ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка інформаційної системи підтримки діяльності комерційного підприємства»

(тема кваліфікаційної роботи українською мовою)

«Development of an information system to support the commercial enterprise activities»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Персидський Микита Вадимович

(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Вохменцева Т.Б. _____

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Щербіна Ю.В.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри

_____ КАЗАКОВА Надія .

(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 22 від 21 червня 2024 р.

Оцінка добре / С / 76 .

(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ КОПИЧЕНКО Іван .

(підпис) (прізвище,ім'я)

Одеса 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

CSS – Cascading Style Sheets

CSRF – Cross-Site Request Forgery

HA – High Availability

HTML – Hypertext Markup Language

RPO – Recovery Point Objective

Virtual DOM – Document Object Model

XSS – Cross-Site Scripting

Activity diagram – діаграм діяльності

ASP.NET – веб-фреймворк від Microsoft

API Gateway – API шлюз

Auth Service – сервіс авторизації

Databases – бази даних

Frontend- rowser – клієнтський рівень

MySQL – СУБД

Usability – Юзабіліті

Use Cases – варіант використання

Order Service – сервіс замовлень

Performance – продуктивність

PostgreSQL – СУБД

Product Service – сервіс товарів

React – це бібліотека JavaScript

Reliability – надійність

Security – безпека

Sequence diagram – діаграм послідовності

SFA – однофакторна аутентифікація

SQL Server – СУБД

ВСТУП

Швидкий прогрес Інтернет-технологій відкриває користувачам нові можливості для бізнесу, сприяючи розвитку віртуальних ділових зв'язків на різних рівнях. Завдяки доступності, глобальності і стандартизації, Інтернет стає найзручнішим середовищем для комунікації, розвитку бізнесу та залучення нових клієнтів. Сьогодні електронна торгівля трансформує торгові відносини між підприємствами, вносячи нові аспекти.

Електронна торгівля змінює бізнес-процеси та формує нові моделі, що підвищують ефективність бізнесу. Отже, розробка інформаційних систем для підтримки електронної комерції стає все більш актуальною та практично значущою в управлінні підприємствами, які займаються електронною торгівлею.

Темою дипломної кваліфікаційної роботи є «Розробка інформаційної системи підтримки діяльності комерційного підприємства». Електронна комерція дозволяє компаніям здійснювати внутрішні операції більш ефективно та гнучко, забезпечуючи щільнішу взаємодію з постачальниками та швидшу реакцію на запити і очікування клієнтів.

Актуальність розробки такої ІС полягає в:

- підвищенні ефективності бізнес-процесів через автоматизацію;
- конкурентоспроможності завдяки швидкому реагуванню на ринкові зміни;
- покращенні обслуговування клієнтів через персоналізацію та швидкий доступ до інформації;
- зниженні витрат шляхом оптимізації ресурсів;
- підвищенні безпеки даних і захисту інформації;
- обґрунтованості рішень завдяки аналітичним можливостям;
- інтеграції з іншими системами, створюючи єдине інформаційне середовище;
- підтримці мобільності, що важливо для віддаленої роботи.

Це робить інформаційні системи критично важливими для успішного функціонування сучасних комерційних підприємств.

Метою роботи є розробка інформаційної системи у вигляді веб-сайту для підтримки діяльності комерційного підприємства з продажу товарів, яка допоможе автоматизувати бізнес-процеси, покращити взаємодію з клієнтами та підвищити ефективність операцій, що в кінцевому підсумку сприятиме збільшенню доходів підприємства.

Об'єктом розробки є процеси комерційного підприємства з продажу товарів, які потребують автоматизації для підвищення ефективності роботи та покращення взаємодії з клієнтами.

Предметом розробки є інформаційна система у вигляді веб-сайту, яка забезпечує автоматизацію ключових бізнес-процесів підприємства.

1 ВИЗНАЧЕННЯ ВИМОГ ДО ОБ'ЄКТУ РОЗРОБКИ

1.1 Характеристика предметної області

Інформаційні системи є ключовим елементом сучасного управління комерційними підприємствами. Їх впровадження дозволяє досягти значних покращень в ефективності, конкурентоспроможності, якості обслуговування клієнтів та зниженні витрат, що робить їх незамінними у сучасних умовах.

Для успішної праці на сучасному ринку підприємству, діяльність якого є реалізація великого різноманіття цибулин тюльпанів з Голландії, необхідна розробка ІС, яка буде представляти весь асортимент товарів та дозволить користувачам оформлювати замовлення і доставку. Актуальність розробки інформаційної системи підтримки діяльності комерційного підприємства є надзвичайно високою з кількох причин:

1. Ефективність бізнес-процесів.

Інформаційні системи дозволяють автоматизувати рутинні завдання, що значно підвищує продуктивність працівників. Завдяки автоматизації можна швидше обробляти великі обсяги даних, знижувати ймовірність помилок та оптимізувати бізнес-процеси.

2. Конкурентоспроможність.

У сучасному ринковому середовищі підприємства, що використовують передові технології, мають значну перевагу над конкурентами. Інформаційна система допомагає швидко реагувати на зміни ринку, аналізувати конкурентне середовище та приймати обґрунтовані управлінські рішення.

3. Поліпшення якості обслуговування клієнтів.

Сучасні інформаційні системи дозволяють краще зрозуміти потреби клієнтів, забезпечувати персоналізований підхід та підвищувати рівень обслуговування. Це сприяє зростанню задоволеності клієнтів і, відповідно, підвищенню лояльності до компанії.

4. Зниження витрат.

Автоматизація процесів часто призводить до зниження операційних витрат. Це може включати зменшення витрат на папір, скорочення часу на виконання завдань та зниження потреби у додатковому персоналі.

5. Підвищення рівня безпеки.

Інформаційні системи можуть забезпечувати більш високий рівень безпеки даних. Сучасні технології дозволяють ефективніше захищати конфіденційну інформацію підприємства та його клієнтів від несанкціонованого доступу і кіберзагроз.

6. Прийняття обґрунтованих рішень.

Завдяки аналітичним можливостям інформаційних систем, менеджмент може отримувати актуальну та точну інформацію для прийняття стратегічних рішень. Системи можуть надавати детальні звіти, прогнози та аналіз даних, що є критично важливим для успішного управління підприємством.

7. Інтеграція з іншими системами.

Інформаційні системи підтримки діяльності комерційних підприємств можуть інтегруватися з іншими системами, такими як CRM, ERP, SCM, що створює єдине інформаційне середовище для ефективного управління всіма аспектами бізнесу.

8. Підтримка мобільності.

Сучасні інформаційні системи часто підтримують роботу через мобільні пристрої, що дозволяє керівникам і співробітникам мати доступ до необхідної інформації в будь-який час і в будь-якому місці. Це особливо важливо в умовах зростаючої потреби у віддаленій роботі.

1.2 Порівнянний аналіз існуючих аналогів

Для поставлення вимог до інформаційної системи слід розглянути декілька аналогів та провести аналітичний огляд щодо їх функціоналу, UX/UI design та змісту контенту.

Перший аналог – це один з найкращих онлайн сервісів доставки квітів «https://www.edenbrothers.com/ » (рис. 1).

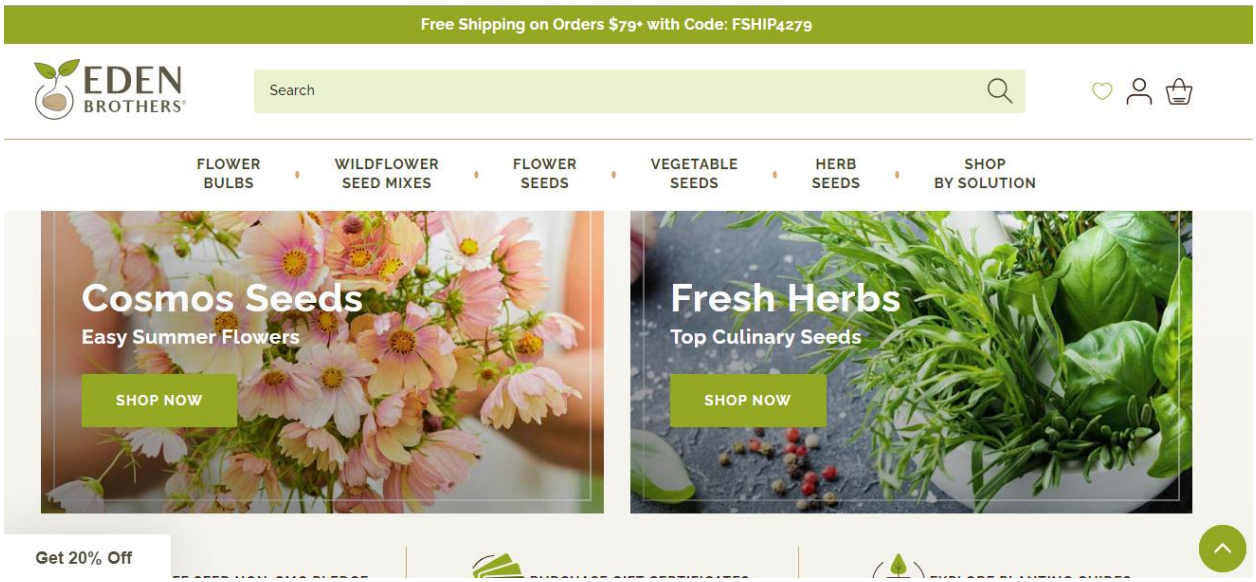

Рисунок 1 – Скріншот головної сторінки сайту «Eden brothers»

«Eden Brothers» може похвалитися понад 600 сортами цибулин. Щоб переконатися, що квіти процвітають у саду покупця, магазин має можливість допомогти визначити зону стійкості USDA. Потім можна сортувати доступні пропозиції за зонами, заощаджуючи час і гроші. Ця компанія продає цибулини лише напередодні вегетаційного періоду квітки, тому, наприклад, осінні посадки потрібно попередньо замовляти влітку. Щоб клієнти були в курсі, розклад доставки кожної квітки зручно вказано на сторінці окремого продукту.

Веб-сайт «Eden Brothers» простий у використанні та добре класифікований. Якщо клієнт не знає назв своїх улюблених рослин, є зображення, і навіть можете відфільтрувати за кольором, щоб знайти варіанти, які відповідають естетиці, наприклад типи апельсинових квітів.

На сайті є таблиця коротких фактів для кожного сорту цибулин, яка розповідає, як саме садити та доглядати за ними. Це чудовий ресурс для

садівників-початківців або садівників, які хочуть експериментувати з новими типами квітів, наприклад, такими як азалії.

Наступний аналог – «https://www.vanengelen.com/flower-bulbs-index/tulips.html» (рис. 2).

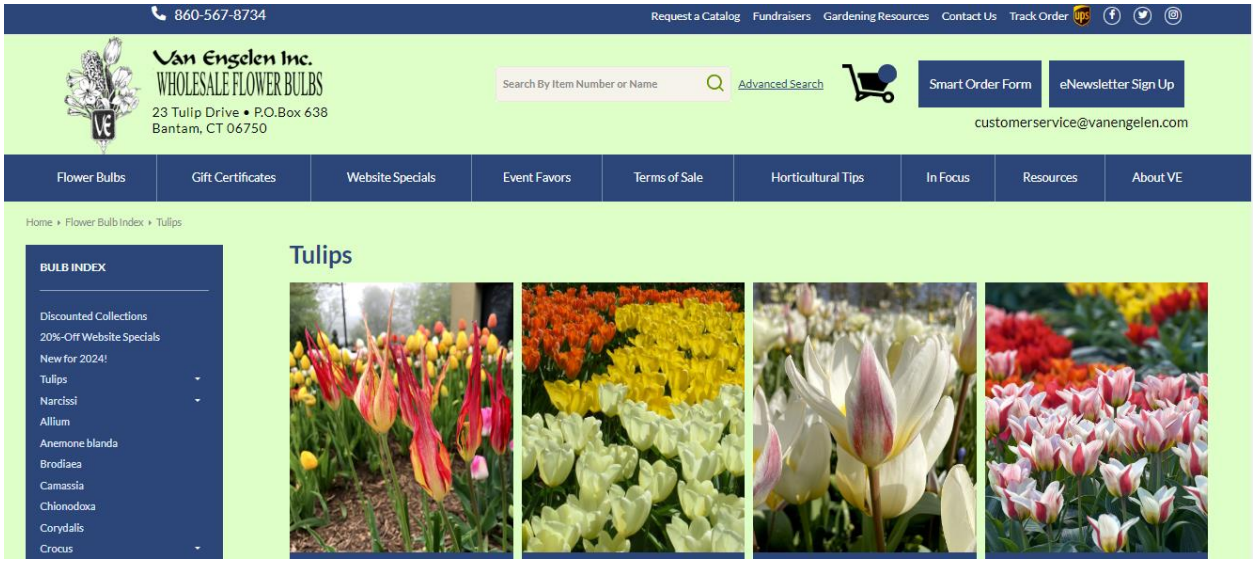

Рисунок 2 – Скріншот головної сторінки сайту «Vanengelen»

«Van Engelen» – один із найкращих онлайн-опцій для купівлі цибулин квітів оптом. Незважаючи на те, що тут відсутня можливість придбати окремі цибулини, ціна компанії зі знижкою на обсяг особливо корисна, якщо покупець хоче створити густий сад. Сервіс надає детальні описи продукції та посібники з посадки для кожного доступного виду квітів.

Якщо мова йде про доставку, якщо клієнт не вкаже певний тиждень, усі замовлення доставляються відповідно до обраної зони вирощування для своєчасного та правильного висаджування, про що поширюється на сторінці продукту кожної цибулини. Також є можливість відстежувати своє замовлення, даючи клієнтам спокій, якщо вони стурбовані своїми ніжними цибулинами. Є також кілька різних способів зв’язатися зі службою підтримки клієнтів, зокрема телефоном, онлайн-формою та електронною поштою.

Ще один онлайн-магазин «www.longfield-gardens.com» (рис. 3):

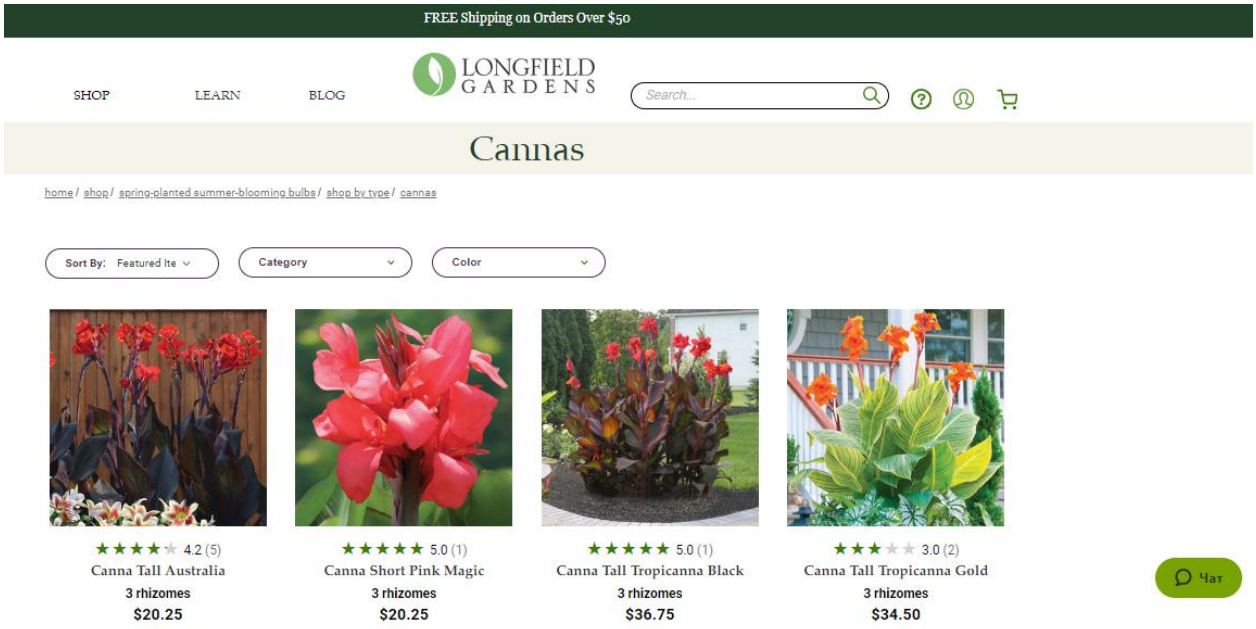

Рисунок 3 – Скріншот головної сторінки сайту «longfield-gardens»

Для садівників-початківців «Longfield Gardens» містить величезний розділ поширених запитань і блог, наповнений інструкціями та порадами щодо вирощування, які допоможуть переконатися, що кожна посадка буде успішною. На цьому ресурсі цілорічна сторінка «Спеціальні пропозиції» містить різноманітні цибулини за зниженими цінами, хоча пропозиції в основному спрямовані на клієнтів, які хочуть придбати від 100 до 250 одиниць за раз.

Графік доставки компанії відповідає вегетаційним сезонам для кожної зони стійкості, тож ви отримаєте цибулини одразу, коли вони будуть готові до посадки. Якщо вам потрібна допомога у визначенні зони витривалості вашої місцевості, просто введіть свій поштовий індекс у зручний інструмент карти служби. Зауважте, що ви можете претендувати на безкоштовну доставку, лише якщо витратите понад 50 доларів США.

1.3 Функціональні вимоги до інформаційної системи

Функціональні вимоги до інформаційної системи – це конкретні описання можливостей та поведінки системи, які необхідні для виконання її основних завдань. Для веб-сайту з продажу одягу ці вимоги включають всі функції, які забезпечують роботу сайту, його інтерактивність, взаємодію з користувачами та іншими системами. Функціональні вимоги можна розглядати як функції, які виявляє користувач. Вони відрізняються від нефункціональних вимог, які визначають, як система повинна працювати всередині (наприклад, продуктивність, безпека тощо).

Функціональні вимоги складаються з двох частин: функції та поведінки. Функція – це те, що виконує система (наприклад, «розрахувати податок з продажу»). Поведінка залежить від того, як система це робить (наприклад, «Система розраховує податок із продажу, помноживши ціну покупки на ставку податку»). Створюючи функціональні вимоги, важливо мати на увазі, що вони повинні бути конкретними, вимірними, досяжними, релевантними та обмеженими у часі (SMART) [1].

Варіанти використання (Use Cases) – це інструмент опису функціональних вимог до програмного продукту. Вони дозволяють описати, як користувачі взаємодіють з системою та як вона повинна реагувати на їх дії. Варіанти використання описують взаємодію між основним актором, рішенням і вторинними акторами, необхідними для досягнення мети основного актора.

Варіанти використання зазвичай запускаються первинним актором, але в деяких методах можуть також запускатися іншою системою, зовнішньою подією або таймером. Варіанти використання пишуться з точки зору актора і уникають опису внутрішньої роботи рішення.

Розглянемо діаграму Use-Case, де актором виступає Користувач (рис. 5). На діаграмі представлено 10 варіантів використання, за якими на завершальному етапі розробки ІС буде проведене тестування. Нижче наведено

опис до кожного варіанту використання, а саме перелік передумов та післяумов.

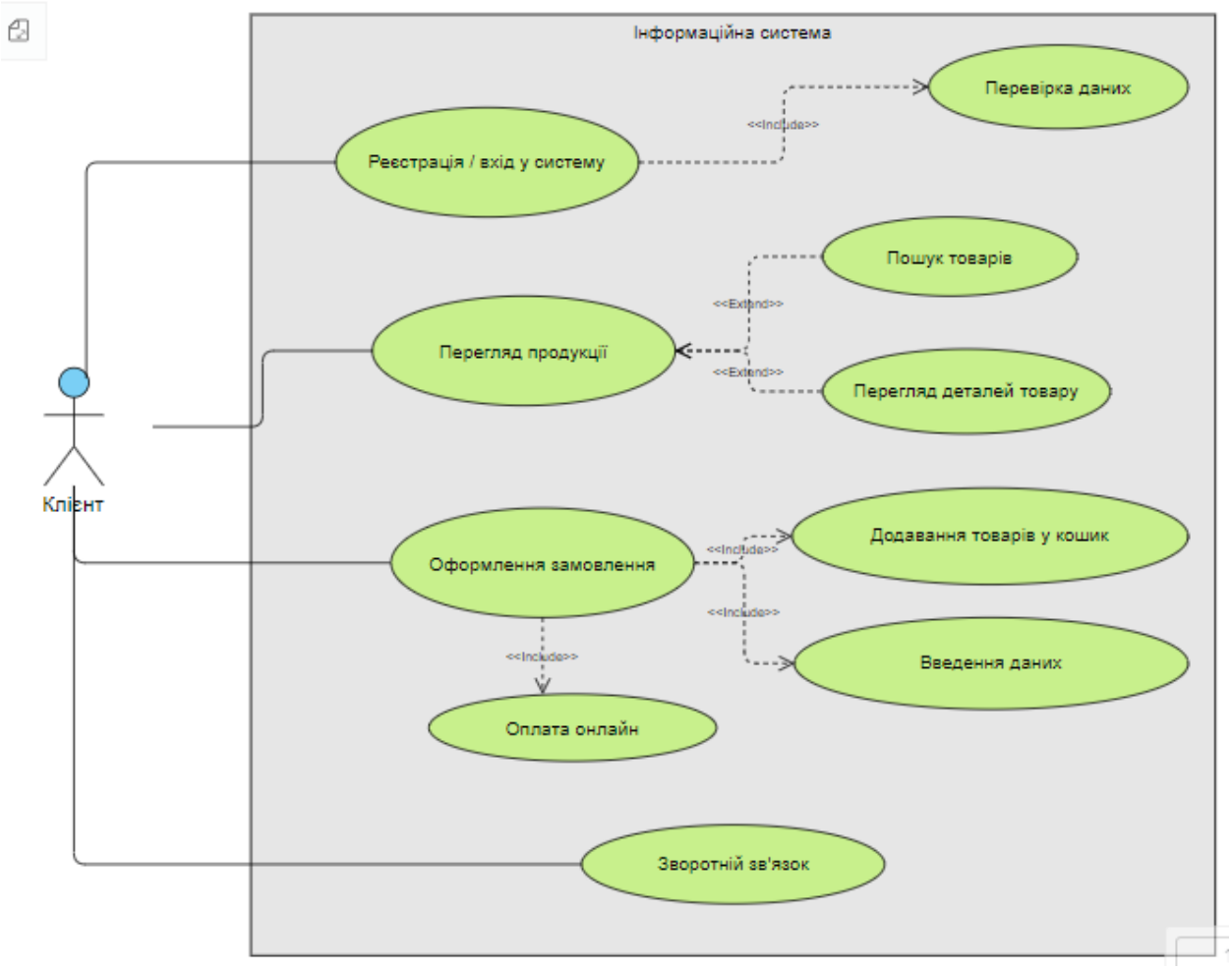

Рисунок 1.4 – Діаграма варіантів використання

Варіант використання №1 «Реєстрація/ вхід у систему».

Актор – Користувач і Система.

Передумови «Реєстрація»:

1. Доступ до веб-сайту: Користувач має доступ до інтернету і відвідав веб-сайт підприємства.
2. Відсутність активного облікового запису: Користувач не повинен мати вже існуючий обліковий запис у системі.

Післяумови «Реєстрація»:

1. Створення облікового запису: новий обліковий запис створено, і його дані збережені в базі даних.
2. Надання доступу: Користувач отримує доступ до функціоналу сайту, доступного тільки зареєстрованим користувачам.
3. Підтвердження реєстрації: Користувач отримує підтвердження реєстрації на свою електронну пошту (за необхідності).

Передумови «Вхід у систему»:

1. Наявність облікового запису Користувача.
2. Користувач має доступ до інтернету і відвідав веб-сайт підприємства.
3. Відомі облікові дані: Користувач знає свої логін і пароль для входу.

Післяумови «Вхід у систему»:

1. Аутентифікація користувача: Користувача успішно аутентифіковано, і йому надано доступ до особистого кабінету та інших функцій, доступних тільки зареєстрованим користувачам.
2. Початок сеансу: сеанс користувача активний, і він може користуватися усіма можливостями системи, призначеними для його ролі.
3. Запис активності: факт входу користувача зареєстровано в системі для цілей безпеки та аналітики.

Варіант використання №2 «Перегляд продукції».

Актори: Користувач, Система.

Передумови:

1. Доступ до веб-сайту: Користувач має доступ до інтернету і відвідав веб-сайт підприємства.
2. Доступність продукції: інформація про продукцію має бути попередньо додана до системи та доступна для перегляду.
3. Вхід у систему (не обов'язково): якщо доступ до повної інформації про продукцію обмежений лише для зареєстрованих користувачів

(оптові покупці), то користувач повинен бути зареєстрований та увійти в систему.

Післяумови:

1. Відображення інформації: Користувач може переглядати список доступної продукції та деталі обраних товарів.
2. Збереження історії переглядів: інформація про переглянуті товари може зберігатися в системі для подальшого аналізу або персоналізації.
3. Оновлення користувацького інтерфейсу: інтерфейс користувача оновлюється відповідно до його дій, відображаючи запитувану інформацію про продукцію.

Варіант використання №3 «Пошук товару».

Актори – Користувач, Система.

Передумови:

1. Доступ до веб-сайту: Користувач має доступ до інтернету і відвідав веб-сайт підприємства.
2. Функціональність пошуку: веб-сайт має функцію пошуку товарів, яка правильно налаштована та працює.
3. Наявність товарів у базі даних: товари, які можна знайти, мають бути додані до бази даних системи.

Післяумови:

1. Відображення результатів пошуку: Користувач бачить список товарів, які відповідають його пошуковому запиту.
2. Доступ до додаткової інформації: Користувач може обрати конкретний товар зі списку результатів для перегляду детальної інформації.
3. Запис пошукових запитів: Система може зберігати пошукові запити для подальшого аналізу.

Варіант використання №4 «Перегляд товару».

Актори – Користувач і Система.

Передумови:

1. Доступ до веб-сайту.
2. Наявність товарів у базі даних.
3. Вхід у систему (не обов'язково): доступ до повної інформації про товар обмежений для зареєстрованих користувачів, то користувач повинен бути зареєстрований та увійти в систему.

Післяумови:

1. Відображення інформації: Користувач бачить детальну інформацію про обраний товар, включаючи опис, зображення, ціну, наявність на складі тощо.
2. Можливість дій з товаром: Користувач може додати товар у кошик, додати у список бажань, поділитися інформацією про товар або повернутися до списку товарів.
3. Запис переглядів: система може зберігати інформацію про переглянуті товари для подальшої персоналізації та аналізу (якщо передбачено).

Варіант використання №5 «Оформлення замовлення».

Актори – Користувач і Система.

Передумови:

1. Доступ до веб-сайту.
2. Кошик з товарами: Користувач додав один або кілька товарів до кошика.
3. Реєстрація та вхід (якщо необхідно): оформлення замовлення доступне лише для зареєстрованих користувачів, отже користувач повинен бути зареєстрований та увійти в систему.
4. Доступність платіжних і сервісів доставки: веб-сайт має інтеграцію з платіжними системами і службами доставки.

Післяумови:

1. Замовлення створено: Система створює нове замовлення та зберігає його у базі даних.
2. Підтвердження замовлення: Користувач отримує підтвердження про успішне оформлення замовлення (на веб-сайті та електронною поштою).
3. Оплата проведена: платіжна система успішно обробляє оплату (якщо оплата здійснюється під час оформлення замовлення).
4. Інформація про доставку: Користувач отримує інформацію про терміни та умови доставки замовлення.
5. Оновлення статусу замовлення: Система оновлює статус замовлення і повідомляє користувача про подальші кроки (наприклад, обробка, доставка).

Варіант використання №7 «Зворотній зв'язок».

Актори – Користувач і Система.

Передумови:

1. Доступ до веб-сайту.
2. Функціональність зворотнього зв'язку: веб-сайт має форму або інший механізм для збору зворотнього зв'язку від користувачів.

Післяумови:

1. Повідомлення користувача успішно збережено в системі.
2. Користувач отримує підтвердження про успішне відправлення повідомлення (на веб-сайті або електронною поштою).
3. Повідомлення передано відповідальним працівникам підприємства для обробки.
4. Система зберігає інформацію для подальшого аналізу та вдосконалення обслуговування.

1.4 Нефункціональні вимоги до ІС

Нефункціональні вимоги для інформаційної системи підтримки діяльності комерційного підприємства у вигляді веб-сайту включають різні аспекти, що впливають на якість, продуктивність, надійність, безпеку та інші важливі характеристики системи. Нижче наведено основні з таких вимог:

1. Продуктивність (Performance): тут слід звернути увагу на час відгуку системи, а саме веб-сторінки повинні завантажуватися протягом 2 секунд для більшості користувачів. Система повинна обробляти щонайменше 100 запитів одночасно без значного зниження продуктивності. Система повинна підтримувати можливість горизонтального та вертикального масштабування для забезпечення стабільної роботи під час пікових навантажень.

2. Надійність (Reliability): система повинна автоматично відновлюватися після збоїв і мінімізувати втрату даних (Recovery Point Objective – RPO не більше 5 хвилин).

3. Безпека (Security): користувачі повинні пройти аутентифікацію для доступу до облікових записів та конфіденційної інформації. Бажано всі дані, передані між користувачем та сервером, повинні бути зашифровані з використанням SSL/TLS. Система повинна бути захищена від атак типу SQL ін'єкцій, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) та інших поширених веб-загроз.

4. Юзабіліті (Usability): інтуїтивний інтерфейс, який легко освоюється новими користувачами без необхідності в навчанні. Він повинен бути адаптивним і добре працювати на різних пристроях (настільні комп'ютери, планшети, смартфони) [2].

2 ПРОЕКТУВАННЯ ІС

2.1 Розробка плану робіт

Загальна структурна декомпозиція робіт проекту по розробці інформаційної системи підтримки діяльності комерційного підприємства включає наступні етапи:

1. Аналіз вимог:
   - вивчення функціональних та нефункціональних вимог до системи;
   - проведення спілкування з користувачами для з'ясування потреб та очікувань;
   - формулювання вимог до функціональності та характеристик системи.
2. Проектування: розробка архітектури інформаційної системи, включаючи визначення компонентів та їх взаємозв'язків та інтерфейсу користувача для зручного використання системи;
3. Розробка фронтенду і бекенду:
   - вибір технологій реалізації;
   - створення макетів інтерфейсу для розробки веб-сторінок та реалізація окремих компонентів інтерфейсу користувача, таких як кнопки, форми, меню;
   - проектування архітектури бекенду, включаючи структуру баз даних та логіку обробки запитів;
   - розробка модулів, що реалізують бізнес-логіку системи (управління замовленнями, обробка платежів, управління клієнтами тощо).
4. Розробка функціональних модулів:
   - каталог продукції: розробка модулів для управління товарами, категоріями, описами та цінами;
   - реалізація функцій додавання товарів до кошика, оформлення замовлень та їх обробки;

- управління клієнтами, а саме створення модулів для реєстрації, входу, управління профілями клієнтів.

5. Тестування:
   - виконання модульних та інтеграційних тестів для перевірки коректності роботи окремих модулів та їх взаємодії;
   - виконання системних тестів для перевірки відповідності системи вимогам та очікуванням користувачів;
   - виправлення виявлених помилок та недоліків.
6. Впровадження та підтримка.

Кожен з цих етапів може бути розділений на дрібніші завдання та підзадачі, які виконуються в рамках команди розробників. Така структурна декомпозиція дозволяє керувати процесом розробки програмного забезпечення та забезпечує його ефективне виконання.

Наведемо структуру декомпозиції робіт проекту (табл. 2.1) та відповідну діаграму Ганта. Вони відображають етапи проекту, їх послідовність та взаємозв’язок. Діаграма Ганта – це інструмент для візуалізації та планування проекту в часовому аспекті. Вона використовується для відображення розкладу робіт по часу, а також для визначення залежностей між завданнями.

Основні цілі та використання діаграми Ганта включають:

1. Планування проекту: діаграма Ганта дозволяє структурувати та організувати роботу по проекту, визначити послідовність виконання завдань та встановити терміни їх виконання.
2. Визначення критичних шляхів: вона допомагає виявити критичні завдання, які мають найбільший вплив на загальний термін виконання проекту.
3. Управління ресурсами (людські, матеріальні, фінансові).
4. Допомагає відстежувати прогрес виконання робіт, ідентифікувати затримки та виявляти можливі проблеми, що допомагає приймати вчасні коригувальні заходи.

5. Комунікація**:** може використовуватися для звітування перед учасниками проекту, внутрішнього спілкування в команді та зовнішнього комунікування з зацікавленими сторонами [3].

Таблиця 2.1 – Структура декомпозиції робіт

№	Назва задачі	Початок	Завершення
1	Специфікація вимог	01.02.2024	13.02.2024
1.1	Опис характеристики предметної області	01.02.2024	06.02.2024
1.2	Функціональні вимоги	07.02.2024	12.02.2024
1.3	Нефункціональні вимоги	13.02.2024	15.02.2024
2	Проектування ІС	14.02.2024	26.03.2024
2.1	Визначення архітектури	14.02.2024	19.02.2024
2.2	Розробка UML-діаграми	20.02.2024	26.03.2024
3	Розробка фронтенду	05.03.2024	20.03.2024
4	Розробка бекенду	21.03.2024	06.05.2024
4.1	Створення архітектури	21.03.2024	26.03.2024
4.2	Реалізація бізнес-логіки	27.03.2024	11.04.2024
4.3	Інтеграція з БД	12.04.2024	25.04.2024
4.4	Тестування окремих модулів та компонентів	26.04.2024	06.05.2024
5	Тестування	07.05.2024	24.05.2024
5.1	Складання тесткейсів	07.05.2024	10.05.2024
5.2	Виправлення помилок	13.05.2024	24.05.2024
6	Впровадження та підтримка (оформлення пояснювальної записки)	27.05.2024	31.05.2024

Діаграму Ганта для ПЗ представлено на рис.2.1.

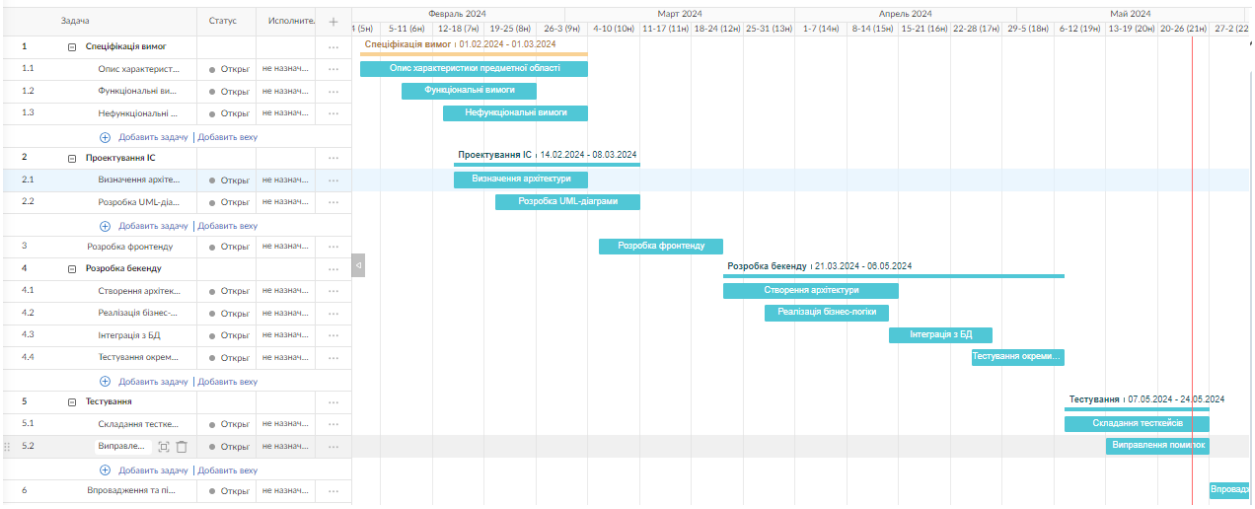

Рисунок 2.1 – Діаграма Ганта

2.2 Опис архітектури інформаційної системи

На етапі вибору архітектури для інформаційної системи слід передбачити майбутні доповнення до функціоналу. Є два типи архітектури для веб-сайтів, які обирають розробники: монолітна та мікросервісна (рис. 2.2).

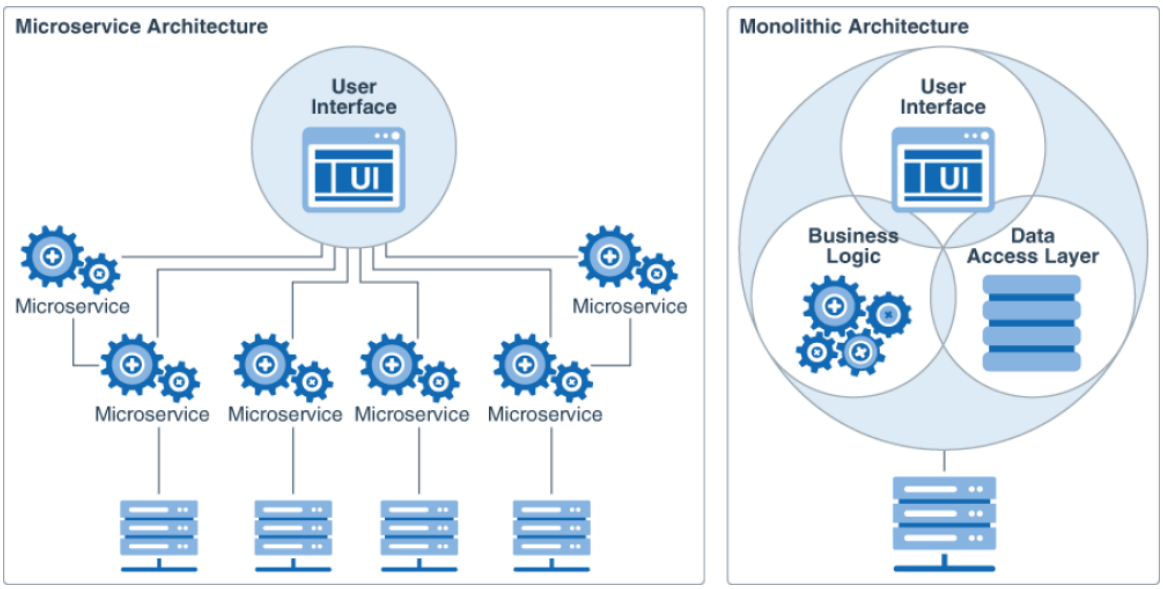

Рисунок 2.2 – Архітектури веб-сайтів

Монолітна архітектура – це архітектурний підхід, в якому вся основна логіка програми зібрана в одному місці. Монолітне додаток складається з одношарового поєднання різних компонентів в одне ціле. Розробка нової функціональності в такому додатку, де великий обсяг застарілого коду, з кожним роком стає дедалі дорожчим для бізнесу. Зі схожою проблемою вже стикалися бекенд-розробники.

Мікросервісна архітектура – це повна протилежність монолітної архітектури. Використовуючи такий підхід замість одного великого додатка, ми створюємо набір невеликих слабозв'язаних і легко замінюваних модулів, які взаємодіють один з одним. Одна з головних переваг мікросервісної архітектури – можливість використовувати найкращий технічний стек для кожного окремого завдання [4].

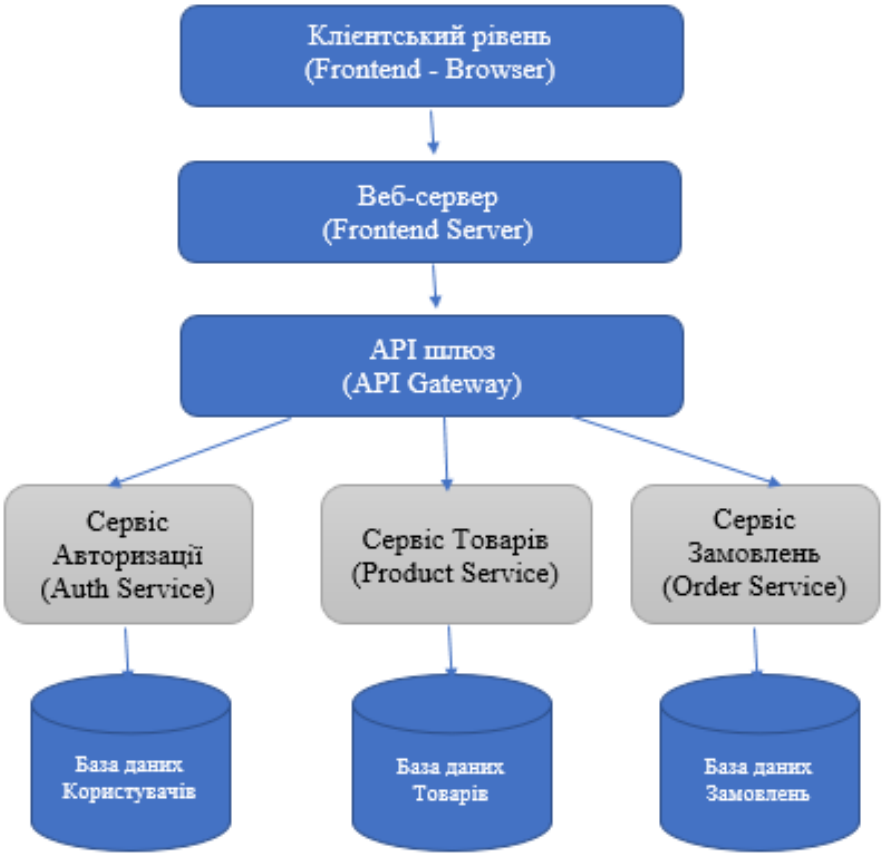

Рисунок 2.3 – Мікросервісна архітектура ІС

Опис діаграми:

1. Клієнтський рівень (Frontend- rowser): веб-браузер користувача взаємодіє з фронтендом веб-сайту.
2. Веб-сервер (Frontend Server), який відповідає за подачу статичних файлів (HTML, CSS, JavaScript) і рендеринг веб-сторінок.
3. API шлюз (API Gateway) служить єдиною точкою входу для всіх API-запитів від фронтенду, а також маршрутизує запити до відповідних сервісів на бекенді.
4. Сервіси (Backend Services):
   - Сервіс авторизації (Auth Service) відповідає за аутентифікацію та авторизацію користувачів;
   - Сервіс товарів (Product Service) вдповідає за управління каталогом товарів, включаючи пошук і перегляд товарів;
   - Сервіс замовлень (Order Service) відповідає за обробку замовлень, включаючи створення, оновлення та відстеження замовлень.
5. Бази даних (Databases):
   - БД користувачів (User DB) зберігає інформацію про користувачів, включаючи облікові записи та профілі;
   - БД товарів (Product DB) зберігає інформацію про товари, включаючи опис, ціни та залишки на складі;
   - БД замовлень (Order DB) зберігає інформацію про замовлення, включаючи деталі замовлень та статуси.

2.3 Проектування моделі даних

Перед безпосередньою реалізацією проекту необхідно розробити макет моделі даних, який дозволить більш продуктивна та продумано розробляти систему споживання контенту. Проектування моделі даних для інформаційної системи підтримки діяльності комерційного підприємства з продажу товарів

включає створення логічної структури бази даних, яка відображає ключові сутності та їх взаємозв'язки. Ось основні сутності та зв'язки між ними:

Основні сутності:

- Користувачі (Users);
- Товари (Products);
- Категорії (Categories);
- Замовлення (Orders);
- Деталі замовлень (OrderItems);
- Відгуки (Reviews);
- Платежі (Payments);
- Адміністратори (Admins);
- Доставки (Deliveries).

Зв'язки між сутностями:

- Користувачі можуть робити замовлення.
- Замовлення складаються з кількох деталей замовлень.
- Деталі замовлень пов'язані з конкретними товарами.
- Товари належать до певних категорій.
- Користувачі можуть залишати відгуки на товари.
- Замовлення можуть мати кілька платежів.
- Адміністратори керують товарами, категоріями, користувачами та іншими сутностями.
- Доставки пов'язані з замовленнями.

Для кожної сутності були визначені поля, які максимально описують її характеристики. Опис атрибутів для сутності «Користувачі» (Users):

- user_id (PK): унікальний ідентифікатор користувача.
- name: ім'я користувача.
- email: електронна пошта користувача.
- password: пароль користувача (хешований).
- інші атрибути: адреса, номер телефону, тощо.

Опис атрибутів для сутності «Товари» (Products):

- product_id (PK): унікальний ідентифікатор товару.
- category_id (FK): ідентифікатор категорії, до якої належить товар.
- name: назва товару.
- description: опис товару.
- price: ціна товару.
- stock: кількість на складі.
- інші атрибути: зображення, специфікації, тощо.

Опис атрибутів для сутності «Категорії» (Categories):

- category_id (PK): унікальний ідентифікатор категорії.
- name: назва категорії.
- description: опис категорії.

Опис атрибутів для сутності «Замовлення» (Orders):

- order_id (PK): унікальний ідентифікатор замовлення.
- user_id (FK): ідентифікатор користувача, який зробив замовлення.
- total_amount: загальна сума замовлення.
- order_date: дата замовлення.

Опис атрибутів для сутності «Деталі замовлень» (OrderItems)

- order_item_id (PK): унікальний ідентифікатор деталі замовлення.
- order_id (FK): ідентифікатор замовлення.
- product_id (FK): ідентифікатор товару.
- quantity: кількість товару.
- price: ціна за одиницю товару.

Опис атрибутів для сутності «Відгуки» (Reviews):

- review_id (PK): унікальний ідентифікатор відгуку.
- user_id (FK): ідентифікатор користувача, який залишив відгук.
- product_id (FK): ідентифікатор товару.
- rating: рейтинг товару.
- comment: коментар користувача.
- review_date: дата відгуку.

Опис атрибутів для сутності «Платежі» (Payments):

- payment_id (PK): унікальний ідентифікатор платежу.
- order_id (FK): ідентифікатор замовлення.
- amount: сума платежу.
- payment_date: дата платежу.

Опис атрибутів для сутності «Доставки» (Deliveries):

- delivery_id (PK): унікальний ідентифікатор доставки.
- order_id (FK): ідентифікатор замовлення.
- delivery_date: дата доставки.
- status: статус доставки (в дорозі, доставлено, тощо).

На рис. 2.4 представлено діаграму сутностей і зв’язків між ними

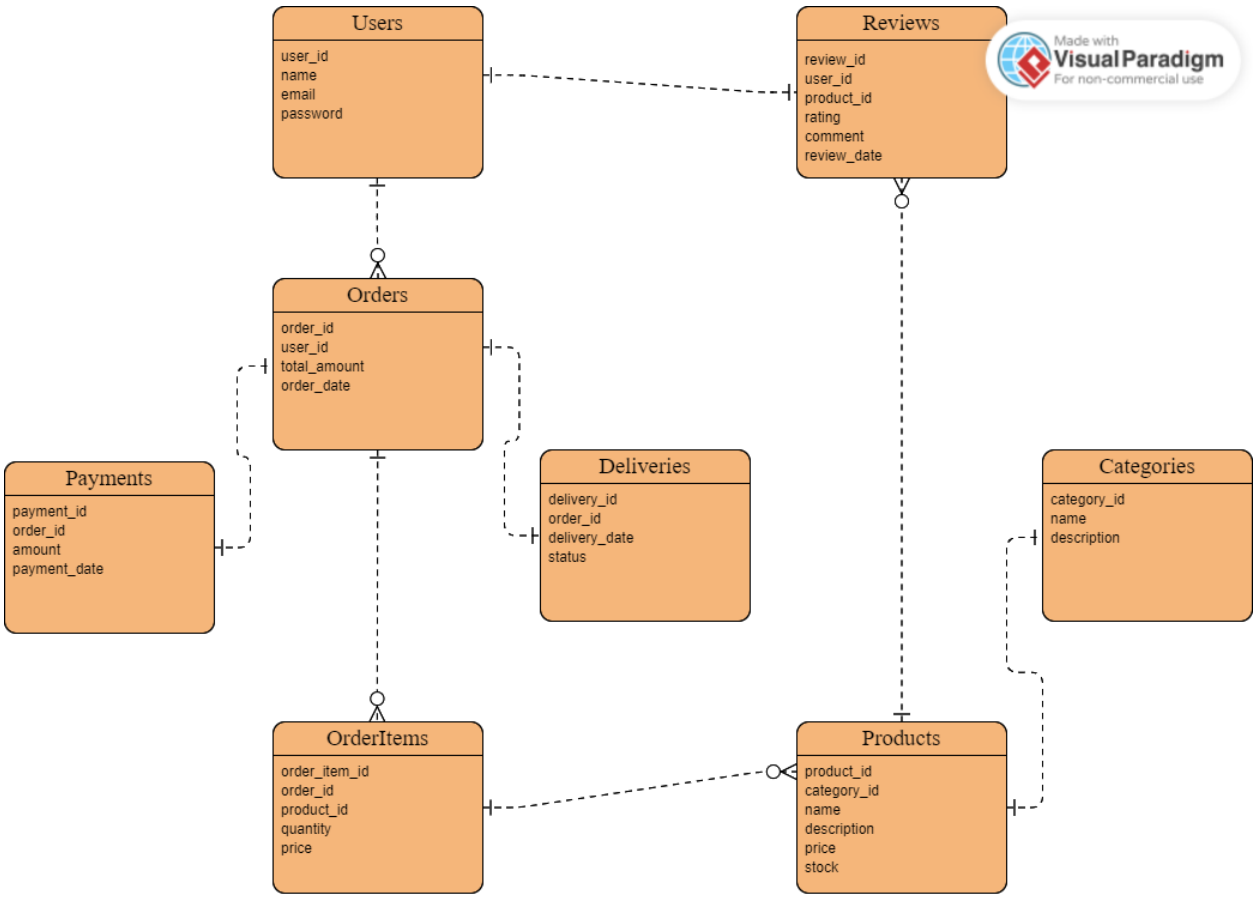

Рисунок 2.4 – ER-діаграма

Користувачі роблять замовлення, які можуть мати декілька деталей замовлень. Деталі замовлень включають товари, які відносяться до певних категорій. Користувачі можуть залишати відгуки на товари. Замовлення мають пов'язані платежі та доставки. Ця модель даних забезпечує необхідну структуру для підтримки всіх ключових функцій інформаційної системи з продажу товарів через веб-сайт.

2.4 Моделювання поведінки системи

Моделювання поведінки інформаційної системи, крім створення сценаріїв використання, передбачає також опис діаграм послідовності, діаграм станів і діяльності. Це допомагає зрозуміти, як система буде функціонувати в різних ситуаціях та як користувачі будуть взаємодіяти з нею. Спочатку слід описати діаграму активності Користувача під час роботи з інформаційною системою (рис. 2.5).

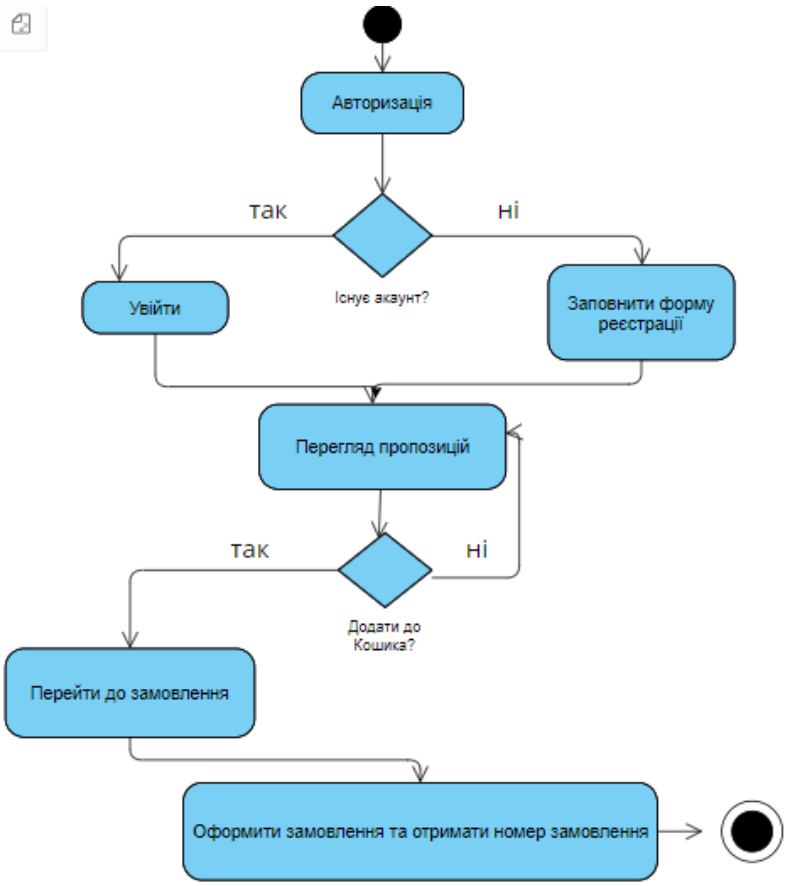

Рисунок 2.5 – Діаграма активності

Тут основну увагу зафіксовано на перевірці наявності створеного акаунту користувача. Від цього залежить перелік додаткових функцій (список уподобань, перегляд історії замовлень та інше). Діаграму станів для діяльності «Пошук товару» представлено на рис. 2.6, на рис.2.7 діаграму станів.

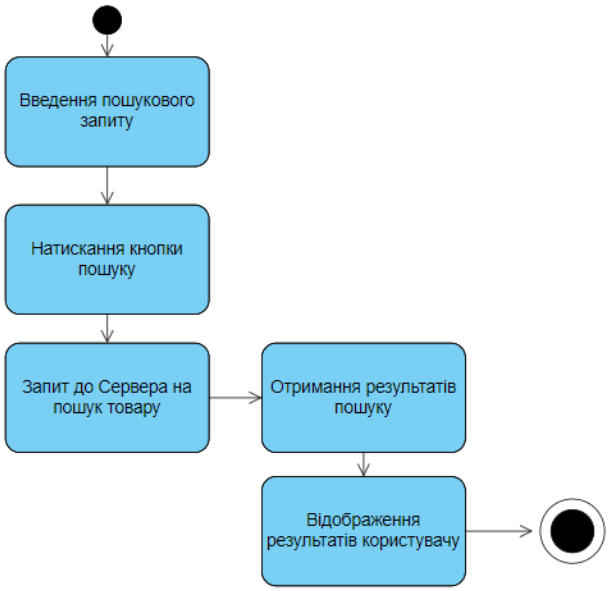

Рисунок 2.6 – Діаграма станів для діяльності «Пошук товару»

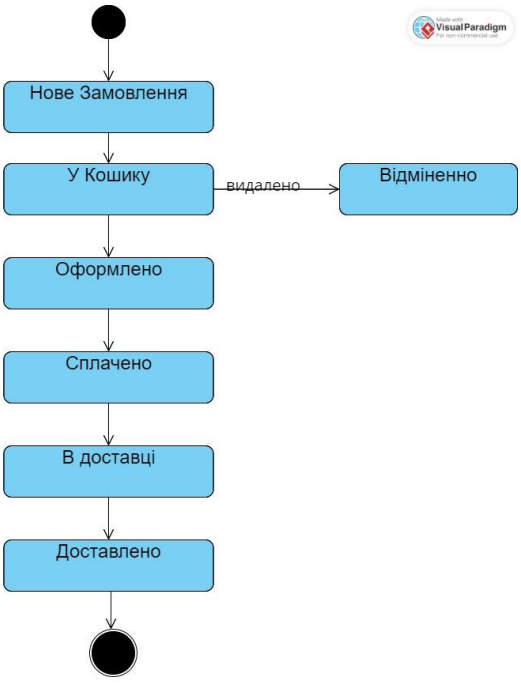

Рисунок 2.7 – Діаграма станів замовлення

3 ВИБІР ЗАСОБІВ РОЗРОБКИ

3.1 Серверна частина

Для серверної частини веб-сайту інформаційної системи з продажу товарів можна використовувати ASP.NET Core та C#. ASP.NET – це веб-фреймворк від Microsoft, що дозволяє створювати динамічні веб-додатки, сервіси та API. Найновіша версія, ASP.NET Core, є кросплатформеним, високопродуктивним і відкритим вихідним кодом.

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний мене-джер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core є одним з найшвидших веб-фреймворків, що забезпечує швидке обслуговування запитів і обробку даних. Він працює на різних платформах, включаючи Windows, Linux і macOS, що дозволяє розгортати додаток на будь-яких серверах. Фреймворк дозволяє підключати тільки необхідні компоненти, що зменшує розмір додатка і підвищує його продуктивність. ASP.NET Core легко інтегрується з іншими фреймворками та бібліотеками, надаючи можливості для масштабування і кастомізації.

В якості інструментарію розробки ми можемо використовувати останні випуски Visual Studio, починаючи з версії Visual Studio 2018. Крім того, ми можемо створювати додатки в середовищі Visual Studio Code, яка є крос-платформної і може працювати як на Windows, так і на Mac OS X і Linux. Для обробки запитів тепер використовується новий конвеєр HTTP, який заснований на компонентах Katana і специфікації OWIN. А його модульність дозволяє легко додати свої власні компоненти.

Крім об’єднання вищезазначених технологій в одну модель в MVC був доданий ряд додаткових функцій. Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #. Використання C# як основної мови програмування дозволяє скористатися

всіма перевагами цієї мови, включаючи сильну типізацію, сучасні функціональні можливості та хорошу підтримку візуальних середовищ розробки (наприклад, Visual Studio).

3.2 Клієнтська частина

3.2.1 HTML, CSS та їх основні елементи

HTML (Hypertext Markup Language – мова гіпертекстової розмітки) – це мова опису структури сторінок документів, яка дозволяє звичайний текст форматувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки. Це текстова мова, в якій інструкції з форматування (теги), вбудовані в розділи документа, які містять конкретну інформацію.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об’єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Разом з Cascading Style Sheets (CSS) і JavaScript, HTML утворює тріаду основних технологій для World Wide Web [11].

CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що використовується для опису їх зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS має різні рівні та профілі. Профілі – сукупність правил CSS од-ного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів.

Одна з головних переваг використання CSS – це можливість розділити зміст сторінки від її оформлення. Таке розділення дозволило покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структуро-

ваним та простим, прибрати повторення та ін. Власне це ж і була основна мета створення цієї технології.

Що дає використання CSS:

- відображати один і той же документ в різних стилях;
- декілька дизайнів сторінки для різних пристроїв.

Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на смартфоні меню буде внизу, під вмістом.

3.2.2 React та JS

ReactJS, також відомий як React, є популярною бібліотекою JavaScript для створення інтерфейсів користувача. Його також називають інтерфейсною бібліотекою JavaScript. Він був розроблений Facebook і широко використовується для створення динамічних та інтерактивних веб-додатків. У цій статті ми розглянемо ключові концепції React.

React – це бібліотека JavaScript для створення інтерфейсів користувача (UI) в Інтернеті. React – це декларативна бібліотека на основі компонентів, яка дозволяє розробникам створювати повторно використовувані компоненти інтерфейсу користувача, і вона дотримується підходу Virtual DOM (Document Object Model), який оптимізує продуктивність відтворення шляхом мінімізації оновлень DOM. React швидкий і добре працює з іншими інструментами та бібліотеками.

React працює, створюючи віртуальний DOM у пам’яті, а не безпосередньо маніпулюючи DOM браузера. Він виконує необхідні маніпуляції в цьому віртуальному представленні перед застосуванням змін до фактичного DOM браузера. React ефективний, змінюючи лише те, що потребує модифікації.

Використання React на клієнтській частині разом з ASP.NET Core на серверній частині забезпечує потужну, гнучку і масштабовану архітектуру для розробки веб-додатків. Така комбінація технологій дозволяє створювати сучасні, динамічні та чуйні веб-додатки, що забезпечують відмінний користувацький досвід [5].

Клієнтська частина (React):

- відповідає за інтерфейс користувача;
- виконує http-запити до серверного API;
- використовує бібліотеки та інструменти для управління станом, маршрутизації, стилізації тощо.

Серверна частина (ASP.NET Core з C#):

- обробляє запити від клієнта;
- виконує бізнес-логіку;
- взаємодіє з базою даних;
- повертає дані у форматі json для використання на фронтенді [6].

Переваги використання React та C#:

1. Чітке розділення обов'язків: фронтенд і бекенд чітко розділені, що полегшує розробку і підтримку.
2. Масштабованість: легко масштабувати як клієнтську, так і серверну частини.
3. Висока продуктивність: React забезпечує високу продуктивність на фронтенді, а ASP.NET Core – на бекенді.
4. Сучасні інструменти для швидкої розробки.

Ця архітектура дозволяє створювати потужні, сучасні веб-додатки, які легко підтримувати та розширювати, забезпечуючи відмінний користувацький досвід.

3.2.3 Вибір СУБД

Вибір бази даних для інформаційної системи залежить від кількох факторів, таких як вимоги проекту, інфраструктура, досвід та переваги розробника. Слід розглянути декілька найпопулярніших представників СУБД, а саме, основні особливості SQL Server, MySQL та PostgreSQL.

1. SQL Server.

Корпорація Майкрософт пропонує SQL Server у чотирьох основних випусках, які надають різні рівні пакетних служб. Безкоштовне повнофункціональне видання Developer – це видання Express, яке можна використовувати для запуску невеликих баз даних із ємністю диска до 10 ГБ. Версія для розробників не має ліцензії на використання у виробництві. Великі додатки, які потребують підтримки на робочому рівні, ліцензуються як Enterprise Edition.

Версія Standard має скорочений набір функцій і обмежену масштабованість через обмеження кількості ядер ЦП, які вона може використовувати, і розміру пам’яті. Через зростання конкуренції наприкінці 2016 року Microsoft зробила функції Enterprise доступними для стандартної версії. До них належать In-Memory OLTP, PolyBase, індекси columnstore, розділення, стиснення даних і можливості захоплення змінених даних.

Програми використовують SQL Server за допомогою багатьох інтерфейсів. Інтерфейс ODBC забезпечує високорівневий інтерфейс SQL, який дозволяє користувачам вбудовувати виклики бази даних у такі програми, як Microsoft Excel. Програми Java використовують драйвер JDBC, щоб дозволити їм отримувати доступ до баз даних за допомогою SQL.

Розробники додатків використовують інтерфейси прикладного програмування (API) для вбудовування операторів SQL у свої додатки, які можна написати, наприклад, на C, Java та Python. Рядки бази даних можна отримувати по одному або пакетами чи масивами [7].

Microsoft Visual Studio включає вбудовану підтримку Microsoft SQL Server. Visual Studio містить конструктор даних для створення, перегляду та редагування схем бази даних у графічному вигляді. Запити також можна створювати візуально.

Переваги:

- інтеграція з продуктами Microsoft: у разі використання інших продуктів Microsoft (наприклад, .NET, Azure), SQL Server може бути зручнішим завдяки тісній інтеграції;
- безпека та підтримка: SQL Server має вбудовані засоби безпеки та шифрування даних, а також підтримується Microsoft, що забезпечує регулярні оновлення та підтримку;
- аналітика та BI: В SQL Server є розширені функції аналітики, інтеграція з Power BI та інші інструменти для бізнес-аналітики;
- High Availability (HA): SQL Server підтримує різні методи забезпечення високої доступності та відновлення після відмови (Failover Clustering, Always On Availability Groups).

Недоліки:

- вартість: ліцензії на SQL Server можуть бути дорогими, особливо для Enterprise Edition;
- платформозалежність: SQL Server переважно працює на Windows, хоча існує і версія для Linux.

2. MySQL.

MySQL є однією з найбільш впізнаваних технологій у сучасній екосистемі великих даних. Її часто називають найпопулярнішою базою даних, яка наразі користується широким та ефективним використанням незалежно від галузі. Зрозуміло, що будь-хто, хто займається корпоративними даними чи загальними IT-технологіями, повинен прагнути принаймні до базових знань MySQL [8].

З MySQL навіть новачки в реляційних системах можуть відразу створювати швидкі, потужні та безпечні системи зберігання даних. Програмний синтаксис та інтерфейси MySQL також є ідеальними шлюзами до широкого світу інших популярних мов запитів і структурованих сховищ даних.

Переваги:

- відкритий код та вартість: MySQL має відкритий код і є безкоштовним для багатьох використань. Також є платні комерційні версії з додатковими функціями;
- широка підтримка та спільнота: MySQL є одним з найпопулярніших СУБД у світі, тому є велика кількість ресурсів та спільнот для підтримки;
- простота у використанні: MySQL відомий своєю простотою в установці та конфігурації;
- крос-платформеність: працює на різних платформах, включаючи Windows, Linux та macOS.

Недоліки:

- функціональність: деякі розширені функції можуть бути менш розвинені, ніж у PostgreSQL або SQL Server;
- відсутність повної підтримки стандартів SQL: деякі розширені можливості SQL можуть бути обмеженими.

3. PostgreSQL.

PostgreSQL – це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші дані. Витоки PostgreSQL сягають 1986 року як частини проекту POSTGRES в Каліфорнійському університеті в Берклі та мають понад 35 років активного розвитку на основній платформі.

PostgreSQL заслужив міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти з відкритим кодом, яка стоїть за програмним забезпеченням, для постійного надання продуктивних та інноваційних рішень. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID з 2001 року та має потужні додаткові модулі, такі як популярний розширювач геопросторових баз даних PostGIS. Не дивно, що PostgreSQL стала реляційною базою даних з відкритим вихідним кодом для багатьох людей і організацій [9].

Переваги:

- відкритий код та вартість: PostgreSQL повністю відкритий і безкоштовний для використання;
- розширені можливості: відомий своєю підтримкою складних SQL-запитів, транзакцій, розширень (як-от PostGIS для геопросторових даних) та додаткових типів даних;
- стандартність: PostgreSQL дотримується стандартів SQL і має розширені можливості для обробки даних;
- надійність та масштабованість: Підтримує потужні можливості для забезпечення цілісності даних та масштабованості.

Недоліки:

- складність: може бути більш складним у налаштуванні та управлінні порівняно з MySQL;
- підтримка: хоча існує велика спільнота, комерційна підтримка може бути менш доступною або потребувати додаткових витрат.

Для дипломної роботи обрано СУБД PostgreSQL.

4 ОПИС РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Карта сайту

Карта сайту є важливим інструментом, який служить для покращення користувацького досвіду, ефективної розробки та дизайну, оптимізації для пошукових систем і управління контентом. Вона забезпечує чітке уявлення про структуру сайту, допомагає уникнути плутанини та спрощує навігацію для всіх зацікавлених сторін.

Ось основні причини, для чого будують карту сайту:

1. Для користувачів:
   - покращення навігації: карта сайту надає користувачам загальний огляд структури сайту;
   - зменшення часу на пошук: чітка карта сайту дозволяє користувачам швидше орієнтуватися на сайті, зменшуючи час на пошук необхідних розділів або сторінок;
   - зручність використання: користувачі можуть легко бачити, як різні розділи сайту пов'язані між собою, що робить використання сайту більш інтуїтивно зрозумілим.
2. Для розробників і дизайнерів:
   - планування структури сайту: як буде організований контент і як користувачі будуть взаємодіяти з різними розділами;
   - полегшення розробки: чітка структура дозволяє розробникам ефективно планувати розробку різних частин сайту, забезпечуючи узгодженість і логічність у побудові інтерфейсу;
   - виявлення недоліків: на етапі планування карта сайту допомагає виявити потенційні проблеми з навігацією або структурою контенту, які можна вирішити до початку розробки.
3. Для пошукових систем (SEO):
   - покращення індексації: допомагає пошуковим системам, краще розуміти структуру сайту та індексувати його сторінки;

- сканування нових сторінок: карта сайту дозволяє пошуковим системам швидше знаходити нові або оновлені сторінки, що може покращити їх рейтинг у результатах пошуку;
- включення важливих сторінок у карту сайту може сигналізувати пошуковим системам і покращити їх видимість.

4. Для управління контентом:
   - контроль над контентом: карта сайту надає адміністраторам інструмент для контролю над структурою і змістом сайту, допомагаючи легко вносити зміни або оновлення;
   - планування контенту: знаючи структуру сайту, менеджери контенту можуть планувати створення нових розділів або сторінок, які логічно вписуються в існуючу структуру.

Рисунок 4.1 – Карта веб-сайту «Tulip Store»

На сторінці «Каталог товарів» доступні такі функції:

- перегляд категорій;

– фільтри товарів;
– сортування товарів.

У «Профілі користувача» будуть доступні функції:

– перегляд профілю;
– редагування профілю;
– історія замовлень;
– відгуки;
– налаштування.

Також на сайті слід реалізувати панель адміністратора для зручного керування контентом та управлінням базою замовлень і клієнтів. Структура панелі адміністратора представлено на рисунку 4.2:

Рисунок 4.2 – Структура панелі адміністратора

4.2 Інтерфейс користувача

Розглянемо структуру інтерфейсі більш детально. У футері представлено інформацію щодо назви кампанії у вигляді логотипу.

Елемент керування «Меню», строка пошуку, а також права частина хедера (рис. 4.3) містить іконки для вибору мови (англійська/українська), вхід до аккаунту, перехід до розділу улюблених товарів та кошик користувача.

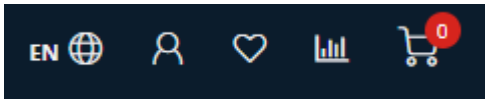

Рисунок 4.3 – Права частина хедера

Для використання функції «улюблені товари» користувачу слід авторизуватися (рис. 4.4) або зареєструватись у разі відсутності особистого аккаунту (рис. 4.5).

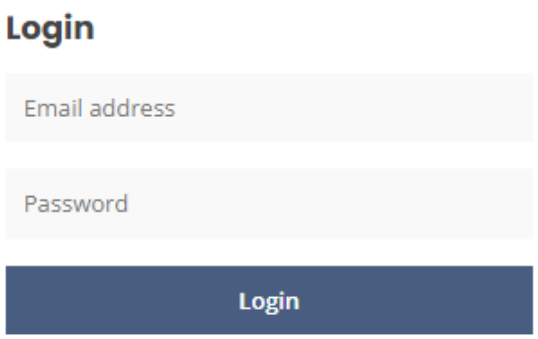

Рисунок 4.4 – Форма для авторизації користувача

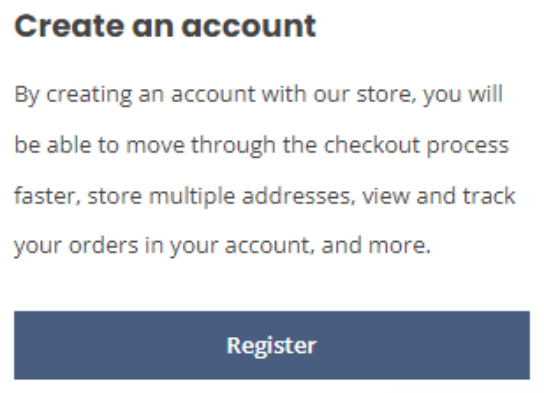

Рисунок 4.5 – Форма для переходу на сторінку реєстрації

Для реєстрації необхідно заповнити поля імені, електронної пошти, номер телефону для зворотнього зв’язку під час відстеження замовлення, а також пароль для входу до особистого кабінету. Сторінка реєстрації нового користувача представлено на наступному рисунку:

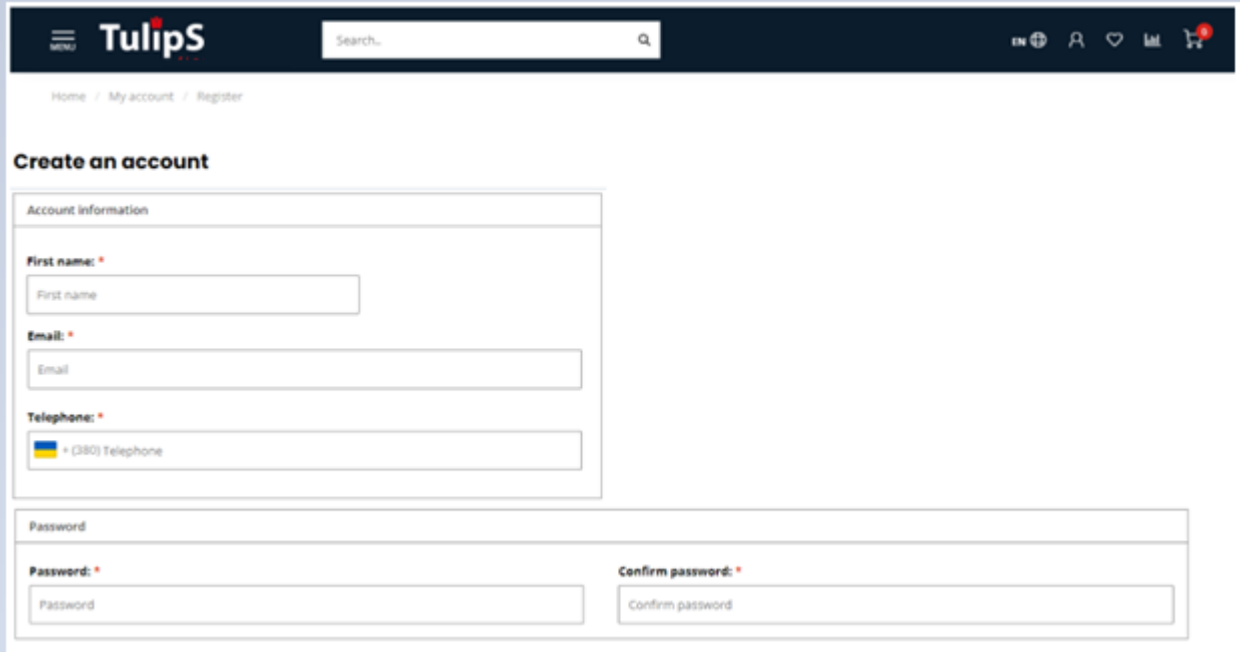

Рисунок 4.6 – Сторінка реєстрації

Центральна частина головної сторінки містить ознайомчу інформацію та кнопку для перегляду всього асортименту пропозицій кампанії. (рис. 4.7). Сторінку товарів представлено на рис. 4.8. З лівої частини розміщено меню за категоріями рослин, у центральній часті представлені елементи з інформацією щодо кожної позиції асортименту: назва квітки, фото, короткий опис, ціна та кількість у одному пакуванні.

Крім того, для кожної квітки є покажчик продажу (5 зірок – у топі продажу). Якщо кількість зірок взагалі невизначено, то це означає або нову квітку у асортименті, або початок сезону для даної рослини.

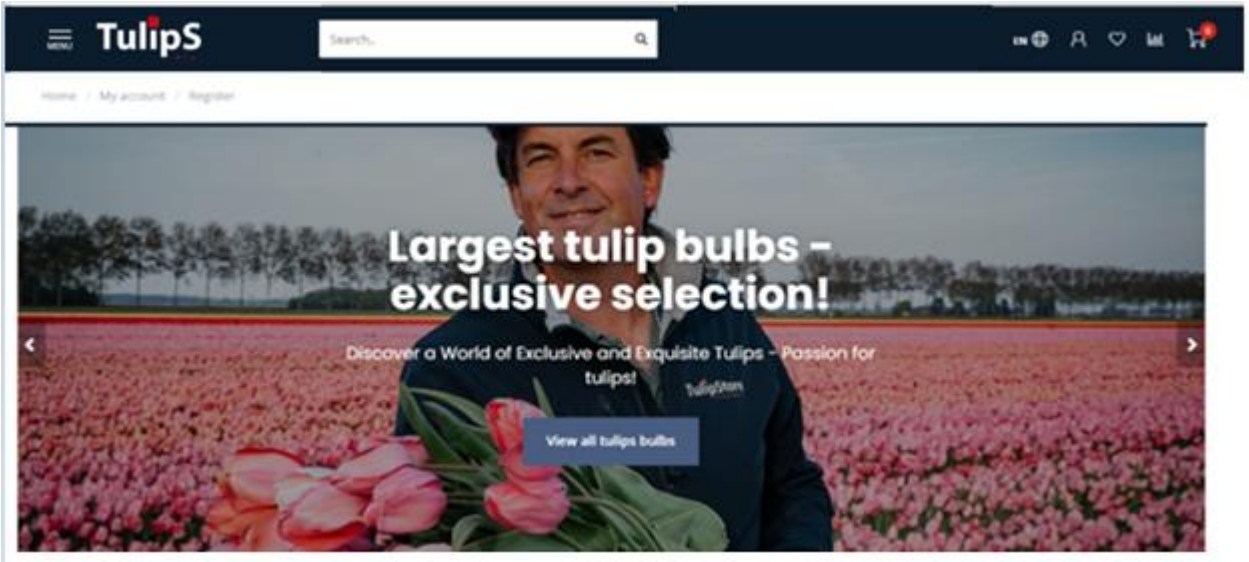

Рисунок 4.7 – Центральний блок головної сторінки

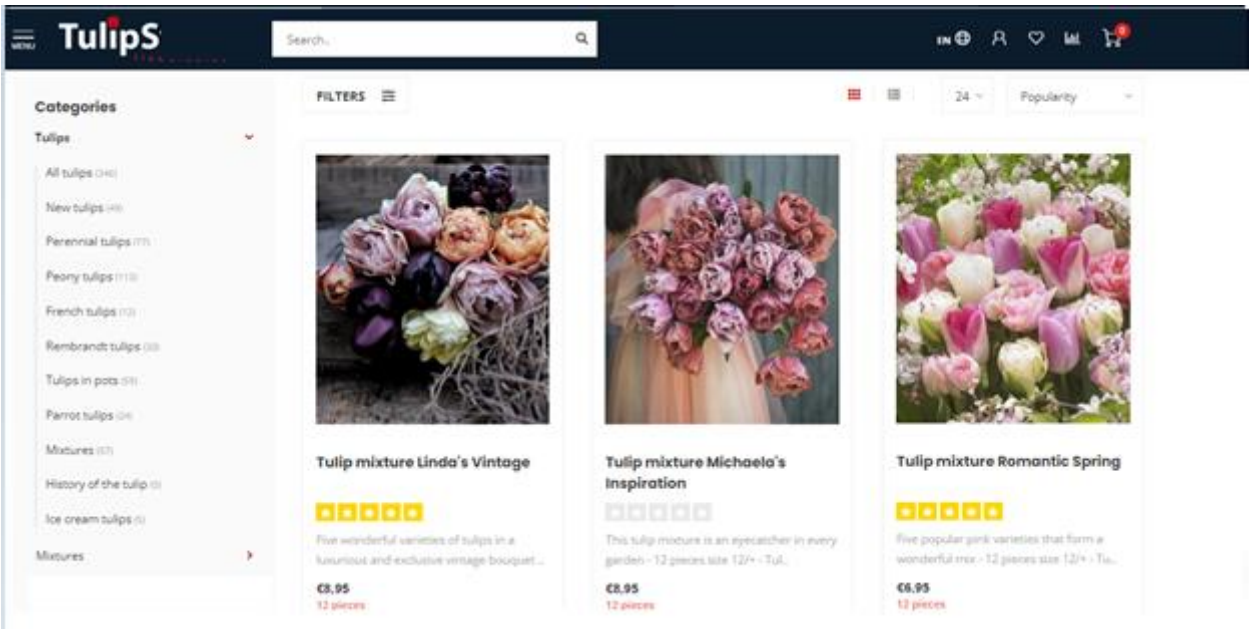

Рисунок 4.8 – Сторінка асортименту пропозицій

Для кожного елементу каталогу при наведені курсора з’являються три додаткові функції (рис. 4.9):

- додаткова інформація (рис. 4.10);
- додавання квітки до списку порівнянь;

– додавання квітки до закладки улюблених товарів.

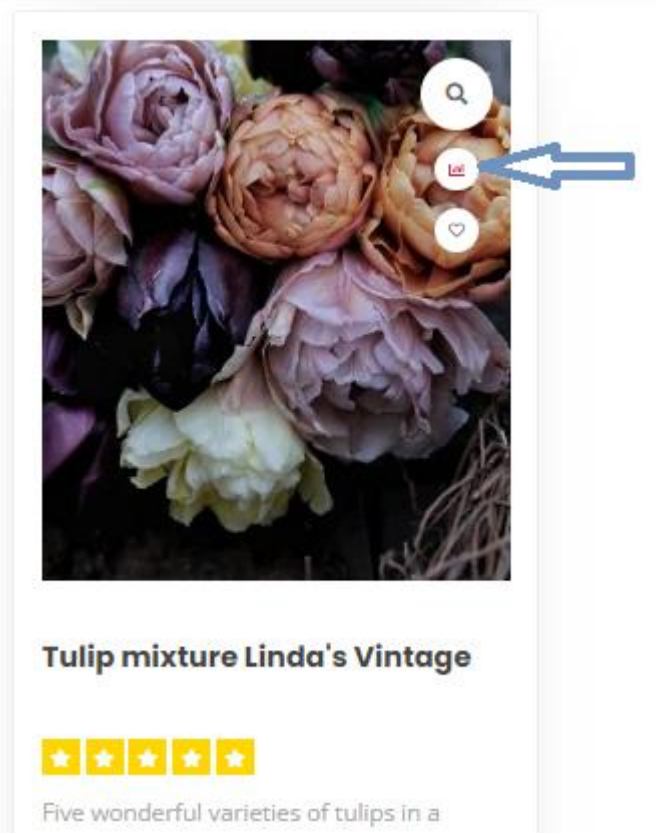

Рисунок 4.9 – Додаткові елементи для роботи з об’єктами

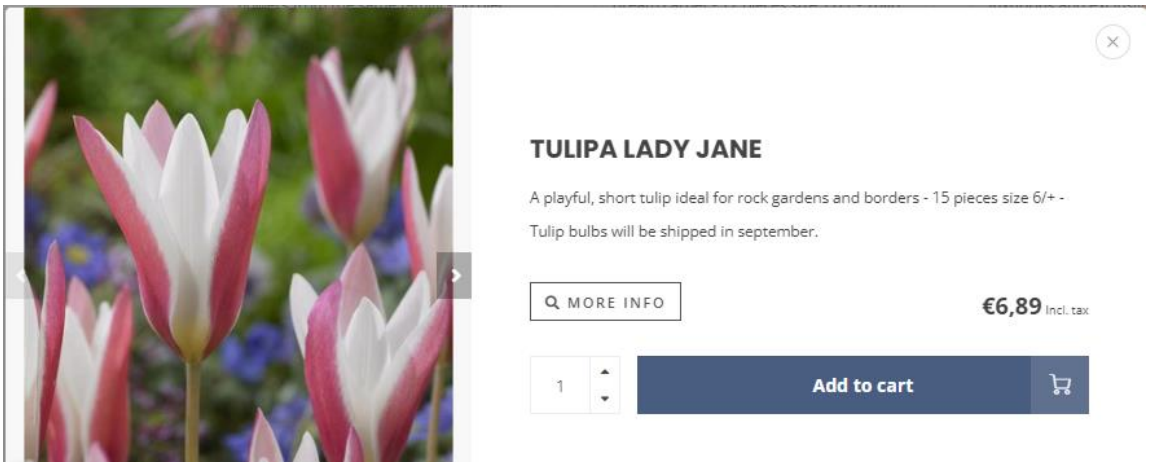

Рисунок 4.10 – Вікно з додатковою інформацією про обрану квітку

Через кнопку «More Info» можна відкрити сторінку з карткою товару, де буде представлено всю інформацію щодо рослини, сезон її цвітіння та інша специфікація (рис. 4.11).

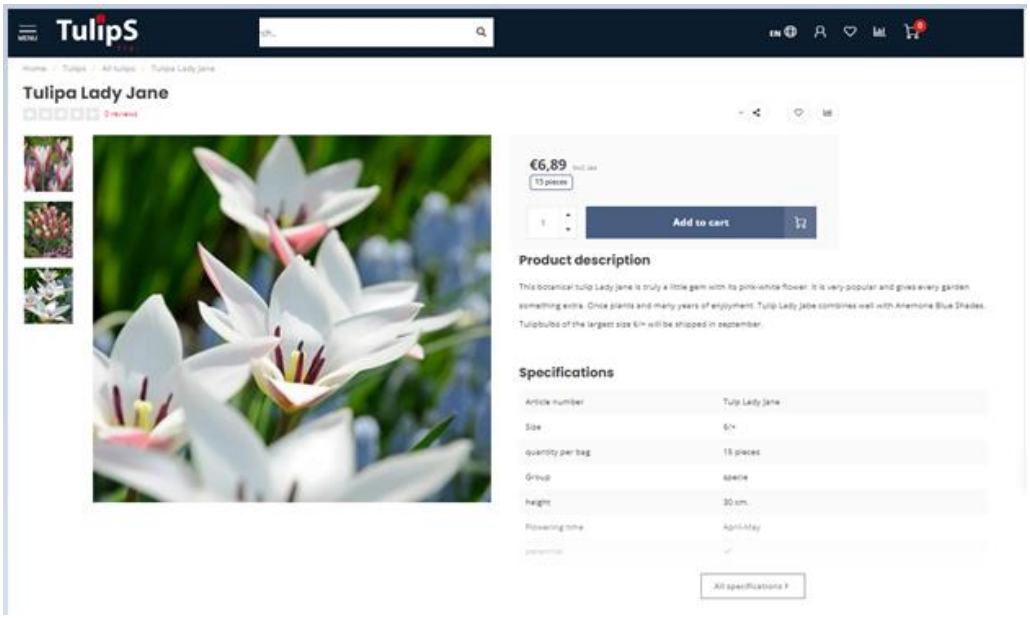

Рисунок 4.11 – Сторінка картки товару

Функція «Додавання квітки до списку порівнянь» є дуже зручною для клієнтів при формуванні свого замовлення. У хедері через «Compare products» (рис. 4.12) можна перейти до наступної таблиці (рис. 4.13).

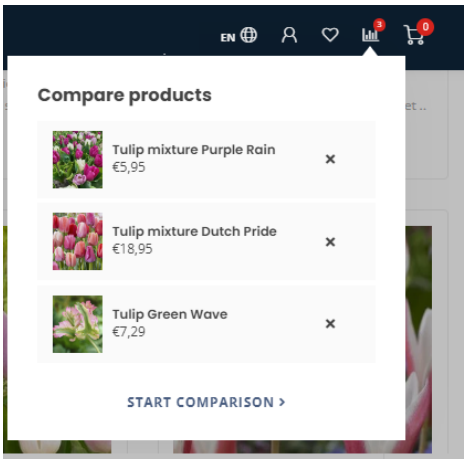

Рисунок 4.12 – Елемент хедеру «Compare products»

Home / Compare products

Compare products

	Tulipa Purple Rain	Tulipa Dutch Pride	Tulipa Green Wave
	 €5,95 Incl. tax	 €18,95 Incl. tax	 €7,29 Incl. tax
Brand:			
Description:	Three early tulips from the same family in a wonderful mix - 12 pieces size 12/+ - Tulip bulbs will be shipped in september.	Seven wonderful, strong tulips with large flowers from the same family - 50 pieces size 12/+ - Tulip bulbs will be shipped in september!	A wonderful, dainty tulip with a large flower - 10 pieces size 12/+ - Tulip bulbs will be shipped in september.
Article number:	Tulp Purple Rain mix	Tulp Dutch Pride mix	Tulp Green Wave
Size:	12/+	12/+	12/+
quantity per bag:	12 pieces	50 pieces	10 pieces
Group:	triumph	darwinhybrid	parrot
height:	35 cm.	60 cm.	50 cm
Flowering time:	April	April	May
perennial:	no	yes	no
odorous:	no	no	no
quantity per m2:	70	70	60
Price per unit:			
Weight:			
Sizes:			
	Add to cart Add to wishlist Delete	Add to cart Add to wishlist Delete	Add to cart Add to wishlist Delete

Рисунок 4.13 – Сторінка з повним описом «Compare products»

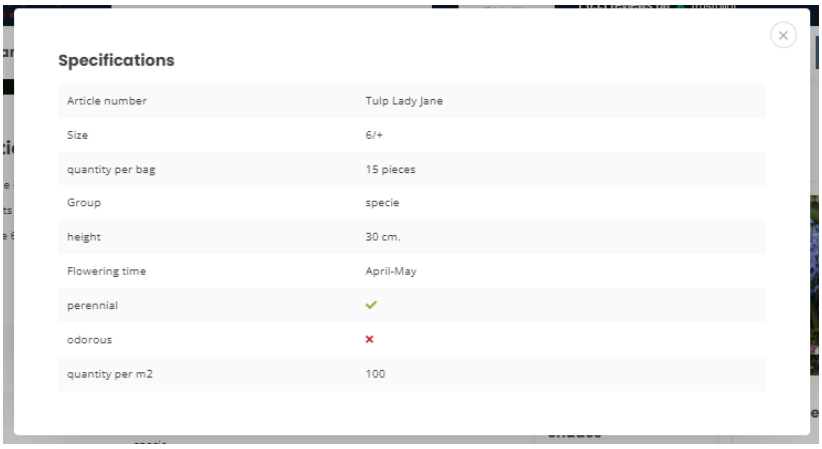

Рисунок 4.14 – Приклад повної специфікації квітки

У футері сайту розміщено блок з новими надходженнями товару (рис. 4.15), а також блог підприємства (рис. 4.16).

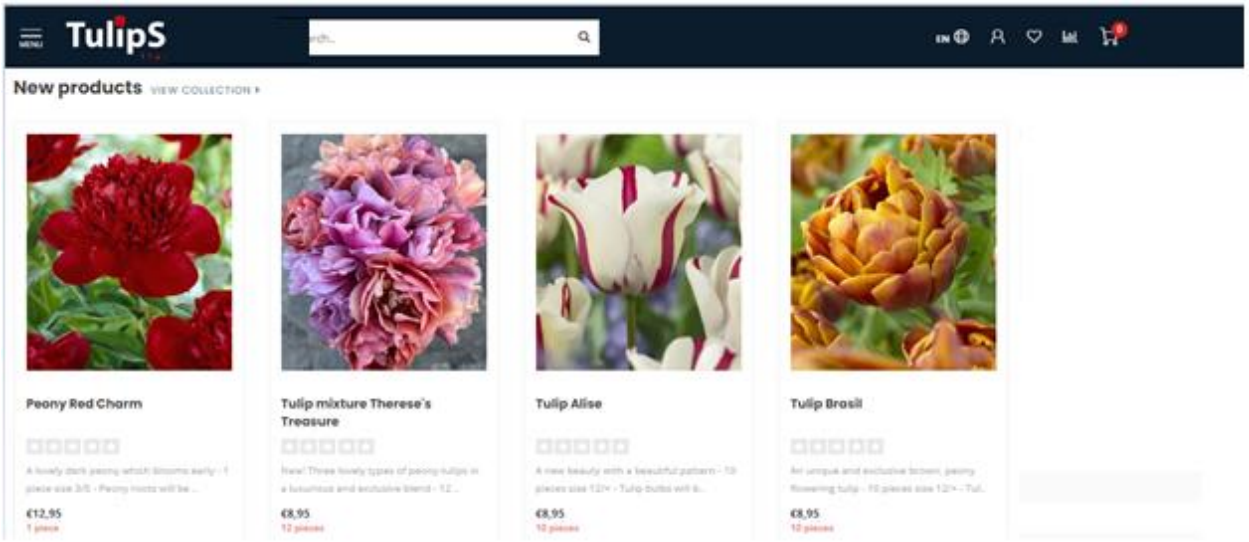

Рисунок 4.16 – Блок з новинками асортименту

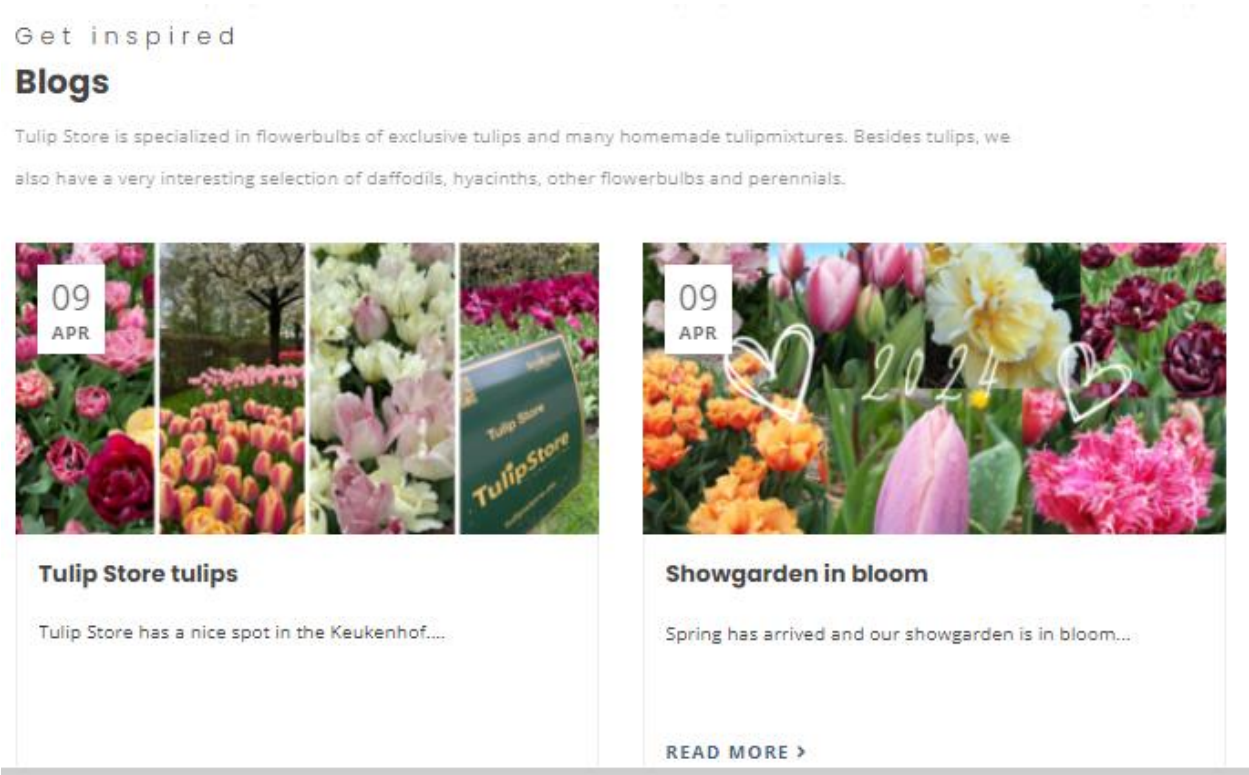

Рисунок 4.17 – Елемент блогу

4.3 Опис реалізації основних функцій

4.3.1 Однофакторна аутентифікація

Однофакторна аутентифікація (SFA) – метод автентифікації з використанням лише однієї категорії. Найбільш поширеним прикладом SFA є облікові дані, пов'язані із запровадженням імені користувача та звичайного пароля. Для поля з паролем наявні вимоги щодо мінімальної довжини пароля та його коректного повторного введення, це зроблено для забезпечення достатньої надійності пароля та впевненості у коректності його вводу. Якщо під час підтвердження паролю значення обох полів не співпадають, система виведе повідомлення [10].

Для авторизації це реалізовано наступним чином:

```
module.exports.login = async function (req, res) {
     try {
          if (!req.body.name) throw new Error(Enter Your
Name!');
          if (!req.body.password) throw new Error('Enter
Password');

          const name = sanitize(req.body.name).trim();
          const password = sanitize(req.body.password).trim();

          if (name.length > 50) throw new Error(The maximum
length of the username is 50 characters');
          if (password.length > 50) throw new Error('The maximum
password length is 20 characters ');

          const dbo = getDBO(req);
          const user = await
dbo.collection(Collection.USER).findOne({name});

          if (!user) {
               // Not Found
               res.status(404).json({
                    message: " Not Found.",
               });
               return;
          }
```

```
          if (user.status !== AccountStatus.ACTIVE) {
               // Inactive
               res.status(401).json({
                    message: "Inactive",
               });
               return;
          }

          const passwordResult = bcrypt.compareSync(password,
user.password);

          if (!passwordResult) {
               // Unauthorized
               res.status(401).json({
                    message: " Username and password are not
valid ",
               });
               return;
          }

          createCookie(res, user);
          res.status(200).json(User.toJson(req, user));
     } catch (error) {
          errorHandler(res, error);
     }
};
```

Описана функція відповідає за авторизацію користувача. Відбувається пошук у базі даних користувача за наданим іменем, перевірка чи є аккаунт активованим та перевірка відповідності паролей.

У разі некоректності отриманих даних сервер відправляє відповідне повідомлення, що буде відображене користувачу на фронтенді. У разі проходження всіх перевірок буде викликано функцію createCookie, для створення cookie з id аккаунта та терміном валідності в 9 годин, та відправка користувачу даних акаунту.

Для реєстрації:

```
module.exports.register = async function (req, res) {
     try {
          const dbo = getDBO(req);
          const name = sanitize(req.body.name)?.toLowerCase();
```

```
          const alreadyCreatedUser = await
dbo.collection(Collection.USER).findOne({name});

          if (alreadyCreatedUser) {
               res.status(403).json({ message: "Ім’я користувача
вже зайнято" });
               return;
          }

          const user = new User(req);
          const result = await
dbo.collection(Collection.USER).insertOne(user);
          const newUser = await
dbo.collection(Collection.USER).findOne({_id:
result.insertedId});

          res.status(200).json(User.toJson(req, newUser));
     } catch (error) {
          errorHandler(res, error);
     }
};
```

Описана функція відповідає за створення нового облікового запису користувача. Після проходження перевірки на унікальність отриманого ім’я облікового запису конструктор класу User створює нового користувача.

Далі функція зберігає нового користувача у БД. У разі існування облікового запису з таким же іменем сервер відправляє відповідне повідомлення, що буде відображене користувачу на фронтенді.

4.3.2 Опис основних компонентів системи

Файл «Program.cs» містить точку входу до додатку і визначає створення та налаштування веб-хосту:

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
```

```
                webBuilder.UseStartup<Startup>();
            });
}
```

Файл «Startup.cs» налаштовує служби та обробку запитів:

```
    public class Startup
    {
        public void ConfigureServices(IServiceCollection
services)
        {
            services.AddControllersWithViews();
            services.AddDbContext<ApplicationDbContext>(options
=>

options.UseSqlServer(Configuration.GetConnectionString("DefaultC
onnection"))); // Налаштування бази даних
            services.AddIdentity<ApplicationUser,
IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>()
                .AddDefaultTokenProviders(); // Налаштування
аутентифікації
        }

        public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {  endpoints.MapControllerRoute(
                    name: "default",
                    pattern:
"{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
```

4.3.3 Реалізація пошукової функції

Для реалізації пошуку товарів за назвою на сайті для користувача з використанням React на клієнтській частині та ASP.NET Core на серверній частині були виконані наступні кроки:

1. У «ProductsController.cs» додано метод пошуку до контролера.

```
 [Route("api/[controller]")]
 [ApiController]
public class ProductsController : ControllerBase
{
    // Поле для збереження контексту бази даних.
    private readonly AppDbContext _context;

    // Конструктор, який отримує контекст бази даних через залежність та зберігає його у приватне поле.
    public ProductsController(AppDbContext context)
    {
        _context = context;
    }
    [HttpGet("search")]

    public async Task<ActionResult<IEnumerable<Product>>> SearchProducts(string query)
    {
        // Перевірка, чи параметр запиту не є порожнім або нульовим.
        if (string.IsNullOrEmpty(query))
        {
            // Повертає HTTP 400 помилку, якщо параметр запиту відсутній.
            return BadRequest("Query parameter is required");
        }

        var products = await _context.Products
            .Where(p => p.Name.Contains(query))
            .ToListAsync();
        return Ok(products);
    }
}
```

Пояснення:

`[Route("api/[controller]")]`: вказує маршрут для цього контролера. [controller] є плейсхолдером, який замінюється на ім'я контролера, тобто products, що відповідає за маршрут api/products.

`[ApiController]`: вказує, що цей клас є контролером API. Цей атрибут додає кілька зручностей, таких як автоматична валідація моделі та автоматичні відповіді на помилки.

`private readonly AppDbContext _context;`: Декларація поля для збереження контексту бази даних.

`public ProductsController(AppDbContext context)`: Конструктор, який приймає контекст бази даних через залежність і зберігає його у приватне поле _context.

`[HttpGet("search")]`: Вказує, що цей метод обробляє HTTP GET запити до api/products/search.

`public async Task<ActionResult<IEnumerable<Product>>> SearchProducts(string query)`: Асинхронний метод, який повертає список продуктів. Метод приймає параметр query, який є строкою для пошуку.

`if (string.IsNullOrEmpty(query))`: Перевіряє, чи параметр query не є порожнім або нульовим.

`return BadRequest("Query parameter is required");` Повертає HTTP 400 (Bad Request) відповідь, якщо параметр query відсутній.

`var products = await _context.Products.Where(p => p.Name.Contains(query)).ToListAsync();` Виконує асинхронний запит до бази даних, щоб знайти продукти, назва яких містить query.

`return Ok(products);` Повертає знайдені продукти у відповідь з HTTP статусом 200 (OK).

2. В «Startup.cs» слід налаштувати необхідні служби:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<AppDbContext>(options                    =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultC
onnection")));
    services.AddCors(options =>
    {
        options.AddPolicy("AllowReactApp",
            builder => builder
                .WithOrigins("http://localhost:3000")
                .AllowAnyHeader()
                .AllowAnyMethod());    });
```

```
    services.AddControllers();
}

public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
   app.UseCors("AllowReactApp");
   app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Пояснення до коду:

`public void ConfigureServices(IServiceCollection services)`: це метод для налаштування служб додатка. Тут реєструються різні служби, які будуть використовуватися додатком.

```
services.AddDbContext<AppDbContext>(options=>
options.UseSqlServer(Configuration.GetConnectionString("DefaultC
onnection")));
```

Тут йде додавання контексту бази даних AppDbContext до служб додатка, налаштовуючи його для використання SQL Server. «Connection string» для підключення до бази даних береться з конфігураційних файлів.

```
services.AddCors(options => { ... });
```

Додає налаштування CORS (Cross-Origin Resource Sharing), що дозволяє додатку приймати запити з інших доменів.

```
services.AddControllers();
```

Додає підтримку контролерів до служб додатка. Це необхідно для обробки HTTP-запитів.

- `public void Configure(IApplicationBuilder app, IWebHostEnvironment env)`: Метод для налаштування конвеєра обробки запитів додатка.

4.3.4 Реалізація БД у PostgreSQL

Реалізація запропонованої бази даних на PostgreSQL виглядає наступним чином:

```
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    address TEXT,
    phone_number VARCHAR(20)
 );

CREATE TABLE Categories (
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT
);

CREATE TABLE Products (
    product_id SERIAL PRIMARY KEY,
    category_id INT REFERENCES Categories(category_id),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL,
    image_url TEXT,
    specifications TEXT
 );

CREATE TABLE Orders (
    order_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES Users(user_id),
    total_amount DECIMAL(10, 2) NOT NULL,
    order_date TIMESTAMP NOT NULL
);

CREATE TABLE OrderItems (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES Orders(order_id),
    product_id INT REFERENCES Products(product_id),
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

CREATE TABLE Reviews (
    review_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES Users(user_id),
    product_id INT REFERENCES Products(product_id),
```

```
    rating INT NOT NULL CHECK (rating >= 1 AND rating <= 5),
    comment TEXT,
    review_date TIMESTAMP NOT NULL
);

 CREATE TABLE Payments (
    payment_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES Orders(order_id),
    amount DECIMAL(10, 2) NOT NULL,
    payment_date TIMESTAMP NOT NULL
);

 CREATE TABLE Deliveries (
    delivery_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES Orders(order_id),
    delivery_date TIMESTAMP NOT NULL,
    status VARCHAR(50) NOT NULL
);
```

Тут «SERIAL»: тип даних, який автоматично генерує унікальні послідовні значення (автоінкрементні значення) для первинних ключів, а «TIMESTAMP»: тип даних, який зберігає дату та час. Ця структура бази даних забезпечує необхідні зв'язки та обмеження для управління користувачами, товарами, замовленнями, відгуками, платежами та доставками.

5 ТЕСТУВАННЯ

5.1 Функціональне тестування

Завершальний етап процесу розробки програмного забезпечення – тестування. Дана процедура грає найважливішу роль в створенні системи, так як саме від якості тестування залежить подальше життя програмного продукту та його успішність впровадження в рамках автоматизації діяльності підприємств транспортної логістики.

Тестування – це перевірка стану між реальною поведінкою програми та її очікуваною поведінкою на кінцевому наборі тестів, обраному певним чином. Тест – це спеціальна, штучно створена ситуація, обрана певним чином, і опис того, які спостереження за програмою потрібно зробити для перевірки її відповідності деяким вимогам [11].

Тесткейси – це документовані вказівки або скрипти, що описують послідовність дій для виконання певного тесту або набору тестів. Вони використовуються для створення, виконання та документування тестів програмного забезпечення. Основна мета тесткейсів – забезпечити повноту, правильність і систематичність в тестуванні програми.

Одними з методів тестування програмного продукту є методи чорного та білого ящиків. Тестування чорного ящика або поведінкове тестування – стратегія тестування функціональної поведінки об'єкта (програми, системи) з точки зору зовнішнього світу, при якому не використовується знання про внутрішній устрій об'єкта. Під стратегією розуміються систематичні методи відбору і створення тестів для тестового набору. Стратегія поведінкового тесту виходить з технічних вимог і їх специфікацій.

Тестування методом «білого ящика» – це тестування коду на предмет логіки роботи програми і коректності її роботи з точки зору компілятора тієї мови, на якої вона була розроблена. Стратегія тестування за принципом «білого ящика», дозволяє перевірити внутрішню структуру програми.

Виходячи з цієї стратегії спеціаліст отримує тестові дані шляхом аналізу логіки роботи програми [12].

Для функціонального тестування роботи інформаційної системи було написано 7 тест-кейсів, які охоплюють головні варіанти використання по роботі ПЗ.

Тест-кейс №1 «Реєстрація».

1. Передумови: Користувач заходить на сторінку реєстрації веб-сайту.
2. Кроки:
   - Користувач заповнює форму реєстрації, вводячи свої дані (ім'я, електронну пошту, пароль тощо);
   - Система перевіряє правильність введених даних та їх унікальність;
   - Користувач натискає кнопку «Зареєструватися».
3. Післяумови: Користувач отримує повідомлення про успішну реєстрацію, його дані збережено в базі даних, та йому надано доступ до закритих розділів сайту.

Тест-кейс №2 «Вхід у систему».

1. Передумови: Користувач заходить на сторінку входу веб-сайту.
2. Кроки:
   - Користувач вводить свої логін і пароль у відповідні поля форми;
   - Система перевіряє відповідність введених даних з тими, що зберігаються в базі даних;
   - Користувач натискає кнопку «Увійти».
3. Післяумови: Користувача успішно аутентифіковано, і він перенаправляється до свого особистого кабінету.

Тест-кейс №3 «Перегляд продукції».

1. Передумови: Користувач заходить на головну сторінку веб-сайту.
2. Кроки:
   - Користувач переходить до розділу «Продукція» або використовує пошук для знаходження конкретного товару;

- Система відображає список категорій продукції або результати пошуку;
- Користувач обирає категорію або конкретний товар для детального перегляду;
- Система відображає детальну інформацію про обраний товар, включаючи опис, зображення, ціну, наявність на складі тощо.

3. Післяумови: Користувач бачить детальну інформацію про продукцію, яку він обрав, та може прийняти рішення щодо подальших дій (наприклад, додавання товару в кошик або повернення до списку товарів).

Тест-кейс №4 «Пошук товару».

1. Передумови: Користувач заходить на головну сторінку веб-сайту.
2. Кроки:
   - Користувач вводить ключові слова у пошукове поле (наприклад, назву товару, категорію);
   - Система обробляє пошуковий запит і виконує пошук у базі даних товарів;
   - Система відображає результати пошуку у вигляді списку товарів, що відповідають запиту.
3. Післяумови: Користувач бачить список товарів, які відповідають його пошуковому запиту, і може обрати товар для перегляду детальної інформації.

Тест-кейс №5 «Перегляд товару».

1. Передумови: Користувач здійснив пошук товару або обрав товар зі списку категорій.
2. Кроки:
   - Користувач натискає на товар зі списку результатів пошуку або категорій для перегляду детальної інформації;

- Система відображає сторінку з детальною інформацією про обраний товар, включаючи опис, зображення, ціну, наявність на складі, відгуки тощо.

3. Післяумови: Користувач має повну інформацію про товар і може прийняти рішення щодо подальших дій, таких як покупка або додавання до списку бажань.

Ці передумови та післяумови забезпечують зручний та ефективний процес пошуку і перегляду товарів для користувачів веб-сайту підприємства.

Тест-кейс №6 «Оформлення замовлення».

1. Передумови: Користувач додав товари до кошика та бажає оформити замовлення.
2. Кроки:
   - Користувач переходить до кошика та натискає кнопку «Оформити замовлення»;
   - Система відображає форму для введення даних про доставку та оплату;
   - Користувач вводить необхідну інформацію (адреса доставки, контактні дані, спосіб оплати);
   - Система перевіряє правильність введених даних та доступність товарів;
   - Користувач підтверджує замовлення та здійснює оплату (якщо оплата здійснюється під час оформлення замовлення).
3. Післяумови:
   - Система створює нове замовлення, зберігає його у базі даних та відображає підтвердження замовлення на екрані;
   - Користувач отримує підтвердження про успішне оформлення замовлення на свою електронну пошту;
   - Платіжна система обробляє оплату (якщо застосовується);
   - Користувач отримує інформацію про терміни та умови доставки;

- Система оновлює статус замовлення до «Обробка» або іншого відповідного статусу і повідомляє користувача про подальші кроки.

Тест-кейс №7 «Зворотній зв'язок»

1. Передумови: Користувач хоче залишити зворотній зв'язок про товар або сервіс підприємства.
2. Кроки:
   - Користувач переходить до розділу «Зворотній зв'язок» або «Контакти» на веб-сайті;
   - Користувач заповнює форму зворотнього зв'язку, вводячи необхідні дані (ім'я, контактну інформацію, повідомлення);
   - Користувач натискає кнопку «Відправити»;
   - Система перевіряє правильність введених даних;
   - Система зберігає повідомлення користувача в базі даних.
3. Післяумови:
   - Користувач отримує підтвердження про успішне відправлення повідомлення (наприклад, повідомлення на екрані або електронний лист);
   - Повідомлення передано відповідальним працівникам для обробки (наприклад, до служби підтримки або відповідному менеджеру);
   - Система зберігає повідомлення для подальшого аналізу, щоб поліпшити обслуговування клієнтів та якість продукції або послуг.

Ці передумови та післяумови забезпечують ефективний і зручний процес надання зворотнього зв'язку для користувачів веб-сайту підприємства.

5.2 Результати тестування

Проведено тестування інформаційної системи відповідно до створених тест-кейсів. Усі функціональні вимоги були виконані. У таблиці 5.1 представлені результуючі дані щодо кожного з сценаріїв тестування.

Таблиця 5.1 – Результати тестування

№ Тест-кейсу	Назва тест-кейсу	Підтвердження очікуваного результату
1	Реєстрація	так
2	Вхід у систему	так
3	Перегляд продукції	так
4	Пошук товару	так
5	Перегляд товару	так
6	Оформлення замовлення	так
7	Зворотній зв'язок	так

В даному розділі описано функціональне тестування інформаційної системи. Для цього етапу розробки ПЗ було створено сім тест-кейсів, які описують основні варіанти використання для об’єкту розробки. Результати тестування демонструють гарну роботу веб-сайту по продажу продукції комерційного підприємства.

ВИСНОВКИ

Розробка та впровадження інформаційної системи для підтримки комерційної діяльності підприємства, представленої у вигляді веб-сайту для продажу товарів, має кілька ключових висновків та переваг:

1. Підвищення ефективності операцій: автоматизація процесів, таких як обробка замовлень і платежів, значно знижує час і зусилля, необхідні для виконання цих завдань вручну.
2. Поліпшення клієнтського досвіду: інтуїтивно зрозумілий і зручний інтерфейс веб-сайту, що дозволяє клієнтам легко знаходити та купувати товари, а також отримувати підтримку через зворотний зв'язок, підвищує задоволеність клієнтів.
3. Кращий аналіз даних: система зберігає детальну інформацію про замовлення, продажі та відгуки клієнтів, що дозволяє підприємству проводити глибокий аналіз і приймати обґрунтовані рішення на основі даних.
4. Збільшення доходів: завдяки зручності для клієнтів і швидшій обробці замовлень, підприємство може очікувати збільшення кількості продажів і, відповідно, доходів.
5. Гнучкість і масштабованість: архітектура системи, побудована з використанням сучасних фреймворків і технологій, таких як ASP.NET Core для серверної частини та React для клієнтської частини, дозволяє легко масштабувати систему в міру зростання бізнесу.

Під час виконання кваліфікаційної роботи були пройдені всі необхідні етапи створення інформаційної системи для підприємства. Поставлена мета роботи була досягнута, а саме розроблена інформаційної системи у вигляді веб-сайту, яка допоможе автоматизувати бізнес-процеси, покращити взаємодію з клієнтами та підвищити ефективність операцій, що в кінцевому підсумку сприятиме збільшенню доходів підприємства.

Для цього був виконаний аналіз предметної області і існуючих сучасних рішень – це допомогло поставити вимоги до об’єкту розробки. Складено діаграму варіантів використання системи з описом акторів і прецедентів. Були виконані етапи проектування, реалізації та кінцевого тестування системи. При розробці була обрана мікросервісна архітектура системи.

Проведено попереднє тестування ІС, розроблені необхідні для цього сценарії тестування та перевірені всі заявлені сценарії роботи системи, її функціоналу.

Як подальший розвиток слід завершити розробку кабінету адміністратору, що дозволить надати йому інструменти для управління контентом, користувачами, замовленнями та іншими аспектами веб-сайту. Додати відображення контенту українською мовою. Крім цього, слід забезпечити можливість прийому онлайн-платежів на сайті.

Все це дозволить забезпечити більш повний контроль за бізнес-процесами, підвищити безпеку та зручність для користувачів, а також покращити загальну ефективність та конкурентоспроможність підприємства.

СПИСОК ДЖЕРЕЛ ПОСИЛАНЬ

1. How A Modern Requirements Management Tool Enables Engineering Teams Achieve Project Success. URL: https://visuresolutions.com/uk/% (дата звернення 05.03.2024)
2. What Are Nonfunctional Requirements and How Do They Impact Product Development? URL: https://www.jamasoftware.com/requirements-management-guide/writing-requirements/how-non-functional-requirements-impact-product-development (дата звернення 07.03.2024)
3. Gantt Chart Online Maker Software. URL: https://www.instagantt.com/ (дата звернення 14.03.2024)
4. What are microservices? URL: https://microservices.io/ (дата звернення 04.04.2024)
5. React. The library for web and native user interfaces. URL: https://react.dev/ (дата звернення 17.04.2024)
6. React Introduction. URL: https://www.geeksforgeeks.org/reactjs-introduction/ (дата звернення 28.04.2024)
7. What is Microsoft SQL Server? URL: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server (дата звернення 03.05.2024)
8. What is MySQL? URL: https://www.oracle.com/mysql/what-is-mysql/ (дата звернення 03.05.2024)
9. PostgreSQL: The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/ (дата звернення 03.05.2024)
10. Одноразовий пароль та інші алгоритми аутентифікації: як вони використовуються в цифровій безпеці. URL: https://introserv.com/ua/blog/one-time-password-and-other-authentication-algorithms-how-are-they-used-in-digital-security/ (дата звернення 18.05.2024)

11. Різниця між функціональним і нефункціональним тестуванням. URL: https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/ (дата звернення 18.05.2024)
12. What is Functional Testing? URL: https://www.opentext.com/what-is/functional-testing (дата звернення 18.05.2024)