Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

з магістр
(назва дисципліни)

на тему: Розробка і оптимізація модулів системи Magento для вебсайтів

Виконала: студент VI курсу, групи СПм-61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Чорний С.С.
(підпис) (прізвище та ініціали)

Керівник
(підпис) (прізвище та ініціали)

Нормоконтроль
(підпис) (прізвище та ініціали)

Завідувач кафедри
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет _____
(повна назва факультету)

Кафедра _____
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістра
(назва освітнього ступеня)

за спеціальністю _____
(шифр і назва спеціальності)

студенту Чорний Сергій Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка і оптимізація модулів системи Magento для вебсайтів.

Керівник роботи Петрик Михайло Романович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2025 року № _____.

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи Вихідними даними для виконання кваліфікаційної роботи є вимоги до функціональності та продуктивності e-commerce вебсайтів, документація платформи Magento 2, матеріали з архітектури CMS та систем електронної комерції, а також технічні специфікації для проєктування, розробки та оптимізації програмних модулів.

4. Зміст роботи (перелік питань, які потрібно розробити)
Проаналізувати предметну область електронної комерції та сучасні e-commerce платформи; дослідити архітектуру та функціональні можливості платформи Magento 2; сформулювати вимоги до програмних модулів та обґрунтувати вибір технологій розробки; спроєктувати структуру програмних модулів та бази даних; розробити та реалізувати власні модулі для платформи Magento 2; дослідити та впровадити методи оптимізації продуктивності модулів; провести тестування розроблених модулів та оцінити їх ефективність; проаналізувати питання охорони праці та безпеки в надзвичайних ситуаціях; оформити пояснювальну записку та зробити висновки за результатами виконаної роботи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Діаграма класів; діаграма послідовності; діаграма компонентів; діаграма діяльності ER-діаграма бази даних; структурна схема модулів Magento 2; схема взаємодії компонентів системи; скріншоти реалізованих модулів та результатів їх тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці			
Безпека в надз. ситуаціях			
Нормоконтроль			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з темою та завданням магістерської роботи.	01.09.2025 – 05.09.2025	Виконано
2	Підбір і аналіз літературних джерел з електронної комерції та Magento.	08.09.2025 – 19.09.2025	Виконано
3	Аналіз предметної області та формування вимог до системи.	22.09.2025 – 03.10.2025	Виконано
4	Дослідження архітектури платформи Magento 2.	06.10.2025 – 17.10.2025	Виконано
5	Проєктування програмних модулів Magento.	20.10.2025 – 31.10.2025	Виконано
6	Реалізація та інтеграція модулів у систему Magento.	03.11.2025 – 21.11.2025	Виконано
7	Оптимізація продуктивності та тестування модулів.	24.11.2025 – 05.12.2025	Виконано
8	Охорони праці.	08.12.2025 –10.12.2025	Виконано
9	Безпека в надзвичайних ситуаціях	11.12.2025 – 12.12.2025	Виконано
10	Формування/зшивання записки кваліфікаційної роботи		Виконано
11	Нормоконтроль		Виконано
12	Перевірка на плагіат	18.12.2025	Виконано
13	Попередній захист кваліфікаційної роботи	15.12.2025	Виконано
14	Захист кваліфікаційної роботи	24.12.2025	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра містить __ с., __ рис., __ табл., __ джерел.

Основні змістові блоки роботи охоплюють: об’єкт, мету та предмет дослідження, аналіз сучасних платформ, проєктування, розробку, оптимізацію й тестування модулів. У роботі використано технології PHP, Magento 2, HTML/CSS, MySQL та JavaScript.

Метою роботи є створення та оптимізація програмних модулів для платформи Magento 2, спрямованих на розширення функціональності системи та підвищення ефективності роботи e-commerce вебсайтів. Додатковим завданням є вивчення архітектури Magento й визначення елементів, які впливають на продуктивність і розширюваність платформи.

Під час виконання роботи проаналізовано сучасні системи керування інтернет-магазинами (WooCommerce, OpenCart, Shopify) та визначено їхні переваги й обмеження порівняно з Magento. Досліджено структуру каталогів і модульну архітектуру Magento 2, розглянуто механізми конфігурації, кешування та індексації.

У практичній частині розроблено два власні модулі для Magento 2, описано їх структуру, логіку роботи та взаємодію з базою даних. Проведено оптимізацію SQL-запитів, використано інструменти кешування та стандартні механізми Magento Framework. Тестування модулів здійснено в умовах реальної роботи вебмагазину.

Отримані результати підтверджують покращення швидкодії та зменшення навантаження на систему під час використання розроблених рішень. Робота може бути корисною розробникам і адміністраторам e-commerce платформ, які працюють із Magento та потребують розширення або оптимізації її функціональності.

ABSTRACT

The master's qualification work contains __ p., __ fig., __ tab., __ sources.

The main content blocks of the work include: the object, purpose and subject of the study, analysis of modern platforms, design, development, optimization and testing of modules. The work uses PHP, Magento 2, HTML/CSS, MySQL and JavaScript technologies.

The purpose of the work is to create and optimize software modules for the Magento 2 platform, aimed at expanding the functionality of the system and increasing the efficiency of e-commerce websites. An additional task is to study the Magento architecture and identify elements that affect the platform's performance and scalability.

During the work, modern online store management systems (WooCommerce, OpenCart, Shopify) were analyzed and their advantages and limitations compared to Magento were identified. The catalog structure and modular architecture of Magento 2 were studied, and the configuration, caching and indexing mechanisms were considered.

In the practical part, two custom modules for Magento 2 were developed, their structure, logic of operation and interaction with the database were described. SQL queries were optimized, caching tools and standard Magento Framework mechanisms were used. Modules were tested in real-world web store conditions.

The results obtained confirm the improvement of performance and reduction of system load when using the developed solutions. The work may be useful to developers and administrators of e-commerce platforms that work with Magento and need to expand or optimize its functionality.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД — база даних;

ООП — об'єктно-орієнтоване програмування;

ПК — персональний комп'ютер;

ПЗ — програмне забезпечення;

СУБД — система керування базами даних;

API (*Application Programming Interface*) — інтерфейс програмування застосунків;

ACL (*Access Control List*) — список контролю доступу;

B2B (*Business-to-Business*) — модель бізнесу «бізнес для бізнесу»;

B2C (*Business-to-Consumer*) — модель бізнесу «бізнес для споживача»;

CLI (*Command Line Interface*) — інтерфейс командного рядка;

CMS (*Content Management System*) — система керування контентом;

CRUD (*Create, Read, Update, Delete*) — чотири базові функції роботи з даними (створення, читання, оновлення, видалення);

CSS (*Cascading Style Sheets*) — каскадні таблиці стилів;

DI (*Dependency Injection*) — впровадження залежностей;

HTML (*HyperText Markup Language*) — мова розмітки гіпертексту;

HTTP (*HyperText Transfer Protocol*) — протокол передачі гіпертексту;

IoC (*Inversion of Control*) — інверсія керування;

JS (*JavaScript*) — мова програмування для створення інтерактивних вебсторінок;

JSON (*JavaScript Object Notation*) — текстовий формат обміну даними;

MVVM (*Model-View-ViewModel*) — архітектурний шаблон проєктування програмного забезпечення;

MySQL — реляційна система керування базами даних;

PHP (*Hypertext Preprocessor*) — мова програмування для розробки вебзастосунків;

PCI DSS (*Payment Card Industry Data Security Standard*) — стандарт безпеки даних індустрії платіжних карток;

SQL (*Structured Query Language*) — мова структурованих запитів;

UML (*Unified Modeling Language*) — уніфікована мова моделювання;

URL (*Uniform Resource Locator*) — єдиний локатор ресурсів (адреса в мережі);

XML (*eXtensible Markup Language*) — розширювана мова розмітки.

ЗМІСТ

ВСТУП

Розвиток електронної комерції вимагає від сучасних вебсистем високої гнучкості, масштабованості та здатності швидко адаптуватися до бізнес-процесів підприємств. В умовах постійного зростання обсягів даних особливої ваги набувають програмні платформи, що забезпечують стабільну роботу, можливість розширення та підтримку складних інтеграцій. Однією з найпотужніших систем цього класу є платформа Magento, яка базується на модульній архітектурі, розвинених механізмах конфігурації та широкому інструментарії для розробки власного функціоналу. Разом із тим, значна складність її внутрішньої структури потребує глибокого розуміння принципів роботи ядра та професійного підходу до реалізації модулів.

Для ефективного функціонування великомасштабних e-commerce рішень критично важливою є не лише наявність додаткових можливостей, а й оптимізація продуктивності, що передбачає зменшення навантаження на сервер, зниження часу відгуку та забезпечення стабільності під високим навантаженням. Саме тому питання створення якісних та оптимізованих модулів Magento є надзвичайно актуальним.

У межах даної роботи досліджуються теоретичні та практичні основи побудови модулів Magento 2, аналізуються сучасні підходи до інтеграції нового функціоналу, вивчаються особливості архітектури системи, механізми керування залежностями, структура конфігураційних файлів та принципи взаємодії з базою даних. Окрему увагу приділено питанням продуктивності, оскільки некоректна реалізація модулів може спричинити надмірну кількість SQL-запитів, перевантаження сервера та суттєве уповільнення роботи інтернет-магазину.

Актуальність теми зумовлена необхідністю створення модулів, які відповідають вимогам сучасної програмної архітектури, забезпечують гнучкість платформи Magento та дозволяють підвищити швидкодію вебресурсів. Для цього в роботі враховано особливості Magento Framework, механізми кешування, індексації та Dependency Injection.

Мета роботи – розробити та оптимізувати програмні модулі для платформи Magento 2, які розширюють функціональні можливості системи та сприяють підвищенню ефективності роботи інтернет-магазину.

Для досягнення мети визначено такі завдання:

- дослідити предметну область електронної комерції та архітектуру Magento 2;
- проаналізувати існуючі платформи й підходи до розширення Magento 2;
- сформувати функціональні та нефункціональні вимоги до модулів;
- виконати проєктування модулів із використанням UML-діаграм;
- реалізувати модулі відповідно до архітектурних принципів Magento Framework;
- провести оптимізацію запитів, механізмів кешування та взаємодії з базою даних;
- виконати тестування та оцінити продуктивність створених модулів.

Об’єкт дослідження – програмні модулі та механізми розширення функціональності платформи Magento 2. Предмет дослідження – методи розробки та оптимізації модулів у середовищі Magento.

Практична цінність роботи полягає у створенні двох робочих модулів та визначенні оптимальних підходів до їх проєктування, інтеграції та підвищення ефективності в реальних умовах функціонування інтернет-магазину.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

У межах даного розділу розглядається предметна область електронної комерції та сучасні програмні платформи для створення інтернет-магазинів. Проаналізовано основні тенденції розвитку e-commerce і можливості використання популярних CMS та e-commerce систем. Особливу увагу приділено архітектурним особливостям платформи Magento 2 та її перевагам порівняно з альтернативними рішеннями. На основі проведеного аналізу обґрунтовується доцільність використання Magento для розробки індивідуальних модулів, а також визначаються ключові проблеми продуктивності, що потребують подальшого дослідження та оптимізації.

1.1 Аналіз предметної області електронної комерції.

Електронна комерція (e-commerce) сьогодні є однією з найдинамічніших сфер цифрової економіки [1]. Вона охоплює процеси купівлі, продажу, оплати та доставки товарів і послуг через Інтернет, істотно впливаючи на розвиток сучасного бізнесу. Основна перевага “e-commerce” полягає у можливості автоматизувати бізнес-процеси, зменшити витрати на обслуговування клієнтів і розширити ринок збуту без значних капіталовкладень [2].

Електронна комерція охоплює широкий спектр бізнес-операцій, що здійснюються в онлайн-просторі. Залежно від того, хто є учасником угоди, виділяють кілька основних моделей електронної комерції [3]:

- **B2C (Business-to-Consumer)** — бізнес продає товари або послуги кінцевому споживачу. Ця модель є найпоширенішою серед онлайн-магазинів і вимагає зручного інтерфейсу користувача, високої швидкодії сайту, інтеграції з платіжними системами та службами доставки.

- **B2B (Business-to-Business)** — взаємодія між підприємствами, наприклад, оптовими постачальниками або корпоративними клієнтами. Для цієї моделі характерні складні системи управління замовленнями, обліку партій товарів, а також персоналізація цін і умов співпраці для різних партнерів.
- **C2C (Consumer-to-Consumer)** — торгівля між користувачами через онлайн-маркетплейси або платформи оголошень, такі як *OLX* чи *eBay*. У цій моделі важливими аспектами є безпека транзакцій, модерація контенту та система відгуків, що формує довіру між користувачами.
- **C2B (Consumer-to-Business)** — модель, у якій користувач пропонує товари або послуги бізнесу, наприклад, фріланс-послуги або контент для компаній.

Кожна з наведених моделей має свої особливості та специфічні вимоги до програмних систем, що автоматизують бізнес-процеси. Від правильно обраної архітектури та інструментів розробки залежить зручність використання системи, її масштабованість, швидкодія та безпека. Саме тому вибір оптимальної e-commerce платформи є ключовим етапом створення сучасного інтернет-магазину або корпоративного вебсайту.

Для ефективної роботи інтернет-магазину необхідна автоматизація основних бізнес-процесів, серед яких управління каталогом товарів, обробка замовлень, прийом платежів, робота з користувачами, аналітика та маркетинг [4].

Управління каталогом товарів передбачає створення, редагування та категоризацію позицій, додавання описів, характеристик і зображень. Це основа будь-якої торгової платформи, від якої залежить зручність пошуку та вибору товарів.

Обробка замовлень включає формування кошика, зміну статусів, відправку повідомлень клієнтам і інтеграцію з кур’єрськими службами для доставки.

Платіжні операції забезпечують прийом онлайн-платежів через банківські картки, електронні гаманці чи платіжні шлюзи. Вони повинні бути безпечними, стабільними та зручними для користувачів.

Управління користувачами включає реєстрацію, авторизацію, ведення особистих кабінетів, перегляд історії замовлень і використання бонусних систем.

Аналітика і звітність допомагають відстежувати продажі, популярність товарів, поведінку користувачів і ефективність маркетингових кампаній.

Крім того, системи електронної комерції мають підтримувати інструменти просування, такі як SEO-оптимізація, email-розсилки, знижки, акції та інтеграція із соціальними мережами. Автоматизація цих процесів дозволяє значно підвищити ефективність роботи магазину, скоротити час обробки замовлень і підвищити рівень задоволеності клієнтів.

Сучасні системи електронної комерції повинні бути зручними, швидкими та безпечними. Вони мають відповідати основним технічним і бізнес-вимогам, щоб забезпечувати стабільну роботу онлайн-магазинів і комфорт користувачів. Основні вимоги до якісної e-commerce системи такі:

- **Масштабованість** — підтримка зростання кількості користувачів і товарів без погіршення продуктивності.
- Гнучкість модульної архітектури — можливість інтеграції сторонніх сервісів і розробки власних модулів.
- Безпека даних — захист персональної інформації, безпечні платіжні транзакції, відповідність стандартам PCI DSS [4].
- Швидкодія та оптимізація — ефективне використання ресурсів сервера, кешування, індексація даних.
- Адаптивність — коректне відображення на різних пристроях і платформах.
- Інтеграційні можливості — підтримка API для взаємодії з ERP, CRM, службами доставки та маркетинговими платформами.

Отже, якісна система електронної комерції повинна поєднувати гнучкість, безпеку, швидкодію та зручність, щоб відповідати вимогам сучасного онлайн-бізнесу.

Сфера електронної комерції постійно розвивається під впливом нових технологій [5]. Сучасні онлайн-магазини прагнуть бути швидшими, зручнішими та більш персоналізованими, тому активно впроваджують інноваційні рішення:

- **Headless Commerce** — розділення фронтенду та бекенду, що дозволяє використовувати різні технології для відображення контенту та управління даними [6].
- **Інтелектуальна аналітика** — застосування машинного навчання для рекомендацій товарів, прогнозування попиту та персоналізації.
- **Мобільна комерція (m-commerce)** — оптимізація інтернет-магазинів для мобільних пристроїв і створення мобільних застосунків.
- **Omni-channel** — інтеграція різних каналів продажу (онлайн, офлайн, маркетплейси) для єдиного обліку та управління.
- **Штучний інтелект і чат-боти** — автоматизація підтримки клієнтів, персоналізовані пропозиції, рекомендаційні системи.
- **Технології блокчейн** — підвищення прозорості транзакцій і безпеки платежів [5].

Попри стрімкий розвиток електронної комерції, компанії стикаються з низкою проблем під час впровадження та експлуатації e-commerce систем. Однією з головних складностей є інтеграція з існуючими корпоративними системами, такими як ERP чи CRM. Невідповідність форматів даних або відсутність гнучких API може ускладнювати обмін інформацією між підсистемами.

Важливою проблемою є високі вимоги до продуктивності, особливо при великій кількості користувачів і товарів у каталозі. Для стабільної роботи необхідна оптимізація бази даних, ефективне кешування та використання потужних серверних рішень.

Не менш актуальним є питання безпеки даних. Під час онлайн-транзакцій потрібно забезпечити захист персональної інформації клієнтів і запобігти можливим кібератакам.

Додаткові труднощі виникають при підтримці багатомовності та багатовалютності, що є необхідним для компаній, які працюють на міжнародних ринках. Ще одним викликом є постійна підтримка та оновлення компонентів, адже програмні рішення швидко застарівають, і для забезпечення стабільної роботи системи необхідно регулярно оновлювати компоненти.

Урахування зазначених факторів є важливим етапом під час вибору платформи та розробки власних модулів, особливо для середніх і великих інтернет-магазинів, які потребують стабільності, безпеки та високої швидкодії.

Таким чином, електронна комерція є ключовим елементом сучасної цифрової економіки. Вона відкриває нові можливості для бізнесу, проте вимагає впровадження гнучких і надійних технологічних рішень. Надалі у роботі буде розглянуто існуючі платформи для реалізації систем електронної комерції, зокрема Magento, яка забезпечує високу гнучкість, масштабованість і можливість розробки власних модулів для розширення функціональності вебсайтів.

1.2 Огляд CMS та e-commerce систем.

Сучасний ринок вебсайтів і онлайн-магазинів формується на основі спеціалізованих програмних платформ, які можна умовно поділити на системи керування контентом (CMS) та e-commerce системи.

Системи керування контентом (CMS) призначені для створення, редагування та публікації цифрового контенту на вебсайтах без глибоких знань програмування [7]. CMS забезпечують зручну роботу з текстами, зображеннями, меню, шаблонами та користувачами. Вони часто використовуються для корпоративних сайтів, блогів та новинних порталів, а також можуть доповнюватися модулями для e-commerce. До популярних CMS належать WordPress, Joomla!, Drupal, які мають велику спільноту розробників та безліч розширень для різних потреб.

E-commerce системи — це платформи, спеціально створені для онлайн-продажу товарів та послуг. Вони включають інструменти для керування каталогом товарів, кошиком, замовленнями, оплатою та доставкою [8]. Деякі e-commerce системи є самостійними продуктами, інші працюють як плагіни до CMS. До популярних e-commerce систем належать Magento, WooCommerce, OpenCart та Shopify:

- Magento — потужна open-source платформа для середніх і великих магазинів, з модульною архітектурою, багатомовністю, багатомагазинністю та розширеними можливостями кастомізації.
- WooCommerce — плагін для WordPress, що перетворює сайт на інтернет-магазин. Простий у використанні та має велику кількість тем і плагінів.
- OpenCart — легка CMS для малих та середніх магазинів, що підтримує базову модульність, багатомовність і різні способи оплати.
- Shopify — SaaS-платформа, що надає готовий хмарний магазин із високою продуктивністю та простотою використання, але обмеженою кастомізацією.

Для наочного порівняння характеристик CMS та e-commerce систем наведена таблиця 1.1. Вона дозволяє проаналізувати сильні та слабкі сторони платформ, а також показує конкурентні переваги Magento як бази для подальшої розробки і оптимізації модулів.

Таблиця 1.1

Система	**Тип**	**Ліцензія**	**Масштабованість**	**Простота адміністрування**	**Модульність**	**Хостинг**
WordPress	CMS	Open Source	Середня	Висока	Висока	Власний
Joomla!	CMS	Open Source	Середня	Середня	Середня	Власний
Drupal	CMS	Open Source	Висока	Середня	Висока	Власний
Magento	E-commerce	Open Source	Висока	Середня	Висока	Власний
WooCommerce	E-commerce	Open Source	Середня	Висока	Середня	Власний
OpenCart	E-commerce	Open Source	Середня	Висока	Середня	Власний
Shopify	E-commerce	SaaS	Висока	Висока	Обмежена	Хмарний

Таблиця 1.1 – Порівняльні характеристики CMS та e-commerce систем

Ця таблиця показує, що Magento поєднує високу модульність, масштабованість та можливість інтеграцій, що робить її оптимальним вибором для розробки та оптимізації модулів у дипломній роботі [8].

1.3 Аналіз вимог до системи

Аналіз вимог включає визначення функціональних можливостей майбутніх модулів Magento, а також нефункціональних характеристик, необхідних для забезпечення стабільної, безпечної та продуктивної роботи системи. Вимоги сформовані з урахуванням особливостей архітектури Magento 2 та специфіки e-commerce застосунків.

1.3.1 Функціональні вимоги

Функціональні вимоги визначають, які конкретні дії та можливості повинні забезпечувати розроблювані модулі. Основні вимоги:

1) Інтеграція з ядром Magento 2 без модифікації системних файлів.
2) Розширення стандартного функціоналу шляхом створення контролерів, моделей, плагінів, обсерверів, UI-компонентів.
3) Можливість конфігурації модулів через адміністративну панель (Stores → Configuration).
4) Логування роботи модулів та фіксація системних подій для подальшого аналізу продуктивності.
5) Підтримка багатомовності та багатомагазинності відповідно до можливостей Magento.
6) Сумісність з REST API та можливість взаємодії з зовнішніми сервісами.
7) Керування модулями через CLI-команди (enable, disable, upgrade).

8) Коректне відображення на фронтенді із використанням стандартних шаблонів та layout-структури.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги описують якість роботи модулів, їх продуктивність, безпеку та масштабованість.

Вимоги до продуктивності:

1) Модулі повинні мінімізувати кількість SQL-запитів та уникати дублювання логіки.
2) Вся логіка повинна бути сумісною з кешуванням Magento (Full Page Cache, Redis, Varnish).
3) Файли CSS/JS мають бути оптимізовані та мінімізовані.
4) Модулі не повинні погіршувати швидкість завантаження сторінок.

Вимоги до безпеки:

1) Використання системи прав доступу ACL для контролю дій адміністратора.
2) Валідація вхідних даних у формах та API-запитах.
3) Захист від типових загроз: SQL-ін’єкцій, XSS, CSRF.
4) Обмежений доступ до налаштувань модулів залежно від ролей користувача.

Вимоги до масштабованості:

1) Модулі повинні забезпечувати коректну роботу при збільшенні каталогу та кількості користувачів.
2) Підтримка роботи в кластерному або хмарному середовищі.
3) Сумісність з сервісами Elasticsearch, RabbitMQ, CDN.
4) Відсутність жорстких залежностей від конкретного серверного середовища.

1.4 Проблеми продуктивності Magento та методи оптимізації.

Magento є потужною платформою, однак його модульна структура потребує значних ресурсів, що підвищує вимоги до серверного оточення [9]. У практичних впровадженнях продуктивність системи може знижуватися внаслідок значної кількості запитів, великого обсягу даних та використання сторонніх модулів. Тому важливо визначити основні типи проблем і методи їх вирішення.

Таблиця 1.2

Проблема	**Методи оптимізації**
Повільне завантаження сторінок	● Використання кешування (Varnish, Redis); ● оптимізація зображень; ● увімкнення кешу Magento
Високе навантаження на базу даних	● Оптимізація запитів; ● використання реплікації; ● застосування індексування БД
Повільна робота фронтенду	● Мінімізація CSS/JS; ● активація стиснення; ● застосування CDN
Низька продуктивність при великій кількості продуктів	● Оптимізація індексаторів; ● використання Elasticsearch; ● оновлення структури каталогів
Недостатня масштабованість	● Використання хмарної інфраструктури; ● горизонтальне масштабування; ● розподіл сервісів
Неефективна робота кешу	● Налаштування Redis/Varnish; ● очищення за розкладом; ● моніторинг кеш-хітів

Таблиця 1.2 – Основні проблеми продуктивності Magento та методи їх оптимізації.

1.4.1 Основні проблеми продуктивності Magento

Перед наведенням методів оптимізації важливо визначити основні фактори, що негативно впливають на роботу платформи [9]:

1) Повільне завантаження сторінок.

Платформа обробляє значний обсяг PHP-коду та виконує багато серверних операцій. Наявність великої кількості розширень може призводити до дублювання обчислень або надмірної кількості запитів.

2) Навантаження на базу даних.

У магазинах із великим каталогом кількість SQL-запитів істотно зростає. Неефективні запити та недостатня оптимізація таблиць спричиняють затримки при формуванні сторінок.

3) Повільна індексація каталогу.

Magento використовує індекси для цін, атрибутів і категорій. При великих обсягах даних процес індексації стає тривалим та може впливати на продуктивність сервера.

4) Надлишкова кількість модулів.

Частина модулів дублює функціональність або виконує ресурсоємні операції, що збільшує час відповіді системи.

5) Неоптимізовані статичні ресурси.

Невідповідно стиснуті та не мінімізовані CSS/JS-файли, а також великі зображення збільшують час завантаження сторінок на клієнтській стороні.

1.4.2 Методи оптимізації продуктивності Magento

Оптимізація продуктивності системи Magento є важливим етапом забезпечення стабільної, швидкої та масштабованої роботи інтернет-магазину. Для підвищення ефективності функціонування застосовуються різні технічні методи, які охоплюють серверну складову, базу даних, фронтенд та роботу встановлених модулів [10].

1) Кешування

Magento підтримує кілька типів кешу, тому одним із перших кроків є увімкнення всіх доступних кешів. Це дозволяє зменшити навантаження на сервер та прискорити відображення сторінок.

2) Оптимізація бази даних

Для зменшення часу виконання запитів проводиться очищення логів, оптимізація індексів та налаштування кешування в пам'яті. У проектах із великим навантаженням додатково застосовується реплікація БД.

3) Оптимізація фронтенду

До покращення швидкодії з боку клієнта належать мінімізація CSS та JavaScript, стиснення зображень і використання CDN. Це зменшує обсяг даних, які отримує браузер, і прискорює завантаження сторінок.

4) Оптимізація модулів

Велика кількість сторонніх модулів може уповільнювати роботу магазину, тому рекомендується періодично перевіряти встановлені розширення, вимкнути непотрібні та контролювати ті, що створюють зайві запити.

5) Серверна оптимізація

Застосування PHP-FPM, сучасних версій PHP, SSD-накопичувачів та Elasticsearch для каталогу дозволяє суттєво прискорити роботу Magento. Також важливі правильні налаштування веб-сервера.

Отже, продуктивність Magento залежить від стану бази даних, кількості модулів, конфігурації сервера та налаштування кешу. Поєднання методів оптимізації дозволяє забезпечити стабільну та швидку роботу інтернет-магазину як у звичайних умовах, так і при високих навантаженнях.

1.5 Висновки до першого розділу

У цьому розділі було проаналізовано предметну область електронної комерції та визначено місце платформи Magento серед сучасних рішень для створення інтернет-магазинів. Magento є популярною платформою серед інших

рішень завдяки високій гнучкості, масштабованості та можливості розробки власних модулів. Розглянуто функціональні й нефункціональні вимоги програмної системи Magento. Досліджено основні проблеми продуктивності Magento та визначено методи їх оптимізації для забезпечення стабільної та безперебійної роботи проектів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У даному розділі розкриваються питання проєктування та розробки програмних модулів для платформи Magento 2. Обґрунтовується вибір технологій і інструментів розробки, аналізується архітектура Magento, її модульна структура та механізми конфігурації. Виконується проєктування структури модулів із використанням UML-діаграм, визначається логіка взаємодії основних компонентів і, за потреби, структура бази даних. Також розглядаються особливості реалізації модулів, їх інтеграції у систему та підходи до оптимізації продуктивності.

2.1 Обґрунтування вибору технологій та інструментів (PHP, Magento 2, MySQL)

Для ефективної розробки та оптимізації модулів системи Magento необхідно використовувати сучасні та підтримувані технології, що забезпечують стабільну роботу вебсайтів, високу продуктивність та безпеку. При виборі технологій враховувалися офіційні вимоги Magento, рекомендації спільноти розробників та досвід успішних проєктів e-commerce [11].

2.1.1 Версія Magento

Для реалізації модулів обрано Magento Open Source 2.4.x, що є останньою стабільною версією платформи [11]. Основні переваги вибору:

- Підтримка PHP 8.x та сучасних стандартів об’єктно-орієнтованого програмування (ООП), що дозволяє створювати гнучкі та масштабовані модулі.
- Модульна архітектура, що забезпечує можливість додавати нові функції без зміни ядра системи.
- Сумісність з MySQL 8.x та MariaDB 10.x, що гарантує швидку та надійну роботу бази даних.

- Покращена безпека, регулярні оновлення та підтримка спільноти.
- Підтримка сучасних front-end технологій та інтеграцій із сторонніми сервісами (Elasticsearch, Redis, Varnish).

2.1.2 Мови програмування та технології

Розробка модулів Magento потребує використання декількох мов і технологій:

- PHP 8.x – основна мова для реалізації бізнес-логіки, взаємодії з базою даних, створення модулів та обробки запитів користувачів [12].
- HTML, CSS, JavaScript – для формування інтерфейсу користувача, стилізації сторінок, реалізації динамічних та інтерактивних елементів [13 – 15].
- XML – для конфігурації модулів, layout, dependency injection та налаштування маршрутизації запитів [16].
- SQL / MySQL – для зберігання та обробки даних про користувачів, товари, замовлення та транзакції [17 – 18].

2.1.3 Додаткові технології та інструменти

Для забезпечення стабільності, продуктивності та зручності розробки використовуються:

- Composer – керування залежностями та пакетами Magento, що спрощує встановлення та оновлення модулів.
- Git – контроль версій коду, спільна робота над проєктом, історія змін та відкат до попередніх станів.
- Redis / Varnish – кешування сторінок та даних для прискорення роботи сайту та зменшення навантаження на сервер.
- Elasticsearch – потужний механізм пошуку по каталогу товарів, що забезпечує швидкий та точний пошук для користувачів.

- Docker (опційно) – створення ізольованого середовища розробки та тестування, що дозволяє уникати конфліктів між різними версіями технологій.

2.2 Аналіз архітектури Magento: структура каталогів, модульність, XML-конфігурації

Magento є однією з найпотужніших e-commerce платформ, архітектура якої побудована за принципами високої модульності, розширюваності та суворого розподілу відповідальності. Magento 2 реалізує шаблон Model–View–ViewModel (MVVM) та активно використовує інверсію керування (IoC), dependency injection (DI), service contracts, а також декларативне налаштування через XML-файли [19].

2.2.1 Структура каталогів Magento

Коренева структура Magento 2 містить такі основні каталоги [20]:

- app/ – зберігає модулі, теми, локалізації та глобальні конфігурації системи. Саме тут розміщуються користувацькі модулі (app/code/VendorName/ModuleName).
- vendor/ – містить ядро Magento та сторонні бібліотеки, встановлені через Composer.
- pub/ – публічна частина проєкту, в якій знаходяться статичні ресурси (CSS, JS, зображення) та точка входу index.php.
- generated/ – результати автоматичної генерації Magento (інтерсептори, проксі, згенеровані класи).
- var/ – кеш, сесії, логи, звіти.
- bin/ – утиліти командного рядка, зокрема bin/magento.
- setup/ – скрипти ініціалізації та оновлення застосунку.

Така структура забезпечує розділення ядра та кастомізації, що робить систему гнучкою та масштабованою.

2.2.2 Модульність та принципи побудови модулів

У Magento кожен функціональний елемент оформлюється як ізольований модуль [21]. Цей підхід забезпечує масштабованість, безпечну інтеграцію та значно спрощує процеси оновлення платформи.

Стандартна структура модуля розташовується в директорії app/code/VendorName/ModuleName/ і складається з наступних ключових елементів [21]:

- Controller/ — містить класи, що відповідають за обробку HTTP-запитів користувачів та маршрутизацію.
- Model/ — інкапсулює бізнес-логіку та механізми взаємодії з базою даних.
- View/ — включає елементи представлення для frontend та adminhtml.
- Etc/ — директорія конфігураційних файлів, які декларативно описують функціональність модуля.
- Setup/ — містить скрипти для встановлення та оновлення структури бази даних модуля.
- registration.php — обов'язковий файл, що здійснює реєстрацію модуля в системі Magento.

2.2.3 XML-конфігурації та їх роль у системі

Magento широко використовує XML-файли для декларативного опису налаштувань [22]:

- module.xml – реєстрація модуля та залежностей.
- di.xml – dependency injection: заміна класів, визначення інтерфейсів, налаштування плагінів та інтерсепторів.
- events.xml – підписка на події ядра.
- routes.xml – оголошення маршрутів для frontend та adminhtml.
- acl.xml – визначення прав доступу в адмін-панелі.

- menu.xml – формування меню в адміністративному інтерфейсі.
- system.xml – сторінки конфігурації модуля в адмін-панелі.
- db_schema.xml – декларативне визначення таблиць БД.
- layout XML – формування сторінок і блоків у frontend та adminhtml.

2.2.4 Взаємодія компонентів у Magento 2

Архітектура Magento 2 передбачає тісну взаємодію між основними програмними компонентами, що забезпечує гнучкість та розширюваність системи без втручання в ядро. Основні принципи та механізми функціонування такі [19]:

- Service Contracts.
- Dependency Injection.
- Plugins (Interceptors).
- Observers.
- View Models і Blocks.
- Templating.

У комплексі ці механізми формують високорозширювану архітектуру, в якій майже будь-який компонент може бути модифікований або замінений кастомним модулем без зміни ядра. Це робить Magento 2 гнучкою платформою для створення та масштабування e-commerce рішень будь-якого рівня складності.

2.3 Проєктування структури модуля

Наступним етапом є проєктування структури модуля, а саме визначення основних компонентів, їхніх взаємозв’язків та логіки взаємодії з платформою Magento 2. Для формалізації архітектури модуля та опису логічної структури доцільно використовувати діаграми UML [23]. Це дозволяє показати архітектуру модуля у вигляді формальних моделей.

Розроблюваний модуль призначений для розширення функціональності системи Magento шляхом додавання нового сервісу або функції. Наприклад,

модуль може реалізовувати такі можливості, як “додавання користувацького атрибута”, “оптимізований механізм кешування”, “налаштування в адмін-панелі” тощо. На основі аналізу вимог було визначено основні компоненти, які мають бути реалізовані:

- моделі даних;
- репозиторії;
- контролери;
- блоки відображення;
- XML-конфігурації;
- плагіни або observers;
- допоміжні сервіси.

Взаємодія цих елементів формує загальну архітектуру модуля та забезпечує його інтеграцію з іншими компонентами системи.

Щоб чітко і логічно описати структуру, застосовуються діаграми UML, які відображають залежності між класами, обробку даних та виконання бізнес-логіки. Основу складають такі типи діаграм:

- Діаграма класів
- Діаграма послідовності
- Діаграма компонентів
- Діаграма діяльності

Існують й інші типи діаграм, однак у межах цієї роботи використовуються саме ці чотири. Спочатку визначити значення діаграма:

Діаграма класів – це статичний елемент UML, що показує структуру моделі системи через класи, їхні атрибути, методи та зв'язки між ними. Вона візуалізує статичні аспекти системи, відображаючи взаємозв'язки між класами та їхню внутрішню будову, включаючи типи відносин між ними, але не показує динамічні процеси, такі як виклик методів [24].

Діаграма послідовності показує, як процеси або об'єкти взаємодіють під час виконання сценарію. На діаграмі відображаються класи, необхідні для виконання сценарію, і повідомлення, які вони передають один одному [25].

Діаграма компонентів – це статична структурна діаграма, що показує фізичну структуру програмної системи, розбиваючи її на окремі компоненти та демонструючи залежності між ними [26].

Діаграма діяльності – це UML-діаграма, що моделює процеси, алгоритми та робочі потоки, показуючи послідовність дій, гілкування, умови та [27] паралельність.

Наступний етап описує перший сценарій — “Базова архітектура модуля Magento”. Через діаграму класів показано базову архітектуру розробленого модуля та основні взаємозв’язки між його компонентами.

Модуль складається з наступних ключових класів:

- Controller\Adminhtml\Index — контролер, який приймає запити користувача та відповідає за маршрутизацію.
- Model\Main — головний клас модуля, що інкапсулює бізнес-логіку.
- Model\ResourceModel\Main — відповідає за роботу з базою даних, CRUD-операції.
- Model\ResourceModel\Main\Collection — реалізує вибірку колекцій даних із бази.
- Api\MainRepositoryInterface — інтерфейс сервісного контракту для роботи з даними модуля.
- Model\MainRepository — реалізація інтерфейсу, забезпечує доступ до моделі через DI.

Зв’язки між класами демонструють ін’єкцію залежностей, композицію та реалізацію інтерфейсів. Контролер використовує репозиторій для доступу до даних, який у свою чергу взаємодіє з моделлю та ресурсною моделлю. Така базова архітектура дає можливість додавати нові компоненти або змінювати без ризики шкоді модули для Magento (на рис. 2.3.1).

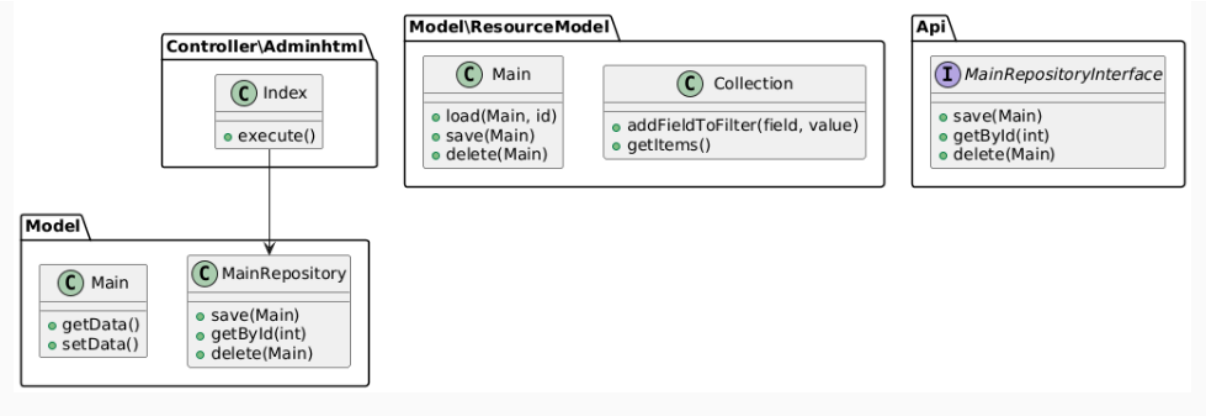


Рисунок 2.3.1 – Базова архітектура модуля Magento

Наступне сценарій – “Покращення модуля через кешування”. Він покажує як модуль обробляє запит користувача з використанням кешування для підвищення продуктивності. Послідовність дій:

1) Користувач відкриває сторінку продукту.
2) Контролер перевіряє Cache Service на наявність готових даних.
3) Якщо кеш існує: дані повертаються одразу → View формує сторінку → користувач отримує відповідь.
4) Якщо кешу немає: контролер звертається до Repository.
5) Repository передає дані в Model для виконання бізнес-логіки.
6) Після обробки інформація зберігається в Cache Service.
7) Контролер передає дані у View, і користувач отримує згенеровану сторінку.

Ця схема демонструє, що використання кешування зменшує навантаження на базу даних та прискорює роботу модуля (рис. 2.3.2).

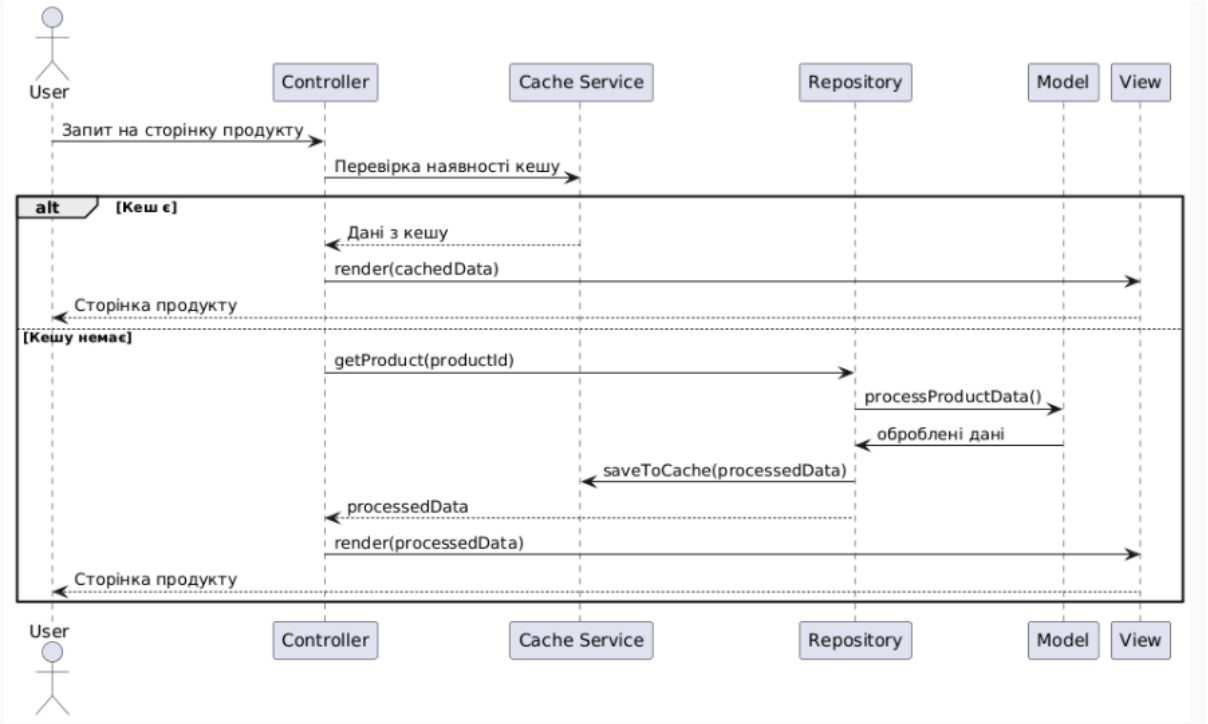


Рисунок 2.3.2 – Покращення модуля через кешування

Наступний сценарій — “Архітектура модуля з розширеною функціональністю”, де розглядається взаємодія компонентів користувацького модуля, який інтегрується з базовими підсистемами Magento.

1) Користувач відкриває сторінку, і Magento через Routing System визначає потрібний контроллер модуля.
2) Controller приймає запит, перевіряє кеш через Cache Service і звертається до Repository, щоб отримати дані.
3) Repository викликає Model, яка виконує бізнес-логіку, може зробити запит до External API Service або надіслати подію через Observer.
4) Якщо потрібні дані з бази, Model працює через ResourceModel, який виконує операції з БД через Database Adapter ядра Magento.
5) Observer реагує на події, використовуючи Event Manager, що дозволяє розширити роботу модуля без зміни основного коду.
6) Файли module.xml, di.xml, routes.xml забезпечують підключення модуля, маршрутизацію та залежності через DI Container.

7) Після обробки всі дані повертаються назад у Controller, який передає їх у View, а той генерує HTML-вивід для користувача.

Схема діаграма показано на рисунок 2.3.3:

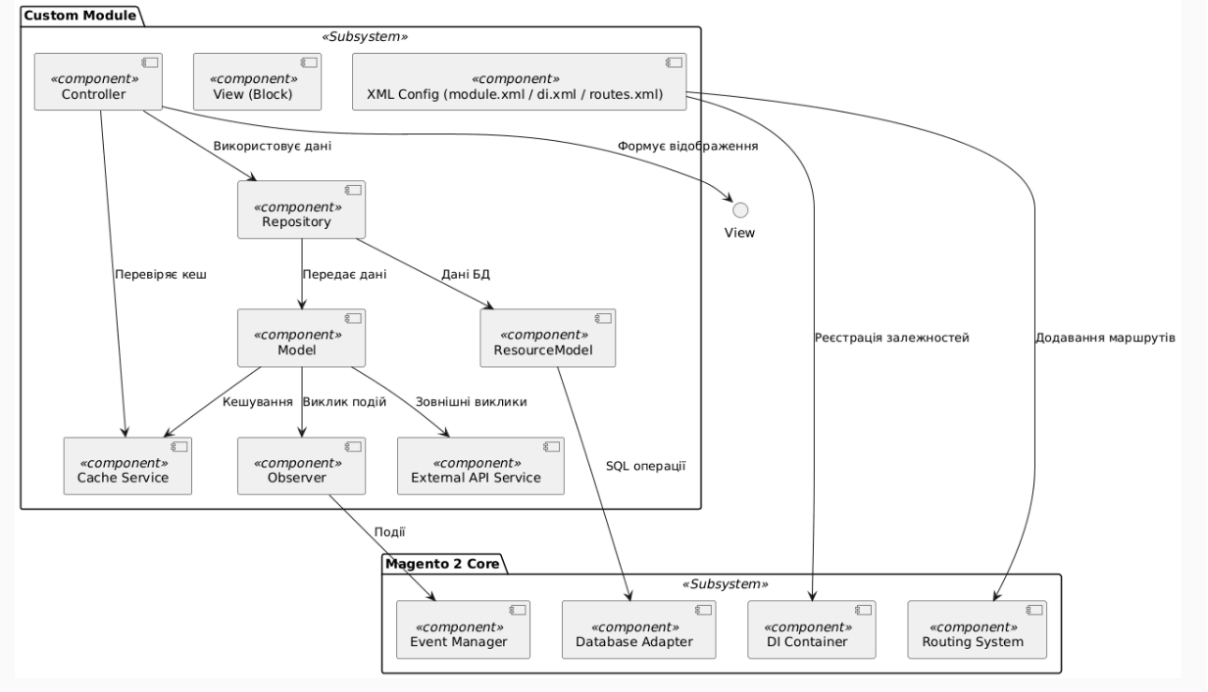


Рисунок 2.3.3 – Архітектура модуля з розширеною функціональністю

Останній сценарій — “Оновлення даних у модулі”. Тут описано етапи оновлення даних у модулі. Спочатку адміністратор відкриває сторінку редагування, вводить нову інформацію та натискає «Зберегти». Система приймає запит і перевіряє правильність введених даних. Якщо виявлено помилки — адміністратор отримує відповідне повідомлення.

Якщо дані валідні, система оновлює записи в базі даних, очищає кеш модуля та генерує подію, яку можуть обробляти інші частини системи. Після успішного оновлення адміністратор бачить повідомлення про завершення операції (рис. 2.3.4).

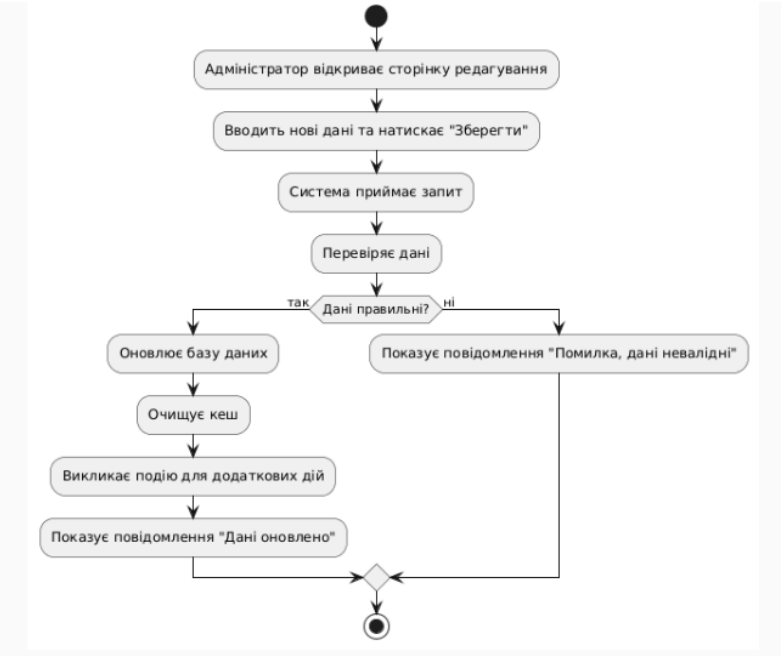


Рисунок 2.3.4 – Оновлення даних у модулі

Отже, наведені сценарії всебічно описують ключові процеси взаємодії користувача та системи в межах розробленого модуля Magento. Кожен сценарій показує логіку виконання окремих функцій — від перегляду даних до їх оновлення та інтегрованої взаємодії системи. У сукупності вони формують цілісне уявлення про роботу модуля та слугують основою для побудови UML-діаграм.

2.4 Проєктування бази даних

У системі Magento 2 база даних є ключовим елементом архітектури, яка зберігає всі дані про товари, категорії, клієнтів, замовлення, налаштування системи та інші бізнес-об’єкти. Для роботи використовується реляційна СУБД MySQL/MariaDB, яка підтримує ACID-властивості, транзакції та складні зв’язки між таблицями.

У системі база даних є ключовим елементом, оскільки забезпечує збереження всієї інформації: даних про товари, категорії, замовлення, користувачів, налаштування та інші параметри. Magento підтримує два підходи до організації бази даних, вертикальна та горизонтальне структура [28].

На початку було обрано вертикальне структура бази даних, оскільки сам проєкт не потребує підтримки у багатьох інтернет-магазин, прості та швидке видалення даних за певною ознакою, що не блокує інші частини таблиці та простота налаштування та використання. Таким чином, вибір вертикальної архітектури бази даних це хороші вибір для даному проекті.

На майбутнє можливо потрібно перейди до горизонтальної структури бази даних. Якщо потрібно розширювати функціональність або обробляти значно більші обсяги інформації для проекти. Горизонтальна структура бази даних (БД) передбачає розділення великої таблиці на менші частини (партиції) за певним критерієм, наприклад, за часом або певним значенням. Це робиться для покращення продуктивності, спрощення управління та оптимізації запитів.

Для ілюстрації взаємодії модуля з базою даних системи наведено структуру двох основних таблиць:

- magefan_faq — таблиця для збереження питань/відповідей.
    - id – первинний ключ запису.
    - question – текст питання.
    - answer – текст відповіді.
    - product_sku – зв’язок із товаром (зовнішній ключ).
    - status – статус публікації (активний/неактивний).
- catalog_category_entity — таблиця категорій товарів.
    - entity_id – первинний ключ категорії.
    - parent_id – ідентифікатор батьківської категорії.
    - name – назва категорії.

На основі структури таблиць catalog_product_entity та magefan_faq була побудована діаграма бази даних (на рис. 3.3.1) [29].

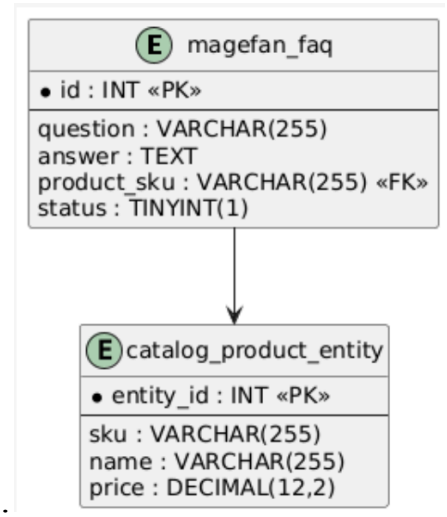


Рисунок 2.4.1 – Діаграма зв’язків таблиць бази даних Magento

2.5 Реалізація модуля: структура файлів, основні компоненти

Для того щоб створити власний модуль Magento потрібно реалізувати основу модуля, визначити основу структура файлів які відповідає до стандартів фреймворку. Кожний елемент архітектури виконує окрему функцію та відповідає певному рівню логіки системи: робота з даними, бізнес-логіка, відображення інформації або взаємодія з користувачем. Щоб зручно створити основу модулів для платформи Magento можна використовувати спеціальні сайт mage2gen – це генератор модулів для платформи Magento [30]. Ця сервіс дозволяє швидко створювати базовий код для нових модулів через зручний інтерфейс, що включає контролери, моделі, блоки, шаблони, плагіни, спостерігачі та інші елементи. Приклад згенерований модулі покажано на рисунок 2.5.1 – 2.5.2.

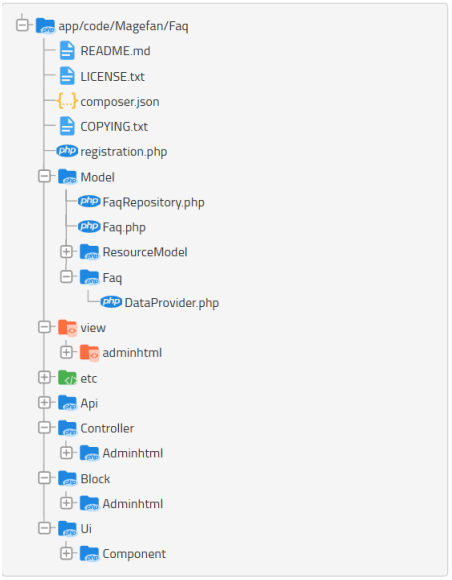


Рисунок 2.5.1 – Базова структура модуля Magento 2 (1)

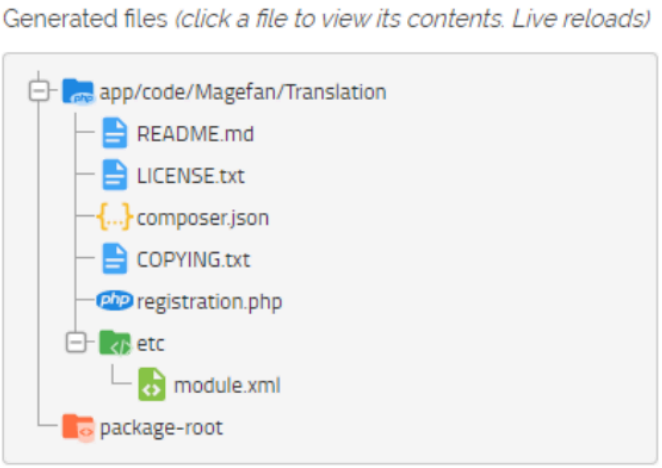


Рисунок 2.5.2 – Базова структура модуля Magento 2 (2)

2.5.1 Структура файлів модуля

Спочатку потрібно створити основні структура, а перед тим потрібно визначити де буде знаходиться, наприклад одні із модуль розміщується у каталозі: …/app/code/Magefan/Faq там у Magefan це <Vendor> а <ModuleName> буде Faq.

Наступне описує усіма основні структура файлів модуля:

- etc/ – конфігураційні файли модуля:

- module.xml – реєстрація модуля, залежності;
- di.xml – налаштування залежностей і впровадження через Dependency Injection;
- routes.xml – визначення маршрутів для контролерів;

- Controller/ – контролери, які обробляють HTTP-запити:
  - Index/Index.php – приклад контролера для сторінки за маршрутом /module/index/index.
- Model/ – бізнес-логіка:
  - ResourceModel/ – робота з базою даних;
  - Repository – реалізація доступу до даних через інтерфейси.
- Api/ – інтерфейси для доступу до даних та репозиторії:
  - EntityRepositoryInterface.php – інтерфейс репозиторію;
  - Data/EntityInterface.php – інтерфейс даних сутності.
- Model/ResourceModel/ – робота з базою даних:
  - EntityResource.php – ресурсна модель для CRUD операцій;
  - EntityResource/Collection.php – колекція об’єктів моделі.
- Block/ або ViewModel/ – логіка представлення для шаблонів:
  - EntityView.php – передача даних у шаблон.
- view/frontend/ – файли фронтенду:
  - templates/view.phtml – шаблон відображення;
  - layout/module_index_index.xml – layout сторінки для контролера.
- registration.php – реєстрація модуля у системі Magento.

Це основні структура но можна або необхідно добавити ще декілька файл для робіт даному модуля.

2.5.2 Основні конфігураційні файли

Конфігурація модуля Magento здійснюється за допомогою XML-файлів, які дозволяють інтегрувати модуль у систему без зміни ядра платформи. Основні конфігураційні файли модуля наведено нижче:

- module.xml — використовується для реєстрації модуля в системі та визначення його залежностей.

Лістинг 2.5.1 — module.xml.

```
<?xml version="1.0" ?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd"
>
      <module name="Magefan_Faq" setup_version="1.0.0"/>
</config>
```

- di.xml — налаштовує механізм Dependency Injection, визначає реалізації інтерфейсів і плагіни.

Лістинг 2.5.2 — di.xml.

```
<type
name="Magento\Framework\View\Element\UiComponent\DataProvider\CollectionFac
tory">
        <arguments>
            <argument name="collections" xsi:type="array">
                <item name="magefan_faq_listing_data_source"
xsi:type="string">MagefanFaqModelResourceModelFaqGridCollection</item>
            </argument>
        </arguments>
</type>
```

- routes.xml — визначає маршрути для обробки HTTP-запитів контролерами.

Лістинг 2.5.3 — routes.xml.

```
<?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="standard">
        <route id="faq" frontName="faq">
            <module name="Magefan_Faq"/>
        </route>
    </router>
</config>
```

2.5.3 Контролери

Наступне основні модуль контролер. Контролер відповідає за обробку HTTP-запиту та підготовку результату. А також він спеціалізований компонент

системи, що призначений для керування зовнішніми пристроями комп'ютера: накопичувачами, відеосистемою та дисплеєм, принтерами тощо [31].

Лістинг 2.5.4 – Controller

```
abstract class Faq extends \Magento\Backend\App\Action
{
    const ADMIN_RESOURCE = 'Magefan_Faq::top_level';
    protected $_coreRegistry;
    public function __construct(
        \Magento\Backend\App\Action\Context $context,
        \Magento\Framework\Registry $coreRegistry
    ) {
        $this->_coreRegistry = $coreRegistry;

          parent::__construct($context);
      }
      public function initPage($resultPage)
      {
          $resultPage->setActiveMenu(self::ADMIN_RESOURCE)
              ->addBreadcrumb(__('Magefan'), __('Magefan'))
              ->addBreadcrumb(__('Faq'), __('Faq'));
          return $resultPage;
      }
  }
```

2.5.4 Моделі та робота з даними

Наступне етап це модель, він реалізує бізнес-логіку. Він поділяється на дві частини:

- ResourceModel/ – робота з базою даних;
- Repository – реалізація доступу до даних через інтерфейси.

Лістинг 2.5.5 – ResourceModel

```
use Magento\Framework\Model\ResourceModel\Db\AbstractDb;
class Faq extends AbstractDb
{
    protected function _construct()
    {
        $this->_init('magefan_faq', 'faq_id');
    }
}
```

Лістинг 2.5.6 – Repository

```
class FaqRepository implements FaqRepositoryInterface
{
 …

 public function __construct(
        ResourceFaq $resource,
        FaqFactory $faqFactory,
        CollectionFactory $collectionFactory,
        CollectionProcessorInterface $collectionProcessor,
        SearchResultsInterfaceFactory $searchResultsFactory
    ) {
        $this->resource = $resource;
        $this->faqFactory = $faqFactory;
        $this->collectionFactory = $collectionFactory;
        $this->collectionProcessor = $collectionProcessor;
        $this->searchResultsFactory = $searchResultsFactory;
    }

…
}
```

2.5.5 Компоненти представлення: Block та ViewModel

У модулі використовуються два типи компонентів для передачі даних у шаблони: Block і ViewModel.

Block — це стандартний елемент Magento, який відповідає за підготовку даних для .phtml-шаблонів та містить просту логіку відображення. Він підключається через layout-файли та викликається під час рендерингу сторінки.

ViewModel — сучасніший підхід, який дозволяє винести логіку з шаблону та зробити код більш чистим і зручним для розширення. ViewModel підключається через di.xml і просто повертає дані, які потрібні на сторінці.

Через дві компоненти дає можливість зробити відображення даних більш зручний і якісне.

Лістинг 2.5.7 – Block

```
class Faq extends Template
{
    public function getFaqText()
    {
        return 'Це сторінка Частих Питань від Magefan';
    }
}
```

Лістинг 2.5.8 – ViewModel

```
class Faq implements ArgumentInterface
{
    protected $collectionFactory;

    public function __construct(CollectionFactory $collectionFactory)
    {
        $this->collectionFactory = $collectionFactory;
    }

    public function getActiveFaqs()
    {
        $collection = $this->collectionFactory->create();
        $collection->addFieldToFilter('status', ['eq' => 1]);
        return $collection;
    }
}
```

2.5.6 Представлення (View / Templates)

У Magento 2 є представлення формується за допомогою шаблонів .phtml, які відповідають за відображення інформації на сторінці. Шаблони отримують дані від Block або ViewModel і виводять їх у вигляді HTML-коду. Сам файл знаходиться в …/view/frontend/templates/. Основна мета шаблонів — показати користувачу дані у зручній формі, не порушуючи розподілення відповідальності в архітектурі Magento.

Лістинг 2.5.9 – view/frontend/templates/faq.phtml

```
<h1>Список активних питань</h1>

<?php if ($faq): ?>
    <div class="faq-item">
        <h2><?= $block->escapeHtml($faq->getQuestion()) ?></h2>
        <p><?= $block->escapeHtml($faq->getAnswer()) ?></p>
    </div>
<?php else: ?>
    <p>Питання не знайдено.</p>
<?php endif; ?>
```

2.5.7 Логіка роботи модуля

Сам модуль побудований на прості схема Magento 2 і описує шлях, як проходять дані по модуля. Коли користувач відкриває сторінку модуля, система звертається до контролера. Контролер приймає запит, виконує основні перевірки та викликає репозиторій для отримання або збереження даних.

Репозиторій працює разом із ResourceModel, яка виконує звернення до бази даних. Отримані дані передаються в модель, де виконується бізнес-логіка модуля: обробка, перевірка, фільтрація та підготовка до відображення. За потреби модуль може використовувати кеш, щоб прискорити роботу проєкту, або викликати Observer, якщо потрібно виконати певні дії після збереження чи оновлення даних.

Далі підготовлені дані передаються у Block або ViewModel, які формують набір змінних для шаблону. Шаблон (.phtml) відповідає за фінальне відображення інформації на сторінці.

Також модуль можна добавити нові або покращувати файлами для виконання необхідних умов або додаткової логіки. Це робить модуль гнучким і дозволяє легко додавати новий функціонал.

2.6 Оптимізація модуля (кешування, індексація, SQL)

Оптимізація модуля у Magento 2 допомагає зробити роботу сайту швидшою і зменшити навантаження на сервер. Основна ідея полягає в тому, щоб менше звертатися до бази даних, кешувати результати обчислень і ефективніше працювати з даними [32].

Наприклад, якщо модуль отримує список товарів для відображення, можна використати кешування, щоб не робити SQL-запит при кожному завантаженні сторінки.

Лістинг 2.6.1 – приклад оптимізація БД:

```
$cacheKey = 'custom_product_list';
$cachedData = $this->cache->load($cacheKey);

if (!$cachedData) {
    // Отримання даних з бази
    $collection = $this->productCollectionFactory->create()
        ->addAttributeToSelect(['name', 'price'])
        ->addFieldToFilter('status', 1)
        ->setPageSize(10)
        ->setCurPage(1);

    $data = $collection->getData();
    $this->cache->save(serialize($data), $cacheKey,
['custom_module_cache'], 3600);
} else {
    $data = unserialize($cachedData);
}
```

У цьому прикладі дані з бази зберігаються у кеші на 1 годину. Це дозволяє значно зменшити кількість запитів до бази і прискорити роботу сторінки.

Також у Magento застосовують індексацію, щоб швидко оновлювати дані каталогу та цін, профілювання для виявлення повільних запитів, а також можна використовувати метод Page Cache та Varnish, щоб кешувати повні сторінки сайту.

Таким чином, оптимізація поєднує кешування, оптимізацію SQL-запитів і профілювання, це дає можливість прискорити роботу сайту та покращити користування сайту.

2.7 Інтеграція модуля у Magento

Після створення та розробки модуля потрібно правильно інтегрувати його в систему Magento 2, щоб він коректно працював із іншими компонентами платформи. Інтеграція включає підключення модуля до конфігурацій Magento, налаштування маршрутизації, створення контролерів та блоків, а також підключення до бази даних, якщо модуль працює з власними таблицями.

Спочатку потрібно реєструвати модуля а для того потрібно створити файл “registration.php”.

Лістинг 2.7.1 – registration.php

```
use Magento\Framework\Component\ComponentRegistrar;
ComponentRegistrar::register(ComponentRegistrar::MODULE, 'Magefan_Faq',
__DIR__);
```

Далі налаштовується конфігурація модуля через файл etc/module.xml, який містить основну інформацію про модуль та його залежності.

Лістинг 2.7.2 – module.xml

```
use Magento\Framew<config
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd"
>
      <module name="Magefan_Faq" setup_version="1.0.0"/>
</config>
```

Після цього створюються маршрути та контролери, щоб модуль міг обробляти запити користувачів. Якщо модуль працює з базою даних, його моделі та ResourceModel підключаються через di.xml.

І нарешті необхідно активутися модуля через командний рядок:

- php8.2 bin/magento module:enable Magefan_Faq
- php8.2 bin/magento setup:upgrade
- php8.2 bin/magento setup:di:compile
- php8.2 bin/magento cache:flush

Після активації модуль готовий до використання. Якщо будуть додані нові функції або код, потрібно повторно виконати відповідні команди CLI, крім першого активації. Звертаю увагу, що Magento працюватиме на останній версії PHP, тому важливо використовувати php8.2 bin/magento, а не просто php bin/magento, інакше команди не запустяться.

Таким чином, інтеграція модуля у Magento включає реєстрацію, конфігурацію, створення маршрутизації та контролерів, підключення моделей і ресурсів, а також активацію через CLI. Після цього модуль готовий до використання та подальшої оптимізації.

2.8 Висновки до другого розділу

У цьому розділі було проведено повний аналіз і проєктування модуля для Magento 2. Спочатку було обґрунтовано вибір технологій та інструментів для створення проєкту, визначено та побудовано UML-діаграми, описано сценарії роботи модуля, спроєктовано базу даних, а також визначено механізми взаємодії між контролерами, моделями, репозиторіями, ресурсними моделями та відображенням. Наведені сценарії роботи підтверджують коректність розробленого рішення та демонструють послідовність обробки даних у модулі. Крім того, було описано методи оптимізації та інтеграції модуля. Підсумком є те, що всі виконані етапи дозволили створити чітко структурований, масштабований і

розширюваний модуль, який відповідає вимогам проєкту та може бути доповнений новим функціоналом у майбутньому.

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ГОТОВИХ МОДУЛІВ

Розділ присвячений практичній реалізації розроблених програмних модулів для платформи Magento 2 та аналізу результатів їх роботи. Описується функціональне призначення кожного модуля, особливості їх інтеграції у вебсайт і взаємодії з іншими компонентами системи. Значну увагу приділено тестуванню розроблених рішень, оцінці стабільності та продуктивності, а також порівнянню показників роботи системи до і після впровадження оптимізованих модулів.

3.1 Опис першого модуля Magefan_Frankenstein

Першим розробленим модулем є Magefan_Frankenstein, призначений для розширення функціональності Magento 2, зокрема механізмів керування доступом та відображенням цін.

Початкова структура модуля була створена вручну в каталозі app/code/Magefan/Frankenstein.

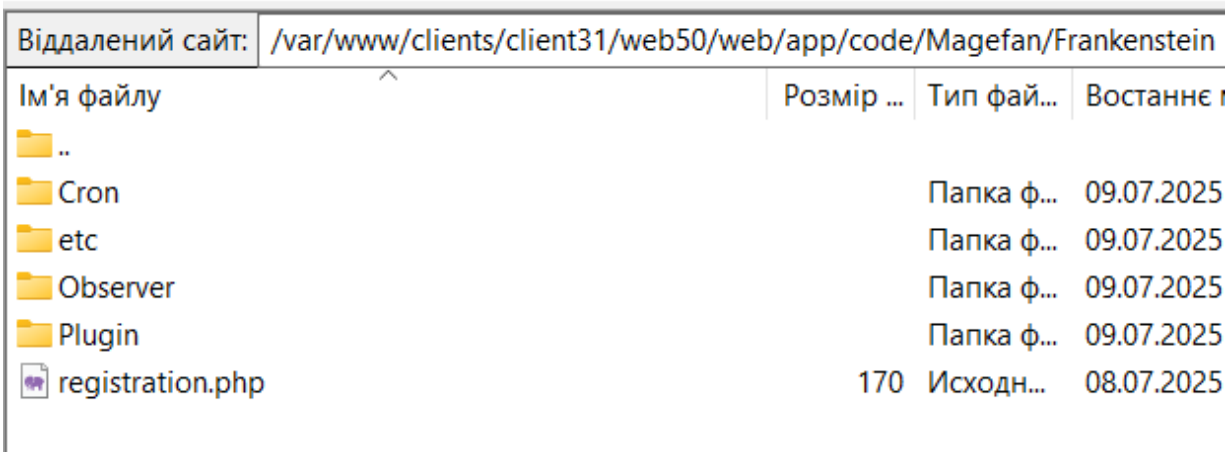


Рисунок 3.1.1 – Базовая структура модуля Frankenstein для Magento.

Після створення базових файлів виконувались стандартні команди встановлення:

- php bin/magento setup:upgrade

- php bin/magento setup:di:compile
- php bin/magento cache:flush

Ці команди забезпечили коректну реєстрацію модуля в системі та підключення його до внутрішніх механізмів Magento 2. У межах модуля визначено три основні зони роботи (areas):

- frontend — відповідає за відображення даних на вітрині;
- adminhtml — забезпечує роботу в адміністративній панелі;
- base — спільна зона для CLI, API, cron-процесів та індексації.

Для реалізації функціональності модуль містить два плагіни:

1) Приховування цін для всіх користувачів, окрім групи «wholesale».
2) Заокруглення цін для відображення лише цілих чисел (без копійок).

Перший плагін перехоплює отримання значення ціни та повертає “null”, якщо користувач не належить до визначеної групи.

Лістинг 3.1 – Фрагмент коду для приховування цін.

```
<?php
namespace Magefan\Frankenstein\Plugin;
use Magento\Customer\Model\Session as CustomerSession;
class HidePricePlugin
{
    protected $customerSession;
    public function __construct(CustomerSession $customerSession)
    {
        $this->customerSession = $customerSession;
    }
    public function aroundGetValue($subject, \Closure $proceed)
    {
        // Отримуємо групу користувача
        $groupId = $this->customerSession->getCustomerGroupId();
        $wholesaleGroupId = 3;
        if ($groupId != $wholesaleGroupId) {
            return null;
        }
        return $proceed();
    }
}
```

Другий плагін виконується після виклик метод форматування, заокруглює суму до найближчого цілого числа.

Лістинг 3.2 – Фрагмент коду для заокруглення цін:

```
<?php
namespace Magefan\Frankenstein\Plugin;

class RoundPricePlugin
{
    // After плагін на format — заокруглюємо вивід ціни
    public function afterFormat(
        $subject,
        $result,
        $amount,
        $includeContainer = true,


            $precision = 2,
            $scope = null,
            $currency = null
        ) {
            // Заокруглюємо ціну до цілих
            $roundedAmount = round($amount);

            return $subject->format($roundedAmount, $includeContainer, 0,
$scope, $currency);
        }
    }
```

У результаті ціни відображаються лише для користувачів групи wholesale, а всі числові значення на вітрині мають цілі значення без копійок.

У модулі також реалізовано обсервер Observer/CheckLoginObserver.php, який перевіряє авторизацію користувача та обмежує доступ до сторінок магазину для неавторизованих відвідувачів.

Дозволений доступ мають лише такі маршрути:

- customer/*
- contact/*
- cms/*

Лістинг 3.3 – Observer/CheckLoginObserver.php

```
<?php
namespace Magefan\Frankenstein\Observer;

use Magento\Framework\Event\Observer;
use Magento\Framework\Event\ObserverInterface;
use Magento\Framework\App\Response\RedirectInterface;
use Magento\Framework\App\ActionFlag;
use Magento\Customer\Model\Session;

class CheckLoginObserver implements ObserverInterface
{
    protected $customerSession;
```

Продовження лістинг 3.3

```
    protected $redirect;
    protected $actionFlag;

    public function __construct(
        Session $customerSession,
        RedirectInterface $redirect,
        ActionFlag $actionFlag
    ) {
        $this->customerSession = $customerSession;
        $this->redirect = $redirect;
        $this->actionFlag = $actionFlag;
    }

    public function execute(Observer $observer)
    {
        $controller = $observer->getControllerAction();
        $request = $controller->getRequest();
        $path = $request->getPathInfo();

        // Якщо користувач не залогінений
        if (!$this->customerSession->isLoggedIn()) {
            // дозволяємо тільки customer, contact, cms
              if (!preg_match('#^/(customer|contact|cms)#', $path) || $path ===
'/') {
                                                    $this->actionFlag->set('',
\Magento\Framework\App\ActionInterface::FLAG_NO_DISPATCH, true);
                            $this->redirect->redirect($controller->getResponse(),
'customer/account/login');
            }
        }
    }
}
```

На завершальному етапі додано власну cron-задачу, яка щоденно о 02:00 (крім суботи та неділі) автоматично збільшує ціну кожного товару на 1$.

Ця функціональність демонструє інтеграцію модуля з планувальником Magento та роботу з моделями і репозиторіями продуктів.

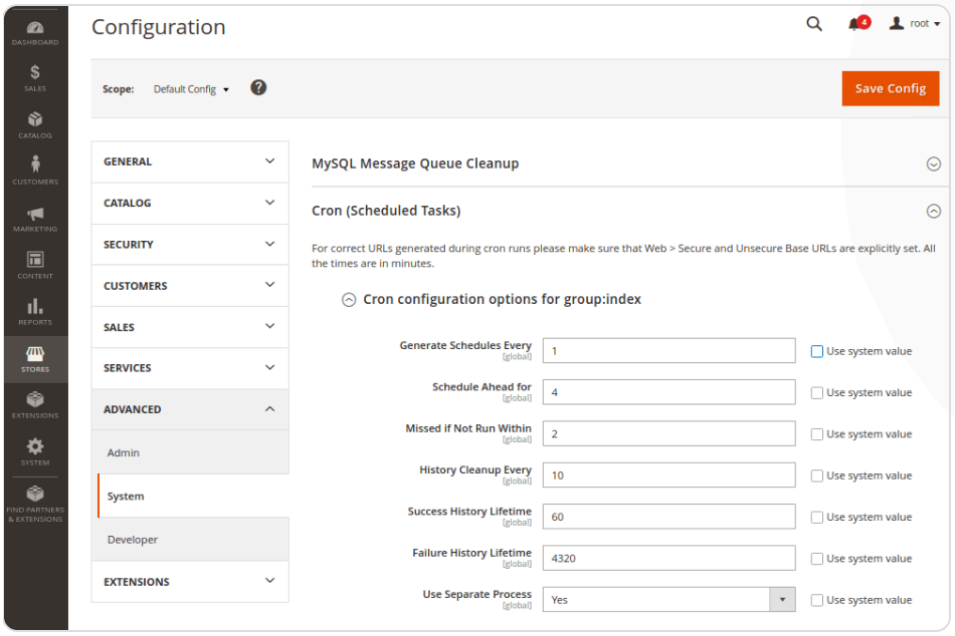


Рисунок 3.1.1 – крон-задачу

3.2 Опис другого модуль Magefan_Faq.

Для того, щоб створити базової структури модуля Magefan_Faq було використано сервіс Mage2Gen, який автоматизує генерацію файлів для реєстрації модуля в системі.

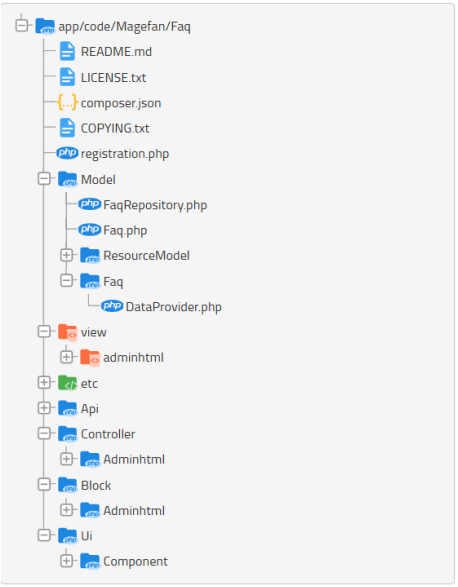
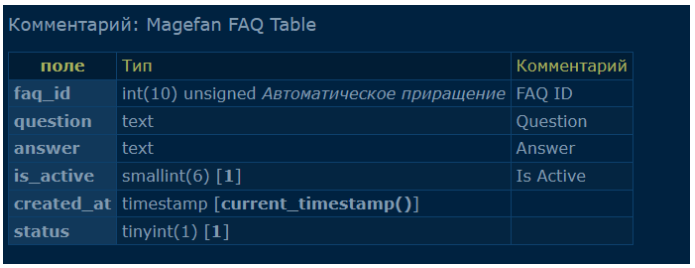


Рисунок 3.2.1 – Базовая структура модуля Magefan_Faq

Наступним кроком було створення таблицю magefan_faq, що зберігає запитання та відповіді для виведення у вітрині магазину.



Комментарий: Magefan FAQ Table

поле	Тип	Комментарий
faq_id	int(10) unsigned *Автоматическое приращение*	FAQ ID
question	text	Question
answer	text	Answer
is_active	smallint(6) [1]	Is Active
created_at	timestamp [current_timestamp()]	
status	tinyint(1) [1]	

Рисунок 3.2.2 – Таблиця БД Magefan Faq

Далі реалізує модуля включає декілька етапів:

1) Створення контролера, блоку, шаблону та layout-файлу для сторінки зі списку FAQ;
2) Створити та перенесення із бізнес-логіку від блоку на View Model;
3) Створити модель, ресурсної моделі, контролерів для роботи з таблицею magefan_faq;
4) Формування колекції та виведення активних записів на сторінці /faq;
5) Завантаження даних через ресурс модель (метод load()).

Після цього створено репозиторій моделі FAQ, що забезпечує роботу через єдиний інтерфейс: “$faq = $this->faqRepository->getById($id);”. У файлі etc/adminhtml/menu.xml додано пункт меню:

Лістинг 3.4 – Фрагмент коду додавання пункту меню

```
<menu>
      <add     id="Magefan_Faq::faq"     title="FAQ"     module="Magefan_Faq"
sortOrder="100" parent="Magento_Backend::content"/>
</menu>
```

Також у etc/acl.xml визначено права доступу, що регулюють роботу адміністратора в розділі Content → FAQ (див. рис. 3.2.3).

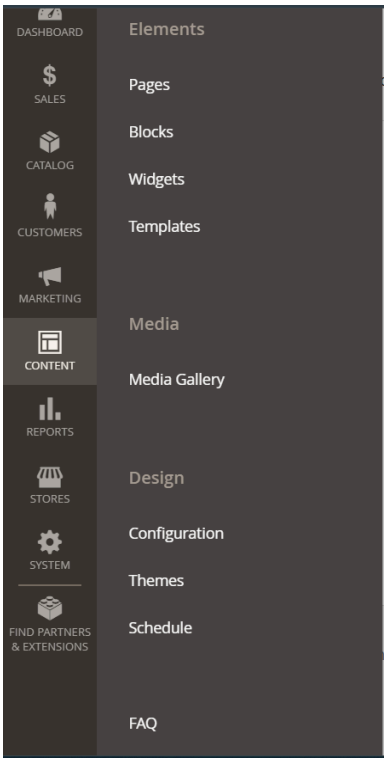


Рисунок 3.2.3 – Меню адмін панелі

Контролери підготовлено для реалізації операцій CRUD-операцій (Create, Read, Update, Delete) у панелі адміністратор.

Для управління даними Magento 2 надає стандартний компонент – UI Grid, який забезпечує

- створення нових записів;
- редагування існуючих даних;
- видалення записів;
- сортування за стовпцями;
- пошук за ключовими словами;
- фільтрація;
- імпорт дани.

Таким чином, UI Grid виступає основним інструментом роботи адміністратора з FAQ (рис. 3.3.4).

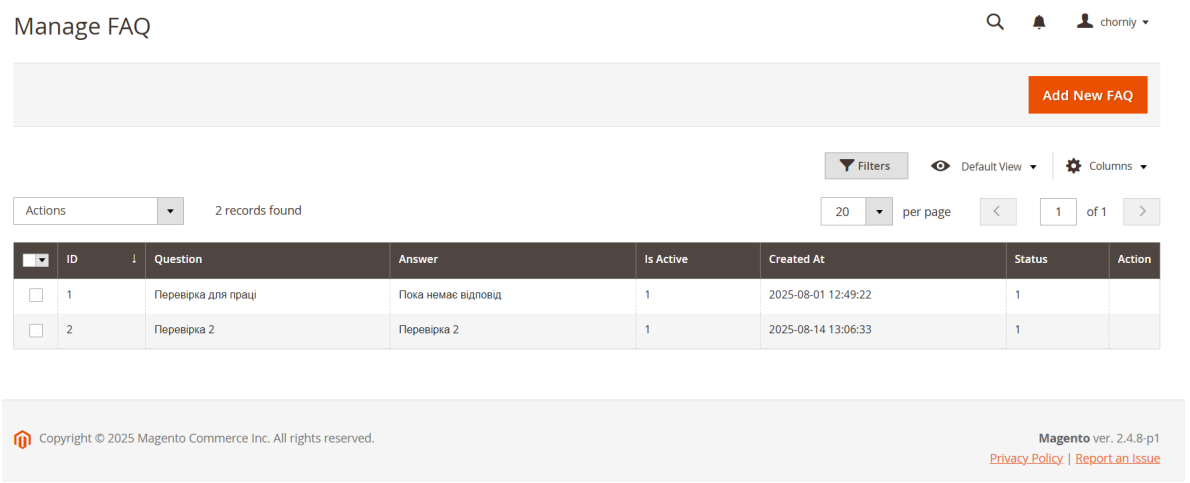


Рисунок 3.3.4 – модулі Magefan_Faq

Для створення та редагування записів Magefan_Faq застосовується UI Form, яка інтегрується з UI Grid та бекенд-моделлю, забезпечуючи:

- створення нових записів;
- редагування існуючих;
- валідацію введених даних;
- збереження або видалення.

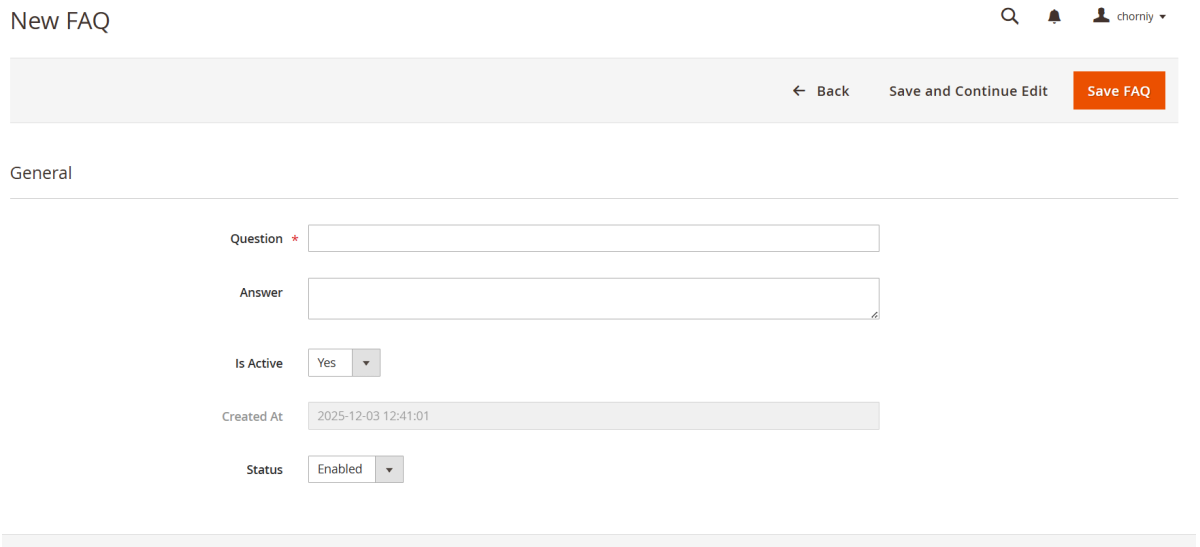


Рисунок 3.2.5 – інтерфейс створення нові дані Magefan_Faq

На завершальному етапі реалізовано власну сторінку налаштувань через механізм System Configuration (файл system.xml).

У меню Stores → Configuration → Magefan → FAQ розміщено перемикач, що дозволяє ввімкнути або вимкнути модуль без редагування коду (див. рис. 3.2.6).

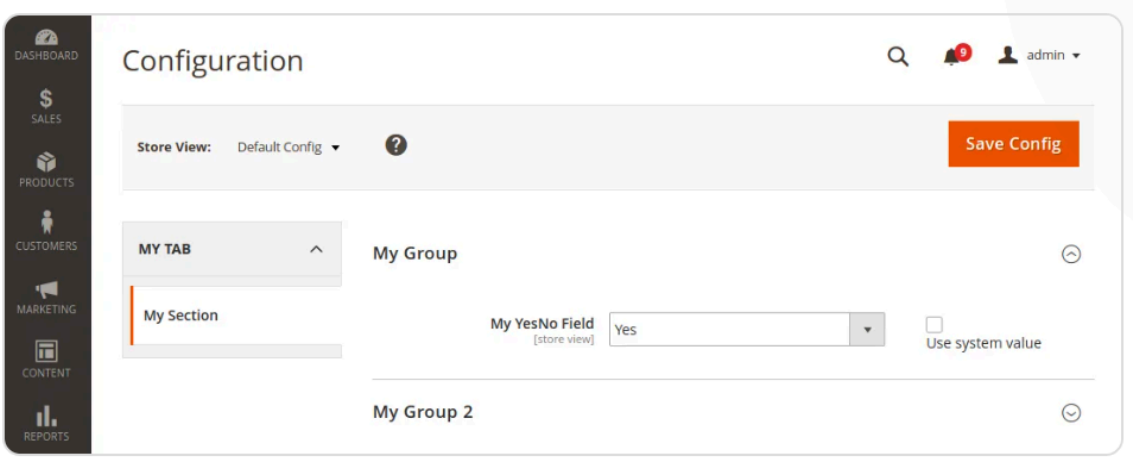


Рисунок 3.2.6 – налаштування модуля Magefan_Faq

3.3 Опис третий модуль Magefan_PromoBanners

Спочатку було створено базову структуру модуля Magefan_PromoBanners. Для цього, як і в попередніх модулях, було використано сервіс Mage2Gen, який дозволяє зручно та швидко згенерувати основні файли модуля для його реєстрації в системі Magento 2.

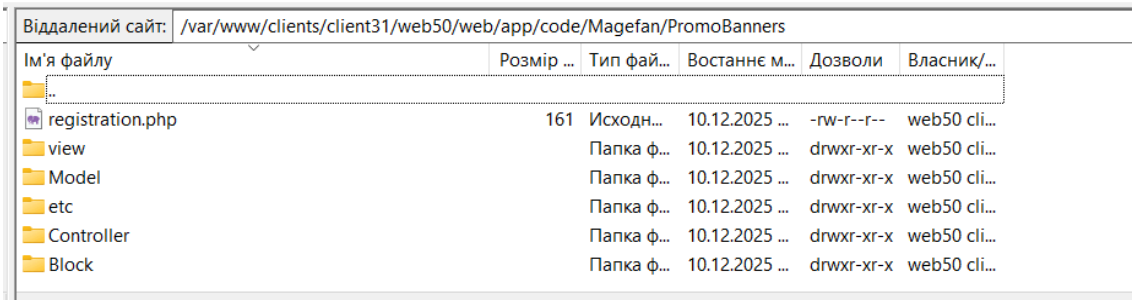


Рисунок 3.3.1 – Базовая структура модуля PromoBanners для Magento.

Разом із основною структурою модуля було передбачено створення таблиці бази даних magefan_promobanners, у якій зберігаються всі необхідні дані для роботи з банерами. У цій таблиці зберігаються основні параметри банера, зокрема назва, статус активності, текстовий вміст, а також службові поля, необхідні для коректної роботи модуля.

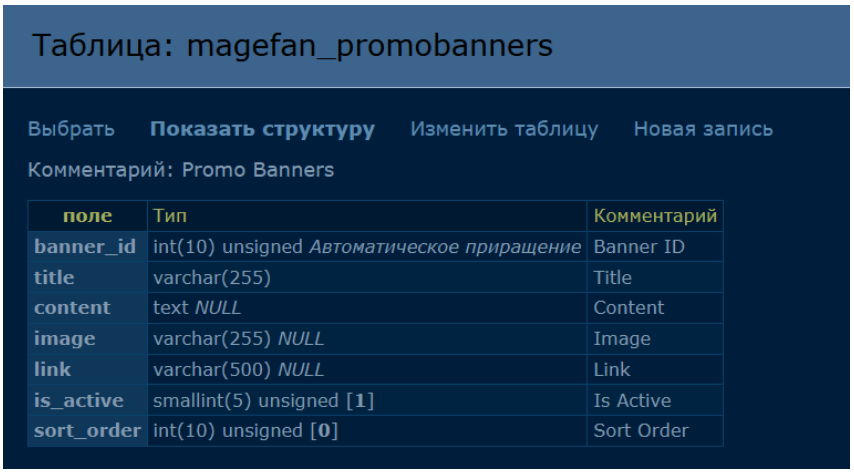


Рисунок 3.3.2 – Таблиця бази даних модуля банерів

Для реалізації модуля банерів необхідно виконати декілька основних етапів. Зокрема, було створено контролери, layout-файли та шаблони для формування сторінок керування банерами в адміністративній частині Magento. Бізнес-логіка та робота з даними реалізовані у відповідних моделях і ресурсних моделях, що забезпечує коректну взаємодію з базою даних.

Також реалізовано механізм формування колекції банерів для їх подальшого відображення та обробки. Завантаження окремих записів здійснюється через ресурсну модель із використанням стандартних методів Magento 2.

Після реалізації основної логіки модуля було створено репозиторій банерів, який забезпечує доступ до даних через єдиний інтерфейс. Використання репозиторію спрощує отримання та обробку інформації про банери в інших частинах системи.

Для зручності роботи адміністратора у файлі etc/adminhtml/menu.xml було додано пункт меню в адміністративній панелі Magento, який забезпечує доступ до розділу керування банерами.

Лістинг 3.5 – Фрагмент коду додавання пункту меню

```
<menu>
    <add id="Magefan_PromoBanners::banners"
         title="Promo Banners"
         module="Magefan_PromoBanners"
         sortOrder="90"
         parent="Magento_Backend::content"/>
</menu>
```

Аналогічно до попереднього модуля, було визначено права доступу для адміністратора, які дозволяють працювати з функціональністю модуля в розділі Content → Promo Banners.

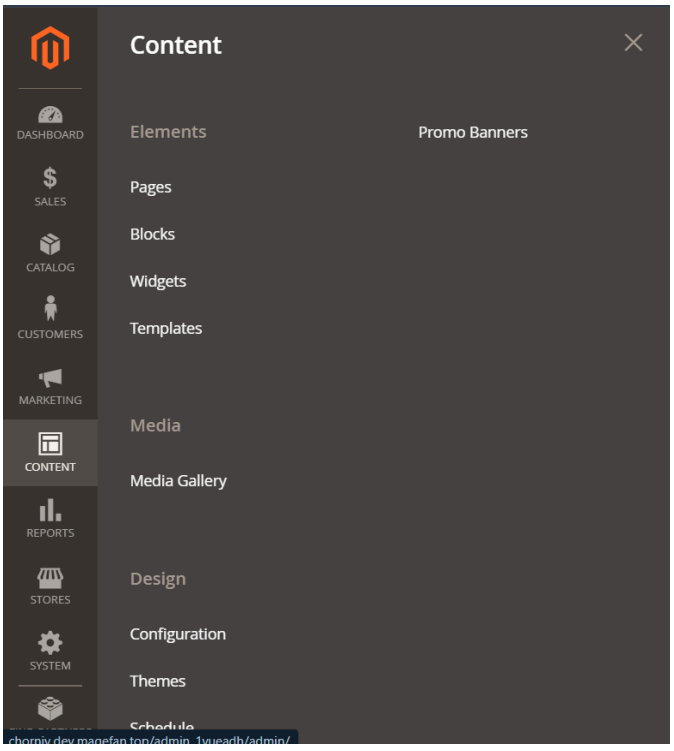


Рисунок 3.3.3 – Розділ керування банерами в адміністративній панелі

Наступним етапом проєктування модуля стала підготовка контролерів для реалізації базових операцій керування даними (CRUD) у панелі адміністратора.

Для управління записами використовується стандартний компонент Magento 2 — UI Grid, який забезпечує відображення списку банерів та виконання основних дій з ними.

Основні функціональні можливості сторінки списку банерів включають:

- перегляд наявних банерів;
- сортування записів за основними полями;
- пошук за ключовими словами;
- перехід до сторінки редагування банера;
- видалення окремих записів.

Таким чином, UI Grid виступає основним інструментом роботи адміністратора з промо-банерами та дозволяє ефективно керувати рекламним контентом магазину.

Для доповнення функціональності модуля реалізовано форму додавання, редагування та видалення банерів, яка пов’язана з бекенд-логікою модуля та забезпечує перевірку коректності введених даних.

Форма передбачає наявність таких основних полів:

- назва банера;
- статус активності (увімкнено / вимкнено);
- текстовий або описовий вміст банера;
- службові поля для збереження та видалення запису.

Після заповнення форми адміністратор має можливість зберегти внесені зміни або скасувати редагування.

На завершальному етапі реалізовано сторінку налаштувань модуля через механізм System Configuration (файл system.xml). У меню Stores → Configuration → Magefan → Promo Banners розміщено параметри, які дозволяють керувати роботою модуля без внесення змін у програмний код.

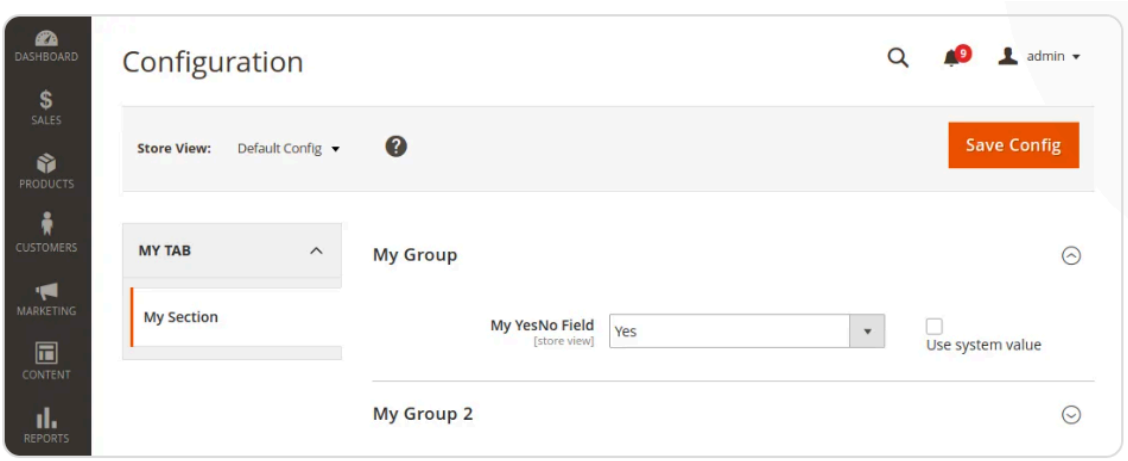


Рисунок 3.3.4 – Налаштування модуля Magefan_PromoBanners

3.4 Тестування модулів

Після розробки та запуску модулів системи було проведено їх тестування. Перевірка розпочалася з базового функціоналу платформи Magento 2 та розроблених розширень. Тестування здійснювалося на робочому стенді, який включає такі програмно-апаратні засоби:

- CMS: Magento 2.4.x
- Web-сервер: Apache / Nginx
- PHP: версія 7.4 / 8.1
- СУБД: MySQL 8.0
- Операційна система: Ubuntu 22.04
- Інструменти розробника: PhpStorm, Magento CLI, Composer, Chrome

Для перевірки роботи інтегрованих компонентів була створена тестова сторінка /my-super-puper-page, яка дозволяє адміністратору формувати власний список продуктів, додавати бренди, коригувати ціни та керувати блогом магазину (див. рис. 3.4.1).

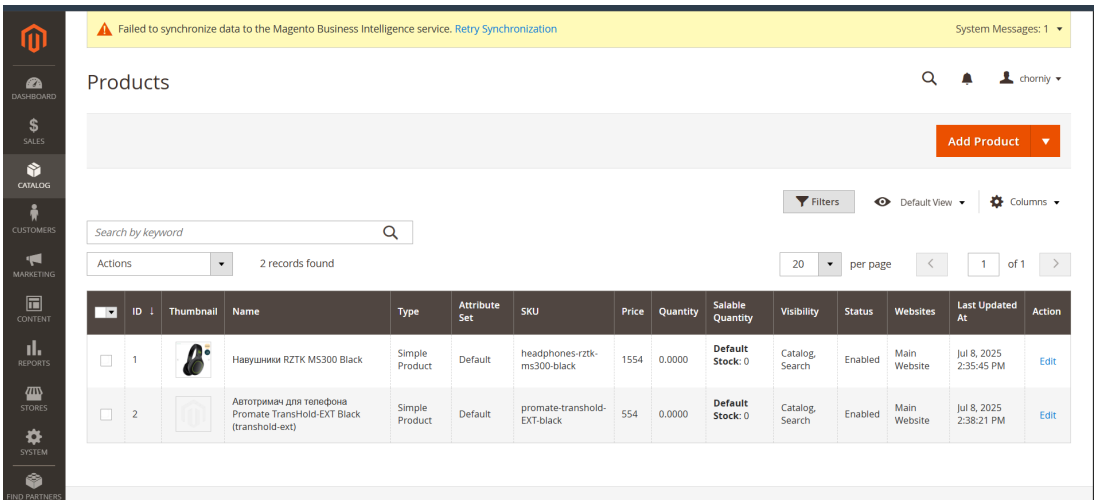


Рисунок 3.4.1 – Власній ціни та блог магазину

Під час тестування були перевірені такі аспекти роботи модулів:

- коректність встановлення модулів через CLI;
- реєстрація маршрутів та доступність сторінок;
- взаємодія з базою даних;
- робота плагінів та обсерверів;
- обмеження доступу для неавторизованих користувачів;
- працездатність cron-задач;
- робота адміністративних таблиць (створення, редагування, видалення записів);
- доступність функцій фільтрації, сортування, пагінації та пошуку в UI Grid.

Окрему увагу було приділено модулю Magefan_Frankenstein, для якого протестовано:

- приховування цін для неавторизованих користувачів або для груп, що не мають доступу;
- коректність заокруглення цін до цілих значень;
- виконання обмеження доступу до сторінок;
- роботу фонової крон-задачі, що автоматично змінює ціни.

Наступним етапом стало тестування модуля Magefan_Faq, який реалізує розширені можливості роботи з даними в адміністративній панелі Magento. Під час перевірки було протестовано відображення таблиці записів (див. рис. 3.4.2).

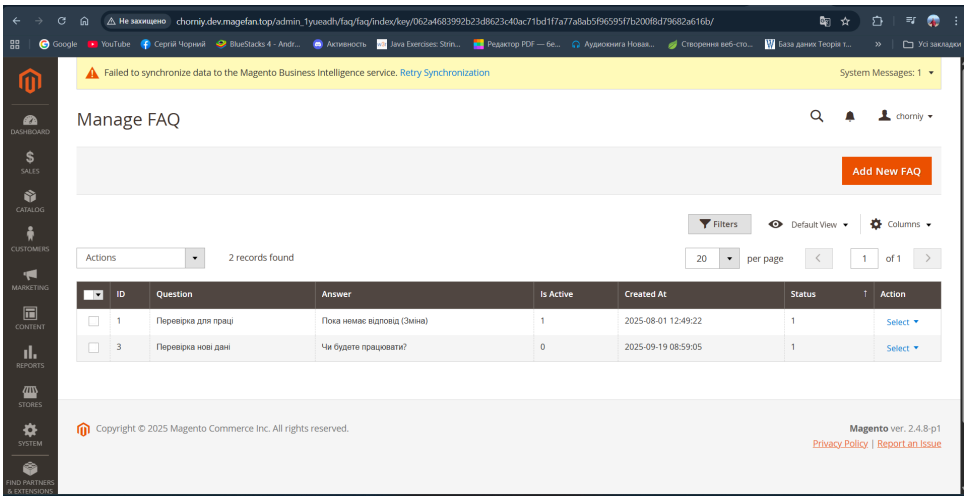


Рисунок 3.4.2 – Таблиця модуль Magefan_Faq

У таблиці відображаються дані, завантажені з бази даних, при цьому всі колонки коректно формуються відповідно до налаштувань UI Components. Окрема увага була зосереджена на роботі з записами. Для додавання нових даних реалізовано окрему форму (див. рис. 3.4.3), яка містить обов’язкові поля та механізм внутрішньої валідації. Після заповнення форми запис успішно створюється та відображається у загальному списку.

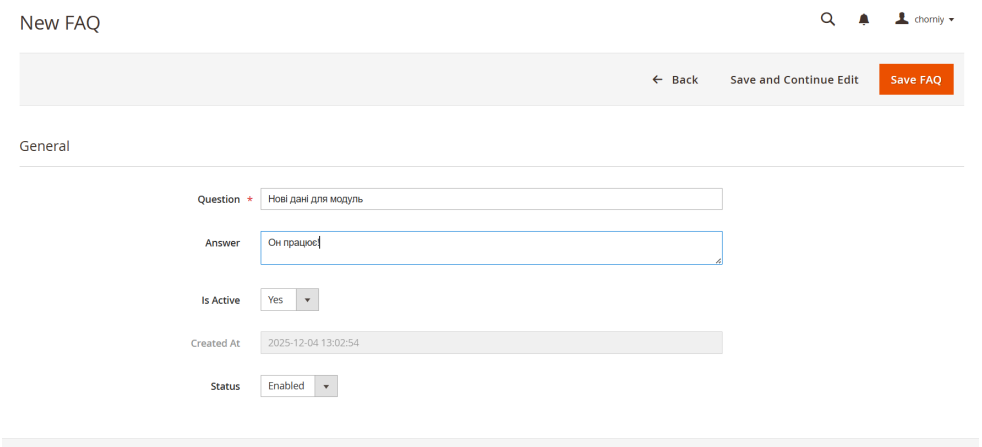


Рисунок 3.4.3 – Сторінка додавання нового дані Magefan_Faq

Редагування записів здійснюється через окрему форму, у якій внесені зміни зберігаються без затримок та одразу відображаються в таблиці (див. рис. 3.4.4).

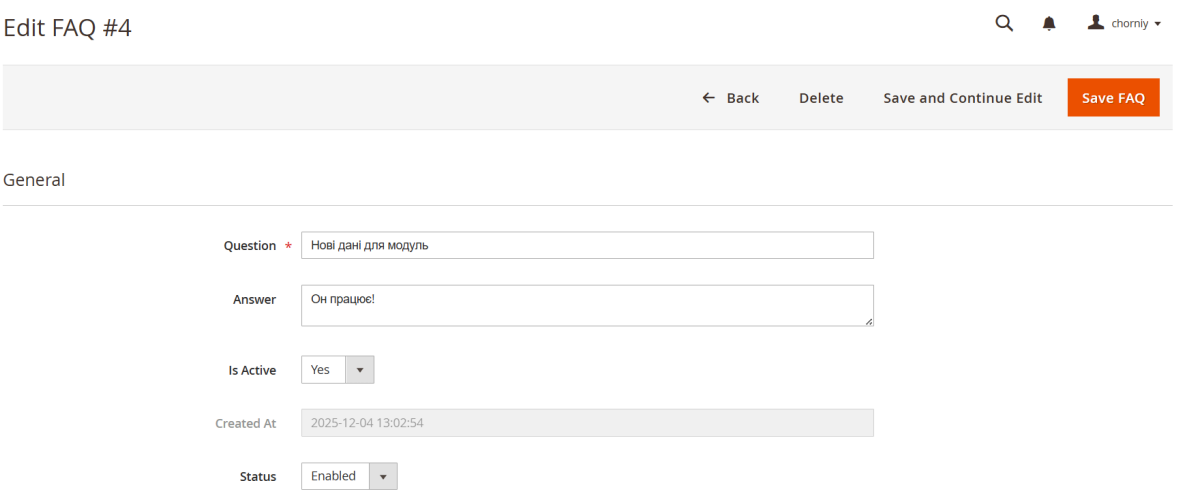


Рисунок 3.4.4 – Форма редагування дані

Система підтримує як видалення окремого запису, так і масове видалення декількох вибраних елементів через панель дій (див. рис. 3.4.5).

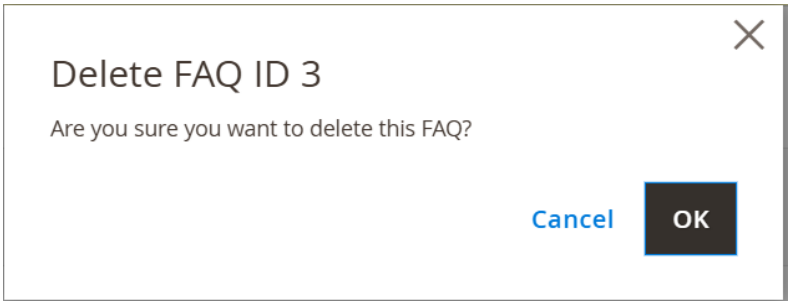


Рисунок 3.4.5 – Видалення записів

У модулі реалізовано функції пошуку та фільтрації, які дозволяють ефективно знаходити необхідну інформацію (див. рис. 3.4.6).

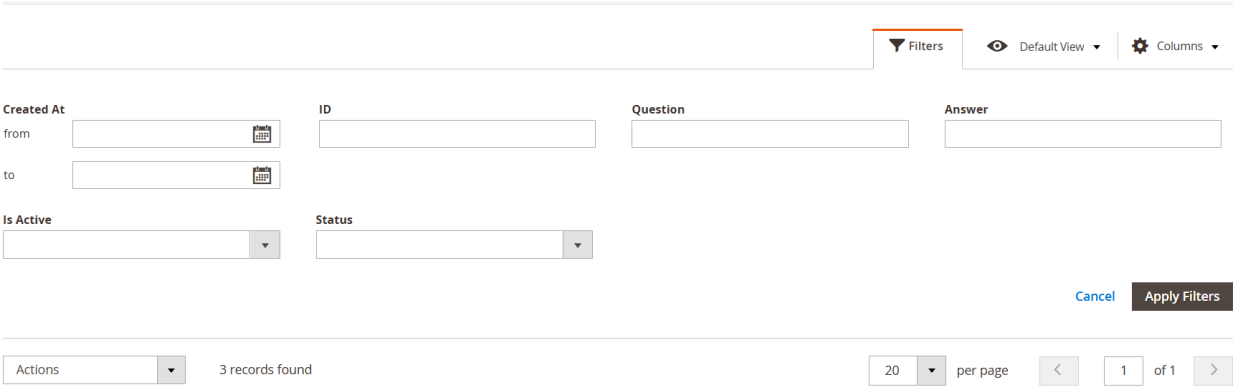


Рисунок 3.4.6 – Пошук і фільтрування таблиці

У пункті “Columns” адміністратор має можливість обрати, які стовпці повинні відображатися в таблиці (див. рис. 3.4.7).

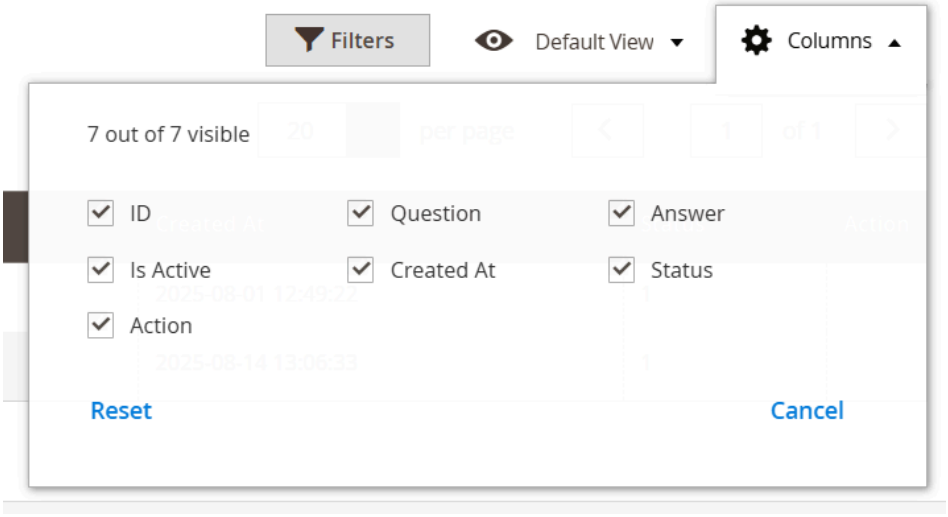


Рисунок 3.4.7 – Налаштування полів таблиці

Також було протестовано пагінацію та зміну кількості записів на сторінці. Сортування здійснюється шляхом натискання на заголовок відповідної колонки (див. рис. 3.4.8).

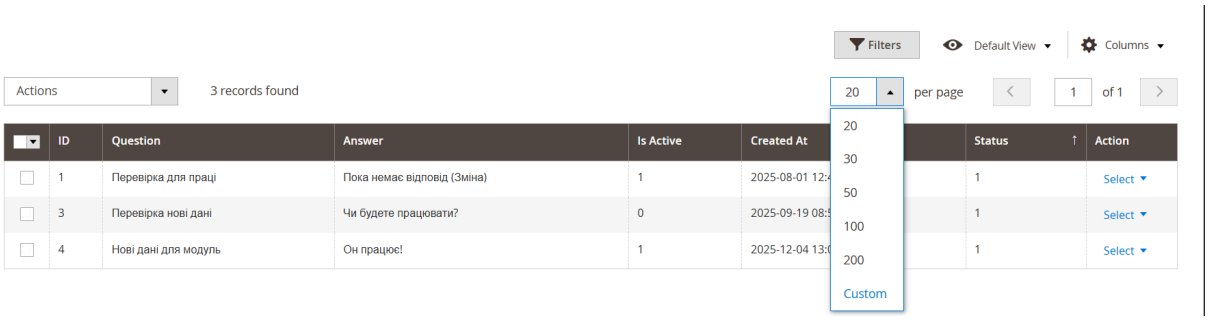


Рисунок 3.4.8 – Налаштування кількості записі на сторінці

Наступне тестування модуля Magefan_PromoBanners, ця модуль призначено для керування промо-банерами в адміністративній частині Magento 2. Тестування модуля здійснювалося в межах тих самих програмно-апаратних умов, що й для інших компонентів системи.

На етапі перевірки було протестовано процес встановлення та реєстрації модуля в системі, зокрема коректність підключення модуля через Magento CLI, наявність записів у таблиці magefan_promobanners, а також підключення структури бази даних, необхідної для зберігання інформації про банери.

Також було перевірено:

- коректність створення таблиці бази даних модуля;
- реєстрацію пункту меню в адміністративній панелі;
- наявність прав доступу, визначених у файлі acl.xml;
- підключення UI Components для формування адміністративного інтерфейсу.

Під час спроби доступу до сторінки керування банерами в адміністративній частині системи було виявлено проблему доступу, що призводила до відображення помилки 404. У межах відведеного часу не вдалося повністю локалізувати та усунути причину цієї проблеми, що унеможливило завершення тестування адміністративного інтерфейсу керування банерами.

Водночас базова архітектура модуля, структура файлів, організація моделей, ресурсних моделей та опис прав доступу були сформовані відповідно до рекомендацій і стандартів Magento 2. Реалізоване архітектурне рішення створює

основу для подальшого налагодження адміністративного інтерфейсу без необхідності зміни загальної структури модуля.

За результатами тестування підтверджено стабільну роботу розроблених модулів Magefan_Frankenstein та Magefan_Faq. Модуль Magefan_Frankenstein демонструє високу швидкодію та мінімальне навантаження на систему, тоді як Magefan_Faq забезпечує повноцінний CRUD-функціонал і ефективну роботу з адміністративними даними. Під час тестування модуля Magefan_PromoBanners було виявлено проблему з відображенням інтерфейсу керування в адміністративній панелі (помилка маршрутизації), що не дозволило повною мірою протестувати функціональні можливості цього модуля на рівні користувацького інтерфейсу, однак базова структура модуля та механізми взаємодії з базою даних були реалізовані відповідно до вимог Magento 2.

3.5 Порівняння результатів, ефективність.

Після проведене тестування трьох модулів показало, що вони виконують різні функціональні завдання та орієнтовані на різні сценарії використання в межах платформи Magento 2.

Модуль Magefan_Frankenstein характеризується високою швидкодією та низьким споживанням ресурсів. Його основний функціонал зосереджений на оптимізації роботи вітрини магазину: приховуванні цін для неавторизованих користувачів або окремих груп, заокругленні вартості товарів, обмеженні доступу до визначених сторінок та автоматизованій зміні цін за допомогою крон-задач. Завдяки мінімальному навантаженню на сервер модуль ефективно підвищує продуктивність вітрини.

Модуль Magefan_Faq орієнтований на адміністрування та роботу з великими обсягами структурованих даних. Він забезпечує повноцінний CRUD-механізм, підтримку сортування, фільтрації, пошуку, пагінації та налаштування відображення таблиць. Хоча цей модуль потребує більше ресурсів, він значно

спрощує керування адміністративними даними та підвищує зручність роботи адміністратора.

Модуль Magefan_PromoBanners поки що не має повністю працюючого інтерфейсу адміністративного управління, через що повноцінне тестування UI Grid і контролерів не виконано. Але сам архітектурний модуля, структура файлів, логіка взаємодії з базою даних та механізм розмежування прав доступу були реалізовані відповідно до стандартів Magento 2. Це дозволяє у подальшому без зміни архітектури завершити налаштування адміністративного інтерфейсу та використовувати модуль для керування промо-банерами.

Узагальнюючи результати порівняння, можна зробити висновок, що модуль Magefan_Frankenstein краще підходить для задач, пов’язаних із відображенням та швидкою обробкою даних на вітрині, тоді як модуль Magefan_Faq оптимізований для ефективної роботи адміністратора з великими наборами структурованої інформації, і модуль Magefan_PromoBanners має потенціал для повноцінного керування промо-банерами після завершення налаштування та виправлення інтерфейсу. Таким чином, усі три модулі доповнюють один одного та забезпечують комплексне підвищення продуктивності і зручності управління магазином.

3.6 Висновок до третий розділу

У цьому розділі було розглянуто процес створення, налаштування та тестування трьох модулів для Magento 2 — Magefan_Frankenstein, Magefan_Faq та Magefan_PromoBanners. Результати тестування підтвердили стабільність їх роботи двох перших модулів та показали потенціал третього.

Перший модуль Magefan_Frankenstein забезпечує приховування цін для певних груп користувачів, коректне заокруглення цін, обмеження доступу до сторінок і автоматичне коригування цін за допомогою крон-задачі. Під час тестування встановлено, що модуль працює швидко та не створює надмірного навантаження на систему.

Друга модуль Magefan_Faq реалізує повноцінну адміністративну підтримку роботи з інформацією у вигляді UI Grid та UI Form. Він дозволяє створювати, редагувати, видаляти, фільтрувати та сортувати записи, забезпечуючи зручну взаємодію з великими обсягами даних. Тестування підтвердило його стабільність та ефективність у роботі адміністратора.

Третій модуль Magefan_PromoBanners наразі не має повністю працюючого інтерфейсу адміністративного керування, через що його повне тестування UI Grid та контролерів не виконано. Проте архітектура модуля, структура файлів, логіка взаємодії з базою даних та механізм розмежування прав доступу були реалізовані відповідно до стандартів Magento 2. Це дозволяє у подальшому без зміни архітектури завершити налаштування адміністративного інтерфейсу та використовувати модуль для керування промо-банерами.

Таким чином, усі три модулі доповнюють один одного та забезпечують комплексне підвищення продуктивності і зручності роботи магазину: Magefan_Frankenstein — на рівні вітрини, Magefan_Faq — у межах адміністративної панелі, а Magefan_PromoBanners — для управління рекламним контентом після завершення налаштування інтерфейсу.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Під час розробки та оптимізації модулів системи Magento для вебсайтів особлива увага приділялася дотриманню вимог охорони праці, електробезпеки, пожежної безпеки та санітарно-гігієнічних норм, які регламентують умови роботи з персональними комп’ютерами. Оскільки виконання програмного проєкту пов’язане з тривалою роботою за ПК, важливою умовою його реалізації є створення безпечного та комфортного робочого середовища, що сприяє збереженню здоров’я виконавця та підвищенню продуктивності праці.

Організація умов праці під час виконання проєкту здійснювалася відповідно до чинної нормативно-правової бази України у сфері охорони праці. Основним документом є Закон України «Про охорону праці» [33], який визначає загальні принципи забезпечення безпечних і нешкідливих умов праці. Також враховувалися Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені наказом Міністерства соціальної політики України № 207 [34], Правила пожежної безпеки в Україні [35] та державні будівельні норми, що регламентують освітлення і мікроклімат приміщень [36]. Контроль за дотриманням вимог охорони праці на кафедрі комп’ютерних систем та мереж здійснюється завідувачем кафедри.

Розробка та тестування модулів Magento виконувалися у комбінованому форматі — як в умовах офісного робочого місця, так і в домашньому середовищі. Такий підхід зумовив необхідність адаптації вимог охорони праці до різних умов виконання робіт. Незалежно від місця роботи, дотримувалися основні принципи безпечної експлуатації комп’ютерної техніки, раціональної організації робочого простору, належного освітлення та електробезпеки.

Організація робочого місця програміста має важливе значення, оскільки діяльність пов’язана з високим рівнем розумового навантаження, тривалою

статичною позою та постійною концентрацією уваги. Під час виконання проєкту дотримувалися основні вимоги ергономіки. Робочий стіл та крісло підбирається з урахуванням можливості регулювання висоти, що дозволяє забезпечити правильне положення тіла. Монітор комп'ютера розташовувався на відстані 60–70 см від очей, а його верхня межа знаходилася на рівні погляду або трохи нижче. Таке розташування сприяє зменшенню навантаження на зоровий апарат і шийний відділ хребта.

Особлива увага приділялася режиму праці та відпочинку. Тривала робота за комп'ютером може призводити до зорової втоми, зниження концентрації та загального перевтомлення. З метою запобігання негативним наслідкам застосовувався раціональний режим праці, який передбачав регулярні перерви тривалістю 10–15 хвилин після кожної години безперервної роботи. Під час перерв виконувалися вправи для очей, легка фізична розминка та зміна виду діяльності, що сприяло відновленню працездатності.

Важливим чинником безпечних умов праці є освітлення робочого місця. Під час виконання проєкту використовувалося комбіноване освітлення, що включає природне та штучне світло. Рівень освітленості підтримувався в межах 300–500 лк відповідно до вимог ДБН В.2.5-28:2018 [36]. Робоче місце було організоване таким чином, щоб уникнути прямих відблисків і засліплення на екрані монітора. Світильники розташовувалися збоку від користувача, а джерела світла мають нейтральну світло температуру, що зменшувало напруження зору.

Під час роботи з комп'ютерною технікою також враховувалися параметри мікроклімату приміщення. Комфортні мікрокліматичні умови безпосередньо впливають на самопочуття та ефективність роботи користувача. Температура повітря в робочому середовищі підтримувалася в межах 20–24 °C, а відносна вологість — 40–60 %, що відповідає чинним нормативним вимогам. Для підтримання належного мікроклімату здійснювалося регулярне провітрювання приміщення, а за необхідності використовувалися допоміжні засоби регулювання температури.

Електробезпека є одним із ключових аспектів охорони праці під час роботи з комп’ютерною технікою. У межах виконання проєкту дотримувалися вимоги Правил безпечної експлуатації електроустановок споживачів [37]. Комп’ютерне обладнання підключалося через справні розетки із заземленням та мережеві фільтри. Для захисту техніки та даних використовувалося джерело безперебійного живлення (UPS), що дозволяло уникнути негативних наслідків раптового зникнення електроенергії. Не допускалося використання пошкоджених кабелів, перевантаження електромережі та застосування несертифікованих подовжувачів.

Пожежна безпека також була важливою складовою під час виконання проєкту. Основними потенційно небезпечними факторами є короткі замикання, перегрів електрообладнання та порушення правил експлуатації електроприладів. В офісному приміщенні забезпечено наявність первинних засобів пожежогасіння, планів евакуації та вільного доступу до евакуаційних виходів відповідно до Правил пожежної безпеки в Україні [35]. У домашніх умовах дотримувалися рекомендацій щодо безпечного використання електроприладів, недопущення перекриття вентиляційних отворів та вимкнення обладнання після завершення роботи.

Таким чином, під час розробки та оптимізації модулів системи Magento було комплексно враховано вимоги охорони праці, електробезпеки та пожежної безпеки. Це дозволило створити безпечні та комфортні умови роботи, знизити вплив шкідливих і небезпечних факторів, а також забезпечити ефективне виконання програмного проєкту відповідно до чинного законодавства України.

4.2 Фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі

Під час роботи з комп’ютерною технікою та використання локальних комп’ютерних мереж користувачі можуть зазнавати впливу різних факторів ризику, які негативно впливають на стан здоров’я та працездатність. Тривале перебування за персональним комп’ютером, високе інформаційне навантаження та

недотримання вимог ергономіки робочого місця можуть призводити до розвитку функціональних порушень і професійно зумовлених захворювань [38].

До основних факторів ризику, характерних для користувачів комп'ютерних мереж, належать:

- Тривале статичне навантаження. Постійне сидіння за комп'ютером призводить до напруження м'язів спини, шиї та плечей, що може викликати остеохондроз та больові синдроми [39].
- Зорове навантаження. Робота з монітором протягом тривалого часу часто стає причиною втоми очей, сухості слизової оболонки, погіршення гостроти зору та розвитку комп'ютерного зорового синдрому.
- Психоемоційне перевантаження. Інтенсивна робота з інформацією, багатозадачність та обмежений час на виконання складних завдань можуть спричиняти стрес, дратівливість та загальне зниження працездатності [39].
- Недостатня організація робочого місця. Невідповідність висоти столу та стільця антропометричним даним користувача, а також неправильне розташування монітора і клавіатури суттєво підвищує ризик захворювань опорно-рухового апарату [38].
- Електромагнітне випромінювання та мікроклімат. Хоча сучасне обладнання має низький рівень випромінювання, у разі великої концентрації техніки в одному приміщенні можливе сумарне негативне навантаження на організм. Також критично важливими є параметри температури, вологості та ефективність вентиляції приміщення.

Можливі порушення здоров'я включають:

- Неврологічні симптоми: регулярний головний біль, підвищена втома та зниження концентрації уваги;
- Офтальмологічні проблеми: порушення зору (запальні процеси, синдром «сухого ока», прогресуюча короткозорість);
- Ортопедичні патології: захворювання опорно-рухового апарату (сколіоз, остеохондроз, стійкі м'язові спазми);

- Психологічні розлади: нервове перенапруження, хронічні стресові стани, порушення режиму сну;
- Загальносоматичні прояви: загальне зниження працездатності та швидке психофізичне виснаження організму.

Таким чином, робота за комп’ютером та використання мережевих технологій створює комплексні ризики для користувачів. Їх необхідно враховувати та мінімізувати шляхом науково обґрунтованої організації робочих умов і суворого дотримання рекомендацій з охорони праці.

ВИСНОВКИ

У ході виконання магістерської роботи було досліджено та розроблено програмне забезпечення на базі платформи Magento 2 для електронної комерції. У першому розділі проведено аналіз предметної області, визначено переваги Magento серед інших платформ для створення інтернет-магазинів, розглянуто функціональні та нефункціональні вимоги, а також проблеми продуктивності системи та способи їх оптимізації. Це дозволило зрозуміти, яким чином забезпечити стабільну та швидку роботу інтернет-магазинів.

Другий розділ присвячено проєктуванню модуля для Magento 2. Було обґрунтовано вибір технологій, розроблено UML-діаграми, описано сценарії роботи модуля, спроєктовано базу даних та визначено механізми взаємодії компонентів системи. Це дало змогу створити структурований, масштабований і розширюваний модуль, що відповідає вимогам проєкту та може бути доповнений новим функціоналом у майбутньому.

У третьому розділі розглянуто створення, налаштування та тестування трьох модулів Magento 2 — **Magefan_Frankenstein**, **Magefan_Faq** та **Magefan_PromoBanners**. Перші два модулі працюють стабільно: Magefan_Frankenstein керує відображенням цін і доступом користувачів, Magefan_Faq забезпечує зручну роботу адміністратора з великими обсягами даних. Інтерфейс Magefan_PromoBanners поки що не повністю функціонує через проблему маршрутизації, але його архітектура та логіка взаємодії з базою даних реалізовані відповідно до стандартів Magento 2, що дозволяє у майбутньому завершити налаштування адміністративного інтерфейсу. Використання всіх трьох модулів підвищує ефективність управління даними та контентом магазину.

Четвертий розділ стосується охорони праці та безпеки розробників. Дотримання вимог щодо освітлення, мікроклімату, шуму, ергономіки, електробезпеки та правил поведінки у надзвичайних ситуаціях забезпечує безпечні умови праці та збереження здоров’я працівників.

Підсумком роботи є створення ефективного, стабільного і безпечного програмного рішення на базі Magento 2, яке відповідає сучасним вимогам електронної комерції та може бути використане для подальшого розвитку інтернет-магазинів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Череп А. В., Власенко Т. А. Розвиток електронної комерції в умовах цифровізації економіки. *Економіка та суспільство*. 2020. № 23. [Електронний ресурс]. – Режим доступу: URL: http://www.economyandsociety.in.ua/journal/23_2020/69.pdf

2. Кузнєцова С. А. Переваги та недоліки електронної комерції для сучасного бізнесу. *Вісник економічної науки України*. 2019. № 2. С. 125–130.

3. Laudon K. C., Traver C. G. *E-commerce 2023: Business. Technology. Society*. 18th ed. Pearson Education, 2022. 928 p.

4. Sholtan N. N. Requirements for E-Commerce Systems Security and PCI DSS Compliance. *International Journal of Computer Science and Network Security*. 2021. Vol. 21, No. 10. P. 13-18.

5. Turban E., Whiteside J. *E-commerce: A Managerial and Social Networks Perspective*. Springer, 2020. 750 p.

6. Frayer M. Headless Commerce: A Deep Dive into its Benefits and Challenges. *E-Commerce Times*. 2023.

7. Що таке CMS та як вона працює? *Hostiq.ua: Вікі*. [Електронний ресурс]. – Режим доступу: URL: https://hostiq.ua/wiki/ukr/cms/

8. E-commerce рішення. *Brander.ua*. [Електронний ресурс]. – Режим доступу: URL: https://brander.ua/what-we-offer/e-commerce

9. Zadorozhny V. V. Performance Challenges and Optimization Strategies in Magento E-Commerce Platform. *Journal of Technical Systems*. 2020. Vol. 10, No. 2. P. 45-53.

10. *Magento Documentation: Performance Optimization Guide*. Adobe Commerce Official Website.

11. *Magento Developer Documentation: System Requirements*. Adobe Commerce Official Website.

12. PHP 8.0 Release Notes. *PHP.watch*. [Електронний ресурс]. – Режим доступу: URL: https://php.watch/versions/8.0

13. HTML Tutorial. *W3Schools UA*. [Електронний ресурс]. – Режим доступу: URL: https://w3schoolsua.github.io/html/

14. CSS Tutorial. *W3Schools UA*. [Електронний ресурс]. – Режим доступу: URL: https://w3schoolsua.github.io/css/

15. JS Tutorial. *W3Schools UA*. [Електронний ресурс]. – Режим доступу: URL: https://w3schoolsua.github.io/js/

16. Що таке XML-файл. *Horoshop Blog*. [Електронний ресурс]. – Режим доступу: URL: https://horoshop.ua/ua/blog/xml-file/

17. Де використовується SQL і чому він потрібен програмістам. *IT Step Blog*. [Електронний ресурс]. – Режим доступу: URL: https://kiev.itstep.org/blog/where-sql-is-used-and-why-programmers-need-it-so-much

18. MySQL. *Вікіпедія*. [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/MySQL

19. Порожній В. Архітектура Magento 2: Dependency Injection та Service Contracts. *Наукові праці Чорноморського державного університету*. 2019. Вип. 21. С. 110-116.

20. Magento 2 Extension File System. *Magefan Blog*. [Електронний ресурс]. – Режим доступу: URL: https://magefan.com/ua/blog/magento2-extension-file-system

21. Іванов П. Модульна архітектура e-commerce платформ. Київ: Прогрес, 2022. 200 с.

22. Сміт Д. XML-конфігурації в Magento 2. *Enterprise E-commerce*. 2020. Т. 5, № 2. С. 45-55.

23. Іванов П. UML: Практичний посібник з моделювання. Київ: АртЕк, 2021. 300 с.

24. Діаграма класів UML. *Вікіпедія*. [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/Діаграма_класів

25. Sequence Diagrams. *Max Zosim Blog*. [Електронний ресурс]. – Режим доступу: URL: https://www.maxzosim.com/sequence-diagrams/

26. UML Component Diagram: повний посібник. *MindOnMap*. [Електронний ресурс]. – Режим доступу: URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/

27. Діаграма діяльності (Activity Diagram). *Studfile*. [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/5200239/page:7/

28. Вертикальна та горизонтальна архітектура баз даних. *Технічний огляд*. 2021.

29. Проєктування БД для e-commerce систем. *Web Development Journal*. 2022.

30. *Mage2Gen: Magento 2 Module Generator*. [Електронний ресурс]. – Режим доступу: URL: https://mage2gen.com/

31. *Magento Developer Documentation: Controllers*. Adobe Commerce Official Website.

32. Коваль О. Методи оптимізації продуктивності веб-систем. – Львів: Наука, 2023. 150 с.

33. Закон України «Про охорону праці»: Закон України від 14.10.1992 № 2694-XII.

34. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями: Наказ Мінсоцполітики України від 14.02.2018 № 207.

35. Правила пожежної безпеки в Україні: Наказ МВС України від 30.12.2014 № 1417.

36. ДБН В.2.5-28:2018. Природне і штучне освітлення. Державні будівельні норми України.

37. Правила безпечної експлуатації електроустановок споживачів: НПАОП 40.1-1.21-98.

38. Стручок В. С. Безпека в надзвичайних ситуаціях: методичний посібник. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

39. Стручок В. С. Техноекологія та цивільна безпека: навчальний посібник. Тернопіль: ФОП Паляниця В. А., 2023. 156 с.

ДОДАТКИ

ДОДАТОК А – Структура модулів Magento

app/code/Magefan/PromoBanners/

- registration.php
- etc/
    - module.xml
    - di.xml
    - frontend/
        - routes.xml
- Controller/
    - Index/
        - Index.php
- Model/
- Banner.php
- ResourceModel/
    - Banner.php
    - Banner/Collection.php
- Block/
    - Banner.php
- view/
    - frontend/
        - layout/
            - promo_index_index.xml
        - templates/
            - banner.phtml

ДОДАТОК Б – Лістинг коду

Лістинг файлу Magefan/Faq/Controller/Index/index.php

```
<?php
namespace Magefan\Faq\Controller\Index;

use Magento\Framework\App\Action\Action;
use Magento\Framework\App\Action\Context;
use Magento\Framework\View\Result\PageFactory;

class Index extends Action
{
    protected $resultPageFactory;

    public function __construct(Context $context, PageFactory $resultPageFactory)
    {
        $this->resultPageFactory = $resultPageFactory;
        parent::__construct($context);
    }

    public function execute()
    {
        return $this->resultPageFactory->create();
    }
}
```

Лістинг файлу Magefan/Faq/Model/ResourceModel/Faq/Collection.php

```
class Collection extends AbstractCollection
{
    protected function _construct()
    {
        $this->_init(
            \Magefan\Faq\Model\Faq::class,
            \Magefan\Faq\Model\ResourceModel\Faq::class
        );
    }
}
```

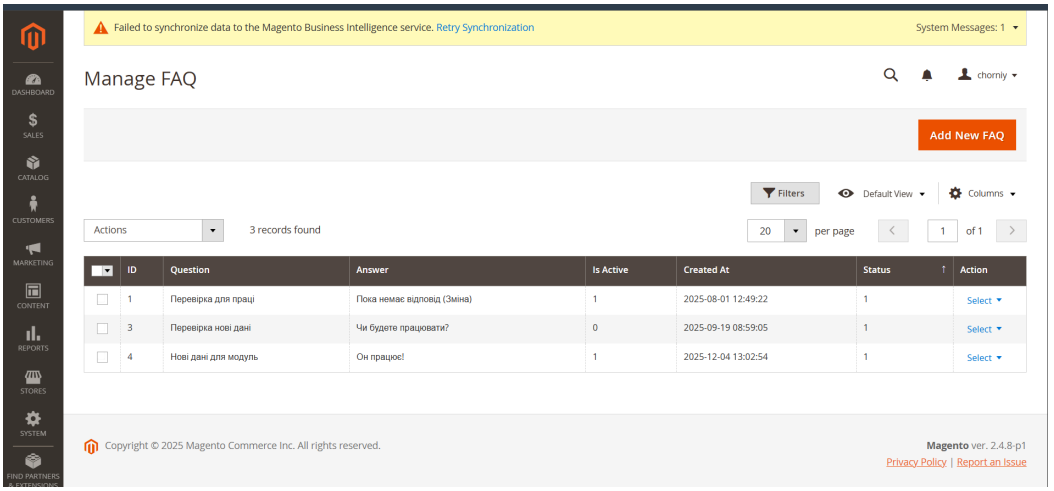
Лістинг файлу Magefan/Faq/ViewModel/Faq.php

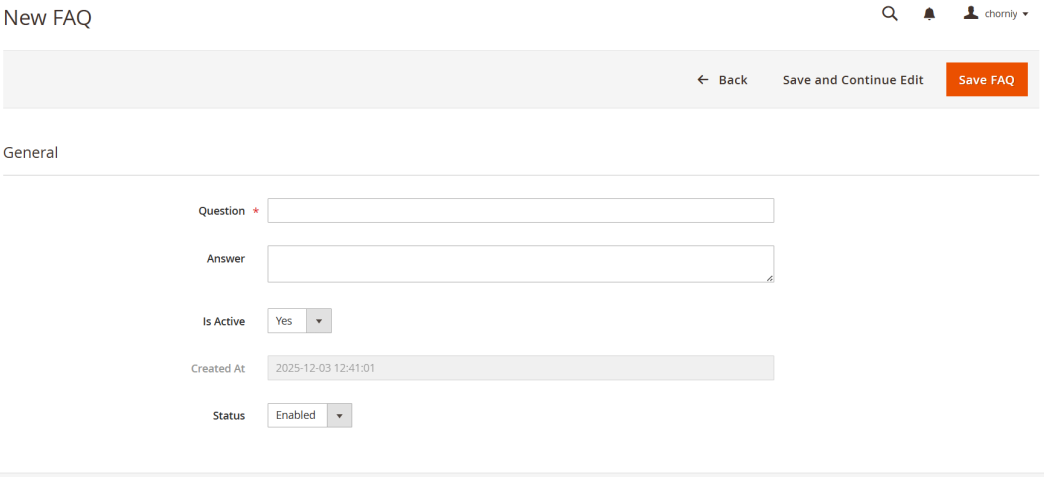
```
class Faq implements ArgumentInterface
{
    protected $collectionFactory;

    public function __construct(CollectionFactory $collectionFactory)
    {
        $this->collectionFactory = $collectionFactory;
    }

    public function getActiveFaqs()
    {
        $collection = $this->collectionFactory->create();
        $collection->addFieldToFilter('status', ['eq' => 1]);
        return $collection;
    }
}
```

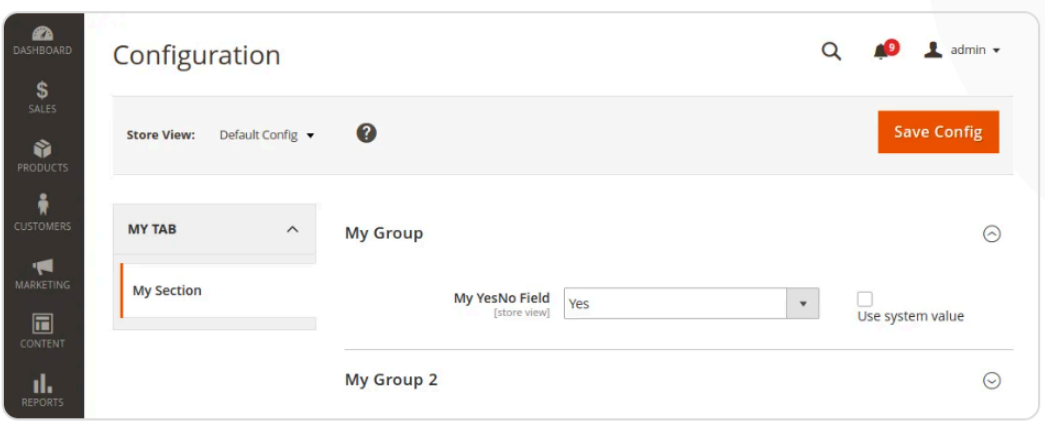
Лістинг файлу Magefan/Faq/Setup/InstallSchema.php

```
class InstallSchema implements InstallSchemaInterface
{
    public function install(SchemaSetupInterface $setup, ModuleContextInterface
$context)
    {
        $setup->startSetup();

        if (!$setup->tableExists('magefan_faq')) {
            $table = $setup->getConnection()->newTable(
                $setup->getTable('magefan_faq')
            )
                ->addColumn('id', Table::TYPE_INTEGER, null, [
                    'identity' => true,
                    'nullable' => false,
                    'primary'  => true,
                    'unsigned' => true,
                ], 'ID')
                ->addColumn('question', Table::TYPE_TEXT, 255, ['nullable' =>
false], 'Question')
                ->addColumn('answer', Table::TYPE_TEXT, '64k', ['nullable' =>
false], 'Answer')
                ->addColumn('product_sku', Table::TYPE_TEXT, 255, ['nullable' =>
false], 'Product SKU')
                ->addColumn('status', Table::TYPE_SMALLINT, null, ['nullable' =>
false, 'default' => 0], 'Status')
                ->setComment('FAQ Table');
            $setup->getConnection()->createTable($table);
        }

        $setup->endSetup();
    }
}
```

ДОДАТОК В – Скріншоти роботи модуля Magefan_Faq







ДОДАТОК Г – Скріншоти роботи модуля Magefan_Frankenstein