МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МАРКЕТПЛЕЙС МОДУЛЯ ДЛЯ CMS MAGENTO 2»**

Виконав: студент 2 курсу, групи 8.1214-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
М.І. Кікош
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кікошу Михайлу Івановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка маркетплейс модуля для CMS Magento 2

керівник роботи Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 326-с

2. Строк подання студентом роботи 03.12.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд засобів для створення сервісів онлайн торгівлі.
2. Проєктний аналіз модуля маркетплейсу. Постановка задачі.
3. Програмна реалізація застосунку та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.03.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	25.03.2025	
2.	Збір вихідних даних.	21.04.2025	
3.	Обробка методичних та теоретичних	20.05.2025	
	джерел.		
4.	Розробка першого та другого розділу.	11.08.2025	
5.	Розробка третього розділу.	24.11.2025	
6.	Оформлення та нормоконтроль	01.12.2025	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	17.12.2025	

Студент _____ М.І. Кікош
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка маркетплейс модуля для CMS Magento 2»: 62 с., 22 рис., 7 табл., 50 джерел.

АРХІТЕКТУРА МОДУЛЯ, ЕЛЕКТРОННА КОМЕРЦІЯ, МАРКЕТПЛЕЙС, МОДУЛЬНА РОЗРОБКА, ОНЛАЙН МАГАЗИН, ПАНЕЛЬ АДМІНІСТРАТОРА, СИСТЕМА УПРАВЛІННЯ КОНТЕНТОМ, УПРАВЛІННЯ ПРОДАВЦЯМИ, УПРАВЛІННЯ ТОВАРАМИ, ФРЕЙМВОРК.

Об’єкт дослідження – модульна архітектура та розширюваність платформи Magento 2 для створення маркетплейс рішень.

Мета роботи: розробка масштабованого програмного рішення для організації багатокористувацької торгівлі на базі платформи Magento 2 з дотриманням стандартів Adobe Commerce.

Метод дослідження – аналіз архітектурних патернів Adobe Commerce, об’єктно-орієнтоване прєктування, модульне програмування, тестування програмного забезпечення, порівняльний аналіз існуючих маркетплейс-рішень.

У процесі розробки проведено аналіз існуючих рішень та модулів маркетплейсів, визначено їх недоліки та переваги. Побудована модульна структура рішення, реалізовані компоненти взаємодії з ядром Adobe Commerce, інтерфейси для адміністраторів та продавців, а також API для можливості подальшої інтеграції з зовнішніми сервісами.

В результаті роботи отримано програмне забезпечення для управління маркетплейсом на базі CMS Magento 2.

SUMMARY

Master’s qualifying paper “Development of a Marketplace Module for CMS Magento 2”: 62 pages, 22 figures, 7 tables, 50 references.

ADMINISTRATIVE PANEL, CONTENT MANAGEMENT SYSTEM, E-COMMERCE, FRAMEWORK, MARKETPLACE, MODULAR DEVELOPMENT, MODULE ARCHITECTURE, ONLINE STORE, PRODUCT MANAGEMENT, SELLER MANAGEMENT.

The object of the study is the modular architecture and extensibility of the Magento 2 platform for building marketplace solutions.

The aim of the study is to develop a scalable software solution for organizing multi-vendor commerce based on the Magento 2 platform in compliance with Adobe Commerce standards.

The method of research is analysis of Adobe Commerce architectural patterns, object-oriented design, modular programming, software testing, and comparative analysis of existing marketplace solutions.

During the development process, existing marketplace solutions and modules were analyzed, and their strengths and weaknesses were identified. A modular structure of the solution was designed, components for interaction with the Adobe Commerce core were implemented, along with interfaces for administrators and sellers, as well as APIs to enable further integration with external services.

As a result of the work, software for marketplace management based on the Magento 2 CMS was developed.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Огляд засобів для створення сервісів онлайн торгівлі 10
1.1 Аналіз існуючих платформ для онлайн-торгівлі що пропонують функціонал маркетплейса 10
1.2 Аналіз існуючих рішень для CMS Magento 2 що реалізують функціонал маркетплейса 11
1.3 Огляд особливостей CMS Magento 2 що мають бути враховані під час розробки функціоналу 14
1.3.1 Архітектурні принципи Magento 2 14
1.3.2 Система плагінів (Interceptors) 15
1.3.3 Система подій (Events) та Observer Pattern 15
1.3.4 Declarative Schema 16
1.3.5 API Layer: REST, GraphQL, SOAP 16
1.3.6 Caching та Indexing 16
1.3.7 ACL та Rights Management 17
1.3.8 UI Components Framework 17
1.4 Постановка задачі для створення власного рішення 17
2 Проєктний аналіз модуля маркетплейсу для CMS Magento 2 21
2.1 Аналіз предметної області та визначення вимог 21
2.2 UML-діаграма прецедентів 22
2.3 Структура бази даних 24
2.3.1 Таблиці модуля та їх взаємодія зі стандартними таблицями Magento 2 25
2.4 Проєктування ключових частин модуля 28

2.4.1 Реєстрація нового маркетплейса 28
2.4.2 Управління товарами 30
2.4.3 Формування та обробка замовлень 30
3 Реалізація функціоналу модуля маркетплейса для CMS Magento 2 32
3.1 Реалізація функціоналу зберігання та доступу до даних 32
3.2 Форма реєстрації нового маркетплейса 36
3.3 Сторінка перегляду маркетплейса 42
3.4 Сторінки категорій та продукта 45
3.5 Кошик 46
3.6 Обробка замовлень 50
3.7 Інтеграційні можливості 51
3.7.1 REST API 51
3.7.2 GraphQL 53
3.8 Тестування 55
Висновки 58
Перелік посилань 59

ВСТУП

Сучасний електронний бізнес характеризується стрімким зростанням онлайн-торгівлі та трансформацією традиційних бізнес-моделей у напрямку створення багатокористувацьких платформ. За даними дослідницької компанії Statista, обсяг глобального ринку електронної комерції у 2023 році перевищив 5,8 трильйонів доларів США, при цьому частка маркетплейсів становить понад 60% від загального обсягу онлайн-продажів [1]. В Україні спостерігається аналогічна тенденція – за даними EVO Company, частка маркетплейсів у структурі e-commerce зросла з 35% у 2020 році до 52% у 2023 році [2].

Платформа Magento 2 (Adobe Commerce) є однією з найпопулярніших систем управління контентом для електронної комерції, яка використовується понад 250 тисячами інтернет-магазинів по всьому світу [3]. Однак базовий функціонал Magento 2 не передбачає можливості створення повноцінного маркетплейсу з підтримкою множинних продавців, складним управлінням товарними позиціями та розподіленим замовленням між різними постачальниками.

Розробка спеціалізованого модуля для перетворення традиційного інтернет-магазину на базі Magento 2 у багатовендорну платформу є актуальним завданням, яке дозволяє власникам бізнесу масштабувати свою діяльність, залучати нових продавців та розширювати асортимент без необхідності зберігання власних товарних запасів.

Метою кваліфікаційної роботи є розробка функціонального модуля для системи управління контентом Magento 2, який забезпечує створення повноцінної маркетплейс-платформи з підтримкою множинних продавців, автоматичним розподілом замовлень та розширеним функціоналом управління товарними позиціями.

У роботі використано методи об’єктно-орієнтованого програмування [4], патерни проєктування (Repository, Factory, Plugin, Observer) [5], принципи

SOLID та Clean Architecture [6]. Для розробки модуля застосовано мову програмування PHP 8.2 [7], фреймворк Magento 2.4 [8], систему управління базами даних MySQL 8.0 [9], інструменти контролю версій Git та засоби автоматизованого тестування PHPUnit [10].

Розроблений модуль MK_Mplace може бути впроваджений у діючі інтернет-магазини на базі Magento 2 для розширення їх функціоналу до рівня повноцінних маркетплейс-платформ.

1 ОГЛЯД ЗАСОБІВ ДЛЯ СТВОРЕННЯ СЕРВІСІВ ОНЛАЙН ТОРГІВЛІ

1.1 Аналіз існуючих платформ для онлайн-торгівлі що пропонують функціонал маркетплейса

Сучасний ринок електронної комерції пропонує значну кількість платформ для створення маркетплейсів, кожна з яких має свої особливості, переваги та обмеження. Проведемо аналіз найпопулярніших рішень для визначення ключових функціональних можливостей та архітектурних підходів.

Shopify Plus є хмарною платформою, яка з 2020 року пропонує функціонал багатокористувацьких (multivendor) магазинів через додаток Shopify Collective [11]. Основні переваги платформи включають швидке розгортання, інтуїтивний інтерфейс та готову інфраструктуру для обробки платежів. Однак Shopify Plus має суттєві обмеження щодо кастомізації бізнес-логіки та високу вартість підписки (від $2000 на місяць), що робить її недоступною для малого та середнього бізнесу [12].

WooCommerce на базі WordPress підтримує маркетплейс-функціонал через плагін Dokan Multivendor Marketplace [13]. Платформа має відкритий вихідний код та велику спільноту розробників. Проте WooCommerce демонструє проблеми з продуктивністю при каталогах понад 10000 товарів та обмежену масштабованість через архітектурні особливості WordPress [14].

CS-Cart Multi-Vendor є спеціалізованим рішенням для створення маркетплейсів з коробковим функціоналом управління продавцями, комісіями та виплатами [15]. Платформа підтримує до 500000 SKU та має вбудовану систему аналітики. Недоліком є закритий вихідний код, що ускладнює глибоку інтеграцію з корпоративними системами, та ліцензійна модель з одноразовим платежем від $1450 [16].

Magento 2 (Adobe Commerce) є enterprise-рівня платформою з відкритим вихідним кодом, яка забезпечує максимальну гнучкість для створення

складних e-commerce рішень [17]. Базова версія не включає маркетплейс-функціонал, однак модульна архітектура дозволяє розширювати систему через custom модулі. Платформа підтримує понад 300000 SKU, має вбудовану підтримку B2B-функціоналу та забезпечує високу продуктивність при правильній конфігурації [18].

Порівняльний аналіз ключових характеристик представлено в таблиці 1.1.

Таблиця 1.1 – Порівняння платформ для створення маркетплейсів

Характеристика	**Shopify Plus**	**WooCommerce**	**CS-Cart**	**Magento 2**
Тип ліцензії	Проприєтарна	Open Source	Проприєтарна	Open Source
Вартість входу	$2000/міс	Безкоштовно	$1450	Безкоштовно
Масштабованість	Висока	Середня	Висока	Дуже висока
Кастомізація	Обмежена	Середня	Середня	Повна
Підтримка API	REST, GraphQL	REST	REST	REST, GraphQL, SOAP

За результатами аналізу можна зробити висновок, що Magento 2 є оптимальною платформою для створення маркетплейс-рішень корпоративного рівня завдяки відкритому коду, потужній архітектурі та повній свободі кастомізації [19].

1.2 Аналіз існуючих рішень для CMS Magento 2 що реалізують функціонал маркетплейса

На ринку представлено декілька готових модулів для перетворення Magento 2 у маркетплейс-платформу. Проведемо детальний аналіз найпопулярніших рішень.

Webkul Marketplace є найбільш поширеним рішенням з понад 5000 інсталяцій станом на 2024 рік [20]. Модуль надає базовий функціонал управління продавцями, включаючи реєстрацію вендорів, створення товарів,

управління замовленнями та систему комісій. Архітектура модуля базується на створенні окремих таблиць бази даних для зберігання інформації про продавців та їх прив'язки до товарів.

Основні функціональні можливості Webkul Marketplace:

- реєстрація та управління профілями продавців через frontend;
- створення та редагування товарів продавцями;
- автоматичний розподіл замовлень між продавцями;
- система нарахування та виплати комісій адміністратору;
- окремі dashboard для кожного продавця з аналітикою продажів;
- REST API для інтеграції з зовнішніми системами.

Недоліки рішення включають:

- високу вартість ліцензії ($599 для Community Edition, $1499 для Enterprise) [21];
- обмежену підтримку B2B-сценаріїв;
- відсутність вбудованої підтримки множинних незалежних маркетплейсів;
- проблеми з продуктивністю при понад 1000 активних продавцях;
- конфлікти з деякими сторонніми модулями через переписування core функціоналу.

CedCommerce Marketplace є альтернативним рішенням з фокусом на інтеграції з зовнішніми маркетплейсами (Amazon, eBay) [22]. Модуль підтримує синхронізацію інвентарю, автоматичне оновлення цін та управління замовленнями з різних каналів продажу.

Ключові особливості:

- автоматична синхронізація з 10+ зовнішніми маркетплейсами;
- Bulk операції для управління великою кількістю товарів;
- підтримка множинних валют та мов;
- система рейтингів та відгуків для продавців [23].

Обмеження:

- фокус на інтеграціях замість внутрішнього маркетплейс-функціоналу;

– вартість від $799 за базову версію;
– складність налаштування для нестандартних бізнес-процесів.

Amasty Multi Vendor Marketplace пропонує баланс між функціональністю та продуктивністю [24]. Модуль використовує принципи dependency injection та service contracts Magento 2 для забезпечення сумісності з іншими розширеннями.

Переваги:

– чистий код згідно Magento coding standards;
– відсутність модифікацій core файлів;
– підтримка Magento 2.4.x;
– інтеграція з Amasty екосистемою модулів [25].

Недоліки:

– обмежений функціонал порівняно з Webkul (відсутність деяких B2B features);
– вартість $349 (найдешевше рішення, але з меншими можливостями);
– залежність від інших Amasty модулів для повного функціоналу.

Порівняльна таблиця функціональності представлена в таблиці 1.2.

Таблиця 1.2 – Порівняння маркетплейс-модулів для Magento 2

Функціонал	**Webkul**	**CedCommerce**	**Amasty**
Управління продавцями	Так	Так	Так
Розподіл замовлень	Так	Так	Так
Система комісій	Так	Так	Так
Множинні маркетплейси	Ні	Ні	Ні
GraphQL API	Частково	Ні	Ні
B2B функціонал	Обмежений	Ні	Обмежений
Вартість	$599-$1499	$799	$349

Критичним обмеженням всіх проаналізованих рішень є відсутність підтримки множинних незалежних маркетплейсів у рамках однієї Magento інсталяції. Існуючі модулі реалізують модель «один магазин = один

маркетплейс», що не дозволяє створювати окремі торгові майданчики з різними правилами, комісіями та асортиментом [26].

1.3 Огляд особливостей CMS Magento 2 що мають бути враховані під час розробки функціоналу

Magento 2 є складною enterprise-платформою з власною архітектурою та принципами розробки, які необхідно враховувати при створенні розширень [27].

1.3.1 Архітектурні принципи Magento 2

Magento 2 побудована на базі модульної архітектури, де кожен модуль є незалежним компонентом з власною бізнес-логікою [28]. Система використовує наступні архітектурні шари:

- Presentation Layer – відповідає за відображення даних користувачу через themes, layouts та templates;
- Service Layer – надає API для взаємодії з бізнес-логікою через service contracts;
- Domain Layer – містить бізнес-логіку у вигляді models, resource models та collections;
- Persistence Layer – забезпечує збереження даних через repositories та data mappers [29].

Dependency Injection (DI) є основним механізмом управління залежностями в Magento 2 [30]. Всі залежності класів описуються в di.xml файлах та автоматично вставляються через конструктори. Це забезпечує слабку з’язаність компонентів та спрощує тестування.

Service Contracts визначають публічний API модуля через PHP interfaces [31].

Magento 2 розрізняє три типи service contracts:

– Data interfaces – описують структуру даних;
– Repository interfaces – визначають методи для роботи з persistence;
– Management interfaces – містять бізнес-логіку.

Використання service contracts забезпечує зворотню сумісність та дозволяє замінювати реалізації без зміни коду, що їх використовує [32].

1.3.2 Система плагінів (Interceptors)

Плагіни (interceptors) є механізмом для модифікації поведінки публічних методів класів без їх переписування [33]. Magento 2 підтримує три типи плагінів:

– before – виконується до виклику оригінального методу, може змінювати аргументи;
– after – виконується після виклику, може модифікувати результат;
– around – повністю контролює виконання оригінального методу.

Плагіни мають пріоритет виконання (sortOrder) та можуть бути відключені через конфігурацію, що робить їх більш гнучкими порівняно з класичним переписуванням [34].

1.3.3 Система подій (Events) та Observer Pattern

Magento 2 активно використовує Observer Pattern для розширюваності [35]. Система dispatch понад 300 стандартних подій у різних точках виконання коду. Розробники можуть створювати власні події через EventManager.

Events є рекомендованим рішенням для реагування на зміни стану системи, оскільки вони не блокують основний код виконання та можуть бути відключені адміністратором [36].

1.3.4 Declarative Schema

Починаючи з Magento 2.3, система використовує Declarative Schema для управління структурою бази даних [37]. Розробники описують бажаний стан схеми в db_schema.xml файлах, а Magento автоматично генерує міграції.

Declarative Schema забезпечує версійність структури бази даних та спрощує rollback змін [38].

1.3.5 API Layer: REST, GraphQL, SOAP

Magento 2 надає три типи API для інтеграції [39]:

- REST API використовує HTTP методи та JSON для обміну даними: кожен service contract автоматично експонується як REST endpoint через webapi.xml конфігурацію;
- GraphQL API забезпечує гнучкі запити з можливістю отримання тільки необхідних полів [40]: Schema визначається через schema.graphqls файли;
- SOAP API підтримується для сумісності з legacy системами, однак не рекомендується для нових інтеграцій через overhead XML processing [41].

1.3.6 Caching та Indexing

Magento 2 використовує багаторівневу систему кешування [42]:

- Configuration cache – зберігає merged конфігурацію з усіх модулів;
- Layout cache – кешує скомпільовані XML layout файли;
- Block HTML cache – кешує згенерований HTML блоків;
- Full Page Cache (FPC) – кешує цілі сторінки через Varnish або Redis.

Indexers оптимізують запити до бази даних, створюючи допоміжні таблиці з денормалізованими даними [43]. При розробці маркетплейс-функціоналу необхідно створити власні індекси для прив’язки товарів до маркетплейсів.

1.3.7 ACL та Rights Management

Access Control List (ACL) визначає права доступу користувачів до функціоналу системи [44]. Розробники описують ресурси в acl.xml.

ACL інтегрується з системою авторизації API та UI компонентів [45].

1.3.8 UI Components Framework

Magento 2 використовує UI Components для створення складних інтерфейсів адміністративної панелі [46]. Компоненти описуються у XML та JavaScript файлах, що дозволяє створювати таблиці (listing), форми та інші елементи без написання HTML.

UI Components забезпечують уніфікований підхід до створення інтерфейсів та підтримують AJAX операції out-of-the-box [47].

1.4 Постановка задачі для створення власного рішення

На основі проведеного аналізу існуючих платформ, модулів для Magento 2 та архітектурних особливостей CMS можна сформулювати наступні *недоліки* наявних рішень.

Відсутність підтримки множинних незалежних маркетплейсів. Існуючі модулі реалізують концепцію «один магазин = один маркетплейс», що

не дозволяє створювати різні торгові майданчики з унікальними правилами комісій, асортиментом та брендингом у рамках однієї Magento інсталяції.

Обмежена гнучкість прив'язки товарів. Більшість рішень використовують жорстку прив'язку товару до одного продавця через foreign key в базі даних, що унеможливлює динамічне призначення товарів до різних маркетплейсів на основі атрибутів або business rules.

Відсутність часткового checkout функціоналу. Існуючі модулі не підтримують оформлення замовлення тільки для вибраного маркетплейсу зі збереженням інших товарів у кошику для наступних покупок.

Неповна інтеграція з Magento 2 API. Більшість комерційних рішень не надають повноцінної підтримки GraphQL API, що є критичним для headless commerce та PWA застосунків [48].

Порушення Magento 2 best practices. Аналіз коду комерційних модулів виявив використання deprecated функцій, direct SQL queries замість repositories та недотримання SOLID принципів, що призводить до проблем з масштабованістю та підтримкою [49].

Висока вартість ліцензій. Ціни на готові рішення ($349-$1499) є бар'єром для малого та середнього бізнесу.

На основі виявлених недоліків формулюється мета розробки модуля MK_Mplace – створити функціональний, масштабований та гнучкий модуль для Magento 2, який:

- підтримує множинні незалежні маркетплейси з можливістю створення, редагування та видалення торгових майданчиків через адміністративну панель;
- реалізує гнучку систему прив'язки товарів до маркетплейсів на основі атрибутів продукції з підтримкою автоматичного призначення через business rules;
- забезпечує partial checkout функціонал з можливістю оформлення замовлення для вибраного маркетплейсу зі збереженням інших товарів у кошику;

– надає повну підтримку REST та GraphQL API для забезпечення інтеграції з headless архітектурами та мобільними застосунками;
– дотримується Magento 2 coding standards та architectural best practices для забезпечення сумісності з майбутніми версіями платформи;
– має відкритий вихідний код та детальну документацію для забезпечення доступності рішення широкому колу розробників та бізнесу.

Функціональні вимоги до модуля MK_Mplace:

– управління маркетплейсами (CRUD операції) через admin panel;
– автоматичне призначення товарів до маркетплейсів на основі атрибута marketplace_id;
– фільтрація товарів на frontend за маркетплейсами через layered navigation;
– групування товарів у кошику за маркетплейсами;
– можливість оформлення замовлення для одного маркетплейсу зі збереженням інших товарів;
– REST API endpoints для управління маркетплейсами;
– GraphQL schema та resolvers для headless інтеграцій;
– ACL для контролю доступу до функціоналу маркетплейсів;
– індексація товарів за маркетплейсами для оптимізації запитів;
– підтримка мультимовності та multi-store режиму Magento 2.

Нефункціональні вимоги:

– сумісність з Magento 2.4.x (PHP 8.1-8.2);
– відсутність модифікацій core файлів Magento;
– покриття коду unit та integration тестами мінімум 70%;
– час відповіді API endpoints не більше 200ms для базових операцій;
– підтримка до 100000 товарів та 1000 маркетплейсів без деградації продуктивності;
– документація всіх публічних API та service contracts;
– відповідність стандартам PSR-1, PSR-2, PSR-4 [50].

Технічний стек розробки:

- PHP 8.2 з використанням typed properties та attributes;
- Magento 2.4.7 framework;
- MySQL 8.0 для персистентності даних;
- Redis для кешування та сесій;
- Elasticsearch 8.x для повнотекстового пошуку;
- PHPUnit 10.x для тестування;
- Composer 2.x для управління залежностями;
- Git для версійного контролю.

2 ПРОЄКТНИЙ АНАЛІЗ МОДУЛЯ МАРКЕТПЛЕЙСУ ДЛЯ CMS MAGENTO 2

2.1 Аналіз предметної області та визначення вимог

Розробка модуля маркетплейсу для Adobe Commerce передбачає створення додаткового функціоналу поверх стандартного ядра платформи, який дозволяє перетворити інтернет-магазин на торгівельний майданчик з багатьма продавцями. Такий модуль забезпечує можливість реєстрації окремих продавців, керування їхніми товарами та замовленнями, а також організацію взаємодії між усіма учасниками – адміністраторами, продавцями та покупцями. У межах модуля також можуть реалізовуватися механізми розрахунку комісій, модерації контенту, аналітики та управління фінансовими операціями.

Основними користувачами системи є: адміністратор маркетплейсу, продавець та покупець.

«Адміністратор маркетплейсу» – відповідає за загальне функціонування платформи.Його основні задачі:

- керування глобальними налаштуваннями маркетплейсу (комісії, правила модерації, способи виплат продавцям);
- модерація реєстрацій продавців (перевірка даних, активація/деактивація облікових записів, блокування при порушеннях);
- контроль товарів продавців (перевірка описів, цін, зображень, відповідності політикам майданчика);
- моніторинг замовлень і фінансових транзакцій, формування звітів та аналітики;
- вирішення спірних ситуацій між продавцями та покупцями, обробка скарг.

«Продавець» – користувач, який має власний кабінет на маркетплейсі

та відповідає за свій асортимент і обслуговування клієнтів.

Функціонал для продавця зазвичай включає:

- реєстрацію та подання заявки на отримання статусу продавця;
- створення, редагування та видалення власних товарів (опис, ціни, залишки, зображення, характеристики);
- керування замовленнями, що містять його товари (підтвердження, зміна статусів, обробка повернень);
- перегляд статистики продажів, обороту і виплат;
- управління профілем магазину (логотип, опис бренду, контактні дані, умови доставки тощо);
- взаємодію з покупцями (відповіді на запитання, робота з відгуками).

«Покупець» – кінцевий користувач платформи, який здійснює покупки на маркетплейсі. Для покупця модуль маркетплейсу розширює стандартні можливості такими опціями:

- перегляд товарів різних продавців у єдиному каталозі;
- можливість фільтрації та вибору товарів за продавцем (сторінки магазинів продавців);
- перегляд історії своїх замовлень, відстеження статусів і, за потреби, ініціювання повернень.

У результаті модуль маркетплейсу формує повноцінну екосистему, у якій кожна роль має чітко визначені права та обов’язки, а адміністратор отримує інструменти для централізованого контролю й розвитку платформи. Продавці користуються готовою інфраструктурою для онлайн-продажів, а покупці – широким асортиментом товарів і можливістю порівнювати пропозиції різних постачальників в одному інтерфейсі.

2.2 UML-діаграма прецедентів

На рисунку 2.1 зображена UML-діаграма прецедентів що відображає взаємодію користувачів з системою.



Рисунок 2.1 – UML-діаграма прецедентів

2.3 Структура бази даних

Платформа Magento 2 використовує реляційну СУБД MySQL (або її сумісні реалізації, наприклад MariaDB) як основне сховище даних. Доступ до бази даних організований через власний ORM-шар, що включає моделі (Model), ресурсні моделі (ResourceModel) та колекції (Collection). Такий підхід приховує SQL-деталі від бізнес-логіки та забезпечує уніфікований доступ до даних.

Розглянемо ключові особливості роботи Magento 2 з БД.

Декларативна схема (Declarative Schema) – структура таблиць описується в конфігураційному файлі etc/db_schema.xml. На основі цього опису Magento автоматично генерує та застосовує SQL-міграції при встановленні та оновленні модулів. Це спрощує керування версіями схеми та зменшує ризик помилок, пов’язаних із ручними міграціями.

Розділення відповідальностей – бізнес-моделі не працюють безпосередньо з SQL. Усі CRUD-операції делегуються ресурсним моделям, які наслідують базові класи фреймворку (\Magento\Framework\Model\ResourceModel\Db\AbstractDb) і знають, з якою таблицею вони пов’язані.

Патерни Repository і Service Contracts – доступ до сутностей з боку бізнес-логіки та API відбувається через репозиторії (клас Model/*Repository.php), що реалізують інтерфейси з директорії Api/. Репозиторії використовують Search Criteria API для побудови гнучких запитів, не прив’язаних до конкретної СУБД.

Використання зовнішніх ключів та індексів – Magento 2 заохочує використання обмежень цілісності на рівні БД (FK, UNIQUE, INDEX), що автоматично описуються в db_schema.xml. Це дозволяє забезпечити узгодженість даних між модулями та оптимізувати продуктивність запитів.

Розроблюваний модуль має повністю відповідати цим підходам: структура таблиць описана декларативно, доступ до даних здійснюється через

ресурсні моделі та репозиторії, а цілісність забезпечується зовнішніми ключами та індексацією.

2.3.1 Таблиці модуля та їх взаємодія зі стандартними таблицями Magento 2

У рамках модуля спроєктовано мінімалістичну, але розширювану модель даних, яка додає до стандартної схеми Magento 2 підтримку маркетплейсів та їх зв’язків з товарами каталогу.

Основними є дві таблиці:

– mk_mplace – зберігає дані про маркетплейси, реалізує зв’язок «один-до-багатьох» з обліковим записом адміністратора (див. табл. 2.1);
– mk_mplace_product_link – реалізує зв’язок «багато-до-багатьох» між маркетплейсами та товарами (стандартна таблиця catalog_product_entity).

Таблиця 2.1 – Опис таблиці mk_mplace

Поле	**Тип**	**Обмеження**	**Опис**
entity_id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор маркетплейсу
name	VARCHAR(255)	NOT NULL, INDEX	Назва маркетплейсу
logo	VARCHAR(255)	NULLABLE	Логотип (шлях)
registration_date	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Дата реєстрації маркетплейсу
about	TEXT	NULLABLE	Опис / інформація про маркетплейс
schedule	TEXT	NULLABLE	Графік роботи
shipping_info	TEXT	NULLABLE	Інформація про доставку
payment_info	TEXT	NULLABLE	Інформація про оплату
return_policy	TEXT	NULLABLE	Політика повернення
warranty_policy	TEXT	NULLABLE	Гарантійна політика
active	SMALLINT	NOT NULL, DEFAULT 1, INDEX	Статус активності (0/1)

Продовження таблиці 2.1

Поле	**Тип**	**Обмеження**	**Опис**
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	NOT NULL, ON UPDATE CURRENT_TIMESTAMP	Дата останнього оновлення
handle	VARCHAR(255)	NOT NULL, UNIQUE	Код маркетплейсу для URL
mplace_user_id	INT	NULLABLE, FK	Посилання на адміністративного користувача

Ключові обмеження та індекси:

- PK по entity_id;
- UQ по handle – гарантує унікальність хендла;
- індекси по name та active для швидкого пошуку й фільтрації;
- зовнішній ключ по mplace_user_id до таблиці адмін-користувачів (наприклад, admin_user.user_id) – забезпечує зв’язок маркетплейсу з конкретним адміністратором/власником в адмін-панелі.

Таким чином, кожен запис у mk_mplace може (але не зобов’язаний) бути прив’язаний до конкретного адміністративного користувача, що спрощує розмежування відповідальності та реалізацію прав доступу.

Таблиця mk_mplace_product_link реалізує зв’язок «багато-до-багатьох» між маркетплейсами та товарами (див. табл. 2.2).

Таблиця 2.2 – опис таблиці mk_mplace_product

Поле	**Тип**	**Обмеження**	**Опис**
link_id	INT	PK, AUTO_INCREMENT	Ідентифікатор запису зв’язку
mplace_id	INT	FK, NOT NULL	Посилання на mk_mplace.entity_id
product_id	INT	FK, NOT NULL	Посилання на catalog_product_entity.entity_id
position	INT	NOT NULL, DEFAULT 0, INDEX	Позиція/порядок відображення у маркетплейсі
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Дата створення

Обмеження та індекси:

– PK по link_id;
– FK mplace_id → mk_mplace.entity_id з ON DELETE CASCADE – при видаленні маркетплейсу видаляються всі зв’язки;
– FK product_id → catalog_product_entity.entity_id з ON DELETE CASCADE – при видаленні товару видаляються всі його зв’язки з маркетплейсами;
– унікальний складений індекс (mplace_id, product_id) – одна пара маркетплейс-товар не може дублюватися;
– окремі індекси по mplace_id, product_id та position для оптимізації пошуку та сортування.

На рисунку 2.2 наведена спрощена схема БД з урахуванням актуальних структур таблиць та FK на адмін-користувачів.

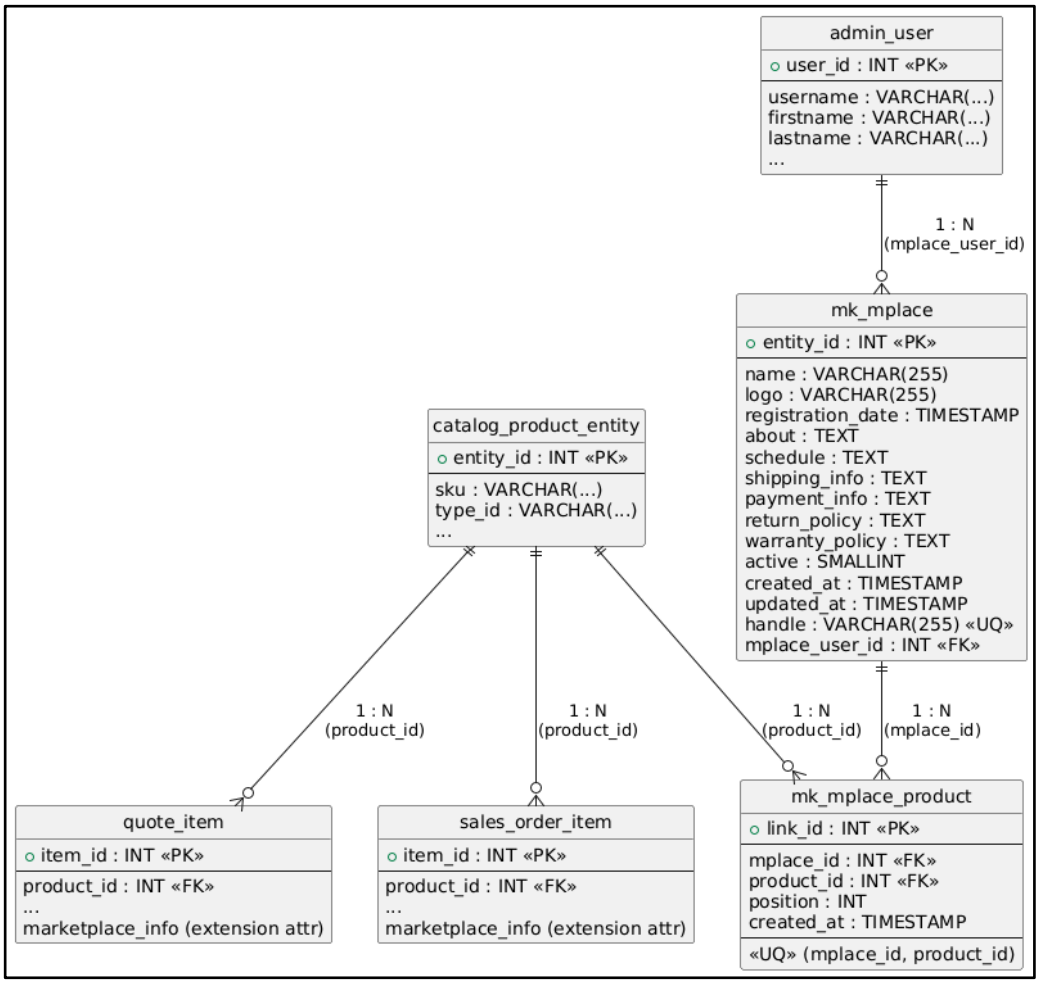


Рисунок 2.2 – Схема бази даних модуля

2.4 Проєктування ключових частин модуля

У процесі розробки модуля для реалізації функціоналу маркетплейса для CMS Magento 2 важливо спроєктувати його складові, що виконують основні функції.

У цьому розділі буде описано проєктування трьох ключових частин функціоналу модуля: логіку реєстрації та створення нового маркетплейса, управління товарами, формування та обробки замовлень.

2.4.1 Реєстрація нового маркетплейса

Оскільки у CMS Magento 2 вже є реалізована повноцінна система управління контентом за допомогою адміністративної панелі, а також досить гнучка система ролей облікових записів і доступу до ресурсів адміністративної панелі (Access Control List або ACL), логічним кроком є використання цих можливостей для реалізації доступу власників маркетплейсу до операцій з управління товарами та замовленнями. Тому у розроблюваному модулі процес створення нового маркетплейса складатиметься з двох логічно пов’язаних етапів:

а) створення сутності Маркетплейс у таблиці mk_mplace;

б) створення і прив’язка маркетплейса до адміністративного користувача Magento 2 з певною роллю.

Такий підхід дозволяє відмовитись від повноцінної реалізації інтерфейсу управління товарами і замовленнями з нуля і зосередитись на реалізації функціоналу маркетплейса.

Алгоритм взаємодії власника маркетплейса з системою розглянемо на діаграмі послідовності (див. рис. 2.3).

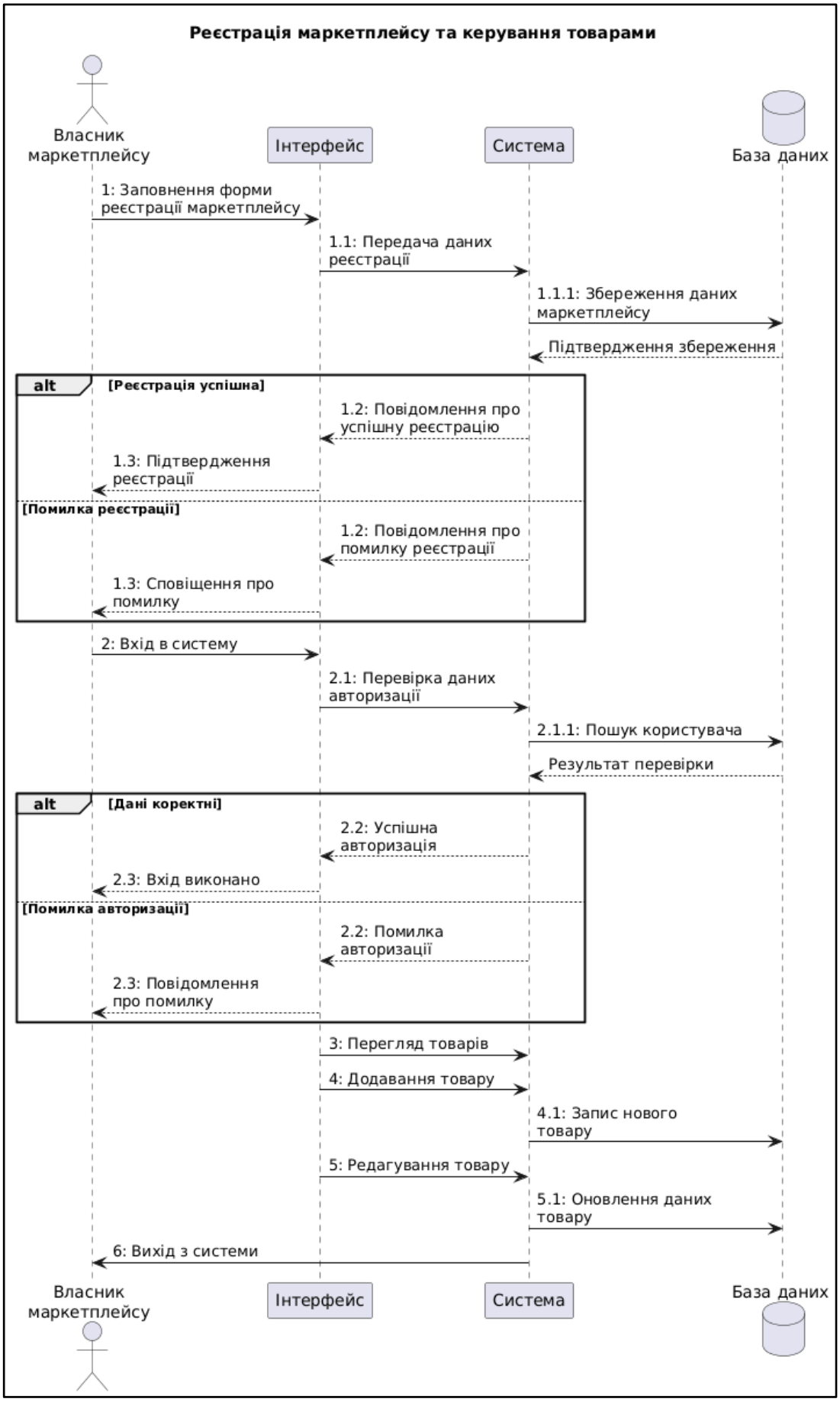


Рисунок 2.3 – Діаграма послідовності взаємодії власника маркетплейса з системою

2.4.2 Управління товарами

У попередньому розділі ми з’ясували, що вже наявна реалізація панелі управління товарами функціонально повністю задовольняє потреби управління контентом. Виходячи з цього залишається реалізувати розмежування (прив’язку) товарів між користувачами з урахуванням архітектури базової системи у всіх потенційних частинах системи, де власник маркетплейсу може взаємодіяти з товарами.

2.4.3 Формування та обробка замовлень

CMS Magento 2 є потужною і гнучкою e-commerce системою з багатим функціоналом що дозволяє глибоко кастомізувати або створювати власну послідовність обробки замовлень включаючи набір статусів.

Проте система не розрахована на роздільну оплату для випадків коли у кошику є декілька товарів від різних продавців. Великі торговельні майданчики вирішують цю проблему виступаючи посередником, вони приймають оплату від покупця і в подальшому за виконання умов замовлення всіма продавцями, що беруть участь у угоді, розподіляють отримані кошти. Реалізація та інтеграція подібного функціоналу для CMS Magento 2 потребує значних матеріальних і часових затрат та може бути об’єктом окремого наукового дослідження.

Натомість для реалізації у модулі був обраний більш простий та логічний варіант, що виключає роль посередника а оплата здійснюється безпосередньо на рахунок продавця. Він полягає у групуванні у корзині товарів по маркетплейсу з можливістю оформити замовлення в одному маркетплейсі за один раз. Це частково зменшує комфорт користування майданчиком проте спрощує регулювання відносин між учасниками угоди у разі можливих повернень або інших непорозумінь. Оскільки система не

призначена для функціонування подібним чином – під час реалізації виникне ряд задач, що потребуватиме зусиль щодо модифікації принципу роботи кошика покупок (shopping cart) із врахуванням архітектури вже наявного функціоналу. Цей процес буде описаний більш детально у відповідному підрозділі наступного розділу.

3 РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ МОДУЛЯ МАРКЕТПЛЕЙСА ДЛЯ CMS MAGENTO 2

3.1 Реалізація функціоналу зберігання та доступу до даних

Однією з особливостей CMS Magento 2 є декларативна схема (Magento 2 Declarative Schema). Декларативна схема – це підхід, за якого структура бази даних описується декларативно в XML-файлах, а не через PHP-скрипти InstallSchema / UpgradeSchema. Magento (починаючи з 2.3) сама обчислює різницю між поточним станом БД і описом у файлах та автоматично виконує потрібні зміни.

Таблиці, описані у пункті 2.3.1 були додані з використанням декларативної схеми за допомогою файла app/code/MK/Mplace/etc/db_schema.xml. Фрагмент, що відповідає за створення таблиці mk_mplace зображено на рисунку 3.1.

```
<?xml version="1.0"?>
<schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">
   <table name="mk_mplace" resource="default" engine="innodb" comment="Marketplace Table">
      <column xsi:type="int" name="entity_id" unsigned="true" nullable="false" identity="true" comment="Entity ID"/>
   …
      <column xsi:type="timestamp" name="registration_date" on_update="false" nullable="false" default="CURRENT_TIMESTAMP" comment="Registration Date"/>
   …
   </table>
```

Рисунок 3.1 – Використання декларативної схеми

Декларативна схема також дозволяє описати обмеження (constraints) та додати індекси, якщо необхідно. Приклад розглянемо на рисунку 3.2.

```
<table name="mk_mplace_product" resource="default" engine="innodb"
comment="Marketplace Product Link Table">
      …
      <constraint xsi:type="primary" referenceId="PRIMARY">
        <column name="link_id"/>
      </constraint>

      <constraint xsi:type="unique"
referenceId="MK_MPLACE_PRODUCT_MPLACE_ID_PRODUCT_ID">
        <column name="mplace_id"/>
        <column name="product_id"/>
      </constraint>

      <constraint xsi:type="foreign"
referenceId="MK_MPLACE_PRODUCT_MPLACE_ID_MK_MPLACE_ENTITY_ID"
              table="mk_mplace_product"
              column="mplace_id"
              referenceTable="mk_mplace"
              referenceColumn="entity_id"
              onDelete="CASCADE"/>

      <index referenceId="MK_MPLACE_PRODUCT_MPLACE_ID_IDX"
indexType="btree">
        <column name="mplace_id"/>
      </index>
   …
        </table>
```

Рисунок 3.2 – Опис обмежень та індексів через декларативну схему

Обмеження описуються полями constraint. У наведеному прикладі є три типи обмежень, вони позначені властивістю xsi:type:

– xsi:type="primary" – створює primary key, гарантує унікальність головного ідентифікатора рядка, забезпечує індексацію по цьому полю;
– xsi:type="unique" – створює UNIQUE constraint (унікальний індекс), забороняє дублювання значень у вказаних колонках (наприклад унікальний handle);
– xsi:type="foreign" – створює ключ foreign key між таблицями, гарантує, що значення колонки існує в пов’язаній таблиці (посилальна цілісність), параметр onDelete="CASCADE" задає

поведінку при видаленні батьківського запису (автоматично видаляє дочірні у пов’язаних таблицях).

Індекси описуються полями index, відповідно. Індекс створює додаткову індексацію по вказаних колонках, це прискорює SELECT з WHERE, ORDER BY, JOIN по цих полях. На рисунку 3.3 зображено створення індексів за допомогою декларативної схеми.

```
    <table name="mk_mplace_product" resource="default" engine="innodb"
comment="Marketplace Product Link Table">
    …
    <index referenceId="MK_MPLACE_PRODUCT_MPLACE_ID_IDX"
indexType="btree">
      <column name="mplace_id"/>
    </index>

    <index referenceId="MK_MPLACE_PRODUCT_POSITION_IDX" indexType="btree">
      <column name="position"/>
    </index>
  </table>
```

Рисунок 3.3 – Створення індексів

Приклад: індекс MK_MPLACE_PRODUCT_MPLACE_ID_IDX робить пошук усіх товарів конкретного маркетплейса по полю mplace_id значно швидшим.

У модулі MK_Mplace за роботу з сутністю mplace відповідають наступні файли:

а) API-інтерфейси:

- app/code/MK/Mplace/Api/MplaceRepositoryInterface.php – контракт репозиторію;
- app/code/MK/Mplace/Api/Data/MplaceInterface.php – контракт моделі;
- app/code/MK/Mplace/Api/Data/MplaceSearchResultsInterface.php – контракт результатів пошуку;

б) моделі:

- app/code/MK/Mplace/Model/Mplace.php – модель маркетплейса (реалізація MplaceInterface);
- app/code/MK/Mplace/Model/MplaceRepository.php – репозиторій: save, get, getList, delete;
- app/code/MK/Mplace/Model/MplaceSearchResults.php – реалізація MplaceSearchResultsInterface;

в) ресурсні моделі, колекції:

- app/code/MK/Mplace/Model/ResourceModel/Mplace.php – робота з таблицею mk_mplace;
- app/code/MK/Mplace/Model/ResourceModel/Mplace/Collection.php – колекція маркетплейсів.

Сутність mplace_product (ProductLink у коді) відповідно описана у:

а) API-інтерфейси:

- app/code/MK/Mplace/Api/ProductLinkRepositoryInterface.php – контракт репозиторію зв’язків;
- app/code/MK/Mplace/Api/Data/LinkInterface.php – контракт одного зв’язку mplace-product;
- app/code/MK/Mplace/Api/Data/MplaceInfoInterface.php – DTO з інформацією про маркетплейс для товару;

б) моделі:

- app/code/MK/Mplace/Model/ProductLink.php – модель для рядка з таблиці mk_mplace_product;
- app/code/MK/Mplace/Model/ProductLinkRepository.php – репозиторій для операцій зі зв’язками;
- app/code/MK/Mplace/Model/ProductLinkSearchResults.php – результати пошуку по зв’язках;

в) ресурсні моделі і колекції:

- app/code/MK/Mplace/Model/ResourceModel/ProductLink.php – робота з таблицею mk_mplace_product;
- app/code/MK/Mplace/Model/ResourceModel/ProductLink/Collection.php – колекція зв’язків.

3.2 Форма реєстрації нового маркетплейса

Після реалізації зберігання і взаємодії сутностей прийшов час для реалізації функціоналу взаємодії користувача із сутностями. Реалізацію було розпочато з панелі управління адміністратора.

Для початку був створений маршрут для адмін панелі за яким повинен бути доступним функціонал модуля. Маршрут описується у файлі MK/Mplace/etc/adminhtml/routes.xml (див. рис. 3.4) і має наступні поля:

- роутер з ідентифікатором, в даному випадку ідентифікатор вказує що роутер об’єднує маршрути що відносяться саме до панелі управління, фронтенд ігноруватиме такий роутер;
- маршрут з ідентифікатором має параметр frontName="mplace" що означає частину URL по якій система визначить як обробляти запит;
- ім’я модуля – вказує, що всі запити з frontName="mplace" треба обробляти контролерами модуля MK_Mplace.

```
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:App/etc/routes.xsd">
   <router id="admin">
      <route id="mplace" frontName="mplace">
         <module name="MK_Mplace" />
      </route>
   </router>
</config>
```

Рисунок 3.4 – Маршрут модуля для панелі управління

Наступним кроком були створені макети (layout) відповідальні за:

- MK/Mplace/view/adminhtml/layout/mplace_mplace_index.xml – відображення списку сутностей за допомогою uiComponent для здійснення CRUD операцій;
- MK/Mplace/view/adminhtml/layout/mplace_mplace_new.xml – відображення форми додавання маркетплейса;

– MK/Mplace/view/adminhtml/layout/mplace_mplace_edit.xml – відображення форми редагування, відповідно.

Далі були створені UI компоненти для списку маркетплейсів і форми додавання маркетплейса. UI components в Magento 2 – це конфігуровані за допомогою XML та JavaScript/Knockout «віджети», з яких будуються складні інтерфейси, переважно в панелі адміністратора (гріди, форми, фільтри, масові дії), а також у деяких частинах фронтенду (checkout, customer forms). По суті, це шар над layout + block + template, який дозволяє описати складні UI-елементи декларативно в XML, а Magento сама збирає потрібні JS, HTML і дані. У нашому випадку ці компоненти описують всі компоненти форми або списку, джерела та провайдери даних, шляхи до контролерів що обслуговуватимуть запити форм. На рисунку 3.5 наведений фрагмент UI компонента MK/Mplace/view/adminhtml/ui_component/mk_mplace_listing.xml, що відповідає за контент сторінки списку маркетплейсів на панелі адміністратора.

```
<?xml version="1.0"?>
<listing xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Ui:etc/ui_configuration.xsd">
   <argument name="data" xsi:type="array">
      <item name="js_config" xsi:type="array">
         <item name="provider"
xsi:type="string">mk_mplace_listing.mk_mplace_listing_data_source</item>
      </item>
   </argument>
   <dataSource name="mk_mplace_listing_data_source"
component="Magento_Ui/js/grid/provider">
      <settings>
         <storageConfig>
            <param name="indexField" xsi:type="string">entity_id</param>
         </storageConfig>
         <updateUrl path="mui/index/render"/>
      </settings>
      <aclResource>MK_Mplace::mplace</aclResource>
```

Рисунок 3.5 – UI компонент, фрагмент декларативного опису (аркуш 1)

```
        <dataProvider
class="Magento\Framework\View\Element\UiComponent\DataProvider\DataProvider"
name="mk_mplace_listing_data_source">
            <settings>
                <requestFieldName>entity_id</requestFieldName>
                <primaryFieldName>entity_id</primaryFieldName>
            </settings>
        </dataProvider>
    </dataSource>
    <listingToolbar name="listing_top">
    …
```

Рисунок 3.5 – UI компонент, фрагмент декларативного опису (аркуш 2)

Наступним кроком були реалізовані контролери (controllers), що обробляють дії форм а також створене нове меню панелі управління для виклику цих форм. Також були описані ACL що відповідають за права доступу до розділу маркетплейсів і основні дії із їх сутностями.

Рисунок 3.6 ілюструє загальний вигляд сторінки списку маркетплейсів (listing).

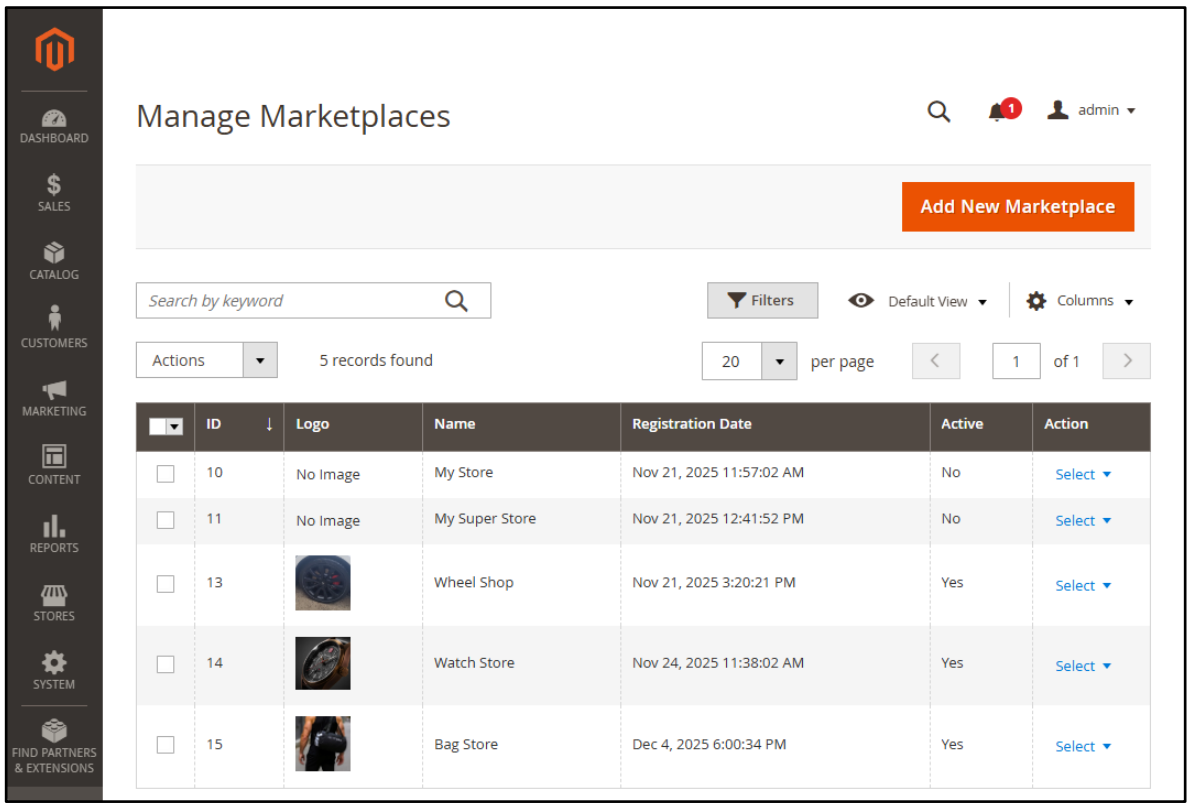


Рисунок 3.6 – Сторінка списку маркетплейсів

На ілюстрації можемо бачити список маркетплейсів а також елементи керування ними: фільтрацію, пагінацію, пошук, дії з маркетплейсами, массові дії (mass actions) а також кнопку додавання нового запису – створення маркетплейса.

На рисунку 3.7 зображено форму створення нового маркетплейса, частину основних полів і доступні дії з формою.

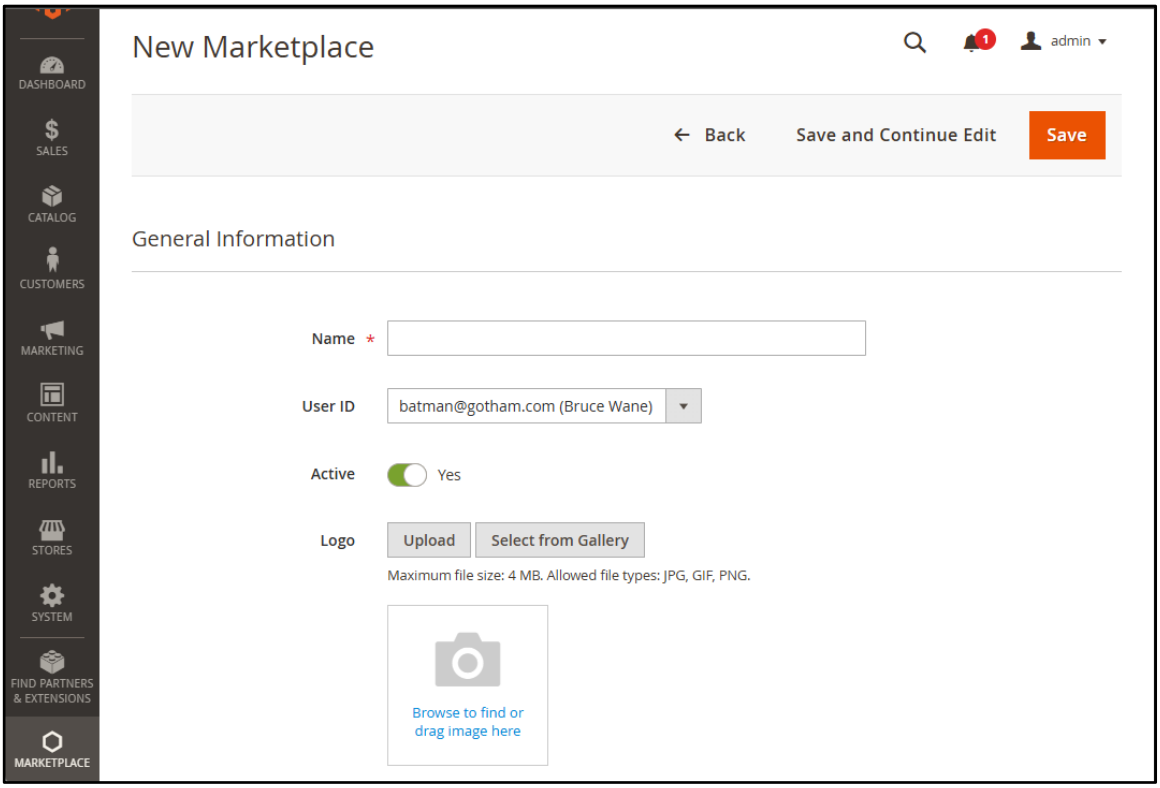


Рисунок 3.7 – Форма створення нового маркетплейса

Перейдемо до реалізації функціоналу реєстрації нового маркетплейса для фронтенда. Як і сторінки адмін панелі, фронтенд-функціонал потребує реалізації згідно з розширеною MVC моделлю CMS Magento 2. Проте на відміну від реалізації сторінок для панелі керування – для фронтенду використовуються блоки (blocks) і шаблони (templates) замість UI компонентів.

Реалізація має наступну структуру:

– MK/Mplace/etc/frontend/routes.xml – реєструє фронтовий маршрут mplace;

– MK/Mplace/view/frontend/layout/mplace_register_index.xml – файл макету (layout), оголошує блок, що відображається на сторінці і шаблон відображення (template);
– MK/Mplace/view/frontend/templates/register.phtml – безпосередньо шаблон контенту що виводиться через рендерінг блоку на фронтенд, містить форму реєстрації, використовує HTML синтаксис з виведенням даних контексту за допомогою операторів PHP;
– MK/Mplace/Block/Register.php – обробляє форму перед рендерінгом;
– MK/Mplace/Controller/Register/Index.php – контролер, показує сторінку з формою;
– MK/Mplace/Controller/Register/Post.php – приймає POST з форми, валідує дані, викликає MplaceRepository::save() і робить редіректі повертає повідомлення, якщо є.

Рисунок 3.8 ілюструє фронтенд форму реєстрації маркетплейса.

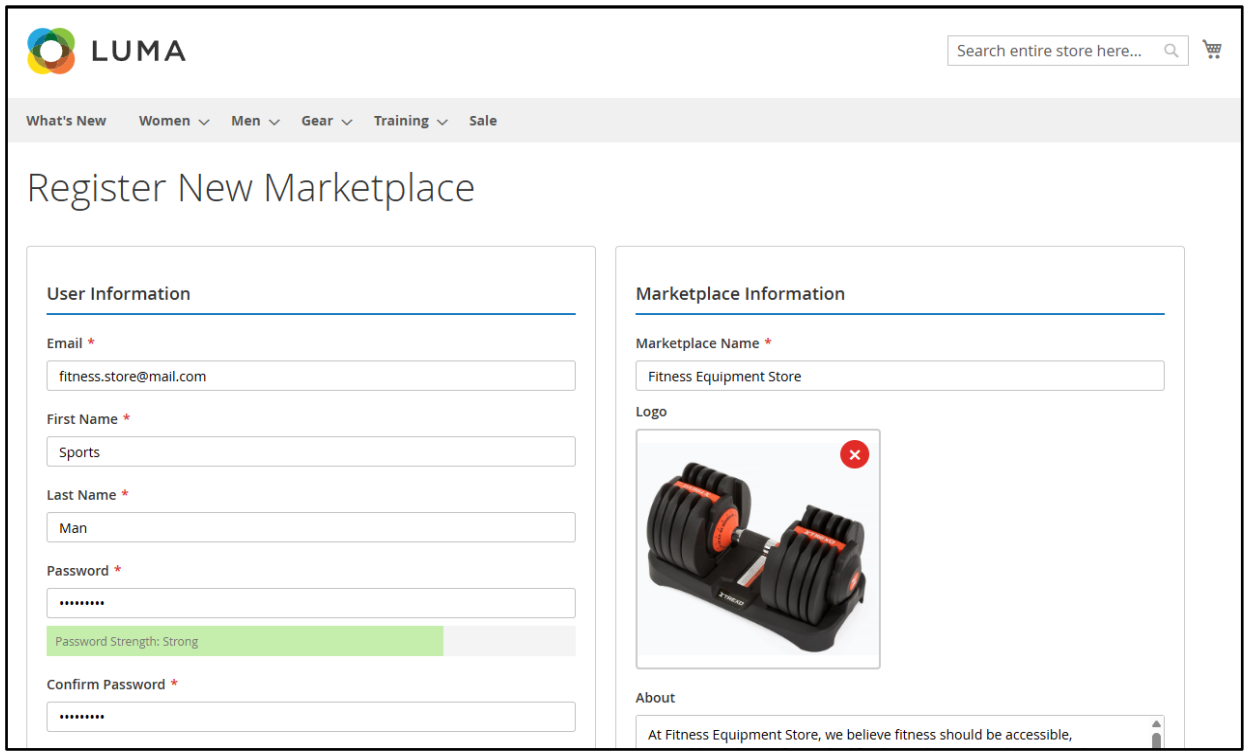


Рисунок 3.8 – Форма створення нового маркетплейса на фронтенді

Додатково на форму була додана валідація полів з використанням вбудованого в систему функціоналу а також імплементовано індикацію

надійності пароля за допомогою вбудованого JS віджета password-strength-meter. Для зручності також був реалізований більш орієнтований на користувача завантажувач зображень, що будуть використані як логотип маркетплейса.

Обробка форми відбувається у POST-контролері MK/Mplace/Controller/Register/Post.php. Згідно до рекомендацій з розробки, для обробки даних форми була створена додаткова сервіс модель MK\Mplace\Model\Service\MplaceEntity. Ця модель ініціалізується у контролері і відповідає за обробку даних. Це дозволило залишити у контролері тільки притаманний йому функціонал а бізнес-логіку делегувати до сервіс моделі. Реалізацію контролера відображає рисунок 3.9.

```
class Post extends Action
{
    public function __construct(
        private readonly Context $context,
        private readonly RedirectFactory $redirectFactory,
        private readonly MplaceEntity $mplaceEntity
    ) {
        parent::__construct($context);
    }

    public function execute() : \Magento\Framework\Controller\Result\Redirect
    {
        $resultRedirect = $this->redirectFactory->create();
        $request = $this->getRequest();
        if ($request->isPost()) {
            $data = $request->getPostValue();
            try {
                $adminUserId = $this->mplaceEntity->createMplaceUserFE($data);
                $logo = $this->mplaceEntity->processLogoFE($_FILES);
                $handle = $this->mplaceEntity->processHandleFE($data['marketplace_name']);
                $this->mplaceEntity->saveMplaceFE($data, $handle, $logo, $adminUserId);
                $this->messageManager->addSuccessMessage(__('Marketplace registered
successfully!'));
                $resultRedirect->setPath('*/*');
return $resultRedirect;
```

Рисунок 3.9 – POST-контролер форми (аркуш 1)

```
            } catch (\Exception $e) {
                $this->messageManager->addErrorMessage($e->getMessage());
                $resultRedirect->setPath('mplace/register');
                return $resultRedirect;
            }
        }
        $resultRedirect->setPath('*/*');

        return $resultRedirect;
    }
}
```

Рисунок 3.9 – POST-контролер форми (аркуш 2)

На ілюстрації можемо бачити наслідування батьківського класу, конструктор поточного класу, публічний метод execute(), у якому відбувається перевірка а потім і делегація обробки даних до сервісної моделі MplaceEntity. Модель створює нового користувача і новий маркетплейс, використовуючи патерн фабрика. Результатом роботи методу є створення нового облікового запису для власника маркетплейса, створення нового екземпляру сутності маркетплейсу зі збереженням у базі даних.

3.3 Сторінка перегляду маркетплейса

Для відображення сторінки маркетплейса без порушення візуальної цілісності сайту були використані стилі і візуальні елементи що використовуються візуальною темою Luma Theme яка є візуальним оформленням за замовчуванням для нових інсталяцій CMS Magento 2. Для шаблону сторінки маркетплейса були використані частини оригінальних шаблонів системи, нові ж елементи стилізувались на основі базових стилів Luma Theme.

Реалізація має наступну структуру:

– MK/Mplace/etc/frontend/routes.xml – містить маршрут, спільний для всіх сторінок маркетплейса;

- MK/Mplace/Controller/Router.php – містить логіку обробки нестандартних маршрутів, визначає чи є URL із запиту посиланням на сторінку маркетплейса;
- MK/Mplace/Controller/Index/Index.php – повертає сторінку маркетплейса, або 404, якщо маркетплейса не існує;
- MK/Mplace/view/frontend/layout/routing_index_index.xml – макет сторінки, містить опис блоків, їх класів і шаблонів а також інформацію про ієрархію блоків на сторінці;
- MK/Mplace/ViewModel/Mplace/View.php – модель відображення (view model), містить реалізації методів блока інформації про маркетплейс;
- MK/Mplace/view/frontend/templates/mplace/marketplace-info.phtml – шаблон блока інформації про маркетплейс;
- MK/Mplace/Plugin/FilterProductCollectionPlugin.php – плагін (plugin), дозволяє модифікувати стандартну логіку колекції і модифікувати набір елементів (в нашому випадку відбувається фільтрація продуктів по ідентифікатору маркетплейса для відображення на сторінці тільки тих продуктів, що відносяться до поточного маркетплейса).

Сторінка має вигляд стандартної сторінки категорій (category page), проте влаштована інакше. При переході на URL що містить у рядку /mplace/ відпрацьовує нестандартний маршрут, кастомний роутер (див. рис. 3.10) визначає чи є у рядку адреси хендл (handle), що належить одному з існуючих маркетплейсів, завантажує відповідний маркетплейс із репозиторія і записує ці дані у реєстр.

Далі відбувається рендеринг сторінки, виводиться блок інформації про маркетплейс і колекція товарів. Безпосередньо перед виведенням товарів спрацьовує плагін, що здійснить фільтрування товарів за ідентифікатором поточного маркетплейса.

Плагіни у CMS Magento 2 реалізовані за допомогою патерну проєктування перехоплювач (interceptor).

```
class Router implements RouterInterface
{
   /**
    * @param ActionFactory $actionFactory
    * @param MplaceRepositoryInterface $mplaceRepository
    */
   public function __construct(
      private readonly ActionFactory     $actionFactory,
      private readonly MplaceRepositoryInterface $mplaceRepository
   ){
   }
     /**
    * @param RequestInterface $request
    * @return ActionInterface|null
    * @throws Exception
    */
   public function match(RequestInterface $request): ?ActionInterface
   {
      $identifier = trim($request->getPathInfo(), '/');
        if (str_contains($identifier, 'mplace/') && !str_contains($identifier, 'mplace/register')) {
          $parts = explode('/', $identifier);

          if (count($parts) < 2 || empty($parts[1])) {
             return null;
          }

          $handle = $parts[1];
       try {
             $mplace = $this->mplaceRepository->getByHandle($handle);
             if (!$mplace->getId()) {
                return null;
             }
             $request->setModuleName('routing');
             $request->setControllerName('index');
             $request->setActionName('index');
             $request->setParams([
                'handle' => $handle
             ]);

             return $this->actionFactory->create(Forward::class, ['request' => $request]);
          } catch (NoSuchEntityException|\Exception $e) {
             throw new Exception('Error in MK Mplace Router: ' . $e->getMessage());
          }
       }

       return null;
   }
}
```

Рисунок 3.10 – Маршрутизатор для обробки запитів URL маркетплейсів

Готова сторінка має вигляд, проілюстрований на рисунку 3.11. На знімку екрану можна побачити сторінку маркетплейса що містить інформаційний блок з логотипом, назвою і описом, неактивні вкладки також містять інші дані маркетплейса. Нижче розміщена панель управління відображенням товарів з опціями сортування і відображення, під нею – безпосередньо список товарів маркетплейса. У лівій частині фільтр по атрибутам товарів, що дозволяє відобразити вибірку товарів по одному або декількох критеріях.

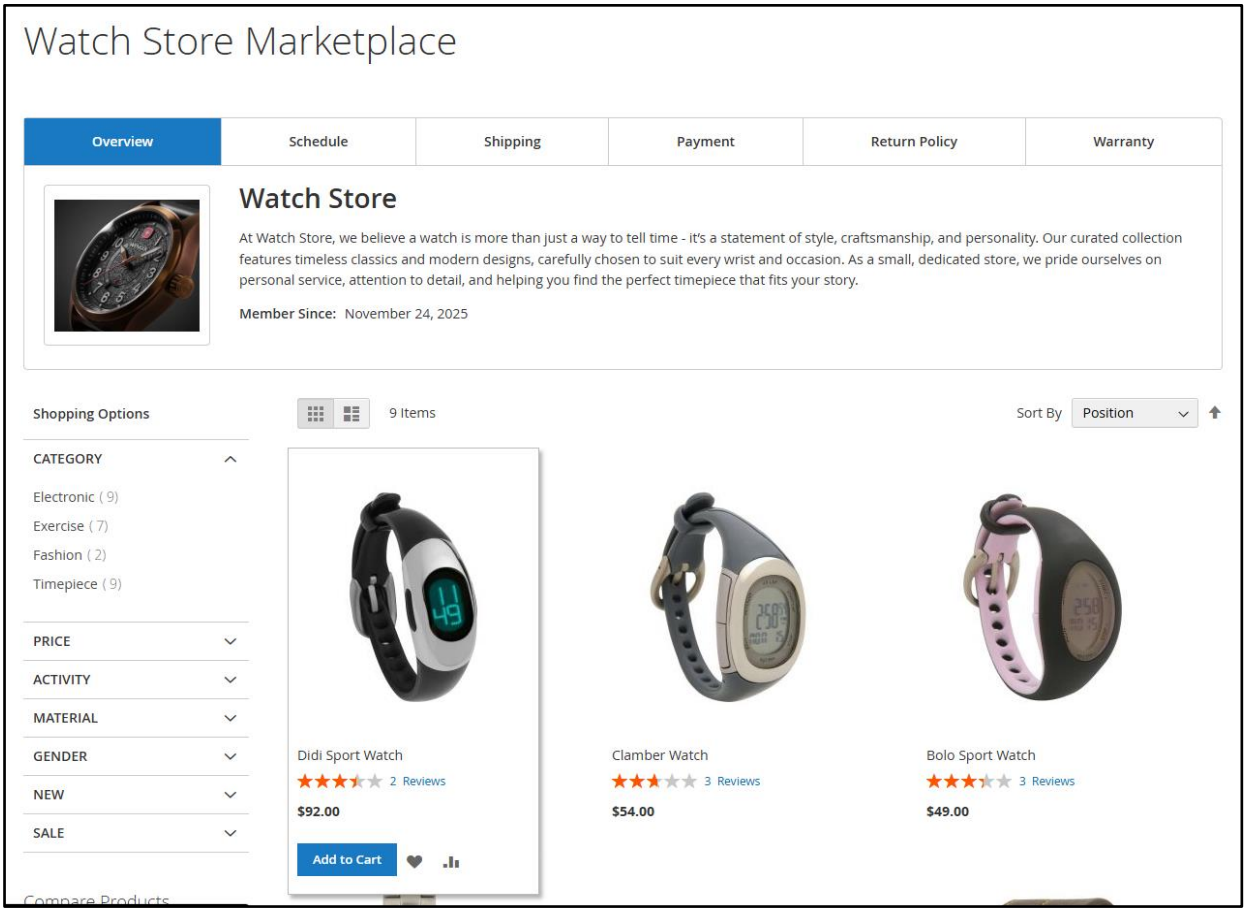


Рисунок 3.11 – Фронтенд сторінка маркетплейса

3.4 Сторінки категорій та продукта

Сторінки категорій та продукта зазнали мінімальних змін. З ціллю покращення інформування користувача магазином на кожен елемент списка продуктів категорії додано блок що інформує про продавця даного товару,

містить лого та назву маркетплейса. Подібний блок виводиться також і на сторінці продукта, на додачу до основної інформації цей блок містить посилання на сторінку зі всіма товарами продавця – сторінку маркетплейса.

Для сторінки категорій це реалізовано за допомогою плагіна, що спрацьовує для колекції товарів перед рендерингом і робить виборку по ідентифікаторам пов’язаних з товарами маркетплейсів, при рендерінгу шаблону ця інформація стає доступною на рівні сутності продукта.

Для сторінки продукта задіяний блок, модель відображення, шаблон і макет, що відповідає за те як і де цей блок виводиться. Вид сторінки продукта ілюструє рисунок 3.12.

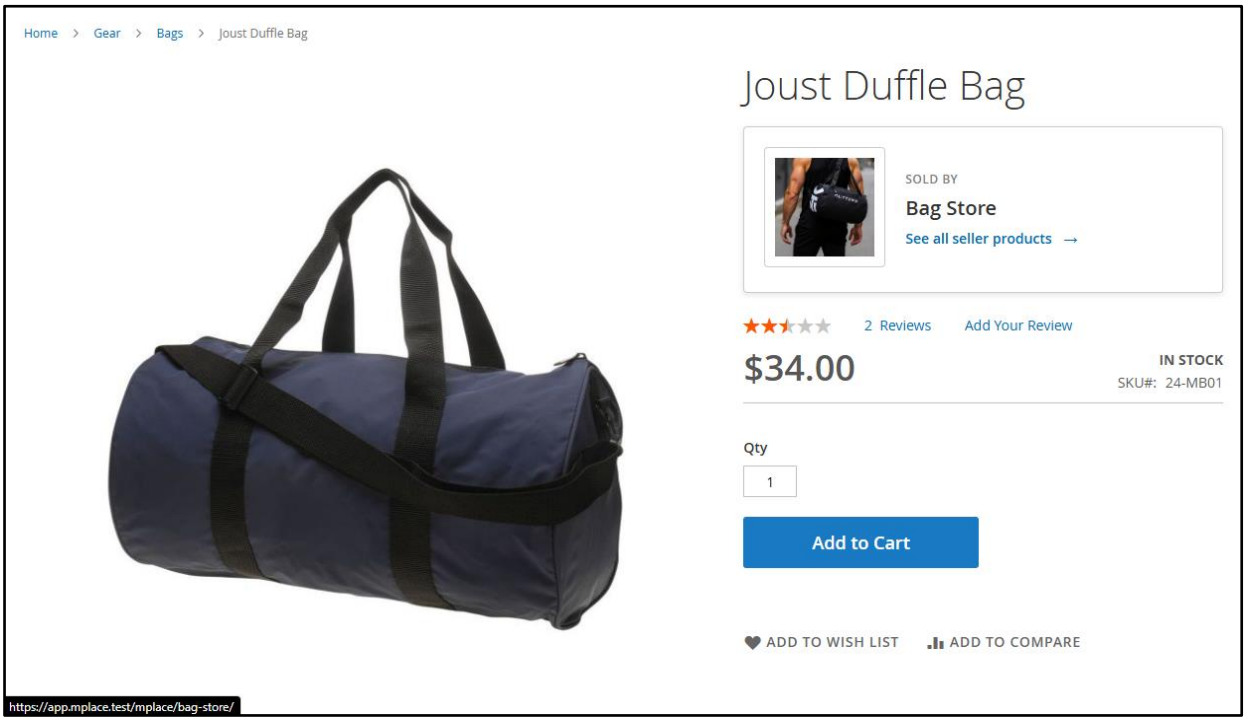


Рисунок 3.12 – Сторінка продукта

3.5 Кошик

Стандартний кошик CMS Magento 2 це передостанній крок перед оформленням замовлення. На цьому етапі користувач (покупець) може редагувати кількість та набір товарів що купує, бачити підсумкові суми до

сплати перед тим як почати оформлення замовлення. За замовчанням стандартний кошик призначений для роботи з одним магазином і оформлення всіх товарів у кошику в одне замовлення. Сесія покупця після оформлення очищується і всі продукти з кошика вилучаються, що логічно бо ці продукти оформлені у замовленні та сплачені. Проте стандартна поведінка кошика не відповідає вимогам функціоналу, коли на платформі є більше ніж один маркетплейс.

На етапі проєктування, у пункті 2.4.3 була прийнята концепція, за якої продукти у кошику групуються по маркетплейсу до якого належать з можливістю оформлення одного замовлення на маркетплейс за один раз.

Окрім вищезазначеного, залишається вирішити проблему з очищенням сесії після часткового оформлення замовлення, щоб дати користувачу змогу продовжити оформлення інших товарів, що знаходяться у кошику.

Це реалізовано наступним чином:

- MK\Mplace\Plugin\Checkout\Model\QuotePlugin::afterGetAllVisibleItems – це плагін, перебирає всі товари з кошика і групує їх список по ідентифікатору маркетплейса і зберігає цю інформацію у сесії;
- MK/Mplace/view/frontend/layout/checkout_cart_index.xml – макет сторінки кошика, перевизначає її шаблон;
- MK/Mplace/view/frontend/templates/checkout/cart/form.phtml – шаблон, відображає модифіковану форму кошика що групує елементи, фактично тепер це окрема форма для групи товарів для кожного маркетплейса;
- MK\Mplace\Plugin\Quote\Model\QuoteManagementPlugin::beforePlaceOrder – плагін, що спрацьовує перед створенням замовлення, зберігає інформацію про товари, які не вибрані для поточного замовлення;
- MK\Mplace\Plugin\Quote\Model\QuoteManagementPlugin::beforePlaceOrder – плагін, спрацьовує одразу після створення нового замовлення, зчитує з сесії необхідну інформацію про товари і знову додає ці товари до кошика.

Таким чином, плагін перехоплює стандартну поведінку Magento і гарантує, що товари інших маркетплейсів залишаються в кошику для подальшого оформлення.

На рисунку 3.13 наведено фрагмент плагіна, а саме метода MK\Mplace\Plugin\Quote\Model\QuoteManagementPlugin::afterPlaceOrder, що відповідає за відновлення товарів у кошику.

```
/**
    * After submit: Remove only selected marketplace items, keep others
    *
    * @param QuoteManagement $subject
    * @param int $result (order ID)
    * @param int $cartId
    * @return int
    */
   public function afterPlaceOrder(
      QuoteManagement $subject,
      $result,
      $cartId
   ): int {
      try {
         $selectedMplaceId = $this->checkoutSession->getSelectedMarketplaceId();
         $selectedItemIds = $this->checkoutSession->getSelectedMarketplaceItems();
         if ($selectedMplaceId !== null && !empty($selectedItemIds)) {
            $nonSelectedItems = $this->checkoutSession->getMplaceNonSelectedItems();
            if (!empty($nonSelectedItems)) {
               $quote = $this->checkoutSession->getQuote();

               if ($quote->getItemsCount() == 0) {
                  foreach ($nonSelectedItems as $itemData) {
                     try {
                           $buyRequest = new
\Magento\Framework\DataObject($itemData['options']);
                        $product = $this->productRepository->getById($itemData['product_id']);
                        if ($product && $product->getId()) {
                           $quote->addProduct($product, $buyRequest);
                         }
                     } catch (\Exception $e) {
                        $this->logger->error('Error restoring item ' . $itemData['sku'] . ': ' . $e-
>getMessage());
```

Рисунок 3.13 – Функціонал відновлення продуктів у кошику (аркуш 1)

```
                    }
                }
                $quote->collectTotals();
                $this->quoteRepository->save($quote);
            }
        }
    $this->checkoutSession->unsMplaceNonSelectedItems();
   }
    } catch (\Exception $e) {
        $this->logger->error('Error in afterPlaceOrder: ' . $e->getMessage());
    }

    return $result;
}
```

Рисунок 3.13 – Функціонал відновлення продуктів у кошику (аркуш 2)

Загальний вигляд сторінки кошика з товарами згрупованими по одному маркетплейсу наведено на рисунку 3.14. На зображенні можна розрізнити повідомлення про те, що у кошику є товари різних маркетплейсів, групу товарів що належать до маркетплейса Bag Store, їх кількість, ціну та попередню суму до сплати. Кінцева сума може відрізнятися, оскільки при оформленні замовлення може додатись оплата за доставку або бути знижка.

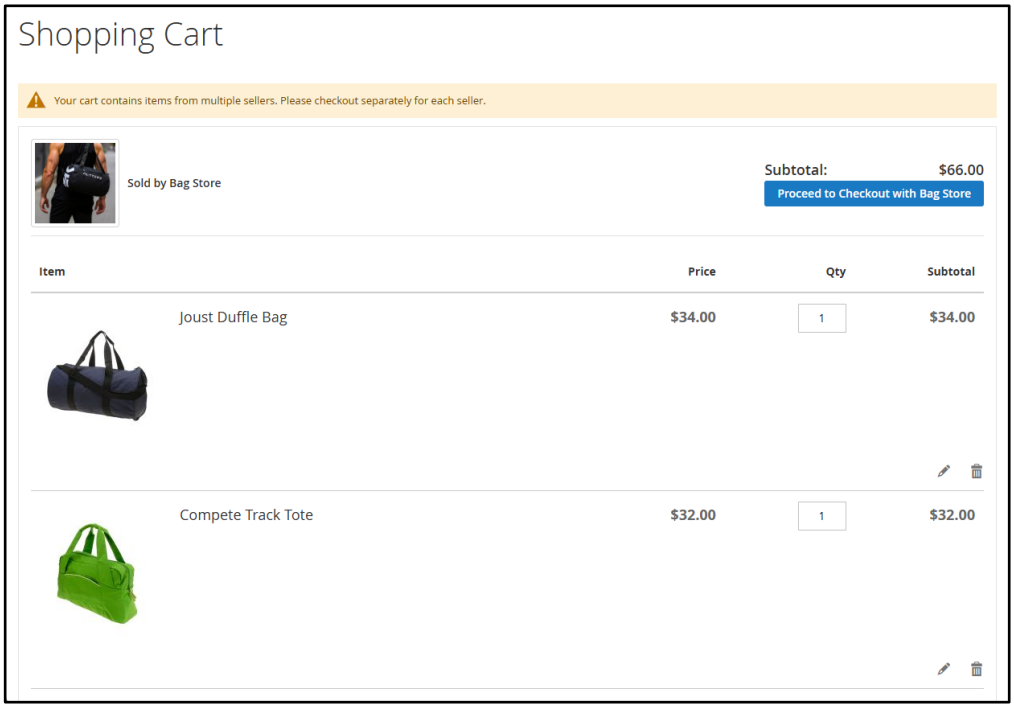


Рисунок 3.14 – Сторінка кошика з продуктами

3.6 Обробка замовлень

Для створення і обробки замовлень майже повністю використовується вбудований функціонал CMS Magento 2. Він дозволяє виконувати всі можливі операції з замовленням.

Власник маркетплейса має доступ до панелі керування CMS Magento 2 з певними обмеженнями, що визначаються роллю Marketplace Owner. Налаштування ролі здійснюється адміністратором системи, додаткові можливості управління можуть бути додані із його згоди.

Загальний вигляд панелі управління для власника маркетплейса зображено на рисунку 3.15.

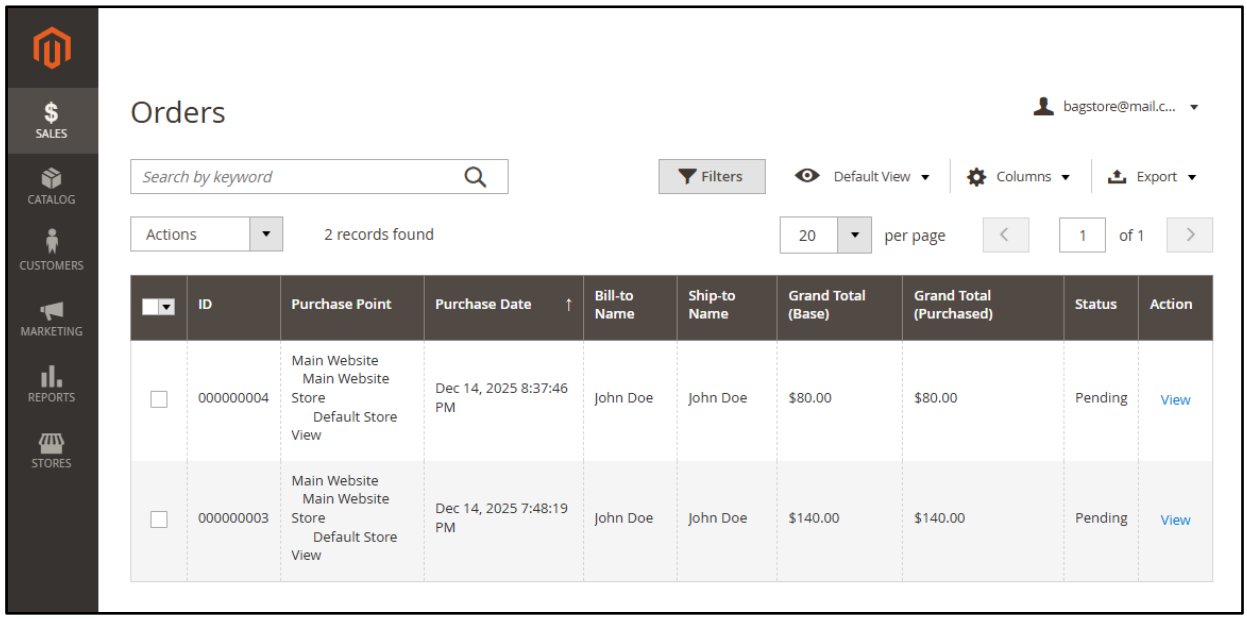


Рисунок 3.15 – Панель управління замовленнями

На ілюстрації можна побачити список замовлень (orders), основні поля що містять коротку інформацію, елементи управління пошуком та фільтрацією для більш зручної навігації.

Кожен власник маркетплейса бачить тільки замовлення що містять продукти з його маркетплейса. Цей функціонал реалізовано за допомогою плагіна, що перед рендерінгом UI компонента зі списком замовлень фільтрує колекцію замовлень за ідентифікатором маркетплейса.

3.7 Інтеграційні можливості

У CMS Magento 2 підтримка SOAP, REST і GraphQL потрібна для того, щоб зробити платформу відкритою й зручною для інтеграцій і побудови «headless»-рішень. Платформа підтримує наступні сценарії інтеграції:

- **SOAP** – історичний, але все ще використовується великими корпоративними системами (ERP, бухгалтерія, старі інтеграції), де прийнято працювати з WSDL-сервісами;
- **REST API** – стандартний спосіб інтеграції з веб- та мобільними застосунками, сервісами доставки, платіжними шлюзами, маркетплейсами, сторонніми CRM/ERP (дозволяє виконувати CRUD-операції над сутностями Magento);
- **GraphQL** – сучасний API для фронтендів і мобільних клієнтів у headless-архітектурі: дає можливість запитувати рівно ті дані, які потрібні, в одному запиті, що особливо важливо для продуктивності SPA/PWA та мобільних застосунків.

Розроблений функціонал модуля маркетплейса MK\Mplace підтримує інтеграційні рішення на основі REST API і GraphQL. Підтримка SOAP з огляду на технічну застарілість і меншу розповсюдженість додана не була, проте, зважаючи на модульність платформи – може бути реалізована пізніше, за потреби.

3.7.1 REST API

Підтримку REST API реалізовано через стандартний механізм CMS Magento 2: webapi.xml + Repository інтерфейси. Реалізація у модулі має наступну структуру:

- MK/Mplace/etc/webapi.xml – оголошує REST-маршрути, кожен маршрут містить посилання на клас і метод, що обслуговують

маршрут а також рівень доступу, необхідний для отримання даних.

– MK/Mplace/Api – містить класи інтерфейсів, так звані Service Contracts;
– MK/Mplace/Api/Data – містить класи інтерфейсів даних;
– MK/Mplace/Model – реалізації вищезазначених інтерфейсів.

Рисунок 3.16 містить фрагмент опису REST маршрутів з webapi.xml.

```
<route url="/V1/mplace" method="POST">
    <service class="MK\Mplace\Api\MplaceRepositoryInterface" method="save"/>
    <resources>
      <resource ref="MK_Mplace::save"/>
    </resources>
  </route>
```

Рисунок 3.16 – Приклад опису REST маршрутів

Перелік REST API маршрутів для модуля маркетплейса MK\Mplace наведений у таблиці 3.1.

Таблиця 3.1 – REST API маршрути модуля MK\Mplace

Метод	**URL**	**Опис**
GET	/V1/mplace/list	Отримання списку маркетплейсів
GET	/V1/mplace/:id	Отримання маркетплейса за ID
GET	/V1/mplace/handle/:handle	Отримання маркетплейса за handle (URL-ключ)
POST	/V1/mplace	Отримання нового маркетплейс
PUT	/V1/mplace/:id	Оновлення існуючого маркетплейса
DELETE	/V1/mplace/:id	Видалення маркетплейса
GET	/V1/mplace/:mplaceId/products	Отримання продукта маркетплейса
POST	/V1/mplace/:mplaceId/products	Додавання продукта до маркетплейса
DELETE	/V1/mplace/:mplaceId/products/:productId	Видалення продукта з маркетплейса

Продовження таблиці 3.1

Метод	**URL**	**Опис**
PUT	/V1/mplace/:mplaceId/products/:productId/position	Оновлення позиції продукта
POST	/V1/mplace/:mplaceId/products/bulk	Масове додавання продуктів до маркетплейса

3.7.2 GraphQL

Підтримку GraphQL реалізовано через стандартний механізм Magento 2: schema.graphqls + Resolver класи. Це означає що реалізація включає у себе:

– MK/Mplace/etc/graphql/schema.graphqls – декларує типи, запити й мутації GraphQL, прив’язує поля до PHP-резолверів;

– MK/Mplace/GraphQl – містить класи резолверів, що обслуговують схему.

Класи-резолвери логічно розподіляються на два типи за результатом запиту: Mutations – це запити на зміни (див. табл. 3.2) і Queries – запити на читання відповідно (див. табл. 3.3).

Таблиця 3.2 – Mutations

Query	**Resolver**	**Опис**
createMarketplace	Mutation\CreateMarketplace	Створення маркетплейса
updateMarketplace	Mutation\UpdateMarketplace	Оновлення маркетплейса
deleteMarketplace	Mutation\DeleteMarketplace	Видалення маркетплейса
addProductToMarketplace	Mutation\AddProductToMarketplace	Додавання продукта
removeProductFromMarketplace	Mutation\RemoveProductFromMarketplace	Видалення продукта
updateMarketplaceProductPosition	Mutation\UpdateMarketplaceProductPosition	Зміна позиції продукта
bulkAddProductsToMarketplace	Mutation\BulkAddProductsToMarketplace	Масове додавання

Таблиця 3.3 – Queries

Query	**Resolver**	**Опис**
marketplaces	Query\Marketplaces	Список маркетплейсів з фільтрами
marketplace(id)	Query\MarketplaceById	Один маркетплейс за ID
marketplaceByHandle(handle)	Query\MarketplaceByHandle	Маркетплейс за URL-ключем
marketplaceProducts(marketplace_id)	Query\MarketplaceProducts	Продукти маркетплейса

Приклад GraphQL запиту наведено на рисунку 3.17.

```
query {
  marketplace(id: 5) {
    id
    name
    handle
    logo_url
    products(pageSize: 10) {
      items {
        product {
          sku
          name
        }
        position
      }
      total_count
    }
  }
}
```

Рисунок 3.17 – Приклад GraphQL запиту

Результатом виконання цього запиту буде масив даних у форматі JSON, що міститиме тільки ті дані що містяться у запиті: ідентифікатор маркетплейса, назва, URL ключ, URL логотипу, список з десяти продуктів, що міститиме їх артикул, назву і позицію для виведення, загальну кількість продуктів.

На відміну від REST API запитів, GraphQL може обирати тільки необхідні для виведення поля, доступ до вкладених даних реалізований через зв’язки у схемі.

3.8 Тестування

CMS Magento 2 – велика, модульна платформа з великою кількістю залежностей (DI, події, плагіни, API). Будь-яка зміна в модулі може вплинути на інші частини системи, тому автоматизовані тести – обов’язкова практика для стабільності, особливо при оновленнях версій ядра чи розгортанні на продакшн.

Основні типи тестів, підтримувані платформою:

- Unit tests (юніт-тести) – тестують окремі класи в ізоляції – моделі, сервіси, хелпери, валідатори використовуючи PHPUnit та мокінг об’єктів;
- Integration tests (інтеграційні тести) – тестують взаємодію коду з реальним середовищем Magento: DI-контейнер, моделі, ResourceModel, події, плагіни, БД (використовують PHPUnit + спеціальне середовище з тестовою БД);
- Magento Functional Testing Framework (MFTF) – тестують end-to-end сценарії через браузер;
- Static tests (статичний аналіз) – тестують стиль і якість коду, відповідність стандартам за допомогою PHP CodeSniffer, MessDetector, PHPStan та інші.

Під час розробки було активно залучене статичне тестування, щоб забезпечити якісний код, що відповідає стандартам. Також поточному етапі розробки було прийняте рішення покрити юніт-тестами плагіни як найбільш ризикові й водночас найбільш зручні для ізольованого тестування точки розширення, де невелика помилка може непомітно вивести з ладу велику

частину функціоналу. Рисунок 3.18 ілюструє синтаксис вкладеного тесту для юніт-тестування плагіна колекції замовлень.

```
public function testBeforeLoadHandlesExceptionInShouldFilterCollection(): void
    {
        $this->collectionMock->expects($this->once())
            ->method('getFlag')
            ->with('mplace_filter_applied')
            ->willReturn(false);

        $this->collectionMock->expects($this->once())
            ->method('isLoaded')
            ->willReturn(false);

        $this->adminSessionMock->expects($this->once())
            ->method('getUser')
            ->willThrowException(new \Exception('Session error'));

        $this->loggerMock->expects($this->once())
            ->method('error')
            ->with($this->stringContains('Error checking if order collection should be filtered'));

        $result = $this->plugin->beforeLoad($this->collectionMock);

        $this->assertSame([], $result);
    }
```

Рисунок 3.18 – Загальний вигляд юніт-тесту

На ілюстрації можна побачити використання макетів (мокінг) – це техніка, коли замість реальних залежностей класу використовуються штучні об’єкти-заглушки (mocks/stubs), які поводяться так, як потрібно для тесту. Якщо коротко – це підміна реальних залежностей на контрольовані об’єкти, щоб юніт-тест був швидким, передбачуваним і перевіряв лише потрібну логіку. Результатом виконання тесту буде повідомлення, як продемонстровано на рисунку 3.19.

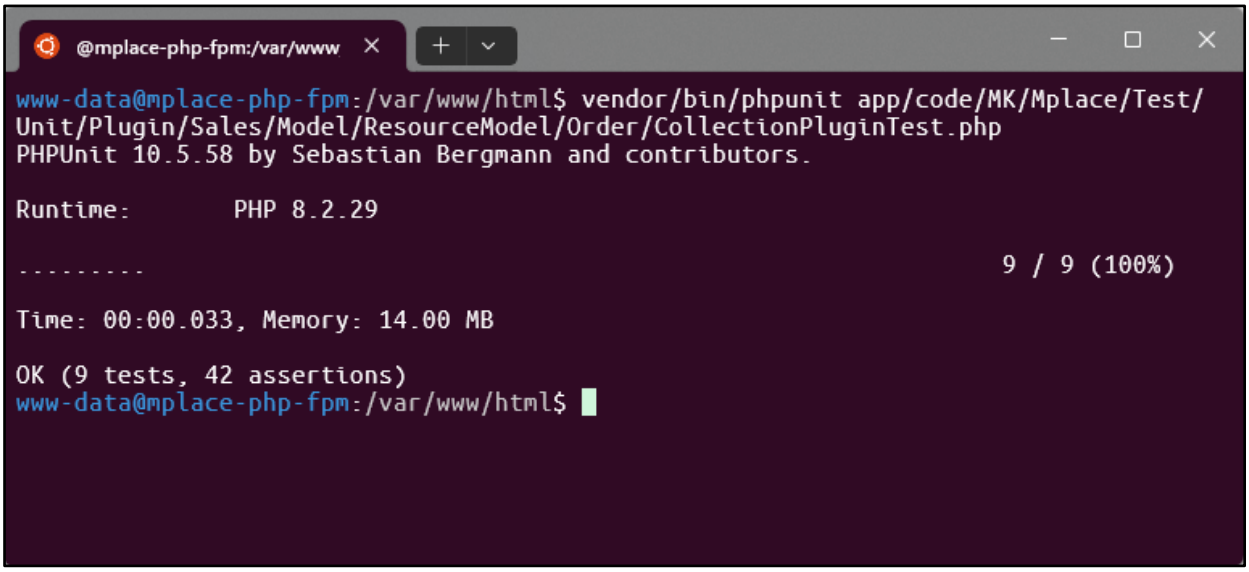


Рисунок 3.19 – Результат виконання юніт-тесту

ВИСНОВКИ

У даній кваліфікаційній роботі розроблено модуль маркетплейса для CMS Magento 2, який забезпечує основні функціональні можливості для користувачів і адміністраторів. Для реалізації проєкту використано PHP, Mysql і весь необхідний і підтримуваний CMS Magento 2 стек технологій для забезпечення ефективної роботи та масштабованості системи.

У першому розділі було проведено обґрунтування вибору технологій для розробки, зокрема, проведено аналіз існуючих платформ для онлайн-торгівлі що пропонують функціонал маркетплейса. Описано основні принципи розробки нового функціоналу для CMS Magento 2 і забезпечення його ефективності.

Другий розділ присвячений опису архітектури розробленого модуля, його основних компонентів, зокрема реалізації моделей даних, алгоритм взаємодії власника маркетплейса з системою, управління товарами, формування та обробки замовлень.

У третьому розділі було проведено розробку та тестування функціональності модуля, зокрема за допомогою статичних тестів і юніт тестування. Описано особливості розробки для CMS Magento 2 з прикладами реалізацій. Також було додано підтримку сучасних сценаріїв інтеграцій за допомогою REST API та GraphQL.

Розроблений модуль є ефективним і гнучким рішенням для онлайн-торгівлі, має високий рівень продуктивності та зручний інтерфейс для користувачів.

Розроблений модуль пропонує стабільний базовий функціонал маркетплейса для CMS Magento 2 а також відкриває масу можливостей для удосконалення під задачі і вимоги кожної окремої інсталяції.

ПЕРЕЛІК ПОСИЛАНЬ

1. E-commerce worldwide – statistics & facts. *Statista*. URL: https://www.statista.com/topics/871/online-shopping/ (дата звернення: 15.05.2025).

2. EVO Company. Ukrainian E-commerce Market Report 2023. Kyiv, 2023. 45 p.

3. Magento Open Source Statistics. *Adobe Commerce*. URL: https://business.adobe.com/products/magento/magento-commerce.html (дата звернення: 15.05.2025).

4. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

5. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

6. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River : Prentice Hall, 2008. 464 p.

7. PHP Documentation. PHP 8.2 Release Notes. URL: https://www.php.net/releases/8.2/en.php (дата звернення: 15.05.2025).

8. Adobe Commerce Developer Documentation. Magento 2.4 Technical Guidelines. URL: https://developer.adobe.com/commerce/php/development/ (дата звернення: 15.05.2025).

9. MySQL Documentation. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 15.05.2025).

10. Bergmann S. PHPUnit Documentation. URL: https://phpunit.de/documentation.html (дата звернення: 15.05.2025).

11. Shopify Collective: Multi-vendor marketplace solution. *Shopify Inc*. URL: https://www.shopify.com/collective (дата звернення: 16.05.2025).

12. Smith J., Anderson K. E-commerce Platform Selection Guide 2024. Boston : Digital Commerce Institute, 2024. 234 p.

13. WooCommerce Documentation. Dokan Multivendor Marketplace Plugin. URL: https://wedevs.com/dokan/ (дата звернення: 16.05.2025).

14. Thompson R. WordPress Performance Optimization for Large Scale E-commerce. O'Reilly Media, 2023. 312 p.

15. CS-Cart Documentation. Multi-Vendor Edition Overview. URL: https://www.cs-cart.com/multivendor.html (дата звернення: 16.05.2025).

16. Brown M., Davis L. Comparative Analysis of E-commerce Platforms. *Journal of Electronic Commerce Research*. 2023. Vol. 24, No. 2. P. 145–167.

17. Magento Open Source Architecture Guide. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/architecture/ (дата звернення: 16.05.2025).

18. Williams P. High-Performance Magento 2: Best Practices and Optimization. Apress, 2023. 456 p.

19. Chen H., Liu Y. Enterprise E-commerce Platform Selection: A Multi-criteria Decision Analysis. *International Journal of Information Management*. 2024. Vol. 75. P. 102–118.

20. Magento 2 Marketplace Extension Statistics. *Webkul Software*. URL: https://webkul.com/blog/magento2-marketplace/ (дата звернення: 16.05.2025).

21. Pricing — Magento 2 Marketplace. *Webkul Software*. URL: https://store.webkul.com/Magento-Multi-Vendor-Marketplace.html (дата звернення: 16.05.2025).

22. Magento 2 Marketplace Extensions Overview. *CedCommerce*. URL: https://cedcommerce.com/magento-2-extensions/ (дата звернення: 16.05.2025).

23. Kumar A., Singh R. Multi-channel E-commerce Integration Patterns. *Software Engineering Journal*. 2023. Vol. 18, No. 4. P. 78–95.

24. Multi Vendor Marketplace for Magento 2. *Amasty*. URL: https://amasty.com/multi-vendor-marketplace-for-magento-2.html (дата звернення: 16.05.2025).

25. Johnson M. Code Quality Analysis of Magento 2 Extensions. Technical Report. GitHub, 2023. URL: https://github.com/reports/magento2-quality (дата звернення: 16.05.2025).

26. Martinez L., O’Brien S. Limitations of Current Marketplace Solutions for Magento 2. *Magento Developer Conference Proceedings*. Las Vegas, 2023. P. 234–249.

27. Magento 2 Developer Documentation. Adobe Commerce. URL: https://developer.adobe.com/commerce/ (дата звернення: 17.05.2025).

28. Schwartz A., McCormick J. Magento 2 Development Cookbook. Birmingham: Packt Publishing, 2021. 456 p.

29. Patel K., Johnson R. Understanding Magento 2 Architecture Layers. *Journal of Software Architecture*. 2023. Vol. 15, No. 3. P. 112–128.

30. Dependency Injection in Magento 2. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/dependency-injection/ (дата звернення: 17.05.2025).

31. Service Contracts. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/service-contracts/ (дата звернення: 17.05.2025).

32. Newman S. Building Microservices: Designing Fine-Grained Systems : 2nd ed. O’Reilly Media, 2021. 612 p.

33. Plugins (Interceptors). *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/plugins/ (дата звернення: 17.05.2025).

34. Wilson D. Advanced Magento 2 Plugin Development. Apress, 2022. 289 p.

35. Events and Observers. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/events-and-observers/ (дата звернення: 17.05.2025).

36. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

37. Declarative Schema. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/declarative-schema/ (дата звернення: 17.05.2025).

38. Chen Y., Liu X. Database Migration Strategies for Enterprise Applications. *Database Systems Journal*. 2023. Vol. 14, No. 2. P. 45–62.

39. Web API Overview. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/webapi/get-started/ (дата звернення: 17.05.2025).

40. Buna S. Learning GraphQL and Relay. Birmingham : Packt Publishing, 2016. 388 p.

41. Richardson C., Smith F. RESTful Web APIs: Services for a Changing World. O'Reilly Media, 2013. 406 p.

42. Caching Overview. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/cache/ (дата звернення: 17.05.2025).

43. Indexing. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/development/components/indexing/ (дата звернення: 17.05.2025).

44. Access Control Lists (ACL). *Adobe Commerce*. URL: https://developer.adobe.com/commerce/php/tutorials/backend/create-access-control-list-rule/ (дата звернення: 17.05.2025).

45. Ferraiolo D., Kuhn R. Role-Based Access Control : 2nd ed. Artech House, 2009. 456 p.

46. UI Components Guide. *Adobe Commerce*. URL: https://developer.adobe.com/commerce/frontend-core/ui-components/ (дата звернення: 17.05.2025).

47. Thompson M. Modern Web Development with Magento 2. Packt Publishing, 2023. 512 p.

48. Rahman A., Patel S. Headless Commerce Architecture Patterns. *IEEE Software*. 2023. Vol. 40, No. 3. P. 78–85.

49. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

50. PHP Framework Interop Group. PSR Standards. URL: https://www.php-fig.org/psr/ (дата звернення: 17.05.2025).