МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ІНТЕРНЕТ-МАГАЗИНІВ НА БАЗІ ВІДКРИТИХ E-COMMERCE-СИСТЕМ»**

Виконав: студент 2 курсу, групи 8.1222-дн
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
В.В. Лобачов
(ініціали та прізвище)
Керівник кандидат технічних наук, доцент, д.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри, кандидат фіз.-мат.наук, доцент Лісняк Андрій Олександрович
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " травня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лобачову Володимири Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка програмного забезпечення для підтримки інтернет-магазинів на базі відкритих e-commerce систем

керівник роботи Борю Сергій Юрійович, д.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 28.02.2024

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

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

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка пз для збору додаткових платежів з клієнтів інтернет-магазину

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 10.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.05.2023	
2.	Збір вихідних даних.	27.06.2023	
3.	Обробка методичних та теоретичних	19.07.2023	
	джерел.		
4.	Розробка першого та другого розділу.	12.08.2023	
5.	Розробка третього розділу.	13.10.2023	
6.	Оформлення та нормоконтроль	01.12.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	13.03.2024	

Студент _____ В.В. Лобачов
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення для підтримки інтернет-магазинів на базі відкритих E-commerce-систем»: 44 с., 4 рис., 15 джерел.

Веб-магазин, комп'ютерна система, клієнтський досвід, програмне забезпечення, розробка програмного забезпечення, magento 2, e-commerce.

Об'єкт дослідження – розробка програмного забезпечення для підтримки інтернет-магазинів на базі відкритих E-commerce-систем

Мета роботи: розробити та впровадити програмний модуль Magento 2 для підтримки інтернет-магазинів на основі відкритих e-commerce систем котрий буде забезпечувати створення журналів продуктів для кожного користувача.

Метод дослідження – описовий аналітичний.

Кваліфікаційна робота розглядає сучасні підходи до розробки програмного забезпечення для підтримки інтернет-магазинів на основі відкритих E-commerce-систем, а також у створенні та інтеграції модуля для Magento 2. Цей модуль спрямований на оптимізацію процесу створення журналів продуктів для кожного користувача, що значно підвищує функціональність та зручність використання платформи для користувачів.

У процесі дослідження детально розглянуті ключові аспекти вибору E-commerce-систем для підтримки інтернет-магазинів та особливості розробки програмного забезпечення для їх ефективного функціонування. Основним результатом дослідження стало успішне створення та інтеграція модуля для Magento 2, який не лише оптимізує процес створення журналів продуктів, але й сприяє підвищенню ефективності та прибутковості бізнесу.

SUMMARY

Master’s qualifying paper «Development of software for supporting internet stores based on open E-commerce systems»: 44 pages, 4 figures, 0 tables, 15 sources, 0 appendices.

Online store, computer system, customer experience, software, software development, Magento 2, e-commerce.

Object of investigation – development of software for supporting internet stores based on open E-commerce systems.

Purpose of the paper: development and implemented a software module for Magento 2 to support internet stores based on open e-commerce systems, which will provide the creation of product journals for each user.

Method of investigation – descriptive analytical.

Master’s qualifying paper explores modern approaches to developing software for supporting internet stores based on open E-commerce systems, as well as the creation and integration of a module for Magento 2. This module is aimed at optimizing the process of creating product journals for each user, significantly enhancing the functionality and convenience of platform use for users.

Key aspects of choosing E-commerce systems to support internet stores and features of software development for their effective operation were thoroughly examined during the research. The main result of the research was the successful creation and integration of a module for Magento 2, which not only optimizes the process of creating product journals but also contributes to increasing the efficiency and profitability of the business.

ЗМІСТ

ВСТУП

У сучасному світі інформаційних технологій електронна комерція (E-commerce) стала невід'ємною частиною бізнесу, надаючи підприємствам можливість розширювати свою аудиторію та ефективно взаємодіяти з клієнтами через Інтернет-магазини. Розробка програмного забезпечення для підтримки інтернет-магазинів є актуальною та важливою задачею в контексті зростання конкуренції та швидкої динаміки ринку. Особливу увагу приділяється створенню модуля, який дозволить інтернет-магазинам додавати каталоги продуктів на свої веб-сайти для полегшення пошуку та покупки продуктів, що надасть їм конкурентні переваги.

Магістерська робота присвячена розробці програмного забезпечення для підтримки інтернет-магазинів з використанням відкритих E-commerce-систем. У роботі вивчається роль E-commerce-систем у функціонуванні інтернет-магазинів, здійснюється вибір платформи Magento 2 для розробки програмного забезпечення та аналіз можливостей цієї системи порівняно з іншими популярними E-commerce-системами.

Цілью роботи є створення модулю, який додає каталоги продуктів на веб-сайти для більш зручного інтерактивного пошуку продуктів та надання інтернет-магазинам конкурентних переваг

Детальний огляд Magento 2 дозволяє з'ясувати технічні аспекти обраної платформи, а розробка модуля для створення журналів продуктів у Magento 2 допомагає детально вивчити технічні особливості розробки та взаємодії модуля з основною системою.

Отже, дана магістерська робота розглядає широкий спектр питань, пов'язаних із розробкою програмного забезпечення для інтернет-магазинів та застосуванням сучасних технологій

1 ОГЛЯД E-COMMERCE-СИСТЕМ І ЇХ РОЛЬ В ІНТЕРНЕТ-МАГАЗИНАХ

1.1 Введення в E-commerce-системи

В сучасному світі, що надзвичайно динамічно розвивається, електронна комерція (E-commerce) стала визначальним елементом бізнес-середовища. Термін "E-commerce" охоплює широкий спектр електронних транзакцій, які здійснюються через Інтернет, від онлайн-купівлі товарів і послуг до електронного обміну даними між бізнесами. Зараз E-commerce-системи виконують значно більше функцій, ніж просто забезпечення можливості онлайн-купівлі. Вони виступають інтегрованими платформами, які спрощують усі етапи від управління запасами до взаємодії з клієнтами та аналізу даних.

Розвиток інтернет-технологій та зростання доступності до мережі призвели до експоненційного зростання популярності інтернет-торгівлі. Для розуміння та оцінки масштабів цього явища важливо розглянути ключові аспекти, що стосуються E-commerce-систем.

Одним із важливих аспектів E-commerce-систем є інтернет-магазини, що стали віртуальними вікнами для товарів та послуг. Їхнє загальне або спеціалізоване спрямування робить їх ефективними інструментами для реалізації широкого спектру товарів та послуг.

Популярність E-commerce-систем зумовлена не лише можливістю глобального розширення бізнесу, але й можливістю забезпечення безперервного доступу до товарів та послуг. Онлайн-торгівля відкриває можливості для підприємств працювати без обмежень територією та годинами, що є важливим елементом в умовах глобалізації та конкуренції.

Концепція E-commerce-систем

E-commerce-системи представляють собою комплекс програмних та апаратних рішень, спрямованих на організацію та здійснення електронної

комерції. Ці системи включають в себе не лише платформи для онлайн-торгівлі, а й інструментарій для управління складським обліком, обробки платежів, взаємодії з клієнтами та аналізу даних. [4]

Ключові елементи E-commerce-систем

Однією з ключових складових E-commerce-систем є інтернет-магазини, які виступають віртуальними платформами для відображення товарів та послуг, а також для здійснення процесу їх придбання. Ці магазини можуть бути загального призначення або спеціалізованими, враховуючи різні галузі бізнесу.

Крім того, системи електронної комерції включають в себе інструменти для управління контентом, аналітики, автоматизації маркетингу, а також забезпечують інтеграцію з іншими системами, такими як системи управління взаємовідносинами з клієнтами (CRM) та системи управління ресурсами підприємства (ERP). [3]

Роль E-commerce-систем у бізнесі

Електронна комерція визначено впливає на стратегії бізнесу, дозволяючи підприємствам розширювати свою аудиторію та здійснювати глобальну торгівлю. E-commerce-системи роблять можливим проведення бізнесу в режимі 24/7, що забезпечує постійну доступність для клієнтів незалежно від їхнього розташування та часового поясу.

Тенденції та перспективи розвитку E-commerce-систем

Розширення мобільних технологій, впровадження штучного інтелекту та розумних аналітичних інструментів - це лише кілька напрямків, які визначають сучасні тенденції розвитку E-commerce-систем. Зростання безпеки та захисту даних клієнтів, а також забезпечення персоналізованого досвіду користувача, є основними завданнями, над якими працюють розробники цих систем.

У подальших розділах магістерської роботи буде проведений докладний аналіз вибору платформи Magento 2 як основи для розробки програмного

забезпечення для підтримки інтернет-магазинів та дослідження сучасних технологій, які можуть поліпшити їхню функціональність.

1.2 Роль E-commerce-систем у сучасних інтернет-магазинах

Роль E-commerce-систем у сучасних інтернет-магазинах неоціненно важлива, оскільки вони визначають ефективність, гнучкість та конкурентоспроможність цих торговельних платформ. Зазначені системи відіграють ключову роль у кожному аспекті функціонування інтернет-магазинів, від управління продуктовим асортиментом до взаємодії з клієнтами та аналізу даних.

E-commerce-системи грають ключову роль у поліпшенні процесу оформлення замовлення для клієнтів. Вони надають можливість легко та швидко додавати товари до кошика, вибирати методи доставки та оплати, а також забезпечують безпеку та захист персональних даних покупців під час транзакцій.

E-commerce-системи надають інструменти для проведення маркетингових кампаній, включаючи розсилки електронною поштою, рекламу на сайті, знижки та промо-акції. Це дозволяє інтернет-магазинам привертати нових клієнтів та стимулювати покупки.

Розглянемо основні аспекти ролі E-commerce-систем у сучасних інтернет-магазинах.

Управління та Відображення Товарів та Послуг

Управління та відображення товарів та послуг у сучасних E-commerce-системах є важливою складовою для ефективного функціонування інтернет-магазинів. Ці системи надають різноманітні інструменти для зручного представлення та організації продукції. Основні аспекти цього процесу включають:

1. **Каталог Товарів:** E-commerce-системи дозволяють легко створювати та управляти каталогами товарів. Адміністраторам магазинів надається можливість додавання нових товарів, редагування існуючих, а також організація товарів у категорії для зручного пошуку покупцями.
2. **Зображення та Опис Товарів:** Інтернет-магазини дозволяють додавати до кожного товару ілюстрації високої якості та докладні описи. Це допомагає покупцям отримати повну інформацію про товар перед покупкою.
3. **Фільтри та Сортування:** Системи електронної комерції надають можливість застосовувати фільтри для швидкого та зручного пошуку товарів за різними параметрами, такими як ціна, бренд, розмір тощо. Також, покупці можуть сортувати товари за різними критеріями.
4. **Інвентаризація та Доступність:** E-commerce-системи ведуть облік залишків товарів, автоматично оновлюючи інформацію про доступність. Це дозволяє уникнути ситуацій з розпродажем товару, якого немає в наявності.
5. **Спеціальні Пропозиції та Знижки:** Магазини можуть легко налаштовувати різноманітні акції та знижки на товари. Інформація про знижки та спеціальні пропозиції відображається яскраво та привертає увагу покупців.

Усі ці функції спрощують процес покупки для клієнтів та дозволяють магазинам ефективно управляти своїм асортиментом, щоб задовольняти потреби різних сегментів ринку.

Інтеграція з Іншими Системами

Інтеграція з іншими системами в сфері електронної комерції є ключовим аспектом для забезпечення безперебійного та ефективного функціонування інтернет-магазинів. Ця інтеграція спрощує багато бізнес-процесів, зменшує ручну працю та підвищує якість обслуговування клієнтів. Розглянемо основні аспекти інтеграції з іншими системами в контексті E-commerce.

1. **CRM (Системи Управління Взаємодією з Клієнтами).** Інтеграція з CRM-системами дозволяє інтернет-магазинам ефективно вести облік клієнтів, їхніх покупок та взаємодії з платформою. Це допомагає впроваджувати персоналізовані підходи до кожного клієнта, аналізувати його покупкову історію та надавати знижки або пропозиції, які відповідають його індивідуальним потребам.
2. **ERP (Системи Управління Ресурсами Підприємства).** Інтеграція з ERP-системами дозволяє автоматизувати багато бізнес-процесів, включаючи управління запасами, облік фінансів та управління замовленнями. Вона забезпечує актуальну інформацію про стан складу, фінансову звітність та загальний економічний стан підприємства.
3. **Платіжні Системи та Банківські Платформи.** Інтеграція з різними платіжними системами (наприклад, PayPal, Stripe) та банківськими платформами забезпечує безпеку та швидкість онлайн-транзакцій. Це дозволяє клієнтам здійснювати оплату безпосередньо на сайті, а також спрощує обробку повернень та відшкодувань.
4. **Системи Аналітики та Звітності.** Інтеграція з системами аналітики, такими як Google Analytics, дозволяє збирати та аналізувати дані про поведінку користувачів. Це стає основою для розробки маркетингових стратегій, визначення ефективності кампаній та покращення конверсії.
5. **Соціальні Мережі та Маркетплейси.** Інтеграція з соціальними мережами та зовнішніми маркетплейсами, такими як Amazon чи eBay, розширює аудиторію та забезпечує більше можливостей для продажів. Також, це спрощує обробку замовлень та ведення уніфікованого обліку товарів.
6. **Автоматизація Маркетингу.** Інтеграція з інструментами автоматизації маркетингу, такими як Mailchimp чи HubSpot, дозволяє відправляти автоматизовані розсилки, створювати кампанії електронної пошти та слідкувати за ефективністю маркетингових заходів.

Інтеграція з іншими системами сприяє підвищенню ефективності бізнес-процесів, дозволяє зосередитися на стратегічних завданнях та створює єдиний екосистемний підхід для всього бізнесу.

Безпека та Захист Даних

Забезпечення безпеки та конфіденційності даних клієнтів є пріоритетною задачею E-commerce-систем. Вони використовують шифрування, сертифікати безпеки та інші заходи для захисту важливої інформації під час транзакцій та обробки даних.

Узагальнюючи, E-commerce-системи не тільки дозволяють інтернет-магазинам ефективно працювати, але й сприяють створенню зручного та безпечного середовища для покупців, що є вирішальним чинником в успіху сучасного електронного бізнесу.

1.3 Висновки з розділу 1

В данному розділі було розглянуто важливість та актуальність E-commerce-систем у контексті інтернет-магазинів. Розглядаючи їхню роль та компоненти, ми розуміємо, що ці системи не лише надають технічну підтримку для онлайн-торгівлі, але і стають стратегічним інструментом для розширення аудиторії, оптимізації операцій та покращення взаємодії з клієнтами. Подальший аналіз у розділах магістерської роботи дозволить ретельно розглянути обрану платформу Magento 2 та вивчити новітні технології, спрямовані на поліпшення функціональності інтернет-магазинів.

1.4 Постановка задачі

З урахуванням актуальності та значущості розробки програмного забезпечення для інтернет-магазинів, метою цієї магістерської роботи є розробка модулю, який надасть інтернет-магазинам можливість додавати каталоги продуктів на їх веб-сайти для полегшення пошуку та покупки продуктів, що в свою чергу призведе до підвищення їхньої конкурентоспроможності. Для досягнення цієї мети необхідно вирішити наступні задачі:

1. Розробити зручний та ефективний інтерфейс для додавання та керування каталогами продуктів на веб-сайтах інтернет-магазинів.
2. Реалізувати засоби взаємодії з інтернет-магазинами для забезпечення надання даних про каталоги продуктів через API.
3. Забезпечити можливість користувачам інтернет-магазинів швидко та зручно здійснювати пошук, перегляд та вибір продуктів у доданих каталогах.
4. Реалізувати механізм логування подій та помилок для відслідковування та виправлення потенційних проблем.
5. Забезпечити автоматизацію певних процесів, таких як синхронізація каталогів продуктів з базою даних інтернет-магазину за допомогою крон-завдань.
6. Забезпечити безпеку та захист конфіденційності даних, передаваних через API, та даних користувачів, що зберігаються в системі.

Вирішення цих задач є ключовими для успішної реалізації поставленої мети та досягнення бажаного результату.

.

2 РОЗРОБКА КОНЦЕПЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення цілі розробки ПЗ для підтримки інтернет-магазинів

Формулювання цілей дослідження

Аналіз сучасного стану E-commerce: дослідження ринку E-commerce включатиме аналіз поточного стану, виявлення трендів та основних учасників на ринку. Важливо дізнатися, які продукти або послуги популярні, які технології використовуються та які проблеми існують у сфері E-commerce.

Визначення ключових вимог та потреб користувачів: проведення досліджень серед потенційних користувачів допоможе визначити їхні основні потреби та очікування від програмного забезпечення для підтримки інтернет-магазинів. Це може включати опитування, фокус-групи, аналіз відгуків користувачів тощо.

Для визначення переваг необхідно провести аналіз конкурентів це допоможе визначити їхні сильні та слабкі сторони, а також визначити можливості для вдосконалення. Розуміння того, що робить конкурентів успішними або невдалими, допоможе вирішити, як позиціонувати наше програмне забезпечення на ринку.

Визначення основних завдань, що стоять перед розробниками

На основі отриманих даних розробники повинні збудувати функціональний склад програмного забезпечення, який задовольнятиме потреби користувачів. Це може включати можливості для створення та керування продуктами, управління замовленнями, оплатою, доставкою та багато іншого.

Під час розробки програмного забезпечення, велика увага має бути приділена захисту від вразливостей та забезпеченню надійності системи. Це включає реалізацію механізмів захисту даних користувачів, захист від кібератак та інших загроз безпеці. [1]

Після випуску програмного забезпечення у продакшн розробники мають продовжувати його підтримку та розвиток. Це може включати виправлення помилок, випуск нових функцій, адаптацію до змін вимог користувачів та технологій тощо.

Це роз’яснення допоможе усвідомити всі аспекти визначення цілей та завдань розробки програмного забезпечення для підтримки інтернет-магазинів.

2.2 Проектування архітектури програмного забезпечення

Вибір підходів та методів розробки

У зв'язку з обмеженим обсягом розробки одного модуля та відсутністю необхідності використання юніт-тестів, буде застосовано простий та ефективний підхід до розробки. Основними методами розробки будуть:

- **Використання принципів SOLID**:

  SOLID — це абревіатура, складена з перших літер п'яти базових принципів об'єктно-орієнтованого програмування та дизайну і запропонована Робертом Мартіном у статті 2000 року англ. Design Principles and Design Pattern. [6]

  Принципи SOLID використовують для дизайну та розробки таких програмних систем, які, з великою ймовірністю, зможуть тривалий час розвиватися, розширятися і підтримуватися.

  Як зазначає сам автор, запропоновані ним принципи не є «істиною в останній інстанції», правилами, або ж суворими законами. У нього немає доказів, що вони завжди працюють, або ж, що їх слід неухильно дотримуватись. Проте, вони були сформульовані на основі спостережень і зазвичай допомагають уникнути проблем.

  При проектуванні архітектури програмного забезпечення для модуля на базі Magento 2, важливо керуватися принципами SOLID. Ці принципи допоможуть забезпечити гнучкість, розширюваність та підтримку модульності коду.

- **Принцип KISS (Keep It Simple, Stupid)**:

KISS («нехай буде просто, дурню» або більш ввічливий — «роби коротше і простіше») — процес і принцип проектування, при якому простота системи декларується як основна мета та/або цінність. Можна розглядати, як узагальнення фізичного закону «відкрита система тяжіє до мінімуму ентальпії». Принцип KISS базується на твердженні, що більшість систем працюють краще, якщо вони прості в користуванні. Виходячи з цього, простота повинна бути головною метою в області дизайну і потрібно намагатися уникати непотрібних складнощів під час проектування. [7]

Старання дотримуватися простоти в проектуванні та реалізації програмного забезпечення. Це допоможе уникнути надмірної складності та забезпечити зрозумілість коду для інших розробників.

- **Принцип DRY (Don't Repeat Yourself)**:

У програмній інженерії, «Don't repeat yourself» (DRY, укр. не повторюйся) — принцип розробки програмного забезпечення, що направлений на уникнення дублювання інформації будь-якого вигляду (наприклад, програмний код чи текст інтерфейсу користувача). Принцип був сформульований Енді Хантом та Дейвом Томасом в їх книзі The Pragmatic Programmer[15] наступним чином: «Будь-яка інформація повинна мати єдине, однозначне, авторитетне представлення в системі». [8]

Уникайте дублювання коду, зосереджуйтеся на створенні загальних функцій та компонентів, які можуть бути використані в різних частинах модуля. Це допоможе підтримувати код в чистому та організованому стані.

- **Використання ООП (Об'єктно-орієнтоване програмування)**:

Об'є́ктно-орієнто́ване програмува́ння (ООП, іноді об'єктно-зорієнто́ване програмува́ння; англ. Object-oriented programming, OOP) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. [9]

Використання об'єктно-орієнтованих принципів при проектуванні архітектури модуля дозволить створити логічну структуру з класами, які

представляють конкретні об'єкти або сутності, та методами, що оперують над ними.

- **Відмова від юніт-тестів**:

У зв'язку з обмеженнями на використання юніт-тестів, які накладається на проект, рекомендується зосередитися на внутрішньому тестуванні, наприклад, за допомогою вручного тестування або використанням інших методів, таких як інтеграційні тести чи тести на рівні системи.

Визначення структури програмного забезпечення та його компонентів

У додатку було вирішено створити наступні структури і компоненти:

1. **API**: Цей компонент буде відповідати за взаємодію з зовнішнім API. Сюди включені класи та методи, які відправляють та отримують дані з зовнішнього сервісу, обробляють запити та повертають результати.
2. **Block**: Блоки використовуються для відображення інформації на сторінках інтернет-магазину. Вони можуть містити HTML-код, PHP-логіку та виклики до моделей та сервісів для отримання даних.
3. **Controller**: Контролери відповідають за обробку вхідних запитів від користувачів та взаємодію з іншими компонентами програмного забезпечення. Вони приймають HTTP-запити, обробляють їх та визначають, які дії потрібно виконати.
4. **etc**: Цей каталог містить конфігураційні файли програмного забезпечення, такі як XML-файли конфігурації для опису маршрутів, налаштувань модуля та інших параметрів.
5. **Logger**: Компонент журналювання використовується для запису журналу подій, помилок та інших важливих повідомлень. Він може використовуватися для відстеження діагностичних повідомлень та відладки програмного забезпечення.
6. **Model**: Моделі відповідають за доступ до даних та бізнес-логіку програмного забезпечення. Вони використовуються для отримання та

збереження інформації в базі даних, а також для виконання різних операцій з цими даними.

7. **Service**: Сервіси містять бізнес-логіку, яка не відноситься безпосередньо до моделей або контролерів. Вони використовуються для виконання різних операцій, обробки даних та взаємодії з іншими компонентами програмного забезпечення.
8. **View**: Цей компонент відповідає за відображення інтерфейсу користувача. Він містить шаблони HTML, CSS та JavaScript, які використовуються для створення зовнішнього вигляду сторінок інтернет-магазину.
9. **Cron**: Компонент крону використовується для планування та виконання автоматичних завдань у визначені часи. Він може використовуватися для регулярного виконання фонових процесів, які потребуються для роботи модуля

2.3 Визначення вимог до функціональності та користувацького інтерфейсу

Аналіз потреб користувачів.

Для визначення потреб користувачів необхідно провести досліджень для з'ясування потреб та очікувань користувачів щодо інтеграції з зовнішнім API. Розуміння, які функції або можливості вони очікують від модуля, допоможе сформулювати вимоги до його функціональності та інтерфейсу.

Основними сценаріями використання модуля будуть: реєстрація нових користувачів на зовнішньому сервісі, створення їх облікових записів та отримання доступу до журналів продуктів через зовнішній API

Формулювання вимог до функціональності та зручності використання.

- **Реєстрація користувачів на зовнішньому сервісі**: Модуль повинен надавати можливість користувачам реєструватися на зовнішньому сервісі за допомогою зазначених ним даних під час реєстрації на нашому веб сайті.

- **Створення облікових записів та надання доступу до журналів**: Після реєстрації нового користувача, модуль повинен автоматично створювати його обліковий запис на зовнішньому сервісі та надавати доступ до журналів через вбудований iframe на сторінці.
- **Зручний інтерфейс користувача**: Інтерфейс модуля повинен бути зрозумілим та інтуїтивно зрозумілим для користувачів. Мінімізація кількості кроків для реєстрації та отримання доступу до журналів допоможе покращити користувацький досвід.
- **Інтеграція з зовнішнім API**: Модуль повинен забезпечити стабільну та надійну інтеграцію з зовнішнім API для забезпечення взаємодії з зовнішнім сервісом та передачі необхідних даних.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір оптимальних інструментів для реалізації поставлених завдань

При виборі технологічного стеку та інструментів розробки для проекту, ключовою частиною є вибір правильних інструментів для взаємодії з API. У даному випадку, бібліотека GuzzleHttp[11] виступає як основний інструмент для виконання цієї задачі.

GuzzleHttp\Client - це основний клас бібліотеки GuzzleHttp, який використовується для виконання HTTP-запитів до зовнішнього API. Він надає зручний і простий інтерфейс для взаємодії з віддаленими серверами через HTTP.

GuzzleHttp\Response - після виконання HTTP-запиту, об'єкт класу Response представляє відповідь від сервера. Він містить всю інформацію про HTTP-статус, заголовки та тіло відповіді, яка може бути оброблена подальше в коді.

GuzzleHttp\Factory - цей клас є фабричним методом, який використовується для створення екземплярів класу Client та інших об'єктів GuzzleHttp з певними налаштуваннями. Він дозволяє створювати об'єкти GuzzleHttp з різними конфігураціями без необхідності в явному створенні та налаштуванні кожного об'єкта окремо.

Обрана бібліотека GuzzleHttp забезпечить зручний та ефективний спосіб взаємодії з API та обробки відповідей від віддалених серверів. Її широкий функціонал та простий інтерфейс дозволять ефективно виконувати завдання, пов'язані зі зв'язком з API в рамках розробки вашого проекту.

Також важливо врахувати можливості, які вже має Magento 2 [2]. Так як ми намір використовувати вбудовані функції Magento 2 для створення своїх логерів, кронів та інших компонентів. [5]

3.2 Процес розробки та тестування ПЗ

Створення базової структури модуля

Для створення каркасу модуля необхідно створити файли registration.php та module.xml для реєстрації модуля. [12] Файл registration.php можна взяти з офіційного сайту та не додавати нової логіки:

```
<?php

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

При створенні module.xml необхідно додати до шаблону залежності від маджento модулів [13], котрі ми будемо використовувати в файлах модуля:

```
<?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="Vendor_ZoomCatalog">
    <sequence>
      <module name="Magento_Customer" />
      <module name="Magento_Store" />
    </sequence>
  </module>
</config>
```

Налаштування API

Для цього було створено ApiManagementInterface з методом getApiData:

```
<?php
declare(strict_types=1);

namespace Vendor\ZoomCatalog\Api;
```

```
interface ApiManagementInterface
{
    /**
     * @param $apiRequestEndpoint
     * @param $params
     * @param $method
     * @return array
     */
    public function getApiData($apiRequestEndpoint, $params, $method): array;
}
```

Наступним кроком було створення моделі для нашого інтерфейсу:

```
<?php
declare(strict_types=1);

namespace Vendor\ZoomCatalog\Model;

use Vendor\ZoomCatalog\Api\ApiManagementInterface;
use Vendor\ZoomCatalog\Service\AdminConfigurations;
use Vendor\ZoomCatalog\Service\ZoomCatalogApiService;
use Magento\Customer\Model\Session;
use Magento\Framework\Webapi\Rest\Request;

class ApiManagement implements ApiManagementInterface
{
    public function __construct(
        private ZoomCatalogApiService $apiService,
        private AdminConfigurations $adminConfigurations,
        private Session $customerSession
    ){}
```

```
    public function getApiData($apiRequestEndpoint, $params, $method = Request::HTTP_METHOD_POST): array
    {
        $response = $this->apiService->doRequest($apiRequestEndpoint, $params, $method);
        $responseContent = $response->getBody()->getContents();

        return  \Safe\json_decode($responseContent, true);
    }

    public function getAccessToken(): array
    {
        $adminConfigurations = $this->adminConfigurations;
        $apiAuthRequestEndpoint = $adminConfigurations->getApiAuthAuthorizeRequestEndpoint();
        $authParams = [
            'json' => [
                'client_id' => $adminConfigurations->getClientId(),
                'client_secret' => $adminConfigurations->getClientSecret(),
                'grant_type' =>'authorization_code',
                'for_username' => $this->customerSession->getCustomer()->getName()
            ]
        ];

        return $this->getApiData($apiAuthRequestEndpoint, $authParams);
    }

    public function getCatalogs(): array
    {
```

```
    $accessToken = $this->getAccessToken();
    $catalogsParams = [
        'headers' => [
            'Authorization' => $accessToken['access_token']
        ]
    ];

    return $this->getApiData(
        $this->adminConfigurations->getApiCatalogsRequestEndpoint(),
        $catalogsParams,
        Request::HTTP_METHOD_GET
    );
}

public function getCustomCatalogs(): array
{
    $accessToken = $this->getAccessToken();
    $customCatalogsParams = [
        'headers' => [
            'Authorization' => $accessToken['access_token']
        ]
    ];

    return $this->getApiData(
        $this->adminConfigurations->getApiCustomCatalogsRequestEndpoint(),
        $customCatalogsParams,
        Request::HTTP_METHOD_GET
    );
}
```

```
public function getFlyers(): array
{
    $accessToken = $this->getAccessToken();
    $catalogsParams = [
        'headers' => [
            'Authorization' => $accessToken['user_token']
        ]
    ];

    return $this->getApiData(
        $this->adminConfigurations->getApiFlyersRequestEndpoint(),
        $catalogsParams,
        Request::HTTP_METHOD_GET
    );
}

public function getSessId(): array
{
    $accessToken = $this->getAccessToken();
    $catalogsParams = [
        'headers' => [
            'Authorization' => $accessToken['user_token']
        ]
    ];

    return $this->getApiData(
        $this->adminConfigurations->getApiAuthAutologinRequestEndpoint(),
        $catalogsParams,
        Request::HTTP_METHOD_GET
    );
```

```
    }

    public function getFlyersCreateUrl(): string
    {
        return $this->adminConfigurations->getApiFlyersCreateRequestEndpoint() .
$this->getSessId()['uri'];
    }
}
```

Цей код відображає клас **ApiManagement**, який відповідає за управління API з Magento 2. Основні функції цього класу включають:

1. **getApiData**: Цей метод виконує запит до API і повертає дані у вигляді асоціативного масиву.
2. **getAccessToken**: Отримує токен доступу для авторизації користувача.
3. **getCatalogs**: Отримує каталоги з API.
4. **getCustomCatalogs**: Отримує спеціальні каталоги з API.
5. **getFlyers**: Отримує літачки з API.
6. **getSessId**: Отримує ідентифікатор сесії для автовходу.
7. **getFlyersCreateUrl**: Повертає URL для створення флаєрів.

Цей клас використовує інші служби, такі як **ZoomCatalogApiService**, **AdminConfigurations**, і об'єкт сесії користувача Magento.

Загалом, цей код дозволяє виконувати різні операції з API, такі як отримання каталогів, флаєрів тощо, з Magento 2.

Розробка блоків

Для отримання даних у view моделі був створений блок, котрий буде передавати необхідну інформацію:

```
<?php
declare(strict_types=1);
```

```
namespace Vendor\ZoomCatalog\Block;

use Vendor\ZoomCatalog\Model\ApiManagement;
use Magento\Customer\Model\SessionFactory;
use Magento\Framework\Message\ManagerInterface;
use Magento\Framework\View\Element\Template;
use Magento\Framework\View\Element\Template\Context;

class DisplayBlock extends Template
{
    public function __construct(
        private Context $context,
        private ApiManagement $apiManagement,
        private ManagerInterface $messageManager,
        private SessionFactory $customerSession
    ) {
        parent::__construct($context);
    }

    public function getCatalogs(): ?array
    {
        try {
            $catalogs = $this->apiManagement->getCatalogs();
            if (empty($catalogs['catalogs'])) {
                throw new \Exception();
            }
            return $catalogs;
        } catch (\Exception $exception) {
            $this->messageManager->addErrorMessage(__('Something went wrong, try a little bit later'));
```

```
            return null;
        }
    }

    public function getCustomCatalogs(): ?array
    {
        try {
            $customCatalogs = $this->apiManagement->getCustomCatalogs();
            if (empty($customCatalogs['catalogs'])) {
                throw new \Exception();
            }
            return $customCatalogs;
        } catch (\Exception $exception) {
            $this->messageManager->addErrorMessage(__('Something went wrong, try a little bit later'));
            return null;
        }
    }

    public function getFlyers(): ?array
    {
        try {
            $flyers = $this->apiManagement->getFlyers();
            if (empty($flyers['flyers'])) {
                throw new \Exception();
            }
            return $flyers;
        } catch (\Exception $exception) {
            $this->messageManager->addErrorMessage(__('Something went wrong, try a little bit later'));
```

```
            return null;
        }
    }

    public function isCustomerLoggedIn(): bool
    {
        return (bool)$this->customerSession->create()->isLoggedIn();
    }

    public function getSessId(): ?string
    {
        try {
            $sessionId = $this->apiManagement->getSessId();
             return substr($sessionId['uri'], 1);
        } catch (\Exception $exception) {
            $this->messageManager->addErrorMessage(__('Something went wrong, try a little bit later'));
            return null;
        }
    }

    public function getFlyersCreateUrl(): ?string
    {
        try {
            return $this->apiManagement->getFlyersCreateUrl();
        } catch (\Exception $exception) {
            $this->messageManager->addErrorMessage(__('Something went wrong, try a little bit later'));
            return null;
        }
```

```
    }
}
```

Розробка контролерів

Для обробки запитів було створено два контролери Index.php:

```
<?php
declare(strict_types=1);

namespace Vendor\ZoomCatalog\Controller\Index;

use Magento\Framework\App\Action\HttpGetActionInterface;
use Magento\Framework\View\Result\PageFactory;
use Magento\Framework\View\Result\Page;

class Index implements HttpGetActionInterface
{
    public function __construct(
        protected PageFactory $resultPageFactory
    ) {}

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

Та Flyers.php:

```
<?php
declare(strict_types=1);

namespace Vendor\ZoomCatalog\Controller\Index;
```

```
use Magento\Framework\App\Action\HttpGetActionInterface;
use Magento\Framework\View\Result\PageFactory;
use Magento\Framework\View\Result\Page;

class Flyers implements HttpGetActionInterface
{
   public function __construct(
      protected PageFactory $resultPageFactory
   ) {}

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

Ці контролери виконуються базову задачу – створення пустих сторінок, котрі будуть наповнені за допомогою view моделі.

Створення файлів конфігурації

Для налаштувань в модулі були створені наступні файли у директорії etc:

- adminhtml/system.xml в цьому файлі були прописані форми для налаштувань в адміністративній панелі.
- frontend/routes.xml використовується для маршрутизації роутів у фронтенд частині.
- frontend/config.xml в цьому файлі були прописані стандартні знення для нашої форми в адмін панелі, такі як: статус модулю, uri, та всі ендпоінти.
- frontend/crontab.xml використовується планування крон-завдань.

- frontend/di.xml був створений для впровадження залежностей. Також в цьому файлі був був створений preference що допомогає використовувати інтрефейс замість моделі в конструкторах та створений logger за допомогою інструментів наданими мадженто 2.

Налаштування логеру, а саме тип помилок для логування та місцезнаходження файлу з логами вказані в Logger/Handler.php:

```
<?php
declare(strict_types=1);

namespace Vendor\ZoomCatalog\Logger;

use Monolog\Logger;

class Handler extends \Magento\Framework\Logger\Handler\Base
{
    /**
     * Logging level
     * @var int
     */
    protected $loggerType = Logger::NOTICE;

    /**
     * File name
     * @var string
     */
    protected $fileName = '/var/log/zoom_catalog.log';
}
```

Розробка допоміжних сервісів

Важливою частиною цього модуля є класи-сервіси такі як:

- AdminConfigurations
- ZoomCatalogApiService

AdminConfigurations – це класс котрий виконує роль сервіс провайдеру, так як в ньому існують всі методи для отримання конфігурацій нашого модуля з адміністративної панелі.

ZoomCatalogApiService – цей клас сторений для створення запитів на API, і оброблює та логує статуси і повідомлення з відповідей (response).

Налаштування представлень

Для цього були створені лаяути zoomcatalog_index_flyers.xml та zoomcatalog_index_index.xml в котрих були вказані зв’язки контролера з блоком та шаблонів:

```
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="1column"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_configuration.xsd">
    <referenceContainer name="content">
      <block class="Vendor\ZoomCatalog\Block\DisplayBlock"
          name="zoom_catalog_api_display"
          template="Vendor_ZoomCatalog::zoomcatalog/items.phtml" />
    </referenceContainer>
</page>
```

В шаблонах view/frontend/templates/zoomcatalog/items.phtml та view/frontend/templates/zoomcatalog/flyers.phtm були створені необхідні елементи інтерфейсу за допомогою HTML та CSS, виконується формування каталогів та

флаєрів. Також тут формуються посилання котрі дозволяють створювати приватні кабінети користувачів та отримувати доступ до каталогів.

Нижче наведений вміст шаблону для каталогів:

```
<?php
/**
 * @var Vendor\ZoomCatalog\Block\DisplayBlock $block
 */
$catalogs = $block->getCatalogs();
$customCatalogs = $block->getCustomCatalogs();
$sessId = $block->getSessId();
?>
<div class="zoom-catalog-wrapper">
    <?php if (isset($catalogs)): ?>
        <h2>Zoom Catalog</h2>
        <?php foreach ($catalogs['catalogs'] as $catalog): ?>
            <div class="zoom-catalog-item">
                <a target="_blank" href="<?= $catalog['url']; ?>">
                    <img src="<?= $catalog['images']['small']; ?>" alt="<?= $catalog['title'];?>">
                </a>
            </div>
        <?php endforeach; ?>
    <?php endif; ?>

    <?php if (isset($customCatalogs) && $block->isCustomerLoggedIn()): ?>
        <h2>Zoom Custom Catalog</h2>
        <?php foreach ($customCatalogs['catalogs'] as $customCatalog): ?>
            <div class="zoom-catalog-item">
```

```
                <a target="_blank" href="<?= $customCatalog['personalize'] . '&' . $sessId;?>">
                    <img src="<?= $customCatalog['images']['small']; ?>" alt="<?= $customCatalog['title'];?>">
                </a>
            </div>
        <?php endforeach; ?>
    <?php endif; ?>
</div>
```

Після реалізації компонентів програмного забезпечення була завершена структура програми(**Помилка! Джерело посилання не знайдено.**). Було використовано рекомендований підхід з офіційної документації. [14]

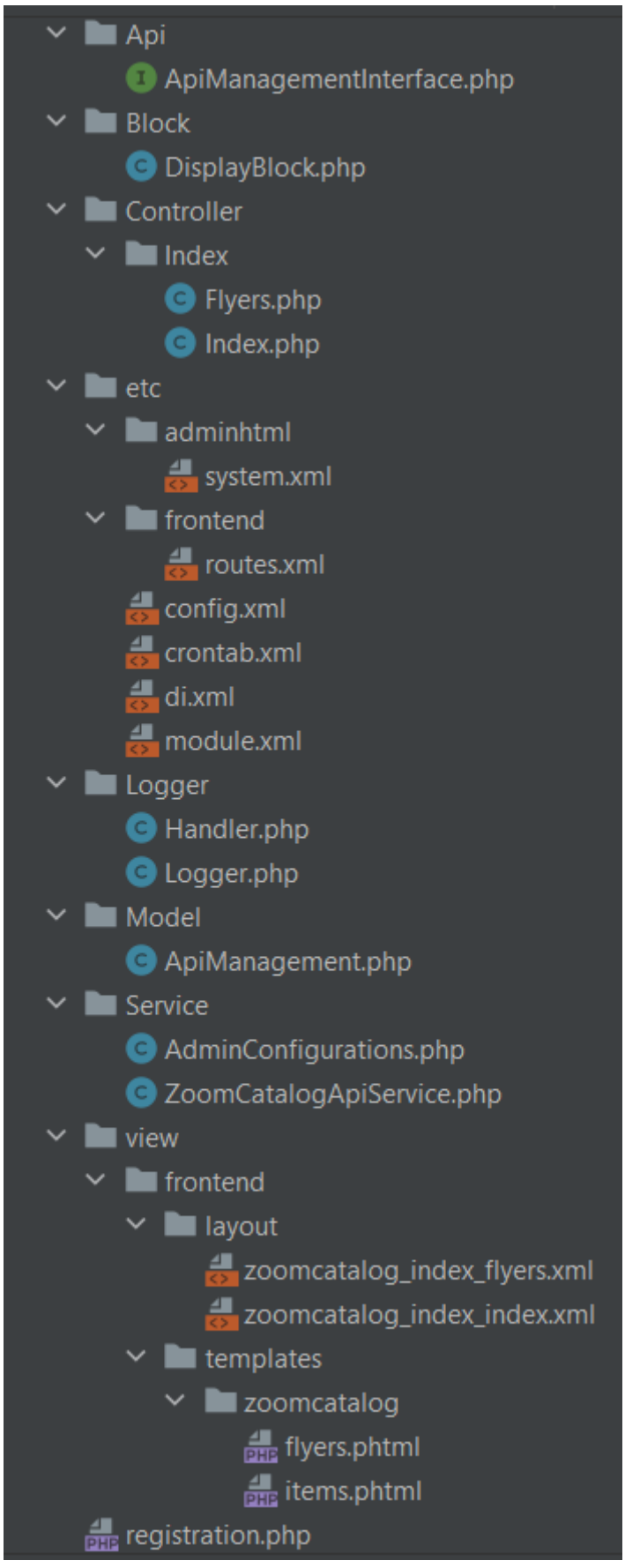

Рис. 1 – Структура створеного модулю.

3.3 Тестування додатку

У цьому розділі будуть представлені методи тестування та забезпечення якості розробки модулю.

Налаштування адміністративних параметрів модулю (**Помилка! Джерело посилання не знайдено.**):

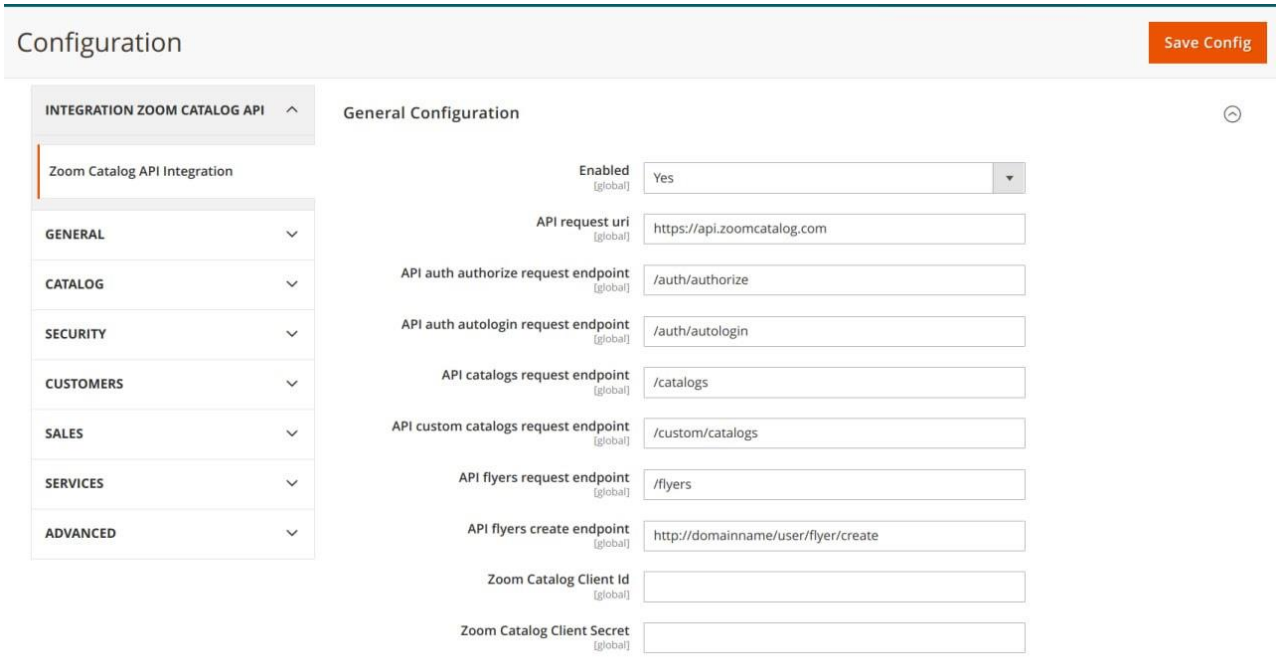

Рис. 2 – Налаштування модулю в адміністративній панелі. Скріншот з адміністративної панелі, де показані всі налаштування модулю, включаючи налаштування API, параметри відображення та інші конфігурації.

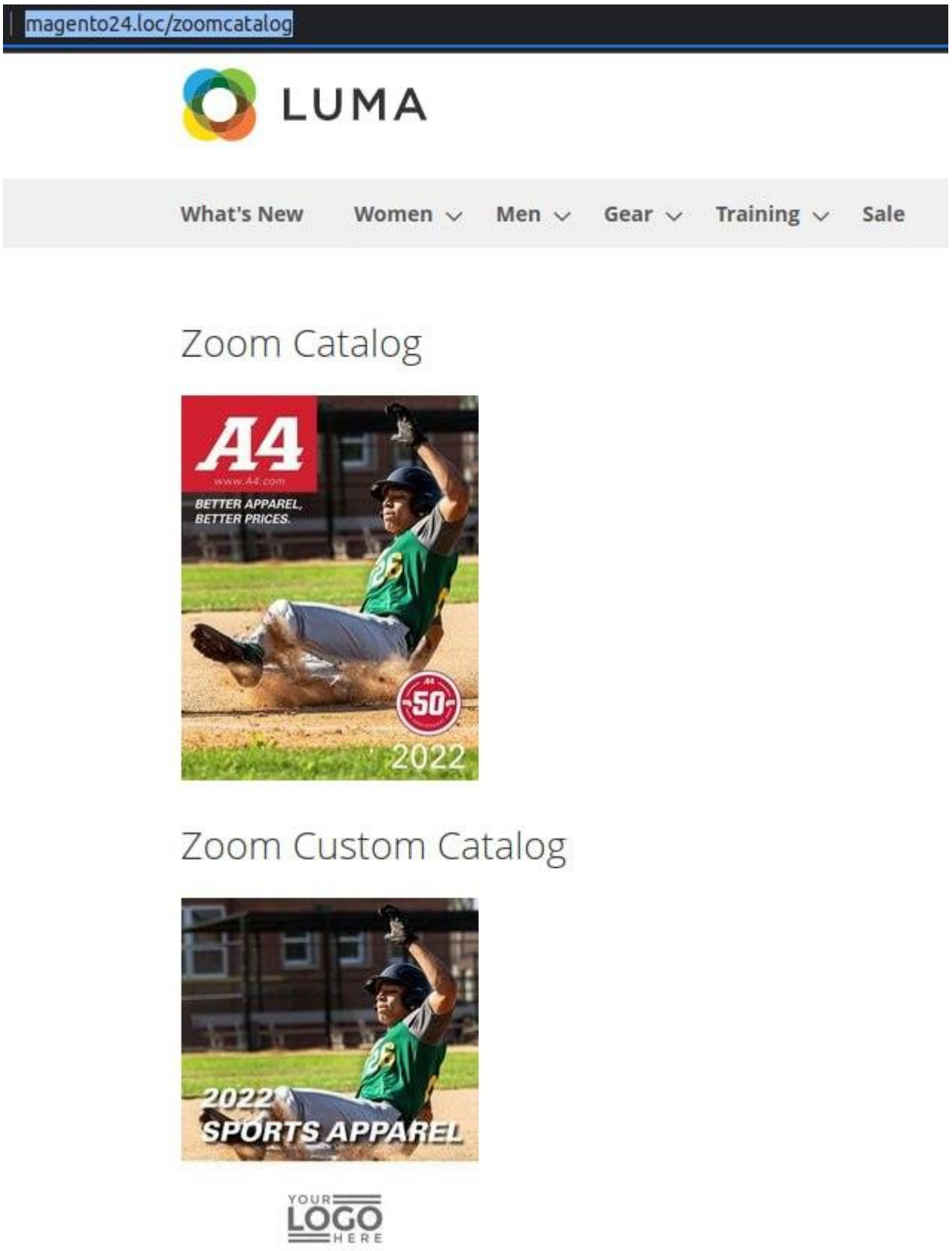

Рис. 3 – Каталог з журналами для користувача.

Скріншот першої сторінки, на якій відображається каталог з журналами. Цей екран демонструє, як користувач може переглядати доступні журнали та їхні сторінки (**Помилка! Джерело посилання не знайдено.**).

Після обрання флаєрів або каталогів – буде відкрита сторінка зовнішнього ресурсу, на котрій користувач може побачити необхідні журнали (Рис. 4). Адміністратор також має можливисть приховати журнали для певної групи покупців.

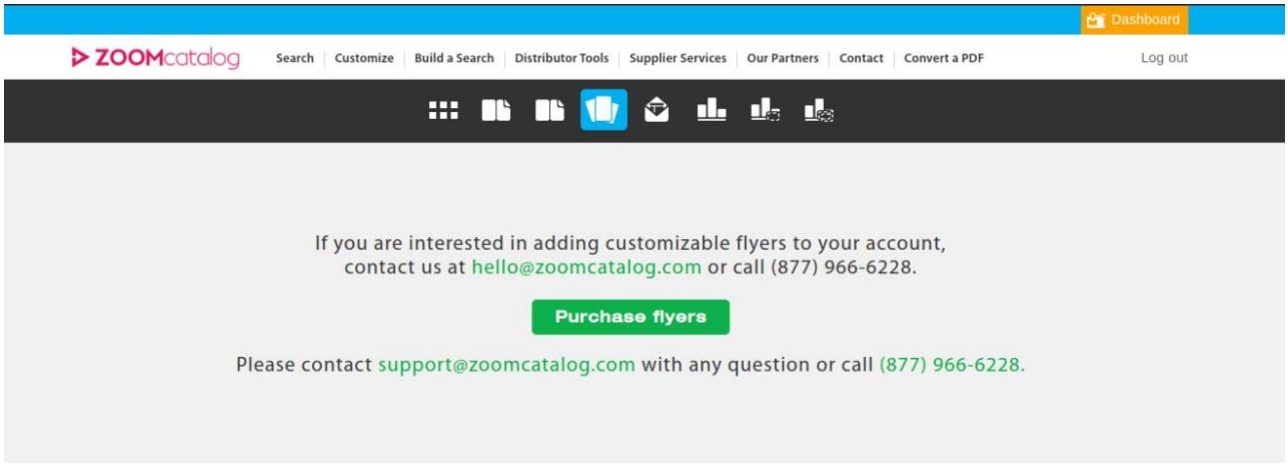

Рис. 4 – Сторінка зовнішнього ресурсу Zoom Catalog [10]

Ці скріншоти допоможуть продемонструвати коректну роботу модулю та його взаємодію з адміністративною панеллю та функціоналом веб-сайту

3.4 Впровадження та підтримка програмного забезпечення

Організація підтримки та оновлень після запуску в експлуатацію

Організація підтримки програмного забезпечення включає в себе наступні аспекти:

Відсутність документації:

Для забезпечення простоти використання та підтримки модулю, всі необхідні інструкції та описи були додані безпосередньо в файл налаштувань system.xml. Це дозволяє забезпечити доступ до всієї необхідної інформації, не переймаючись про роздільні документи.

Встановлення модулю:

Після розробки та тестування, модуль може бути легко встановлений у Magento проекті. Для цього його потрібно розмістити відповідно до стандартів Magento у папці app/code. Після встановлення модуль готовий до використання.

Ці кроки допоможуть забезпечити ефективне впровадження програмного забезпечення та його подальшу підтримку після запуску в експлуатацію.

ВИСНОВКИ

В результаті виконання роботи розроблений та впроваджений програмний модуль Magento 2 для створення інтерактивних каталогів продуктів інтернет-магазинів на основі існуючих продуктів вебсайту. Розробка використовує передові технології, такі як PHP 8.1 та MySQL, і заснована на модульній структурі, що полегшує інтеграцію з існуючими системами.

Важливим аспектом є не лише швидкість та зручність для кінцевого користувача, а й можливість гнучких налаштувань у адміністративній панелі. Ці налаштування спрощують підтримку модуля навіть у випадку змін в API. Крім того, інтуїтивно зрозумілий інтерфейс для адміністраторів полегшує процес налаштування та управління модулем.

Зокрема, впровадження модульної архітектури, яка дозволяє розширювати функціональність за необхідністю, сприяє простоті обслуговування та адаптації до змін потреб бізнесу. Враховуючи вищевказані аспекти, можна визначити, що розробка відповідає високим стандартам електронної комерції та є конкурентоздатною на ринку.

Таким чином, розроблене програмне забезпечення, побудоване на основі Magento 2, не лише відповідає вимогам сучасного електронного бізнесу, а й надає інтернет-магазинам конкурентні переваги.

ПЕРЕЛІК ПОСИЛАНЬ

1. General Data Protection Regulation URL: https://gdpr-info.eu/ (дата звернення : 29.06.2023)
2. Magento, now Adobe Commerce | eCommerce Software. URL: https://business.adobe.com/products/magento/magento-commerce.html (дата звернення : 04.07.2023)
3. Крегул Ю. Правове регулювання міжнародної електронної комерції. Зовнішня торгівля: економіка, фінанси, право. – 2018. – № 2. – С. 136 – 147.
4. Gazieva L.R. The impact of e-commerce on the digital economy. L. R. International Conference on Finance, Entrepreneurship and Technologies in Digital Economy, 2021. – P. 121 – 126.
5. Magento 2 System requirements. URL: https://experienceleague.adobe.com/docs/commerce-operations/installation-guide/system-requirements.html (дата звернення : 20.12.2023).
6. SOLID (об'єктно-орієнтоване програмування). URL: https://en.wikipedia.org/wiki/SOLID (дата звернення : 20.12.2023).
7. KISS. URL: https://en.wikipedia.org/wiki/KISS_principle (дата звернення : 20.12.2023).
8. DRY. URL: https://en.wikipedia.org/wiki/Don%27t_repeat_yourself (дата звернення : 20.12.2023).
9. Об'єктно-орієнтоване програмування. URL: https://en.wikipedia.org/wiki/Object-oriented_programming (дата звернення : 20.12.2023).
10. Zoom Catalog API. URL: https://doc-site.zoomcatalog.com/docs/about/intro (дата звернення : 20.12.2023).
11. Guzzle Documentation. URL: https://docs.guzzlephp.org/en/stable/ (дата звернення : 20.12.2023).

12. Magento 2 Register a component. URL: https://developer.adobe.com/commerce/php/development/build/component-registration (дата звернення : 20.12.2023).

13. Component Load Order. URL: https://developer.adobe.com/commerce/php/development/build/component-load-order (дата звернення : 20.12.2023).

14. Create your component file structure. URL: https://developer.adobe.com/commerce/php/development/build/component-file-structure

15. Hunt D. T. A. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional, 2019.– 352 p.