ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 26.00.00.000 ПЗ

Група ІП-20-2

Країло Богдан

2024

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: “ Розробка хмарного сервісу продажу товарів та послуг через мережу Інтернет ”

(назва згідно з наказом ректора)

спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 26.00.00.000 ПЗ

(позначення)

Консультанти:

з питань нормоконтролю

асист. /М.М. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

Студент

ІП-20-2 /Б.Р.Країло/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /Р.Б.Вовк/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

2024
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2024 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Країлу Богдану Романовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка хмарного сервісу продажу товарів та послуг через мережу Інтернет”

керівник проекту (роботи) Вовк Роман Богданович, доцент, к.т.н.

затверджені наказом закладу вищої освіти від **“** ____ ” травня 2024 р. № _____

2. Строк подання студентом проекту (роботи) 17 червня 2024 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз хмарних технологій для побудови сервісів продажу товарів та послуг через Інтернет
2. Розробка функціональних та нефункціональних вимог
3. Проектування структури та алгоритмічного забезпечення хмарного сервісу продажу товарів
4. Розробка моделі бази даних
5. Програмна реалізація хмарного сервісу продажу товарів та послуг через мережу Інтернет

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Функціональні області у e-commerce системі (рис. 1.1)
2. Види хмарних сервісів (рис. 1.2)
3. Моделі хмарних послуг (рис. 1.3)
4. Сервіси що надає Oracle Commerce Cloud (рис. 1.4)
5. Архітектра e-commerce проекту від Oracle ATG (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **22 травня 2024 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз хмарних технологій для побудови сервісів продажу товарів та послуг через Інтернет	26.05.2024	виконано
2	Розробка функціональних та нефункціональних вимог	29.05.2024	виконано
3	Проектування структури та алгоритмічного забезпечення хмарного сервісу продажу товарів	30.05.2024	виконано
4	Розробка моделі бази даних	04.06.2024	виконано
5	Програмна реалізація хмарного сервісу продажу товарів та послуг через мережу Інтернет	10.06.2024	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2024	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці хмарного сервісу продажу товарів через мережу Інтернет. У контексті зростаючої популярності електронної комерції та хмарних технологій, розробка такого сервісу стає актуальною задачею для підприємств будь-якого масштабу. Робота охоплює аналіз сучасних тенденцій у сфері електронної комерції та хмарних технологій, визначає основні вимоги до функціоналу сервісу, такі як каталог товарів, система оплати, управління замовленнями та інші.

КЛЮЧОВІ СЛОВА: WEB–ДОДАТОК, ЕЛЕКТРОННА КОМЕРЦІЯ ХМАРНІ ТЕХНОЛОГІЇ, GOOGLE CLOUD, СЕРВІС ПРОДАЖУ ТОВАРІВ, МАСШТАБУВАННЯ

ANNOTATION

This thesis is devoted to the development of a cloud service for the sale of goods via the Internet. In the context of the growing popularity of e-commerce and cloud technologies, the development of such a service becomes an urgent task for enterprises of any scale. The work analyzes current trends in the field of e-commerce and cloud technologies, requiring basic requirements for a functional service, such as a product catalog, payment systems, order management, etc.

KEY WORDS: WEB APPLICATION, ELECTRONIC COMMERCE, CLOUD TECHNOLOGIES, GOOGLE CLOUD, SELLING SERVICE, SCALING

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9

1. АНАЛІЗ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ СЕРВІСІВ ПРОДАЖУ ТОВАРІВ ТА ПОСЛУГ ЧЕРЕЗ МЕРЕЖУ ІНТЕРНЕТ 10
1.1. Важливість сервісів продажу товарів та електронної комерції у сучасному світі 10
1.2. Вимоги до систем електронної комерції 12
1.3. Принцип роботи хмарних систем 15
1.4. «Віртуальні» рішення для бізнесу 17
1.5. Порівняльна характеристика e-commerce платформ 19
1.6. Огляд інтеграторів для хмарного розгортання сервісів продажу товарів 24
1.6.1. Google Cloud Platform (GCP) 24
1.6.2 Amazon Web Services (AWS) 30
1.6.3. Azure 33

2. ПРОЕКТУВАННЯ СТРУКТУРИ ТА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ХМАРНОГО СЕРВІСУ ПРОДАЖУ ТОВАРІВ 36
2.1. Розробка функціональних та нефункцональних вимог 36
2.1.1. Функціональні вимоги 36
2.1.2. Нефункціональні вимоги 38
2.2. Моделювання та аналіз програмного забезпечення 38
2.3. Архітектура програмного забезпечення хмарного сервісу продажу товарів 39

2.4. Розробка діаграм послідовностей хмарного сервісу 43
2.4.1. Розробка діаграми послідовності оформлення замовлення користувачем 43
2.4.2. Розробка діаграми послідовності оформлення замовлення адміністратором 44
2.4.3. Розробка діаграми послідовності редагування замовлення адміністратором 46
2.5. Розробка моделі бази даних 48

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО СЕРВІСУ ПРОДАЖУ ТОВАРІВ ТА ПОСЛУГ ЧЕРЕЗ МЕРЕЖУ ІНТЕРНЕТ 53
3.1. Розробка діаграм варіантів використання 53
3.1.1 Діаграма варіантів використання Інтернет магазину 53
3.1.2. Діаграма варіантів використання можливостей адміністратора 56
3.2. Реалізація вимог до інтерфейсу користувача панелі адміністратора 59
3.3. Розробка інтерфейс системи для користувача Інтернет магазину 66
3.4. Приклад здійснення замовлення користувачем 71

ВИСНОВКИ 75
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 76
ДОДАТКИ
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

EDA – Event–Driven Architecture – подійно–орієнтована архітектура

BPMN – Business Process Model and Notation – модель та нотація бізнес-процесів

ORM – Object–Relational Mapping – об'єктно-реляційна проєкція

JSX – JavaScript Syntax Extension – розширення синтаксису JavaScript

JPA – Java Persistence API – Java API збереження в базу даних

vCPU – Virtual Central Crocessing Unit – віртуальний центральний процесор

CI/CD – Continuous Integration and Continuous Delivery – безперервна інтеграція та безперервна доставка

ВСТУП

Про «хмарні» технології та їх новизну потрібно говорити, не про інноваційність «хмарних» технологій, а про перехід кількості в якість, тобто про ефект масштабування. Як приклад цього виду послуг можна навести віртуально виділений сервер - сучасну технологію хостингу, що поєднує в собі потужність виділеного сервера з гнучкістю і простотою управління. Причому в деяких застосуваннях «хмарні» обчислення можуть стати альтернативою суперкомп’ютерів.

Для "хмарних" технологій особливе значення мають масштабування та балансування навантаження. Комп'ютерні технології щодня пропонують все нові і нові варіанти використання персональних серверів і комп'ютерів.

Також проблемою є вибір засобу балансування навантаження, що відповідає поставленим задачам та критеріям, його налаштування у відповідності до поставлених вимог

Для розробки високонавантажених інтернет магазинів, компанії наймають штат співробітників, або стороннього вендора з штатом співробітників для розробки інтернет магазину використовуючи такі рішення, як Oracle Commerce, SAP Hybris, та інші.

Відомі рішення, як правило побудовані на базі монолітної архітектури, тому не ефективно використовують ресурси при розгортанні на хмарних платформах, так як для масштабування їм потрібно розгортати репліки цього додатку.

Аналіз ринку показав, що немає рішень, які задовольняють потреби у навантажені на сайти великих інтернет–магазинів та ефективно використовують ресурси хмарних платформ, тому було прийнято рішення про розробку даного проєкту.

1. АНАЛІЗ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ СЕРВІСІВ ПРОДАЖУ ТОВАРІВ ТА ПОСЛУГ ЧЕРЕЗ МЕРЕЖУ ІНТЕРНЕТ

1.1. Важливість сервісів продажу товарів та електронної комерції у сучасному світі

Недавні дослідження чітко вказують на те, що електронна комерція в даний час є найбільш популярним способом, яким люди купують продукцію. Ринок електронної комерції також набув великої популярності серед західних країн, зокрема, Європи та США. Ці країни характеризувалися споживчими товарами.

Електронна комерція стала важливим інструментом для малих і великих підприємств у всьому світі - не лише для продажу товарів клієнтам, але й для їх залучення нових покупців. Для традиційних підприємств одне дослідження заявило, що інформаційні технології та транскордонна електронна торгівля є гарною можливістю для швидкого розвитку та зростання підприємств. Багато компаній вклали величезні обсяги інвестицій в мобільні додатки. Немає ніяких обмежень у часі та просторі, є більше можливостей для спілкування з клієнтами по всьому світу, а також для скорочення непотрібних проміжних зв'язків, тим самим знижуючи собівартість.

Ринки електронної комерції зростають помітними темпами. Традиційні ринки лише очікують зростання на 2% за той же час. Багато великих роздрібних підприємств можуть підтримувати як фізичні магазини так і магазини в мережі Інтернет, пов'язуючи фізичні та мережеві пропозиції. Електронна комерція дозволяє клієнтам долати географічні бар'єри і дозволяє їм купувати продукти в будь-який час і з будь-якого місця. Інтернет і традиційні ринки мають різні стратегії ведення бізнесу. Традиційні роздрібні торговці пропонують менший асортимент продукції через місця на полицях,

де інтернет-магазини часто не мають інвентаризації, але відправляють замовлення клієнтів безпосередньо на виробництво. Стратегії ціноутворення також відрізняються для традиційних і інтернет-магазинів. Традиційні роздрібні торговці базують свої ціни на трафіку в магазинах і вартість зберігання запасів.

Інтернет-магазини роздрібної торгівлі базують ціни на швидкість доставки. Для маркетологів існує два способи ведення бізнесу через електронну комерцію: повністю онлайн або онлайн, разом зі складом та фізичним магазином. Інтернет-маркетологи можуть запропонувати більш низькі ціни, більший вибір продукції та високу ефективність. Багато клієнтів віддають перевагу онлайн ринкам, якщо продукти можна швидко доставити за відносно низькою ціною. Тим не менш, інтернет-магазини не можуть запропонувати ефект присутності, який можуть надати традиційні роздрібні магазини. Іншим питанням, що стосується онлайн-ринку, є занепокоєння щодо безпеки онлайн-транзакцій.

Безпека є основною проблемою для електронної торгівлі в розвинених і в країнах, які розвиваються. Безпека електронної комерції захищає веб-сайти та клієнтів від несанкціонованого доступу, використання, зміни чи знищення їх даних. До типів загроз можна віднести: шкідливі коди, небажані програми, фішинг, хакерство та кібер-вандалізм. Веб-сайти електронної комерції використовують різні інструменти для запобігання загроз. Ці інструменти включають брандмауери, програмне забезпечення для шифрування, цифрові сертифікати та паролі. Інтернет-магазин — це форма електронної комерції, яка дозволяє споживачам безпосередньо купувати товари або послуги від продавця через Інтернет за допомогою веб-браузера. Споживачі знаходять продукт, який їм цікавий, безпосередньо або відвідуючи веб-сайт роздрібного продавця, або шукаючи серед альтернативних постачальників, використовуючи пошукову систему для покупок, яка відображає доступність та цінову оцінку того ж продукту в різних електронних магазинах.

Інтернет-магазини зазвичай дозволяють покупцям використовувати функції "пошуку" для пошуку конкретних моделей, брендів або предметів. Онлайн-клієнти повинні мати доступ до Інтернету та дійсний спосіб оплати для завершення операції, наприклад, кредитної картки або дебетової картки. Інтернет-магазини зазвичай доступні цілодобово, і багато споживачів у західних країнах мають доступ до Інтернету як на роботі, так і вдома. Інші установи, такі як інтернет-кафе, громадські центри та школи, також надають доступ до Інтернету. Конфіденційність особистої інформації є важливим питанням для більшості споживачів. Багато споживачів хочуть уникнути спаму та телемаркетингу, які можуть виникнути в результаті постачання контактної інформації в мережі Інтернет. У відповідь на це багато торговців обіцяють не використовувати споживчу інформацію для цих цілей. Багато веб-сайтів відстежують звички споживачів покупок, щоб пропонувати предмети та інші веб-сайти для перегляду. Побудова систем для електронної комерції є ключовим фактором успішності бізнесу, адже більшість, або навіть усі операції проводяться у мережі Інтернет. Тому розроблені додатки мають бути надійними, швидко працювати, а також мають бути безпечними, як для покупців, так і для продавців.

Таким чином, електронна комерція є тою, що швидко розвивається та актуальною сферою, тому що приносить великий прибуток. Обсяг даних, що треба обробити стає все більшим. Через це системи стають все більш складнішими та потребують нових архітектурних підходів для того, щоб відповідати вимогам бізнесу та споживачів.

1.2. Вимоги до систем електронної комерції

Для того щоб спроектувати систему електронної комерції використовуючи мікросервісний архітектурний стиль, треба виділити основні функціональні області типового e-commerce додатка. В залежності від

складності системи ці функціональні області потрібно бути розбити на менші сервіси та продумати зв’язки між ними.

Типова e-commerce система складається з таких областей: каталог продуктів, управління складськими запасами, обробка замовлення, кошик, персональний аккаунт покупця, обробка оплати замовлення, обробка повернення замовлення, організація доставки, управління кампаніями (рис 1.1).

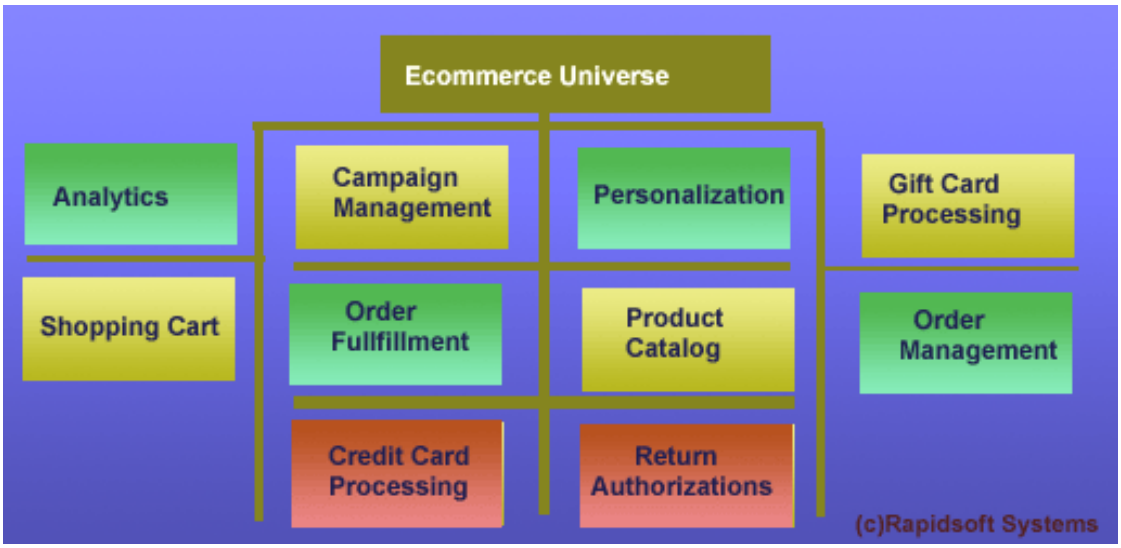


Рисунок 1.1 - Функціональні області у e-commerce системі

Розглянемо деякі з них більш детально:

— каталог продуктів. Він є невід’ємною частиною кожного e-commerce додатка. Чим більший асортимент магазину, тим складнішим може буде імплементація цією функціональної області. У великих магазинів продукт зазвичай розподіляється на декілька типів. Наприклад, у одягу є фіксовані розміри, а у деяких товарів покупець може задавати персональний розмір. Деякі товари можна повернути згідно з законодавством, а деякі ні. І всі ці товари можуть потенційно продаватися в одному магазині, що може ускладнити відображення товарів покупцю та обробку операцій з товарами;

— управління складськими запасами. До цієї функціональної області можна віднести управління онлайн запасами, сповіщення покупців про

товари, які знову в наявності. У разі якщо магазин перепродає товари інших продавців за управління складськими запасами може відповідати безпосередньо продавець, тому сюди також можна віднести синхронізацію даних про запаси з іншими продавцями;

— обробка замовлення. Є однією з найважливіших частин e-commerce систем та відповідає за резервацію товару на початку обробки вашого замовлення, збір інформації про адресу доставки та підбір наявних методів доставки згідно з нею, збір інформації про метод оплати замовлення та повторна спроба списання коштів у разі невдачі, відправка повідомлення у разі успішного замовлення. Обробка замовлення може бути різною в залежності від типу продукта;

— персональний аккаунт покупця. Включає зберігання даних про адресу покупця, його платіжну інформацію, історію замовлень, переліки побажань та інше;

— обробка оплати замовлення. Може включати в себе обробку оплати різними методами - кредитна картка, PayPal та інші платіжні системи, подарункові карти, кредити аккаунта, відкладена оплата та інші.

Сучасні системи електронної комерції знаходяться в більш жорстким умовах конкуренції, завдяки розповсюдження онлайн торгівлі та можливості замовляти товари з інших країн. Тому дуже важливо, щоб e-commerce система викликала довіру у користувача. Вона потрібна бути надійною та захищеною з точки зору зберігання чутливих даних користувача - це найголовніше. Не менш важливою є прозорість для користувача щодо списання коштів. Інформування користувача про стан його замовлення та інші операції, які він виконував, у різних каналах зв’язку надає йому відчуття спокою та викликає довіру. І звичайно система повинна бути швидкою та комфортною у користуванні.

Таким чином, метою роботи є проектування оптимальної архітектури великої e-commerce системи з точки зору швидкості роботи, надійності,

швидкості розробки, швидкості та ціни розгортання, вартості підтримки. Для цього є необхідним розглянути наявні на ринку хмарні інтегратори додатків, обрати оптимальний патерн розгортання сервісу.

1.3. Принцип роботи хмарних систем

Хмарні обчислення - це певний базис-вектор, отриманий в результаті синтезу цілого ряду технологій і підходів. Хмарні технології - це набір засобів, що виконує обчислення за допомогою віддалених серверів і програм без безпосереднього залучення ресурсів комп'ютера користувача. Можливо, в майбутньому комп‘ютери будуть представляти один лише екран з мікропроцесором, а всі обчислення і потужності будуть розташовані і виконуватися віддалено на серверах «хмари».

Нині хмарний сервіс включає три основні характеристики, які відрізняють його від звичайного сервісу:

- режим "ресурси за запитом";
- еластичність;
- незалежність від елементів управління інфраструктурою.

Національним інститутом стандартів і технологій США зафіксовані наступні обов'язкові характеристики хмарних обчислень:

Самообслуговування на вимогу. Це означає, що споживач визначає і змінює потреби і швидкість обробки даних, об'єм даних, що зберігаються, без взаємодії з представником постачальника послуг. Тобто з великого переліку послуг користувач обирає лише ті, що потрібні саме йому в залежності від масштабів компанії та основних бізнес-процесів.

Універсальний доступ по мережі. Як зазначали вище, послуги доступні споживачам по мережі передачі даних незалежно від пристрою та місця розташування. Головне – наявність Інтернет-підключення.

Хмарна IT-інфраструктура для бізнесу базується в дата-центрах, які для підвищення безпеки раніше дуже часто розташовувалися за кордоном. Відбір співробітників в штат центру обробки даних проводиться так само ретельно, як в банківські структури. Клієнтська інформація зберігається на захищених від несанкціонованого доступу високонадійних носіях. Важливі дані в разі помилки клієнтів можуть бути відновлені за рахунок спеціальної системи резервного копіювання. Рівень захисту персональної та конфіденційної інформації такий, що багато провайдерів хмарних послуг не завжди знають, які програми розміщують в хмарі клієнта.

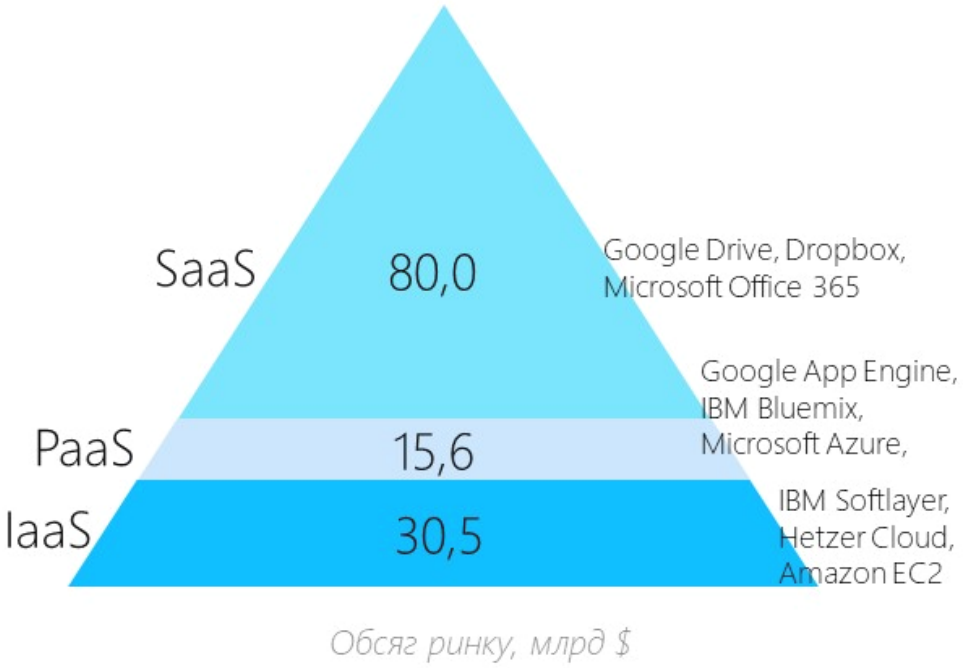


Рисунок 1.2 – Види хмарних сервісів

Практично будь-який хмарний сервіс можна запустити в термін від декількох годин до декількох днів. Для цього потрібно оформити заявку на сайті компанії-провайдера і внести передоплату. Після надходження грошей клієнту надається доступ до сервісів класів IaaS (Infrastructure-as-a-Service), SaaS (Software-as-a-Service), PaaS (Platform-as-a-Service).

1.4. «Віртуальні» рішення для бізнесу

Найбільшу популярність в даний час набули такі моделі хмарних технологій:

Оренда віртуального сервера (хмарний центр обробки даних). Дата-центр, побудований на хмарних технологіях, є сервери, диски і мережі, доступні через Інтернет або виділені канали зв'язку. Для клієнта створюється портал самообслуговування, через який він зможе здійснювати управління даними. Це дозволяє повністю контролювати всі розміщені в хмарі можливості незалежно від провайдера послуг.

Віртуальний офіс. Робоче місце можна організувати, чи не прив'язуючи його до певного комп'ютера, - у віртуальному просторі. В хмарі відтворюється внутрішня мережа компанії, включаючи мережеві диски, загальні папки, програми-планувальники. Віртуальний офіс дозволяє повноцінно замінити стаціонарні робочі комп'ютери скрізь, де є Інтернет.

Резервне копіювання. Можливість для приватного віртуального дата-центру, передбачена для того, щоб забезпечити збереження даних в надзвичайних випадках.

Система резервного копіювання гнучко настроюється під поточний обсяг інформації, що дозволяє скоротити часові і фінансові витрати.

Катастрофостійкість (DRaaS). Сервіс індивідуальних рішень по забезпеченню катастрофостійкості об'єднує кілька хмарних майданчиків з виходом на декількох операторів зв'язку. Безпека даних досягається шляхом перехресного резервного копіювання в автоматизованому або ручному форматі.

Гібридна хмара. Це можливість підключити приватне хмара до публічної хмарної структурі провайдера, щоб розвантажити власні потужності в період високої ділової активності - наприклад, при здачі важливих звітів. Все некритичні на даний момент процеси переносяться в

хмару провайдера, а потім повертаються назад. Це дозволяє істотно прискорити процеси у власному хмарі.

Оренда додатків. Програмне забезпечення можна не купувати для установки на комп'ютер, а отримувати через Інтернет, сплативши користування на потрібний термін. При цьому клієнту гарантується цілодобова технічна підтримка і безпеку даних.

Віртуальний контакт-центр. Організація традиційного контакт-центру вимагає спеціального приміщення, обладнання робочих місць та оплати праці офісних працівників. Віртуальний контакт-центр, організований за хмарної технології, дозволяє вивільнити величезну кількість ресурсів і розгорнути роботу за два дні з моменту подачі заявки провайдеру.



Рисунок 1.3 - Моделі хмарних послуг

Приватна хмара. Це віртуальна інфраструктура, створена для кількох підрозділів однієї компанії, її клієнтів і підрядників. Приватне хмара може бути шлюзом до загальнодоступного хмарі, одночасно забезпечуючи

використання постійно зростаючого набору послуг і зберігаючи важливі для конкретного бізнесу інформаційні системи всередині.

1.5. Порівняльна характеристика e-commerce платформ

Більшість CMS (система керування контентом) орієнтовані на інтернет-магазини для роздрібної торгівлі. Сучасна електронна комерція пропонує можливості і для сектора B2B, наприклад, як ATG Oracle – ефективне рішення для великого бізнесу. Розробники постаралися врахувати всі потреби великих компаній, створивши безліч додаткових інструментів для роботи онлайн.

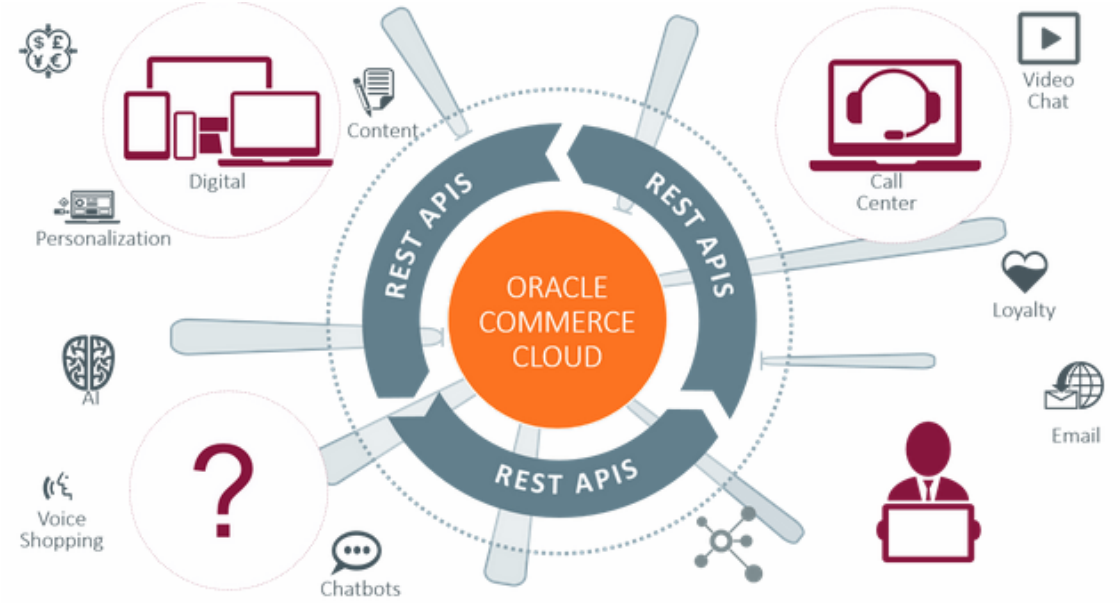


Рисунок 1.4 – Сервіси що надає Oracle Commerce Cloud

Oracle Commerce Cloud - це хмарний сервіс електронної комерції, який надається Oracle Corporation. Ось деякі з основних особливостей Oracle Commerce Cloud:

- Швидкість і масштабованість: Oracle Commerce Cloud пропонує швидкість обробки та масштабованість для підтримки великих обсягів трафіку і транзакцій.

- Мобільна оптимізація: Платформа надає інструменти для оптимізації інтерфейсів мобільних пристроїв, забезпечуючи зручний і функціональний досвід для користувачів.

- Персоналізація: Oracle Commerce Cloud дозволяє персоналізувати вміст, рекомендації та пропозиції для кожного користувача на основі їхнього поведінки та інших факторів.

- Управління контентом: Платформа має різноманітні інструменти для створення, редагування та управління контентом на сайті електронної комерції без потреби в глибоких технічних знаннях.

- Аналітика та звітність: Oracle Commerce Cloud забезпечує різноманітні інструменти для аналізу даних та створення звітів, що дозволяє магазинам ефективно відстежувати та аналізувати продажі, поведінку користувачів та інші метрики.

- Інтеграція з іншими системами: Платформа може легко інтегруватися з іншими системами, такими як системи управління відносинами з клієнтами (CRM), системи управління складом (ERP) та інші.

- Безпека і забезпечення відповідності: Oracle Commerce Cloud надає різні заходи безпеки та дотримання вимог, щоб забезпечити захист конфіденційності та безпеки даних клієнтів.

Ці особливості роблять Oracle Commerce Cloud потужним інструментом для створення та управління магазинами електронної комерції в різних сегментах ринку.

Oracle ATG (Art Technology Group) - це платформа для створення та управління електронними комерційними сайтами та додатками, що належить компанії Oracle Corporation. Ось деякі особливості Oracle ATG:

1. Електронна комерція: Oracle ATG надає потужні засоби для створення та управління онлайн-магазинами, включаючи можливості обробки платежів, каталогізації товарів, керування замовленнями та інші.

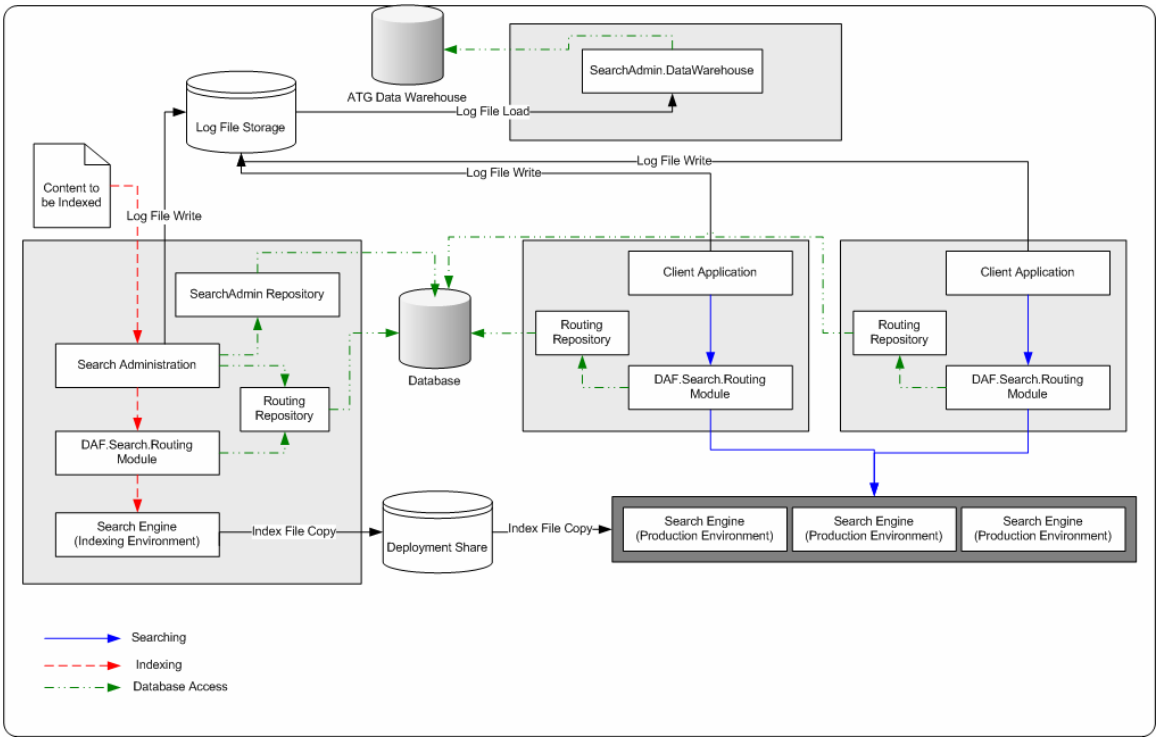


Рисунок 1.5 – Архітектра e-commerce проекту від Oracle ATG

2. Персоналізація: Платформа дозволяє персоналізувати вміст та пропозиції для кожного користувача на основі їхнього поведінки, історії покупок, демографічних даних та інших факторів.

3. Управління контентом: Oracle ATG має різноманітні інструменти для створення, редагування та управління контентом на веб-сайті, включаючи статичний контент, акції, новини та інше.

4. Маркетинг та аналітика: Платформа забезпечує інструменти для проведення маркетингових кампаній, а також аналізу даних, щоб відстежувати та аналізувати результати та ефективність дій.

5. Інтеграція з іншими системами: Oracle ATG може легко інтегруватися з іншими системами, такими як системи управління

відносинами з клієнтами (CRM), системи управління складом (ERP), системи управління контентом (CMS) та інші.

6. Масштабованість та надійність: Платформа забезпечує можливості масштабування для підтримки великих обсягів трафіку та транзакцій, а також надійності для забезпечення безперебійної роботи магазину.

SAP Hybris є продукцією сімейства німецької фірми Hybris, яка є програмним забезпеченням для e-commerce, marketing, sales, maintenance and product content management. SAP Hybris забезпечує рішення, що сприяють будь-якій організації зменшення вартості, обмеження часу, зменшення складності, і потребує невеликої уваги до того, щоб досягти вищої вартості експертів.

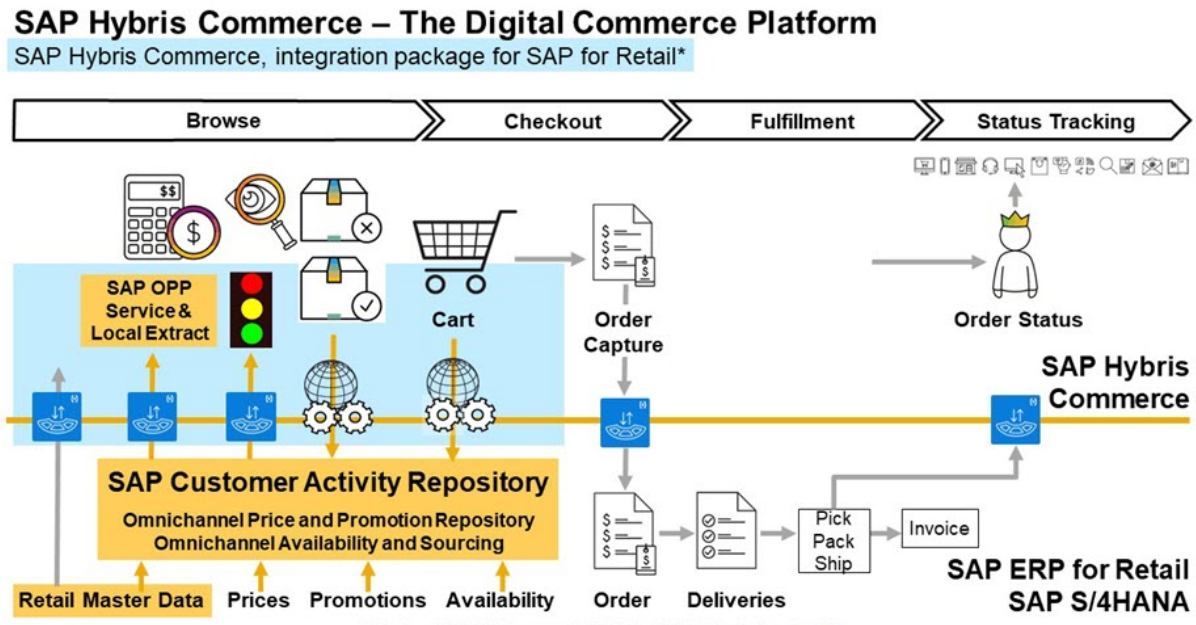


Рисунок 1.6 - SAP Commerce, інтеграційний пакет для SAP for Retail

Хмара SAP Hybris для продажу отримує інформацію про клієнтів із серверної системи, передає її групі зовнішніх клієнтів та дозволяє їм зрозуміти цільових клієнтів та дізнатися, як використовувати кожну можливість нового продажу. Це забезпечує інформацію для менеджера з

продажу на місцях та на мобільному пристрої в руках менеджера з продажу. Він включає продукт для роздрібного виконання, управління ефективністю продажів і автоматизації продажів.

SAP Hybris for Service дозволяє компанії надавати своїм клієнтам винятковий досвід обслуговування та, отже, чудовий досвід взаємодії з клієнтами.

Використовуючи Hybris Service організації можуть надавати своїм клієнтам правильний сервіс на потрібному каналі. Він включає продукт для Omni Channel Call Center, Проактивне обслуговування на місцях і Комплексне самообслуговування.

Hybris як послуга (YaaS) – це одна з найбільш розвинених екосистем мікро-послуг, яка дозволяє компанії розробляти користувацькі програми на існуючій платформі. YaaS дозволяє компаніям повторно збирати та адаптувати існуючі сервіси для створення інтерфейсів користувача без необхідності їх розробки з нуля.

SAP Hybris надає продукти для електронної комерції, маркетингу, продажу та обслуговування, доходів та міжфункціональних рішень, і вони покликані допомогти організаціям створювати цінні взаємодії зі своїми клієнтами та підтримувати різні типи галузей.

Hybris включає портфель продуктів для наступних можливостей:

− товари для торгівлі;

− продукти для маркетингу;

− продукти для продажу;

− товари для сервісу;

− продукти для виставлення рахунків;

− крос-функціональне рішення.

До переваг можна віднести:

− динамічне профілювання клієнтів;

− сегментація та управління кампаніями;

− комерційний маркетинг;

− керування лояльністю;

− управління маркетинговими ресурсами;

− маркетинговий аналіз;

− управління маркетингом;

− атрибуція клієнтів;

− архітектура та технологія.

Недоліки має наступні:

− неможливість роботи з фідбеками;

− одночасний аналіз тільки за одною метрикою;

− дорога вартість підписки

1.6. Огляд інтеграторів для хмарного розгортання сервісів продажу товарів

Розгортання мікросервісних додатків є більш складним, ніж розгортання монолітних додатків, хоча завдяки цьому є більш гнучким. Для того щоб полегшити цей процес можно використати існуючі на ринку IaaS, PaaS рішення.

Окрім полегшення розгортання вони також надають такі необхідні речі моніторінг, зберігання журналів, сповіщення про помилки у різних каналах зв’язку та інші необхідні застосування для зручної підтримки мікро сервісних додатків. Далі будуть розглянуті найбільші з таких інтеграторів - AWS та Google Cloud Platform.

1.6.1. Google Cloud Platform (GCP)

GCP складається з набору фізичних активів, таких як комп'ютери та жорсткі диски, і віртуальні ресурси, такі як віртуальні машини (VM), які містяться в центрах даних Google по всьому світу. Кожне розташування

центру обробки даних знаходиться в глобальному регіоні. Кожна зона ідентифікується назвою, що поєднує в собі ідентифікатор букви з назвою регіону. Такий розподіл ресурсів надає декілька переваг, включаючи зменшення затримки за допомогою розміщення ресурсів ближче до клієнтів. Цей розподіл також вводить деякі правила про те, як можна використовувати ресурси разом.

У хмарних обчисленнях те, що зазвичай називається програмне та апаратне забезпечення, стає сервісами. Ці сервіси надають доступ до базових ресурсів. Список доступних послуг GCP досить великий, і він постійно зростає. Коли ви розробляєте свій веб-сайт або додаток на GCP, ви поєднуєте ці сервіси у комбінаціях, які надають потрібну вам інфраструктуру, а потім додаєте код, щоб увімкнути сценарії, які потрібно створити.

GCP надає вам такі варіанти для обчислень і хостингу: працювати у безсерверному середовищі; використовувати платформу керованих програм; використовувати контейнерні технології, щоб отримати велику гнучкість; створити власну хмарну інфраструктуру, щоб мати найбільший контроль і гнучкість.

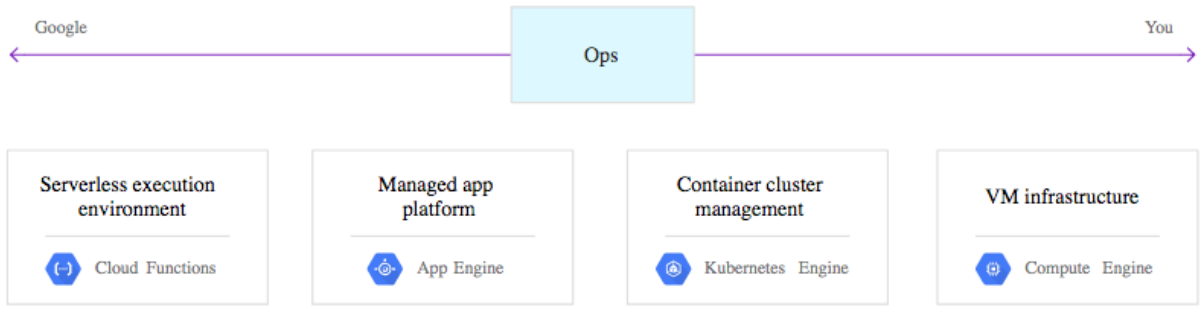


Рисунок 1.7 - Google Cloud Platform engines

Google Cloud Functions, функції GCP як сервіс (FaaS) надають середовище виконання сервера без створення серверів і підключення хмарних сервісів. За допомогою Cloud Functions ви можете писати прості, одноцільові функції, які приєднуються до подій, що піднімаються за

допомогою хмарної інфраструктури та служб. Cloud Function запускається, коли піднята спостережувана подія. Ваш код виконується в повністю керованому середовищі; Вам не потрібно надавати будь-яку інфраструктуру або турбуватися про керування будь-якими серверами.

Хмарні функції є гарним вибором для випадків використання, які включають:

— обробка даних та операції ETL для сценаріїв, таких як транс кодування відео та потокові дані IoT;

— Webhooks реагують на тригери HTTP;

— легкі API, які містять вільно пов'язану логіку в додатках;

— мобільні функції бекенда.

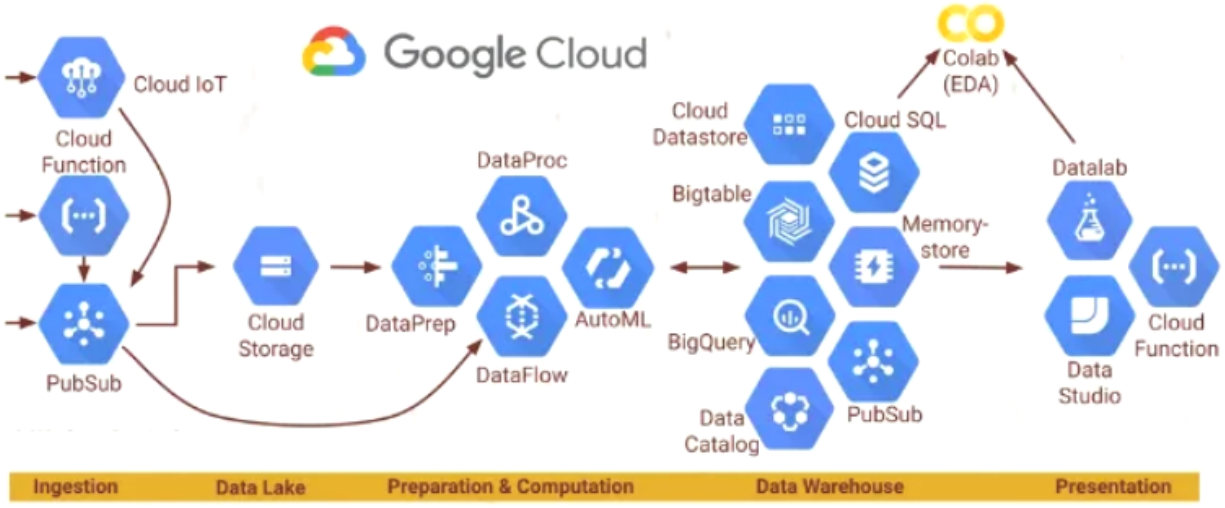


Рисунок 1.8 - Базові сервіси, що надає Google Cloud Platform

Google App Engine - це платформа GCP як сервіс (PaaS). За допомогою App Engine Google виконує більшу частину керування ресурсами. Наприклад, якщо програма потребує більшої кількості обчислювальних ресурсів, оскільки трафік на сайт збільшується, Google автоматично масштабує систему, щоб забезпечити ці ресурси. Якщо системному програмному забезпеченню потрібне оновлення для системи безпеки, яке також

використовується для вас. Коли створюється програма на App Engine, то користувач може:

— створити свою програму на базі стандартного середовища запуску App Engine на мовах, які підтримує стандартне середовище, включаючи: Python 2.7, Java 8, Java 7, PHP 5.5 і Go 1.8, 1.6;

— створити свою програму на базі гнучких середовищ App Engine на мовах, які підтримує гнучка програма App Engine, зокрема: Python 2.7 / 3.6, Java 8, Go 1.8, Node.js, PHP 5.6, 7, .NET і Ruby. Або використати спеціальний режими виконання, щоб використовувати альтернативну реалізацію підтримуваної мови або будь-якої іншої мови;

— дозволити Google керувати хостингом програм, масштабуванням, моніторингом та інфраструктурою;

— використати SDK App Engine для розробки та тестування на локальному комп'ютері в середовищі, що імітує App Engine на GCP;

— легко використовувати технології зберігання, призначені для підтримки в стандартних і гнучких середовищах. Google Cloud SQL — це база даних SQL, яка підтримує MySQL або PostgreSQL. Google Cloud Datastore є сховищем даних NoSQL.

Google Cloud Storage надає простір для великих файлів. У стандартному середовищі можна також вибрати з безлічі баз даних сторонніх виробників для використання з такими програмами, як Redis, MongoDB, Cassandra і Hadoop. У гнучкому середовищі можна легко використовувати будь-яку базу даних сторонніх виробників, яку підтримує ваша мова, якщо ця база даних доступна з екземпляра Google App Engine. У будь-якому середовищі ці бази даних третіх сторін можуть розміщуватися на комп'ютері Compute Engine, розміщеному на іншому постачальнику послуг у хмарі, розміщеному на місці або керованим постачальником сторонніх розробників;

— використати Cloud Endpoints у стандартному середовищі для створення API та бібліотек клієнтів, які можна використовувати для

спрощення доступу до даних з інших програм. Кінцеві точки спрощують створення веб-інтерфейсу для веб-клієнтів і мобільних клієнтів, таких як Android або iOS;

— використати вбудовані керовані служби для таких дій, як керування електронною поштою та користувачам;

— використати Cloud Security Scanner для виявлення вразливостей безпеки як доповнення до існуючих безпечних процесів проектування та розробки.

Програму може бути розгорнута за допомогою програми GUI для запуску App Engine на MacOS або Microsoft Windows або за допомогою командного рядка. За допомогою обчислень на базі контейнерів ви можете зосередитися на своєму коді програми, а не на розгортанні та інтеграції в середовище хостингу.

Google Kubernetes Engine, контейнери GCP як сервіс (CaaS), побудований на відкритій системі Kubernetes, що дає вам гнучкість локальних або гібридних хмар, на додаток до інфраструктури публічної хмари GCP. Коли ви будуєте з двигуном Kubernetes, можна:

– створювати та керувати групами екземплярів Compute Engine, що виконують Kubernetes, звані кластерами. Двигун Kubernetes використовує екземпляри Compute Engine як вузли кластера. Кожен вузол запускає середовище виконання Docker, агент вузла Kubernetes, який контролює стан вузла, і простий мережний проксі;

— визначити вимоги до ваших контейнерів Docker, створивши простий конфігураційний файл JSON;

— використати Google Container Registry для безпечного приватного зберігання зображень Docker. Ви можете покласти зображення до реєстру, а потім витягувати зображення до будь-якого екземпляра Compute Engine або власного обладнання за допомогою кінцевої точки HTTP;

— створити одно- та багатоконтейнерні поди (pod — це найменший, найосновніший об'єкти у Kubernetes. Pod представляє один екземпляр запущеного процесу у вашому кластері.). Кожна пачка являє собою логічний хост, який може містити один або більше контейнерів. Контейнери в блоці працюють разом, обмінюючись ресурсами, такими як мережеві ресурси. У сукупності набір подів може містити цілий додаток, мікросервіс або один шар у багаторівневій програмі;

— створити та керувати “replication controllers”, які керують створенням і видаленням реплікантів на основі шаблону. Контролери реплікації допомагають забезпечити, щоб у вашій програмі були ресурси, необхідні для надійного та відповідного масштабування;

— створити та керувати сервисами. Сервіси створюють шар абстракції, який відокремлює клієнтів інтерфейсу від подів, які надають функції бекенда. Таким чином, клієнти можуть працювати без занепокоєння щодо того, які поди створюються та видаляються в будь-який момент;

— створити балансування зовнішньої мережі.

Некерованим обчислювальним сервісом GCP є Google Compute Engine. Можна вважати, що Compute Engine надає інфраструктуру як послугу (IaaS), оскільки система забезпечує надійну обчислювальну інфраструктуру, але ви повинні вибрати і налаштувати компоненти платформи, які ви хочете використовувати. Завдяки Compute Engine ваша відповідальність - налаштувати, адмініструвати та контролювати системи. Google забезпечить наявність, надійність і готовність ресурсів для використання, але ви повинні приготувати їх і керувати ними. Перевага тут полягає в тому, що ви маєте повний контроль над системами і необмежену гнучкість. Коли ви будуєте на Compute Engine, ви можете:

— використовувати віртуальні машини (VMs), які називаються екземплярами, щоб побудувати свою програму, подібно до того, як це було б, якщо б ви мали власну апаратну інфраструктуру. Ви можете вибрати з різних

типів екземплярів, щоб налаштувати конфігурацію відповідно до ваших потреб і бюджету;

— обрати, які глобальні регіони та зони розгортатимуть свої ресурси, надаючи вам контроль над тим, де зберігаються й використовуються ваші дані;

— обрати, які операційні системи, стеки розробки, мови, фреймворки, служби та інші технології програмного забезпечення ви віддаєте перевагу;

— створити екземпляри із відкритих або приватних зображень;

— використати технології зберігання даних GCP або будь-які інші технології, які ви віддаєте перевагу;

— використати Google Cloud Launcher для швидкого розгортання попередньо настроєних програмних пакетів;

— створити групи екземплярів, щоб легше керувати кількома екземплярами разом;

— використати автоматичне масштабування з групою екземплярів для автоматичного додавання та видалення можливостей;

— при необхідності приєднати та від'єднати диски;

— використати SSH для підключення безпосередньо до своїх екземплярів.

Також GCP дозволяє використовувати різні сервіси, які можуть бути корисними для вашого додатку — Google Maps API, Big Query для зберігання бізнес аналітики, що є дуже необхідним для e-commerce додатків, Google Pub/Sub для асинхронного спілкування між вашими мікросервісами та багато інших.

1.6.2 Amazon Web Services (AWS)

Amazon Web Services (AWS) є одним із лідерів на ринку IaaS (інфраструктура як сервіс) і PaaS (платформа як сервіс) для хмарних

екосистем пов'язані з наданням інфраструктури (обчислення, зберігання та мережа) та управління.

За допомогою AWS ви можете вибрати конкретні рішення, які вам потрібні, і платити тільки за те, що ви використовуєте, що призводить до зниження капітальних витрат і швидшого часу для оцінки, не жертвуючи при цьому продуктивністю програми або користувачем.

Amazon пропонує цілий універсал продуктів і послуг для побудови або розширення вашого середовища в хмарі. А саме створення віртуальних машин (екземпляри EC2), зберігання даних (S3), машинне навчання, розробка ігор і багато іншого (більше 8000 сервісів). І на основі подібності між цими послугами, вони класифікуються в різні розділи консолі AWS. Опишемо Compute сервіси:

— EC2 (Elastic Compute Cloud): їх можна розглядати як віртуальні машини, які можна побудувати всередині облачної платформи AWS. А AWS не обмежує вас віртуальними машин, ви також можете створювати фізичні виділені машини за допомогою сервісу EC2;

— EC2 Container Service (ECS): Ця послуга дозволяє легко запускати, масштабувати та захищати програми Docker на платформі AWS;

— Elastic Beanstalk: З сервісом Elastic Beanstalk, розробники, які не розуміють AWS концепції можуть завантажувати свої веб-додатків, а потім Elastic Beanstalk сервіс автоматично обробить розгортання, балансування навантаження, буде автоматично масштабуватися і моніторити здоров'я програми;

— Lambda: За допомогою сервісу AWS Lambda ви можете завантажити свій код і запустити його на платформі AWS. При цьому вам не потрібно керувати серверами, віртуальними машинами або будь-якими базовими речами. Таким чином, це легка і дуже популярна послуга серед громади. В даний час ця послуга дозволяє коди, написані на мовах Node.js, Java, C #, Go і Python;

— Lightsail: Lightsail — віртуальна приватна послуга Amazon (служба VPS). Це означає, що він призначений для звичайних користувачів, які не розуміють нічого про концепції AWS. Таким чином, у пакеті пропонуються віртуальна машина, пам'ять, управління DNS і статичний IP. Але в інших службах, таких як EC2, буде надаємо екземпляр EC2 і решта функцій буде встановлена і налаштована;

— Batch: за допомогою Batch сервісу програміст може виконувати сотні або тисячі пакетних обчислень у хмарі AWS.

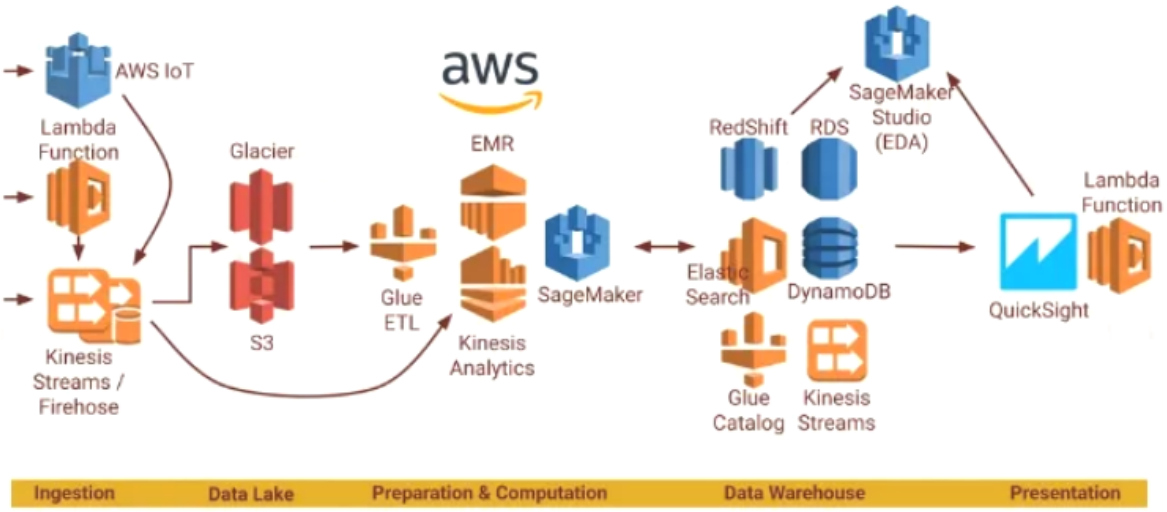


Рисунок 1.9 – Сервіси, що надаються AWS

Розглянемо наявні сервіси бази даних:

— RDS (служба реляційних баз даних): за допомогою цієї служби ви можете отримати реляційні бази даних, такі як MySQL, MSSQL, PostgreSQL, Oracle, Arora (версія MySQL від Amazon) і т.д;

— DynamoDB: це нереляційна служба баз даних (NoSQL). Це повністю керована хмарна база даних, яка підтримує моделі зберігання документів і ключів;

— Elasticache: Це служба кешування доступна для баз даних. Це допомагає покращити продуктивність веб-додатків, отримуючи інформацію з кешу, замість того, щоб повністю покладатися на більш повільні бази даних

на дисках. Таким чином, ви можете кешувати важливі дані в кеш і використовувати цю послугу для підвищення продуктивності.

— Red Shift: За допомогою служби Red Shift ви можете зберігати ваші дані та використовувати інструменти бізнес-аналітики для ефективного аналізу вартості даних. Таким чином, першим кроком до створення сховища даних є запуск набору вузлів, які називаються кластером Amazon Redshift.

Таким чином, як AWS так і GCP інтегратори надають багато різноманітних сервісів, які задовільняють вимогам більшості додатків, у тому числі і ecommerce.

У кожного з них є зручна веб-консоль завдяки якій ви можете моніторити свої додатки, підключати нові сервіси, слідкувати за витраченими коштами та інше. Це є дуже зручним для мікросервісних додатків, тому що всі їх дані та конфігурація будуть зручно розташовані в одному місці.

1.6.3. Azure

Платформа Azure - це хмарна комп'ютерна платформа від Microsoft, яка надає різноманітні послуги для розробки, тестування, впровадження та керування додатками через Інтернет. Ось деякі особливості цієї платформи:

- Гнучкість та масштабованість: Azure дозволяє масштабувати ресурси залежно від потреб вашого бізнесу. Ви можете збільшувати або зменшувати використання ресурсів у залежності від обсягу роботи.

- Різноманітність послуг: Azure пропонує широкий спектр послуг, включаючи обчислення, зберігання, бази даних, машинне навчання, аналіз даних, Інтернет речей (IoT) та багато іншого.

- Безпека: Azure надає різні інструменти та служби для захисту вашої інформації та додатків, включаючи інструменти керування доступом, шифрування даних, служби виявлення загроз і т. д.

- Інтеграція з існуючими системами Microsoft: Оскільки Azure розроблено Microsoft, вона добре інтегрується з іншими продуктами та

сервісами від Microsoft, такими як Office 365, Dynamics 365 та Windows Server.

- Географічна розгортка: Azure пропонує можливість розгортати додатки та зберігати дані в різних регіонах світу, що дозволяє забезпечити швидкий доступ до даних користувачів у різних частинах світу.

- Підтримка відкритих стандартів: Azure підтримує відкриті стандарти та мови програмування, такі як Java, Python, Node.js, PHP, та інші.

- Платіж за фактом використання: Azure пропонує гнучку систему ціноутворення, де ви платите лише за ті послуги та ресурси, які ви використовуєте, що може значно зменшити витрати на IT-інфраструктуру.

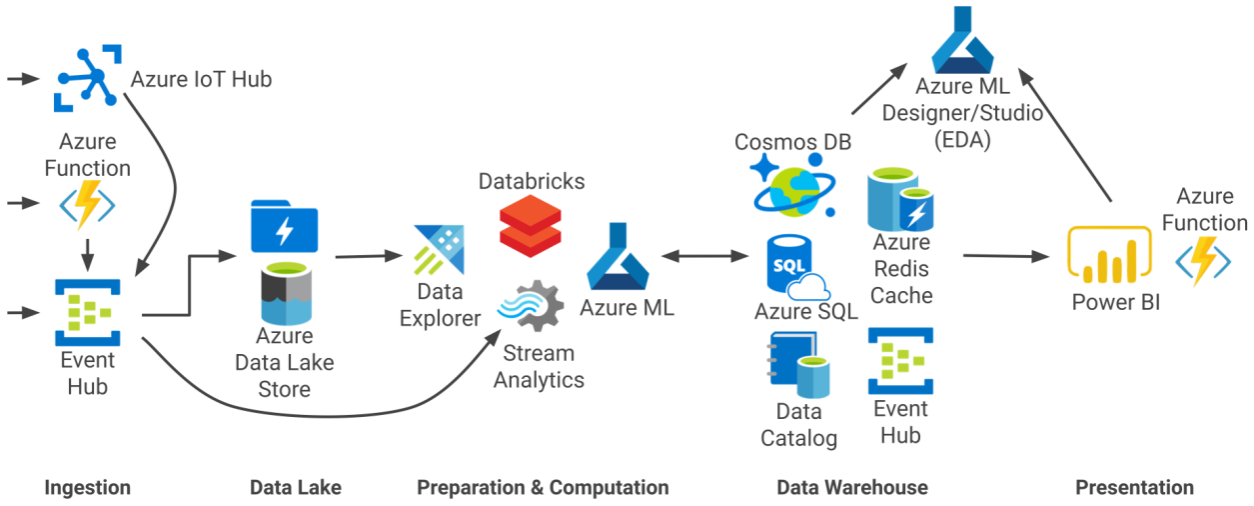


Рисунок 1.10 – Сервіси Azure

Платформа надає трохи менше, ніж AWS. Та її головна перевага — широка інтеграція з продуктами материнської компанії та різноманітні партнерські програми (VMware, NetApp, Red Hat, Cray, Databricks і так далі). Також в Azure активно розвиваються напрями Edge computing та IoT.

За даними Gartner, до недоліків платформи часто відносять зауваження клієнтів на періодичні проблеми з SLA у критичних сервісах, складнощі під

час укладання комерційних контрактів та недостатню якість платної технічної підтримки.

У підсумку можна зазначити, що в існуючих рішень є свої переваги, але вони існують на ринку дуже давно і не використовують сучасні підходи в хмарних технологіях, тому хмарний сервіс продажу товарів та послуг через мережу Інтернет використовуючи сучасні підходи у створенні програмного забезпечення на базі хмарних технологій може бути конкурентним.

2. ПРОЕКТУВАННЯ СТРУКТУРИ ТА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ХМАРНОГО СЕРВІСУ ПРОДАЖУ ТОВАРІВ

2.1. Розробка функціональних та нефункцональних вимог

2.1.1. Функціональні вимоги

Клієнтська частина програмного забезпечення розділена на дві складові – портал адміністратора та сайт інтернет магазину. Відповідні моделі вимог зображено на рисунках 2.1 і 2.2.



Рисунок 2.1 – Модель вимог до розробки сайту Інтернет магазину на основі хмарних рішень

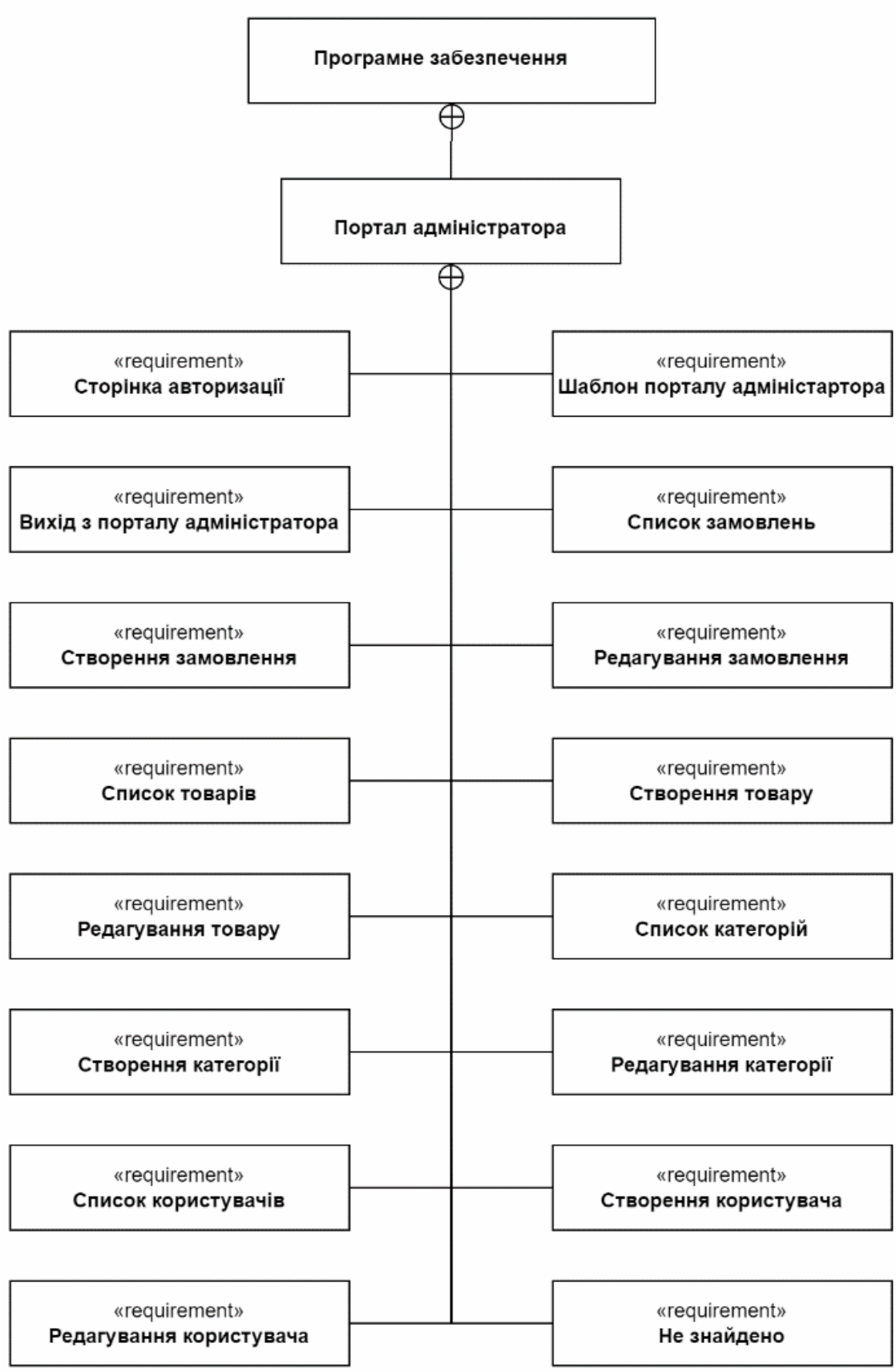


Рисунок 2.2 – Модель вимог до розробки порталу адміністратора на основі хмарних рішень

2.1.2. Нефункціональні вимоги

Реалізувати нефункціональні вимоги, які наведено нижче:

– Розробка окремого сайту інтернет магазину для кожного замовника з використанням існуючого функціоналу серверної частини та порталу адміністратора.

– Оновлення та доповнення функціоналу без необхідності зупиняти роботу додатку.

– Автоматичне відновлення системи при порушеннях роботи системи.

– Швидкість кожної дії на сайті в межах однієї секунди.

– Автоматичне масштабування при динамічній зміні навантаження.

– Розгортання окремої копії додатку для окремого замовника з використанням окремого проекту Google Cloud з окремими базами даних.

– Контроль введення інформації та захист від некоректних дій користувача.

– Цілісність інформації в кожній базі даних окремо.

2.2. Моделювання та аналіз програмного забезпечення

Для опису основного бізнес процесу програмного забезпечення, а саме оформлення замовлення використовується BPMN модель зображена на рисунку 2.3.

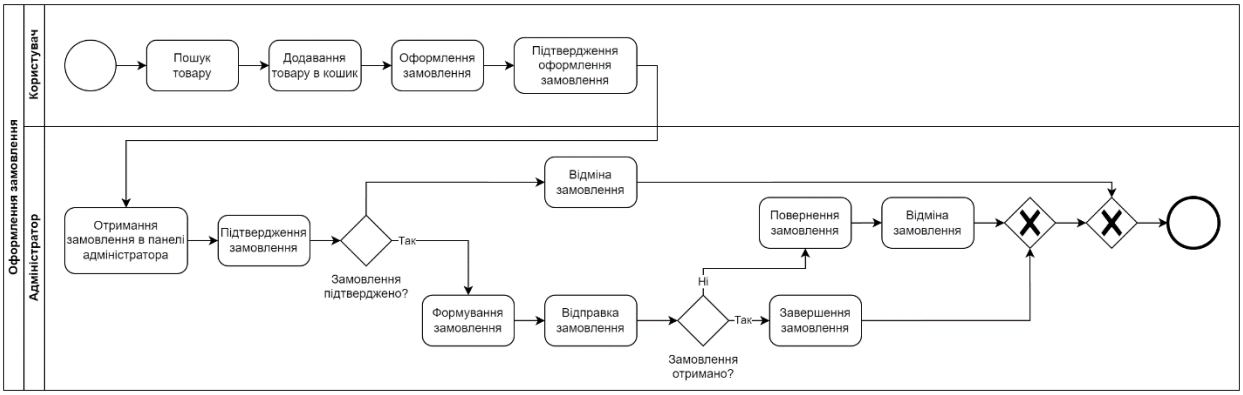


Рисунок 2.3 – BPMN модель оформлення замовлення

Опис послідовності оформлення замовлення користувачем на сайті інтернет магазину наведено нижче.

– Користувач заходить на сторінку сайту інтернет магазину.

– Користувач шукає бажаний товар.

– Користувач додає бажаний товар до кошика.

– Користувач оформлює замовлення.

– Користувач отримує підтвердження замовлення.

Опис послідовності роботи з оформленим замовленням адміністратором на порталі адміністратора наведено нижче:

– Адміністратор отримує замовлення на порталі адміністратора.

– Адміністратор підтверджує замовлення.

Якщо замовлення не підтверджено то виконуються кроки нижче.

– Адміністратор відміняє замовлення.

Якщо замовлення підтверджено то виконуються кроки нижче.

– Адміністратор формує замовлення.

– Відправка замовлення.

Якщо замовлення отримано то виконуються кроки нижче.

– Адміністратор завершує замовлення.

Якщо замовлення не отримано то виконуються кроки нижче.

– Замовлення повертається.

– Адміністратор відміняє замовлення.

2.3. Архітектура програмного забезпечення хмарного сервісу продажу товарів

Діаграма архітектури розроблюваного хмарного сервісу у вигляді схеми структурна компонентів програмного забезпечення представлена на рисунку 2.4.

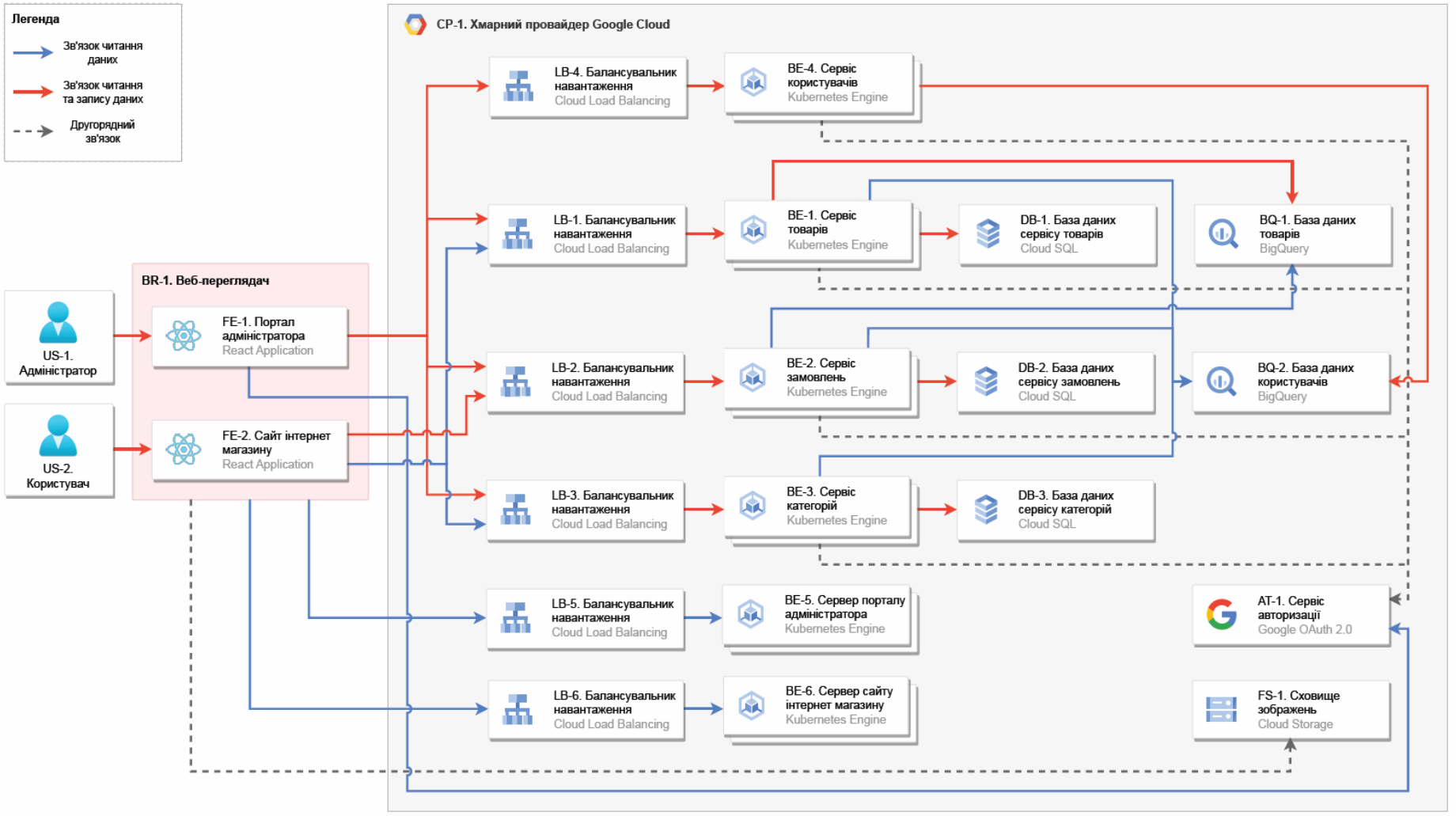


Рисунок 2.4 - Структурна схема (діаграма) компонентів програмного забезпечення

Розглянемо кожен компонент діаграми архітектури окремо.

US–1 – адміністратор, зареєстрований користувач в порталі адміністратора інтернет магазину, який може редагувати товари та категорії, користувачі, та працювати з замовленнями користувачів відповідно до його дозволів.

US–2 – користувач сайту інтернет магазину, що може переглядати товари та категорії, та оформлювати замовлення.

BR–1 – веб–переглядач, це програма установлена на пристрої користувача в якому запускаються веб–сторінки.

FE–1 – односторінковий веб–застосунок порталу адміністратора написаний з використанням бібліотеки React, який запускається у веб–переглядачі.

FE–2 – односторінковий веб–застосунок сайту інтернет магазину написаний з використанням бібліотеки React, який запускається у веб–переглядачі.

CP–1 – хмарна платформа Google Cloud, що використовується для розгортання програмного забезпечення CloudSales.

LB–1 – LB–5 – це балансування навантаження Cloud Load Balancing, які розподіляють навантаження між репліками серверних сервісів. Кожен сервіс має свій окремий балансування навантаження.

BE–1 – серверний сервіс керування товарами, який обробляє REST виклики, як для порталу адміністратора так і для сайту інтернет магазину. Має приватний для порталу адміністратора та публічний для сайту API.

BE–2 – серверний сервіс керування категоріями, який обробляє REST виклики, як для порталу адміністратора так і для сайту інтернет магазину. Має приватний для порталу адміністратора та публічний для сайту API.

BE–3 – серверний сервіс керування замовленнями, який обробляє REST виклики, як для порталу адміністратора так і для сайту інтернет магазину. Має приватний для порталу адміністратора та публічний для сайту API.

BE–4 – серверний сервіс керування користувачами, який обробляє REST виклики порталу адміністратора, та має виключно приватний API.

BE–5 – сервер порталу адміністратора інтернет магазину, який дозволяє завантажити односторінковий застосунок порталу адміністратора.

BE–6 – сервер сайту інтернет магазину, який дозволяє завантажити односторінковий застосунок сайту.

BE–1 – BE–6 використовують Google Kubernetes Engine, який дозволяє розмножувати кількість реплік сервісів в залежності від навантаження.

DB–1 – база даних сервісу товарів PostgreSQL розгорнута на Google Cloud SQL використовується для зберігання детальної інформації про товари сервісом товарів.

DB–2 – база даних сервісу замовлень PostgreSQL розгорнута на Google Cloud SQL використовується для зберігання детальної інформації про замовлення сервісом замовлень.

DB–3 – база даних сервісу категорій PostgreSQL розгорнута на Google Cloud SQL використовується для зберігання детальної інформації про категорії товарів сервісом категорій.

BQ–1 – база даних товарів в, якій зберігається ціна товару з прив’язкою до ідентифікатора товару, необхідна для уточнення ціни товару сервісом замовлень, щоб виключити пряму взаємодію між сервісом замовлень та сервісом товарів.

BQ–2 – база даних користувачів та дозволів користувачів панелі адміністратора, необхідна для уточнення дозволів на виконання дій в серверних сервісах, щоб виключити пряму взаємодію між серверними сервісами та сервісом користувачів.

BQ–1 та BQ–2 розгорнуті з використанням BigQuery, NoSQL бази даних, яка дозволяє швидкий запис та читання даних з вбудованим балансуванням навантаження.

AT–1 – сервіс авторизації Google OAuth 2.0, що використовується для авторизації користувачів в панелі адміністратора.

FS–1 – сховище зображень побудоване на сервісі Cloud Storage, яке використовується, як публічне сховище зображень товарів та категорій.

2.4. Розробка діаграм послідовностей хмарного сервісу

2.4.1. Розробка діаграми послідовності оформлення замовлення користувачем

Розглянемо детально діаграму послідовності оформлення замовлення користувачем на сайті, що зображено на рисунку 2.5.

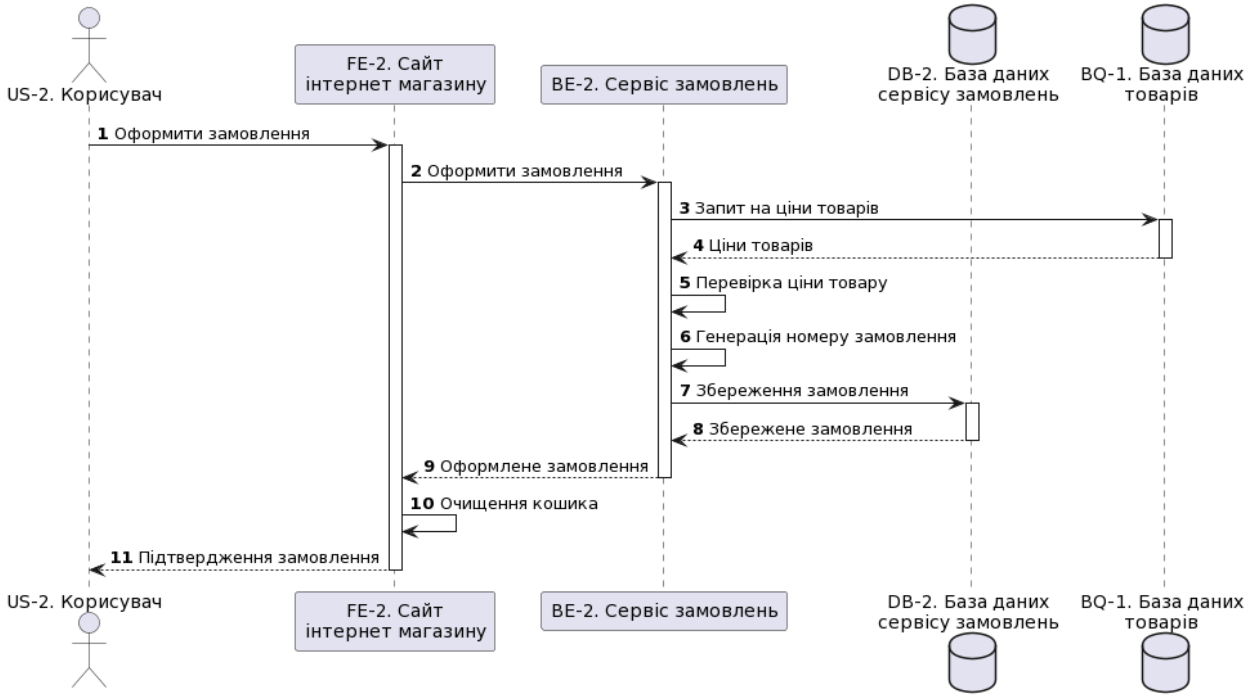


Рисунок 2.5 – Діаграма послідовності оформлення замовлення користувачем

Крок 1. Користувач (US–2) відправляє запит на сайт інтернет магазину (FE–2) на оформлення замовлення.

Крок 2. Сайт (FE–2) відправляє запит на сервіс замовлень (BE–2).

Крок 3. Сервіс замовлень (BE–2) відправляє запит до бази даних товарів (BQ–1).

Крок 4. База даних (BQ–1) надсилає сервісу (BE–2) ціни товарів.

Крок 5. Сервіс (BE–2) перевіряє ціну товару.

Крок 6. Сервіс (BE–2) генерує номер замовлення.

Крок 7. Сервіс (BE–2) зберігає замовлення у базі даних сервісу замовлень (DB–2).

Крок 8. База даних (DB–2) надсилає інформацію про збережене замовлення сервісу (BE–2).

Крок 9. Сервіс (BE–2) надсилає інформацію про оформлене замовлення сайту (FE–2).

Крок 10. Сайт (FE–2) очищає кошик.

Крок 11. Сайт (FE–2) відправляє повідомлення про підтвердження замовлення користувачеві (US–2).

2.4.2. Розробка діаграми послідовності оформлення замовлення адміністратором

Розглянемо детально діаграму послідовності оформлення замовлення адміністратором на порталі адміністратора, що зображено на рисунку 2.6.

Крок 1. Адміністратор (US–1) надсилає запит на портал адміністратора (FE–1) на перегляд замовлення.

Крок 2. Портал (FE–1) надсилає запит на сервіс замовлень (BE–2) на завантаження списку замовлень.

Крок 3. Сервіс (BE–2) надсилає запит до бази даних користувачів (BQ–2) на завантаження користувача.

Крок 4. База даних (BQ–2) надсилає інформацію про користувача на сервіс (BE–2).

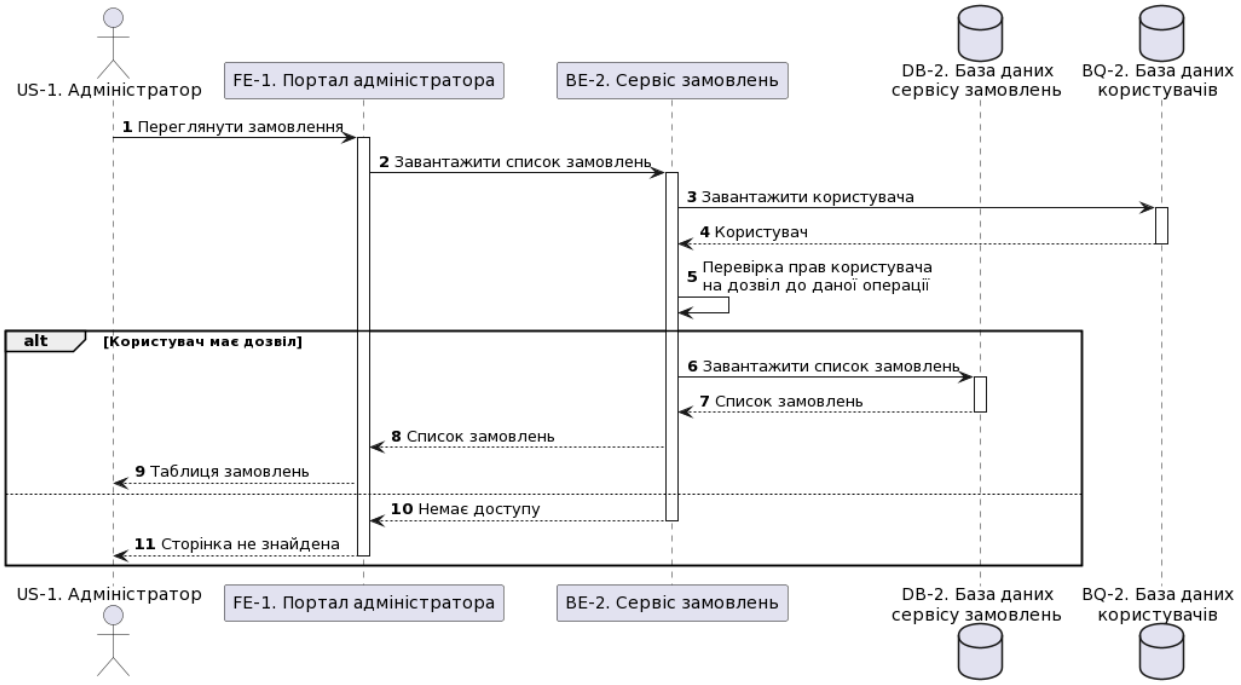


Рисунок 2.6 – Діаграма послідовності перегляду списку замовлень адміністратором

Крок 5. Сервіс (BE–2) перевіряє права користувача на дозвіл до даної операції.

Після перевірки прав користувача у випадку, якщо користувач має дозвіл відбувається послідовність крок 6 – крок 9, якщо немає – крок 10 – крок 11.

Крок 6. Сервіс (BE–2) надсилає запит до бази даних сервісу замовлень (DB–2) на завантаження списку замовлень.

Крок 7. База даних (DB–2) надсилає інформацію про список замовлень на сервіс (BE–2).

Крок 8. Сервіс (BE–2) надсилає інформацію про список замовлень на портал (FE–1).

Крок 9. Портал (FE–1) надсилає інформацію про таблицю замовлень адміністратору (US–1).

Крок 10. Сервіс (BE–2) надсилає інформацію на портал (FE–1) про відсутність доступу.

Крок 11. Портал (FE–1) надсилає повідомлення адміністратору (US–1) про відсутність сторінки.

2.4.3. Розробка діаграми послідовності редагування замовлення адміністратором

Розглянемо детально діаграму послідовності редагування замовлення адміністратором на порталі адміністратора, що зображено на рисунку 2.7.

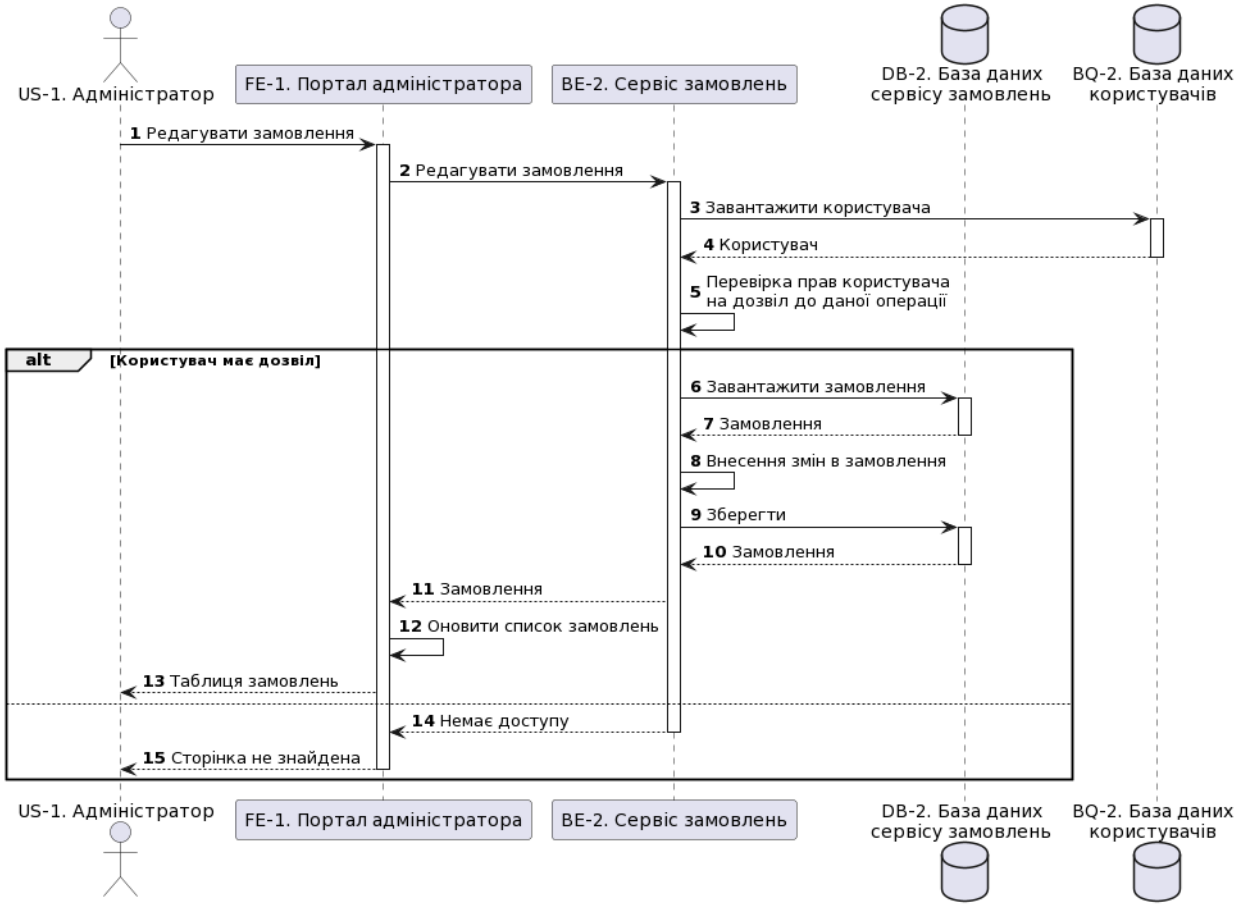


Рисунок 2.7 – Діаграма послідовності редагування замовлення адміністратором

Крок 1. Адміністратор (US–1) надсилає запит на портал адміністратора (FE–1) на редагування замовлення.

Крок 2. Портал (FE–1) надсилає запит на сервіс замовлень (BE–2) на редагування замовлення.

Крок 3. Сервіс (BE–2) надсилає запит до бази даних користувачів (BQ–2) на завантаження користувача.

Крок 4. База даних (BQ–2) надсилає інформацію про користувача на сервіс (BE–2).

Крок 5. Сервіс (BE–2) перевіряє права користувача на дозвіл до даної операції.

Після перевірки прав користувача у випадку, якщо користувач має дозвіл відбувається послідовність крок 6 – крок 13, якщо немає – крок 14 – крок 15.

Крок 6. Сервіс (BE–2) надсилає запит до бази даних сервісу замовлень (DB–2) на завантаження замовлення.

Крок 7. База даних (DB–2) надсилає інформацію про замовлення на сервіс (BE–2).

Крок 8. Адміністратор (US–1) вносить зміни у замовлення.

Крок 9. Сервіс (BE–2) зберігає замовлення у базі даних (DB–2).

Крок 10. База даних (DB–2) надсилає сервісу (BE–2) інформацію про замовлення.

Крок 11. Сервіс (BE–2) надсилає порталу (FE–1) інформацію про замовлення.

Крок 12. Портал (FE–1) оновлює список замовлень.

Крок 13. Портал (FE–1) надсилає адміністратору (US–1) таблицю замовлень з оновленою інформацією про замовлення.

Крок 14. Сервіс (BE–2) надсилає інформацію на портал (FE–1) про відсутність доступу.

Крок 15. Портал (FE–1) надсилає повідомлення адміністратору (US–1) про відсутність сторінки.

Структурою мікросервісів є рівнева архітектура, тому розглянемо детальніше її особливості.

1–й рівень архітектури – це контролери, вони виконують роль посередника між моделлю та представленням, обробляючи запити користувача та керуючи потоком даних між ними. Так як було обрано платформу Spring, то для забезпечення роботи даного рівня буде використовуватися Spring Web MVC, як один із ключових модулів даної платформи. Він дозволяє ефективно керувати потоком даних, розділяти бізнес–логіку та представлення, а також підтримує стандарти REST, які були обрані для комунікації клієнтської та серверної частини програмного забезпечення.

2–й рівень архітектури – це бізнес–логіка, що у веб–додатку визначає основні правила та функціональність, які визначають, як додаток працює з даними та виконує бізнес–процеси. Вона описує логіку та правила, які визначають, які дії можуть бути виконані з даними та які перетворення або обчислення потрібно застосувати.

3–й рівень архітектури – це репозиторії, які відповідають за доступ до даних, що знаходяться у базах даних або інших джерелах даних. В даній роботі це буде реалізовано через ORM – це технологія програмування, яка дозволяє розробникам звертатися до бази даних з використанням об'єктно–орієнтованого підходу. ORM забезпечує автоматичне відображення даних з бази даних на об'єкти в програмі та навпаки, що дозволяє зберігати та отримувати дані без необхідності написання багато коду SQL.

2.5. Розробка моделі бази даних

Для розробки і функціонування сервісу необхідно спроектувати п’ять баз даних, такі як:

- база даних товарів DB–1,
- база даних замовлень DB–2,
- база даних категорій DB–3,

- база даних товарів BQ–1,
- база даних користувачів BQ–2.

База даних категорій представлена у вигляді таблиці на рисунку 2.8, а опис таблиці “categories” наведено у таблиці 2.1.

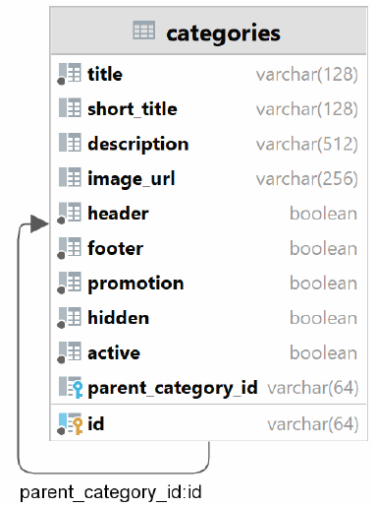


Рисунок 2.8 – Структура БД категорій

Таблиця 2.6 – Опис полів таблиці “categories”

Назва поля	Тип даних	Опис
id	varchar(64)	Ідентифікатор категорії
title	varchar(128)	Назва категорії
short_title	varchar(128)	Коротка назва категорії
description	varchar(512)	Опис категорії
image_url	varchar(256)	Ідентифікатор зображення категорії
header	boolean	Відображення категорії у верхньому колонтитулі
footer	boolean	Відображення категорії у нижньому колонтитулі
promotion	boolean	Відображення категорії у вигляді акційної категорії

hidden	boolean	Категорія прихована в загальному списку категорій
active	boolean	Активна категорія
parent_category_id	varchar(64)	Ідентифікатор категорії вищого рівня

База даних товарів представлена у вигляді таблиць на рисунку 2.9. Опис таблиць “products”, “product_categories” та “product_images” наведено у таблицях 2.2 – 2.4.

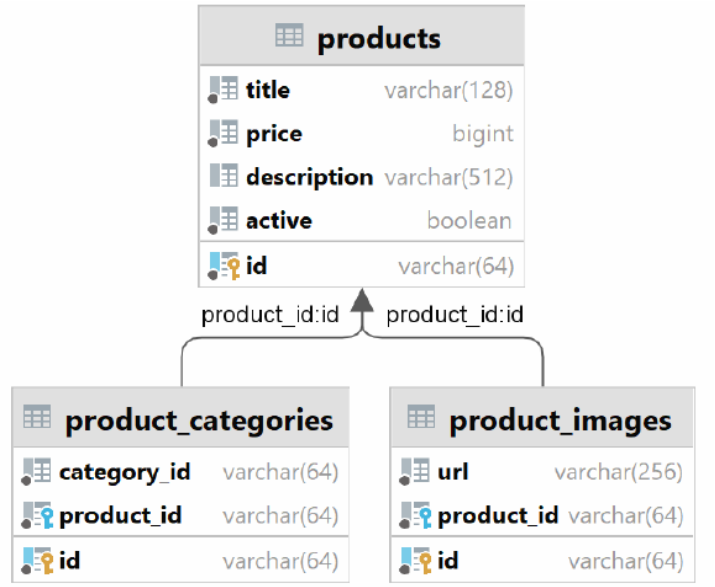


Рисунок 2.9 – Структура БД товарів

Таблиця 2.2 – Опис полів таблиці “products”

Назва поля	Тип даних	Опис
id	varchar(64)	Ідентифікатор товару
title	varchar(128)	Назва товару
price	bigint	Ціна товару
description	varchar(512)	Опис товару
active	boolean	Стан товару

Таблиця 2.3 – Опис полів таблиці “product_categories”

Назва поля	Тип даних	Опис
id	varchar(64)	Ідентифікатор товару
category_id	varchar(64)	Ідентифікатор категорії
product_id	varchar(64)	Прив’язка категорії до товару

Таблиця 2.4 – Опис полів таблиці “product_images”

Назва поля	Тип даних	Опис
id	varchar(64)	Ідентифікатор товару
url	varchar(256)	Ідентифікатор зображення
product_id	varchar(64)	Прив’язка зображення до товару

База даних замовлень представлена у вигляді таблиць на рисунку 2.10, а опис таблиць “orders”, та “order_entries” наведено у таблицях 2.5 – 2.6.

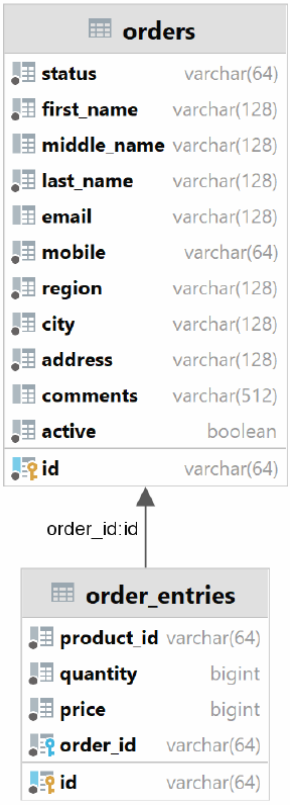


Рисунок 2.10 – Структура БД замовлень

Таблиця 2.5 – Опис полів таблиці “orders”

Назва поля	**Тип даних**	**Опис**
id	varchar(64)	Ідентифікатор замовлення
status	varchar(64)	Статус замовлення
first_name	varchar(128)	Ім’я замовника
middle_name	varchar(128)	Прізвище замовника
last_name	varchar(128)	По батькові замовника
email	varchar(128)	Електронна адреса замовника
mobile	varchar(64)	Мобільний телефон замовника
region	varchar(128)	Регіон замовника
city	varchar(128)	Місто замовника
address	varchar(128)	Адреса замовника
comments	varchar(512)	Коментарі до замовлення
active	boolean	Стан замовлення

Таблиця 2.6 – Опис полів таблиці “order_entries”

Назва поля	**Тип даних**	**Опис**
id	varchar(64)	Ідентифікатор замовлення
product_id	varchar(64)	Ідентифікатор товару замовлення
quantity	bigint	Кількість товарів у замовленні
price	bigint	Ціна товарів у замовленні
order_id	varchar(64)	Прив’язка товару, кількості та ціни до замовлення

Отже, було виконано проектування структури та алгоритмічного забезпечення хмарного сервісу продажу товарів, розроблено діаграми послідовностей, архітектуру системи та виконано побудову моделі бази даних.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО СЕРВІСУ ПРОДАЖУ ТОВАРІВ ТА ПОСЛУГ ЧЕРЕЗ МЕРЕЖУ ІНТЕРНЕТ

3.1. Розробка діаграм варіантів використання

Головною функцією програмного забезпечення є продаж товарів через Інтернет магазин. Складовими частинами даної функції є оформлення замовлення користувачем та опрацювання замовлення адміністратором. Далі розглянемо окремо вимоги до двох складових Інтернет магазину: портал адміністратора та сайт інтернет магазину.

3.1.1 Діаграма варіантів використання Інтернет магазину

Діаграма варіантів використання сайту інтернет магазину подано на рисунку 3.1.

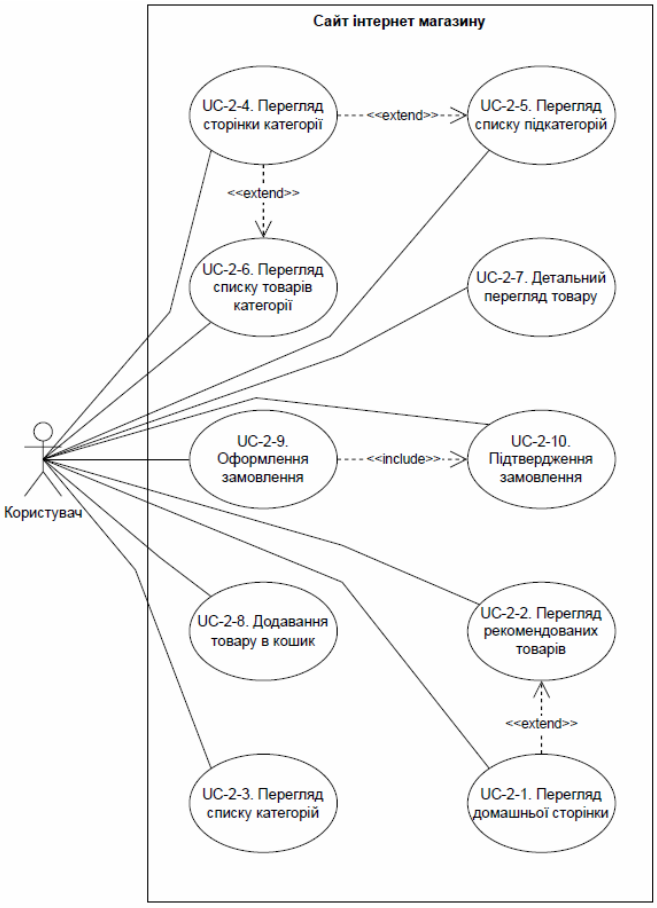


Рисунок 3.1 – Діаграми варіантів використання для Інтернет магазину

В таблицях нижче наведені деякі варіанти використання сайту інтернет магазину.

Таблиця 3.1 – Варіант використання “Перегляд рекомендованих товарів”

Ідентифікатор	UC–2–2
Ціль	Переглянути рекомендовані товари
Дійова особа	Користувач
Причина	Користувач бажає переглянути рекомендовані товари
Початкові умови	Відкрита сторінка сайту інтернет магазину
Послідовність подій	Користувач переходить на домашню сторінку та переходить до рекомендованих товарів.
Кінцеві умови	Відкрита сторінка списку рекомендованих товарів

Таблиця 3.2 – Варіант використання “Детальний перегляд товару”

Ідентифікатор	UC–2–7
Ціль	Переглянути детальну інформацію про товар
Дійова особа	Користувач
Причина	Користувач бажає переглянути детальну інформацію про товар
Початкові умови	Відкрита сторінка товарів інтернет магазину
Послідовність подій	Користувач натискає на бажаний товар. Користувача перенаправляє на сторінку з детальною інформацією про товар.
Розширення	На сторінці товару можна переглянути його детальний опис
Кінцеві умови	Відкрита сторінка товару інтернет магазину

Таблиця 3.3 – Варіант використання “Додавання товару в кошик”

Ідентифікатор	UC–2–8
Ціль	Додати товар в кошик
Дійова особа	Користувач
Причина	Користувач бажає додати товар в кошик
Початкові умови	Відкрита сторінка товару інтернет магазину
Послідовність подій	Користувач натискає на кнопку “Add to cart”. Один товар додається в кошик або кількість відповідних товарів у кошику збільшується на один.
Кінцеві умови	Відкрита сторінка товару інтернет магазину

Таблиця 3.3 – Варіант використання “Оформлення замовлення”

Ідентифікатор	UC–2–9
Ціль	Оформити замовлення
Дійова особа	Користувач
Причина	Користувач бажає оформити замовлення
Початкові умови	Відкрите діалогове вікно кошика замовлень
Послідовність подій	Користувач натискає на кнопку “Continue to checkout”. Після цього його перенаправляє у діалогове вікно оформлення замовлення. Користувач заповнює поля необхідною інформацією. Натискає на кнопку “Checkout”.
Розширення	У випадку введення не валідних даних та натискання на кнопку “Checkout” з'являються повідомлення про помилки.
Кінцеві умови	Відкрите діалогове вікно кошика замовлень

3.1.2. Діаграма варіантів використання можливостей адміністратора

Діаграма варіантів використання порталу адміністратора представлена на рисунку 3.2.

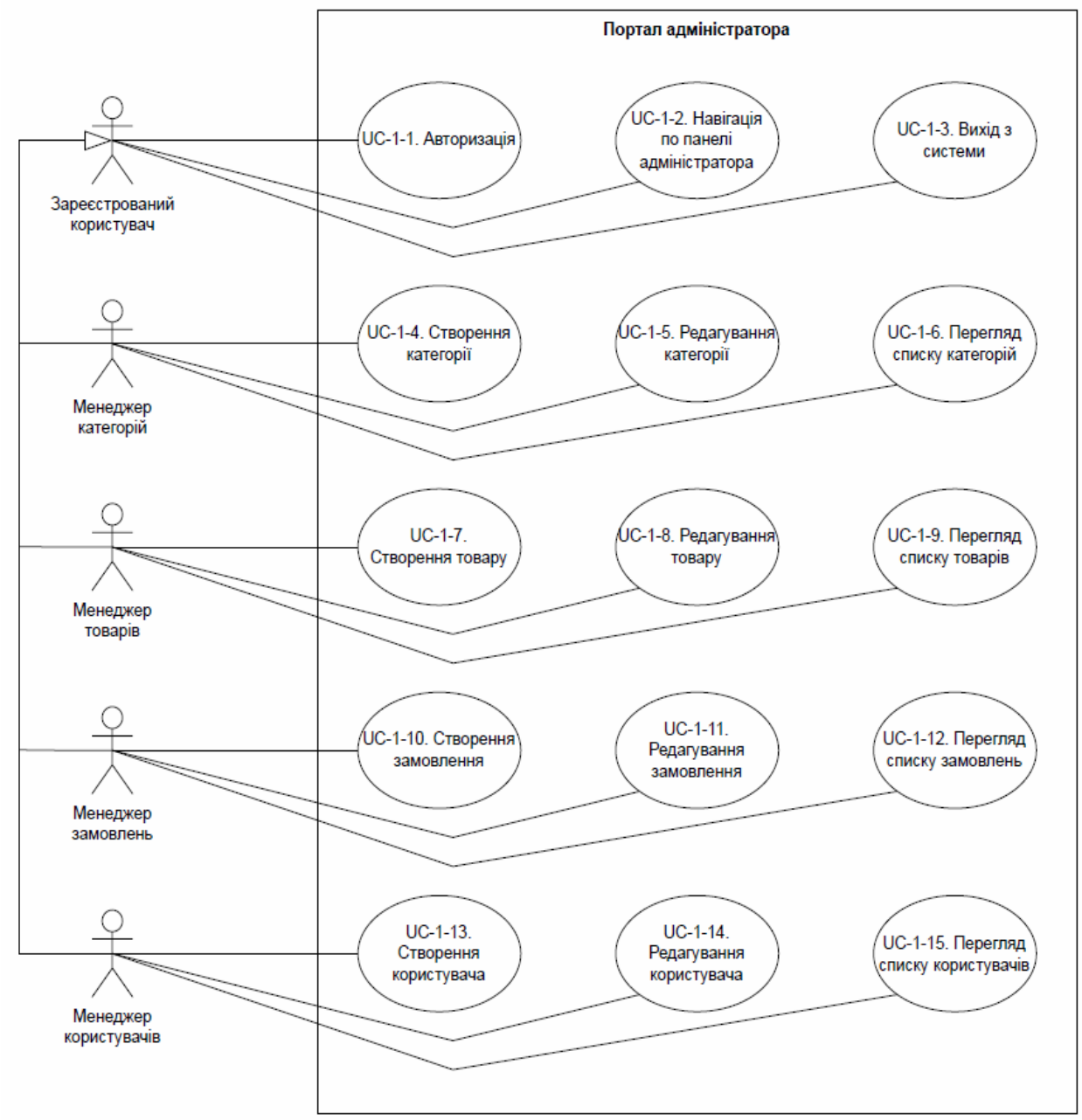


Рисунок 3.2 - Діаграма варіантів використання порталу адміністратора

В таблицях нижче наведені основні варіанти використання щодо роботи зі сторони а адміністратора.

Таблиця 3.4 – Варіант використання “Авторизація”

Ідентифікатор	UC–1–1
Ціль	Авторизація зареєстрованого користувача
Дійова особа	Зареєстрований користувач
Причина	Користувач бажає авторизуватися
Початкові умови	Відкрита сторінка авторизації
Послідовність подій	Користувач переходить на сторінку авторизації. Натискає на кнопку “Sign in with Google”, слідує інструкціям та перенаправляється на доступний розділ.
Розширення	У випадку відсутності прав доступу користувача в панелі адміністратора з’являється повідомлення: “Sorry, we haven’t found your access rights in the system. Please contact responsible person or try use another account.”
Кінцеві умови	Відкрито перший доступний розділ

Таблиця 3.5 – Варіант використання “Створення товару”

Ідентифікатор	UC–1–7
Ціль	Створити новий товар
Дійова особа	Менеджер товарів
Причина	Менеджер бажає створити новий товар
Початкові умови	Відкрита сторінка товарів порталу адміністратора
Послідовність подій	Менеджер натискає кнопку “Create new product”, йому відкривається діалогове вікно з порожніми полями для введення інформації про товар. Менеджер вносить інформацію про товар та натискає кнопку “Save”.
Кінцеві умови	Відкрита сторінка товарів

Таблиця 3.6 – Варіант використання “Створення замовлення”

Ідентифікатор	UC–1–10
Ціль	Створити нове замовлення
Дійова особа	Менеджер замовлень
Причина	Менеджер бажає створити нове замовлення
Початкові умови	Відкрита сторінка замовлень порталу адміністратора
Послідовність подій	Менеджер натискає кнопку “Create new order”, йому відкривається діалогове вікно з порожніми полями для введення інформації про замовлення. Менеджер вносить інформацію про замовлення та натискає кнопку “Save”.
Розширення	–
Кінцеві умови	Відкрита сторінка замовлень порталу адміністратора

Таблиця 3.7 – Варіант використання “Створення користувача”

Ідентифікатор	UC–1–13
Ціль	Створити нового користувача
Дійова особа	Менеджер користувачів
Причина	Менеджер бажає створити нового користувача
Початкові умови	Відкрита сторінка користувачів порталу адміністратора
Послідовність подій	Менеджер натискає кнопку “Create new user”, йому відкривається діалогове вікно з порожніми полями для введення інформації про користувача. Менеджер вносить інформацію про користувача та натискає кнопку “Save”.
Кінцеві умови	Відкрита сторінка користувачів порталу адміністратора

Таблиця 3.8 – Варіант використання “Створення категорії”

Ідентифікатор	UC–1–4
Ціль	Створити нову категорію товарів
Дійова особа	Менеджер категорій
Причина	Менеджер бажає створити нову категорію
Початкові умови	Відкрита сторінка категорій порталу адміністратора
Послідовність подій	Менеджер натискає кнопку “Create new category”, йому відкривається діалогове вікно з порожніми полями для введення інформації про категорію. Менеджер вносить інформацію про категорію та натискає кнопку “Save”.
Кінцеві умови	Відкрита сторінка категорій порталу адміністратора

Таблиця 3.9 – Варіант використання “Вихід з системи”

Ідентифікатор	UC–1–3
Ціль	Вийти з порталу адміністратора
Дійова особа	Зареєстрований користувач
Причина	Користувач бажає вийти з порталу адміністратора
Початкові умови	Відкрита будь–яка сторінка порталу адміністратора
Послідовність подій	Користувач натискає на іконку користувача та в випадаючому меню натискає кнопку “Sign out”. Інвалідується авторизація користувача. Користувач перенаправляється на сторінку авторизації.
Кінцеві умови	Відкрита сторінка авторизації

3.2. Реалізація вимог до інтерфейсу користувача панелі адміністратора

Програмне забезпечення повинно забезпечувати виконання наступних основних функції. Основні сторінки для адміністратора зображено нижче.

Користувач зареєстрований в порталі адміністратора має можливість авторизації в порталі адміністратора. Сторінка авторизації зображена на рисунку 3.3 відкривається при переході користувача за посиланням порталу адміністратора. На сторінці авторизації присутня кнопка “Sign in with Google” і при натисканні на неї користувач з правами доступу в порталі адміністратора перенаправляється на перший доступний розділ порталу адміністратора.

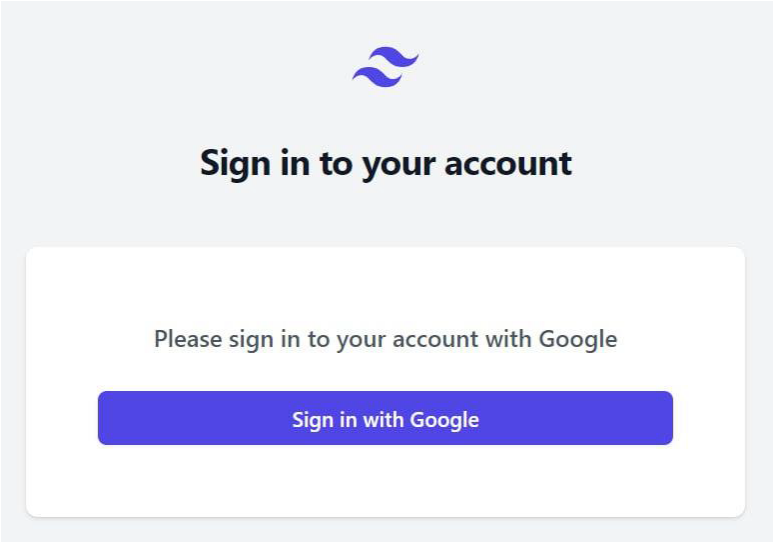


Рисунок 3.3 – Сторінка авторизації

При натисканні на кнопку “Sign in with Google” для користувача без прав доступу в порталі адміністратора з’являється повідомлення: “Sorry, we haven’t find your access rights in the system. Please contact responsible person or try use another account”.

Користувач авторизований на порталі адміністратора має можливість вийти з порталі адміністратора. Меню користувача реалізовано відповідно до прототипу зображеного на рисунку 3.4, а при натисканні на кнопку “Sign out” авторизований користувач перенаправляється на сторінку авторизації.

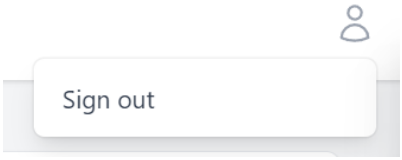


Рисунок 3.4 – Меню користувача для виходу з системи

Користувач авторизований в порталі адміністратора має можливість навігації. Меню розділів порталу адміністратора зображено на рисунку 3.5 і користувач з правами доступу “Orders Manager”, при натисканні на пункт меню “Orders”, перенаправляється на розділ “Orders”, відповідно для користувача без прав доступу “Orders Manager” пункт меню “Orders” відсутній.

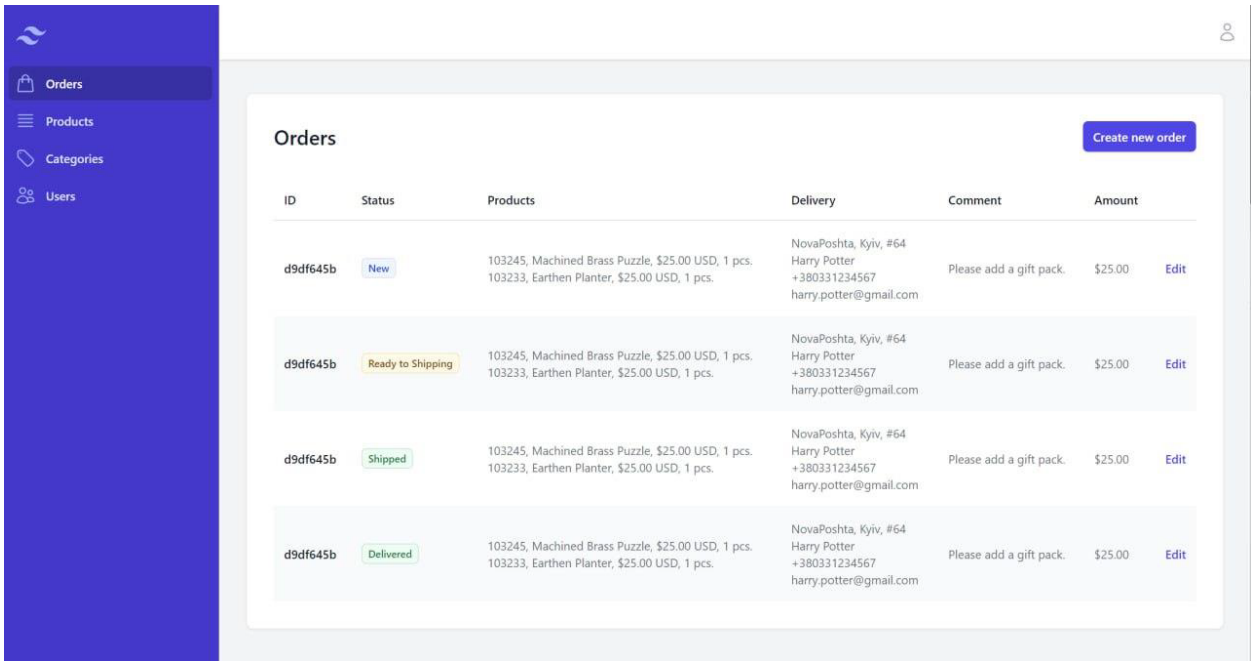


Рисунок 3.5 – Сторінка із переліком замовлень

Користувач з правами доступу “Products Manager”, при натисканні на пункт меню “Products”, перенаправляється на розділ “Products” , а для користувача без прав доступу “Products Manager” пункт меню “Products” відсутній. Користувач з правами доступу “Users Manager”, при натисканні на пункт меню “Users”, перенаправляється на розділ “Users”.

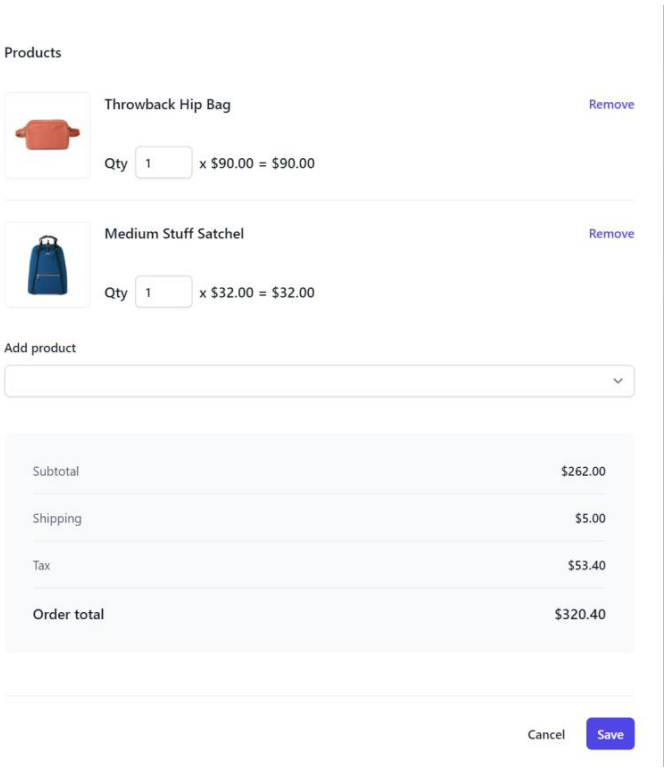


Рисунок 3.6 – Вікно створення, перегляду та редагування замовлення

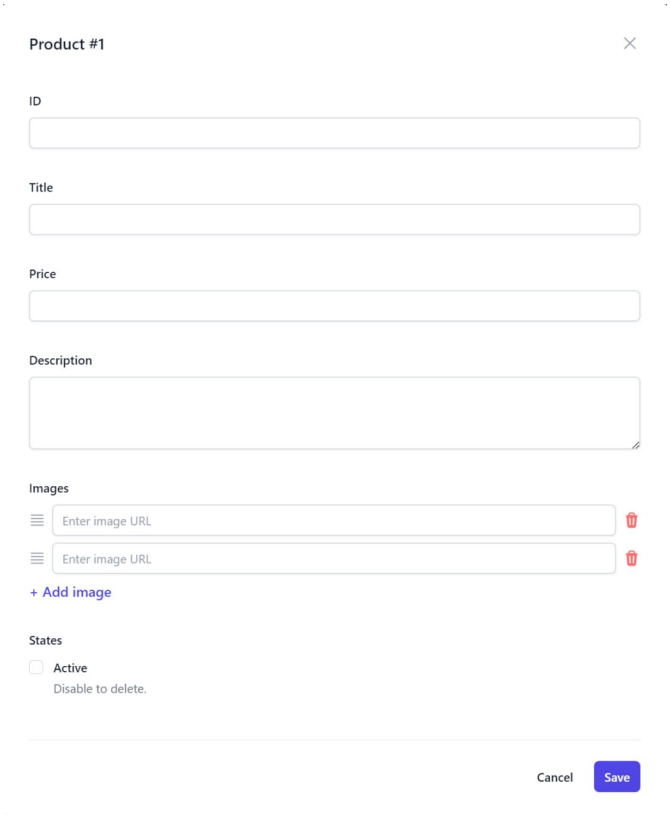


Рисунок 3.7 – Вікно створення, перегляду та редагування товару

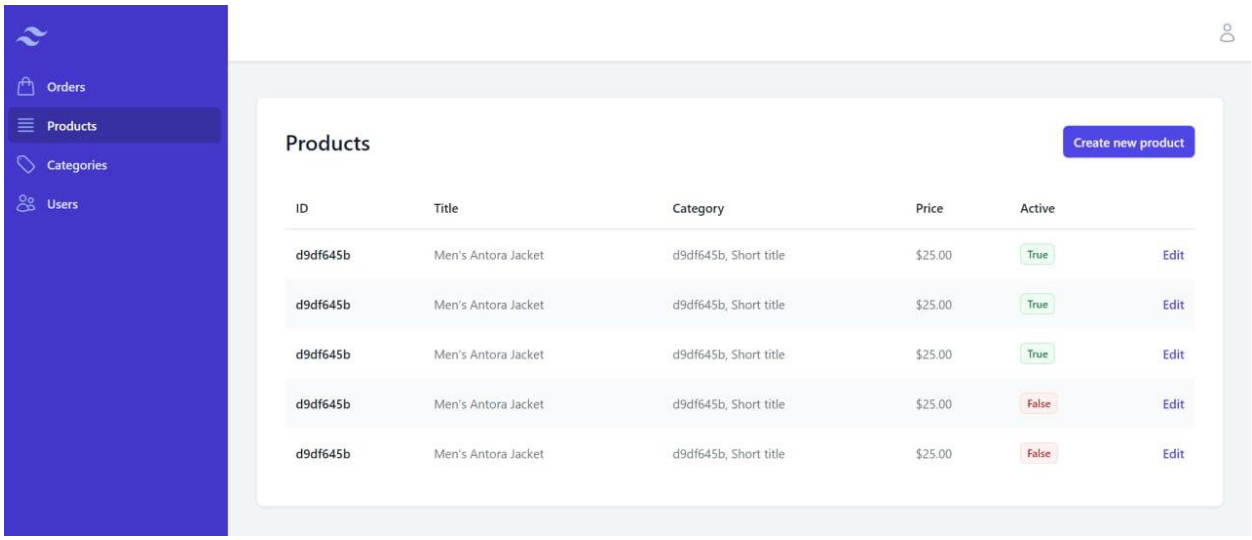


Рисунок 3.8 – Вигляд сторінки товарів

Користувач має право переглядати товарів, а також список товарів. Сторінку з товарами зображено на рисунку 3.8. Користувач з правами доступу “Products Manager” в розділі “Products” бачить табличку з наступними полями “ID”, “Title”, “Category”, “Price”, “Active”. При натисканні на кнопку “Edit” відкривається діалогове вікно редагування товару, а при натисканні на кнопку “Create new product” відкривається діалогове вікно створення товару.

Менеджер товарів має можливість створювати товар, діалогове вікно створення товару зображено на рисунку 3.9. У діалоговому вікні створення товару присутні поля “ID”, “Title”, “Price”, “Description”; поле Images, для додавання зображень товару. Поле Images має кнопку “Add image”, яка додає поле для введення URL зображення, а також поле Images має кнопки видалення та переміщення зображення.

У діалоговому вікні створення товару присутнє поле States для позначення стану товару, кнопка “Cancel” скасовує усі зміни та закриває діалогове вікно. У діалоговому вікні створення товару присутня кнопка “Save”, що зберігає всі зміни та закриває діалогове вікно.

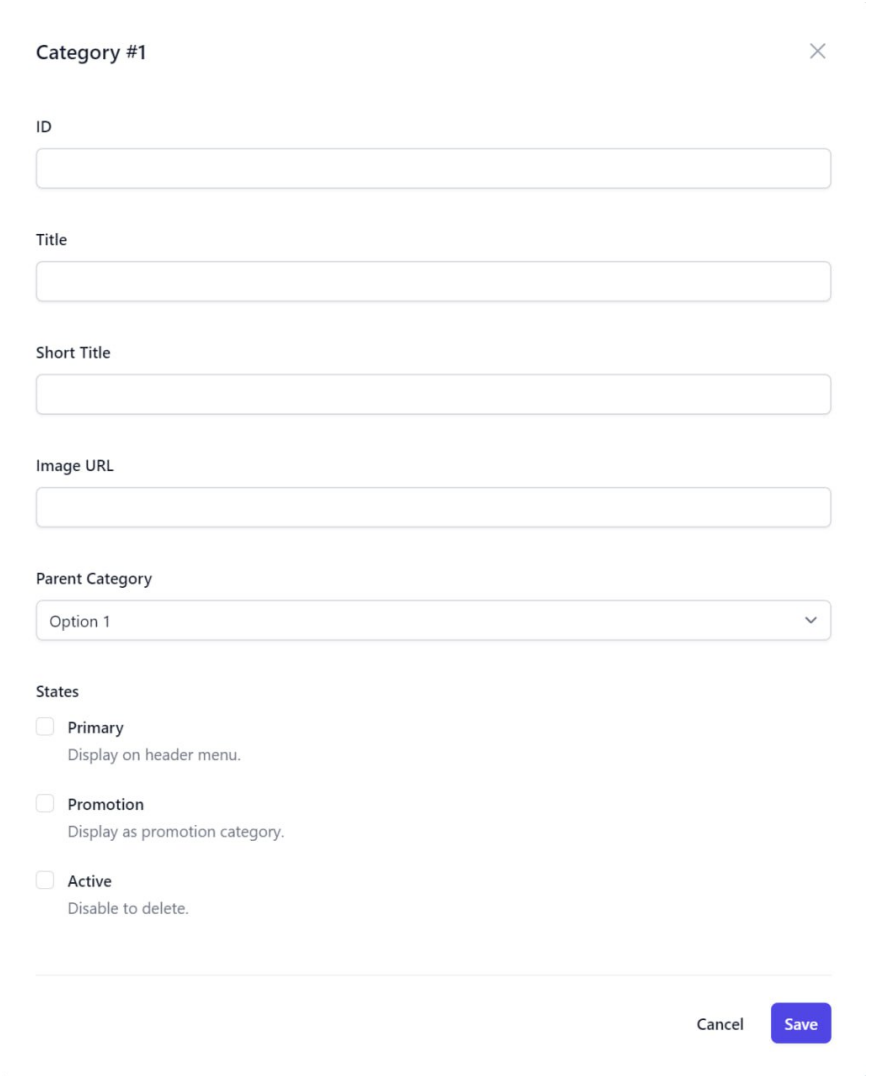


Рисунок 3.9 – Вікно створення, перегляду та редагування категорії

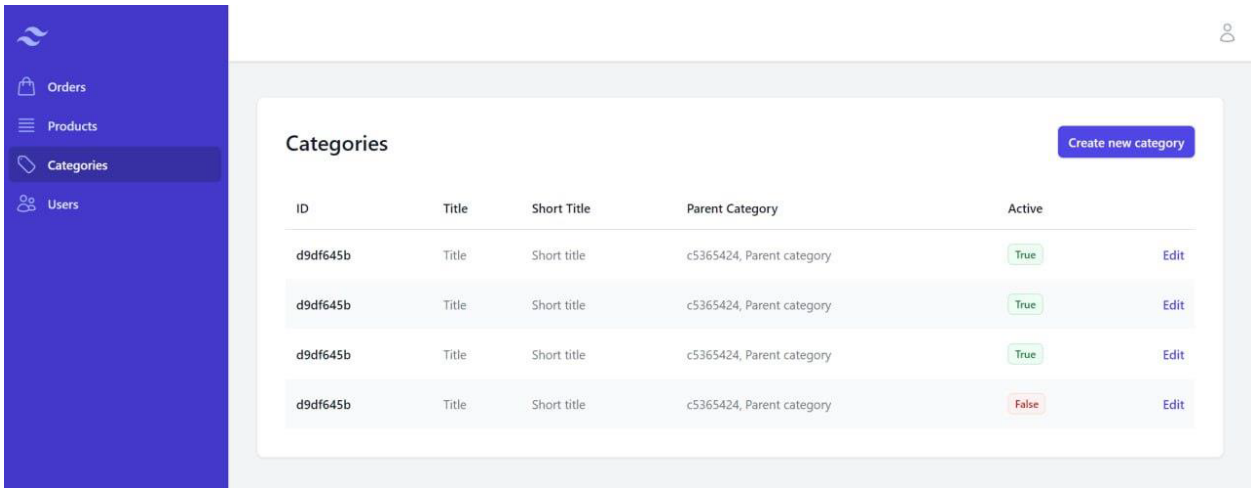


Рисунок 3.10 – Сторінка категорій

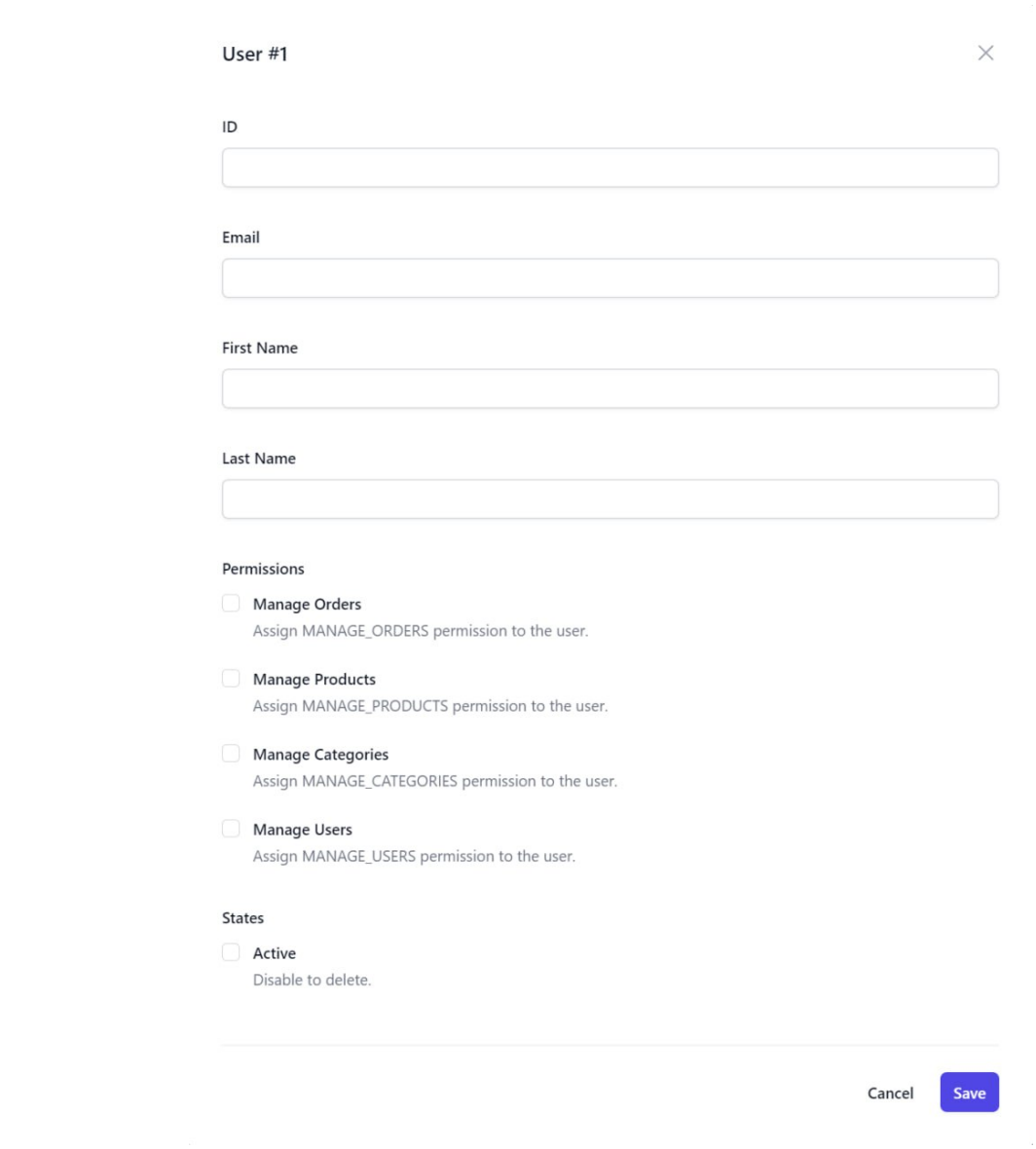


Рисунок 3.11 – Вікно створення, перегляду та редагування користувача

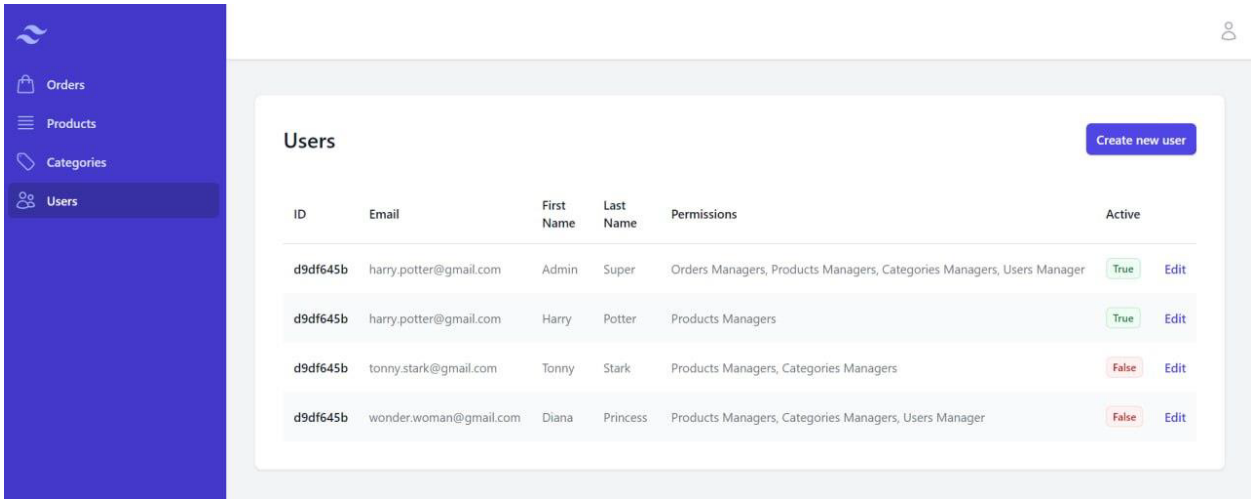


Рисунок 3.12 – Сторінка користувачів

3.3. Розробка інтерфейс системи для користувача Інтернет магазину

Основні сторінки з описами сайту Інтернет магазину зображено на рисунках, що подано нижче.

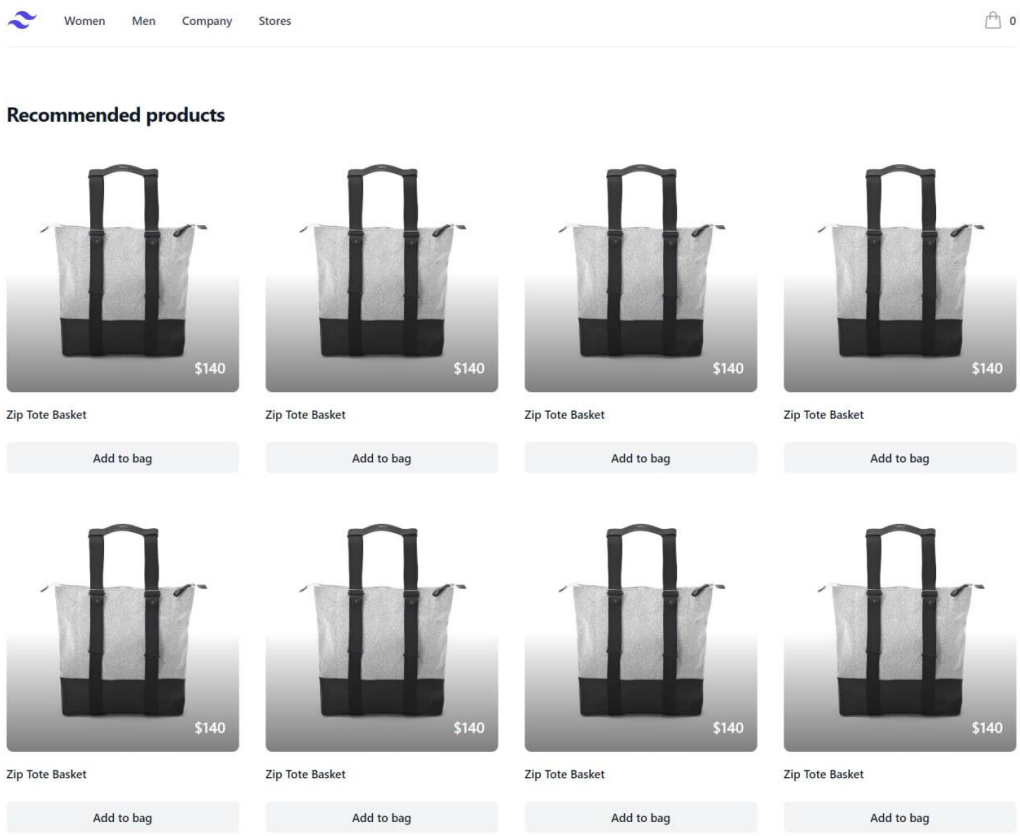


Рисунок 3.13 – Макет головної сторінки Інтернет магазину

На головній сторінці (рис. 3.13) знаходяться рекомендовані товари, при натисканні на картинку товару, відкривається сторінка відповідного товару, а при натисканні на заголовок товару, відкривається сторінка відповідного товару.

У верхньому колонтитулі сайту інтернет магазину знаходяться кнопки категорій, в яких порожнє поле Parent та активний прапорець Primary. При наведені курсора миші на кнопку категорії відкривається діалогове вікно зі

списком під категорій (рис. 3.14). При натисканні на кнопку категорії, відкривається сторінка категорії.

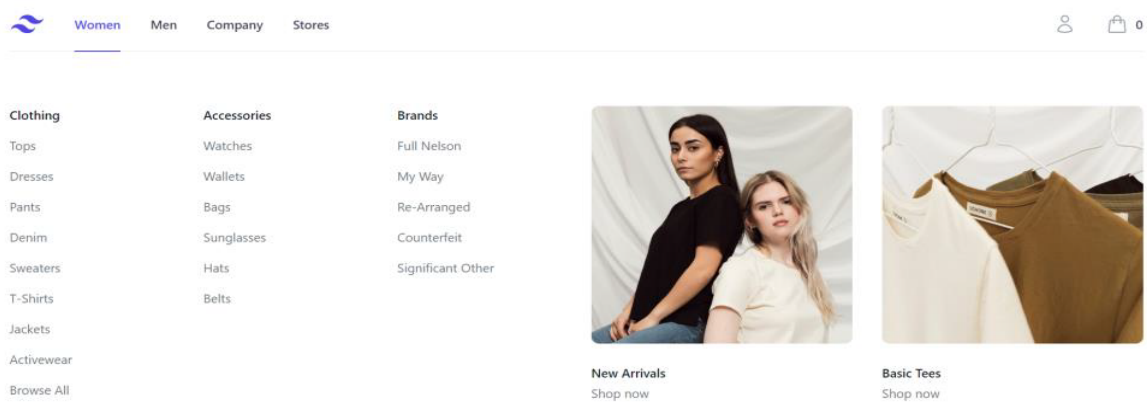


Рисунок 3.14 – Сторінка з переліком категорій

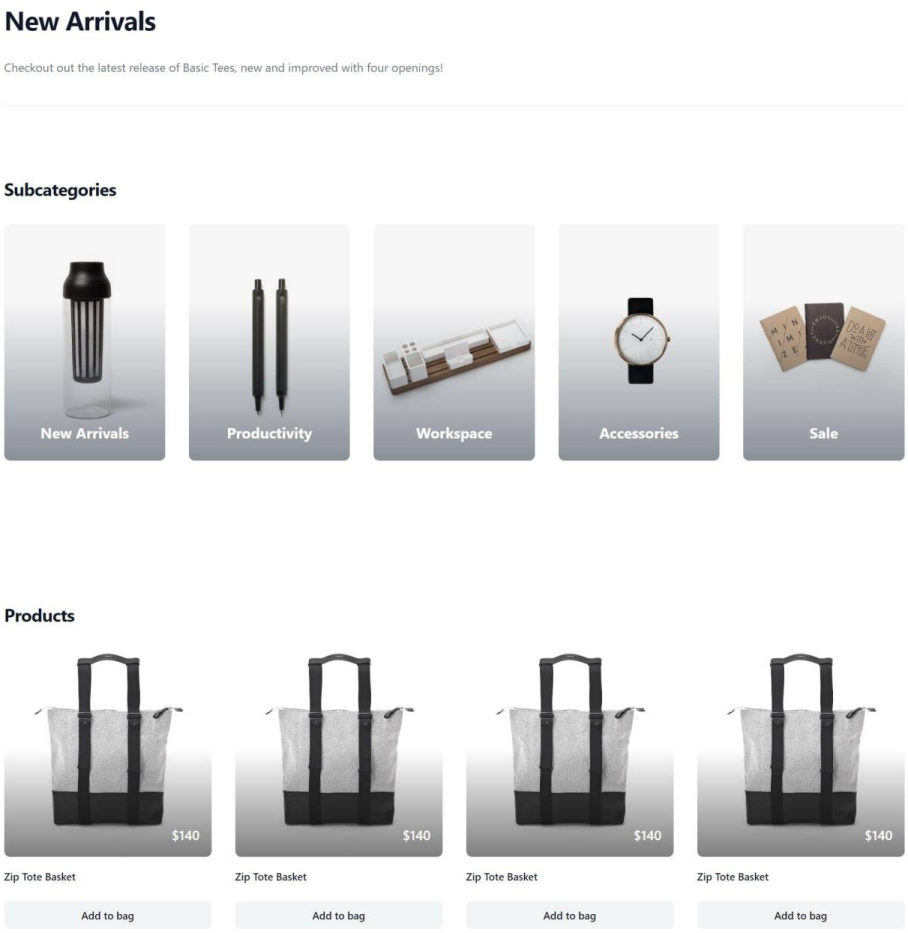


Рисунок 3.15 – Вигляд сторінки категорії

Користувач має можливість відкрити категорію та переглянути дочірні категорії та товари даної категорії та всіх дочірніх категорій на сторінці категорії сайту інтернет магазину.

Сторінка із категоріями показана на рисунку 3.15 де на сторінці категорії відображаються всі під категорії і відповідно на сторінці категорії відображаються товари даної категорії та всіх підкатегорій.

Користувач має можливість оформити замовлення товарів в кошику на сайті інтернет магазину. Діалогове вікно показано на рисунках 3.16 та 3.17. При відсутності товарів у кошику кнопка “Continue to checkout” не активна.

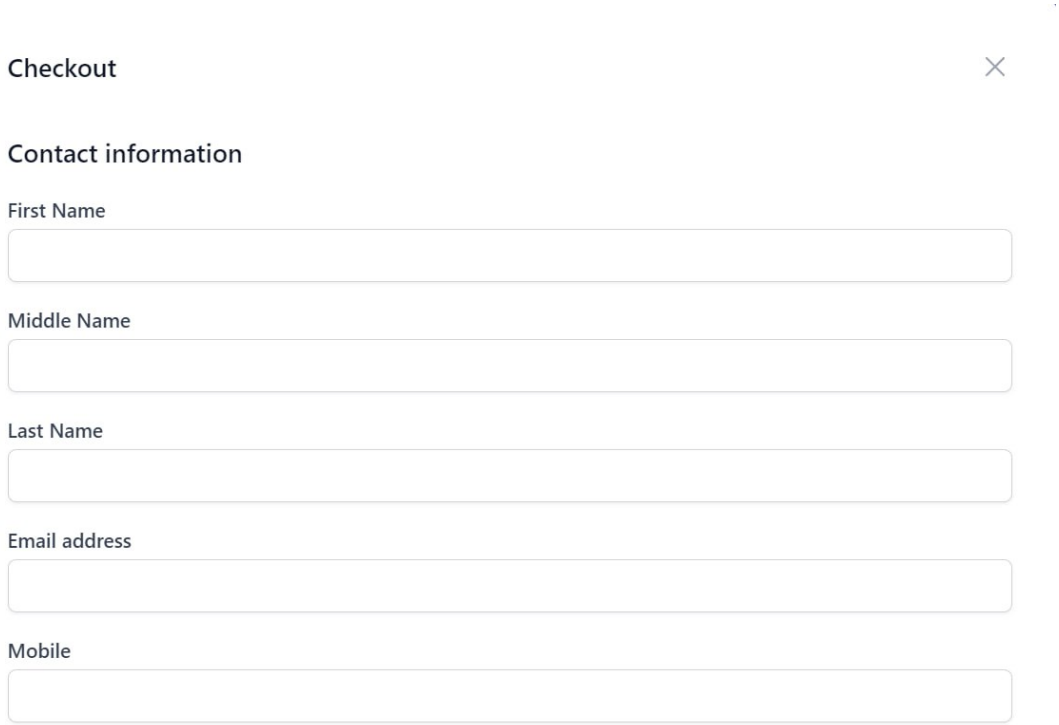


Рисунок 3.16 – Сторінка оформлення замовлення

У діалоговому вікні оформлення замовлення присутні поля Contact information такі як: First Name, Middle Name, Last Name, Email address, Mobile; Delivery information такі як: Region, City, Address, Additional information. Також тут присутня загальна вартість товарів, вартість доставки, податок та загальна сума замовлення.

У вікні оформлення замовлення присутня кнопка “Back” для повернення до кошика та кнопка “Checkout” для оформлення замовлення.

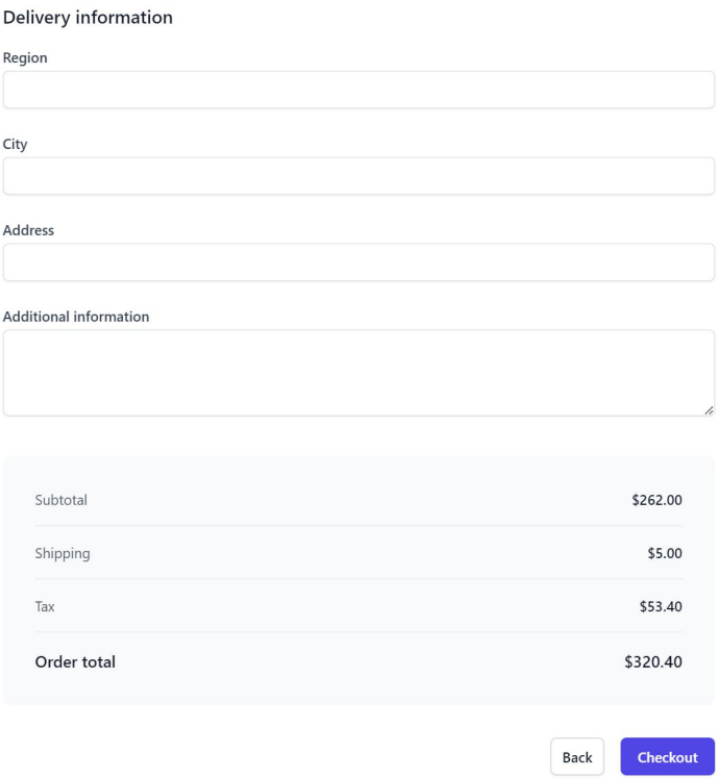


Рисунок 3.17 – Вигляд сторінки оформлення замовлення (адреса доставки і сума замовлення)

При введені в поле “First Name” тексту довжиною більше 128 символів та натисканні кнопки “Save” відображається червона рамка навколо поля вводу та текст “Value is to long” . Якщо порожнє поле “First Name” та відбувається натискання кнопки “Save” відображається червона рамка навколо поля вводу та текст “Value is required”.

При введені в поля “Middle Name”, “Last Name”, “Email address” тексту довжиною більше 128 символів та натисканні кнопки “Save” відображається червона рамка навколо поля вводу та текст “Value is to long”;

При введені в поле “Mobile” тексту довжиною більше 64 символів та натисканні кнопки “Save” відображається червона рамка навколо поля вводу

та текст “Value is to long” і при введенні в поле “Mobile” некоректного формату даних та натисканні кнопки “Save” відображається червона рамка навколо поля вводу та текст “Value format is incorrect”;

Якщо в поле “Additional information” ввести текст довжиною більше 512 символів та натиснути кнопку “Save” то відображається червона рамка навколо поля вводу та текст “Value is to long”.

Користувач має можливість додати товар в кошик на сайті інтернет магазину. Для цього у верхньому колонтитулі реалізовано значок кошика з кількістю товарів в кошику, а діалогове вікно зображеного на рисунку 3.18.

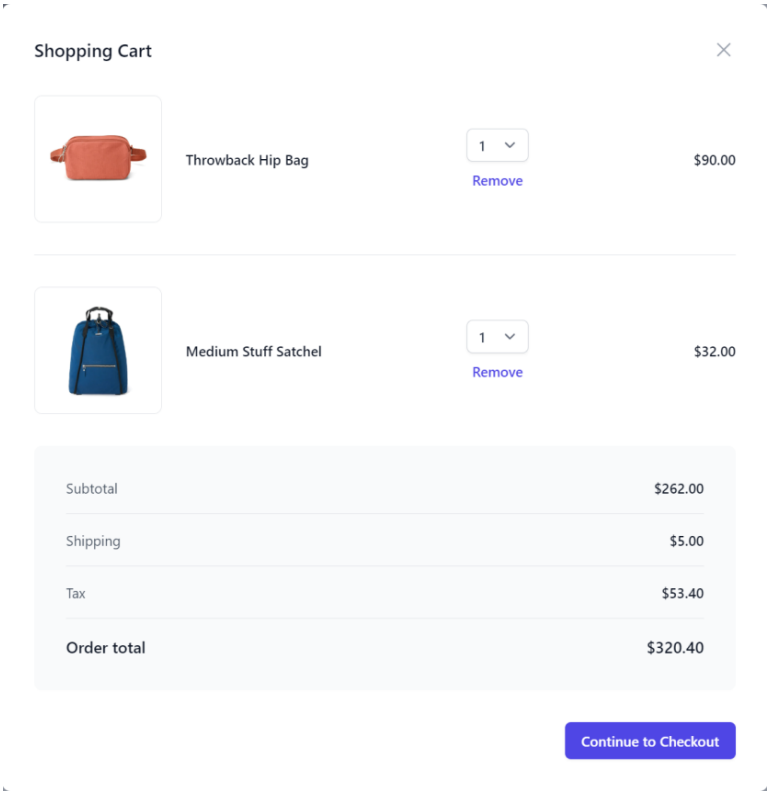


Рисунок 3.18 – Сторінка з товарами (кошик)

При додаванні товару у кошик на значку у верхньому колонтитулі кількість збільшується на один, тобто при натисканні кнопки “Add to cart” на сторінці товару, у кошик додається товар. При додаванні до кошику товару, що вже присутній у ньому, кількість товарів у кошику збільшується на один.

При натисканні на кошик відкривається діалогове вікно з товарами доданими до нього, а у випадку відсутності товарів у кошику в діалоговому вікні кошика відображається напис “Cart is empty”. Також в нижньому вікні діалогового вікна кошика присутня кнопка “Continue to checkout” , а у випадку порожнього кошика кнопка “Continue to checkout” не активна.

При натисканні кнопки “Continue to checkout” у діалоговому вікні кошика відкривається діалогове вікно оформлення замовлення. У вікні кошика відображається ціна за одиницю товару та загальна вартість товарів у кошику.

3.4. Приклад здійснення замовлення користувачем

В якості прикладу розглянемо процес оформлення замовлення користувачем на сайті інтернет магазину. Для початку створимо декілька рекомендованих товарів на порталі адміністратора для демонстрації.

Користувач на головній сторінці сайту інтернет магазину у розділі рекомендованих товарів, що зображено на рисунку 3.19, натискає на кнопку “Add to cart” першого товару двічі та другого товару один раз.

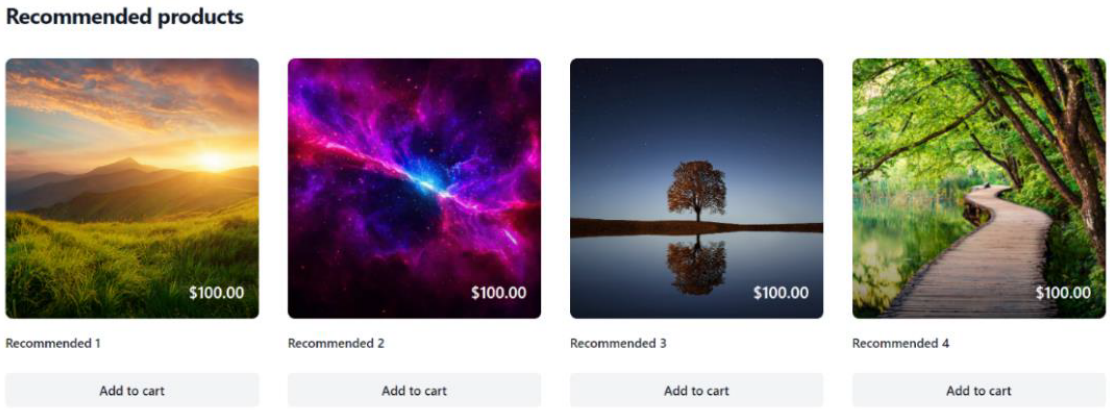


Рисунок 3.19 – Рекомендовані товари

Після цього користувач натискає на іконку кошика і йому відкривається діалогове вікно кошика, що зображено на рисунку 3.20.

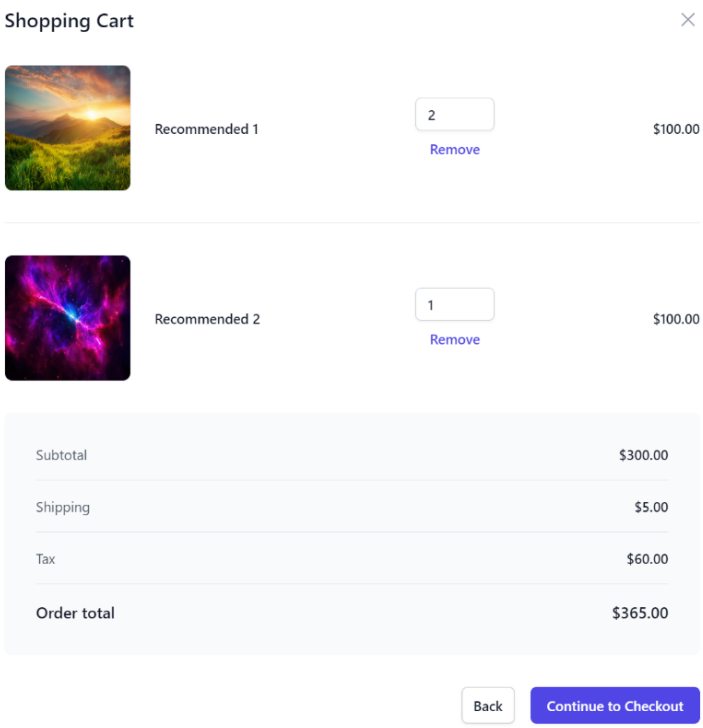


Рисунок 3.20 – Діалогове вікно кошика

У вікні натискає кнопку “Continue checkout” та його перенаправляє у діалогове вікно оформлення замовлення, що зображено на рисунках 3.21, 3.22.

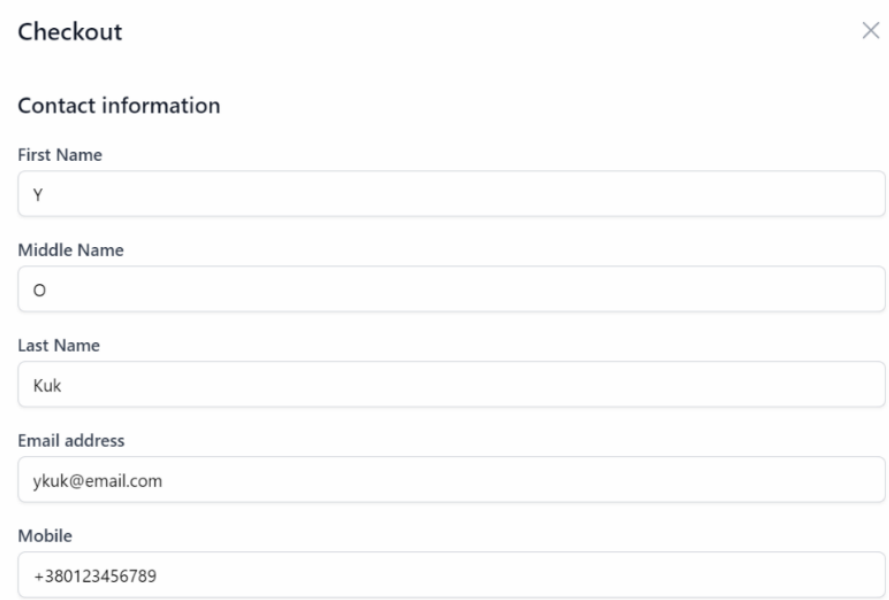


Рисунок 3.21 – Введення особистих даних для оформлення замовлення

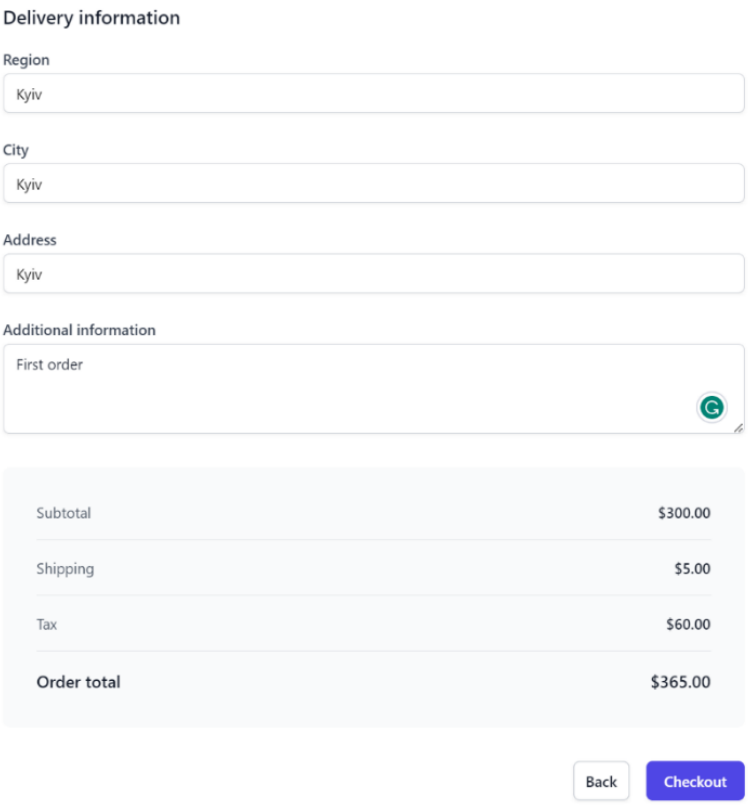


Рисунок 3.22 – Внесення даних про доставку для оформлення замовлення

У діалоговому вікні оформлення замовлення користувач заповнює всі поля. Після цього натискає кнопку “Checkout” та отримує повідомлення, що зображено на рисунку 3.23, про успішно оформлене замовлення.

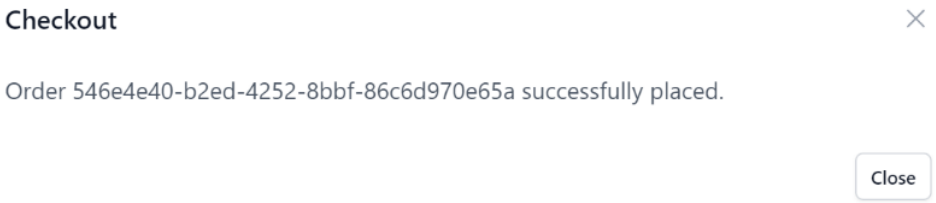


Рисунок 3.23 – Повідомлення про успішно оформлене замовлення

Після успішного оформлення замовлення користувачем на сайті інтернет магазину воно з’являється у розділі замовлень на порталі адміністратора зі статусом “New”, що зображено рисунку 3.24.

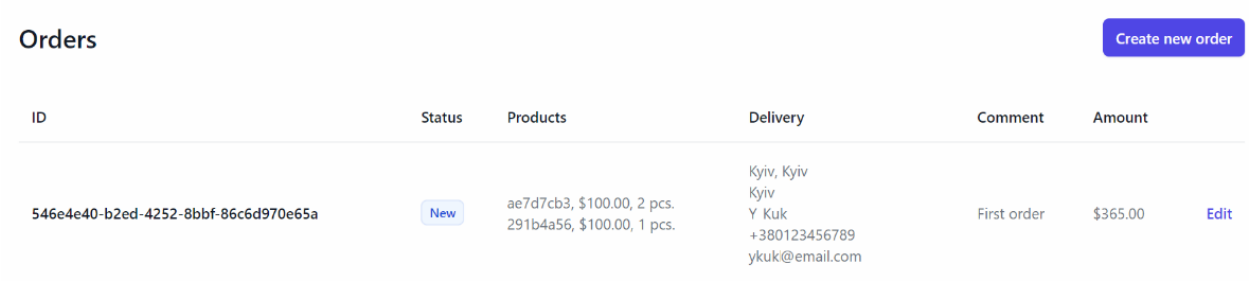


Рисунок 3.24 – Вигляд нового замовлення у розділі замовлень для панелі адміністратора

Змінюємо статус на “Delivered” для перевірки оновлення статусу замовлення в розділі замовлень порталу адміністратора та натискаємо кнопку “Save”. Після збереження змін діалогове вікно закривається, а список замовлень оновлюється. Оновлений список замовлень зображено на рисунку 3.25.

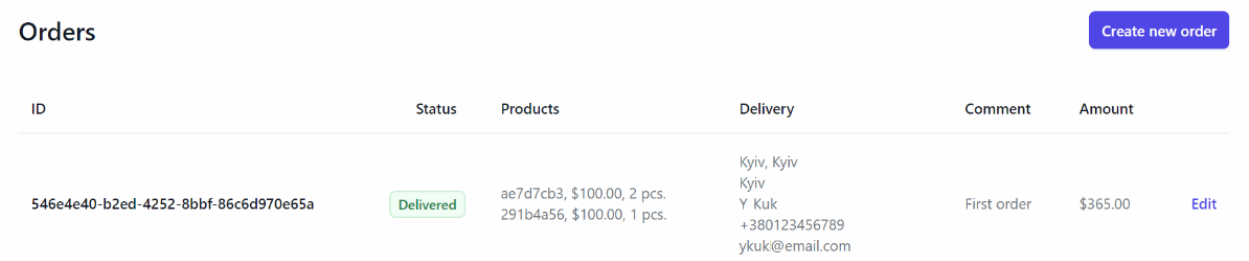


Рисунок 3.25 – Оновлений статус замовлення у розділі замовлень для панелі адміністратора

Отже, було представлено Розробка діаграм варіантів використання та наведено інтерфейс в режимі адміністратора і звичайного користувача розроблюваного хмарного сервісу продажу товарів та послуг через мережу Інтернет.

ВИСНОВКИ

В дипломній роботі проведена розробка хмарного сервісу продажу товарів та послуг через мережу Інтернет.

Розглянуто та проаналізовано галузь електронної комерції, проаналізовано існуючі проблеми електронної комерції та їх вирішення за допомогою хмарних технологій. Проаналізовано існуючі рішення, їх переваги та недоліки, розроблено та описано варіанти використання та технічні вимоги для розробки програмного забезпечення.

Головною функцією розроблюваного хмарного сервісу є продаж товарів через Інтернет магазин. Складовими частинами даної функції є оформлення замовлення користувачем та опрацювання замовлення адміністратором.

Проведено розробку архітектури хмарного сервісу і здійснено процес тестування. Виконано розгортання програмного забезпечення на хмарній платформі Google Cloud, а також локальне розгортання програмного забезпечення для підтримки. У якості бази даних икористано PostgreSQL, що розгорнуто в Google Cloud SQL та BigQuery на платформі Google Cloud.

Для реалізації цієї задачі використовуються сучасні інструменти та технології, зокрема платформу Google Cloud. Розглядаються архітектурні рішення для створення масштабованого та надійного сервісу, забезпечення безпеки та захисту даних клієнтів. Проєкт було розгорнуто на платформі Google Cloud з використанням сервісу Google Kubernetes Engine, що дозволило виконати вимоги по автоматичному масштабуванні та доповнення функціоналу без необхідності зупиняти роботу сервісу.

Завдяки використанню хмарних технологій, розроблений сервіс має великий потенціал для масштабування та гнучкості, що дозволяє підприємствам швидко реагувати на зміни в ринкових умовах та потреби споживачів.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Gary Schneide. Electronic Commerce 12th Edition. Cengage Learning, 2016. 624 p.
2. Retail e-commerce sales worldwide from 2014 to 2026. URL: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales
3. B.B. Tiwari, Bobby W. Lyall. E-Business: Issues and Challenges of 21st Century. Allied Publishers, 2020. 304 p.
4. SAP Hybris: Cloud-based e-commerce solutions. URL: https://www.sap.com/ukraine/products/acquired-brands/what-is-hybris.html.
5. Mark Richards, Neal Ford. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 419 p.
6. Tanner Larsson. Ecommerce Evolved: The Essential Playbook To Build, Grow & Scale A Successful Ecommerce Business. CreateSpace Independent Publishing Platform, 2016. 372 p.
7. Chris Richardson Choosing a Microservices Deployment Strategy // Технічний блог nginx 10.02.2016
8. What is SOA (Service-Oriented Architecture) URL: http://www.whatissoa.com
9. Oracle Documentation. Introduction to the Oracle Database. URL: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm
10. Microsoft Learn. Databases. URL: https://learn.microsoft.com/en-us/sql/relational-databases/databases/databases
11. JetBrains Documentation. IntelliJ IDEA Getting started. URL: https://www.jetbrains.com/help/idea/getting-started.html
12. Postman Learning center. Introduction. URL: https://learning.postman.com/docs/introduction/overview

13. Google Cloud. Google Cloud documentation. URL: https://cloud.google.com/docs
14. Google Cloud Documentation. What is Cloud SQL? URL: https://cloud.google.com/sql/docs/introduction
15. Kubernetes Documentation. Learn Kubernetes Basics. URL: https://kubernetes.io/docs/tutorials/kubernetes-basics
16. Google Cloud. Cloud Load Balancing Documentation. Choose a load balancer. URL: https://cloud.google.com/load-balancing/docs/choosing-load-balancer
17. Google Cloud. Google Kubernetes Engine (GKE) Documentation. URL: https://cloud.google.com/kubernetes-engine/docs
18. Google Cloud. BigQuery Documentation. URL: https://cloud.google.com/bigquery/docs
19. Google Cloud. Cloud Storage Documentation. URL: https://cloud.google.com/storage/docs
20. Kubernetes documentation // Офіційний сайт Kubernetes URL: https://kubernetes.io/docs/home/
21. Google Cloud Learn. What is a relational database? URL: https://cloud.google.com/learn/what-is-a-relational-database
22. Perforce. What Is Static Analysis? Static Code Analysis Overview. URL: https://www.perforce.com/blog/sca/what-static-analysis
23. Gradle Documentation. What is Gradle? URL: https://docs.gradle.org/current/userguide/what_is_gradle.html

ДОДАТКИ

Додаток А

Фрагменти програмних кодів

UsersController.java

```
package tech.cloudsales.users.controllers;
import tech.cloudsales.users.data.UserDto;
import java.util.List;
@RequestMapping("/api/v1/users")
public interface UsersController {
@GetMapping
@PreAuthorize("@authorizationService.hasRole(authentication,
'MANAGE_USERS')")
List<UserDto> getUsers();
@PostMapping
@PreAuthorize("@authorizationService.hasRole(authentication,
'MANAGE_USERS')")
UserDto createUser(@RequestBody UserDto userDto);
@PutMapping("/{userId}")
@PreAuthorize("@authorizationService.hasRole(authentication,
'MANAGE_USERS')")
UserDto updateUser(@RequestBody UserDto userDto, @PathVariable String
userId);
}
```

DefaultUsersController.java

```
package tech.cloudsales.users.controllers;
import lombok.AllArgsConstructor;
import tech.cloudsales.users.data.UserDto;
import tech.cloudsales.users.services.UsersService;
import java.util.List;
@RestController
@AllArgsConstructor
public class DefaultUsersController implements UsersController {
private UsersService usersService;
@Override
public List<UserDto> getUsers() {
return usersService.getUsers();
```

```
}
@Override
public UserDto createUser(UserDto userDto) {
return usersService.createUser(userDto);
}
@Override
public UserDto updateUser(UserDto userDto, String userId) {
return usersService.updateUser(userDto, userId);
}
}
```

UserDto.java

```
package tech.cloudsales.users.data;
import lombok.*;
import java.util.List;
@Getter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UserDto {
private String id;
private String email;
private String firstName;
private String lastName;
private List<String> permissions;
private boolean active;
}
```

User.java

```
package tech.cloudsales.users.model;
import lombok.*;
import java.util.List;
@Getter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User {
private String id;
```

```
private String email;
private String firstName;
private String lastName;
private List<String> permissions;
private boolean active;
}
```

UsersRepository.java

```
package tech.cloudsales.users.repositories;
import tech.cloudsales.users.model.User;
public interface UsersRepository {
List<User> findAll();
Optional<User> findByEmail(String email);
User save(User user);
}
```

BigQueryUsersRepository.java

```
package tech.cloudsales.users.repositories;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.extern.slf4j.Slf4j;
import tech.cloudsales.users.model.User;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.function.Predicate;
import java.util.stream.Collectors;
@Slf4j
@Repository
@RequiredArgsConstructor
public class BigQueryUsersRepository implements UsersRepository {
@Setter
@Value("${cloudsales.bigquery.users.project}")
private String project;
@Setter
@Value("${cloudsales.bigquery.users.dataset}")
private String dataset;
@Setter
```

```
@Value("${cloudsales.bigquery.users.table}")
private String table;
private final BigQuery bigQuery;
@Override
public List<User> findAll() {
var query = String.format("WITH last_updated_rows AS (SELECT *, " +
"ROW_NUMBER() OVER (PARTITION BY email ORDER BY created_date DESC) AS
email_row_number, " +
"ROW_NUMBER() OVER (PARTITION BY id ORDER BY created_date DESC) AS
id_row_number, " +
"FROM `%s.%s.%s`) SELECT * FROM last_updated_rows WHERE
email_row_number = 1 and id_row_number = 1;",
project, dataset, table);
var queryConfig = QueryJobConfiguration.newBuilder(query).build();
try {
var jobId = JobId.newBuilder().build();
var queryJob =
bigQuery.create(JobInfo.newBuilder(queryConfig).setJobId(jobId).build(
));
queryJob = queryJob.waitFor();
var result = queryJob.getQueryResults();
return result.streamAll()
.map(row -> User.builder()
.id(row.get("id").getStringValue())
.email(row.get("email").getStringValue())
.firstName(row.get("first_name").getStringValue())
.lastName(row.get("last_name").getStringValue())
.active(row.get("active").getBooleanValue())
.permissions(Optional.ofNullable(row.get("permissions"))
.filter(Predicate.not(FieldValue::isNull))
.map(FieldValue::getStringValue)
.map(this::readPermissions)
.orElseGet(List::of))
.build())
.collect(Collectors.toList());
} catch (InterruptedException e) {
throw new RuntimeException(e);
}
```

```
}
@Override
public Optional<User> findByEmail(String email) {
var query = String.format("WITH last_updated_rows AS (SELECT *, " +
"ROW_NUMBER() OVER (PARTITION BY email ORDER BY created_date DESC) AS
email_row_number, " +
"ROW_NUMBER() OVER (PARTITION BY id ORDER BY created_date DESC) AS
id_row_number, " +
"FROM `%s.%s.%s`) SELECT * FROM last_updated_rows WHERE
email_row_number = 1 and id_row_number = 1 " +
"and email = '%s';", project, dataset, table, email);
var queryConfig = QueryJobConfiguration.newBuilder(query).build();
try {
var jobId = JobId.newBuilder().build();
var queryJob =
bigQuery.create(JobInfo.newBuilder(queryConfig).setJobId(jobId).build(
));
queryJob = queryJob.waitFor();
var result = queryJob.getQueryResults();
return result.streamAll()
.map(row -> User.builder()
.id(row.get("id").getStringValue())
.email(row.get("email").getStringValue())
.firstName(row.get("first_name").getStringValue())
.lastName(row.get("last_name").getStringValue())
.active(row.get("active").getBooleanValue())
.permissions(Optional.ofNullable(row.get("permissions"))
.filter(Predicate.not(FieldValue::isNull))
.map(FieldValue::getStringValue)
.map(this::readPermissions)
.orElseGet(List::of))
.build())
.findFirst();
} catch (InterruptedException e) {
throw new RuntimeException(e);
}
}
@Override
```

```
public User save(User user) {
var builder = InsertAllRequest.newBuilder(createTableId());
var formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
builder.addRow(Map.of("id", user.getId(),
"email", user.getEmail(),
"first_name", user.getFirstName(),
"last_name", user.getLastName(),
"active", String.valueOf(user.isActive()),
"permissions", writePermissions(user.getPermissions()),
"created_date", formatter.format(Instant.now())));
var response = bigQuery.insertAll(builder.build());
if (!response.getInsertErrors().isEmpty()) {
throw new RuntimeException(response.getInsertErrors().toString());
}
return user;
}
private TableId createTableId() {
return TableId.of(project, dataset, table);
}
private List<String> readPermissions(String permissions) {
var objectMapper = new ObjectMapper();
try {
return objectMapper.readValue(permissions, new TypeReference<>(){});
} catch (JsonProcessingException e) {
throw new RuntimeException(e);
}
}
private String writePermissions(List<String> permissions) {
var objectMapper = new ObjectMapper();
try {
return objectMapper.writeValueAsString(permissions);
} catch (JsonProcessingException e) {
throw new RuntimeException(e);
}
}
}
```

DefaultAuthorizationService.java

```
package tech.cloudsales.users.services;
import lombok.AllArgsConstructor;
import
org.springframework.security.oauth2.server.resource.authentication.Jwt
AuthenticationToken;
import org.springframework.stereotype.Service;
import tech.cloudsales.users.repositories.UsersRepository;
import java.util.Map;
import java.util.Optional;
@Service("authorizationService")
@AllArgsConstructor
public class DefaultAuthorizationService implements
AuthorizationService {
private final UsersRepository usersRepository;
@Override
public boolean hasRole(Authentication authentication, String role) {
return findUserEmail(authentication)
.flatMap(usersRepository::findByEmail)
.map(user -> user.getPermissions().contains(role))
.orElse(false);
}
private Optional<String> findUserEmail(Authentication authentication)
{
return findJwt(authentication)
.map(Jwt::getClaims)
.filter(this::containsVerifiedEmailClaim)
.flatMap(this::findEmail);
}
private Optional<Jwt> findJwt(Authentication authentication) {
return findJwtAuthenticationToken(authentication)
.map(JwtAuthenticationToken::getPrincipal)
.filter(Jwt.class::isInstance)
.map(Jwt.class::cast);
}
private Optional<JwtAuthenticationToken>
findJwtAuthenticationToken(Authentication authentication) {
return Optional.ofNullable(authentication)
```

```
.filter(JwtAuthenticationToken.class::isInstance)
.map(JwtAuthenticationToken.class::cast);
}
private boolean containsVerifiedEmailClaim(Map<String, Object> claims)
{
return Optional.ofNullable(claims.get("email_verified"))
.filter(Boolean.class::isInstance)
.map(Boolean.class::cast)
.orElse(false);
}
private Optional<String> findEmail(Map<String, Object> claims) {
return Optional.ofNullable(claims.get("email"))
.filter(String.class::isInstance)
.map(String.class::cast);
}
```

Order.java

```
package tech.cloudsales.orders.model;
import jakarta.persistence.*;
import lombok.*;
import java.util.List;
@Entity
@Table(name = "orders")
@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Order {
@Id
private String id;
private String status;
private String firstName;
private String middleName;
private String lastName;
private String email;
private String mobile;
private String region;
```

```
private String city;
private String address;
private String comments;
@OneToMany(mappedBy = "order", fetch = FetchType.EAGER, cascade =
CascadeType.ALL, orphanRemoval = true)
private List<OrderEntry> entries;
}
```

OrderEntry.java

```
package tech.cloudsales.orders.model;
import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
@Entity
@Table(name = "order_entries")
@Getter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class OrderEntry {
@Id
private String id;
private String productId;
private Long quantity;
private Long price;
@JsonIgnore
@ManyToOne
@JoinColumn(name = "order_id", referencedColumnName = "id")
private Order order;
}
```

ProductStock.java

```
package tech.cloudsales.orders.model;
import lombok.Builder;
import lombok.Getter;
```

```
@Builder
@Getter
public class ProductStock {
private String id;
private Long price;
private boolean active;
}
```

User.java

```
package tech.cloudsales.orders.model;
import lombok.NoArgsConstructor;
import java.util.List;
@Getter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User {
private String id;
private String email;
private String firstName;
private String lastName;
private List<String> permissions;
private boolean active;
}
```

UsersRepository.java

```
package tech.cloudsales.orders.repositories;
import tech.cloudsales.orders.model.User;
import java.util.Optional;
public interface UsersRepository {
Optional<User> findByEmail(String email);
}
```

BigQueryUsersRepository.java

```
package tech.cloudsales.orders.repositories;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import lombok.RequiredArgsConstructor;
```

```
import lombok.Setter;
import lombok.extern.slf4j.Slf4j;
import tech.cloudsales.orders.model.User;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.function.Predicate;
import java.util.stream.Collectors;
@Slf4j
@Repository
@RequiredArgsConstructor
public class BigQueryUsersRepository implements UsersRepository {
@Setter
@Value("${cloudsales.bigquery.users.project}")
private String project;
@Setter
@Value("${cloudsales.bigquery.users.dataset}")
private String dataset;
@Setter
@Value("${cloudsales.bigquery.users.table}")
private String table;
private final BigQuery bigQuery;
@Override
public Optional<User> findByEmail(String email) {
var query = String.format("WITH last_updated_rows AS (SELECT *, " +
"ROW_NUMBER() OVER (PARTITION BY email ORDER BY created_date DESC) AS
email_row_number, " +
"ROW_NUMBER() OVER (PARTITION BY id ORDER BY created_date DESC) AS
id_row_number, " +
"FROM `%s.%s.%s`) SELECT * FROM last_updated_rows WHERE
email_row_number = 1 and id_row_number = 1 " +
"and email = '%s';", project, dataset, table, email);
var queryConfig = QueryJobConfiguration.newBuilder(query).build();
try {
var jobId = JobId.newBuilder().build();
var queryJob =
bigQuery.create(JobInfo.newBuilder(queryConfig).setJobId(jobId).build(
));
```

```
queryJob = queryJob.waitFor();
var result = queryJob.getQueryResults();
return result.streamAll()
.map(row -> User.builder()
.id(row.get("id").getStringValue())
.email(row.get("email").getStringValue())
.firstName(row.get("first_name").getStringValue())
.lastName(row.get("last_name").getStringValue())
.active(row.get("active").getBooleanValue())
.permissions(Optional.ofNullable(row.get("permissions"))
.filter(Predicate.not(FieldValue::isNull))
.map(FieldValue::getStringValue)
.map(this::readPermissions)
.orElseGet(List::of))
.build())
.findFirst();
} catch (InterruptedException e) {
throw new RuntimeException(e);
}
}
private List<String> readPermissions(String permissions) {
var objectMapper = new ObjectMapper();
try {
return objectMapper.readValue(permissions, new TypeReference<>(){});
} catch (JsonProcessingException e) {
throw new RuntimeException(e);
}
}
private String writePermissions(List<String> permissions) {
var objectMapper = new ObjectMapper();
try {
return objectMapper.writeValueAsString(permissions);
} catch (JsonProcessingException e) {
throw new RuntimeException(e);
}
}
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Розробка хмарного сервісу продажу товарів та послуг через мережу Інтернет ”

Обсяг пояснювальної записки: 77 аркушів.

Дата завершення роботи: 17 червня 2024 р.

Підпис студента _____