Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр післядипломної освіти

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система керування продажами інтернет-магазину
(тема)

Виконав:
студент 4 курсу, групи ПЗПІп-22-2

Буряк І.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Лановий О.Ф.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Буряку Ігорю Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система керування продажами інтернет-магазину

Затверджена наказом по університету від №588 Ст від 17.06.24

2. Термін подання студентом роботи до екзаменаційної комісії 14.07.2024

3. Вихідні дані до роботи

Розробити програмну систему керування продажами інтернет-магазину, котра дозволятиме керувати товарами, додавати їх в корзину, оплачувати замовлення кредитною карткою і змінювати статуси замовлень, мовою програмування TypeScript.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2024	*виконано*
2	Створення специфікації ПЗ	25.05.2024	*виконано*
3	Проектування ПЗ	01.06.2024	*виконано*
4	Розробка ПЗ	10.06.2024	*виконано*
5	Тестування ПЗ	30.06.2024	*виконано*
6	Оформлення пояснювальної записки	01.07.2024	*виконано*
7	Підготовка презентації та доповіді	06.07.2024	*виконано*
8	Попередній захист	14.07.2024	*виконано*
9	Нормоконтроль, рецензування	15.07.2024	*виконано*
10	Здача роботи у електронний архів	16.07.2024	*виконано*
11	Допуск до захисту у зав. кафедри	17.07.2024	*виконано*

Дата видачі завдання 16 травня 2024р.

Студент (ка)_____ Буряк І.О.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Лановий О.Ф.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 49 стор., 14 рис., 1 табл, 7 джерел.

ЕЛЕКТРОННА КОММЕРЦІЯ, ПРОГРАМНИЙ ЗАСТОСУНОК, HEADLESS, JETBRAINS WEBSTORM, MACH, MONGODB, NODEJS, TYPESCRIPT

Об’єкт розробки – програмна система керування продажами інтернет-магазину.

Мета розробки – створення програмного застосунку, який дозволятиме придбати товари в інтернет-магазині та керувати продажами.

Метод рішення – середовище розробки JetBrains Webstorm, мова програмування TypeScript.

У результаті розробки створено програмний web-застосунок, який дозволяє покласти товари в корзину, сплатити за товари кредитною карткою, переглядати замовлення змінювати їх статуси.

ECOMMERCE, SOFTWARE APPLICATION, HEADLESS, JETBRAINS WEBSTORM, MACH, MONGODB, NODEJS, TYPESCRIPT

The object of development is a program system for the ecommerce sales management.

The purpose of the work is to develop software web-application that allows to buy items online and manage sales.

Solution method – JetBrains Webstorm development environment, TypeScript programming language.

As a result of the development, a software web-application has been created, that allows to put items to the cart, pay with credit card, review the orders list and manage order statuses.

Я, Буряк Ігор Олександрович, студент гр. ПЗПІп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система керування продажами інтернет-магазину», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 8
ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 10
1.1 Аналіз предметної галузі 10
1.2 Виявлення та вирішення проблем 11
1.3 Постановка задачі 11
2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 12
2.1 Загальні вимоги 12
2.2 Загальні обмеження 12
2.3 Припущення і залежності 12
2.4 Вимоги до зовнішніх інтерфейсів 13
2.4.1 Інтерфейс користувача 13
2.4.2 Програмний інтерфейс 14
2.5 Вимоги до бази даних 16
3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ 17
3.1 UML проєктування ПЗ 17
3.2 Проєктування архітектури ПЗ 18
3.3 Проєктування структури зберігання даних 25
3.4 Створення UI/UX 27
4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ 33
4.2 Інтеграція з пошуковим сервісом 35
5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 37

5.1 Тестування програмного застосунку 37
5.2 Приклади критичних помилок..... 37
6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ..... 39
6.1 Визначення плану впровадження 39
ВИСНОВКИ..... 40
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 41
ДОДАТОК А..... 42
ДОДАТОК Б 43

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

MACH – Microservices, API-first, Cloud native ,Headless

HTTP – Hypertext Transfer Protocol

PCI DSS – Payment Card Industry Data Security Standard

EDA – Event-Diven Architecture

ВСТУП

Темою кваліфікаційної роботи є програмний веб-застосунок для керування продажами інтернет-магазину. Основне завдання програмного веб-застосунку – надати можливість купити товари через браузер, оплатити їх кредитною карткою та керувати продажами. Зроблено детальний аналіз предметної галузі, спираючись на власний досвід та доступні друковані матеріали, в якому описано еволюцію інтернет-продажів в Україні на протязі останніх 25 років. Також було розглянуто основні недоліки існуючих аналогічних програмних продуктів, саме ці недоліки я спробував усунути в розробці програмної систми керування продажами інтернет-магазину.

Метою роботи є розробка програмної системи, яка складається з серверної частини та фронтенд-додатку на ReactJS. Для розробки продукту використовувався фреймворк Payload та мова програмування TypeScript. Для написання коду використовується середовище розробки JetBrains Webstorm. В межах підготовки кваліфікаційної роботи я застосував декілька архітектур програмного забезбпечення – мікросервісну в якості головної аріхітектури застосунку, та подійно-орієнтовану для інтеграції розробленої програмної системи з пошуковим сервісом. Також було детально розглянуто безпековий стандарт PCI DSS в контексті інтеграції платіжних сервісів в програмні веб-застосунки.

Заважаючи на особливості проектування сутностей в фреймворку Payload, вдалося зберегти час на розробку саме адміністративної системи керування продажами, і таким чином проробити клієнтську частину розміщення замовлення більш детально.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Розробка інтернет-магазинів - це сфера, що постійно розвивається і змінюється відповідно до потреб ринку та технологічних тенденцій.

25 років тому продажі через інтернет полягали в тому, що на електронну пошту можна було отримати прайс-лист від продавця і далі також на електронну пошту відправити перелік товарів, які цікавлять. Для оплати товару треба було йти в банк, і потім також електронною поштою пересилати підтвердження платежу. Все це займало дуже багато часу, і все ж таки надавало зовсім інший рівень можливостей покупки товарів. Порівнювати ціни різних постачальників та аналізувати асортимент вдома було значно зручніше, ніж фізично відвідувати магазини.

Наступним кроком була розробка сайтів-каталогів. В цей час вже не потрібно було робити запити доступного асортименту – компанії розробляли свої сайти, де додавали наявний асортимент продукції. Таким чином вдавалося зберегти багато часу, але розміщення замовлень та оплата все ще займала багато часу і була не зручною.

В Україні можливість здійснювати продажі через інтернет з'явилася приблизно 20 років тому, коли масово почали з'являтися інтернет-магазини з повним функціоналом покупки. Популярність інтернет-магазинів в Україні зростала з появою швидких та дешевих способів доставки товарів покупцю.

Після того, як всі зрозуміли що продажі через інтернет це дуже потужний канал продажів, індустрія почала розвиватися значно швидше. З плином часу виникали різні проблеми, для яких індустрія інформаційних технологій мала знайти технологічні та ефективні відповіді та рішення.

1.2 Виявлення та вирішення проблем

Наразі існує дуже багато технічних рішень для організації продажів товарів та послуг через інтернет, але ідеального рішення поки що не існує.

Багато компаній не готові платити багато грошей за ліцензії на готові програмні продукти, а безкоштовні версії не відповідають вимогам функціоналу, безпеки та мають складності з масштабуванням.

Додатковою проблемою є те, що багато безкоштовних програмних продуктів здобули популярність в часи, коли головною мовою програмування веб-застосунків був PHP. Наразі популярність цієї мови програмування падає з року в рік, що в решті решт може привести до спрацювання додаткових ризиків бізнесу по можливості технічної підтримки програмного рішення в наступні роки.

З точки зору організації команд розробки та складання оптимального бюджету бажано щоб всі сервіси, з яких складається програмний продукт, були створені з використанням мінімуму різних мов програмування. Таким чином, розробники в команді можуть бути взаємозамінні і будуть краще розуміти програмний продукт в цілому.

Останні роки багато уваги прикуто до персональних даних та їх безпечного зберігання. В контексті продажів через інтернет це дуже актуально, тому що кожен продаж містить персональні дані покупця, і ці дані мають зберігатися згідно відповідного законодавства.

1.3 Постановка задачі

В межах даної роботи пропонується розробити програмну систему керування продажами інтернет-магазину, яка дозволить обрати товари з каталогу, оплатити їх кредитною карткою та в подальшому мати можливість змінювати замовлення.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальні вимоги

Програмна система керування продажами інтернет-магазину має забезпечувати можливість для адміністратора магазину керувати інформацією на всіх сторінках, наповнювати каталог товарів та обробляти замовлення [1].

Користувач магазину повинен мати можливість переглядати інформацію на сайті через браузер, додати потрібні товари в корзину, оплатити замовлення кредитною карткою та розмістити його в системі.

2.2 Загальні обмеження

Зважаючи на вимоги до розробки, були розглянуті наступні обмеження:

- всі компоненти програмної системи мають бути запрограмовані з використанням JavaScript/TypeScript;
- передбачити можливість розміщення даних програмної системи в межах України.

2.3 Припущення і залежності

Допускається використання та залежність від API чи сервісів третіх сторін для функцій, таких як опрацювання платежів, аналітика або інтеграція з соціальними мережами.

2.4 Вимоги до зовнішніх інтерфейсів

2.4.1 Інтерфейс користувача

Зрозумілий і зручний інтерфейс значно підвищує комфортність в роботі із програмною системою. Головні вимоги по інтерфейсу для різних типів користувачів наведені нижче:

Зручність використання:

- інтуїтивно зрозуміле і легко навіговане меню для користувачів різних ролей (адміністратори, клієнти);
- простий та зрозумілий процес оформлення замовлення для клієнтів без зайвих кроків.

Адаптивний дизайн:

- підтримка різних типів пристроїв (комп'ютери, планшети, смартфони) для зручного перегляду та використання сайту;
- адаптивний дизайн, який змінюється в залежності від розміру екрану для оптимального відображення контенту.

Доступність інформації:

- чітка та зрозуміла інформація про товари (опис, ціна, наявність тощо).

Управління замовленнями:

- зручний інтерфейс для адміністраторів магазину для обробки та відстеження замовлень;
- система сповіщень або повідомлень для клієнтів про статус їхніх замовлень.

Безпека і конфіденційність:

- захист особистих даних користувачів та платіжної інформації;
- наявність системи аутентифікації та авторизації для доступу до різних функцій в залежності від ролі користувача.

Аналітика та звіти:

- можливість перегляду аналітичної інформації та звітів про продажі для адміністраторів магазину;
- відображення ключових метрик та статистики для прийняття управлінських рішень.

2.4.2 Програмний інтерфейс

Зважаючи на MACH архітектуру [2], вся взаємодія між мікросервісами відбувається через REST API.

Автентифікація та безпека:

- аутентифікація: API повинен забезпечувати методи для аутентифікації користувачів та отримання токенів доступу;
- авторизація: різні ролі користувачів (адміністратори, клієнти) повинні мати відповідний доступ до функціоналу API згідно їхніх прав доступу.

Операції з товарами та категоріями:

- отримання списку товарів: метод для отримання списку товарів з різними параметрами фільтрації та сортування;
- деталі товару: запит для отримання детальної інформації про конкретний товар за його ідентифікатором.

Управління замовленнями:

- отримання списку замовлень: метод для отримання списку замовлень з можливістю фільтрації за різними критеріями;
- деталі замовлення: запит для отримання детальної інформації про конкретне замовлення за його ідентифікатором;
- оновлення статусу замовлення: метод для зміни статусу замовлення після оплати чи доставки.

Операції з користувачами:

- реєстрація та аутентифікація користувачів: методи для реєстрації нових користувачів та аутентифікації існуючих;
- управління особистими даними: можливість зміни особистих даних користувача (адреса доставки, контактні дані тощо).

Платіжні операції:

- оформлення платежів: метод для оформлення платежів за замовлення з використанням різних платіжних систем;
- підтвердження платежів: можливість підтвердження успішно проведених платежів та оновлення статусу замовлень.

Аналітика та звіти:

- запити для аналітики: методи для отримання статистичної інформації про продажі, популярність товарів тощо;
- генерація звітів: можливість генерації звітів з різними параметрами для управлінського аналізу.

2.5 Вимоги до бази даних

В розробці бажано використовувати нереляційну базу даних – MongoDB або аналогічну. Дозволяється використовувати або хмарний сервіс, або локальну базу даних. При роботі з базою даних рекомендовано використовувати програмний інтерфейс, який реалізує концепцію ODM (Object-Document Mapper). Це дозволить не прив’язуватися до конкретного типу документо-орієнтованої бази даних, і в майбутньому за необхідності змінити тип бази даних на інший.

У випадку якщо використовується MongoDB, в якості інтерфейсу роботи за базою даних потрібно використовувати бібліотеку Mongoose. При розгортанні програмного додатку важливо передбачити механізм створення структур даних для початку роботи програмної системи.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ

3.1 UML проєктування ПЗ

Програмна система передбачає 2 ролі – адміністратор магазину і покупець. Відповідно до ролі, користувачі мають доступ до різного функціоналу системи. Доступні функції описані за допомогою рисунку 3.1.

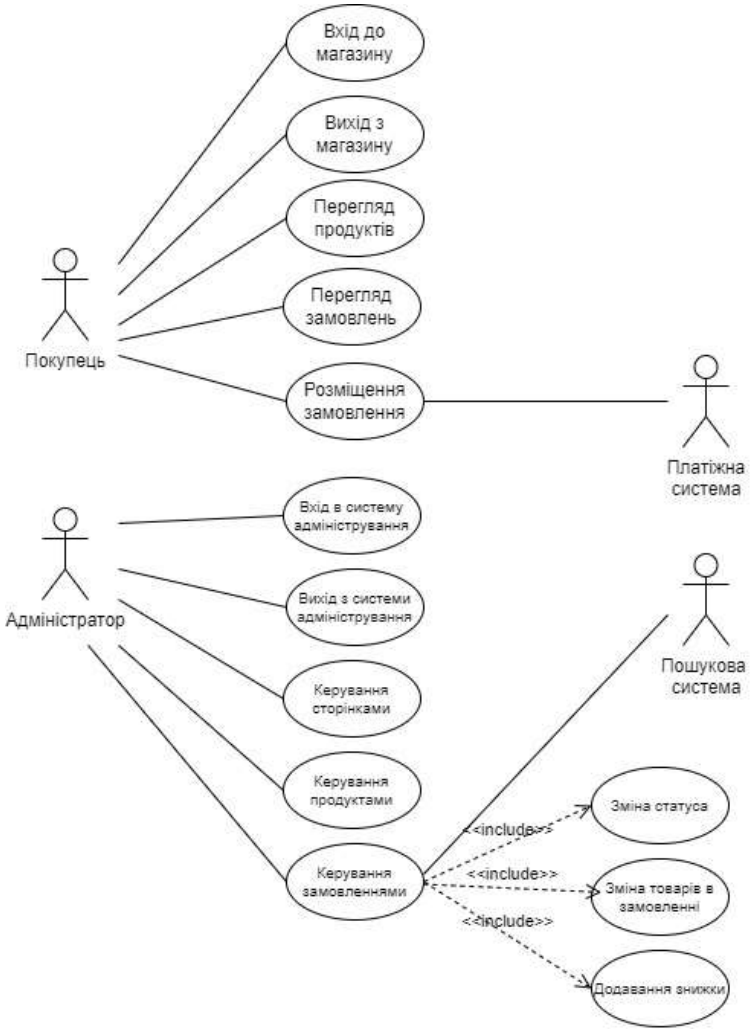


Рисунок 3.1 – Загальна Use-case діаграма програмної системи, показує різні ролі та можливості використання системи для них (рисунок виконаний самостійно)

3.2 Проєктування архітектури ПЗ

В якості центральної архітектури була обрана мікросервісна архітектура, а саме MACH-парадігма. Головна ідея цього підходу полягає в використанні багатьох невеликих сервісів, які можуть взаємодіяти між собою. Акронім MACH включає в себе наступні компоненти:

- m – microservices, головні будувальні блоки програмної системи;
- a – API-first, підхід до проектування мікросервісів, який передбачає наявність у кожного з мікросервісів програмного інтерфейсу керування;
- c – cloud-native SaaS, можливість розміщення компонентів програмної системі в хмарній інфраструктурі;
- h – headless, відокремлення клієнтської частини програмної системи від серверної, яке прибирає будь-які обмеження в розробці інтерфейсів користувача та дозволяє проектувати системи з декількома типами клієнтів (наприклад, веб-клієнт та мобільний додаток).

Таким чином, цей підхід пропонує не лише розбивати більш складну систему на менші сервіси з відокремленими клієнтами, але й передбачає використання готових хмарних мікросервісів для окремих специфічних задач (наприклад, пошук, оплата товарів, тощо). Для розробників це дозволяє значно прискорити розробку програмної системи, тому що при наявності вже готових сервісів їх можна інтегрувати в програмний продукт без необхідності їх безпосередньої розробки. Для бізнесу цей підхід дозволяє зменшити витрати на розробку та убезпечити себе від зайвої залежності від конкретної компанії-розробника.

Розглянемо детальніше головні компоненти програмної системи керування продажами інтернет-магазину (рисунок 3.2).

Рисунок 3.2 – Архітектурна діаграма програмної системи (рисунок виконано самостійно)

Серверна частина інтернет-магазину була реалізована на базі фреймворка Payload. Нижче наведені його головні характеристики:

API-first:

- GraphQL і REST API: Payload надає можливість доступу до контенту через GraphQL та REST API, що дозволяє легко інтегрувати CMS з іншими додатками і сервісами;
- зручність використання API: проектування API на першому етапі забезпечує чітке визначення структур даних і взаємодії між компонентами.

Гнучкість та розширюваність:

- конфігурований інтерфейс: інтерфейс користувача можна налаштовувати під специфічні потреби проекту;
- підтримка кастомних плагінів: можливість створення та інтеграції власних плагінів для розширення функціональності системи.

Безпека:

- автентифікація та авторизація: вбудована підтримка різних методів автентифікації та гнучка система управління ролями та правами доступу;
- захист даних: забезпечує високий рівень безпеки для захисту даних користувачів і контенту.

Хмарна підтримка:

- підтримка розгортання в хмарі: легко інтегрується з різними хмарними платформами для зберігання і обробки даних;
- масштабованість: забезпечує можливість масштабування відповідно до зростаючих потреб проекту.

Гнучкий редактор контенту:

- багатокористувацька підтримка: підтримує спільну роботу над контентом з можливістю відстеження змін і управління версіями;
- підтримка мультимедіа: легко працює з різними типами контенту, включаючи зображення, відео та інші мультимедійні файли.

Клієнтська частина була розроблена з використанням фреймворку Next.js. Next.js є потужним інструментом для створення сучасних веб-додатків з React. Завдяки серверному рендерингу, статичній генерації та іншим можливостям, Next.js забезпечує високу продуктивність, покращене SEO та зручний процес розробки. Це робить його відмінним вибором для створення різноманітних веб-проектів – від простих статичних сайтів до складних веб-додатків.

В якості пошукового сервісу я обрав Typesense. Typesense є потужним та гнучким інструментом для реалізації пошуку в додатках, надаючи розробникам

швидкий, безпечний та легко налаштовуваний пошуковий сервіс. Його висока продуктивність, типобезпечність та розширена функціональність роблять його ідеальним вибором для широкого спектра застосувань — від невеликих проектів до масштабних систем з високими вимогами до пошуку.

Вибір Stripe як платіжної системи для інтернет-магазину забезпечує високий рівень безпеки, гнучкість налаштувань, підтримку багатьох методів оплати та валют, а також простоту інтеграції. Ці переваги роблять Stripe відмінним вибором для будь-якого інтернет-магазину, який прагне забезпечити зручність та надійність обробки платежів для своїх клієнтів. Додатково Stripe надає необхідні бібліотеки, які дозволяють організувати взаємодію між інфраструктурою платіжної системи і клієнта без необхідності проходження відповідності PCI DSS.

Тепер розглянемо більш детально процес розміщення замовлення більш детально (рисунок 3.3). Покупка починається з додавання товару в корзину. На цьому етапі має відбуватися перевірка наявності товару. Покупець може додавати декілька товарів в корзину і потім здійснювати розміщення замовлення.

Для завершення процедури оформлення замовлення потрібно перевірити, чи всі продукти в корзині мають відповідники в платіжній системі. Це потрібно для правильної роботи платіжної інтеграції.

Якщо всі товари зконфігуровані правильно, покупець бачить форму для введення даних кредитної картки і моє можливість оплатити покупку. В результаті клієнтський додаток відобразить номер замовлення.

Коли статус транзакції в платіжній системі зміниться, вона надсилає запит на зареєстрований вебхук, таким чином асинхронно сповіщуючи програмну систему про зміну стану замовлення.

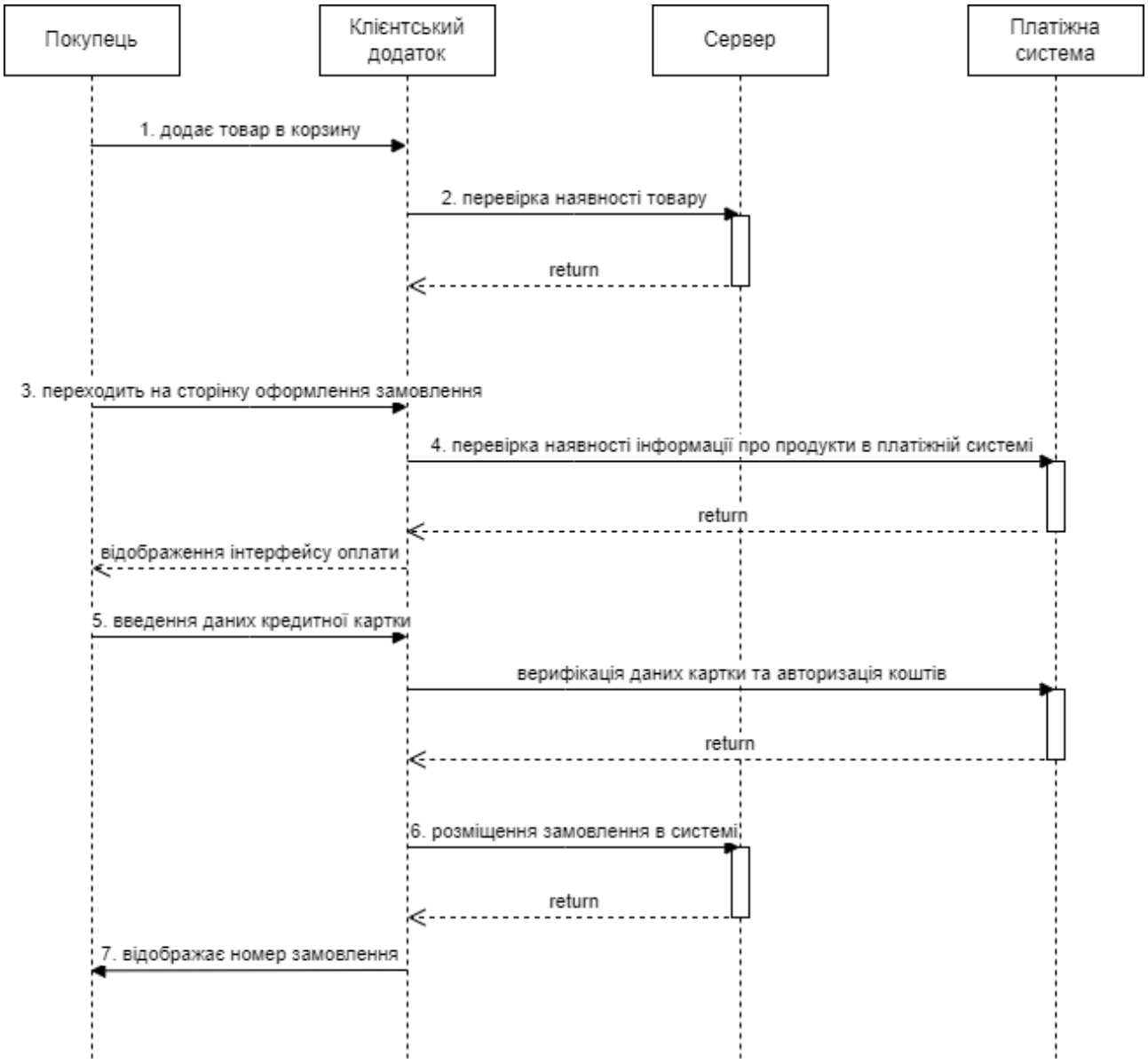


Рисунок 3.3 – Діаграмма послідовностей процессу розміщення замовлення (рисунок виконано самостійно)

PCI DSS (Payment Card Industry Data Security Standard) — це набір стандартів безпеки, розроблених для забезпечення безпеки даних платіжних карток і захисту від шахрайства [3]. Цей стандарт був створений Радою стандартів безпеки платіжних карток (PCI SSC), до складу якої входять основні платіжні системи, такі як Visa, MasterCard, American Express, Discover та JCB.

Основні цілі PCI DSS:

- захистити дані платіжних карток: забезпечити надійний захист даних власників карток від несанкціонованого доступу;
- запобігти шахрайству та витоку даних: зменшити ризик шахрайства з платіжними картками шляхом впровадження жорстких вимог безпеки;
- підтримувати довіру клієнтів: забезпечити споживачам впевненість у безпеці їхніх платіжних даних під час транзакцій.

Вимоги PCI DSS:

Встановити та підтримувати конфігурацію захищеного мережевого середовища:

- встановлення та підтримка конфігурації міжмережевого екрану для захисту даних власників карток;
- не використовувати паролі за замовчуванням та інші параметри безпеки, що надаються постачальниками.

Захистити дані власників карток:

- захист збережених даних власників карток;
- шифрувати передавання даних власників карток через відкриті публічні мережі.

Підтримувати програмне забезпечення з поточними оновленнями:

- використовувати та регулярно оновлювати антивірусне програмне забезпечення або програми для захисту від шкідливого ПЗ.

Контроль доступу до даних власників карток:

– обмежити доступ до даних власників карток за принципом "необхідності знати";
– впровадити унікальні ідентифікатори для кожного користувача з доступом до комп'ютерів.

Регулярний моніторинг та тестування мереж:

– відстежувати та контролювати всі доступи до мережевих ресурсів та даних власників карток;
– регулярно тестувати системи безпеки та процеси.

Забезпечення політики інформаційної безпеки:

– забезпечити створення, підтримку та дотримання політики інформаційної безпеки для всіх співробітників.

Зважаючи на особливості запропонованої архітектури, дані кредитних карток передаються безпосередньо між браузером покупця і платіжною системою. Таким чином для даного проекту доводити відповідність PCI DSS не обов’язково.

3.3 Проєктування структури зберігання даних

Розглянемо діаграму класів (рис. 3.4) програмної системи керування продажами інтернет-магазину.

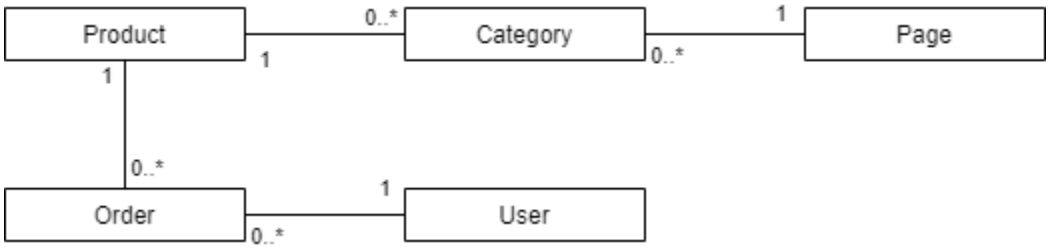


Рисунок 3.4 – Діаграма класів програмної системи керування продажами інтернет-магазину (рисунок виконано самостійно)

Продукти можуть бути об’єднані у категорії. Сторінки дозволяють додавати туди різний контент, в тому числі перелік продуктів за категорій. Замовлення складаються з продуктів, кожен продукт може бути придбаний декілька разів. Кожне замовлення має користувача.

Фреймворк Payload дозволяє описати класи через певну конфігурацію (рис. 3.5), і таким чином пропонує абстракцію над рівнем зберігання даних [4]. В результаті автоматично генерується код який дозволяє додавати та змінювати дані через систему адміністрування даних.

Для цього проекту було обрано нереляційну базу даних MongoDB [5]. Головні сутності проекту – це складні веб-сторінки, які треба мати можливість гнучко конфігурувати. Використання нереляційної бази даних допомогло спростити модель даних, пришвидшити читання з бази даних та таким чином пришвидшити роботу програмної системи.

В якості програмного інтерфейса для роботи з базою даних використовується бібліотека Mongoose [6].

```
export const Orders: CollectionConfig = {
  slug: 'orders',
  hooks: {
    afterChange: [updateUserPurchases, clearUserCart],
  },
  fields: [
    {
      name: 'orderedBy',
      type: 'relationship',
      relationTo: 'users',
      hooks: {
        beforeChange: [populateOrderedBy],
      },
    },
    {
      name: 'stripePaymentIntentID',
      label: 'Stripe Payment Intent ID',
      type: 'text',
      admin: {
        position: 'sidebar',
        components: {
          Field: LinkToPaymentIntent,
        },
      },
    },
    {
      name: 'total',
      type: 'number',
      required: true,
      min: 0,
    },
    {
      name: 'status',
      type: 'select', // required
      hasMany: false,
      defaultValue: 'new',
      options: [
        {
          label: 'New',
          value: 'new',
        },
        {
          label: 'In progress',
          value: 'inProgress',
        },
        {
          label: 'Closed',
          value: 'closed',
        },
      ],
    },
    {
      name: 'items',
      type: 'array',
      fields: [
        {
          name: 'product',
          type: 'relationship',
          relationTo: 'products',
          required: true,
        },
        {
          name: 'price',
          type: 'number',
          min: 0,
        },
        {
          name: 'quantity',
          type: 'number',
          min: 0,
        },
      ],
    },
  ],
}
```

Рисунок 3.5 – Приклад конфігурації сутності за допомогою фреймворка Payload (рисунок виконано самостійно)

3.4 Створення UI/UX

Розглянемо процес розміщення замовлення в інтернет-магазині. Згідно з мапою сценарію користувача (рис. 3.6) спочатку користувач переходить на головну сторінку інтернет-магазину. На цій сторінці (рис. 3.7) він бачить рекламний блок, декілька блоків з продуктами (новинки та найбільш популярні товари) та загальний текст про інтернет-магазин.

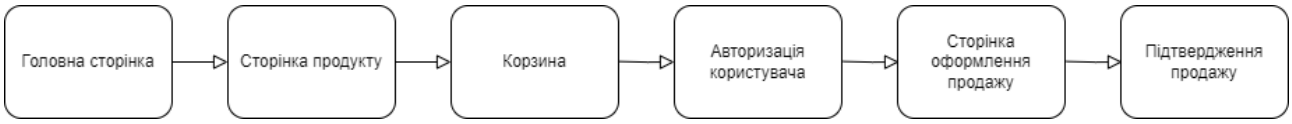


Рисунок 3.6 – Мапа сценарію користувача з процессу розміщення замовлення (рисунок виконано самостійно)

Обравши продукт, який зацікавив користувача, він може перейти на сторінку з детальним описом продукта, натиснувши на його назву або зображення.

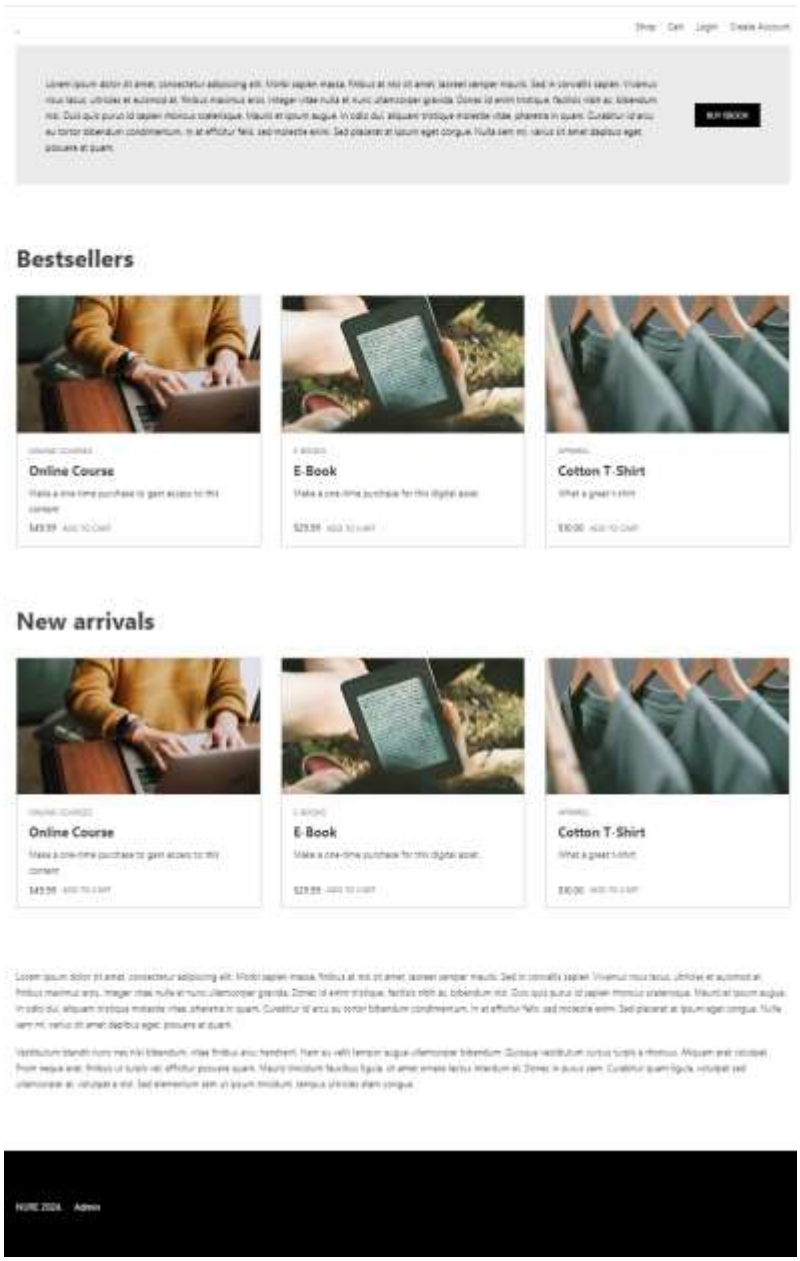


Рисунок 3.7 – Макет головної сторінки інтернет-магазину (рисунок виконано самостійно)

На сторінці продукта (рис. 3.6) користувач може подивитись фото продукта більш детально, та ознайомитись з додатковою інформацією про товар, ціною та наявністю товару. Особливістю цієї сторінки є те, що в панелі адміністрування можна гнучко оперувати різними блоками з контентом, розміщуючи їх в будь-якому порядку. Це надає можливість відобразити

максимум інформації в зручній формі, що спонукає користувача додати товар в корзину. Для того щоб додати товар в корзину, користувач має натиснути відповідну кнопку біля вартості товару.

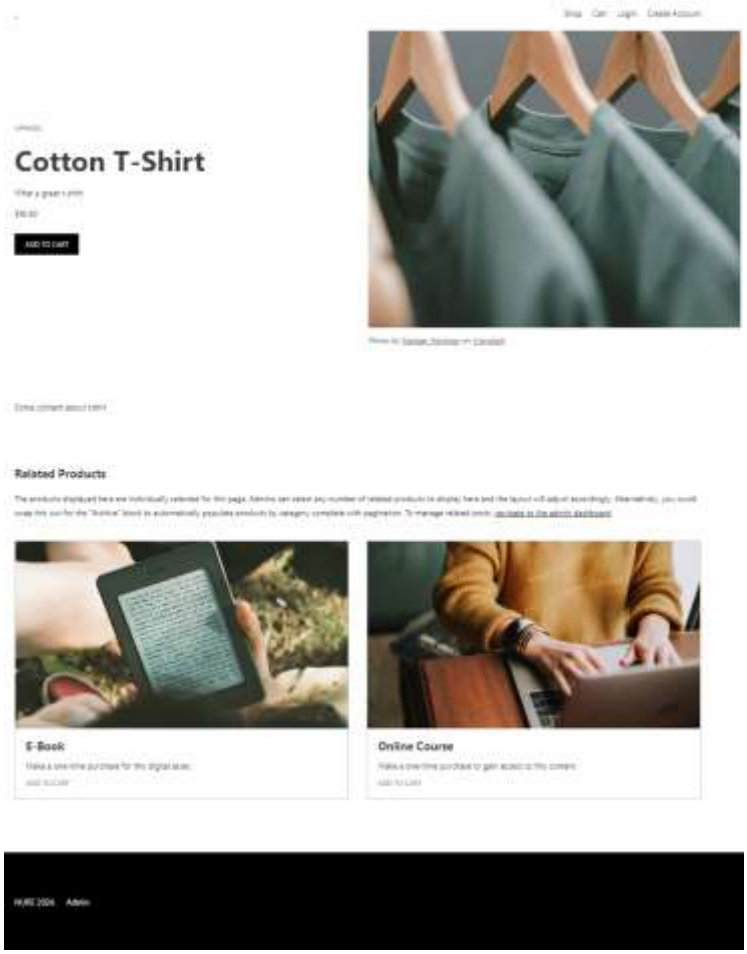


Рисунок 3.8 – Макет сторінки з детальним описом продукта (рисунок виконано самостійно)

Далі користувач переадресується на сторінку корзини (рис. 3.9). Тут він бачить всі товари, які він додав в корзину, їх вартість і кількість (кількість можна змінити) Користувач може або повернутися на головну сторінку магазину, або перейти до процедури оформлення замовлення. Для цього потрібно авторизуватися (рис 3.10), якщо обліковий запис користувача вже був створений, або пройти процедуру реєстрації.

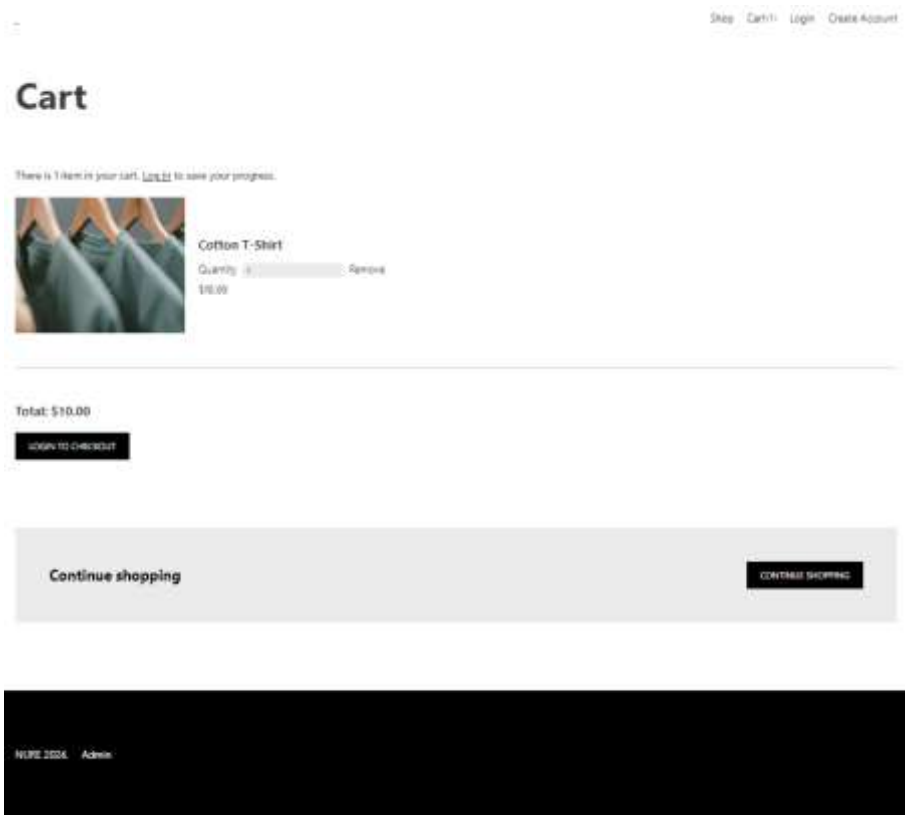


Рисунок 3.9 – Макет сторінки корзини (рисунок виконано самостійно)

Рисунок 3.10 – Макет сторінки авторизації (рисунок виконано самостійно)

Після успішної авторизації користувач буде переадресовано на сторінку розміщення замовлення (рис. 3.11).

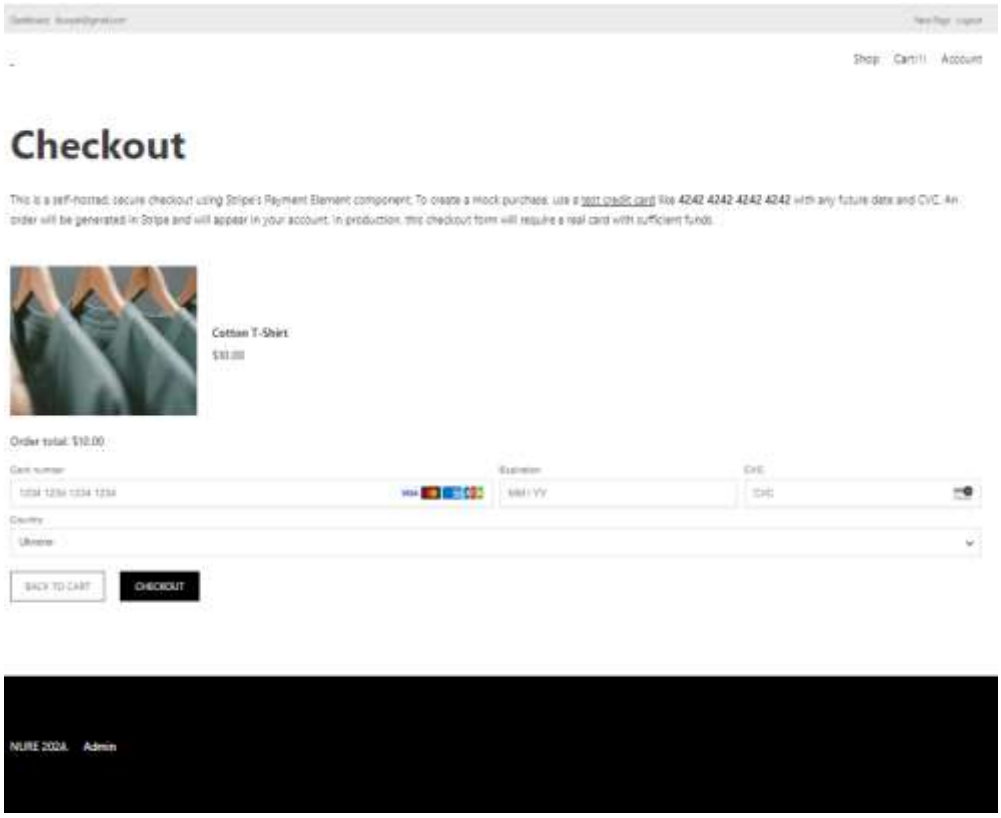


Рисунок 3.11 – Макет сторінки оплати замовлення (рисунок виконано самостійно)

На цій сторінці відображено зміст покупки, загальну вартість покупки та форма для введення даних кредитної картки.

Після успішної оплати користувач побачить сторінку з підтвердженням розміщення замовлення (рис. 3.12) та кнопки переходу на весь перелік замовлень даного користувача або на щойно розміщене замовлення (рис. 3.13).

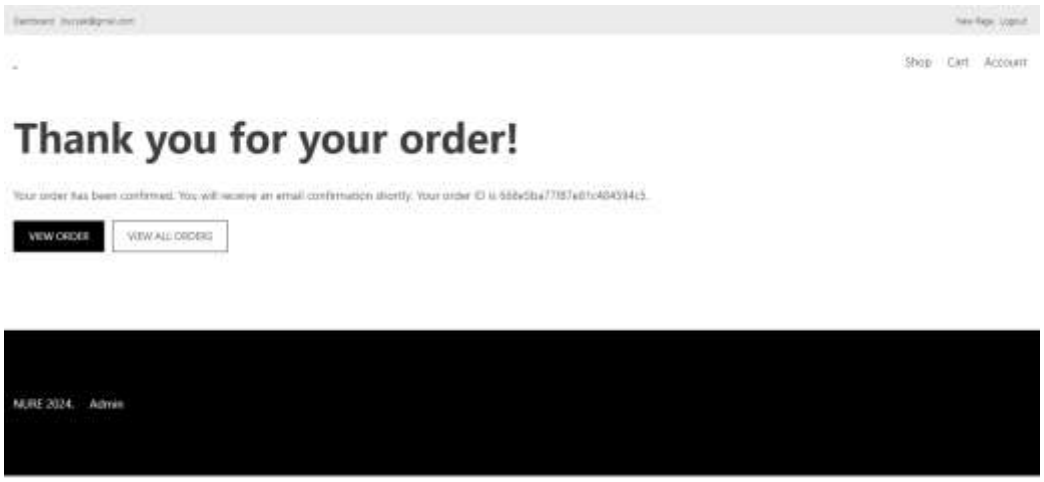


Рисунок 3.12 – Макет сторінки підтвердження замовлення (рисунок виконано самостійно)

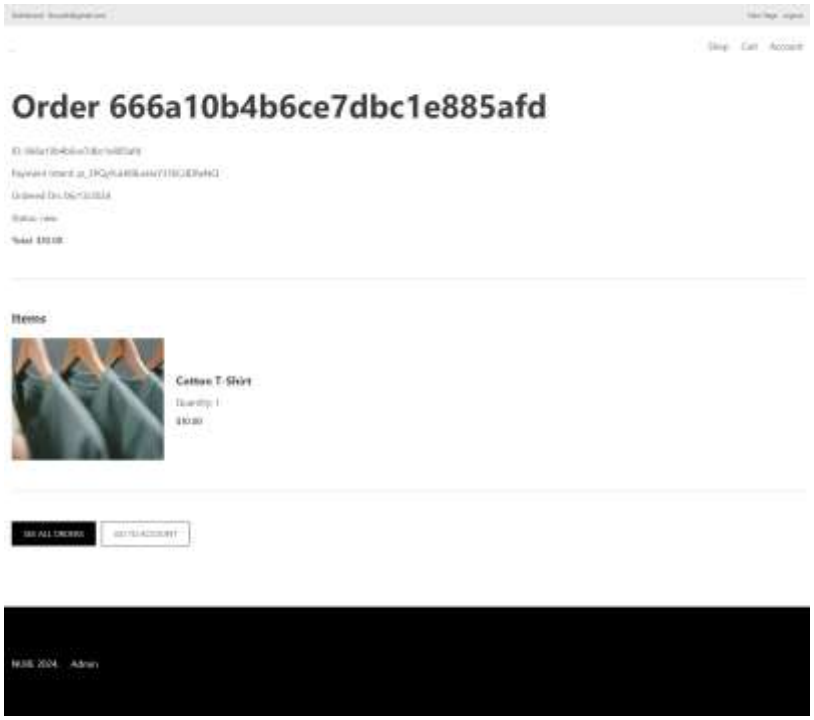


Рисунок 3.13 – Макет сторінки замовлення (рисунок виконано самостійно)

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Інтеграція з платіжною системою

Головна особливість інтеграції з платіжною системою полягала в тому, щоб серверна частина ніяк не взаємодіяла з чутливими даними користувача (номер кредитної картки, строк дії картки на CVC код). Можливість не доводити відповідність стандарту PCI DSS значно спрощує та пришвидшує розробку, тому компанія Stripe пропонує офіційне рішення для безпечного збору платіжної інформації.

Взаємодія з Stripe API значно спрощується у випадку використанння офіційної клієнтської бібліотеки stripe-js. Додатково була розроблена бібліотека елементів інтерфейсу Stripe Web Elements, яку можно використовувати разом з React.JS за допомогою пакету react-stripe-js.

Розглянемо деталі технічної реалізації. Elements провайдер дозволяє використовувати Stripe Web Elements компоненти та отримати доступ до глобального Stripe об’єкту в будь-якому із дочірніх елементів. Оскільки Stripe використвується лише на сторінці оформлення продажу, доцільно додавати провайдер лише на цю сторінку, але обгортати весь програмний код форми оплати замовлення щоб мати доступ до необхідних об’єктів.

Спочатку треба ініціалізувати під’єднання до Stripe за допомогою функції loadStripe, в яку треба передати публічний ключ. Отриманий об’єкт треба передати в Elements провайдер в якості параметра, таким чином ініціалізувавши провайдер. Провайдер також може приймати багато різних параметрів та змінних, які дозволяють його гнучке конфігурування.

```
{clientSecret && (
  <Fragment>
    {error && <p>{`Error: ${error}`}</p>}
    <Elements
      stripe={stripe}
      options={{
        clientSecret,
        appearance: {
          theme: 'stripe',
          variables: {
            colorText: cssVariables.colors.base1000,
            fontSizeBase: '16px',
            fontWeightNormal: '500',
            fontWeightBold: '600',
            colorBackground: cssVariables.colors.base0,
            fontFamily: 'Inter, sans-serif',
            colorTextPlaceholder: cssVariables.colors.base500,
            colorIcon: cssVariables.colors.base1000,
            borderRadius: '0px',
            colorDanger: cssVariables.colors.error500,
            colorDangerText: cssVariables.colors.error500,
          },
        },
      }}
    >
      <CheckoutForm />
    </Elements>
  </Fragment>
)}
```

Рисунок 4.1 – Приклад ініціалізації провайдера для Stripe Web Elements (рисунок виконано самостійно)

Після відправки форми оплати спочатку треба відправити запит в платіжну систему. У випадку, якщо запит повертає помилку, цю помилку потрібно відобразити в формі. Після успішного завершення оплати згідно до реалізації ми створюємо замовлення в системі керування продажами і виконуємо переадресацію користувача на сторінку підтвердження замовлення.

В наступному, якщо платіжна система змінить статус транзакції, то вона автоматично викличе спеціальний API програми керування продажами інтернет-магазину, який оновить статус замовлення.

4.2 Інтеграція з пошуковим сервісом

Проблема реалізації інтеграції з стороннім пошуковим сервісом виникає дуже часто. Особливість цієї інтеграції в тому, що дані в пошуковому сервісі повинні завжди відповідати даним в головній системі. Варіантів реалізації такої інтеграції зазвичай декілька – синхронізація даних за розкладом або синхронізація даних за подією.

Синхронізація за розкладом – найбільш проста опція, але вона має недоліки. Головний – зважаючи на потребу завжди тримати актуальний набір даних в пошуковому сервісі, у випадку коли дані оновлюються в різний час, синхронізація повинна відбуватись досить часто (наприклад, раз на хвилину). Це призводить до додаткового навантаження на сервер, базу даних та мережеву інфраструктуру.

Альтернативою синхронізації за розкладом є залучення принципів подійно-орієнтованої архітектури (event-driven architecture, EDA). Згідно цього підходу, у випадку якщо відбувається подія зміни даних в головній системі (наприклад, додавання нового продукту або оновлення існуючого), ми можемо підписатися на конкретну подію і виконати програмний код, який має обробити цю подію [7]. В нашому випадку, при обробленні події повинне відбуватися оновлення даних, які змінилися, в пошуковому сервісі.

Недоліком такого підходу до синхронізації є можливе збільшення кількості запитів у випадках, коли багато сутностей змінюються за короткий проміжок часу. Наприклад, розглянемо ситуацію з імпортом продуктів до системи. В цьому випадку на кожен продукт буде згенеровано окрему подію, і якщо обробник події викликає пошуковий сервіс для оновлення даних, то пошуковий сервіс може буди перевантажений через велику кількість запитів за короткий проміжок часу.

В якості рішення цієї проблеми може використовуватися проміжний накопичувач, який буде зберігати запити на оновлення даних в пошуковий сервіс та відправляти їх разом, як пакет оновлень замість окремих оновлень. Саме такий підхід було реалізовано при розробці програмної системи керування продажами інтернет-магазину.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування програмного застосунку

Для перевірки функціональних можливостей програмної системи було застосоване мануальне тестування. Автоматизація окремих тест-кейсів в майбутньому допоможе пришвидшити релізи нових функціональних можливостей, а також покращити якість програмної системи за рахунок пришвидшення регресійного тестування.

5.2 Приклади критичних помилок

Приклади критичних помилок наведено в таблиці 6.1. Критичними визначені помилки, які заважають клієнту розмістити замовлення.

Таблиця 5.1 – Тест-кейс №1 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Оплата замовлення кредитною карткою		
Власник тесту:	Буряк Ігор Олександрович		
Дата створення:	5.07.2024		
Мета тесту:	Перевірити коректність реалізації функціоналу оплати замовлення кредитною карткою		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити головну сторінку програмної системи	Користувач має доступ до системи	Пройдено
2	Додає товар у корзину	Товар відображено на сторінці корзини	Пройдено
Оплата замовлення кредитною карткою			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити сторінку оформлення замовлення	З’являється форма для оплати товару	Пройдено

Кінець таблиці 5.1

2	Додає товар у корзину	Товар відображено на сторінці корзини	Пройдено
2	Заповнити обов’язкові поля «Card number», «Expiration», «CVC»	Кількість символів у полях вказана правильно	Пройдено
3	Натиснути на кнопку «Checkout»	Користувача переадресовано на сторінку підтвердження оплати замовлення.	Пройдено
4	Підтвердити оплату	Оплата була підтверджена	Пройдено
5	Почекати	Користувача переадресовано на сторінку підтвердження замовлення.	Пройдено
Результати тестування			
Тестувальник: Буряк І.О.	Дата прогону тесту: 05.07.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Визначення плану впровадження

Для правильного формування етапу застосування онлайн-системи необхідно скласти таблицю етапів. Це допоможе уникнути надмірності помилок. Етапи впровадження та дати наведено в таблиці 6.1.

Таблиця 6.1 – План впровадження програмної системи

Назва етапу	Дата початку	Дата завершення
Внутрішнє тестування	01.07.2024	10.07.2024
Підготовка до закритого тестування	11.07.2024	14.07.2025
Закрите тестування на фокус-групі	15.07.2024	22.07.2024
Підготовка до виходу в «Production»	23.07.2024	25.07.2024
Вихід в «Production»	29.07.2024	29.07.2024

ВИСНОВКИ

Результатом роботи є розробка програмної системи керування продажами інтернет-магазину з використанням MACH-архітектури. Система включає в себе серверну частину на фреймворку Payload та клієнтську частину на фреймворку Next.js які взаємодіють з платіжним сервісом та сервісом пошуку.

Було розглянуто переваги обраної архітектури та використання подійно-орієнтованої архітектури для асинхронної реакції на певні події в системі (зміна статусів платежів, необхідність оновлення пошукового індексу).

Додатково був зроблений огляд стандарту PCI DSS в контексті розробки програмної системи керування продажами інтернет-магазину.

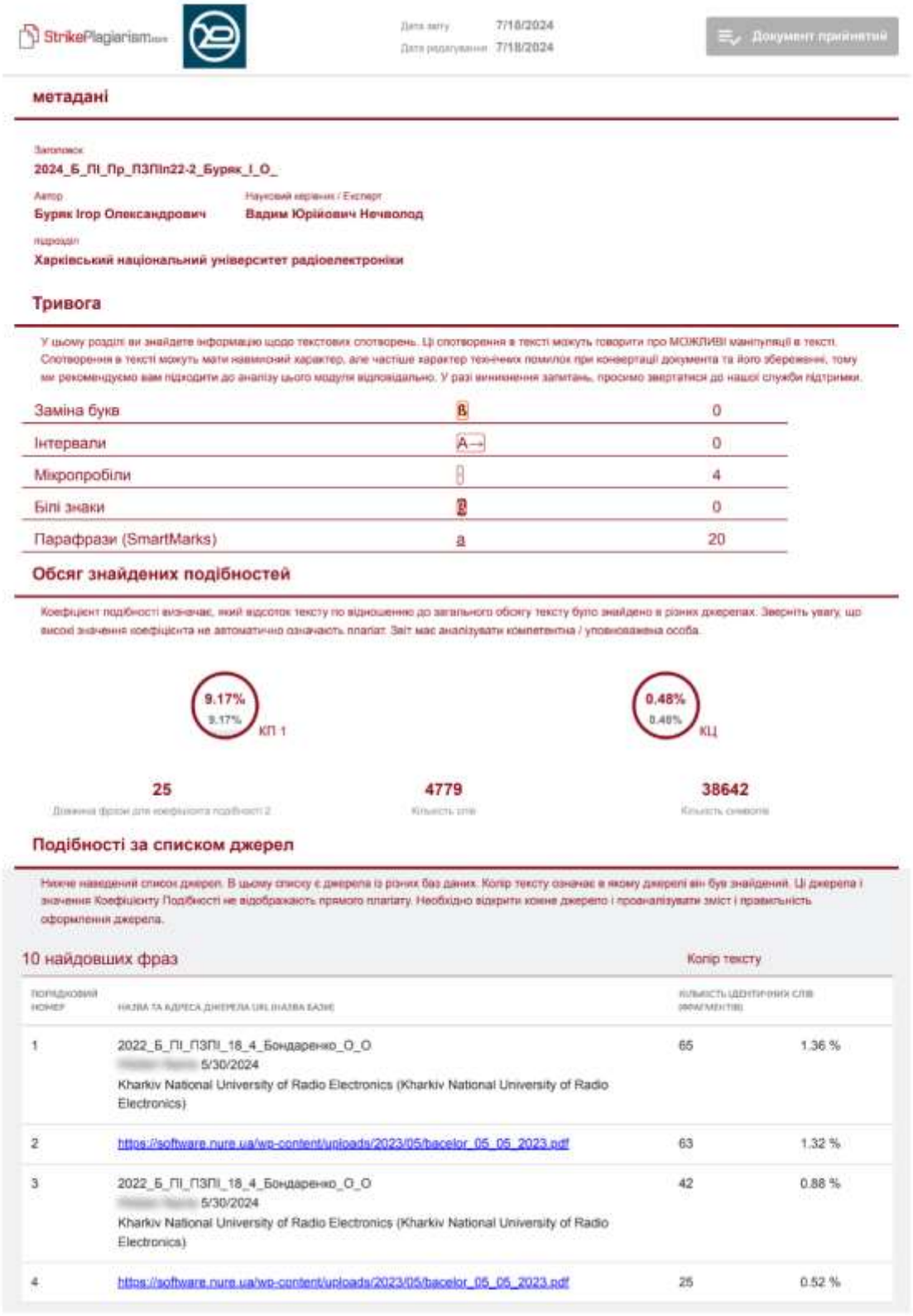
Вважаю, що розроблена програмна система не містить недоліків, які мають аналогічні вже існуючі програмні продукти, є розробленою згідно загально прийнятих в індустрії стандартів та готова до використання реальними користувачами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IEEE 830-1993 IEEE Recommended Practice for Software Requirements Specifications [Електронний ресурс] – URL: https://standards.ieee.org/ieee/830/1221/ (дата звернення: 01.04.2024)
2. MACH Alliance official website [Електронний ресурс] – URL: https://machalliance.org/ (дата звернення: 05.04.2024)
3. PCI DSS Quick Reference Guide: Understanding the Payment Card Industry Data Security Standard version 4.0. [Електронний ресурс] – URL: https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Supporting%20Document/PCI_DSS-QRG-v4_0.pdf (дата звернення: 01.06.2024)
4. Payload Configuration Documentation [Електронний ресурс] – URL: https://payloadcms.com/docs/configuration/overview (дата звернення: 14.06.2024)
5. MongoDB Documentation [Електронний ресурс] – URL: https://www.mongodb.com/docs/ (дата звернення: 15.06.2024)
6. Mongoose ODM Documentation [Електронний ресурс] – URL: https://mongoosejs.com/docs/guide.html (дата звернення: 15.06.2024)
7. Adam Bellemare – Building Event-Driven Microservices: Leveraging Organizational Data at Scale / O'Reilly Media, 2020 (дата звернення 24.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism

Дата звіту 7/18/2024
Дата редагування 7/18/2024

Документ прийнятий

метадані

Заголовок
2024_Б_ПІ_Пр_ПЗПІп22-2_Буряк_І_О_

Автор
Буряк Ігор Олександрович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

Підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	В	0
Інтервали	A→	0
Мікропробіли		4
Білі знаки		0
Парафрази (SmartMarks)	a	20

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

9.17% 9.17% КП 1

0.48% 0.48% КЦ

25
Довжина фрази для коефіцієнта подібності 2

4779
Кількість слів

38642
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL НАЗВА БАЗИ	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	2022_Б_ПІ_ПЗПІ_18_4_Бондаренко_О_О [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	65	1.36 %
2	https://software.nure.ua/wp-content/uploads/2023/05/bacelor_05_05_2023.pdf	63	1.32 %
3	2022_Б_ПІ_ПЗПІ_18_4_Бондаренко_О_О [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	42	0.88 %
4	https://software.nure.ua/wp-content/uploads/2023/05/bacelor_05_05_2023.pdf	25	0.52 %

ДОДАТОК Б

Слайди презентації

Кваліфікаційна робота бакалавра

Програмна система керування продажами інтернет-магазину

Виконав:
ст. гр. ПЗПІп-22-2
Буряк І.О.

Науковий керівник:
доц. кафедри ПІ
Лановий О.Ф

1

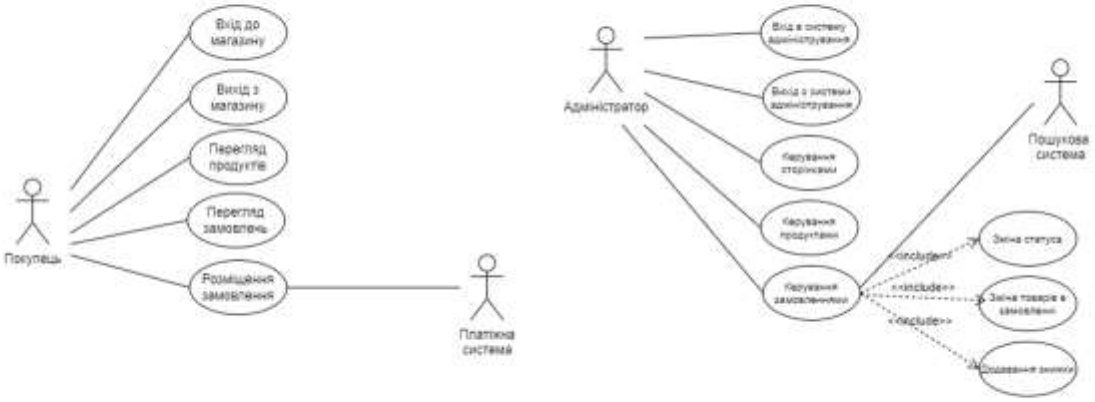
Мета роботи

- Виконати аналіз предметної галузі і виявити недоліки існуючих аналогічних програмних систем
- Спроектувати архітектуру системи, яка б була достатньо гнучкою та відкритою для змін, і з іншого боку вирішувала проблеми існуючих аналогів
- Розробити програмну систему керування продажами інтернет-магазину
- Спроектувати інтерфейс користувача
- Виконати тестування системи
- Проаналізувати результати роботи

2

Use-case діаграма

ХНУРЕ



3

Проблеми існуючих аналогів

ХНУРЕ

- Висока вартість ліцензії, проблеми з функціональністю у безкоштовних аналогів
- Написані на мові програмування PHP, яка втрачає популярність
- Безпека зберігання чутливих даних
- Недостатня функціональність з точки зору формування контенту сторінок і товарів
- Архітектура не дозволяє використовувати разом з сучасним відокремленим клієнтським додатком

4

MACH-архітектура

5

Архітектурна діаграма

Payload

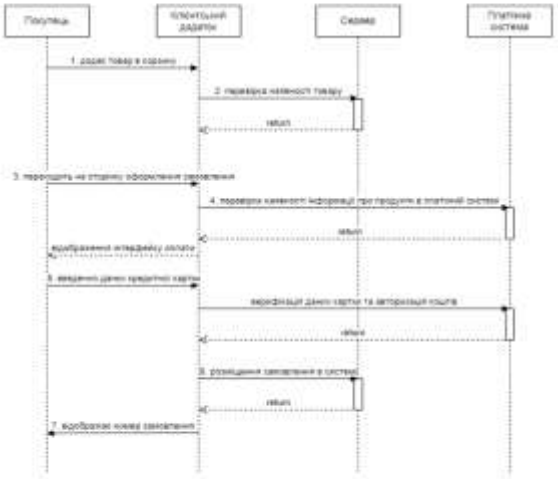
NEXT.JS

typesense

stripe

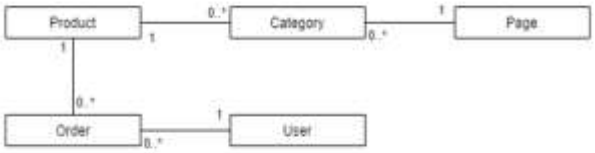
6

Процес розміщення замовлення



7

Проектування структури зберігання даних

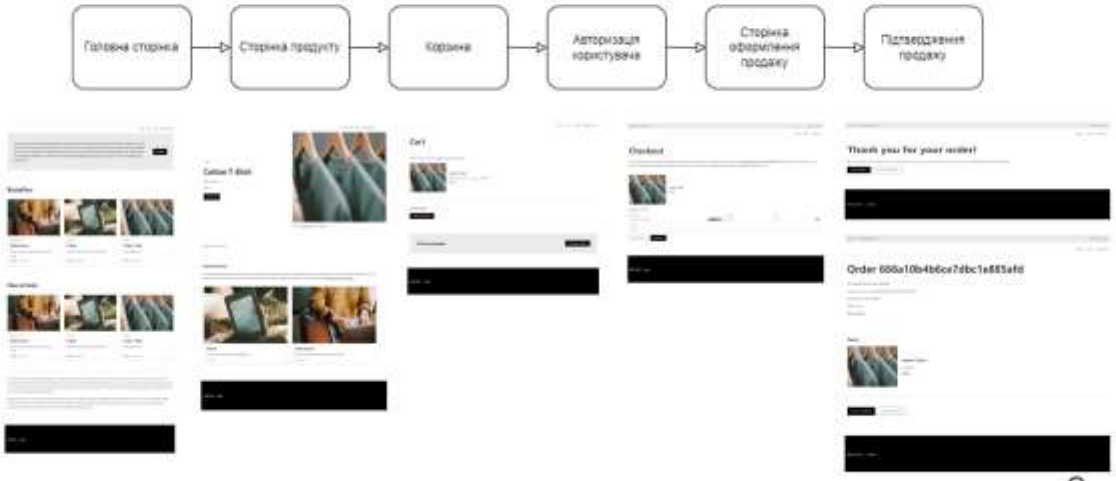


- Продукти можуть бути об'єднані у категорії.
- Сторінки дозволяють додавати туди різний контент, в тому числі перелік продуктів за категорій.
- Замовлення складаються з продуктів, кожен продукт може бути придбаний декілька разів.
- Кожне замовлення має користувача.

8

Проектування інтерфейсу

ХНУРЕ



9

Програмна реалізація інтерфейсу

ХНУРЕ

TypeScript – мова програмування з підтримкою типів.

TypeScript

Reactjs – популярна бібліотека для побудови компонентів.

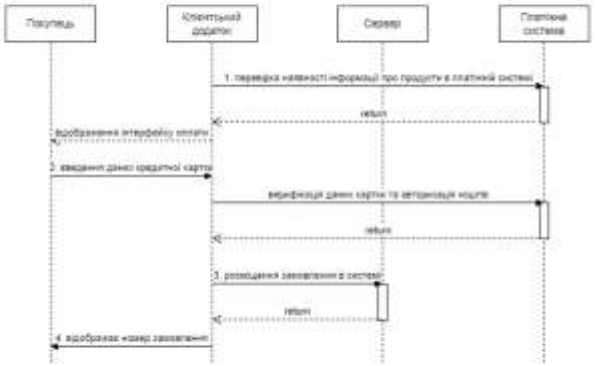
Nextjs – фронтенд фреймворк, підтримує Server-Side Rendering та Static Site Generation підходи.

NEXT.JS

10

Інтеграція з платіжною системою

ХНУРЕ



PCI DSS (Payment Card Industry Data Security Standard) — це набір стандартів безпеки, розроблених для забезпечення безпеки даних платіжних карток і захисту від шахрайства.

Stripe Web Elements дозволяє уникнути необхідності PCI DSS сертифікації.

11

Інтеграція з пошуковим сервісом

ХНУРЕ

typesense

Інтеграція з пошуковим сервісом TypeSense є прикладом використання подійно-орієнтованої архітектури в даній кваліфікаційній роботі

12

Результати роботи

ХНУРЕ

В результаті проведеного аналізу наявних на ринку продуктів було виявлено певні недоліки. Кваліфікаційна робота була спрямована на розробці програмного продукту, який усуває зазначені недоліки та є придатним до використання в реальному житті.

Було розроблено систему адміністрування, клієнтську частину для користувачів, виконані інтеграції з необхідними сервісами та проведене тестування програмного продукту.

13

Демонстрація роботи програмної системи

14