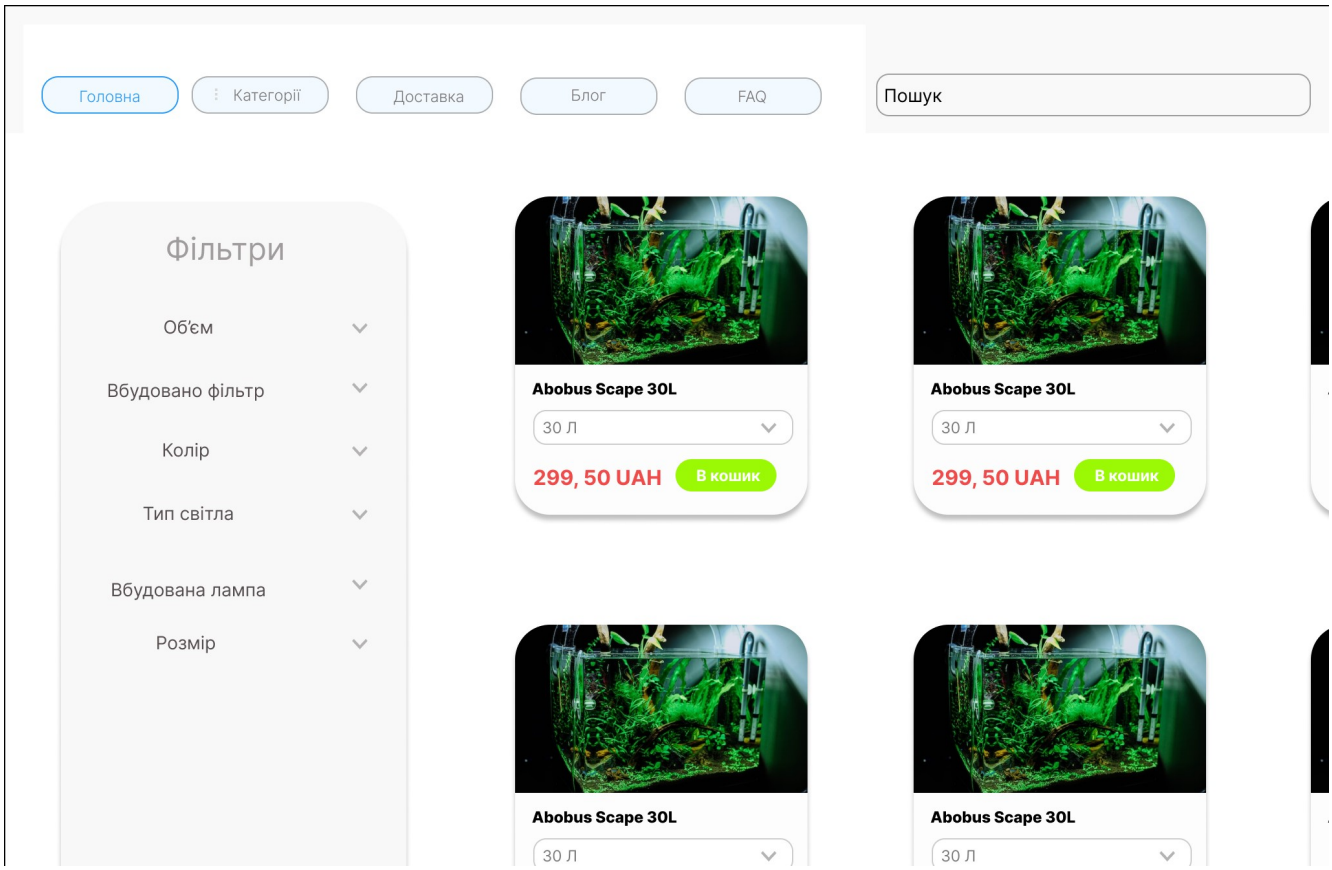


Рис.3.26. Головна сторінка магазину акваріумістики

Головна сторінка складається з кількох функціональних блоків:

1. Головна сторінка.
2. Каталог товарів.
3. Особистий кабінет.
4. Доставка та оплата.
5. Контакти.
6. Відгуки.
7. Корисна інформація (лайфхаки, поради щодо вибору тощо).

Товари на головній сторінці повинні вибудовуватися в спеціальні ряди, згідно з алгоритмом "Оцінки дій користувача". Суть алгоритму зводиться до того, що всі дії користувача записуються й аналізуються щодо того, в яких категоріях

товару він перебував найдовше, в яких категоріях товару він найбільше відкривав посилань, які категорії товару найчастіше були куплені. Згідно з цією проаналізованою інформацією можна скласти певні захоплення і бажання користувача, за допомогою штучного інтелекту продумати його подальші бажання, ґрунтуючись на заздалегідь створених попередніх установках. Прикладом вгадування бажання може слугувати встановлення CO2 в акваріумі. Приблизна система може створюватися з таких основних компонентів:

Балон із рідким або газоподібним CO2

Регулятор CO2: пристрій, який контролює тиск і потік CO2 з балона.

Контролер CO2: прилад, який вимірює концентрацію CO2 в акваріумі і регулює його потік за допомогою регулятора.

Реактор CO2: пристрій, наприклад, спайнер, який розсіює CO2 в акваріумі і допомагає його рівномірно розподіляти.

Датчик pH: прилад для вимірювання рівня pH в акваріумі, щоб контролювати рівень CO2

Додатково, може використовуватися соленоїдний насос для контролю потоку і забезпечення безпеки.

Однак, кожен з цих компонентів повинен з'єднуватися з іншими, за допомогою сполучних і ущільнювальних кілець.

Якщо користувач побажає купити новий регулятор CO2, система може запропонувати йому товари, пов'язані із заміною: ущільнювачі, кільця, перехідники та інше.

За такої умови, основна вимога для головної сторінки - не бути одноманітною і пристосовуватися для кожного користувача.

У розділі FAQ (поширені запитання) користувач може знайти відповіді на найпоширеніші запитання щодо роботи магазину, наприклад: інформацію про доставку та оплату, гарантії та повернення, способи звернення до служби підтримки та іншу корисну інформацію. Розділ FAQ також може містити інструкції та описи продуктів або послуг, що може бути корисно для покупців.[9]

Вікно входу в особистий кабінет може мати стандартний шаблонний варіант, без додаткової кастомізації. Використання такого варіанту істотно заощадить час розробника на розробці та створенні прототипу і дизайну сторінки входу. Однак, сам механізм валідації входу користувача має бути ретельно продуманий, щоб не допустити витоку інформації та відкриття дірок у системі безпеки.

3.2.3.2. Сторінка товару

Коли користувач натискає на картку товару, має відбуватися перехід на сторінку товару, де відображається більш детальна інформація про товар, така як опис, фотографії, характеристики і ціна. На цій сторінці мають бути доступні кнопки для додавання товару в кошик або вибране, а також можливість залишити відгук або поставити запитання про товар.

Навігація: меню з категоріями товарів, кнопка кошика і пошук.

- Зображення товару: велике зображення товару з можливістю перегляду додаткових фотографій при наведенні або натисканні.
- Інформація про товар: назва, ціна, опис, характеристики, наявність на складі та доступність для замовлення.
- Кнопки дій: кнопка "Додати в кошик", "Додати в обране", "Написати відгук" і "Поставити запитання".
- Відгуки: секція з відгуками клієнтів, включно з оцінкою, ім'ям клієнта і коментарем.
- Схожі товари: секція зі схожими товарами, які можуть бути цікаві клієнту.

Варто також пам'ятати про те, що сервер не завжди може встигнути швидко надіслати сторінку товару на комп'ютер клієнта, щоб запобігти зависанню браузера (що доволі часто трапляється в браузерах, заснованих на браузерній основі Chromium), варто створити легковажний завантажувальний макет, який відображає всі доступні опції, але в зафарбованому градієнтному вигляді.[1]

Вона має виконувати такі функції:

- Інформувати користувача про те, що відбувається завантаження даних, і дати йому зрозуміти, що він повинен почекати.
- Запобігати взаємодії користувача зі сторінкою доти, доки дані не завантажені.
- Надавати індикацію прогресу, щоб користувач міг бачити, наскільки близьке до завершення завантаження.
- Надавати можливість скасування запиту даних у разі довгого завантаження або помилки.

Важливо зазначити, що заглушка має бути якомога простішою і не захаращувати інтерфейс, оскільки це може викликати дискомфорт у користувача. Приклад сторінки, яка не встигла завантажитись представлено на Рис. 3.27

Рис. 3.27. Макет завантаження сторінки товару

Після завантаження сторінки, завантажувальний макет повинен зміниться елементами товару, такими як фото, опис та інше.

У разі довгого завантаження сторінки або невдалого запиту даних, має бути показана відповідна помилка. Це може містити в собі таке:

- Повідомлення про помилку: текстове повідомлення, яке пояснює причину помилки і дає інструкції для вирішення проблеми.
- Іконка помилки: візуальне представлення помилки, наприклад, іконка "мережева помилка"
- Кнопка для повторної спроби

Завантажена сторінка товару представлена на Рис.3.28

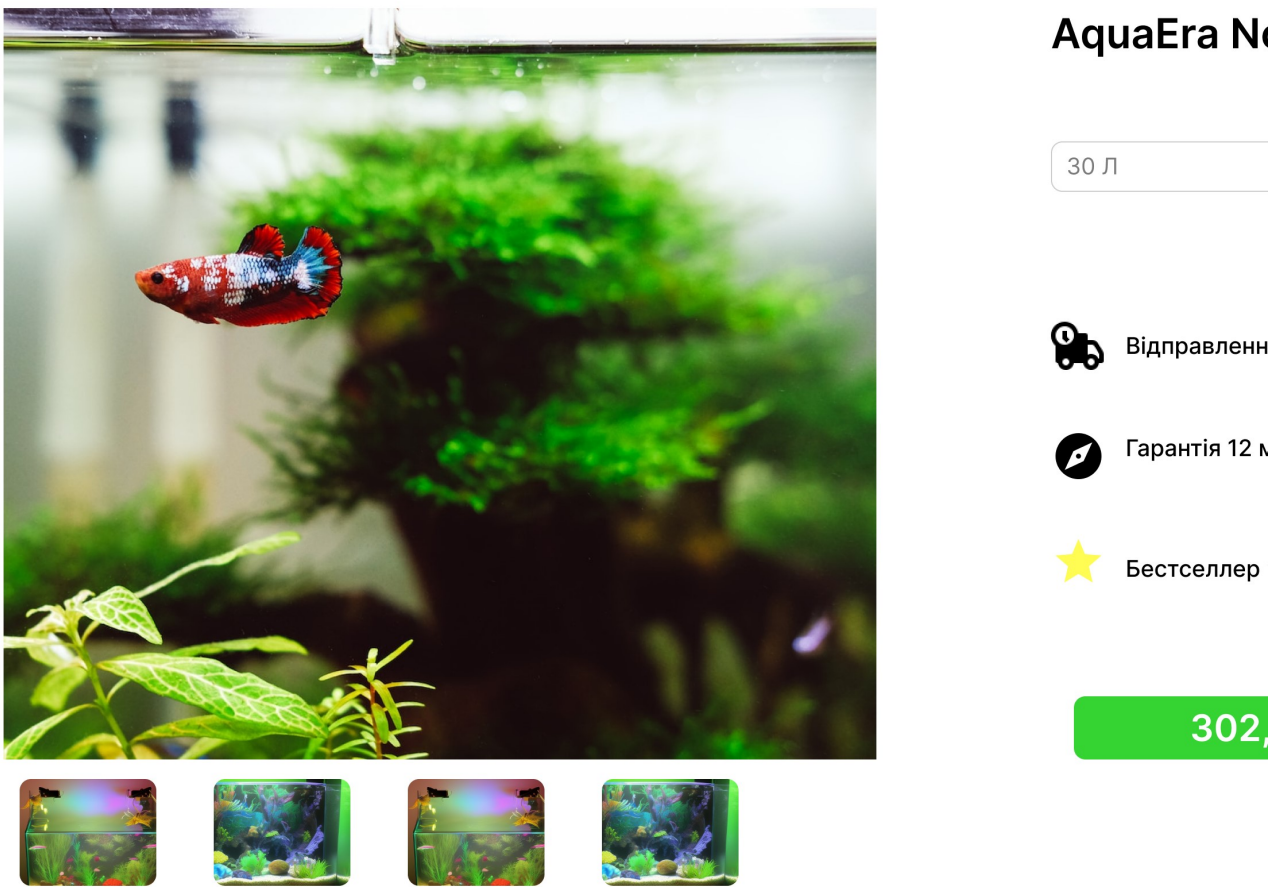


Рис.3.28. Завантажена сторінка товару

Оскільки користувача може відлякати висока ціна, або невпевненість, що цей товар йому може бути потрібен, необхідно "підштовхнути" його до вибору.

Найчастіше під час купівлі люди ставлять собі запитання "Чи правильний я роблю вибір?", ґрунтуючись на тому, що страх не виправдати свої очікування вищий, ніж страх витратити гроші та час. Для зменшення цього почуття існує концепт як показати людині, що цей товар їй знадобиться, - показати, що цей товар цікавий багатьом людям, даючи змогу покупцеві втішити себе думкою про те, що раз товар беруть багато людей, він однозначно є хорошим

Необхідно також забезпечити інтеграцію системи із зовнішніми системами управління онлайн магазинами, CRM. У разі використання модульної структури онлайн-сервісу, ми надаємо можливість інтеграції систем бізнес-аналізу, систем управління контентом і систем адміністрування товару з раніше створеними або зовнішніми системами сторонніх розробників, націлених на цей сегмент, без необхідності істотних змін з боку програмної частини.

Використання модульності є одним із концептів алгоритму "Розділяй і пануй", що дає змогу розподілити навантаження системи на зовнішні елементи та сервери, даючи можливість кожній системі виконувати свої завдання за мінімально-середнього навантаження на її компоненти.

Розділ 4. ОХОРОНА ПРАЦІ

Охорона праці є важливим аспектом у будь-якому професійному середовищі. Це не лише забезпечує комфорт та безпеку робітників, але також полегшує продуктивну та ефективну роботу. У цьому розділі ми подетально опишемо всі вимоги та рекомендації щодо охорони праці в офісних приміщеннях, які допоможуть вам підтримувати здоров'я та безпеку на весь час роботи.

Головні правила дотримування безпечної праці:

Обладнання:

- Проведення регулярних дезінфекцій поверхонь, дверей, меблів та відділень від хвороб присутніх у приміщенні.
- Підтримка чистоти відходів та відпочинкових зон.
- Забезпечення достатньої вентиляції та світла.
- Забезпечення достатньої кількості води та повітря.

Людські ресурси:

- Наявність достатньої кількості туалетів та умивальників.
- Забезпечення достатньої кількості рушників та мила.
- Забезпечення достатньої кількості настільних пристроїв, таких як кулери та кавомашини.

Безпека здоров’я працівників:

- Забезпечення працівникам захисної одежі та масок при необхідності.
- Перевірка здоров'я працівників перед початком роботи.

Робочі процедури:

- Регулярне миття рук та відпочинок від роботи в течії доби.
- Віддалене роботу в разі відсутності симптомів захворювань.
- Наявність альтернативного робочого місця в разі потреби.

Інше:

- Наявність плану дій в разі виникнення інфекційних захворювань.
- Наявність апарату для вимірювання температури.
- Регулярне підкорення інструкцій та правил здоров'я та безпеки.

Пожежна безпека на робочому місці передбачає заходи та процедури, спрямовані на запобігання виникненню пожежі та захист працівників у разі пожежі. Сюди входять регулярні протипожежні тренування, вогнестійкі матеріали, пожежна сигналізація, вогнегасники, аварійне освітлення та чіткі шляхи евакуації. Роботодавці також повинні навчати своїх працівників процедурам пожежної безпеки та правильному використанню протипожежного обладнання. Належна пожежна безпека на робочому місці має важливе значення для захисту здоров'я та безпеки працівників і мінімізації ризику пошкодження майна, пов'язаного з пожежею.

Правила пожежної безпеки на робочому місці, згідно з українським технічним регламентом (ДСТУ), включають наступне:

Провести оцінку пожежного ризику та розробити план пожежної безпеки. (ДСТУ Б.В.2.6-173:2010)

Встановити засоби протипожежного захисту та пожежогасіння відповідно до вимог нормативних документів. (ДСТУ Б.В.2.6-173:2010)

Забезпечити безпечне встановлення та обслуговування електричного обладнання та систем. (ДСТУ EN 60204-1:2010)

Зберігайте легкозаймисті та небезпечні матеріали безпечно та відповідно до нормативних вимог. (ДСТУ Б.В.2.6-173:2010)

Забезпечити навчання працівників правилам пожежної безпеки та регулярно проводити протипожежні тренування. (ДСТУ Б.В.2.6-173:2010)

Чітко позначте та утримуйте в належному стані пожежні виходи та шляхи евакуації (ДСТУ Б.В.2.6-173:2010)

Забезпечити належне технічне обслуговування та перевірку пожежної сигналізації та датчиків диму. (ДСТУ Б.В.2.6-173:2010)

Ці норми та правила розроблені з метою мінімізації ризику виникнення пожежі та забезпечення безпеки працівників у разі пожежі. Роботодавці несуть відповідальність за те, щоб їхні робочі місця відповідали правилам і нормам пожежної безпеки, а їхні працівники були навчені процедурам пожежної безпеки.

ВИСНОВКИ

Отже, інтернет-магазин акваріумів - це платформа електронної комерції, яка дозволяє клієнтам переглядати і купувати товари, пов'язані з акваріумістикою, у зручний і простий спосіб. Система призначена для демонстрації широкого асортименту товарів, надання детальної інформації про кожен товар, а також для того, щоб клієнти могли легко здійснювати покупки і відстежувати свої замовлення. Крім того, система може включати в себе такі функції, як відгуки клієнтів, порівняння товарів і персоналізовані рекомендації для покращення досвіду покупок. Створюючи інтернет-магазин, власник може охопити ширшу аудиторію, збільшити продажі та зменшити накладні витрати. Персоналізований досвід також допомагає утримати клієнтів і підвищити ймовірність повторних покупок. Загалом, інтернет-магазин акваріумів може забезпечити багато переваг як для покупців, так і для власника, що робить його цінним доповненням до індустрії роздрібної торгівлі.

Інтернет-магазин акваріумів також може надати власнику можливість розширити асортимент продукції, оскільки він має можливість охопити більшу клієнтську базу, ніж фізичний магазин. Це може призвести до збільшення доходів і прибутковості для власника. Крім того, інтернет-магазин може надавати цінні дані про клієнтів та аналітику, які можна використовувати для покращення купівельного досвіду та цілеспрямованих маркетингових зусиль.

Однією з головних переваг інтернет-магазину акваріумів є можливість надати покупцям широкий асортимент товарів, включаючи рідкісні або важкодоступні товари, які можуть бути відсутніми в фізичних магазинах. Це може привабити клієнтів, які шукають конкретні товари, і збільшити клієнтську базу магазину.

Крім того, інтернет-магазин може також забезпечити платформу для залучення клієнтів, наприклад, через соціальні мережі, відгуки клієнтів і публікації в блогах. Це може допомогти побудувати почуття спільноти і довіри серед клієнтів, підвищити їхню лояльність і збільшити кількість повторних покупок.

Таким чином, інтернет-магазин акваріумів може забезпечити багато переваг як для покупців, так і для власника. Він пропонує зручність і широкий асортимент товарів, покращене обслуговування клієнтів, а також цінні дані та аналітику. Крім того, він може надати власнику можливість розширити асортимент продукції і збільшити дохід. Загалом, інтернет-магазин акваріумів може стати цінним доповненням до роздрібної торгівлі та чудовим способом для клієнтів купувати товари, пов'язані з акваріумістикою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. БЕКВІТ ГАРРІ Продаючи невидиме. Керівництво з сучасного маркетингу послуг [Книга]. - Київ : Книжковий клуб "Клуб Сімейного Дозвілля", 2018.
2. Васильєв Олексій Алгоритми [Книга]. - Київ : Ліра-К, 2022.
3. Васильєв Олексій Програмування мовою Java [Книга]. - [б.м.] : Навчальна книга - Богдан, 2020.
4. Гайдаржи Володимир и Ізварін Ігор Бази даних в інформаційних системах [Книга]. - [б.м.] : Університет "Україна", 2018.
5. Дубілет Дмитро БІЗНЕС НА ЗДОРОВОМУ ГЛУЗДІ. 50 ІДЕЙ, ЯК ДОМОГТИСЯ СВОГО [Книга]. - Київ : BookChef, 2020.
6. Каплунов Денис КОРОЛІ СОЦІАЛЬНИХ МЕРЕЖ [Книга]. - Київ : BookChef, 2022.
7. Порселло Е. Бенкс А React: сучасні шаблони для розробки додатків [Книга]. - 2022. - Т. 2.
8. Управління ІТ проектами: лабораторний практикум до виконання лабораторних робіт для студентів напряму підготовки 6.050101 «Комп`ютерні науки» денної та заочної форм навчання. уклад. Хлобистова О.А., Гладка М.В – К.: НУХТ, 2013. – 108 с. – [Електронний ресурс] – Режим доступу: http://library.nuft.edu.ua/
9. Проектування інформаційних систем. [Електронний ресурс]: лабораторний практикум для студ. освітнього ступеню „бакалавр” спец. 122 “Комп`ютерні науки ” денної і заочної форм навчання.Частина 2 „Проектування клієнтського додатку” Уклад.: О.М. М’якшило, О.В. Харкянен – К.: НУХТ, 2017 – 33с.

ДОДАТКИ

Додаток А

Скрипт створення схеми aquafront

```
CREATE TABLE Connection_Properties
(
        Prop_ID             integer  NOT NULL ,
        Prop_Code           varchar(20)  NULL ,
        Prop_Name           varchar(20)  NULL ,
        Prop_Value          varchar(20)  NULL ,
        Connection_Type_ID   integer  NULL
) go
ALTER TABLE Connection_Properties
        ADD  CONSTRAINT  XPKConnection_Properties  PRIMARY  KEY    CLUSTERED
(Prop_ID ASC)
go
CREATE TABLE Connection_Types
(
        Type_ID             integer  NOT NULL ,
        Type_Name           varchar(20)  NULL
)
go
ALTER TABLE Connection_Types
        ADD CONSTRAINT XPKConnection_Types PRIMARY KEY  CLUSTERED (Type_ID
ASC)
go
CREATE TABLE Languages
(
        LANG_ID             integer  NOT NULL ,
        Language            varchar(20)  NULL
)
go
ALTER TABLE Languages
        ADD CONSTRAINT XPKLanguages PRIMARY KEY  CLUSTERED (LANG_ID ASC)
go
CREATE TABLE Messages
```

```
(
        Message_ID          integer  NOT NULL ,
        CODE                varchar(20)  NULL ,
        LANG_ID             integer  NULL ,
        Message             varchar(20)  NULL
) go
ALTER TABLE Messages
        ADD CONSTRAINT XPKMessages PRIMARY KEY  CLUSTERED (Message_ID ASC)
go
CREATE TABLE Style_Properties
(
        ID                  integer  NOT NULL ,
        Code                varchar(20)  NULL ,
        Value               varchar(20)  NULL
)
go
ALTER TABLE Style_Properties
        ADD CONSTRAINT XPKStyle_Properties PRIMARY KEY  CLUSTERED (ID ASC)
Go
```

Схема aquaftont

Додаток Б

Скрипт створення схеми aquaback

```
CREATE TABLE Properties
(
        ID                  integer  NOT NULL ,
        Name                varchar(20)  NULL ,
        Value               varchar(20)  NULL ,
        CODE                varchar(20)  NULL
)
go
ALTER TABLE Properties
        ADD CONSTRAINT XPKProperties PRIMARY KEY  CLUSTERED (ID ASC)
Go
```

Схема aquaback

Додаток В.

Скрипт створення схеми personal_data

```
CREATE TABLE Addresses
(
	Address_ID	integer  NOT NULL ,
	Country_ID	integer  NULL ,
	City		varchar(20)  NULL ,
	Post_Code	varchar(20)  NULL ,
	Street		varchar(20)  NULL ,
	Home		varchar(20)  NULL ,
	Room		varchar(20)  NULL
)
go
ALTER TABLE Addresses
	ADD CONSTRAINT XPKAddresses PRIMARY KEY  CLUSTERED (Address_ID
ASC)
go
CREATE TABLE Clients
(
	Client_Pesonal_ID	integer  NOT NULL ,
	Client_Credentials_ID integer  NULL ,
	Client_Saved_Address_ID integer  NULL ,
	Client_Contact_ID	integer  NULL ,
	db_create_time	datetime  NULL ,
	db_modify_time	datetime  NULL ,
	is_Locked	varbinary  NULL
)
go
ALTER TABLE Clients
	ADD CONSTRAINT XPKClients PRIMARY KEY CLUSTERED
(Client_Pesonal_ID ASC)
go
CREATE TABLE Clients_Orders
(
```

```
	Order_ID	integer  NOT NULL ,
	Client_ID	integer  NULL ,
	Product_ID	integer  NULL ,
	Cost	money  NOT NULL ,
	db_create_time	datetime  NULL ,
	db_modify_time	datetime  NULL ,
	Pay_ID	integer  NULL ,
	Delivery_Option_ID	integer  NULL ,
	is_realised	varbinary  NULL
)
go
ALTER TABLE Clients_Orders
	ADD CONSTRAINT XPKClients_Orders PRIMARY KEY  CLUSTERED (Order_ID
ASC)
go
CREATE TABLE Contact_Information
(
	Contact_ID	integer  NOT NULL ,
	Phone	varchar(20)  NULL ,
	Mail	varchar(20)  NULL ,
	Phone_2	varchar(20)  NULL ,
	Fax	varchar(20)  NULL ,
	Social_Links	varchar(20)  NULL ,
	Contact_Person_Name	varchar(20)  NULL
)
go
ALTER TABLE Contact_Information
	ADD CONSTRAINT XPKContact_Information PRIMARY KEY  CLUSTERED
(Contact_ID ASC)
go
CREATE TABLE Countries
(
	Country_ID	integer  NOT NULL ,
	Country_Name	varchar(20)  NULL ,
	Country_Code	varchar(20)  NULL ,
	is_Forbidden	varbinary  NULL
```

```
)
go
ALTER TABLE Countries
	ADD CONSTRAINT XPKCountries PRIMARY KEY CLUSTERED (Country_ID
ASC)
go
CREATE TABLE Credentials
(
	Credential_ID	integer NOT NULL ,
	Login		varchar(20) NULL ,
	pass		varchar(20) NULL ,
	Last_Success_Enter	datetime NULL ,
	Logit_attempt_counter integer NULL ,
	Last_change_password_time datetime NULL ,
	Last_changed_password varchar(20) NULL
)
go
ALTER TABLE Credentials
	ADD CONSTRAINT XPKCredentials PRIMARY KEY CLUSTERED (Credential_ID
ASC)
go
CREATE TABLE Deliver_Options
(
	Delivery_Option_ID	char(18) NOT NULL ,
	Firm_ID		integer NULL ,
	Delivery_Type_ID	integer NULL ,
	Delivery_Cost	money NOT NULL
)
go
ALTER TABLE Deliver_Options
	ADD CONSTRAINT XPKDeliver_Options PRIMARY KEY CLUSTERED
(Delivery_Option_ID ASC)
go
CREATE TABLE Delivery_Types
(
	Type_ID		integer NOT NULL ,
```

```
	Type_CODE	varchar(20)  NULL ,
	Type_Def	varchar(20)  NULL
)
go
ALTER TABLE Delivery_Types
	ADD CONSTRAINT XPKDelivery_Types PRIMARY KEY  CLUSTERED (Type_ID
ASC)
go
CREATE TABLE Document_Types
(
	Type_ID	integer  NOT NULL ,
	Type_Name	varchar(20)  NULL
)
go
ALTER TABLE Document_Types
	ADD CONSTRAINT XPKDocument_Types PRIMARY KEY CLUSTERED
(Type_ID ASC)
go
CREATE TABLE Firm_Vacations
(
	Vacation_ID	integer  NOT NULL ,
	Vacation_From	datetime  NULL ,
	Vacation_To	datetime  NULL ,
	Firm_ID	integer  NULL
)
go
ALTER TABLE Firm_Vacations
	ADD CONSTRAINT XPKFirm_Vacations PRIMARY KEY CLUSTERED
(Vacation_ID ASC)
go
CREATE TABLE Firms
(
	Firm_ID	integer  NOT NULL ,
	Firm_Name	varchar(20)  NULL ,
	Firm_Type	char(18)  NULL ,
	is_global	varbinary  NULL ,
```

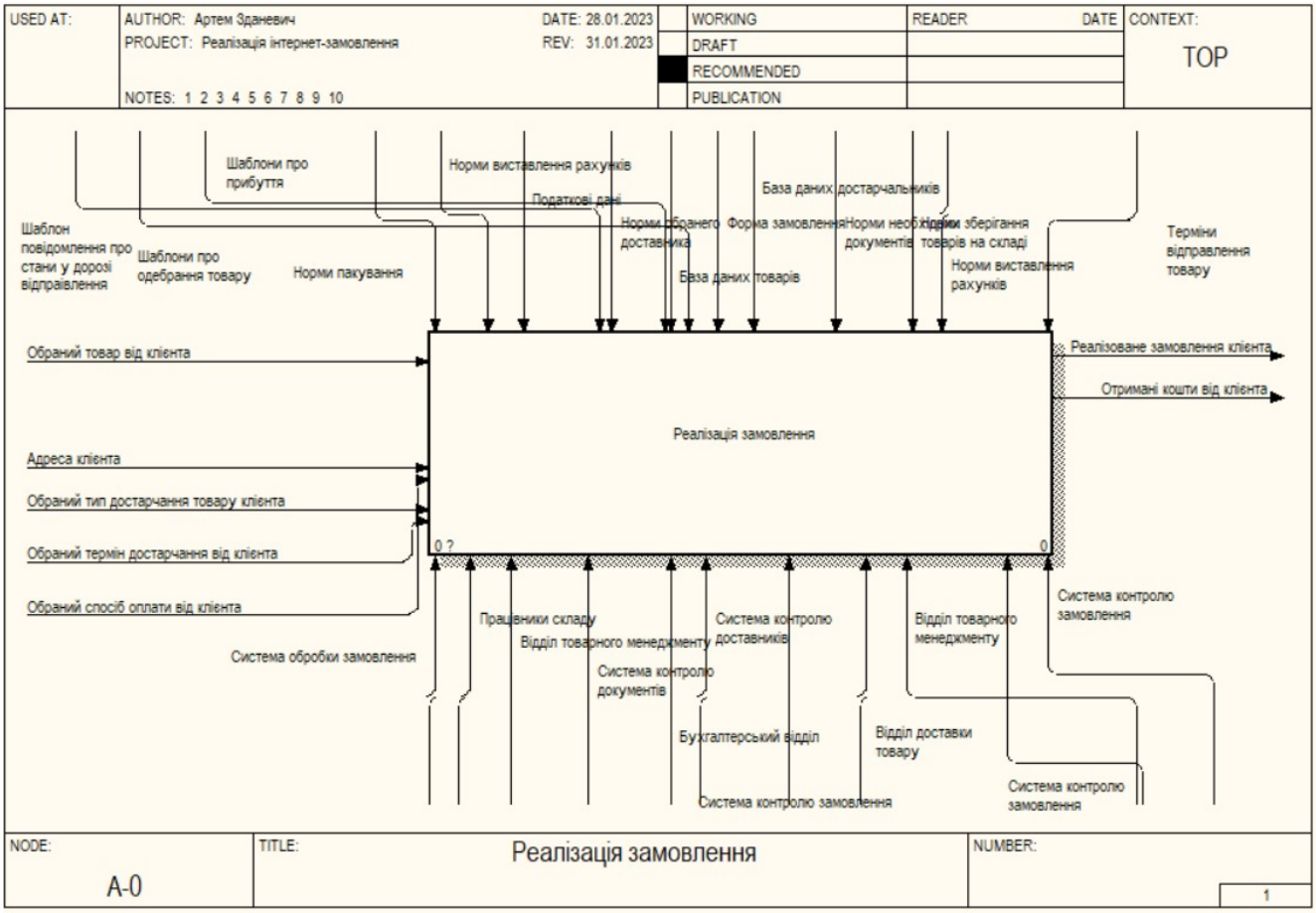
```
	Contact_ID		char(18)  NULL ,
	Address_ID		char(18)  NULL ,
	Official_ID		char(18)  NULL ,
	db_create_time		datetime  NULL ,
	db_modify_time		datetime  NULL
)
go
ALTER TABLE Firms
	ADD CONSTRAINT XPKFirms PRIMARY KEY  CLUSTERED (Firm_ID ASC)
go
CREATE TABLE Firms_Opening_Hours
(
	Open_Day_Hour_ID	integer  NOT NULL ,
	Hour_From		datetime  NULL ,
	Hour_To			datetime  NULL ,
	Firm_ID			integer  NULL ,
	Week_Day		varchar(20)  NULL
)
go
ALTER TABLE Firms_Opening_Hours
	ADD  CONSTRAINT  XPKFirms_Opening_Hours  PRIMARY  KEY   CLUSTERED
(Open_Day_Hour_ID ASC)
go
CREATE TABLE Official_Information
(
	Official_ID		integer  NOT NULL ,
	Document_Type_ID	integer  NULL ,
	Document_Number		varchar(20)  NULL ,
	Tax_Number		varchar(20)  NULL ,
	is_Legal		varbinary  NULL ,
	db_create_time		datetime  NULL ,
	db_modify_time		char(18)  NULL
)
go
ALTER TABLE Official_Information
```

```
	ADD CONSTRAINT XPKOfficial_Information PRIMARY KEY CLUSTERED
(Official_ID ASC)
go
CREATE TABLE Pay_Types
(
	Pay_Type_ID	integer NOT NULL ,
	Name		varchar(20) NULL ,
	Definition	varchar(20) NULL
)
go
ALTER TABLE Pay_Types
	ADD CONSTRAINT XPKPay_Types PRIMARY KEY CLUSTERED (Pay_Type_ID
ASC)
go
CREATE TABLE Pays
(
	Pay_ID		integer NOT NULL ,
	Transaction_Number	integer NULL ,
	Pay_Type_ID	integer NULL ,
	db_create_time	datetime NULL ,
	db_modify_time	datetime NULL ,
	confirm_transaction_time datetime NULL
)
go
ALTER TABLE Pays
	ADD CONSTRAINT XPKPays PRIMARY KEY CLUSTERED (Pay_ID ASC)
go
CREATE TABLE Product_Infos
(
	Info_ID		integer NOT NULL ,
	Title		varchar(20) NULL ,
	Description	varchar(20) NULL ,
	Varialble_Scope	integer NULL ,
	Price		integer NULL ,
	Is_available	varbinary NULL ,
	db_create_time	datetime NULL ,
```

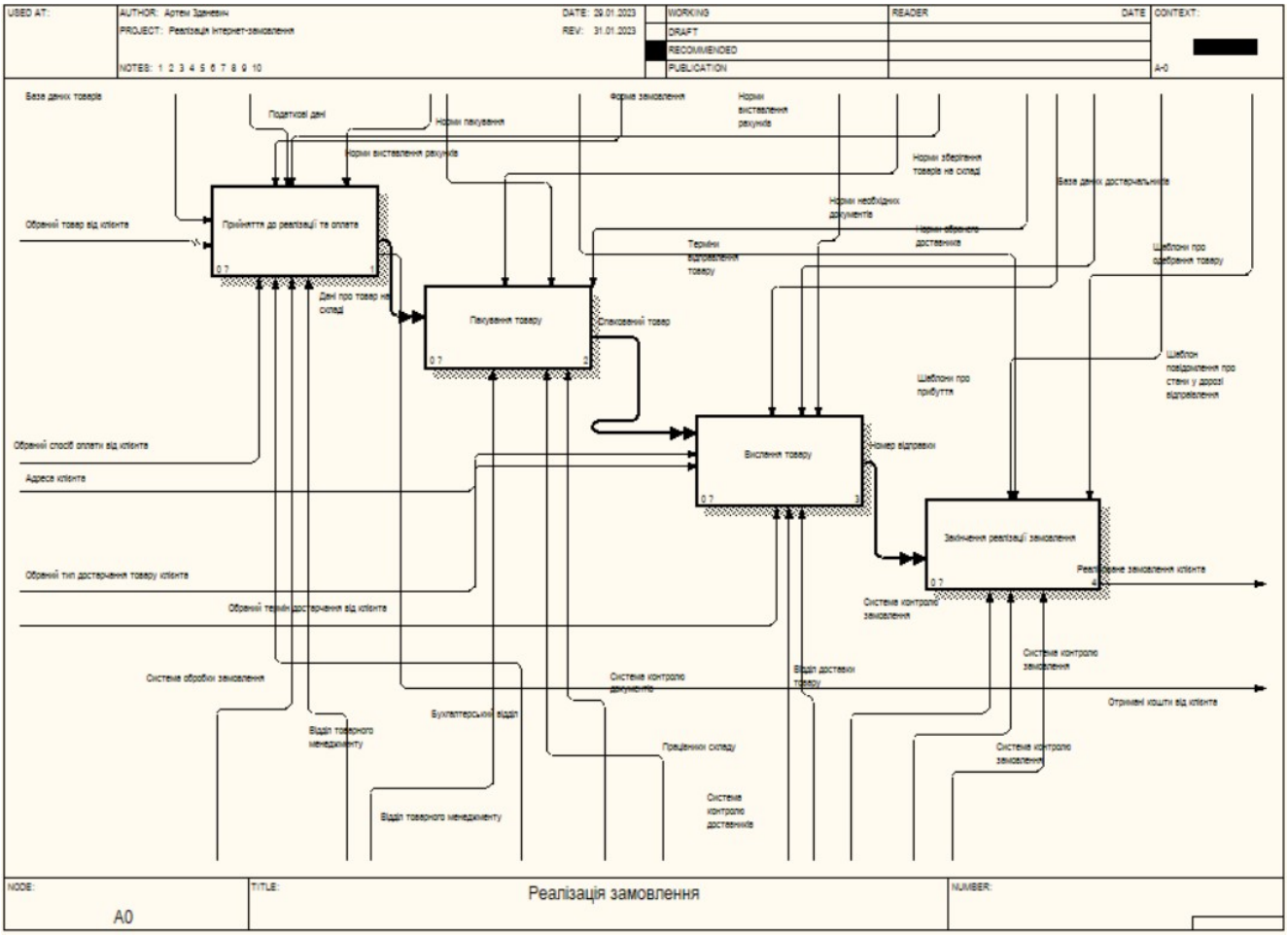
```
        db_modify_time      datetime  NULL
)
go
ALTER TABLE Product_Infos
        ADD CONSTRAINT XPKProduct_Infos PRIMARY KEY  CLUSTERED (Info_ID
ASC)
go
CREATE TABLE Product_Variables
(
        Variable_ID         integer  NOT NULL ,
        Product_ID          integer  NULL ,
        Product_Info_ID     char(18)  NULL ,
        db_create_time      char(18)  NULL ,
        db_modify_time      datetime  NULL
)
go
ALTER TABLE Product_Variables
        ADD CONSTRAINT XPKProduct_Variables PRIMARY KEY  CLUSTERED
(Variable_ID ASC)
go
CREATE TABLE Products
(
        Product_ID          integer  NOT NULL ,
        Product_Info_ID     integer  NULL ,
        is_available        varbinary  NULL ,
        db_create_time      datetime  NULL ,
        db_modify_time      datetime  NULL
)
go
ALTER TABLE Products
        ADD CONSTRAINT XPKProducts PRIMARY KEY  CLUSTERED (Product_ID
ASC)
Go
```

Схема personal_data

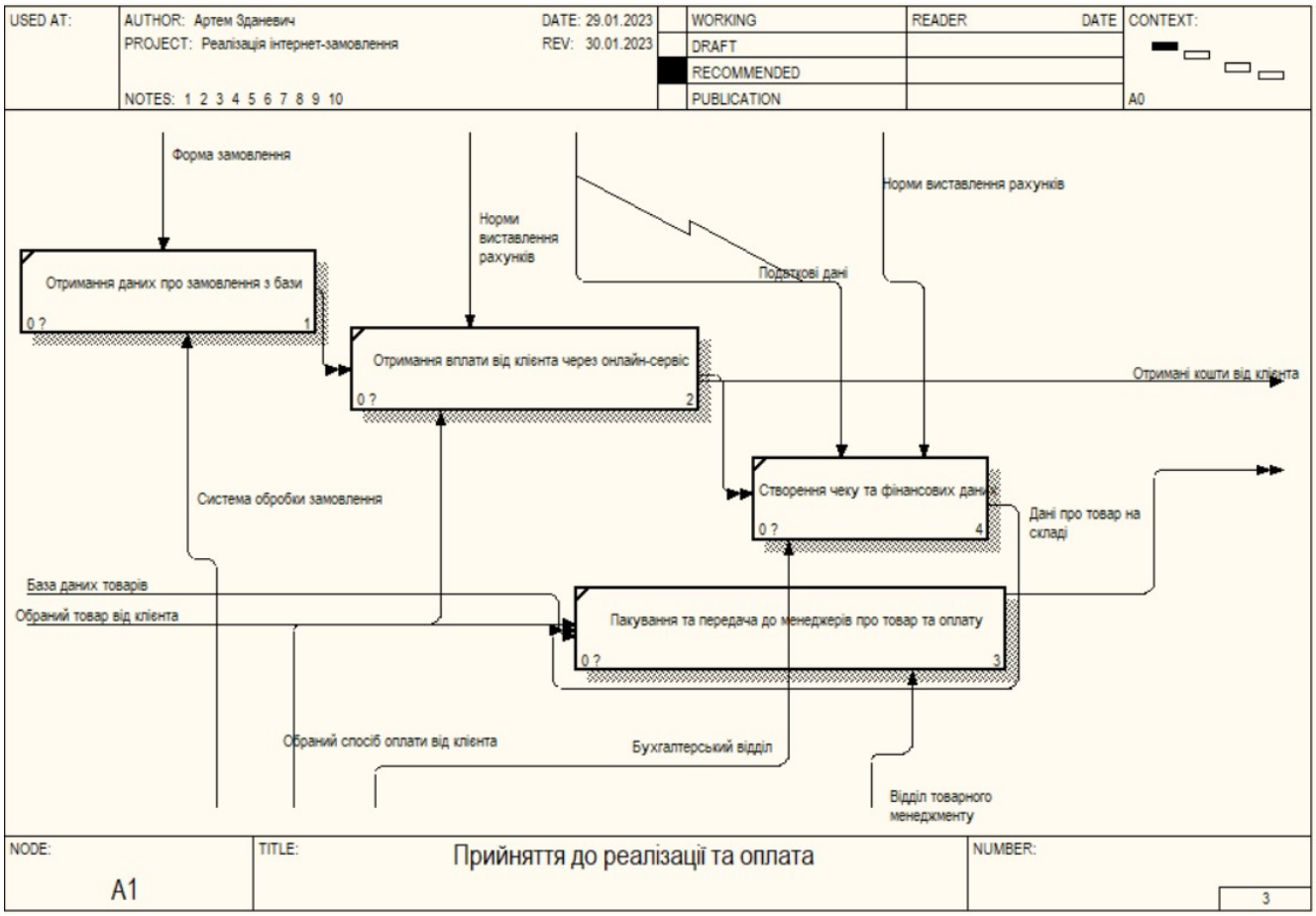
Додаток Г



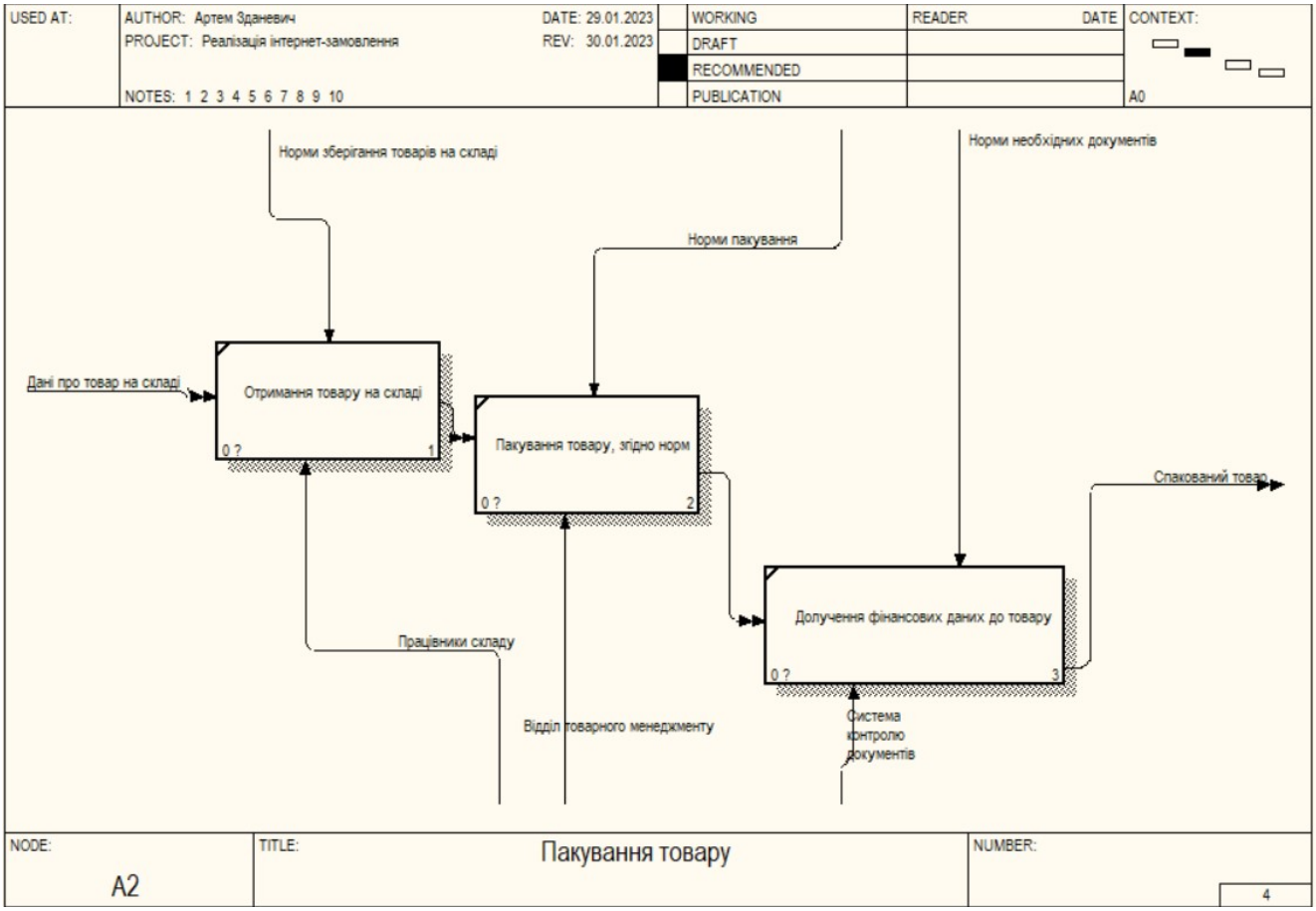
Діаграма А0


USED AT:
AUTHOR: Артем Заневич
PROJECT: Реалізація інтернет-замовлення
DATE: 29.01.2023
REV: 31.01.2023
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A-0
База даних товарів
Податкові дані
Норми пакування
Норми виставлення рахунків
Форма замовлення
Норми виставлення рахунків
Норми зберігання товарів на складі
Норми необхідних документів
Норми обмеження доставників
База даних доставників
Обраний товар від клієнта
Прийняття до реалізації та оплата
0.7
1
Терміни відповідання товару
Шаблони про одержання товару
Дані про товар на складі
Пакування товару
0.7
2
Опакований товар
Шаблони про прибуття
Шаблон повідомлення про стани у дорозі відправлення
Обраний спосіб оплати від клієнта
Виставлення товару
0.7
3
Номер відправки
Адреса клієнта
Завершення реалізації замовлення
0.7
4
Обраний тип достарчання товару клієнта
Реалізоване замовлення клієнта
Обраний термін достарчання від клієнта
Система контролю замовлення
Система обробки замовлення
Система контролю документів
Відділ доставки товару
Система контролю замовлення
Отримані кошти від клієнта
Бухгалтерський відділ
Відділ товарного менеджменту
Працівник складу
Система контролю замовлення
Система контролю доставників
Відділ товарного менеджменту
NODE:
A0
TITLE:
Реалізація замовлення
NUMBER:

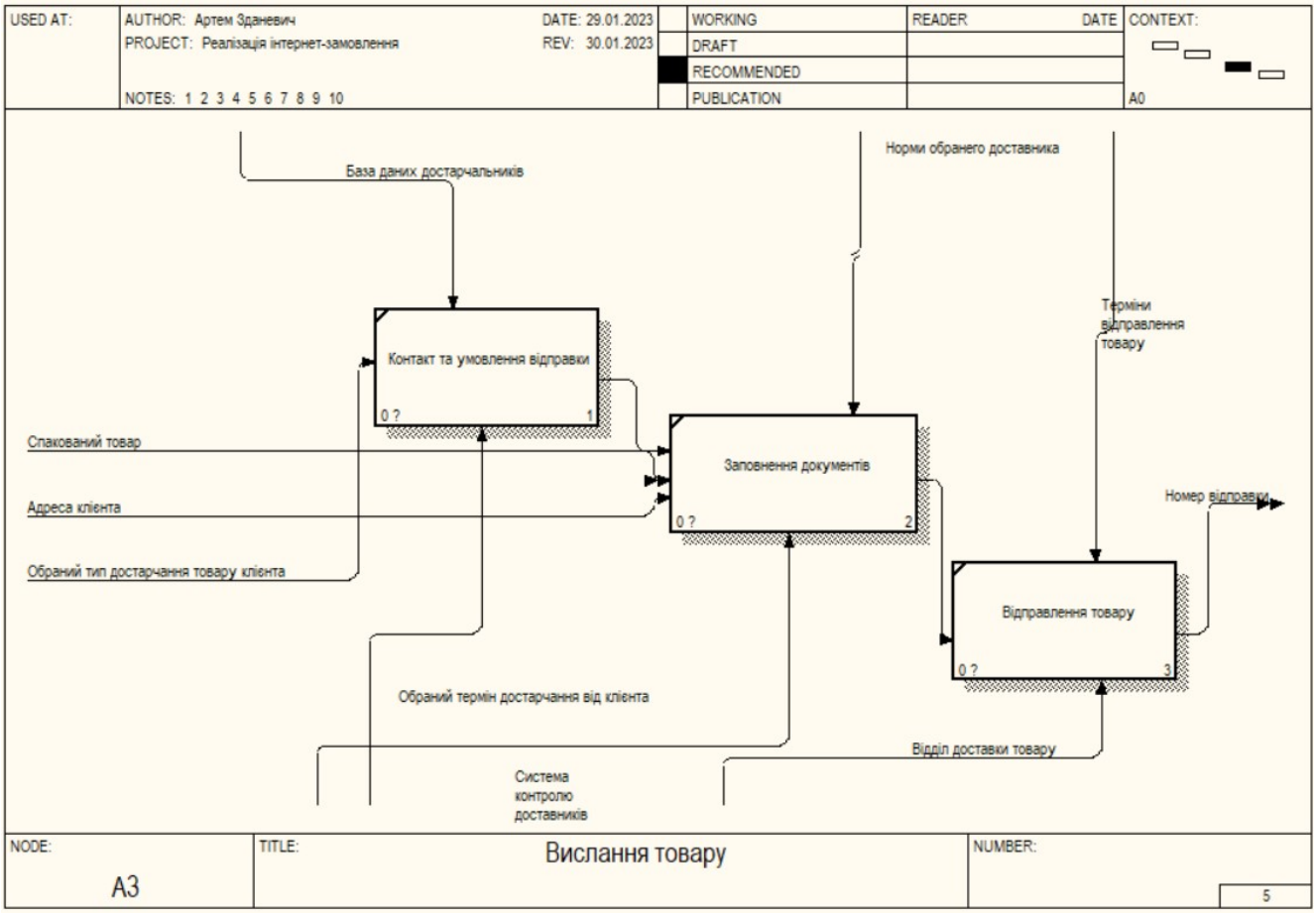
Діаграма А1



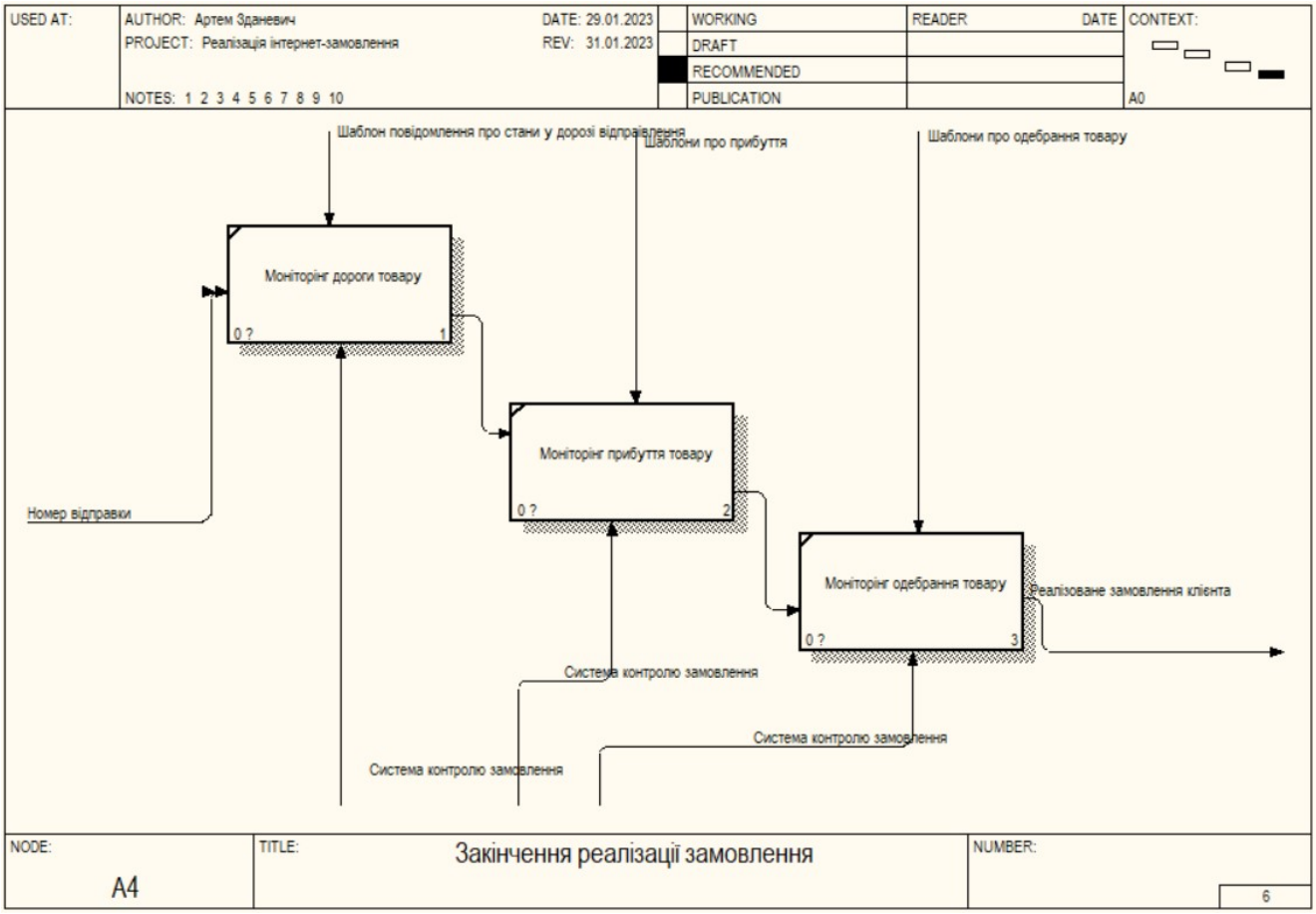
Діаграма декомпозиції 2-го рівня «Прийняття до реалізації та оплата»



Діаграма декомпозиції 3-го рівня «Пакування товару»



Діаграма декомпозиції 4-го рівня «Відправлення товару»



Діаграма декомпозиції 5-го рівня «Закінчення реалізації замовлення»