Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ПЛАТІЖНОГО ПЛАГІНУ ДЛЯ CMS MAGENTO З ВИКОРИСТАННЯМ PHP, JAVASCRIPT ТА БІБЛІОТЕКИ I18 ДЛЯ ПІДТРИМКИ ПЕРЕКЛАДІВ

DEVELOPMENT OF A PAYMENT PLUGIN FOR THE MAGENTO CMS USING PHP, JAVASCRIPT AND THE I18 LIBRARY FOR TRANSLATION SUPPORT

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Прокоф'єв Максим Олександрович

Керівник:
Вознюк Анастасія Вадимівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«*20*» *12* 20*24* р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Прокоф'єву Максиму Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка платіжного плагіну для CMS magento з використанням PHP, JavaScript та бібліотеки i18 для підтримки перекладів»*
Керівник роботи: *асистент Вознюк А. В.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*

3. Вихідні дані до роботи: *Magento 2, PHP, JavaScript, Composer, PHPUnit, MySQL, Git, CSS, методичні вказівки до виконання кваліфікаційної роботи бакалавра*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз електронної комерції та порівняння платформ, формування функціональних вимог, вибір технологій, реалізація плагіна, тестування, інтеграція з БД, створення інсталяційного-пакета, створення документації.*

5. Перелік графічного матеріалу: *11 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Вознюк А. В.*		
Специфікація вимог до розробленої системи	*Вознюк А. В.*		
Розробка об`єкта проектування	*Вознюк А. В.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,81 %		
Академічна доброчесність	*Вознюк А. В.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об`єкту проектування	до 01.03.2025 р.	Виконано
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Виконано
4	Розробка функціонально-структурної схеми роботи об`єкта проектування та проектування бази даних	до 29.03.2025 р.	Виконано
5	Практична реалізація об`єкта проектування та розробка бази даних	до 26.04.2025 р.	Виконано
6	Тестування та налагодження об`єкта проектування	до 03.05.2025 р.	Виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Виконано

Здобувач вищої освіти Максим ПРОКОФ`ЄВ

Керівник кваліфікаційної роботи Анастасія ВОЗНЮК

АНОТАЦІЯ

Прокоф’єв М. О. Розробка платіжного плагіну для CMS Magento з використанням PHP, JavaScript та бібліотеки i18 для підтримки перекладів. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі здійснено аналіз сучасного стану електронної комерції, досліджено особливості платформи Magento 2, а також сформульовано цілі та завдання роботи.

У другому розділі визначено функціональні та нефункціональні вимоги до розроблюваного плагіна, обґрунтовано вибір технологій та інструментів, описано структуру системи та підходи до реалізації.

У третьому розділі подано практичну реалізацію плагіна, результати тестування, опис бази даних, створення інсталяційного пакета та документації до програмного продукту.

У висновках узагальнено результати виконаної роботи, обґрунтовано ефективність розробленого рішення, наведено перспективи його подальшого розвитку.

Ключові слова: Magento 2, платіжний плагін, PHP, JavaScript, i18n, Composer, API, електронна комерція.

ABSTRACT

Prokofiev M. Development of a Payment Plugin for the Magento CMS Using PHP, JavaScript and the i18 Library for Translation Support. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references.

The first chapter analyzes the current state of e-commerce, explores the features of the Magento 2 platform, and defines the goals and objectives of the study.

The second chapter outlines the functional and non-functional requirements for the developed plugin, justifies the selection of technologies and tools, and describes the system architecture and implementation approaches.

The third chapter presents the practical implementation of the plugin, testing results, database structure, installation package creation, and accompanying documentation.

The conclusions summarize the work completed, justify the effectiveness of the developed solution, and suggest directions for further development.

Keywords**:** Magento 2, payment plugin, PHP, JavaScript, i18n, Composer, API, e-commerce.

ЗМІСТ

ВСТУП

Актуальність теми. Електронна комерція є однією з найбільш динамічних галузей сучасної цифрової економіки. За останнє десятиріччя відбулося суттєве зростання обсягів онлайн-продажів, що стало можливим завдяки швидкому розвитку інтернет-технологій, глобальній цифровізації та зміні споживчої поведінки. У зв’язку з цим онлайн-магазини та торговельні платформи мають забезпечувати не лише зручний інтерфейс, а й ефективні та безпечні засоби обробки транзакцій. Особливо це актуально в умовах жорсткої конкуренції та постійно зростаючих очікувань з боку кінцевих споживачів щодо швидкості, зручності та надійності процесу оплати. Платіжна інфраструктура є критичним компонентом будь-якої системи електронної комерції, що визначає як успішність завершення покупки, так і загальний рівень довіри до платформи. При цьому особливо важливим є дотримання міжнародних стандартів інформаційної безпеки, які регламентують вимоги до обробки, зберігання та передачі платіжних даних. Успішне виконання цих вимог вимагає не лише глибоких знань архітектури конкретної платформи, а й професійного підходу до розробки програмного забезпечення, яке взаємодіє з платіжними шлюзами.

Однією з найпопулярніших платформ для електронної торгівлі з відкритим вихідним кодом є Magento 2. Вона поєднує у собі потужні інструменти для масштабування, високу гнучкість і можливість глибокої кастомізації. Завдяки модульній архітектурі та підтримці сучасних технологій Magento дозволяє реалізовувати спеціалізовані рішення для інтеграції з платіжними системами, розширювати функціональність магазину відповідно до вимог конкретного бізнесу, підтримувати декілька валют, мов і методів доставки. Водночас складність архітектури платформи та вимоги до якості інтеграційних рішень ставлять високі вимоги до розробника.

Незважаючи на широкий спектр вже існуючих платіжних модулів, багато з них мають обмеження у кастомізації, слабку інтеграцію з локальними платіжними сервісами, або ж не відповідають оновленим вимогам безпеки. Це

створює потребу у створенні нових, більш адаптованих до реалій українського та міжнародного ринку рішень, здатних забезпечити одночасно надійність, зручність і технічну досконалість. Розробка такого модуля є не лише практичною задачею для забезпечення ефективної платіжної логіки у межах конкретного інтернет-магазину, а й репрезентує сучасні підходи до побудови масштабованих і безпечних систем електронної комерції в цілому.

Мета роботи полягає в аналізі наявних технологій платіжної інтеграції та безпосередній розробці спеціалізованого платіжного плагіна для Magento 2, який би забезпечував безпечну і стабільну обробку фінансових транзакцій та відповідав сучасним вимогам як з боку бізнесу, так і з боку кінцевих користувачів.

Об’єкт кваліфікаційної роботи бакалавра – процес інтеграції платіжних систем у вебплатформи електронної комерції.

Предмет роботи – методи та засоби розробки платіжного плагіна для Magento 2 з урахуванням вимог безпеки, надійності та зручності для користувачів.

Для досягнення цієї мети були визначені наступні завдання:

– проаналізувати поточний стан електронної комерції, визначити її сучасні виклики та перспективи, а також провести аналіз платформи Magento 2 шляхом порівняння з альтернативними рішеннями, такими як Shopify і WooCommerce;

– сформулювати функціональні та нефункціональні вимоги до програмного забезпечення з урахуванням архітектури Magento 2, стандартів безпеки та потреб цільових користувачів – адміністраторів магазину та покупців;

– обґрунтувати вибір технологій, мов програмування PHP та JavaScript, інструментів Composer та PHPUnit і архітектурних рішень необхідних для реалізації надійного, масштабованого плагіна;

– реалізувати основні компоненти платіжного плагіна: логіку ініціації транзакцій, взаємодію з API платіжного шлюзу, обробку вебхука,

перенаправлення користувача, адміністративну панель керування параметрами модуля;

– провести модульне та мануальне тестування розробленого плагіна у середовищі Magento 2 з метою перевірки його стабільності, коректності роботи всіх сценаріїв оплати та відповідності вимогам функціоналу;

– інтегрувати розроблений модуль з базою даних Magento 2 відповідно до стандартів платформи: використовуючи репозиторії, ресурсні моделі, поля розширення та індекси, що забезпечують продуктивність і збереження цілісності даних;

– створити інсталяційний пакет плагіна у вигляді Composer-пакета, який дозволить легко підключати модуль до сторонніх проєктів, автоматично реєструвати його в системі Magento та оновлювати в майбутньому;

– підготувати супровідну документацію з описом системних вимог, інструкціями з встановлення, налаштування, використання модуля, а також інформацією для адміністраторів щодо підтримки і оновлення.

РОЗДІЛ 1
АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СФЕРІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сучасна електронна комерція демонструє впевнене зростання і трансформується у важливий елемент глобальної цифрової економіки. За оцінками аналітиків, обсяг світового ринку e-commerce щороку зростає на десятки відсотків, і ця тенденція супроводжується активною автоматизацією бізнес-процесів, зростанням вимог до якості обслуговування клієнтів, а також постійною потребою у впровадженні інноваційних платіжних рішень. В умовах високої конкуренції успішність онлайн-бізнесу значною мірою залежить від технічної надійності та гнучкості цифрових інструментів, зокрема систем управління контентом і платформ для електронної торгівлі.

На етапі вибору технологічної основи для реалізації платіжного рішення було проведено порівняльний аналіз трьох найпопулярніших платформ: Shopify, WooCommerce та Magento 2. Кожна з них має свою цільову аудиторію, архітектуру, рівень відкритості та набір функцій.

Shopify – це хмарна SaaS-платформа, орієнтована на малий та середній бізнес. Її головна перевага – простота у використанні та швидкість розгортання. Для запуску магазину достатньо кількох кліків, усі технічні аспекти обслуговує сам Shopify. Однак ця зручність має свою ціну: платформа є закритою, що істотно обмежує можливості глибокої кастомізації. Платіжні інструменти здебільшого прив’язані до власного сервісу Shopify Payments або сторонніх рішень із високими комісіями. Створення або підключення власного платіжного шлюзу вимагає дотримання суворих стандартів Shopify і часто обмежене в функціональності.

WooCommerce – це безкоштовне розширення до WordPress, яке користується популярністю серед малого бізнесу. Його головна перевага – доступність та гнучкість. WooCommerce дозволяє швидко створити

інтернет-магазин на основі вже існуючого сайту. Проте платформа має низку технічних недоліків. Вона менш оптимізована для високих навантажень, що створює труднощі при масштабуванні. Залежність від екосистеми WordPress зменшує рівень ізоляції бізнес-логіки від CMS, що ускладнює реалізацію спеціалізованих функцій, наприклад, складної логіки обробки транзакцій або гнучкого управління замовленнями.

Magento 2, натомість, є повноцінною самостійною платформою для електронної комерції з відкритим вихідним кодом. Основні переваги Magento – це масштабованість, модульність і висока гнучкість. Magento має чітко структуровану архітектуру, підтримує багатосайтовість, мультимовність, мультивалютність і дозволяє створювати кастомні модулі без порушення роботи ядра системи. Крім того, вона розрахована на великі обсяги даних і високий трафік, що робить її придатною не тільки для малого, а й для великого бізнесу. Саме Magento дозволяє повністю реалізувати кастомні платіжні сценарії, інтегрувати специфічні локальні методи оплати, підключати зовнішні API, а також впроваджувати механізми автоматичного або ручного захоплення коштів, повернень, повторних дебетів тощо.

Порівняння показало, що Shopify та WooCommerce більше підходять для типових магазинів із стандартною логікою оплати, де відсутня потреба у глибокій кастомізації або контролі на рівні коду. У випадку розробки спеціалізованого платіжного плагіна, який повинен відповідати міжнародним стандартам безпеки PCI DSS, мати можливість тонкої інтеграції з внутрішніми бізнес-процесами, гнучко налаштовувати параметри транзакцій і взаємодіяти з нестандартними API – саме Magento 2 є найбільш придатною платформою. На додаток, Magento має активну спільноту розробників, добре структуровану документацію, регулярну підтримку з боку Adobe, а також перевірену екосистему модулів і тем. Це дозволяє скоротити час на інтеграцію, пришвидшити процес тестування і отримати більш стабільний результат.

Попри широкий вибір готових платіжних модулів для Magento 2, більшість із них мають обмеження, які знижують їхню ефективність у реальних

комерційних проєктах. Наприклад, EMS Online Payment позиціонується як універсальне рішення для обробки транзакцій, однак демонструє низку критичних недоліків. Модуль має обмежену підтримку кастомізації – неможливо налаштувати редіректи залежно від результату транзакції, відсутній повноцінний багатомовний інтерфейс, а внутрішня логіка побудована на закритому коді без розширюваної архітектури. Крім того, код не відповідає стандартам Magento 2 та замість використання DI і подій часто застосовуються жорстко прописані залежності, що ускладнює підтримку і масштабування.

Ще один приклад – LiqPay Magento 2 плагін від ПриватБанку. Попри популярність цієї платіжної системи, її офіційний модуль має низку технічних обмежень: відсутність офіційної підтримки останніх версій Magento, мінімальні можливості конфігурації (зокрема, щодо обробки вебхук-подій), нестабільне оновлення, що порушує сумісність, а також слабку інтеграцію з адміністративною панеллю Magento. У ньому практично немає гнучкого керування транзакціями, відсутня підтримка часткових або відкладених платежів, логування виконано на примітивному рівні. Така ситуація створює ризики для використання цього рішення у проєктах, де важлива стабільність, безпека та відповідність стандартам PCI DSS.

У результаті аналізу сучасного стану платіжних рішень на ринку, а також оцінки можливостей платформ, було прийнято обґрунтоване рішення реалізувати платіжний плагін саме для Magento 2. Це дозволить забезпечити високу надійність та безпеку обробки транзакцій, гнучке налаштування, а також можливість подальшого масштабування і адаптації під потреби конкретного бізнесу. Вирішення зазначеної проблеми дозволить розширити спектр доступних платіжних інструментів для Magento-спільноти та забезпечити ефективну підтримку користувачів, що прагнуть інтегрувати сучасні фінансові механізми у свої проєкти.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

В ході роботи над кваліфікаційною роботою бакалавра, були поставлені наступні завдання:

– проаналізувати поточний стан електронної комерції, визначити її сучасні виклики та перспективи, а також провести аналіз платформи Magento 2 шляхом порівняння з альтернативними рішеннями, такими як Shopify і WooCommerce;

– сформулювати функціональні та нефункціональні вимоги до програмного забезпечення з урахуванням архітектури Magento 2, стандартів безпеки та потреб цільових користувачів – адміністраторів магазину та покупців;

– обґрунтувати вибір технологій, мов програмування PHP та JavaScript, інструментів Composer та PHPUnit і архітектурних рішень необхідних для реалізації надійного, масштабованого плагіна;

– реалізувати основні компоненти платіжного плагіна: логіку ініціації транзакцій, взаємодію з API платіжного шлюзу, обробку вебхуків, перенаправлення користувача, адміністративну панель керування параметрами модуля;

– провести модульне та мануальне тестування розробленого плагіна у середовищі Magento 2 з метою перевірки його стабільності, коректності роботи всіх сценаріїв оплати та відповідності вимогам функціоналу;

– інтегрувати розроблений модуль з базою даних Magento 2 відповідно до стандартів платформи: використовуючи репозиторії, ресурсні моделі, поля розширення та індекси, що забезпечують продуктивність і збереження цілісності даних;

– створити інсталяційний пакет плагіна у вигляді Composer-пакета, який дозволить легко підключати модуль до сторонніх проєктів, автоматично реєструвати його в системі Magento та оновлювати в майбутньому;

– підготувати супровідну документацію з описом системних вимог, інструкціями з встановлення, налаштування, використання модуля, а також інформацією для адміністраторів щодо підтримки і оновлення.

Висновки до розділу 1

У першому розділі було проведено аналіз сучасного стану розвитку платіжних систем для електронної комерції на базі платформи Magento 2. Результати дослідження показали, що існуючі рішення мають ряд обмежень, зокрема недостатню гнучкість, складність у налаштуванні під специфічні потреби бізнесу та неповну відповідність сучасним вимогам інформаційної безпеки. Аналіз існуючих платіжних модулів виявив об’єктивну необхідність у створенні нового спеціалізованого плагіну, який забезпечить надійну інтеграцію із зовнішніми платіжними сервісами, відповідатиме стандартам безпеки, таким як PCI DSS, та надаватиме розширені можливості налаштування сценаріїв обробки транзакцій.

На підставі проведеного аналізу було сформульовано перелік основних завдань, спрямованих на розробку платіжного плагіну для Magento 2, що відповідає сучасним вимогам до функціональності, безпеки та масштабованості. Визначені цілі та завдання слугуватимуть основою для подальшого проєктування та розробки системи, що розглядаються у наступних розділах роботи.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

У процесі створення платіжного плагіну для платформи Magento 2 було обрано ітеративну модель життєвого циклу програмного забезпечення, що базується на принципах гнучкої методології Agile. Такий підхід дозволив гнучко адаптувати реалізацію до змін у зовнішньому середовищі, оперативно реагувати на нові технічні виклики та покращувати функціональність без необхідності переробки проєкту на фундаментальному рівні. Відмова від каскадної моделі була обумовлена недоцільністю жорсткої послідовності етапів у контексті інтеграції з зовнішнім платіжним шлюзом, де зміни в API або вимогах безпеки можуть виникати вже під час розробки. Використання Agile було особливо виправданим для даного типу програмного забезпечення, адже плагін мав тісну інтеграцію з зовнішніми сервісами, які можуть змінюватися незалежно від логіки магазину, а також повинен був швидко адаптуватися до нових регуляторних або функціональних вимог.

Спринти в межах проєкту мали чітко визначену структуру та складалися з етапів планування, реалізації, рев'ю та тестування. У першому спринті було створено базову структуру розширення, конфігурацію середовища Magento 2 та початкову реалізацію службових файлів. У наступних ітераціях поступово реалізовувалася логіка ініціації платежу, логування, обробка вебхуків, редірект користувача після транзакції, підтримка багатомовності та функція ручного захоплення платежів для кредитних карток. Після кожного завершеного спринту здійснювалось внутрішнє модульне тестування реалізованих компонентів, з перевіркою коректної взаємодії з API платіжного сервісу в тестовому середовищі Magento 2. Така поетапна перевірка функціональності дозволила вчасно виявляти та усувати недоліки, забезпечуючи високу стабільність плагіну вже на етапі розробки.

Другим етапом у процесі розробки стало виявлення функціональних та нефункціональних вимог до створюваного плагіну. Джерелами для формування вимог слугували аналіз існуючих платіжних модулів для Magento 2, технічна документація самої платформи, особливості інтеграції з зовнішніми платіжними API, а також практичні обмеження середовища електронної комерції. Було враховано дві основні групи користувачів: адміністраторів магазину, які взаємодіють з плагіном через інтерфейс керування, та кінцевих покупців, для яких пріоритетом є простота, швидкість і безпека транзакції. Особливу увагу приділено відповідності плагіну внутрішнім вимогам Magento 2, таким як система кешування, механізми маршрутизації та робота з DI-контейнером, що вимагало дотримання принципів модульності та чіткої структури.

У результаті аналізу сформульовано перелік функціональних вимог, які включають: ініціацію платіжної транзакції з передачею параметрів до шлюзу, обробку вебхук-повідомлень про зміну статусу платежу, логування ключових етапів обробки, реалізацію перенаправлення користувача після завершення транзакції, ручне захоплення коштів для кредитних карток, а також підтримку багатомовного інтерфейсу. До нефункціональних вимог належать: відповідність стандартам захисту даних (зокрема використання захищеного протоколу передачі HTTPS), стабільність роботи при високому навантаженні, сумісність з поточними версіями Magento 2, можливість масштабування під інші платіжні сервіси без модифікації ядра плагіну, доступність для адміністраторів через зручний UI, а також наявність технічної документації щодо налаштування та експлуатації. Таким чином, сформульовані вимоги охоплюють як базову функціональність, так і інфраструктурні особливості проєкту, що забезпечує високу якість, гнучкість та відповідність сучасним очікуванням у сфері електронної комерції.

Виявлені вимоги були класифіковані на обов’язкові, необхідні для базового функціонування модуля, та бажані, які покращують користувацький досвід або розширюють можливості подальшої інтеграції. Типовий сценарій взаємодії передбачає, що після оформлення замовлення користувач обирає

відповідний платіжний метод, після чого ініціюється транзакція з передачею даних до платіжного шлюзу. У разі успішної чи неуспішної обробки відповідне повідомлення повертається системі, що оновлює статус замовлення та виконує логування. Реалізація мультимовного інтерфейсу стала логічною вимогою для забезпечення підтримки магазинів із міжнародною аудиторією. Усі компоненти модуля мають створюватися відповідно до вимог офіційної архітектури Magento 2, із дотриманням структури директорій, правилами реєстрації сервісів та впровадженням обробників подій, що гарантує їх сумісність з екосистемою платформи.

Специфікація вимог до створюваного програмного забезпечення є ключовим етапом у проєктуванні системи, оскільки дозволяє чітко сформулювати функціональні, нефункціональні, технічні та інтеграційні характеристики майбутнього модуля. Формалізація вимог здійснювалася на основі раніше зібраної інформації щодо типових сценаріїв використання, аналізу архітектури Magento 2, особливостей реалізації REST API у зовнішньому платіжному сервісі, а також з урахуванням обмежень, які накладаються платформою електронної комерції. Такий підхід дозволив уникнути двозначностей при інтерпретації бізнес-логіки плагіну, забезпечивши підґрунтя для створення модульної, масштабованої та безпечної системи.

Ретельне документування вимог до функціональності сприяє зниженню ризику появи невідповідностей між очікуваннями кінцевих користувачів і реалізованим функціоналом. Визначення ключових компонентів системи дозволяє сформувати чітку карту відповідальності кожного з модулів, що своєю чергою полегшує тестування, модифікацію та супровід програмного забезпечення після впровадження. Особливо важливою є фіксація залежностей між підсистемами, зокрема між частинами, що відповідають за обробку транзакцій, взаємодію з API, адміністрування та логування. Така деталізація підвищує стійкість системи до збоїв, а також сприяє її відповідності принципам розділення відповідальностей (SoC) та інкапсуляції.

На основі аналізу сформовано перелік ключових функціональних блоків, які повинні бути реалізовані у складі плагіну:

– модуль ініціації платежу, що передає параметри транзакції до зовнішнього шлюзу за допомогою захищеного API-запиту;

– механізм обробки зворотних повідомлень про зміну статусу платежу, який автоматично оновлює стан замовлення в системі;

– функція ручного захоплення коштів для платежів у кабінеті адміністратора;

– реалізація редіректу користувача на сторінку підтвердження або помилки, залежно від результату операції;

– система журналювання подій з можливістю моніторингу та аудиту транзакцій;

– інтерфейс адміністрування, що дозволяє керувати параметрами модуля;

– підтримка перекладів інтерфейсу на чотири мовні локалі (англійська, німецька, нідерландська, французька).

Усі запити між плагіном та зовнішнім платіжним сервісом здійснюються через REST API з передачею структурованих даних у форматі JSON. Для забезпечення конфіденційності та цілісності інформації використовується HTTPS-протокол із ключовою аутентифікацією. Обов’язковою є перевірка коректності вхідних параметрів перед виконанням транзакційного запиту, що дозволяє уникнути небажаних операцій та збоїв у логіці оплати. Передбачено обробку виняткових ситуацій, включаючи помилки мережі, неправильні відповіді від шлюзу, відмову транзакції або втрату з’єднання під час процесу оплати.

Для кращого розуміння внутрішньої архітектури рішення та взаємодії його компонентів було прийнято рішення про побудову візуальної моделі у вигляді UML-діаграми. Таке графічне представлення дає змогу наочно відстежити основні виклики, переходи між станами та логіку обміну інформацією між ключовими елементами системи. з передачею структурованих даних у форматі JSON. Для забезпечення конфіденційності та цілісності

інформації використовується HTTPS-протокол із ключовою аутентифікацією. Обов’язковою є перевірка коректності вхідних параметрів перед виконанням транзакційного запиту, що дозволяє уникнути небажаних операцій та збоїв у логіці оплати. Передбачено обробку виняткових ситуацій, включаючи помилки мережі, неправильні відповіді від шлюзу, відмову транзакції або втрату з’єднання під час процесу оплати. З метою ілюстрації логіки роботи системи була складена UML-діаграма послідовності (рис. 2.1), яка демонструє основні етапи взаємодії між користувачем, внутрішніми сервісами плагіна, зовнішнім API та вебхук-механізмом. Діаграма відображає послідовність викликів, необхідних для формування транзакції, перенаправлення користувача, обробки відповіді від шлюзу та оновлення статусу замовлення в системі Magento 2.

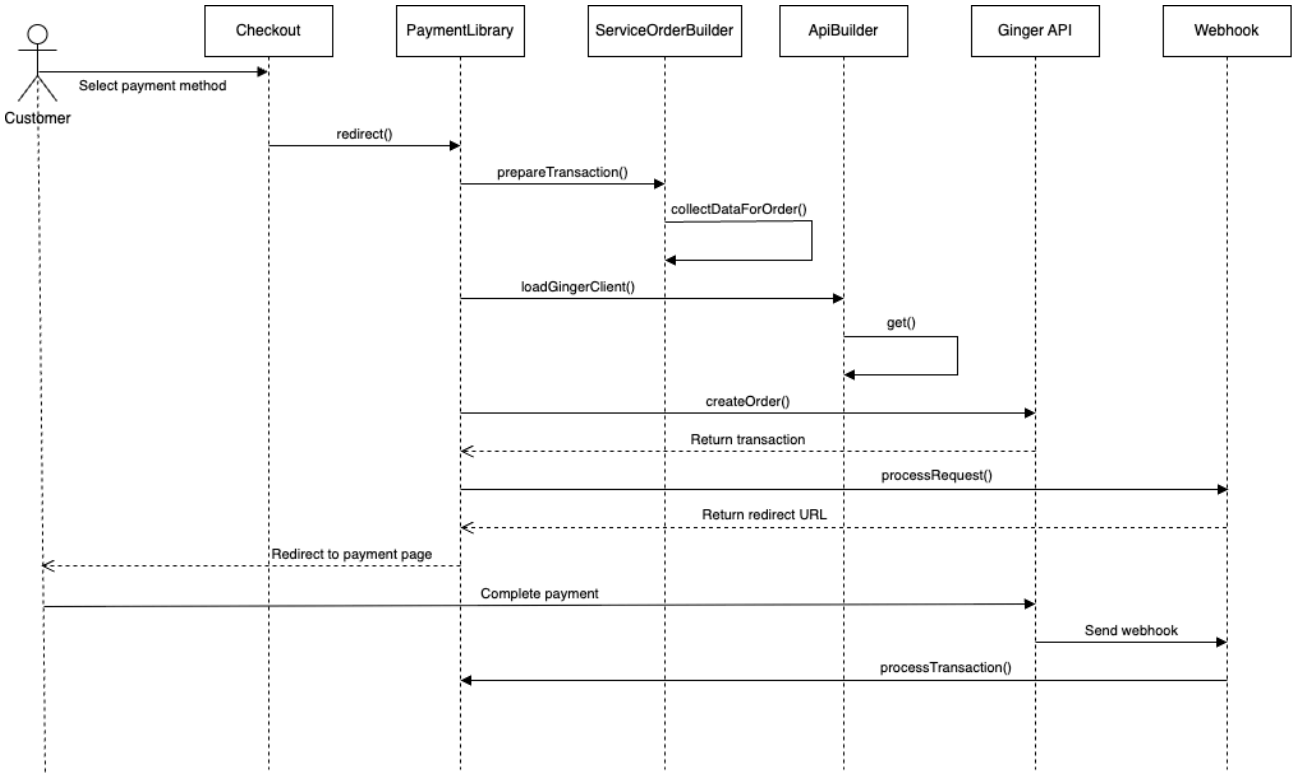

Рисунок 2.1 – Діаграма послідовності взаємодії між користувачем, плагіном та зовнішнім платіжним шлюзом

З метою детального представлення об’єктної структури основних компонентів платіжного модуля також було розроблено діаграму класів (рис. 2.2). На ній відображено ключові сутності, що реалізують

бізнес-логіку плагіна: класи, відповідальні за ініціацію транзакцій, обробку замовлень, конфігураційні налаштування, формування запитів до API, а також механізми логування та повторного використання логіки. Зв'язки між класами демонструють принципи успадкування, залежностей та використання зовнішніх інтерфейсів. Така діаграма дає змогу чітко зрозуміти, як структуровані функціональні елементи модуля та яким чином вони взаємодіють між собою для досягнення основної мети – обробки платежів.

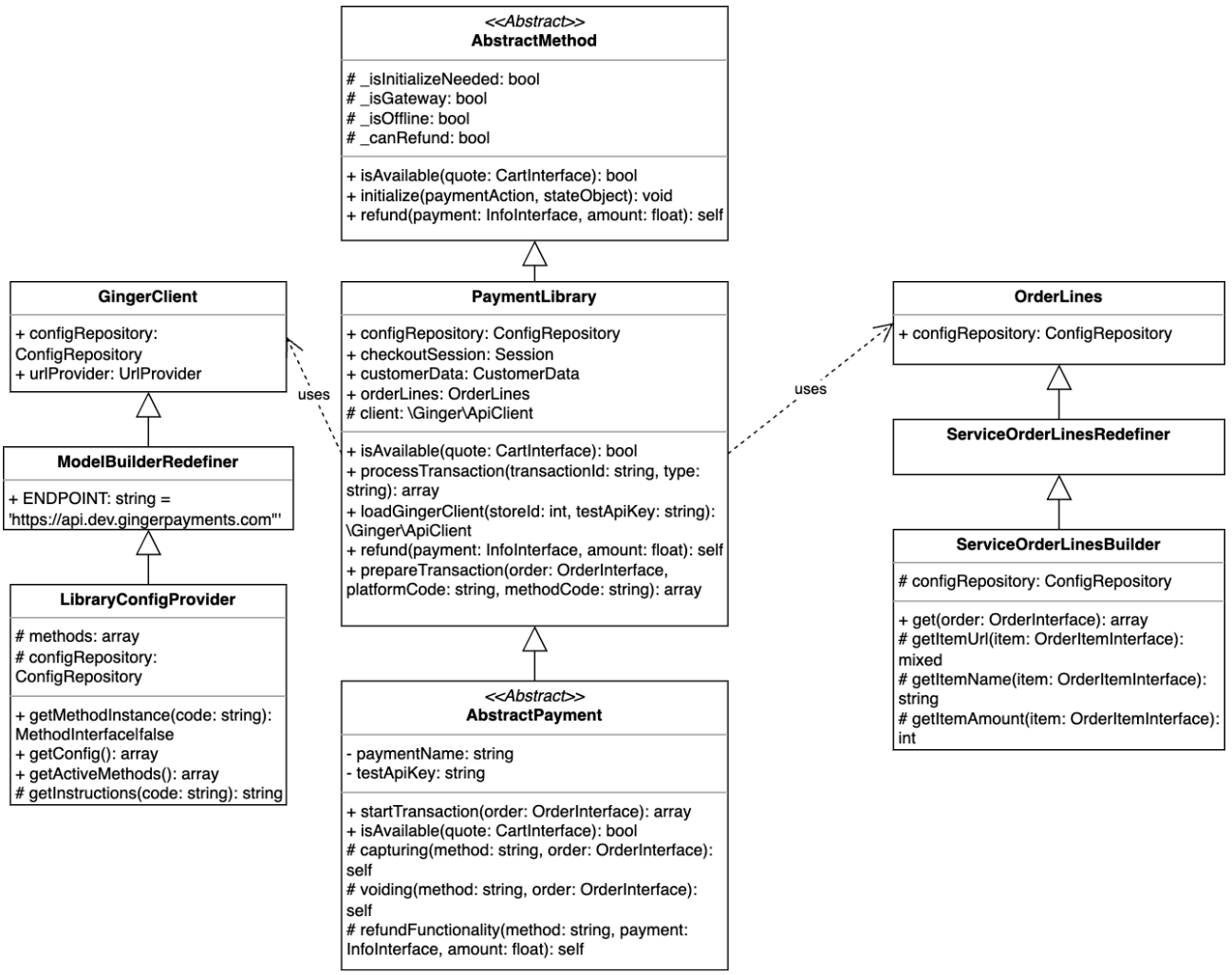

Рисунок 2.2 – Діаграма класів платіжного плагіна Magento 2

Для опису зовнішньої взаємодії різних користувачів із плагіном, а також представлення основних сценаріїв використання, було побудовано діаграму варіантів використання (рис. 2.3). Вона демонструє ключові ролі: покупця, адміністратора магазину, платіжного провайдера та API шлюзу, а також їхню

участь у процесах ініціації й завершення платежу, налаштуванні модуля, поверненні коштів і обробці зворотних повідомлень. Це дозволяє узагальнено представити всі функціональні взаємозв'язки між користувачами та компонентами системи в контексті електронної комерції.

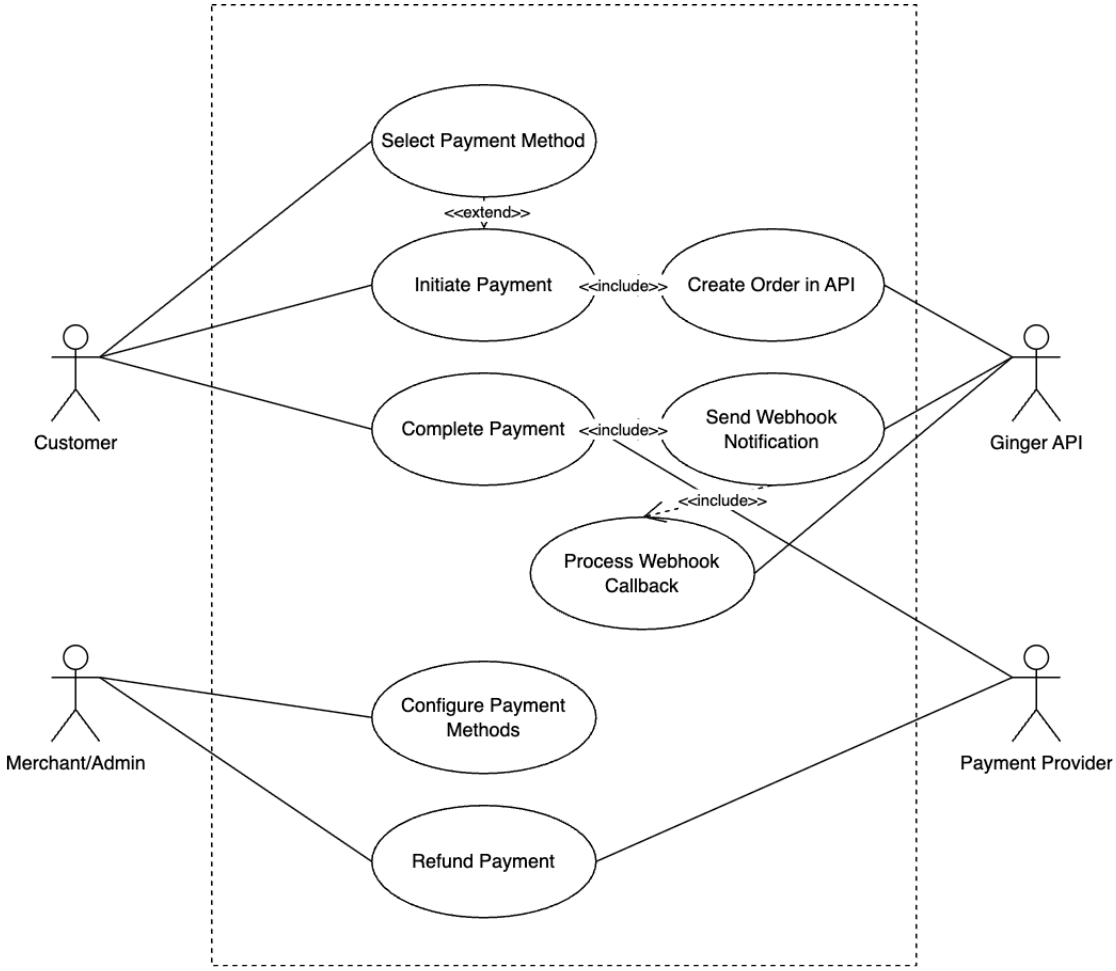

Рисунок 2.3 – Діаграма варіантів використання платіжного модуля

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка платіжного плагіну для системи керування електронною комерцією Magento 2 вимагає ретельного вибору інструментів, бібліотек, архітектурних підходів і алгоритмів, що забезпечують відповідність функціональних можливостей програмного продукту потребам цільових користувачів. Відповідно до вимог поставленого завдання, першочерговим

кроком стала оцінка доступних платформ, які дозволяють реалізовувати плагіни для онлайн-оплат. Порівняльний аналіз таких систем, як WooCommerce та Shopify виявив, що саме Magento 2, попри вищу складність інтеграції, пропонує розвинуту систему подій, гнучку архітектуру модулів та стабільний механізм розмежування прав доступу, що є критичним для платіжних рішень.

Основною мовою реалізації обрано PHP яка є однією з найпоширеніших мов, що використовуються у сфері веброзробок. Вона виникла наприкінці 90-х років і відтоді перетворилася на потужний інструмент для створення динамічних вебсторінок, серверних API, CMS, електронної комерції та SaaS-рішень. PHP вирізняється високою гнучкістю, відкритим кодом, простим синтаксисом і великим набором розширень. Завдяки широкому поширенню, PHP має потужну екосистему пакетів, які поширюються через менеджер залежностей Composer. PHP активно підтримує інтеграцію з фронтенд-технологіями (HTML, CSS, JavaScript), має розвинені можливості шаблонізації (Twig, Blade) і часто використовується разом із фреймворками, такими як Laravel, Symfony, Zend Framework. Завдяки підтримці модульної структури, об’єктноорієнтованих підходів, строгій типізації (у новіших версіях) і великому вибору розширень PHP дозволяє розробляти як невеликі вебсайти, так і складні системи з великою кількістю користувачів. У поєднанні з потужною документацією, низьким порогом входження та широкою підтримкою спільноти ця мова залишається однією з найпопулярніших у світі веброзробка. Однією з ключових переваг PHP є можливість гнучкої інтеграції з різними базами даних (MySQL, PostgreSQL, SQLite тощо), підтримка об’єктноорієнтованого програмування, розвинені механізми обробки помилок та сумісність із найпопулярнішими вебсерверами (Apache, Nginx). PHP забезпечує можливість динамічного генерування HTML-контенту, інтеграції з системами керування базами даних, а також високий рівень сумісності з вебсерверами. Версія PHP, що використовувалась у проєкті, є сумісною з вимогами Magento 2, що гарантує стабільність функціонування модуля у межах цієї екосистеми. Крім цього, слід зазначити, що останні версії PHP підтримують

такі концепції, як простори імен, анонімні класи, властивості тільки для читання (readonly), а також можливість оголошення перерахувань (enum), що дозволяє створювати більш безпечні та масштабовані рішення. Ці можливості є особливо цінними при роботі з великою кодовою базою, характерною для модульних систем на кшталт Magento. Загалом, PHP залишається стабільною та гнучкою основою для створення масштабованих вебзастосунків, забезпечуючи швидкість розробки та легкість підтримки за рахунок широкого набору інструментів та бібліотек. Однією з ключових переваг PHP є можливість гнучкої інтеграції з різними базами даних (MySQL, PostgreSQL, SQLite тощо), підтримка об’єктноорієнтованого програмування, розвинені механізми обробки помилок та сумісність із найпопулярнішими вебсерверами (Apache, Nginx). PHP забезпечує можливість динамічного генерування HTML-контенту, інтеграції з системами керування базами даних, а також високий рівень сумісності з вебсерверами. Версія PHP, що використовувалась у проєкті, є сумісною з вимогами Magento 2, що гарантує стабільність функціонування модуля у межах цієї екосистеми [1].

Magento 2 вимагає дотримання шаблону проєктування MVC (Model-View-Controller), який забезпечує чіткий розподіл обов’язків між компонентами модуля, що спрощує його обслуговування, масштабування та тестування. Архітектура Magento 2 також активно використовує принципи SOLID, модульну структуру та систему подій. Всі компоненти функціонують у межах чітко структурованої файлової системи, де кожен модуль має окремий простір імен, власні залежності, контролери, моделі, ресурси, шаблони та конфігурації. Підключення нових функціональних блоків можливе без втручання в ядро системи, що відповідає сучасним вимогам до підтримуваності та безпеки. Magento використовує спеціалізований Dependency Injection-контейнер, який дозволяє передавати об’єкти без жорсткого зв’язування між ними, спрощуючи модульне тестування та супровід. XML-файли конфігурації дозволяють описувати маршрути, залежності, компоненти UI і параметри компонування без необхідності вносити зміни до

коду, що додатково підвищує масштабованість. Layout-файли, які відповідають за візуальну структуру сторінки, дозволяють формувати динамічні інтерфейси, а ViewModel-класи допомагають із передачею даних до шаблонів. У межах системи діє жорстка відповідність іменування класів і директорій згідно зі стандартом PSR-4, що забезпечує узгодженість та автоматичне завантаження класів.

JavaScript є невід'ємною складовою розробки клієнтської частини платіжного плагіна для Magento 2, оскільки саме ця мова забезпечує логіку взаємодії користувача з інтерфейсом у режимі реального часу. JavaScript (JS) – це легка інтерпретована (або скомпільована просто в часі) мова програмування з першокласними функціями. Хоча вона найбільш відома як мова сценаріїв для вебсторінок, багато не браузерних середовищ також використовують її, наприклад, Node.js, Apache CouchDB та Adobe Acrobat. JavaScript – це багатопарадигмальна, однопотокова, динамічна мова, що базується на прототипах, підтримує об'єктноорієнтовані, імперативні та декларативні стилі [2].

У межах Magento JavaScript не виконує другорядну або допоміжну роль, а виступає ключовим інструментом, без якого неможливо реалізувати повноцінний функціонал інтерактивних компонентів. Його використання є обов'язковим для реалізації динамічних платіжних форм, валідації введених даних, побудови реактивних елементів інтерфейсу та забезпечення швидкої реакції системи на дії користувача – усе це критично важливо для сучасної електронної комерції.

Однією з причин широкого використання JavaScript у Magento є потреба в асинхронній обробці даних, що дозволяє уникнути повного перезавантаження сторінки при зміні окремих її фрагментів. Наприклад, під час вибору методу оплати, зміни валюти або перевірки коректності заповнених полів – усі ці дії обробляються миттєво, завдяки можливості JavaScript надсилати запити до сервера, обробляти відповіді та відображати їх на сторінці без втрати контексту користувача. Такий підхід значно підвищує зручність та ефективність взаємодії

з платіжним інтерфейсом, що особливо важливо в процесі завершення покупки, де кожна зайва секунда або зайвий клік можуть призвести до втрати клієнта [3].

Magento 2 активно використовує можливості JavaScript як у стандартних елементах, так і в кастомних рішеннях. Мова дозволяє гнучко керувати DOM-структурою, додавати або видаляти елементи на сторінці, змінювати стилі, вміст блоків або логіку поведінки залежно від контексту. Це дозволяє реалізовувати складні сценарії, як-от умовне відображення форм залежно від обраного способу доставки чи автоматичне оновлення загальної суми замовлення при зміні параметрів. Важливим аспектом є й те, що JavaScript у Magento 2 використовується в тісному зв’язку з іншими технологіями, такими як CSS, HTML та серверна частина на PHP. Вся логіка платіжного процесу – від ініціалізації сесії до фінального підтвердження транзакції – часто супроводжується клієнтською логікою, реалізованою саме на JavaScript. Завдяки підтримці сучасних стандартів мови (ES6+), розробник має змогу використовувати розширені синтаксичні конструкції, функціональні можливості та структурування коду, що підвищує його читабельність і підтримуваність.

CSS, або каскадні таблиці стилів, є основним інструментом для оформлення зовнішнього вигляду вебсторінки, і в контексті розробки платіжного плагіна для Magento 2 його значення важко переоцінити. CSS – спеціальна мова, яка використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найбільш часто CSS використовують для візуальної презентації сторінок, написаних на HTML та XHTML, але формат CSS може застосовуватись і до інших видів XML-документів [4].

Цей інструмент відповідає не лише за естетичне оформлення, але й за структурну цілісність та зручність інтерфейсу. У середовищі Magento саме за допомогою CSS забезпечується єдиний візуальний стиль компонентів, включаючи поля для введення даних, кнопки підтвердження, інформаційні повідомлення та інші елементи платіжної форми.

У межах платформи Magento 2 стилі впроваджуються за допомогою сучасного підходу, який базується на використанні препроцесора LESS. Ця

технологія дозволяє спростити розробку стилів шляхом застосування змінних, шаблонів, вкладеності та інших можливостей, які підвищують зрозумілість та повторне використання коду. LESS у Magento компілюється у звичайний CSS-код, що виконується автоматично під час збирання проєкту. Це дозволяє зменшити кількість ручної роботи та забезпечити узгодженість оформлення на різних сторінках сайту.

Особливу увагу під час розробки було приділено адаптивності. Завдяки використанню медіа-запитів вдалося налаштувати автоматичну зміну вигляду інтерфейсу залежно від розміру екрана користувача. Це дозволяє гарантувати, що платіжні форми однаково зручно використовувати як на комп’ютері, так і на планшеті чи мобільному телефоні. У поєднанні з можливостями гнучкого розміщення елементів, які забезпечуються сучасними властивостями CSS, такими як блокове вирівнювання або сіткова розкладка, вдалося досягти високого рівня ергономічності.

Під час розробки платіжного плагіна для Magento 2 важливою вимогою стало забезпечення багатомовності інтерфейсу, оскільки користувачі інтернет-магазину можуть перебувати в різних мовних середовищах. З цією метою було впроваджено механізми інтернаціоналізації i18n, яка дозволя адаптувати текстову частину інтерфейсу до конкретної мови користувача без зміни загальної логіки роботи програмного забезпечення. Інтернаціоналізація охоплює процеси та практики, спрямовані на розробку застосунку в такий спосіб, щоб він був готовий до підтримки різних мов і культурних норм. Це і підтримка безлічі мов, і форматування дат і чисел відповідно до різних стандартів, і адаптація до різних систем письма, тощо. У Magento 2 підтримка локалізації є системною функцією і реалізується через файлова структуру мовних перекладів, спеціальні словники та відповідну конфігурацію модулів [5].

Система інтернаціоналізації у Magento побудована таким чином, щоб розробник мав можливість легко додавати нові мовні версії без потреби змінювати вихідний код. Усі текстові повідомлення винесені у файли перекладів

у вигляді таблиць відповідності: оригінальний текст – переклад. Це дозволяє централізовано керувати мовними ресурсами й оперативно вносити зміни без ризику порушення структури коду або функціональності плагіна. Такий підхід підвищує підтримуваність рішення та спрощує його масштабування на інші ринки. Для динамічного відображення мовного контенту на стороні користувача в плагіні також було використано сторонню бібліотеку, яка підтримує оновлення тексту в режимі реального часу без перезавантаження сторінки. Завдяки цьому стало можливим створити інтерфейс, що автоматично адаптується до вибраної мови, уникаючи дублювання шаблонів або створення окремих сторінок для кожної локалі. Особливо актуально це при роботі з платіжними формами, де точність формулювань, зрозумілість підказок і повідомлень про помилки мають вирішальне значення [6].

Таким чином, впровадження i18n у платіжний плагін для Magento 2 дозволило не лише реалізувати підтримку багатьох мов, а й підвищити якість користувацького досвіду за рахунок врахування мовних і культурних особливостей аудиторії. Це робить рішення придатним для використання у міжнародних торговельних середовищах та відповідає сучасним вимогам до програмного забезпечення у сфері електронної комерції.

Висновки до розділу 2

У другому розділі було детально визначено вимоги до створюваного платіжного плагіна для системи Magento 2, а також обґрунтовано вибір методів, інструментів і технологій, необхідних для його реалізації. Проведений аналіз функціональних і нефункціональних характеристик дозволив сформувати чітке бачення структури майбутнього модуля, його основних компонентів і способів взаємодії з зовнішнім платіжним API. В якості основної моделі розробки обрано ітеративний підхід, що забезпечив гнучкість у впровадженні змін, адаптацію до зовнішніх викликів і поступове вдосконалення функціональності. Сформульовані вимоги охоплюють як технічні аспекти (підтримка REST API,

DI, кешування, інтернаціоналізація), так і користувацькі сценарії (ініціація платежу, редірект, ручне захоплення, обробка вебхуків). Обґрунтовано вибір мови програмування PHP як основної для реалізації бізнес-логіки, а також використання JavaScript і CSS для побудови інтерактивного і адаптивного інтерфейсу. Ретельно підібрані бібліотеки, стандарти й архітектурні підходи такі як PSR, MVC, SOLID дозволяють забезпечити модульність, масштабованість і безпеку системи. Також було приділено увагу локалізації інтерфейсу плагіна, що робить його придатним до використання в міжнародному середовищі.

Отже, виконана в рамках цього розділу підготовча робота створила надійне підґрунтя для практичної реалізації платіжного модуля, що відповідає актуальним вимогам сучасної електронної комерції.

РОЗДІЛ 3
РОЗРОБКА ПЛАТІЖНОГО ПЛАГІНУ ДЛЯ CMS MAGENTO

3.1 Практична реалізація об’єкта проєктування

У процесі розробки платіжного плагіна для системи Magento 2 реалізація велась на основі поетапного впровадження модулів відповідно до визначених у специфікації функціональних блоків.

Основною мовою програмування обрано PHP, яка повністю підтримується архітектурою Magento та забезпечує реалізацію логіки обробки транзакцій, взаємодії з платіжним шлюзом через API, обробки вебхуків і внутрішнього логування.

Сервісні класи були впроваджені із використанням принципів інверсії залежностей та шаблону MVC, що є обов’язковим у середовищі Magento. Усі компоненти модуля зареєстровані згідно зі стандартом PSR-4, а підключення реалізоване через Dependency Injection-контейнер. Dependecy Injection – це проєктувальний патерн, за яким об’єкт або функція отримує інші об’єкти або функції, від яких вони залежить. DI має на меті розділення процесів створення та використання об’єктів іншими класами, що призводить до певної децентралізації системи [7].

Процес практичної реалізації розпочато зі створення структури модуля – каталогів Controller, Model, Helper, Observer, view тощо. Було впроваджено логіку ініціації транзакцій на основі параметрів кошика користувача з формуванням відповідного запиту до API платіжного шлюзу. Відповідь шлюзу обробляється для здійснення перенаправлення користувача до відповідної сторінки – підтвердження або повідомлення про помилку.

Метод prepareTransaction в одному з основних сервісів плагіна відповідає за ініціацію запиту до платіжного API, як представлено на лістингу 3.1. У ньому дані з об’єкта замовлення збираються, структуруються та передаються до шлюзу.

Лістинг 3.1 – Код методу prepareTransaction

```
public function prepareTransaction(OrderInterface $order, $platformCode,
$methodCode): array
{
    $custumerData = $this->customerData->get($order, $methodCode);
    $additionalData = $order->getPayment()->getAdditionalInformation();
    switch ($platformCode) {
        case 'afterpay':
                                                                $testApiKey    =
$this->configRepository->getAfterpayTestApiKey((int)$order->getStoreId())
;
            $testModus = $testApiKey ? 'afterpay' : false;
            if (isset($additionalData['terms']))
            {
                                        ($additionalData['terms']  ==  1)  ?
$verifiedTermsOfService = true : $verifiedTermsOfService = false;
            }
            break;
        case 'klarna-pay-later':
                                                                $testApiKey    =
$this->configRepository->getKlarnaTestApiKey((int)$order->getStoreId());
            $testModus = $testApiKey ? 'klarna' : false;
            break;
        case 'ideal':
                                                            $additionalData    =
$order->getPayment()->getAdditionalInformation();
            if (isset($additionalData['issuer']))
            {
                $issuer = $additionalData['issuer'];
            }
            break;
    }
                                                    $paymentDetails           =
$this->orderDataCollector->getTransactions($platformCode,
$verifiedTermsOfService, $methodCode);
     $orderData = $this->orderDataCollector->collectDataForOrder($order,
$methodCode,  $this->urlProvider,  $this->orderLines,  $paymentDetails,
$custumerData);
          $client  =  $this->loadGingerClient((int)$order->getStoreId(),
$testApiKey);
    $transaction = $client->createOrder($orderData);
    return $this->processRequest($order, $transaction, $testModus);
}
```

кінець лістингу 3.1

Окрема увага приділялась реалізації адміністративного інтерфейсу для конфігурації модуля, який вбудовується в адмінпанель Magento. Для цього було

створено окремі XML-файли конфігурацій, секції у меню та групи полів. Адміністратор може вказати API-ключі, активувати або деактивувати методи оплати, визначити поведінку транзакцій (автоматичне або ручне захоплення), а також протестувати підключення до шлюзу. Всі елементи були створені із використанням стандартного шаблонізатора Magento та оформлені згідно з поточною темою магазину, що дозволяє зберігати єдність стилю.

Особливу увагу приділено підтримці багатомовності інтерфейсу (i18n). Для цього всі текстові повідомлення були винесені у відповідні словникові файли Magento, що дозволяє легко перекладати інтерфейс плагіна на будь-яку мову без зміни коду. Такий підхід забезпечив гнучку адаптацію до потреб міжнародних користувачів та підтримку вимог локального законодавства щодо мовної політики.

Реалізація функціональних і візуальних компонентів плагіна завершилася формуванням цілісного, технічно узгодженого модуля, що здатен стабільно функціонувати у середовищі Magento 2 та відповідати актуальним вимогам безпеки, зручності та гнучкості налаштування. Зокрема, інтерфейс адміністративної панелі, як зображено на рисунку 3.1, дозволяє ефективно керувати параметрами модуля, контролювати активність платіжних методів і взаємодіяти з API платіжного шлюзу.

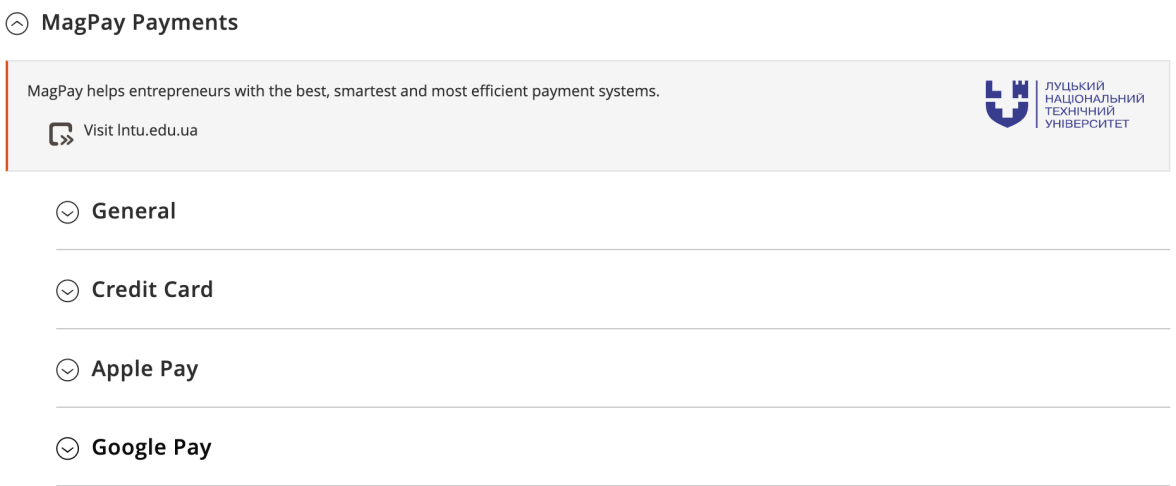

Рисунок 3.1 – Головне меню конфігурації платіжного модуля MagPay

У межах розділу «General» передбачено ряд ключових параметрів, зокрема: версію модуля, статус активності, API Key, опції тестування підключення до шлюзу, а також налаштування режиму дебагу, що ілюструється на рисунку 3.2. Така структура забезпечує як прозорість адміністрування, так і швидку валідацію працездатності інтеграції без потреби в залученні додаткових інструментів. У процесі тестування було виявлено, що можливість негайно протестувати API-ключ дає змогу суттєво зменшити кількість помилок на етапі впровадження. Проста форма з підтвердженням «Success!» дозволяє не лише перевірити зв’язок із сервером платіжної системи, але й формує в адміністратора впевненість у коректності налаштувань.

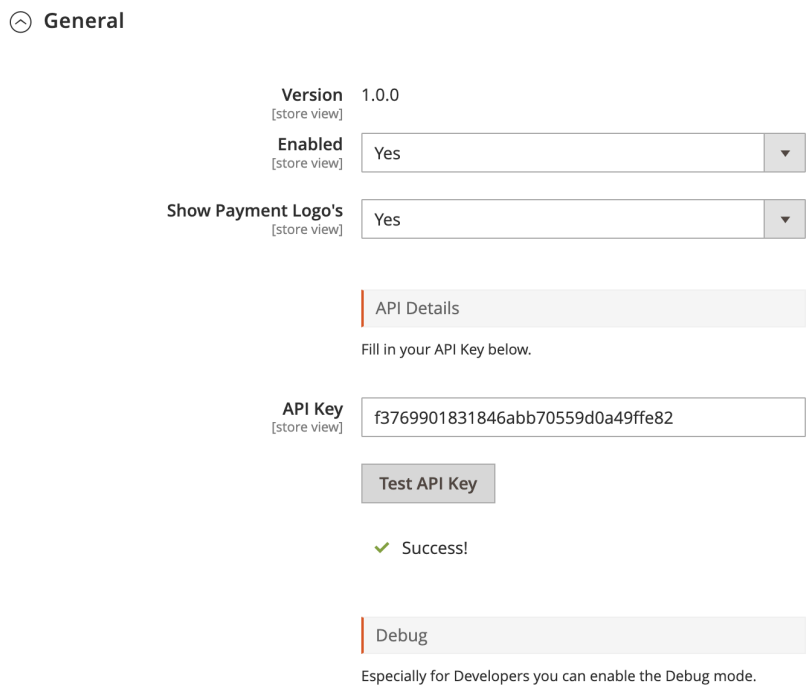

Рисунок 3.2 – Загальні параметри плагіну: API ключ, версія, тестування підключення

Окрему увагу було приділено налаштуванням для методу оплати банківською карткою. У секції «Credit Card» адміністратор має змогу встановити поведінку замовлення відповідно до статусу транзакції: очікування, авторизація, обробка, як показано на рисунку 3.3. Також тут задається шаблон тексту, який відображається клієнтові під час здійснення оплати, що позитивно

впливає на прозорість UX. Режим «Manual Capture» протестовано на предмет коректності створення транзакцій для подальшого ручного захоплення. Ця гнучкість забезпечує адаптивність модуля до специфіки фінансових процесів конкретного бізнесу, де автоматичне захоплення не завжди є бажаним.

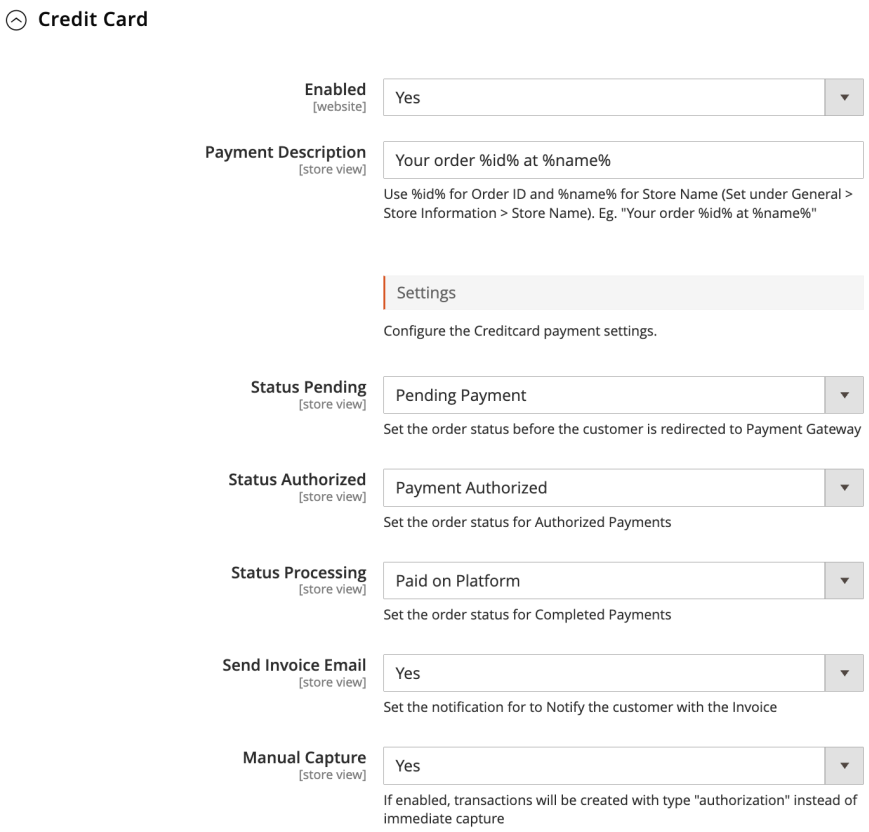

Рисунок 3.3 – Налаштування методу Credit Card: статуси, опис, захоплення

На стороні користувача було перевірено візуальне відображення платіжних методів на сторінці оформлення замовлення, як показано на рисунку 3.4. Плагін підтримує розпізнавання валідних методів оплати згідно з конфігурацією й забезпечує відображення відповідних логотипів у зрозумілому вигляді. Це підвищує довіру клієнта та дозволяє обрати зручний метод оплати вже на етапі оформлення замовлення. Візуальна структура інтерфейсу зберігає корпоративний стиль магазину, що формує єдине середовище взаємодії для користувача.

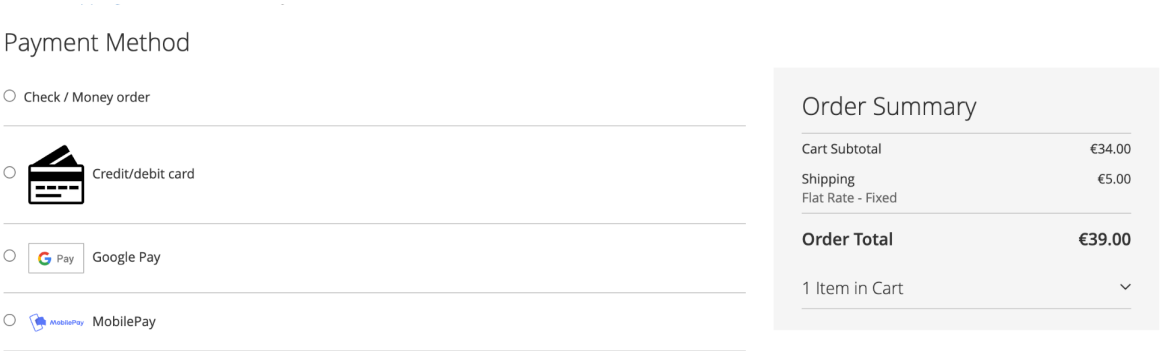

Рисунок 3.4 – Вибір платіжного методу на сторінці оформлення замовлення

Подальший крок включає перенаправлення користувача до зовнішнього платіжного шлюзу, як продемонстровано на рисунку 3.5. Хоча ця форма не є частиною безпосередньої реалізації плагіну, вона становить ключовий етап у загальному користувацькому сценарії. У ході тестування було перевірено коректність перенаправлення, стабільність завантаження сторінки шлюзу, обробку позитивних і негативних сценаріїв введення реквізитів, а також взаємодію користувача в мобільному середовищі. Особливу увагу приділено перевірці збереження контексту транзакції після повернення користувача до магазину.

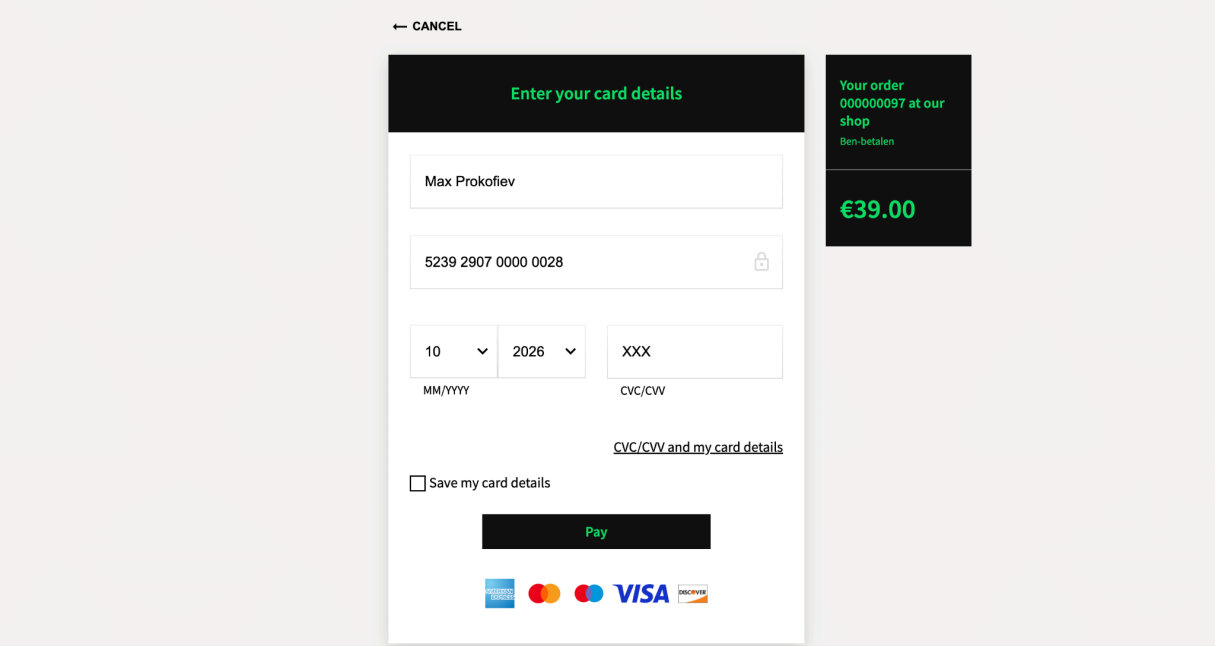

Рисунок 3.5 – Форма введення даних банківської картки на стороні платіжного шлюзу

Завершальним етапом стало повідомлення про успішну оплату. У випадку валідної транзакції, система коректно формує сторінку підтвердження замовлення із зазначенням його номера та додатковими інструкціями, як зображено на рисунку 3.6. Це є підтвердженням завершення бізнес-логіки та гарантією для користувача. Окрема увага приділялась валідації, чи зберігається стабільність при переході з вікна платіжного шлюзу назад на сторінку магазину. Підтвердження, отримане клієнтом після успішної транзакції, є критично важливим з огляду на UX та запобігання двозначного трактування статусу оплати.

Рисунок 3.6 – Підтвердження про успішне оформлення замовлення

Усі етапи розробки проводилися із використанням тестових облікових даних у середовищі, що дозволяє безпечно імітувати реальні транзакції. Під час перевірки були усунуті логічні неточності у статусах, які не впливали на функціональність, але могли створювати неоднозначність для користувача. У результаті сформовано стабільне, надійне рішення, яке може бути рекомендоване до впровадження в електронні торговельні системи, побудовані на Magento 2, з мінімальними ризиками і високою готовністю до використання в комерційному середовищі.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У процесі створення платіжного плагіна для Magento 2 було реалізовано комплекс заходів, спрямованих на забезпечення стабільної, безпечної та прогнозованої роботи програмного забезпечення. Особливу увагу приділено тестуванню функціональних компонентів плагіна з використанням як

автоматизованих, так і ручних методів перевірки. Основним інструментом для автоматизованого контролю якості реалізації виступило модульне тестування, що дозволяє ізольовано перевіряти логіку окремих класів і методів без потреби завантаження повної інфраструктури Magento. Цей підхід було обрано з огляду на складність взаємодії між компонентами платформи та необхідність точного контролю за бізнес-логікою, зокрема –n у процесах, пов’язаних з фінансовими транзакціями. Завдяки модульному тестуванню стало можливим оперативно перевіряти реакцію окремих частин системи на різні вхідні дані, передбачувано обробляти виключення, а також мінімізувати ймовірність виникнення неочікуваних побічних ефектів при внесенні змін до коду.

Тестування реалізовувалося з використанням фреймворку PHPUnit, який дозволив побудувати повноцінну систему автоматизованого контролю якості. У ході тестування були охоплені як сценарії із валідними транзакціями, так і випадки, коли система стикається з некоректними або відсутніми даними. Одним із прикладів стало моделювання ситуації, коли плагін отримує вебхук-запит із ID транзакції, що не пов’язаний із жодним замовленням у системі. Такий сценарій особливо важливий для виявлення прогалин в обробці помилок, що можуть призвести до невірних оновлень статусу або порушення логіки роботи магазину [8].

В лістингу 3.2 наведено фрагмент юніт-тесту, який перевіряє, чи коректно метод processTransaction обробляє ситуацію, коли замовлення не знайдено. За допомогою мок-об’єкта змодельовано відсутність результату пошуку, після чого виконується перевірка, що система повертає відповідь з ознакою помилки.

Лістинг 3.2 – Модульний-тест для перевірки обробки відсутнього замовлення у транзакції

```
public function testProcessTransactionWithOrderNotFound()
{
    $transactionId = 'test_transaction_id';
    // Configure mocks
    $this->getOrderByTransactionMock->expects($this->once())
        ->method('execute')
```

```
        ->with($transactionId)
        ->willReturn(null);
     $result = $this->paymentLibrary->processTransaction($transactionId,
'webhook');
    // Assert result contains error
    $this->assertTrue($result['error']);
    $this->assertArrayHasKey('msg', $result);
}
```

кінець лістингу 3.2

Такий підхід до тестування дозволяє не лише перевірити позитивні сценарії, а й забезпечити стійкість системи до виняткових ситуацій, які можуть виникати у процесі реальної експлуатації, зокрема при втраті зв’язку, дублюванні запитів або частковій синхронізації між системами.

Окрім юніт-тестів, велике значення мало ручне тестування у середовищі Magento 2 із підключеним тестовим акаунтом платіжного провайдера. Під час цього етапу перевірялися типові користувацькі сценарії: оформлення замовлення, вибір платіжного методу, перенаправлення на шлюз, обробка відповіді та оновлення статусу. Було перевірено коректність логіки переходів між станами транзакції та відповідність результатів дій очікуваній поведінці в інтерфейсі адміністратора.

Наслідком поетапного тестування стало виявлення кількох незначних логічних похибок, які були оперативно усунуті без змін до загальної архітектури. Ефективне логування та наявність чітко структурованої системи модульних перевірок дозволили локалізувати помилки, підвищити стабільність коду та підготувати систему до подальшого масштабування й продуктивної роботи у реальному середовищі.

3.3 Розробка бази даних

На відміну від створення автономної бази даних, розробка платіжного плагіна для Magento 2 була зосереджена на інтеграції з уже існуючою складною структурою бази даних платформи. Magento використовує модульну,

розширювану архітектуру, яка дає змогу безпечно додавати нові атрибути, таблиці та зв’язки без порушення цілісності або сумісності з ядром системи. У межах цього підходу було реалізовано низку змін, що дозволяють плагіну взаємодіяти з даними про замовлення, транзакції, платежі та товари [9].

Одним з ключових аспектів реалізації стало доповнення стандартної таблиці sales_order новим полем gingerpay_transaction_id, що дозволяє зберігати унікальний ідентифікатор транзакції, присвоєний зовнішнім платіжним шлюзом Ginger. Це поле слугує точкою зв’язку між замовленням у Magento та відповідною транзакцією у зовнішній платіжній системі. Для підвищення швидкодії запитів, пов’язаних із пошуком замовлень за ідентифікатором транзакції, було створено індекс для цього поля, що суттєво оптимізує вибірку в умовах високого навантаження. Окрім розширення існуючих таблиць, плагін також створює власну спеціалізовану таблицю gingerpay_product, у якій зберігається інформація про товари, пов’язана з логікою інтеграції. Ця таблиця містить базові атрибути товарів – такі як назва, опис, ціна – а також службові поля created_at та updated_at, що використовуються для контролю версійності та зміни записів. Збереження інформації відбувається за допомогою моделі Magento з прив’язкою до відповідної ресурсної моделі, що дозволяє виконувати операції над даними у спосіб, узгоджений із внутрішньою логікою платформи.

Плагін повністю дотримується архітектурних принципів Magento щодо доступу до даних. Всі операції читання та запису виконуються за допомогою патерна Repository, що дозволяє абстрагуватись від конкретної реалізації схеми БД і зосередитись на бізнес-логіці. Для створення умов вибірки застосовується SearchCriteriaBuilder, який дозволяє динамічно формувати запити на основі параметрів, переданих у методи сервісів. Наприклад, для отримання замовлення за транзакційним ідентифікатором формується фільтр, що звужує вибірку до одного результату, що забезпечує високу точність і швидкість роботи [10].

Також реалізовано механізм додавання записів до історії замовлень – зокрема, у відповідь на зміну статусу транзакції після її обробки. Це дозволяє адміністраторам отримувати повну інформацію про перебіг платежу та

реагувати на виняткові ситуації. Коментарі зберігаються через відповідні моделі, без прямого доступу до бази, із можливістю налаштування повідомлень для клієнтів. Інтеграція із системою кешування також є важливою частиною взаємодії з БД. Для оптимізації запитів до зовнішнього API реалізовано кешування даних про доступні валюти та конфігурації. Кеш зберігається локально у файловій системі, але логіка роботи з ним може бути розширена до збереження в базі або системному кеші Magento, що забезпечує гнучкість налаштувань і адаптацію під вимоги конкретного середовища [11].

Для ілюстрації архітектури реалізованої взаємодії між платіжним модулем MagPay і таблицями Magento була створена діаграма структури бази даних. Вона відображає як розширення стандартних таблиць, так і взаємозв'язки між ними та новими елементами, впровадженими модулем. Зокрема, показано, як поле gingerpay_transaction_id пов'язане з іншими таблицями, такими як sales_order_payment та sales_order_grid, а також яким чином таблиця gingerpay_product вписується в загальну структуру модуля, як показано на рисунку 3.7.

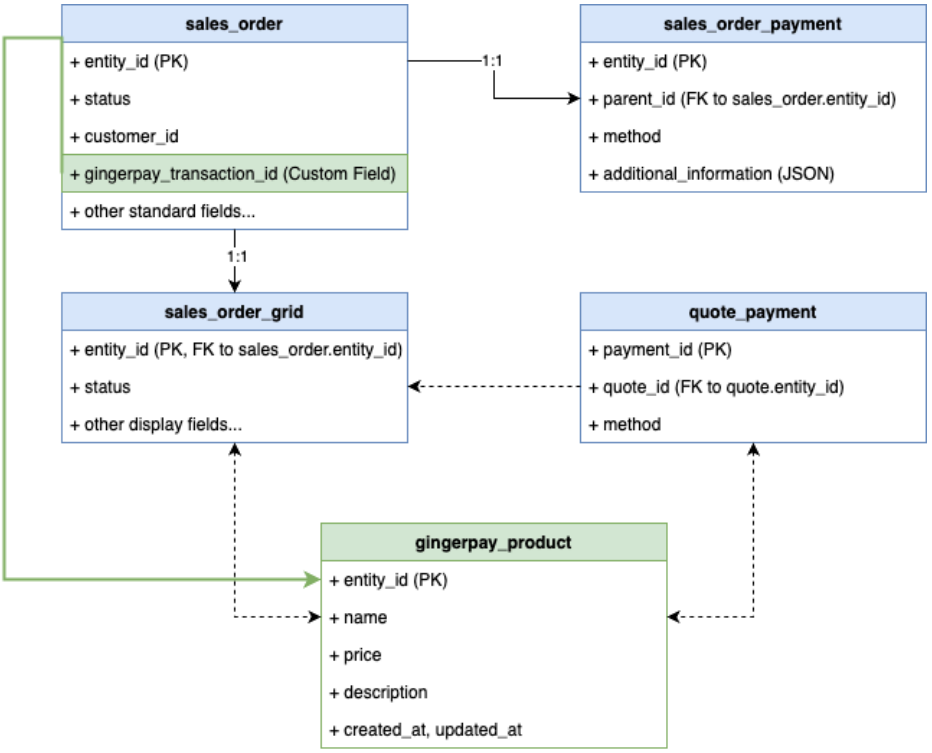

Рисунок 3.7 – Структурна діаграма таблиць і взаємозв'язків бази даних модуля

У підсумку реалізована база даних забезпечує надійну інтеграцію з внутрішніми механізмами Magento 2 без порушення її архітектурних принципів. Завдяки використанню вбудованих засобів роботи з даними, таких як репозиторії, ресурсні моделі та setup-скрипти, вдалося розширити існуючу структуру таблиць і додати підтримку додаткової інформації, необхідної для обробки транзакцій. Такий підхід гарантує стабільність роботи, збереження сумісності з ядром платформи, а також готовність до масштабування й подальшого розширення функціональності без ризику втрати цілісності даних.

3.4 Розробка інсталяційного пакета

Інсталяційний пакет розробленого модуля був створений з урахуванням вимог до модульної архітектури платформи Magento 2 та сучасних підходів до розгортання PHP-застосунків. Основним способом розповсюдження і підключення плагіна виступає менеджер пакетів Composer. Composer – це інструмент управління залежностями в проектах PHP. Він забезпечує ефективний спосіб управління бібліотеками, завантаження необхідних пакетів та створення середовища розробки з мінімальними зусиллями. Composer – пакетний менеджер для PHP. Він дозволяє завантажувати з офіційного репозиторію packagist.org використовувані проектом бібліотеки і їх залежності. За рахунок цього кожну з них не потрібно завантажувати і підключати вручну, а також немає необхідності зберігати їх безпосередньо в самому проекті, що скорочує його розміри. Пакет розміщується в окремому репозиторії з коректно налаштованим файлом composer.json, який містить основну службову інформацію про модуль: його унікальну назву, тип (magento2-module), версію, авторів, опис, залежності від версій PHP та Magento, а також налаштування автозавантаження класів за стандартом PSR-4. Така структура дозволяє Magento розпізнавати модуль автоматично після встановлення та виклику стандартних команд налаштування середовища [12].

Каталоги модуля структуровано відповідно до функціонального розподілу: окремі простори імен призначені для моделей, ресурсних класів, конфігурацій, сервісів, контролерів, шаблонів та обробників подій. Усі компоненти модуля організовано у спосіб, який повністю відповідає стандартам Magento, що дозволяє легко підтримувати, оновлювати та масштабувати плагін у майбутньому. Реєстрація модуля в системі здійснюється автоматично під час виконання команд setup:upgrade та module:enable. У процесі активації система Magento ініціює створення або оновлення таблиць, перевірку залежностей і оновлення службового кешу. Наявність файлів registration.php та module.xml забезпечує ідентифікацію модуля у реєстрі модулів Magento, що є обов'язковою умовою для його коректної роботи [13].

Конфігурація параметрів модуля зберігається у внутрішньому реєстрі системи Magento, який представлений у вигляді таблиці core_config_data. Усі змінні, що зчитуються або змінюються через адміністративну панель, зберігаються у вигляді структурованих значень, прив'язаних до відповідного шляху конфігурації. Це дозволяє зберігати налаштування незалежно від коду та забезпечує гнучке керування поведінкою модуля без внесення змін до його логіки. Окрім базового функціоналу, інсталяційний пакет також містить XML-конфігурації, які відповідають за створення адміністративного інтерфейсу. Меню плагіна, блоки форм, поля вводу, а також повідомлення для адміністратора формуються через окремі файли в каталозі view/adminhtml/ui_component та etc/adminhtml. Такий підхід дозволяє автоматично інтегрувати плагін у адміністративний інтерфейс Magento без потреби створення ярликів або додаткових налаштувань вручну [14].

З метою полегшення розгортання плагіна у сторонніх проєктах було реалізовано його публікацію у вигляді Composer-пакета на платформі Packagist. Це дозволяє будь-якому користувачеві інтегрувати модуль у власний проєкт за допомогою єдиної команди встановлення, без потреби копіювати файли вручну або налаштовувати залежності локально. Сторінка пакета містить опис,

інструкції з встановлення та оновлення, а також дозволяє слідкувати за його версіями (рис. 3.8).

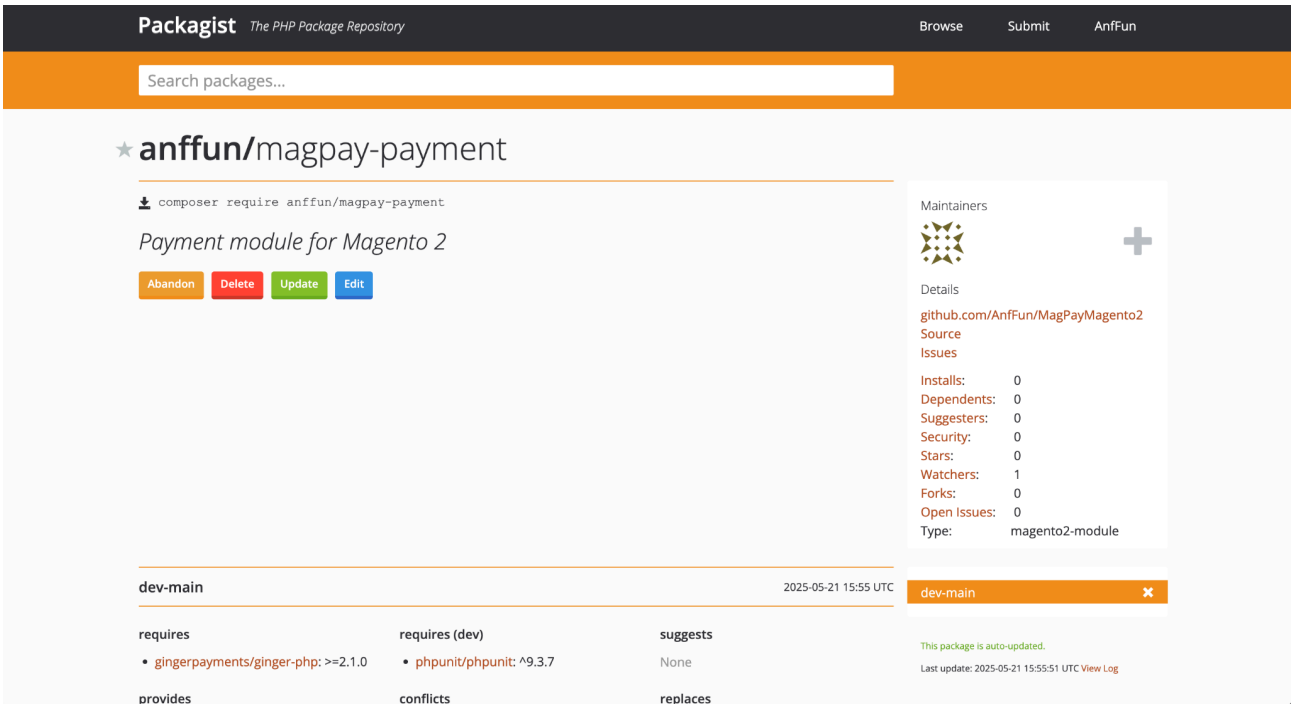

Рисунок 3.8 – Публікація інсталяційного пакета модуля у Packagist

Таким чином, структура інсталяційного пакета відповідає усім вимогам до модульного програмного забезпечення на базі Magento 2. Завдяки підтримці Composer, автоматизованій реєстрації, стандартизованій структурі каталогів та централізованому зберіганню конфігурацій, модуль є повністю сумісним із системою і придатним до подальшого розширення та обслуговування.

3.5 Розробка документації для програмного продукту

Для забезпечення зручності використання платіжного плагіна для Magento 2 паралельно з розробкою програмного забезпечення також здійснюється розробка супровідної документації. Основним завданням документації є надання користувачам усіх необхідних інструкцій щодо встановлення, налаштування та експлуатації плагіна. На поточному етапі створено базовий файл документації у форматі README.md. README це

текстовий файл, що містить інформацію про інші файли в тому ж каталозі або архіві, він зазвичай супроводжує дистрибутив програми при розповсюдженні. Цей файл буде доповнюватись в процесі завершення проєкту більш розгорнутими довідковими матеріалами [15].

Для коректної роботи плагіна користувач повинен мати сервер з попередньо встановленою платформою Magento версії 2.4.6 або новішої, а також підтримкою PHP від версії 8.2 до 8.4. Обов’язковою є наявність менеджера залежностей Composer, доступ до бази даних MySQL, вебсервера Apache або Nginx та можливості виконання команд у терміналі. Також необхідно надати права на запис у системні директорії Magento. Основні команди для встановлення плагіна вже наведені в документації, і включають послідовне виконання команд composer require, php bin/magento module:enable, php bin/magento setup:upgrade, а також очистку кешу й, за потреби, деплой статичного контенту. У майбутніх версіях плагіна планується створення окремої розширеної документації у вигляді PDF-файлу або інтегрованого довідкового модуля в адмін-панелі Magento, що міститиме структурований опис можливостей, технічні вимоги, приклади використання та відповіді на типові питання. Також передбачено розробку коротких пояснень та підказок безпосередньо в інтерфейсі налаштувань плагіна, що допоможе зробити взаємодію з модулем максимально інтуїтивною для адміністратора магазину.

Загалом документація розробляється з урахуванням принципів доступності, послідовності та практичної орієнтації, що дозволить користувачам ефективно впроваджувати та використовувати платіжне рішення у своїй e-commerce інфраструктурі.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було здійснено безпосередню реалізацію розробленого платіжного плагіна для платформи Magento 2 відповідно до визначених вимог. Розробка велась із дотриманням архітектурних

принципів системи, стандартів безпеки та з урахуванням кращих практик модульної розробки.

Було реалізовано повнофункціональний платіжний модуль, що забезпечує ініціацію транзакцій, взаємодію з платіжним шлюзом через API, обробку повідомлень про зміну статусу платежу, підтримку багатомовного інтерфейсу та конфігурацію з боку адміністратора. У рамках розробки також було впроваджено можливість ручного захоплення коштів, логування транзакцій та перенаправлення користувача залежно від результату оплати. Особливу увагу приділено тестуванню модуля, включаючи як модульні, так і інтеграційні тести у середовищі Magento 2. Це дозволило виявити та усунути логічні похибки ще до етапу впровадження, забезпечивши стабільну та передбачувану поведінку системи у різних сценаріях.

Розробка структури взаємодії з базою даних проводилась у межах архітектури Magento, із використанням репозиторіїв, ресурсних моделей та підтримкою розширення стандартних таблиць. Це дозволило інтегрувати нові функціональні можливості без порушення цілісності основної системи.

Було створено інсталяційний пакет модуля з підтримкою Composer, що дозволяє швидко встановити плагін у будь-яке Magento-середовище без додаткових налаштувань. Усі компоненти відповідають стандартам PSR-4, що забезпечує сумісність і легкість супроводу.

Завершальним етапом стала розробка документації, що містить інструкції з інсталяції, конфігурації та використання плагіна. Документація дозволяє адміністраторам магазину самостійно налаштовувати модуль без залучення розробника.

Таким чином, розроблений платіжний плагін є завершеним, стабільним і готовим до використання в реальному електронному бізнесі. Він відповідає актуальним вимогам безпеки, підтримує гнучке налаштування та може бути масштабований під потреби конкретного підприємства.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було комплексно досліджено проблематику розробки платіжного плагіна для платформи Magento 2, з урахуванням актуальних викликів у сфері електронної комерції, вимог до інформаційної безпеки та потреб кінцевих користувачів. Поставлені завдання дозволили реалізувати структурований підхід до вирішення обраної теми на теоретичному, методологічному та практичному рівнях.

Було проаналізовано сучасний стан електронної комерції, визначено її основні виклики та перспективи розвитку, а також здійснено порівняльний аналіз платформи Magento 2 з альтернативними рішеннями, такими як Shopify і WooCommerce, що дозволило обґрунтувати доцільність вибору Magento 2 для реалізації платіжного плагіна.

Сформульовано функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення, враховуючи особливості архітектури Magento 2, вимоги безпеки та потреби ключових користувачів – адміністраторів магазину та покупців.

Проведено аналіз інструментів розробки, обґрунтовано вибір мов програмування PHP і JavaScript, інструментів Composer, PHPUnit, а також архітектурних підходів, що дозволили створити масштабований, підтримуваний і безпечний плагін.

Реалізовано основну логіку плагіна, зокрема ініціацію транзакцій, взаємодію з API платіжного шлюзу, обробку вебхуків, перенаправлення користувача після оплати та адміністративну панель налаштувань, що забезпечує зручність керування модулем.

Проведено ретельне модульне й ручне тестування плагіна в середовищі Magento 2 з метою перевірки стабільності, коректної роботи та відповідності вимогам функціональності на всіх етапах взаємодії з системою.

Інтегровано модуль із базою даних Magento 2 згідно зі стандартами платформи – з використанням ресурсних моделей, репозиторіїв, розширених

полів і системи індексації, що забезпечило оптимальну продуктивність і цілісність даних.

Створено інсталяційний Composer-пакет для спрощення підключення плагіна до сторонніх проєктів, а також забезпечено можливість автоматичної реєстрації та подальшого оновлення модуля у межах Magento 2.

Розроблено повну технічну документацію до плагіна, яка містить системні вимоги, інструкції з встановлення, налаштування та експлуатації модуля, а також рекомендації для адміністраторів щодо підтримки та оновлень, що сприяє ефективному впровадженню рішення в реальні проєкти.

Таким чином, усі поставлені в процесі дослідження завдання були успішно виконані, а результати кваліфікаційної роботи підтвердили як доцільність обраної теми, так і практичну цінність створеного програмного рішення для сучасного ринку електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проектів Вікімедіа. PHP – Вікіпедія. URL: https://uk.wikipedia.org/wiki/PHP (дата звернення: 03.02.2025).

2. JavaScript – MDN. MDN Web Docs. URL: https://developer.mozilla.org /en-US/docs/Web/JavaScript (дата звернення: 08.02.2025).

3. How to improve usability for Magento 2 add to cart process. Inchoo.net. URL: http://inchoo.net/magento-2/improve-usability-magento-2-add-cart-process/ (дата звернення: 11.02.2025).

4. Ukrainian W. Уроки від W3Schools українською онлайн. W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів. URL: https://w3schoolsua.github.io/css/index.html (дата звернення: 04.03.2025).

5. I18n що це і навіщо потрібна інтернаціоналізація в розробці. FoxmindEd. URL: https://foxminded.ua/i18n-shcho-tse/ (дата звернення: 12.03.2025).

6. Translation dictionaries and language packages – Adobe Commerce. Experience League – Adobe. URL: https://experienceleague.adobe.com/en/docs/commerce-operations/configuration-guide/cli/localization (дата звернення: 18.03.2025).

7. Fomenko I. Dependecy injection та Singletone. Все, що треба знати, і ще трошки. Medium. URL: https://medium.com/@ivanfomenko/dependecy-injection-та-singletone-все-що-треба-знати-і-ще-трошки-2e738373eb94 (дата звернення: 19.03.2025).

8. Unit Testing in Magento 2. Installation Guide for Mage-OS – Mage-OS – Community driven eCommerce. URL: https://devdocs.mage-os.org/docs/main/unit-testing/ (дата звернення: 22.03.2025).

9. How to Create Database Table in Magento 2? Best Managed Magento Hosting Platform in 2025. URL: https://www.mgt-commerce.com/tutorial/database-table-in-magento-2/ (дата звернення: 27.03.2025).

10. Searching with repositories – commerce PHP extensions. Adobe Developer Website. URL: https://developer.adobe.com/commerce/php/development/components/searching-with-repositories/ (дата звернення: 02.04.2025).

11. How to add comment history to order in magento 2 – Magepow Blog. Magepow Blog. URL: https://magepow.com/blog/how-to-add-comment-history-to-order-in-magento-2/?srsltid=AfmBOooZBBpIbF-Gwj4Plqhf_K33_pABMweoB6S5qQSI4nH-USGd79WL/ (дата звернення: 11.04.2025).

12. Учасники проектів Вікімедіа. Composer – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Composer (дата звернення: 17.04.2025).

13. Nidhi Patel. How to setup configuration setting for a module in Magento 2? Aureate Labs. URL: https://aureatelabs.com/blog /how-to-setup-configuration-setting-for-a-module-in-magento-2/ (дата звернення: 19.04.2025).

14. Create a module – Adobe Commerce. Experience League – Adobe. URL: https://experienceleague.adobe.com/en/docs/commerce-learn/tutorials/backend-development/create-module (дата звернення: 23.04.2025).

15. Учасники проектів Вікімедіа. README – Вікіпедія. URL: https://uk.wikipedia.org/wiki/README (дата звернення: 29.04.2025).