Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

ІНТЕГРАЦІЯ ПЛАТІЖНОЇ СИСТЕМИ PAYPAL З E-COMMERCE СИСТЕМАМИ НА ОСНОВІ CMS PRESTASHOP
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Чантурія Т.З.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Чантурії Тамарі Зурабівні
(прізвище, ім'я, по батькові)

1. Тема роботи Інтеграція платіжної системи PayPal з e-commerce системами на основі CMS PrestaShop

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, документація електронної платіжної системи PayPal, система керування вмістом з відкритим кодом PrestaShop.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз електронної платіжної системи PayPal.
2. Аналіз системи керування вмістом PrestaShop.
3. Огляд методів інтеграції платіжної системи PayPal з e-commerce системами.
4. Програмна реалізація модуля для інтеграції платіжної системи PayPal з системою керування вмістом PrestaShop.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми інтеграції електронної платіжної системи PayPal з e-commerce системами, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних вимог до модуля	21.04.23-30.04.23	
5	Розробка модуля	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	05.06.23	
12	Попередній захист кваліфікаційної роботи	05.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Кобилін О.А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 71 с., 23 рис., 40 джерел.

РОЗРОБКА МОДУЛЯ, ІНТЕГРАЦІЯ ПЛАТІЖНОЇ СИСТЕМИ, PAYPAL, СИСТЕМА КЕРУВАННЯ ВМІСТОМ, CMS PRESTASHOP.

Об’єктом роботи є інтеграція платіжної системи PayPal з e-commerce системами.

Метою роботи є розробка модуля, який дозволяє інтегрувати платіжну систему PayPal з e-commerce системами керування вмістом (CMS) PrestaShop для інтернет-магазинів.

Для реалізації зазначеної мети було використано мову програмування PHP, JavaScript. Було проведено роботу з базами даних MySQL. Також були розроблені HTML-сторінки та таблиці стилів CSS.

Під час розробки на робочому місці було встановлено групу компонентів LAMP (Linux, Apache2, MySQL, PHP). Для доступу до бази даних – phpMyAdmin, а для програмування – інтегроване середовище розробки PhpStorm. В якості системи управління версіями було використано Git.

У результаті роботи проведено дослідження нових шаблонів проєктування та здійснена програмна реалізація модуля інтеграції платіжної системи PayPal з e-commerce системами на основі CMS PrestaShop.

MODULE DEVELOPMENT, PAYMENT GATEWAY INTEGRATION, PAYPAL, CONTENT MANAGEMENT SYSTEM, CMS PRESTASHOP.

The object of the research is PayPal payment gateway integration with e-commerce systems.

The aim of the research is to develop a module integrating the PayPal payment system with the e-commerce Content Management System PrestaShop.

The module used PHP and JavaScript programming languages and MySQL databases. HTML pages and CSS style sheets were developed to reach the goals.

During the development process, the following tools were installed in the workplace and used: components of the LAMP stack (Linux, Apache2, MySQL, PHP), phpMyAdmin for managing the Databases, Integrated Development Environment PhpStorm for PHP programming. Git was used as a version control system.

As a result of the work, actioned research of the new development templates and implemented software implementation of the module integrating PayPal payment system with e-commerce CMS PrestaShop.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CMS − Content Management System (система керування вмістом)

ООП − об’єктно-орієнтоване програмування

REST − Representational State Transfer (передача репрезентативного стану)

API − Application Programming Interface (прикладний програмний інтерфейс)

SDK − Software Development Kit (набір із засобів розробки)

IDE − Integrated development environment (інтегроване середовище розробки)

PCI − Payment Card Industry (індустрія платіжних карток)

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

HTTP − Hypertext Transfer Protocol (протокол передачі гіпертексту)

HTTPS − Hypertext Transfer Protocol Secure (захищений протокол передачі гіпертексту)

OSL − Open Software License (ліцензія на відкрите програмне забезпечення)

SSL − Secure Sockets Layer (рівень захищених сокетів)

TLS − Transport Layer Security (захист на транспортному рівні)

GCP − Google Cloud Platform

ВСТУП

За останні десятиліття в комп'ютерному світі відбулася справжня революція. Ця революція є вибухом інтернету, який став головним механізмом спілкування, дослідження, обміну інформацією та надання послуг.

Інтернет став незамінною частиною сучасного суспільства і використовується в різних сферах, таких як освіта, комунікації, комерція, розваги та багато інших. Він змінив спосіб, яким ми спілкуємося, отримуємо інформацію і здійснюємо роботу. Інтернет став успішним прикладом того, як постійні інвестиції та відданість дослідженням та розробкам можуть сприяти створенню потужної обчислювальної інфраструктури, яка надає користувачам доступ до широкого спектру послуг та знань.

Сучасний бізнес не може існувати поза впливом інтернету. Існують різні види бізнесу в інтернеті, але об'єднує їх те, що це бізнес-процеси, які орієнтовані на інтеграцію виробників товару та їх реалізаторів, споживачів і постачальників. Інтернет дійсно відкриває безліч можливостей для створення онлайн-сервісів і доступу до них з будь-якої точки світу та на різних пристроях. Крім того, за допомогою програмних рішень та інтернет-технологій, багато бізнес-процесів можуть бути виконані в напівавтоматичному або повністю автоматичному режимі. Це дозволяє компаніям розширювати свої ринки і привертати більше клієнтів.

Електронні магазини, також відомі як інтернет-магазини або онлайн-магазини, є одним з найпопулярніших прикладів використання Інтернету в сфері бізнесу. Вони дозволяють компаніям продавати товари та послуги через Інтернет без необхідності фізичного присутності покупця в магазині.

У сучасній торгівлі в інтернеті вирішальну роль відіграють платіжні системи, які дозволяють підприємствам приймати платежі від клієнтів у всьому світі, виходячи за межі географічних кордонів. Саме тому я обрала всесвітньо відому платіжну систему PayPal для своєю кваліфікаційної роботи,

адже актуальність цієї системи та можливостей її інтеграції в час інтернету тільки зростає.

Крім того, для бізнеса дуже важливо правильно підтримувати вебсайти, володіючи інструментами, призначеними для цієї мети. Практичним, простим і дуже популярним способом серед продавців, які хочуть мати свої сайти електронної комерції, є використання систем керування вмістом. Процес вибору системи керування вмістом (CMS) часто є складною темою під час створення інтернет-магазину.

ІТ-відділ дуже часто шукає перевірене рішення, яке пропонує підтримку як з точки зору його використання, так і самої технологічної платформи. Також необхідно брати до уваги сумісність і взаємозв’язок з існуючими або додатковими рішеннями та інфраструктурою.

Маркетингові команди шукають зручний і простий у використанні інструмент, який пропонує їм повну автономію для досягнення своїх маркетингових цілей. Пропоновані функції, такі як можливості налаштування вмісту, часто є вирішальним елементом.

Для кваліфікаційної роботи було обрано систему керування вмістом з відкритим вихідним кодом PrestaShop, яка пропонує низку функцій та інструментів, має велику спільноту розробників, встановлюється на власному хостингі з повним доступом для налаштування та підтримки.

Актуальність роботи полягає у можливості інтеграції з провідними платіжними системами, такими як PayPal, для бізнесу в сфері електронної комерції, який має намір підвищити свою конкурентоспроможність і забезпечити зручність і безпеку для своїх клієнтів.

1 ОГЛЯД ОСНОВНИХ ЕЛЕКТРОННИХ СИСТЕМ ПЛАТЕЖІВ ТА МЕТОДІВ ЇХ ІНТЕГРАЦІЙ З E-COMMERCE СИСТЕМАМИ

1.1 Що таке електронна платіжна система

Електронна платіжна система – це електронний сервіс, який дозволяє здійснення розрахунків між фінансовими установами, бізнес організаціями та інтернет користувачами в процесі купівлі продажу товарів і послуг через інтернет. Споживачі активно використовують такі системи щоб купувати товари онлайн. Вони можуть використовувати різні типи методів онлайн-платежів, зокрема дебетові та кредитні картки, банківські перекази, мережевий банкінг і цифрові гаманці [1].

Продавець, в свою чергу, може безпечно перевіряти і приймати оплату від споживачів. Невід'ємною частиною онлайн-платіжної системи є еквайринг. Складова частина еквайрингу – перевірка номеру кредитної картки та кредитної історії платника, а також підтвердження банком, який обслуговує цю кредитку, можливості здійснити платіж на запитувану суму в зазначеному напрямку.

На сьогоднішній день існує декілька десятків платіжних систем, однієї із найпопулярніших є платіжна система PayPal (рис. 1.1).

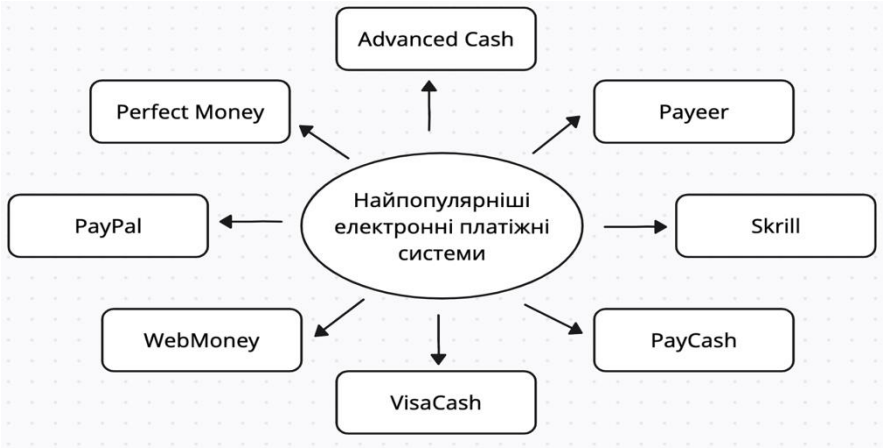

Рисунок 1.1 – Найпопулярніші електронні платіжні системи

1.2 Сервіси для оптимізації процесу приймання оплати

Для подальшої роботи необхідно спочатку зробити обзор існуючих рішень для оптимізації процесу приймання оплати. Для полегшення електронних транзакцій використовуються платіжні шлюзи, які діють як міст між фінансовую установою клієнта та банком продавця, що дозволяє безпечно обмінюватися платіжною інформацією.

Принцип роботи таких сервісів у тому, що коли клієнт робить покупку онлайн або через мобільний застосунок, платіжний шлюз шифрує його платіжні дані та надсилає інформацію до банку продавця для обробки. Після авторизації платежу сервіс надсилає повідомлення про підтвердження продавцю, який потім може завершити транзакцію.

Платіжні шлюзи можуть підтримувати різні методи оплати, включаючи кредитні та дебетові картки, цифрові гаманці та банківські перекази. Зазвичай вони стягують комісію за кожну транзакцію у вигляді відсотка від суми транзакції або фіксовану ставку за транзакцію.

Сервіси для оптимізації процесу приймання оплати розроблені використанням комбінації мов програмування, програмних засобів і API (інтерфейсів прикладного програмування).

Процес розробки таких систем зазвичай включає наступні етапи:

– збір вимог. Першим кроком є розуміння бізнес-вимог платіжної системи. Наприклад, типи підтримуваних методів оплати, необхідні функції безпеки та параметри інтеграції з різними торговими платформами;

– дизайн архітектури. На основі вимог проходить розробка архітектури платіжного шлюзу, включаючи схему бази даних, інфраструктуру сервера та протоколи зв’язку;

– реалізація. Платіжна система, або платіжний шлюз, реалізується за допомогою мов програмування, таких як Java, Python або PHP. Команда розробників використовує програмні інструменти, такі як інтегровані середовища розробки (IDE), системи контролю версій і інфраструктури тестування, щоб забезпечити якість коду;

– інтеграція. Після того, як завершується розробка платіжної системи, він інтегрується з вебсайтом продавця або мобільним застосунком за допомогою API або SDK (комплектів розробки програмного забезпечення);

– тестування та налаштування тестового середовища. Платіжна система ретельно перевіряється з використанням різних методологій тестування, таких як модульне тестування, інтеграційне тестування та тестування продуктивності. Після тестування в Sandbox середовищі платіжний шлюз розгортається у робочому середовищі;

– технічне обслуговування та оновлення. Платіжні системи вимагають постійного обслуговування та оновлень для забезпечення їх безпеки та функціональності. Це передбачає виправлення помилок, додавання нових функцій і підтримку найновіших стандартів платіжної галузі.

Розробка платіжного шлюзу вимагає досвіду розробки програмного забезпечення, безпеки та обробки платежів. Це може бути складним і трудомістким процесом, але він важливий для підприємств, які хочуть приймати платежі онлайн [2].

Далі розглянемо приклади систем для оптимізації процесу приймання оплати.

1.2.1 Stripe

Stripe – це платформа обробки онлайн-платежів, яка дозволяє компаніям приймати платежі онлайн. У даний момент це один із найпопулярніших платіжних шлюзів для вебсайтів електронної комерції та мобільних додатків, який має сертифікацію PCI (перший найбільш високий рівень серед всіх існуючих платіжних систем).

Stripe є популярним вибором для багатьох бізнесів завдяки своїй простоті в використанні та розширеним можливостям.

Ось деякі ключові особливості Stripe:

– обробка платежів. Stripe дозволяє компаніям приймати платежі від клієнтів за допомогою різних методів оплати, включаючи кредитні та дебетові картки, цифрові гаманці та банківські перекази [3];

– керування підпискою. Компанії можуть використовувати Stripe для керування регулярними платежами за підписку, наприклад, місячними або річними підписками на програмне забезпечення;

– запобігання шахрайству. Stripe містить вбудовані інструменти виявлення шахрайства, які допомагають запобігти шахрайським транзакціям і мінімізувати повернення платежів;

– міжнародні платежі. Stripe підтримує платежі в понад 135 валютах і дозволяє підприємствам приймати платежі від клієнтів з усього світу;

– інструменти для розробки. Stripe пропонує API та комплекти розробки програмного забезпечення (SDK), щоб допомогти розробникам з інтеграціями обробки платежів у свої вебсайти та мобільні програми;

– звітність і аналітика. Stripe надає детальну звітність і аналітику, щоб допомогти компаніям відстежувати свої продажі та платіжну діяльність;

– настроюваний процес оплати. Компанії можуть налаштувати свій досвід оплати за допомогою Stripe, включаючи брендинг і можливість приймати платежі без перенаправлення клієнтів на окрему платіжну сторінку.

Stripe стягує комісію за кожну транзакцію, а також додаткові комісії за деякі функції, такі як підписки або міжнародні платежі. Більшість клієнтів користуються модулями для інтеграції платіжної системи Stripe, які у свою чергу розміщують iFrame Stripe на своїх вебсайтах.

iFrame Stripe – це функція, яка дозволяє компаніям безпечно вставляти форму оформлення замовлення Stripe безпосередньо на свій вебсайт, без необхідності перенаправляти клієнтів на окрему сторінку оплати. iFrame відображається як невелике вікно на сторінці оформлення замовлення продавця, що дозволяє клієнтам вводити платіжну інформацію та завершувати транзакцію, не залишаючи вебсайт продавця.

Stripe iFrame надає кілька переваг, зокрема:

– безпека. iFrame розроблено для захисту даних клієнтів шляхом безпечної передачі платіжної інформації безпосередньо на сервери Stripe, не проходячи через вебсайт продавца;

– налаштування. Продавці можуть налаштувати зовнішній вигляд iFrame відповідно до фірмової символіки свого вебсайту, забезпечуючи клієнтам зручну процедуру оплати;

– зменшення кількості залишень. Утримуючи клієнта на вебсайті продавця протягом усього процесу оформлення замовлення, iFrame може допомогти зменшити кількість залишень у кошику та збільшити конверсію;

– мобільна оптимізація. iFrame зручний для мобільних пристроїв, і до нього можна легко отримати доступ зі смартфонів і планшетів, що робить iFrame зручним варіантом для клієнтів, які віддають перевагу робити покупки на мобільних пристроях.

Щоб використовувати Stripe iFrame, торговці повинні інтегрувати його на свій вебсайт за допомогою API Stripe [4] або попередньо створеного плагіна чи розширення. Після інтеграції iFrame можна використовувати для одноразових платежів, підписок та інших типів платежів, які підтримує Stripe.

1.2.2 Braintree

Braintree – це платформа обробки онлайн-платежів, багато в чому схожа на Stripe. Ця платформа була заснована у 2007 році та придбана PayPal у 2013 році. Braintree пропонує широкий спектр послуг обробки платежів, включаючи платежі кредитними та дебетовими картками, цифрові гаманці та банківські перекази.

Braintree сертифіковано як постачальник послуг, сумісний з PCI DSS рівня 1, що означає, що він відповідає суворим вимогам безпеки, встановленим радою стандартів безпеки індустрії платіжних карток. Braintree також використовує низку розширених функцій безпеки, включаючи токенізацію, виявлення та запобігання шахрайству та автентифікацію 3D Secure [5].

Принципи роботи Braintree та Stripe дуже схожі. Деякі ключові відмінності між Braintree і Stripe:

– материнська компанія. Braintree належить PayPal, тоді як Stripe є незалежною компанією;

– методи оплати. Хоча і Braintree, і Stripe підтримують широкий спектр методів оплати, Braintree приділяє особливу увагу мобільним платежам і цифровим гаманцям із підтримкою Apple Pay, Google Pay і Venmo;

– функціональність платіжного шлюзу. Braintree надає повне рішення платіжного шлюзу, що означає, що користувачі можуть використовувати Braintree для обробки повного процесу платежу, від оформлення замовлення до розрахунку. З іншого боку, Stripe пропонує безпосередньо послуги обробки платежів і покладається на сторонні шлюзи для деяких функцій, таких як виявлення шахрайства;

– інтерфейс користувача. Braintree має інший інтерфейс користувача, ніж Stripe, і деяким користувачам може бути легшим у навігації;

– комісії. структура комісії Braintree дещо відрізняється від структури Stripe: фіксована ставка за транзакцію плюс додаткова комісія у відсотках. Braintree може бути кращим варіантом для підприємств із великим обсягом транзакцій з невеликою вартістю.

Загалом і Braintree, і Stripe є авторитетними платформами обробки платежів, які пропонують широкий спектр функцій і переваг для бізнесів. Вибір між Braintree і Stripe залежатиме від конкретних потреб бізнесу, технічної складності і бюджету.

Щоб використовувати Braintree, компанії повинні інтегрувати платіжний шлюз у свій вебсайт або мобільний застосунок за допомогою API Braintree або комплектів розробки програмного забезпечення (SDK). API та SDK доступні різними мовами програмування, включаючи Java, Ruby, Python і PHP, що дозволяє компаніям легко інтегрувати Braintree у свій існуючий стек технологій.

1.3 Міжнародна електронна платіжна система PayPal

PayPal – це американська система електронних платежів, заснована в грудні 1998 року Пітером Тхілом і Максом Левчіном (Peter Thiel і Max Levehin), яку використовують як платіжну систему для малого бізнесу і торгівлі онлайн. Ця система – беззаперечний лідер у сфері онлайн-платежів, електронний гаманець, який дозволяє покупцям і продавцям надсилати та отримувати платежі онлайн. Користувачі системи можуть переказувати кошти безпосередньо на рахунок PayPal або прив'язувати до нього свою банківську картку, отримуючи таким чином найвищий рівень захисту особистих і банківських даних (рис. 1.2).

Рисунок 1.2 – Міжнародна електронна платіжна система PayPal

Paypal є першим платіжним посередником в Інтернеті. Його використовують найбільші світові компанії: банки, професіонали, асоціації тощо.

Це надбезпечний спосіб оплати. Споживачі мають можливість платити та отримувати гроші через свій обліковий запис Paypal. У цих випадках споживачі повинні ввести свої банківські реквізити, а також номери кредитної картки [6].

Переваги PayPal:

– безпека. Користувачі роблять покупки, не повідомляючи свої банківські реквізити третім особам. API PayPal автоматично перенаправляє користувачів на захищену сторінку сервісу, на стороні якого проводяться всі

транзакції. Окрім того, PayPal дотримується стандартів безпеки PCI, прийняті Visa, MasterCard, American Express та іншими компаніями;

– швидкість. Платежі здійснюються майже миттєво та набагато швидше, ніж надсилання чеків або готівки;

– вигідність. PayPal, як і інші платіжні сервіси, стягує комісію за проведення операцій. Розмір комісії залежить безпосередньо від обсягу продажів (чим він більший, тим менший відсоток). Середня ставка – 3 %. Це рівноцінна сума в порівнянні з іншими сервісами, але користувачи не повинні платити за підписку кожен місяць;

– простота. Відправлення платежу здійснюється на електронну адресу;

– глобальність. PayPal підтримує платежі в шістнадцяти валютах, і більше 100 мільйонів рахунків відкрито в 202 країнах і регіонах по всьому світу;

– інноваційність. PayPal постійно розробляє нові послуги та інструменти для розвитку вашого бізнесу в Інтернеті.

Окрім основних платіжних послуг, PayPal пропонує низку додаткових функцій і послуг, таких як виставлення рахунків, підписка та торгові послуги для компаній. PayPal також пропонує мобільний застосунок, що полегшує користувачам надсилання та отримання платежів у дорозі.

PayPal можна інтегрувати у вебсайти та програми різними методами [7], зокрема:

– кнопки PayPal. PayPal надає готові кнопки, які можна додавати на вебсайти за допомогою лише кількох рядків коду. Ці кнопки дозволяють користувачам здійснювати платежі чи пожертви одним натисканням кнопки, не залишаючи вебсайт;

– PayPal Checkout. PayPal Checkout – це більш настроюваний метод інтеграції, який дозволяє компаніям додавати кнопку оплати на свій вебсайт, яка відкриває платіжну сторінку PayPal у новому вікні. Цей метод дозволяє більше налаштувати процес оформлення замовлення, включаючи додавання спеціальних полів і брендування;

– PayPal REST API [8]. PayPal також надає набір REST API, які розробники можуть використовувати для створення індивідуальних інтеграцій. Ці API дозволяють розробникам інтегрувати можливості обробки платежів PayPal безпосередньо на свій вебсайт або програму. У рамках кваліфікаційної роботи також буде використано PayPal REST API для розробки модуля;

– PayPal SDK [9]. PayPal пропонує пакети розробки програмного забезпечення (SDK) для різних мов програмування, включаючи PHP, Java та .NET. Ці пакети SDK містять попередньо зібраний код, який розробники можуть використовувати для швидкої інтеграції PayPal у свої програми;

– інтеграція сторонніх розробників. Багато популярних платформ електронної комерції, таких як Shopify і WooCommerce, мають вбудовану інтеграцію з PayPal. Це дозволяє компаніям, які використовують ці платформи, легко приймати платежі через PayPal без необхідності виконувати будь-які спеціальні інтеграційні роботи.

Загалом PayPal пропонує низку варіантів інтеграції, які можна пристосувати до потреб підприємств і розробників будь-якого рівня кваліфікації.

Підсумовуючи, можна сказати, що PayPal – всесвітньо відомий та дуже надійний платіжний інструмент, з яким знайома більшість людей у світі.

У рамках кваліфікаційної роботи було проведено роботу саме з електронною системою платежів PayPal.

1.4 Методи інтеграцій платіжних систем з e-commerce системами

Інтеграція електронних платіжних систем з e-commerce системами може здійснюватися різними методами, залежно від потреб бізнесу і можливостей платіжної системи [10]. Ось декілька типових методів інтеграції, деякі із яких вже було частково описано вище:

– вбудовані кнопки. Більшість електронних платіжних систем, таких як PayPal, Stripe і Braintree, надають код для вбудованих кнопок оплати на сторінки товарів або послуг на e-commerce сайті. Кнопки можуть бути різних типів, наприклад, кнопки «Купити зараз», «Додати в кошик» або «Оплатити»;

– API. Деякі платіжні системи, такі як Stripe, надають API, які дозволяють розробникам інтегрувати платіжну систему безпосередньо в свій e-commerce сайт. API надають доступ до функцій, таких як створення транзакцій, отримання інформації про статус платежів, керування рахунками користувачів і багато іншого. Цей метод передбачає написання спеціального коду для підключення платіжного шлюзу до CMS, що може забезпечити більшу гнучкість і можливості налаштування;

– платіжні шлюзи. Платіжні шлюзи, такі як Authorize.Net і 2Checkout, дозволяють інтегрувати кілька різних платіжних систем в один e-commerce сайт. Ці шлюзи надають міжфейс для взаємодії з різними платіжними системами, що дозволяє бізнесу працювати з кількома провайдерами платіжних послуг, не змінюючи коду свого сайту;

– підписки. Багато e-commerce сайтів пропонують можливість підписки на товари або послуги, що передбачає регулярні платежі. Деякі платіжні системи, такі як PayPal і Stripe, надають функцію підписок, яка дозволяє автоматично здійснювати повторювані платежі за підпискою за розкладом;

– використання модулів. Деякі електронні платіжні системи мають готові модулі для популярних e-commerce платформ, таких як WooCommerce та Magento. Ці модулі дозволяють користувачам швидко налаштувати оплату через платіжну систему у своєму інтернет-магазині, не потребуючи написання власного коду. Модулі можуть спростити процес інтеграції, надаючи попередньо зібраний код, який можна легко налаштувати. Мета цієї кваліфікаційної роботи – написання такого модуля для e-commerce платформи PrestaShop;

– інтеграція вручну. якщо платіжний шлюз не пропонує модулів або API для CMS, його все одно можна інтегрувати вручну, додавши спеціальний

код до шаблонів або сторінок CMS [11-14]. Цей спосіб може бути більш складним і трудомістким, але він дозволяє повністю контролювати процес оплати.

Загалом вибір методу інтеграції залежить від потреб користувача та можливостей платіжного шлюзу. Готові рішення та плагіни можуть бути корисними для користувачів, які хочуть швидко налаштувати оплату, тоді як API більше підходять для розробників, які хочуть налаштувати процес оплати.

Незалежно від методу інтеграції, важливо переконатися, що платіжний шлюз є безпечним і відповідає всім необхідним вимогам відповідності, таким як PCI DSS. Також важливо ретельно протестувати інтеграцію перед запуском, щоб переконатися, що платежі обробляються правильно та дані передаються безпечно.

1.5 Постановка задачі

Останнім кроком кожної покупки і найважливішим етап для бізнесу є оплата. Дуже важливо, щоб методи оплати, які пропонуються користувачам, були добре відомі та безпечні. Зараз існує велика кількість інтернет-магазинів, які пропонують користувачам дуже схожі або навіть ідентичні продукти. У боротьбі за нових клієнтів важлива кожна дрібниця, а процес оформлення онлайн замовлення має чималий вплив на репутацію бренду та дохід. Людина, швидше за все, зробить покупку на вебсайті, де їй пропонують уже відомий, зручний і безпечний метод оплати.

Отже, можна визначити певні вимоги до інструменту для інтеграції платіжної системи, наприклад:

– спосіб оплати повинен бути добре відомий користувачеві;

– спосіб оплати повинен бути зручним і займати якомога менше часу на здійснення платежу;

– спосіб оплати повинен бути безпечним і надійним;

– розроблений інструмент повинен швидко встановлюватися, легко налаштовуватися та бути інтуїтивно зрозумілим.

Розробляючи такий інструмент для інтеграції платіжної системи PayPal, важливо дотримуватися найкращих практик безпеки та відповідності, наприклад використовувати HTTPS і придержуватися вимог PCI DSS.

Системою керування вмістом для інтеграції інтернет-магазинів з електронною платіжною системою PayPal було обрано PrestaShop.

Об’єктом роботи є інтеграція платіжної системи PayPal з e-commerce системами.

Метою роботи є розробка модуля, який дозволяє інтегрувати платіжну систему PayPal з e-commerce системами керування вмістом (CMS) PrestaShop для інтернет-магазинів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз електронної платіжної системи PayPal;

– провести аналіз системи керування вмістом (CMS) PrestaShop;

– провести аналіз методів інтеграції PayPal;

– провести аналіз методів безпеки;

– розробити модуль для інтеграції PayPal з інтернет-магазинами, які використовують CMS PrestaShop;

– ретельно перевірити правильність роботи модуля.

2 РОЗРОБКА МОДУЛЯ ДЛЯ ІНТЕГРАЦІЇ PAYPAL

2.1 Вимоги до інтеграції платіжної системи PayPal

Платіжна система PayPal існує вже більше 20 років. За цей час світ електронної комерції став дуже популярним. PayPal, як лідер ринку, має залишатися гнучким, відкритим до нових ідей і пропонувати своїм користувачам широкий вибір фінансових рішень.

На цьому етапі важливо розглянути основні вимоги до інтеграції платіжної системи PayPal.

2.1.1 Різні функціональні можливості залежно від країни користувача

На жаль, не всі продукти PayPal доступні для нас. Це пов’язано з законодавством та операційними процедурами фінансових установ країни, в якій відбувається фінансова операція. Наприклад, у деяких країнах можуть бути обмеження щодо виведення коштів з PayPal-рахунку на банківські рахунки або картки, обмеження щодо відправки платежів або обмеження щодо отримання платежів в певних валютах.

Це означає, що під час розробки модуля інтеграції платіжної системи PayPal потрібно пропонувати різні функціональні можливості залежно від країни користувача та країни, в якій він здійснює операції.

PayPal працює в багатьох країнах світу, і хоча основні функції однакові, можуть бути деякі відмінності в тому, як послуга пропонується в кожній країні. Нижче наведено ключові відмінності, пов’язані із використанням PayPal у різних країнах:

– прийнятні валюти. PayPal підтримує багато різних валют, але перелік валют, які приймаються, може відрізнятися залежно від країни. Наприклад, у

деяких країнах PayPal може підтримувати лише місцеві валюти, а в інших він може підтримувати ширший діапазон валют;

– доступні способи оплати. Способи оплати, доступні в кожній країні, також можуть відрізнятися. Наприклад, у деяких країнах PayPal може підтримувати місцеві банківські перекази, тоді як в інших він може підтримувати лише платежі кредитною або дебетовою карткою;

– комісії. Комісії, які стягуються з користувача під час використання PayPal, також можуть відрізнятися залежно від країни. Наприклад, плата за отримання платежів може відрізнятися залежно від країни, у якій ви перебуваєте;

– спеціальні правила. Нормативне середовище в кожній країні також може впливати на роботу PayPal. Наприклад, у деяких країнах можуть існувати спеціальні правила щодо використання PayPal для певних типів транзакцій;

– служба підтримки клієнтів. Рівень і якість підтримки клієнтів, яку пропонує PayPal, також може відрізнятися залежно від країни. У деяких країнах можуть існувати спеціальні групи підтримки клієнтів або доступна підтримка місцевою мовою.

Важливо пам’ятати про ці відмінності під час використання PayPal та розробки модуля для інтеграції.

2.1.2 Гнучкість до змін

Важливо також враховувати, що згодом до модуля можуть бути додані нові функції, оскільки PayPal може оновити свій REST API або випустити новий продукт, який слід також впровадити в модуль. Таким чином, дуже важливо, щоб модуль був гнучким до змін і добре масштабованим [15].

PayPal REST API – це вебінтерфейс API, який надає можливість легко інтегрувати функцію обробки платежів PayPal у вебсайт користувача. REST

означає Representational State Transfer – стиль програмної архітектури, який використовується для створення вебслужб.

Використовуючи PayPal REST API, розробники можуть створювати налаштовані процедури оформлення замовлення, керувати платежами та транзакціями та відстежувати інформацію про клієнтів. API підтримує різні мови програмування, включаючи Java, .NET, PHP, Ruby та Python.

Щоб використовувати PayPal REST API, розробникам потрібно створити обліковий запис розробника PayPal і отримати облікові дані API. API пропонує як середовище ізольованого програмного середовища для тестування, так і робоче середовище для живих транзакцій [16].

PayPal періодично оновлює PayPal REST API, зазвичай для додавання нових функцій, підвищення продуктивності або виправлення помилок. Частота та обсяг цих оновлень можуть відрізнятися залежно від потреб.

Щоб розробити модуль, який враховує ці зміни, необхідно ознайомитися з останньою версією документації, примітками до випуску та журналами змін, наданих PayPal. Ці ресурси допоможуть у розумінні, які зміни були внесені та як вони можуть впливати на модуль.

Важливою практикою щодо контролю версій є підтримка окремих гілок для кожної версії модуля. Таким чином, зміні внесені в кожну версію не будуть впливати на інші версії, полегшуючи підтримку сумісності з PayPal REST API.

Крім того, можна розглянути можливість використання набору програмного забезпечення (SDK), наданого PayPal [17]. PayPal SDK надають спрощений і узгоджений інтерфейс для розробників, абстрагуючись від багатьох низькорівневих деталей REST API. Це може допомогти заощадити час і зменшити складність під час розробки функцій обробки платежів.

Загалом, важливо бути в курсі змін в PayPal REST API, розуміти важливість гнучкості модуля та завчасно вживати заходи для забезпечення сумісності змін в PayPal REST API з модулем. Таким чином, модуль буде залишатися функціональним і безпечним протягом тривалого часу.

2.1.3 Принципи SOLID

Щоб полегшити технічне обслуговування програми, код програми має бути легким для читання та застосовувати принципи SOLID [18]. SOLID – це абревіатура п’яти основних принципів проєктування які широко використовуються в об’єктно-орієнтованому програмуванні для створення надійних, гнучких програмних систем, які можна підтримувати:

– single responsibility principle: єдина відповідальність – принцип єдиної відповідальності. Цей принцип гарантує, що клас зосереджений і не має надто багато обов’язків, які можуть ускладнити його модифікацію та підтримку;

– open closed principle: відкрито-закрито – принцип відкриття/закриття. Програмні об’єкти мають бути відкритими для розширення, але закритими для модифікації. Цей принцип заохочує використання абстракції та поліморфізму, щоб дозволити додавати нові функції без зміни існуючого коду;

– liskov substitution principle: Заміщення Ліскова – принцип заміщення Барбари Лісков. Підтипи повинні бути замінними для своїх базових типів. Цей принцип гарантує, що похідний клас може бути використаний замість свого базового класу, не спричиняючи помилок або неочікуваної поведінки;

– interface segregation principle: interface segregation – принцип поділу інтерфейсу. Користувачі не повинні залежати від інтерфейсів, які вони не використовують. Цей принцип заохочує використання менших і більш сфокусованих інтерфейсів замість великих інтерфейсів загального призначення;

– dependency inversion principle: dependency inversion – принцип інверсії залежностей. Модулі високого рівня не повинні залежати від модулів низького рівня. Обидва типи мають залежати від абстракцій. Цей принцип заохочує використання ін’єкції залежностей та інверсії керування, щоб зробити програмні системи більш гнучкими та легшими для модифікації.

2.1.4 Підтримка різних версій

Важливо зауважити, що розроблений модуль повинен працювати під різними версіями обраної системи керування вмістом (PrestaShop) та під різними версіями PHP, з різними версіями MySQL. Потрібно перевірити та звітувати, якщо середовище, де встановлено модуль, не має бібліотек, необхідних для його належної роботи [19-21].

Розробка модуля для інтеграції платіжної системи PayPal з CMS, який буде сумісним з різними версіями PHP, MySQL і безпосередньо CMS, вимагає ретельного планування, тестування та уваги до деталей.

Перш за все, потрібно визначити мінімальні вимоги до модуля, включаючи необхідні версії PHP, MySQL і CMS. Таку інформацію слід шукати безпосередньо в документації для CMS і PayPal API.

Під час розробки слід використовувати зворотно сумісний код, тобто код, сумісний зі старими версіями PHP, MySQL і CMS. Слід уникати використання функцій, які доступні лише в нових версіях цих технологій. Звісно, важливо також дотримуватися стандартів кодування та найкращих практик, щоб переконатися, що код чистий, добре організований і простий у обслуговуванні.

Модуль також слід перевіряти на кількох платформах з різними версіями PHP, MySQL і CMS, щоб переконатися в сумісності. Для цього можна використовувати віртуальні машини або контейнери для створення різних середовищ тестування.

Крім того, корисно використовувати бібліотеки та фреймворки, які забезпечують підтримку зворотної сумісності, наприклад розширення mysqli у PHP, яке забезпечує підтримку старіших версій MySQL [22].

Далі слід стежити за змінами в PHP, MySQL і CMS і заздалегідь планувати майбутні оновлення модуля. Це допоможе забезпечити сумісність модуля з новими версіями цих технологій.

Будь-які вимоги чи обмеження сумісності повинні бути зазначені в документації для модуля наряду з інструкцією зі встановлення. Це допоможе користувачам зрозуміти, як правильно встановити та використовувати модуль.

Виконуючи ці кроки, можна розробити модуль, який інтегрує PayPal з CMS, а також залишається сумісним з різними версіями PHP, MySQL і CMS. Тестування на кількох платформах і планування оновлень допоможуть переконатися, що модуль залишатиметься функціональним і безпечним протягом тривалого часу, навіть коли технології розвиваються та змінюються.

2.1.5 Можливість повернення коштів

Клієнти цінують гнучкість і впевненість, що вони можуть скасувати замовлення і отримати повернення грошових коштів у разі потреби. Це дозволяє їм зробити покупку без ризику, знаючи, що є можливість оформити повернення, якщо вони змінять свої плани або не будуть задоволені продуктом або послугою. Добре відомо, що наявність політики повернення коштів відображається на репутації бренду.

Таким чином, модуль повинен дозволяти не тільки здійснити оплату замовлення, але й повернути гроші при скасуванні замовлення. Потрібно враховувати, що задля повернення коштів потрібно спочатку перевіряти, чи відповідає транзакція вимогам для відшкодування відповідно до політики PayPal. Наприклад, деякі транзакції можуть бути недоступні для повернення або вже повернуті.

Сума для повернення повинна обчислюватися на основі початкової суми транзакції та будь-яких відповідних комісій. Обчислення може бути реалізовано за допомогою PayPal REST API або програмно за допомогою правил розрахунку комісії, наданих PayPal.

Отже, ініціація відшкодування може бути реалізовано за допомогою PayPal REST API. Важливо переконатися, що відшкодування обробляється

належним чином і що користувач отримує сповіщення про статус відшкодування.

Для коректного ведення бухгалтерського обліку, бухгалтерські записи, пов’язані з початковою транзакцією, повинні бути оновлені відповідним чином під час повернення коштів.

Крім того, користувачі повинні бути в курсі статусу транзакції та відшкодування.

2.1.6 Система журналювання

Модуль повинен мати систему реєстрації подій [23-28]. Це полегшить подальше обслуговування модуля та виявлення помилок.

Важливо зазначити, що PayPal надає власну систему журналювання, яка допомагає у діагностиці та вирішенні проблем під час інтеграції з PayPal REST API. Така система реєстрації подій надає детальну інформацію про запити та відповіді, якими обмінюються модуль та сервери PayPal.

Ось деякі ключові особливості системи реєстрації PayPal:

– кілька рівнів журналювання. Система журналювання PayPal забезпечує різні рівні докладності. Таким чином, можливо точно налаштувати кількість деталей, які будуть фіксуватися в журналах. Наприклад, можна реєструвати лише події високого рівня або фіксувати детальну інформацію про кожен виклик API;

– система журналювання дозволяє налаштувати формат виводу подій відповідно до ваших потреб. Ви можете виводити події у вигляді звичайного тексту або у структурованому форматі, наприклад JSON [29-31];

– пакети SDK PayPal розроблені для бездоганної інтеграції з популярними фреймворками журналювання, такими як log4j і log4net;

– доступ до журналів можна легко отримати програмним шляхом через PayPal REST API [8]. Це дозволяє включати дані журналу у власні системи моніторингу та оповіщення.

Підсумовуючи, збираючи детальну інформацію про виклики та відповіді API через систему журналювання, можна швидко виявити проблеми та вжити заходи для їх усунення. Система журналювання також є цінним джерелом інформації для налаштування та оптимізації продуктивності.

2.1.7 Тестовий режим та перевірки функціональності інтеграції

Модуль повинен мати режим Sandbox, тобто режим, який дозволяє робити тестові покупки. При здійсненні таких операцій слід використовувати спеціальні тестові рахунки. Це надзвичайно важливо для такого типу застосування.

Для впровадження режиму Sandbox, перш за все необхідно зареєструвати обліковий запис розробника PayPal та створити обліковий запис Sandbox середовища. Створивши обліковий запис Sandbox середовища, можна отримати облікові дані API, необхідні для інтеграції з Sandbox середовищем. Ці облікові дані містять ідентифікатор клієнта та секретний ключ. Після цього потрібно оновити параметри конфігурації модуля, щоб використовувати облікові дані API Sandbox програмного середовища замість поточних облікових даних. Залежно від структури модуля, це може включати оновлення коду або конфігураційних файлів.

Після налаштування тестового середовища, важливо протестувати роботу модуля та інтеграцію. Це робиться шляхом створення тестових транзакцій.

В цілому, інтеграцію повинно бути ретельно протестовано, щоб забезпечити, що вона працює коректно та забезпечує безпеку. PayPal надає інструменти для тестування платежів та перевірки функціональності інтеграції.

2.2 Система керування вмістом PrestaShop

Система керування вмістом PrestaShop була запущена у 2007 році студентами Epitech і швидко досягла великого успіху. Сьогодні понад 250 000 сайтів електронної комерції використовують це PrestaShop.

PrestaShop є дуже повним і в той же час простим у використанні та безкоштовним програмним забезпеченням з ліцензією Open Software License (OSL). Це означає, що можна використовувати, змінювати і розповсюджувати PrestaShop безкоштовно.

Кілька сотень функцій електронної комерції пропонуються нативно або у формі розширень (безкоштовних або платних). Цілком можливо використовувати PrestaShop, керувати замовленнями, запасами та продуктами, не маючи технічних знань програмування [32]. Рішення також пропонує можливість експорту продукції клієнта на основні ринки.

PrestaShop надає чимало функції, багато в чому завдяки своєму статусу програмного забезпечення з відкритим кодом. Приклади таких інструментів, необхідних для створення інтернет-магазину:

– настроювані теми для створення онлайн-магазину. Є понад 2000 високоякісних графічних тем на вибір, щоб налаштувати магазин. Усі теми оптимізовано для мобільних пристроїв і їх можна налаштовувати, змінюючи кольори, зображення, макет і типографіку;

– повний і 100% безпечний хостинг за допомогою Google Cloud Platform. Завдяки угоді з Google усі магазини електронної комерції будуть розміщені на Google Cloud Platform. GCP надає можливість масштабувати хостинг в залежності від потреб бізнесу, має широкий вибір високопродуктивних сервісів, таких як Compute Engine та Kubernetes Engine, а також має розгалужену мережу дата-центрів по всьому світу, що дозволяє розміщувати вебсайти ближче до певної аудиторії. Таким чином, ця послуга пропонує все, що потрібно торговцям, щоб зробити свій сайт електронної комерції успішним і розвиватися. Запустивши PrestaShop на GCP, можна також використовувати інші послуги для вдосконалення магазину електронної

комерції, наприклад Google Cloud Storage для зберігання зображень і відео продуктів і Google Cloud AI Platform для впровадження моделей машинного навчання для покращення продуктивності інтернет-магазину;

– управління каталогом продуктів. Можливість швидко й легко керувати повним каталогом із тисячами продуктів і сотнями категорій;

– різні засоби оплати. Користувачам доступні різні засоби оплати, щоб потенційні покупці могли завершити свої покупки простим і безпечним способом. Є можливість запропонувати покупцям оплату карткою чи електронним гаманцем, передоплатою, накладеним платежем або готівкою в магазині;

– просте керування замовленнями. Керування замовленнями в режимі реального часу, доступ до даних про доставку, автоматичне змінення рахунку-фактури та спілкування з провідними компаніями доставки на ринку для доставки продуктів клієнтам;

– просте керування доставкою та податками. PrestaShop надає інструменти для керування варіантами доставки та розрахунку податків, гарантуючи, що інтернет-магазин відповідає місцевим нормам. У PrestaShop податки розраховуються на основі кількох факторів, зокрема правил оподаткування продукту, місцезнаходження клієнта та податкової конфігурації магазину;

– підтримка кількох мов і кількох валют. PrestaShop підтримує кілька мов і валют, що полегшує створення глобального онлайн-магазину, який обслуговує клієнтів з усього світу;

– зручність для пошукових систем. PrestaShop розроблено з урахуванням оптимізації пошукових систем (SEO) із такими функціями, як настроювані мета-теги, структура URL-адрес і карта сайту. Ці функції допомагають покращити видимість інтернет-магазину в пошукових системах;

– інструменти електронного маркетингу для збільшення продажів в Інтернеті. PrestaShop також пропонує маркетингові інструменти для покращення трафіку, збільшення продажів і збільшення обороту. Є модулі, які дозволять клієнтам створювати промо-коди, пропозиції продуктів, блог,

електронну пошту, настроювані витрати на доставку та все необхідне для успіху електронної комерції.

Важливою перевагою PrestaShop, на мою думку, є розділення коду HTML і PHP за допомогою архітектури Model-View-Controller (MVC) [33]. Таке розділення допомагає зберігати код чистим, легким для розуміння та підтримки, а також забезпечує спрощення обслуговування.

В архітектурі MVC «Model» представляє дані та бізнес–логіку програми, «View» представляє інтерфейс користувача, а «Controller» діє як посередник між «Model» та «View», обробляючи введені користувачем дані та відповідно оновлюючи «Model» та «View».

PrestaShop використовує Smarty, механізм створення шаблонів на основі PHP, для обробки рівня презентації (View) програми. Код PHP (контролер і модель) взаємодіє з базою даних і виконує необхідні обчислення, а шаблони Smarty обробляють відображення цих даних користувачеві.

Починаючі з вірсії 1.7, PrestaShop надає опціональну можливість використовувати механізм шаблонів Twig, який замінює механізм шаблонів Smarty. Twig – це популярна система створення шаблонів для PHP, яка пропонує ряд переваг перед Smarty, включаючи покращену продуктивність, кращу безпеку та більшу гнучкість.

Як і Smarty, Twig відокремлює рівень презентації (View) від бізнес-логіки (Controller і Model). Шаблони Twig написані за синтаксисом, подібним до HTML, але з додатковою функціональністю, яка дозволяє генерувати динамічний вміст і маніпулювати ним.

У PrestaShop код PHP (контролер і модель) взаємодіє з базою даних і виконує необхідні обчислення, а шаблони Twig обробляють відображення цих даних для користувача. Поділ завдань, який забезпечує архітектура Model-View-Controller (MVC), у поєднанні з гнучкістю та безпекою Twig полегшує підтримку та оновлення програми з часом.

Такий поділ завдань допомагає гарантувати, що програма підтримується та масштабується, а також полегшує оновлення чи зміну окремих частин програми, не впливаючи на всю систему.

Проте, важливо також розглянути недоліки системи керування вмістом PrestaShop:

– велика частина тем та модулів – це платні інструменти;

– функціональність «блогу» є дуже обмеженою;

– нативні функції SEO недостатньо розроблені (але це може бути вирішено за допомогою модулів);

– платна персоналізована технічна допомога;

– низька швидкість. У порівнянні з деякими іншими CMS, PrestaShop може працювати повільніше через складні алгоритми обробки платежів та збір даних.

CMS PrestaShop дозволяє вдосконалити базові функціональні можливості, використовуючи різноманітні розширення. Ці розширення можуть впливати на різні аспекти продукту, такі як керування запасами, оплата, керування користувачами тощо. Наприклад, можна налаштувати розширення, здатне синхронізувати запаси магазину з ERP, запровадити нові засоби оплати, які не підтримуються спочатку.

Отже, платформа PrestaShop дуже модульна, з великою спільнотою розробників, які створюють і діляться додатками, які можна легко інтегрувати в платформу для розширення її функціональності.

В третьому розділі кваліфікаційної роботи ми більш детально розглянемо основні технічні особливості CMS PrestaShop, які було використано під час розробки модуля для інтеграції платіжної системи PayPal.

Загалом PrestaShop – це потужна та гнучка платформа електронної комерції, яка пропонує низку функцій та інструментів, які допомагають компаніям створювати та керувати своїми онлайн-магазинами. Його природа з відкритим вихідним кодом і велика спільнота розробників роблять його популярним вибором для компаній будь-якого розміру, які прагнуть створити онлайн-присутність.

2.3 Методи безпеки модуля для інтеграції платіжної системи PayPal з системою керування вмістом

Важливо зазначити, що інтеграція платіжної системи PayPal з системою керування вмістом або універсальною платформою електронної комерції вимагає чималої уваги до забезпечення безпеки модуля. Розглянемо декілька загальних міркувань безпеки, про які слід пам’ятати під час розробки модуля для інтеграції PayPal:

– необхідно перевіряти дані, введені користувачами, з метою запобігати будь-якому зловмисному введенню, такому як впровадження SQL-ін’єкцій або міжсайтовий скриптинг (XSS). Існує кілька способів перевірки даних, введених користувачами:

1) валідація даних. Тобто перевірка, чи відповідають введені користувачем дані очікуваному формату і типу. Наприклад, перевірка, чи введене значення є числом, електронною адресою або URL-адресою;

2) екранування даних. Захист даних від SQL-ін’єкцій шляхом екранування спеціальних символів, таких як одинарні лапки, або використання параметризованих запитів;

3) обмеження довжини даних. Введені дані не повинні перевищувати обмеження довжини, встановленого для конкретного поля;

4) фільтрація даних. Корисним є використання фільтрів, таких як фільтр для HTML-коду, для запобігання XSS-атак;

5) використання токенів безпеки для захисту від CSRF-атак;

6) обмеження доступу до функцій або даних в залежності від ролі користувача;

– для забезпечення безпечного зв’язку між вебсайтом і API PayPal, зв’язок повинен бути зашифрований за допомогою SSL та TLS (криптографічні протоколи). Рукостискання SSL використовує комбінацію шифрування відкритим і закритим ключами для встановлення безпечного з’єднання між клієнтом і сервером. Процес зазвичай працює таким чином: клієнт надсилає запит на сервер для встановлення безпечного з’єднання,

сервер надсилає свій сертифікат SSL/TLS, який містить його відкритий ключ. Далі клієнт перевіряє сертифікат і генерує ключ сеансу, який використовується для шифрування даних протягом сеансу, потім клієнт надсилає сеансовий ключ на сервер, зашифрований відкритим ключем сервера. Сервер розшифровує сеансовий ключ своїм закритим ключем і використовує його для шифрування даних, що передаються клієнту. Далі клієнт розшифровує дані за допомогою сеансового ключа. Процес рукостискання SSL зображено на рисунку 2.1 [34];

Рисунок 2.1 – Рукостискання SSL

– необхідно реалізувати перевірку всіх транзакцій. Модуль повинен перевіряти, перш ніж обробляти транзакцію, що транзакція є законною та не шахрайською. Зі свого боку, PayPal надає певні інструменти для виявлення шахрайських транзакцій. Наприклад, продавець може відстежувати транзакції з особливо великою сумою покупки, транзакції, що відбуваються з незвичайних місць або IP-адрес, або транзакції, здійснені покупцями, які мають історію шахрайства. Після виявлення потенційно шахрайської транзакції сервер продавця може вжити відповідні заходи, такі як скасування

замовлення, утримання замовлення та позначення покупця, щоб переконатися, що під час роботи з ними в майбутньому вжито додаткових заходів обережності.

Крім того, PayPal використовує алгоритми машинного навчання для аналізу даних і виявлення шаблонів, які можуть свідчити про шахрайську діяльність. Наприклад, система виявлення аномалій, яка використовує алгоритми для виявлення транзакцій, які сильно відрізняються від інших транзакцій, що здійснюються користувачами на регулярній основі. Тобто, якщо користувач здійснює покупки на суму, яка значно перевищує його звичайні звички, система може автоматично відмінити транзакцію як підозрілу.

Також використовується система виявлення спільних ознак між транзакціями, які були пов’язані зі шахрайською діяльністю. Наприклад, якщо виявлено, що певний IP-адрес або електронна адреса були пов’язані з кількома підозрілими транзакціями, система може автоматично відмітити інші транзакції, які пов’язані з цими підозрілими даними;

– важливо зберігати конфіденційні дані користувачів у безпеці. До конфіденційних даних відносять, наприклад, ключі API та токени доступу. Таким чином, необхідно реалізувати шифрування даних та подальше їх збереження у безпечному місці, такому як бази даних або файли конфігурації;

– модуль слід регулярно оновлювати, на це є декілька причин. Перш за все, важливо переконатися, що будь-які вразливості безпеки завчасно виправлено. По-друге, модуль повинен залишався сумісним з останньою версією CMS.

Дотримуючись зазначених вище заходів безпеки, можна переконатися, що розроблений модуль PayPal є безпечним для користувачів.

2.4 Особливості розробки модуля для інтеграції платіжної системи PayPal з системою керування вмістом

2.4.1 Механізм взаємодії між клієнтом і сервером

Модуль повинен вміти передавати дані між системою керування вмістом і платіжною системою PayPal. Це може включати передачу даних про товари, ціни, кількості, дані клієнтів і т. д. Важливо забезпечити правильну інтеграцію і зберігання цих даних.

Під час розробки модуля варто розуміти механізм взаємодії між клієнтом і сервером PayPal. Модуль для інтеграції платіжної системи PayPal з CMS діє як посередник між користувачем і сервером PayPal. Зазвичай такий процес працює наступним образом:

– клієнт ініціює платіж на вебсайті, натиснувши кнопку або посилання, пов’язане з платіжним шлюзом PayPal;

– модуль отримує запит від клієнта і тому надсилає платіжний запит на сервер PayPal, який перевіряє платіжні деталі та повертає відповідь модулю;

– модуль отримує відповідь на платіж від PayPal і обробляє його відповідно до статусу платежу;

– статус платежу оновлюється в CMS, а клієнт у свою чергу також отримує сповіщення про статус платежу.

Для обміну запитами (для надсилання платіжних запитів і отримання відповідей від сервера PayPal) зазвичай використовується PayPal REST API.

REST API – це вебінтерфейс, який дозволяє розробникам взаємодіяти з серверами PayPal через HTTP(S) за допомогою стандартних методів HTTP, таких як GET, POST, PUT і DELETE. API використовує JSON (JavaScript Object Notation) як формат даних, який є легким і широко використовуваним форматом обміну даними.

Щоб надіслати платіжний запит на сервер PayPal, модуль має надіслати запит HTTP POST до кінцевої точки REST API PayPal, яка відповідає платіжному ресурсу. Запит POST міститиме платіжні деталі в тілі запиту, закодовані у форматі JSON.

Приклад типового платіжного запиту у форматі JSON, який було використано під час кваліфікаційної роботи, наведено нижче (рис. 2.2).

```
{
    "intent": "sale",
    "payer": {
        "payment_method": "paypal"
    },
    "transactions": [
        {
            "amount": {
                "total": "10.00",
                "currency": "USD"
            },
            "description": "Payment for order #123"
        }
    ],
    "redirect_urls": {
        "return_url": "https://example.com/return",
        "cancel_url": "https://example.com/cancel"
    }
}
```

Рисунок 2.2 – Приклад типового платіжного запиту у форматі JSON

Наведений вище запит вказує основний намір платежу, спосіб платежу (у цьому випадку PayPal), суму й опис транзакції, а також URL-адреси повернення та скасування платежу. Коли модуль надсилає цей запит на сервер PayPal, сервер перевіряє деталі платежу та надсилає відповідь модулю зі статусом платежу.

Підсумовуючи можна сказати, що модуль, який інтегрує PayPal з CMS, зазвичай використовує PayPal REST API для надсилання платіжних запитів і отримання відповідей від сервера PayPal, використовуючи запити HTTP(S) і корисні дані JSON для передачі платіжних даних між модулем і сервером.

Модуль також обробляє інші функції, пов’язані з оплатою, наприклад повернення коштів, оскарження та скасування. Коли клієнт запитує

відшкодування або подає спір, модуль зв'язується з сервером PayPal для обробки запиту та оновлення статусу платежу в CMS.

Для забезпечення безпеки платіжних операцій модуль використовує шифрування SSL/TLS для захисту конфіденційних платіжних даних під час передачі. Модуль також може зберігати платіжні дані в безпечній базі даних або зашифрованій файловій системі для запобігання несанкціонованому доступу.

Таким чином, модуль, який інтегрує PayPal з системою керування вмістом, діє як міст між клієнтом і сервером PayPal, обробляючи платіжні запити та відповіді. Основна функція модуля полягає в обробці запитів від клієнтів і передачі їх на сервер PayPal, а також управлінні результатами цих запитів та відповідями від платіжної системи PayPal шляхом використання PayPal REST API. Отже, роль модуля полягає у взаємодії з обома сторонами та передачі інформації між ними. Це допомагає суттєво спростити процес оплати для клієнтів, одночасно забезпечуючи безпеку та цілісність платіжних операцій.

2.4.2 Етапи реалізації модуля для інтеграції PayPal з CMS PrestaShop

Написання модуля для інтеграції PayPal з PrestaShop передбачає певні кроки, які необхідно розуміти заздалегідь. На цьому етапі ми коротко розглянемо основний підхід до розробки модуля інтеграції платіжної системи PayPal з CMS PrestaShop [35]. В третьому розділі кваліфікаційної роботи буде описано безпосередньо процес реалізації модуля та наведено особливості розробки, а також характерні відмінності від кроків, наведених нижче.

В першу чергу, потрібно створити новий каталог для модуля в каталозі «modules» інсталяції PrestaShop. Цей каталог має таку саму назву, як і модуль.

Далі необхідно створити файл конфігурації модуля в кореневому каталозі модуля. Цей файл буде визначати назву, версію, автора та інші метадані розробленого модуля.

Наступним кроком буде створення файлу класу для модуля в каталозі «classes» цього модуля. Цей файл класу у подальшому буде визначати функціональність модуля, наприклад, функції для обробки платежів і обробки зворотних викликів від PayPal.

Далі буде створено інсталяційний файл для модуля в кореневому каталозі цього модуля [36]. Цей файл буду визначати усі SQL-запити, необхідні для створення відповідних таблиць бази даних і параметрів для модуля.

Наступний крок – це створення файлів відображення модуля в каталозі «views/templates» модуля. Цей файл буде безпосередньо визначати HTML і CSS дані, необхідні для відображення кнопки оплати PayPal та інших елементів інтеграції PayPal.

Звісно, на завершення необхідно перевірити та, за необхідністю, налагодити розроблений модуль. Після реалізації основних кроків створення модуля слід ретельно перевірити правильність роботи модуля, включаючи обробку платежів, політику повернення коштів, а також відповідне оновлення статусів, щоб переконатися, що платежі обробляються належним чином і всі зворотні виклики від PayPal також обробляються правильно. На цьому етапі будуть використані інструменти налагодження для усунення будь-яких проблем, які виникають під час тестування.

Підсумовуючи, написання модуля для інтеграції PayPal з системою керування вмістом PrestaShop потребує глибокого розуміння PHP, MySQL і фреймворку PrestaShop. Однак завдяки ретельному плануванню та уважності до деталей можливо створити міцну та надійну інтеграцію з електронною платіжною системою PayPal, яка допоможе збільшити продажі та покращити процес оплати.

Написання такого модуля для інтеграції у рамках кваліфікаційної роботи допоможе вдосконалити навички вебінтеграції, навички в галузі обчислювальної техніки загалом, а також знання з різних мов програмування, таких як HTML, CSS, PHP, JavaScript.

3 РЕАЛІЗАЦІЯ ІНТЕГРАЦІЇ ПЛАТІЖНОЇ СИСТЕМИ PAYPAL З СИСТЕМОЮ КЕРУВАННЯ ВМІСТОМ PRESTASHOP

3.1 Структура розробленого модуля для інтеграції платіжної системи PayPal з системою керування вмістом PrestaShop

Модулі PrestaShop складаються з великої кількості файлів. Усі ці файли зберігаються в папці з такою самою назвою, як і модуль (назва модуля повинна містити лише малі літери та цифри), яка, у свою чергу, зберігається в папці «modules» у корені основної папки PrestaShop: /modules/<modulename> /.

Під час розробки структури модуля для CMS PrestaShop було використано наступні елементи:

– основний файл модуля. Це головний файл модуля, і він повинен мати назву за модулем. Це файл PHP, який містить оголошення модуля, конфігурацію та код встановлення;

– папка «config», де зберігаються важливі файли конфігурації, які визначають різноманітні налаштування та параметри модуля;

– контролери. Це файли PHP, які обробляють запити користувачів та повертають відповіді. У PrestaShop контролери звертаються до класів, щоб отримувати доступ до різних функцій та даних, необхідних для обробки запитів та взаємодії з базою даних та іншими компонентами системи. Контролери розміщуються в певній папці модуля в залежності від того, якою частиною офісу вони керують:

1) /controllers/admin: контролери бек-офісу модуля;
2) /controllers/front: контролери переднього офісу модуля;

– класи PHP, які використовує модуль. Вони містять бізнес-логіку модуля та відповідають за перевірку даних, обробку помилок та інші операції, пов’язані з функціональністю модуля;

– папка «translations», яка містить файли перекладів, які дозволяють відображати формулювання модуля різними мовами;

– папка «upgrade», яка містить сценарії оновлення, які будуть виконуватися під час оновлення модуля з попередньої версії;

– папка «vendor», яка містить бібліотеки, імпортовані через Composer, а також його автозавантажувач;

– views. Це шаблони, які містить файли шаблонів модуля (.tpl для Smarty або .html.twig для Twig), а також статичні ресурси, які використовуються модулем (css, js або файли зображень). Кожен тип має бути розташований у власних папках: /views/{js, css, img, fonts}. Залежно від потреб, файли шаблонів розташовані в різних вкладених папках:

1) /views/templates/admin: файли шаблонів Smarty або Twig, які використовуються контролерами бек-офісу модуля;

2) /views/templates/front: файли шаблонів Smarty, які використовуються контролерами переднього офісу модуля;

3) /views/templates/hook: файли шаблонів Smarty, які використовуються хуками модуля;

– файл config.xml, який містить кешовану копію властивостей основного класу модуля для оптимізації продуктивності списків модулів. Користь цього файла у тому, що цей файл дозволяє виконувати сценарії оновлення (в upgrade/) відразу після завантаження zip. Крім того, цей файл автоматично генерується PrestaShop під час встановлення модуля, якщо він ще не існує;

– файл піктограми logo.png розміром 32×32 пікселя, який відображатиметься у списках модулів.

Структура модуля PrestaShop може змінюватися залежно від конкретних вимог до модуля, але наведені вище компоненти зазвичай включені в більшість модулів.

Під час розробки структури модуля було використано загальну схему, надану в документація PrestaShop та зображену на рисунку 3.1.

```
mymodule
├── config
│   ├── admin
│   │    └── services.yml
│   ├── front
│   │   └── services.yml
│   └── services.yml
├── controllers
├── override
├── src
│   ├── Controller
│   └── Entity
├── translations
├── upgrade
├── vendor
├── views
│   ├── css
│   ├── img
│   ├── js
│   └── templates
├── config.xml
├── logo.png
└── mymodule.php
```

Рисунок 3.1 – Загальна структура модуля PrestaShop

3.2 Програмна реалізація

3.2.1 Встановлення модуля

Встановлення модуля – це процес побудови структури модуля. В першу чергу було створено новий каталог для модуля в каталозі «modules» інсталяції PrestaShop. Цей каталог має таку саму назву, як і модуль: paypal. Далі створено головний файл модуля в кореневому каталозі модуля, який також має назву за модулем: paypal. Це файл PHP, який визначає назву модуля, версію, автора та інші метадані розробленого модуля.

Наступним кроком є створення класу для модуля в основному файлі модуля paypal.php. Цей клас має таку саму назву, як і модуль: paypal. Важливо зазначити, що цей клас успадковує стандартний класс PaymentModule, який використовує PrestaShop, який у свою чергу успадковує стандартний класс Module. Створений файл класу у подальшому буде визначати функціональність модуля, наприклад, функції для обробки платежів і обробки зворотних викликів від PayPal.

В цьому файлу класу використовується метод construct, який задає початковий стан модуля. Цей метод використовується для встановлення основних властивостей модуля, таких як його назва, версія, автор, опис модуля. Він також викликає конструктор батьківського класу parent::__construct().

Завдяки наявності файлу класу та методу construct, розроблений модуль може бути легко встановлений через CMS PrestaShop [37]. При завантаженні модуля в PrestaShop відбувається наступний процес:

– завантаження модуля в архіві з розширенням .zip на сервер PrestaShop. Вебсервер розархівовує файл та зберігає модуль в директорії /modules/;

– далі, перед встановленням модуля PrestaShop перевіряє права на запис для директорії /modules/. Якщо цих прав немає, система попереджає про це користувача;

– реєстрація модуля. Під час завантаження модуля PrestaShop перевіряє файл /modules/modulename/modulename.php. Якщо файл знайдено, система реєструє модуль у базі даних, додаючи інформацію про його назву, версію, автора та інші параметри;

– створення таблиць в базі даних, необхідних для роботи модуля;

– після встановлення модуля PrestaShop оновлює кеш системи, щоб новий модуль був доступним для використання.

Після встановлення модуля в PrestaShop користувач може налаштовувати модуль через панель адміністрування PrestaShop. Можливості налаштування параметрів розробленого модуля будуть описані далі. Також модуль може бути вимкнений або видалений будь-коли через панель адміністрування PrestaShop (рис. 3.2).

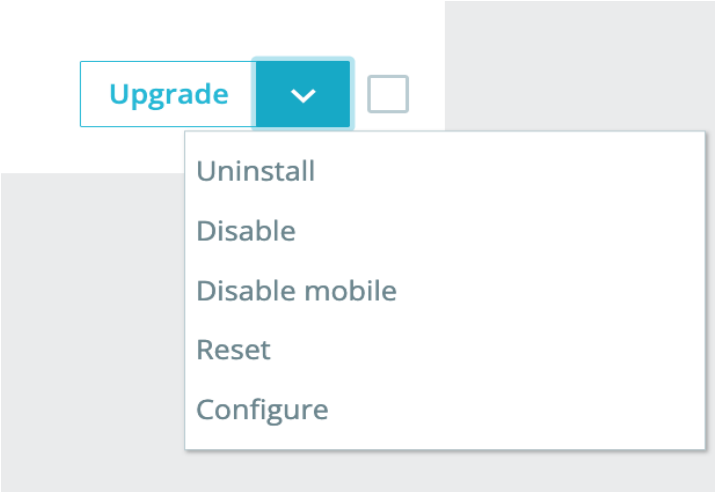

Рисунок 3.2 – Доступ до налаштування модуля

3.2.2 Керування параметрами конфігурації модуля

Для коректної роботи модуля з можливістю налаштування та підключення PayPal аккаунта користувача, необхідно створити спеціальну сторінку, доступну через адміністративну панель PrestaShop. Саме з цією метою в стандартній версії CMS PrestaShop існує AdminController – базовий клас PHP для створення адміністративних контролерів (controllers) модулів. Цей клас дозволяє створювати сторінки та обробляти дії в адміністративній панелі PrestaShop.

Крім того, в CMS PrestaShop також існує клас ModuleAdminController, який успадковує клас AdminController. Це успадкування зроблено для ефективного використання функціональних можливостей і можливостей, наданих класом AdminController, який є спеціалізованим класом, спеціально розробленим для керування адміністративними завданнями, пов’язаними з модулями.

Під час розробки модуля для інтеграції платіжної системи PayPal було створено новий клас AdminPaypalSetupController, який у свою чергу спадкує стандартний клас ModuleAdminController і доповнює його. Отже, клас AdminPaypalSetupController отримує доступ до набору попередньо визначених методів і властивостей класу ModuleAdminController, які використовуються в

управлінні модулями, та розширює функціональні можливості класу. Це включає керування надсиланням форм, відображення списків і деталей даних модуля, обробку розбиття на сторінки, керування дозволами та контролем доступу тощо. Коли запит робиться до ModuleAdminController, PrestaShop може використовувати інформацію в URL-адресі, щоб визначити, який метод у контролері має обробити запит.

Таким чином, розроблений модуль надає користувачам доступ до сторінки налаштування із серверної частини. Ця сторінка доступна через адміністративну панель PrestaShop (рис. 3.3).

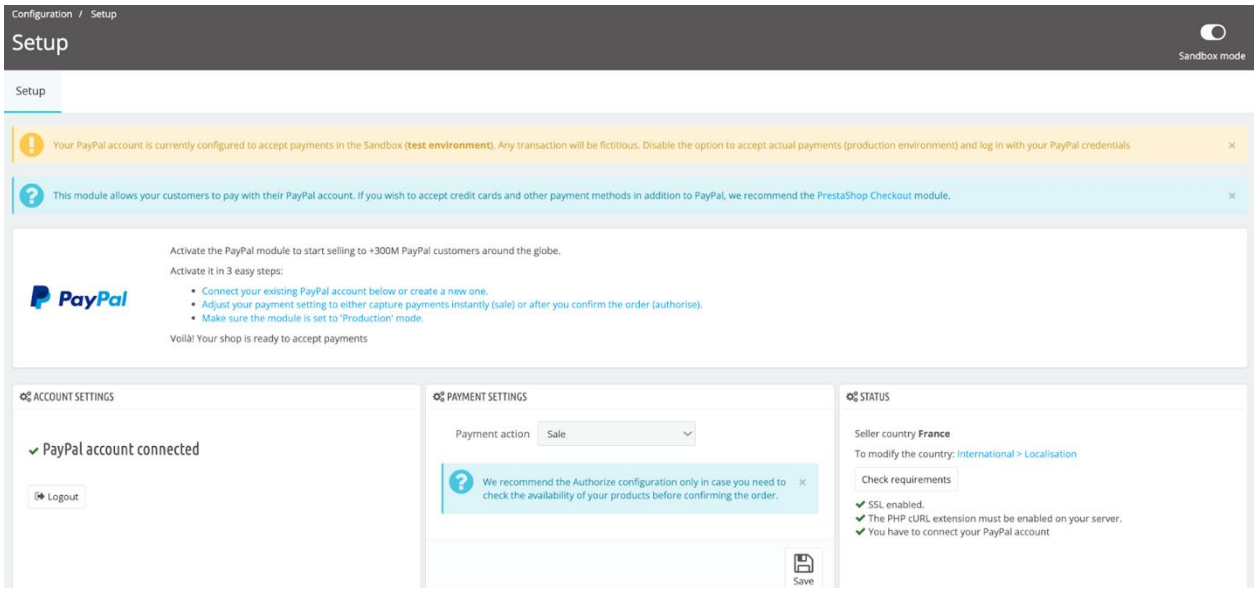

Рисунок 3.3 – Сторінка налаштування модуля із серверної частини

ModuleAdminController містить методи для відображення та обробки форм, збереження налаштувань та обробки інших дій, пов’язаних із конфігурацією модуля. Таким чином, ці методи взаємодіють з базою даних PrestaShop.

Загалом, AdminPaypalSetupController відіграє важливу роль у розробці модуля для PrestaShop, оскільки він надає спосіб керувати параметрами конфігурації модуля зручним для користувача способом із адміністративної панелі магазину.

Через AdminPaypalSetupController реалізовано процес налаштування зв’язку з обліковим записом PayPal продавця (рис. 3.4). Процес налаштування буде описано в підпункті 3.2.3.

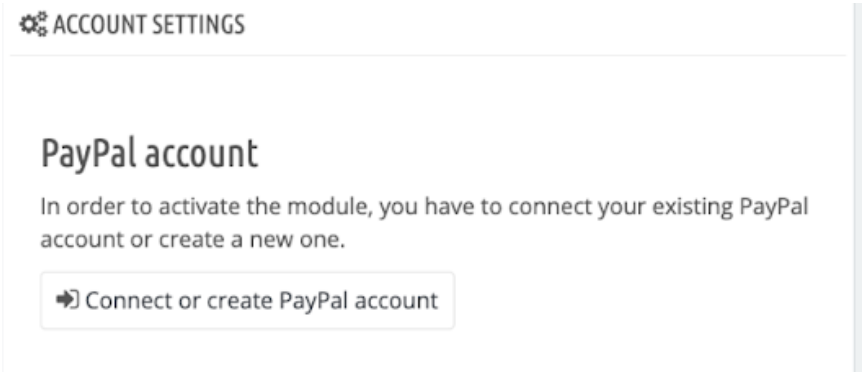

Рисунок 3.4 – Налаштування зв’язку з обліковим записом PayPal

Через AdminPaypalSetupController також реалізовано процес налаштування режиму роботи модуля (рис. 3.5).

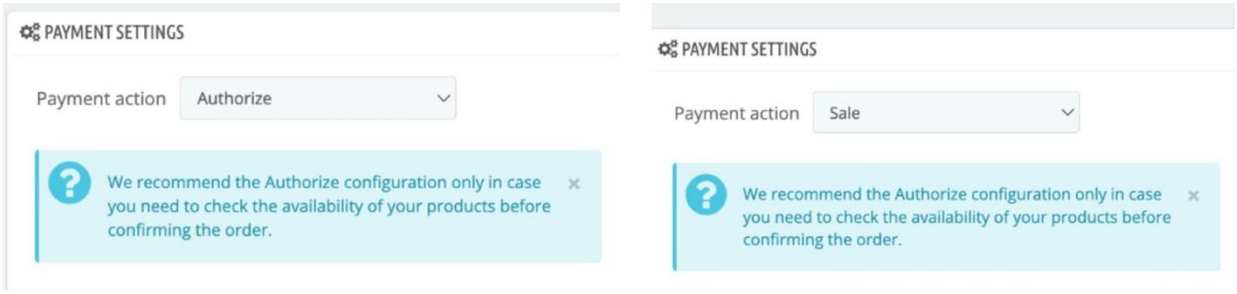

Рисунок 3.5 – Налаштування режиму роботи модуля

Розроблений модуль має два режими платіжних дій, доступні користувачу:

– authorize. Коли користувач авторизує платіж, модуль надсилає запит на авторизацію платежу на платіжний шлюз PayPal. Далі PayPal отримує запит та виконує серію перевірок, включаючи перевірку платіжних даних клієнта, наявність коштів і перевірки на потенційне шахрайство. Якщо перевірку пройдено, PayPal резервує необхідну суму для майбутньої транзакції. Дозволену суму затримують на певний період, зазвичай кілька днів. Протягом цього часу кошти не відразу переказуються з рахунку клієнта

на рахунок продавця. Авторизація використовується, щоб переконатися, що клієнт має достатньо кошти, і зарезервувати суму для захоплення пізніше. Крім того, таким чином продавець може перерахувати авторизовані кошти саме тоді, коли буде готовий виконати замовлення або надати продукти чи послуги клієнту.

При налаштуванні модуля на режим Authorize, ініціація транзакції не відбувається відразу після розміщення замовлення, але PayPal відразу резервує суму замовлення. В CMS PrestaShop такі замовлення створені зі статусом Awaiting for PayPal payment (рис. 3.6). Такий статус створюється в CMS PrestaShop під час установки розробленого модуля з використанням стандартного класу OrderState. Для авторизації транзакції необхідно або вручну оновити статус замовлення на Payment Accepted в панелі адміністратора CMS PrestaShop, або авторизувати транзакцію через PayPal аккаунт продавця. Лише після авторизації транзакції продавець отримує кошти на свій рахунок PayPal з рахунку клієнта;

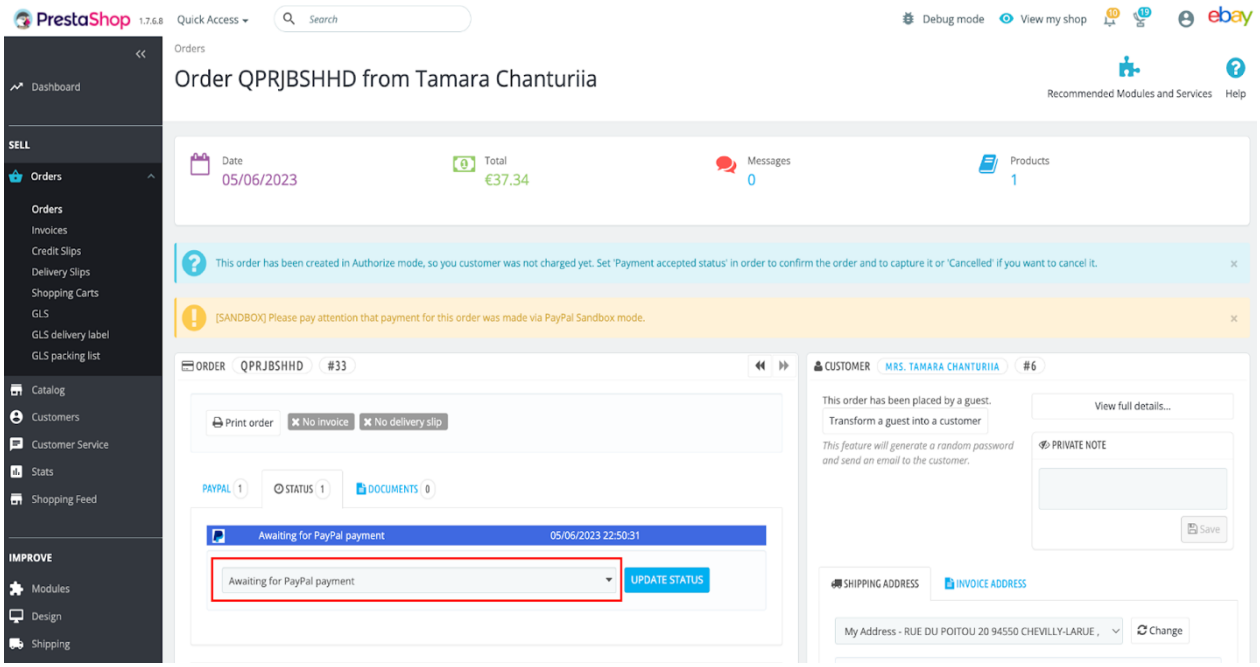

Рисунок 3.6 – замовлення зі статусом Awaiting for PayPal payment

– sale. Дія продажу виконується для фактичного переказу авторизованих коштів з рахунку клієнта на рахунок продавця. Цей режим

налаштування є найпоширенішим та означає, що продавець готовий відразу виконувати замовлення. Він відразу ініціює переказ коштів з рахунку клієнта на рахунок продавця.

При налаштуванні модуля на режим Sale, ініціація транзакції відбувається відразу після розміщення замовлення. В CMS PrestaShop такі замовлення створені зі статусом Payment Accepted (рис. 3.7).

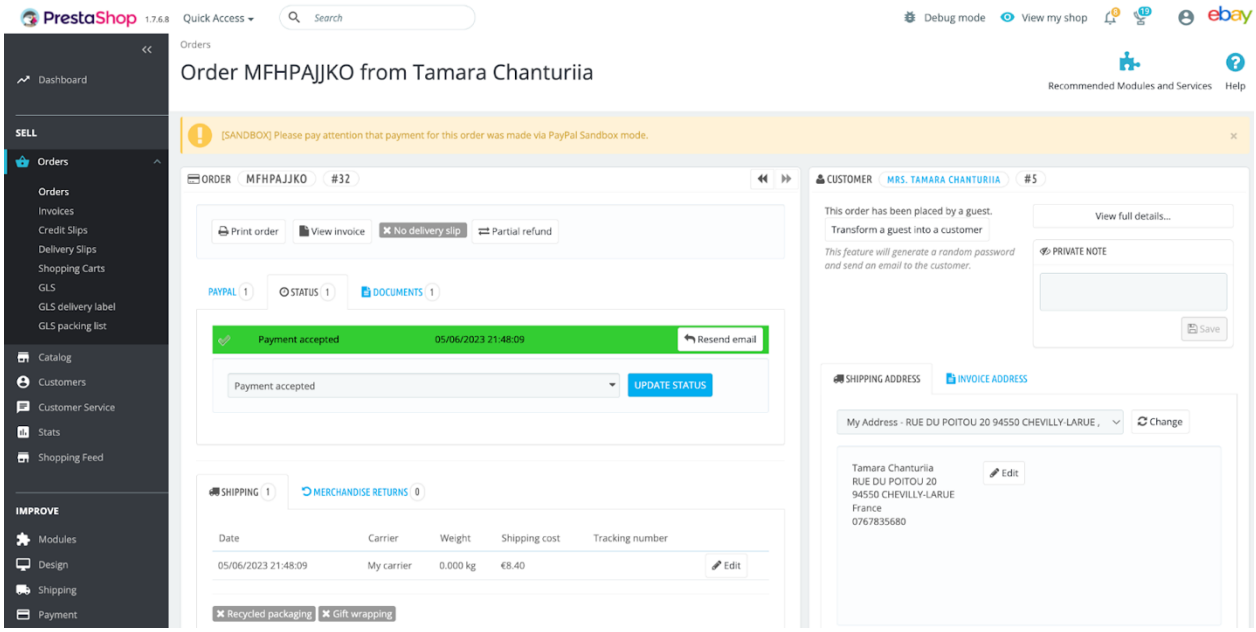

Рисунок 3.7 – замовлення зі статусом Payment Accepted

Усі параметри конфігурації зберігаються в базі даних. Для збереження та управління конфігураціями реалізованого модуля було використано стандартний клас PrestaShop Configuration. Цей клас відповідає за керування налаштуваннями конфігурації модуля. Він надає набір методів для отримання, встановлення та видалення значень конфігурації, які можна використовувати для керування різними аспектами поведінки модуля.

Клас Configuration реалізований як статичний клас, що означає, що його методи можна викликати без створення екземпляра об’єкта класу. Розроблений модуль використовує наступні методи, надані класом Configuration:

– get(). Отримує значення налаштування конфігурації. Метод приймає назву налаштування як параметр і повертає його значення у вигляді рядка;

– updateValue(). Оновлює значення налаштування конфігурації. Метод приймає назву налаштування та його нове значення як параметри та повертає логічне значення, яке вказує, чи було оновлення успішним;

– deleteByName(). Видаляє налаштування конфігурації. Метод приймає назву параметра як параметр і повертає логічне значення, яке вказує, чи було видалення успішним.

Клас Configuration широко використовується в PrestaShop для керування різними налаштуваннями. Можна також створювати власні параметри конфігурації за допомогою методу Configuration::updateValue() і отримувати їх пізніше за допомогою методу Configuration::get().

Варто зазначити, що параметри конфігурації зберігаються в базі даних, а зміни, зроблені за допомогою класу Configuration, негайно відображаються в базі даних. Важливо бути обережними, змінюючи налаштування конфігурації, оскільки неправильні значення можуть мати небажані наслідки для поведінки магазину.

3.2.3 Налаштування звї’язку з обліковим записом PayPal користувача

Для консолідації облікових даних користувача та спрощення процесу входу для користувачів, під час розробки модуля було використано протокол авторизації OAuth 2.0 [38]. OAuth 2.0. протокол використовується модулем для отримання облікових даних користувача (CLIENT_ID та SECRET_ID) під час підключення PayPal аккаунту в панелі Адміністратора. Цей крок є необхідним для подальшої роботи модуля.

OAuth – це відкритий стандартний протокол, який надає безпечний і стандартизований спосіб доступу до даних користувача від імені користувача, не передаючи фактичні облікові дані (наприклад, ім’я користувача та пароль). Це дозволяє користувачам надавати обмежений доступ до своїх захищених ресурсів іншій програмі чи службі, не передаючи свої облікові дані. Процес надання доступу зображено на рисунку 3.8.

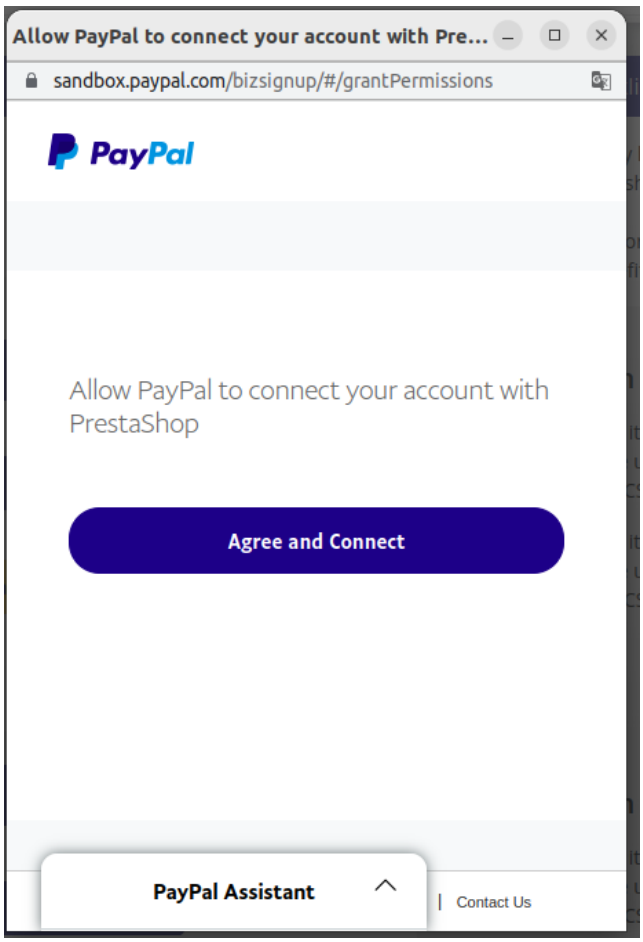

Рисунок 3.8 – Доступ до даних користувача

Таким чином вводиться рівень авторизації та контролюється обмін даними між розробленим модулем та платіжною системою PayPal, зберігаючи при цьому приватні дані користувачів в безпеці. Розроблений модуль використовує протокол авторизації OAuth 2.0, який використовує токени доступу.

У OAuth клієнт запитує доступ до ресурсів, які контролюються власником ресурсу та розміщені на сервері ресурсів, і отримує інший набір облікових даних, ніж власник ресурсу. Замість того, щоб використовувати облікові дані власника ресурсу для доступу до захищених ресурсів, клієнт отримує токен доступу – хеш-строку, що вказує певний обсяг, термін служби та інші атрибути доступу. Токени доступу видаються стороннім клієнтам сервером авторизації зі схвалення власника ресурсу. Потім клієнт використовує токен доступу для доступу до захищених ресурсів, розміщених на сервері ресурсів.

Оскільки токени доступу недовговічні та закінчуються через певний період часу (це додає додатковий рівень безпеки), модуль використовує токени доступу для отримання довгострокових токенів – CLIENT_ID та SECRET_ID.

У модулі реалізовано процес налаштування з використанням протоколу OAuth наступним чином:

– першим кроком є підключення бібліотеки PayPal;

– далі модуль перенаправляє клієнта на платіжну систему PayPal для підтвердження доступу до ресурсів;

– після підтвердження доступу до ресурсів, PayPal генерує два недовговічні токени доступу. З метою надання додаткового рівня безпеки, ці токені діють лише декілька секунд;

– модуль використовує ці два токени доступу для генерації іншого токену, більш довгострокового. Генерація іншого токену реалізована шляхом використання PayPal REST API;

– модуль використовує новий, більш довгостроковий токен доступу для генерації токенів CLIENT_ID та SECRET_ID.

CLIENT_ID та SECRET_ID – довгострокові токени, які використовуються у всіх запросах через PayPal SDK.

На рисунку 3.9 зображено схематичний приклад роботи OAuth та отримання модулем токену доступа користувача. Особливістю розробленого модуля є ще один проміжний етап, коли недовговічні токени доступу використовуються для генерації більш довгострокових токенів. Після цього довгострокові токени використовуються для отримання CLIENT_ID та SECRET_ID. Такий функціонал було розроблено з метою полегшення процесу налаштування, щоб користувачі мали змогу користуватися довгостроковими токенами і не проходити процес авторизації кожного разу.

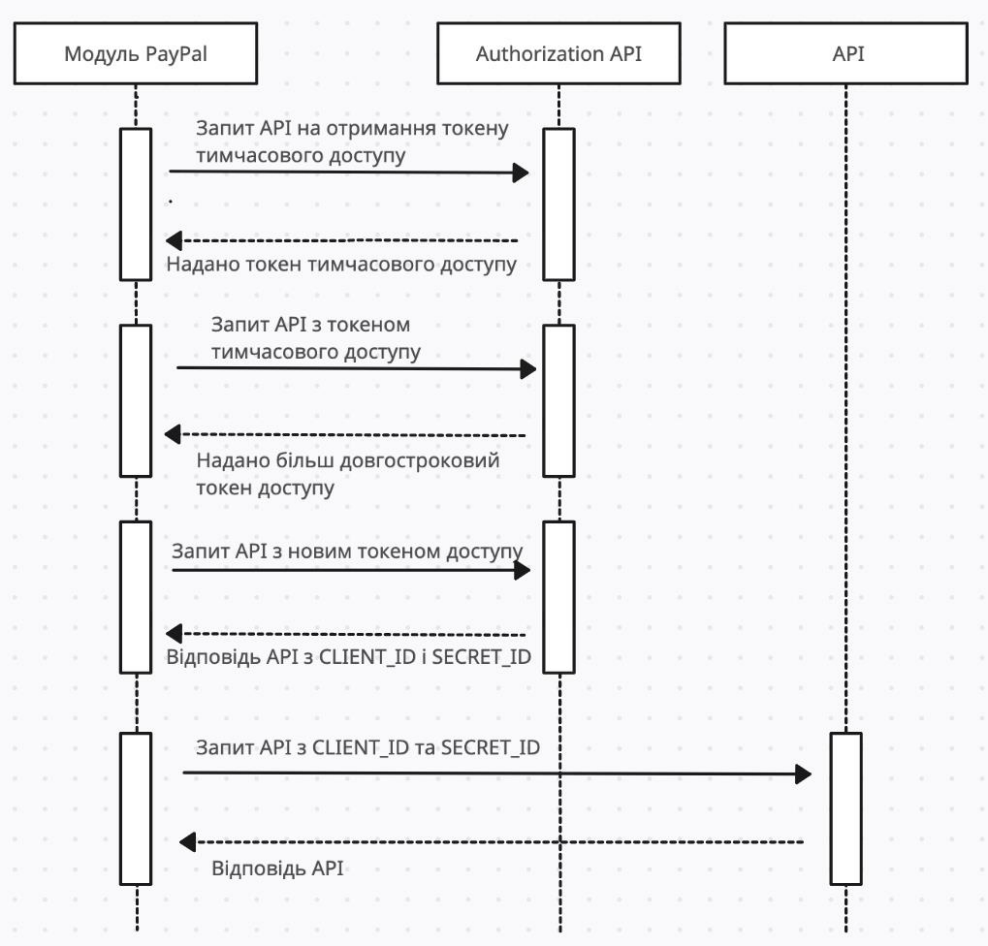

Рисунок 3.9 – Схема генерації токенів доступу, CLIENT_ID та SECRET_ID

3.2.4 Відображення модуля на передній частині вебсайту та використання хук PaymentOption

PrestaShop надає надійну систему хуків (hooks), яка дозволяє розробникам розширювати та змінювати функціональні можливості основної системи та модулів у певних точках виконання всередині PrestaShop без необхідності зміни вихідного коду ядра. Тобто хуки – це попередньо визначені точки в коді, куди можна вставляти спеціальний код, щоб додавати нові функції, змінювати наявні функції або взаємодіяти з даними.

Хуки працюють за принципом «подія-реакція». Ядро або модуль генерує подію (hook), а зареєстровані модулі можуть підписатися на цю подію та виконати свій код (хук-обробник) у відповідь на подію. Це дозволяє модулям

вносити зміни до виведення сторінки, додавати додатковий функціонал, змінювати дані та багато іншого.

PrestaShop пропонує різні типи хуків, які охоплюють різні області системи, наприклад хуки відображення, хуки дії та хуки фільтрів. Хуки відображення використовуються для додавання вмісту в певні позиції на зовнішніх сторінках, хуки дій запускаються, коли відбуваються певні дії, а хуки фільтрів дозволяють змінювати дані перед їх відображенням.

Крім того, хуки стратегічно розміщені по всій кодовій базі PrestaShop, дозволяючи підключатися до різних етапів візуалізації сторінки, виконання модуля, обробки замовлень тощо. Кожен хук має певне ім’я та місце в коді, куди можна додати спеціальний код.

Коли сторінка або дія виконується, PrestaShop автоматично запускає відповідні хуки та виконує зареєстрований код [39, 40]. Кілька модулів або нестандартних фрагментів коду можна приєднати до одного хука, і вони виконуються в порядку, визначеному послідовністю встановлення модуля або пріоритетом модуля.

На додаток до використання існуючих хуків, PrestaShop дозволяє розробникам створювати користувацькі хуки у своїх власних модулях або темах. Це забезпечує ще більшу гнучкість і можливості налаштування.

Під час розробки модуля було використано хук PaymentOption для відображення варіанту оплати «Pay with PayPal» на сторінці оформлення замовлення на вебсайті (рис. 3.10). Модуль реєструє хук PaymentOption під час встановлення, що було реалізовано через метод install під час розробки модуля. Хук PaymentOption дозволяє додавати власні методи оплати до наявного списку варіантів оплати.

Після натискання кнопки «Order with an obligation to pay», клієнта буде перенаправлено на сторінку замовлення в платіжній системі PayPal для підтвердження оплати.

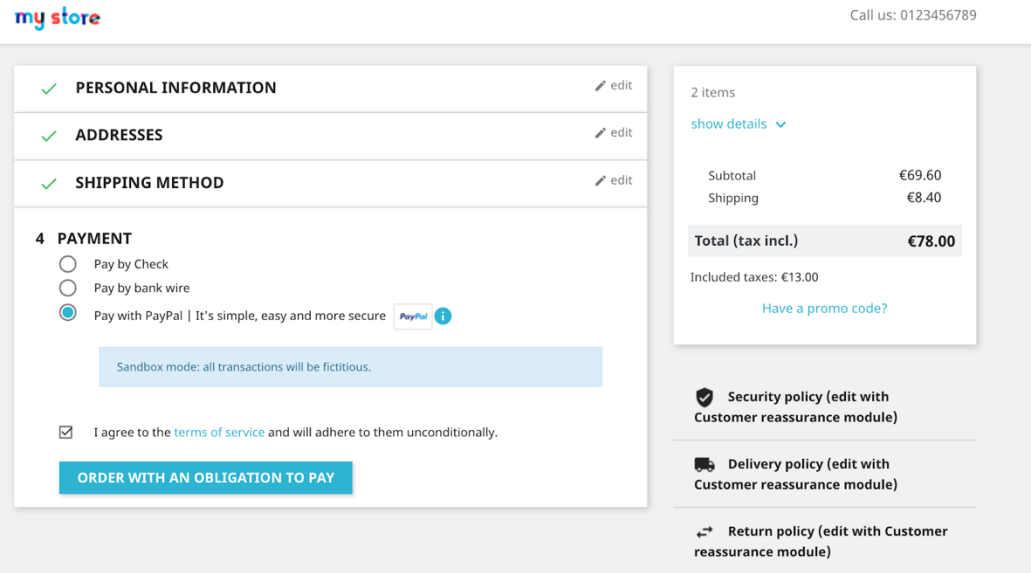

Рисунок 3.10 – Варіант оплати «Pay with PayPal»

Для використання PaymentOption було виконано наступні дії:

– створено новий модуль для CMS PrestaShop;

– створено новий файл PHP у каталозі модуля, який містить код способу оплати;

– у файлі PHP створено функцію, яка повертає екземпляр класу PaymentOption, який представляє розроблений спосіб оплати, тобто PayPal;

– у головному файлі модуля реалізовано функцію hookPaymentOption. Ця функція буде викликана PrestaShop щоразу, коли параметри оплати відображатимуться на сторінці оформлення замовлення;

– усередині функції hookPaymentOption викликається функція, створена на третьому кроці, яка повертає екземпляр PaymentOption для методу оплати.

3.2.5 Бізнес-логіка модуля

Після відображення модуля на передній частині вебсайту, наступним етапом кваліфікаційної роботи було розроблено бізнес-логіку модуля. Коли клієнт обирає варіант оплати Pay with PayPal на сторінці оформлення

замовлення і ініціює платіж, натиснувши кнопку, пов’язану з підтвердженням замовлення, розроблений модуль отримує запит від клієнта і надсилає платіжний запит на сервер PayPal, використовуючи PayPal REST API. У цьому запросі модуль передає усі необхідні дані клієнта, зібранні на сторінці оформлення замовлення, наприклад, суму замовлення та адресу доставки, а також дві URL адреси:

– куди перенаправляти клієнта, якщо платіж не буде підтверджено;
– куди перенаправляти клієнта після підтвердження платежу.

Отримавши API запит від модуля, платіжна система PayPal створює це замовлення та повертає у відповідь унікальний номер замовлення (ID). Розроблений модуль використовує отриманий унікальний номер замовлення, щоб перенаправити клієнта на сторінку цього замовлення в платіжній системі PayPal для підтвердження оплати.

Модуль було розроблено таким чином, що якщо клієнт не підтверджує платіж на сторінці замовлення в платіжній системі PayPal, він повертається на сторінку оформлення замовлення вебсайту.

Якщо користувач підтверджує платіж, то PayPal повертає результат операції. Далі платіжна система PayPal автоматично направляє користувача на один із контролерів розробленого модуля, який називається ecValidation (дані, куди перенаправляти користувача після підтвердження платежу, було визначено у платіжному запиті розробленого модуля до платіжної системи PayPal). На цьому етапі модуль відправляє новий запит до платіжної системи PayPal, використовуючи PayPal REST API, з метою захоплення грошей клієнта. Якщо операція захоплення грошей клієнта проходить успішно, модуль отримує відповідь Success від платіжної системи PayPal. Після цього використовується стандартний клас PaymentModule для створення замовлення в панелі адміністрації CMS PrestaShop. Оскільки було використано стандартний клас PrestaShop, замовлення буде створено автоматично системою керування вмістом PrestaShop.

Для роботи з базою даних модуль також використовує стандартний клас PrestaShop, який називається ObjectModel та забезпечує послідовний спосіб

взаємодії з базою даних і пропонує набір методів, які полегшують створення, читання, оновлення та видалення записів модулем. Розроблений модуль використовує стандартний клас ObjectModel як батьківський клас для власного класу PayPalOrder. Цей клас заощаджує багато часу та зусиль під час роботи із завданнями, пов'язаними з базою даних. PayPalOrder використовує шаблон активного запису, що означає, що кожен екземпляр класу представляє один запис у базі даних. Властивості класу відповідають полям у таблиці бази даних, а методи забезпечують спосіб маніпулювання цими полями. Розроблений модуль використовує наступні методи, надані ObjectModel:

– add() – створює новий запис у базі даних;
– update() – оновлює існуючий запис у базі даних;
– delete() – видаляє запис з бази даних;
– getFields() – отримує поля об'єкта у вигляді асоціативного масиву;
– validateFields() – перевіряє значення полів об'єкта.

3.2.6 Реалізація політики повернення

Клієнти цінують гнучкість і впевненість, що вони можуть скасувати замовлення і отримати повернення коштів у разі потреби. Таким чином, наявність політики повернення коштів має великий вплив на репутацію бренду. Саме тому розроблений модуль дозволяти не лише здійснювати оплату замовлення, але й повертати кошти при скасуванні замовлення.

Політику повернення було реалізовано шляхом використання стандартного хука PrestaShop, який називається actionOrderStatusUpdate. Цей хук спрацьовує, коли оновлюється статус замовлення, та дозволяє виконувати спеціальні дій на основі зміни статусу замовлення.

CMS PrestaShop надає можливість продавцям швидко та легко оновлювати статус замовлення через панель адміністрації (рис. 3.11). Після зміни статусу замовлення, модуль отримує актуальну інформацію про замовлення. Якщо статус замовлення було змінено на Refunded, модуль

відправляє новий запит на сервер PayPal, використовуючи PayPal REST API, з метою повернення коштів клієнту.

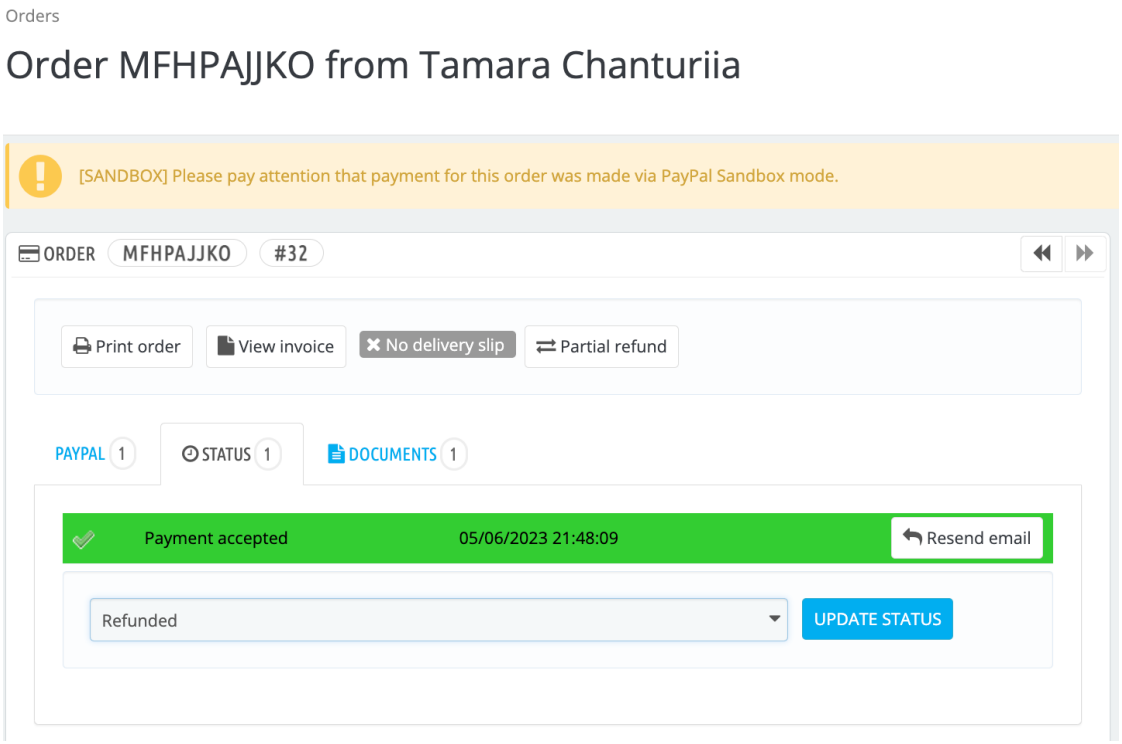

Рисунок 3.11 – Оновлення статусу замовлення для повернення коштів

Сума для повернення обчислюється на основі початкової суми транзакції та будь-яких відповідних комісій.

Отримавши API запит від модуля, платіжна система PayPal повертає кошти клієнту та відправляє відповідь модулю. В CMS PrestaShop продавець може переглянути деталі транзакції після повернення коштів на сторінці замовлення (рис. 3.12).

Усі платіжні операції, у тому числі повернення коштів, також доступні для перегляду через обліковий запис PayPal, який було використано під час розміщення замовлення.

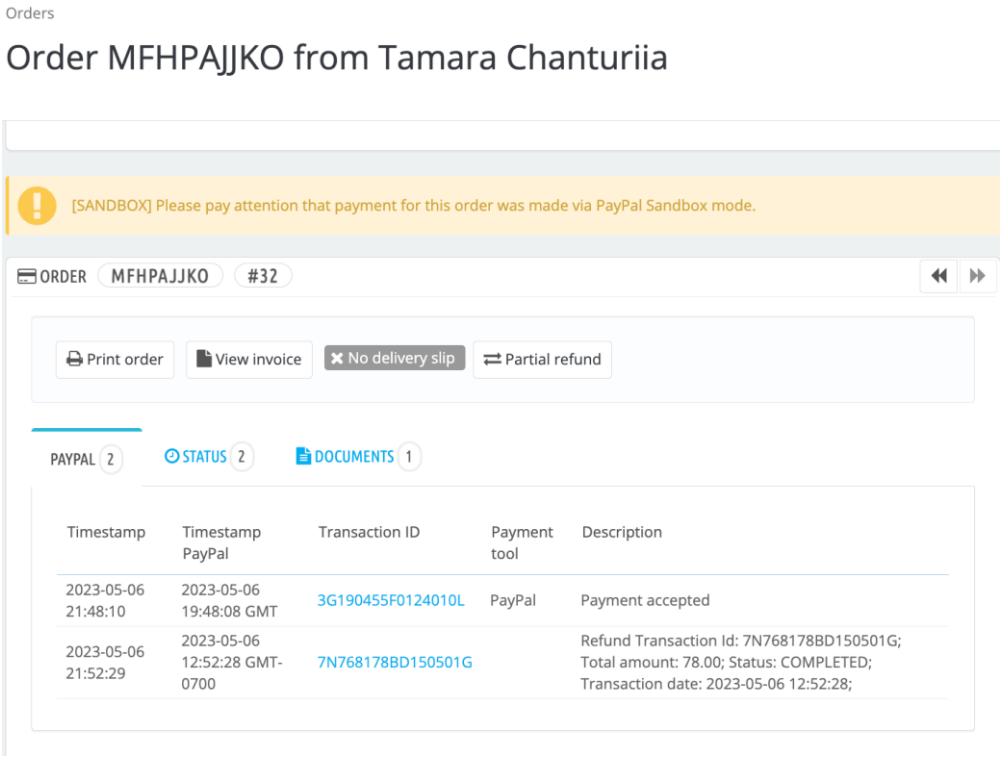

Рисунок 3.12 – Деталі транзакції після повернення коштів

3.3 Інструкція користувача

Розроблений модуль можна розділити на дві частини:

– адміністративна панель. Ця частина не є доступною для користувачів вебсайту та використовується безпосередньо для налаштування модуля. Елементи, використані для розробки даної частини, було більш детально описано в підрозділі 3.2;

– передня частина. Це та частина, яку користувачі можуть побачити на вебсайті, наприклад відображення варіанту оплати «Pay with PayPal» на сторінці оформлення замовлення онлайн-магазину.

Вкладка «Setup» розробленого модуля в адміністративній панелі PrestaShop дозволяє налаштовувати зв’язок модуля з платіжною системою PayPal, налаштовувати режими роботи модуля «Sandbox» та «Live», налаштовувати режим роботи модуля «Sale» або «Authorize», а також

переглядати основну інформацію про статус модуля. Вкладку «Setup» зображено на рисунку 3.13.

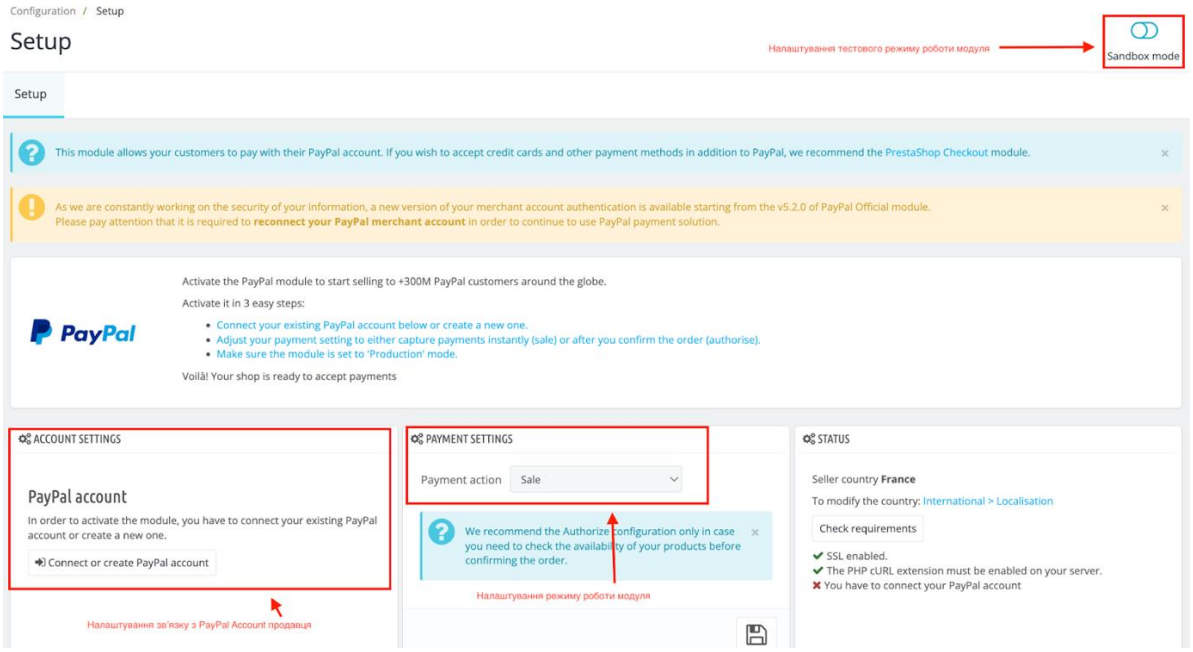

Рисунок 3.13 – Вкладка «Setup» розробленого модуля

Для налаштування модуля в першу чергу необхідно отримати облікові дані користувача (CLIENT_ID та SECRET_ID) за допомогою протоколу OAuth 2.0. Процес налаштування зв’язку з використанням протоколу OAuth 2.0 було описано у підпункті 3.2.3. Користувач може налаштувати зв’язок з платіжною системою PayPal у вкладці Account Settings.

Розроблений модуль також має параметри конфігурації для ввімкнення режиму Sandbox, тобто режиму, який дозволяє робити тестові покупки. Такий режим модуля використовує тестові рахунки та є надзвичайно важливим. Оскільки цей режим використовує окремий рахунки, тобто тестовий рахунок, усі параметри конфігурацій відокремлено від параметрів конфігурації модуля у режимі «Live».

Для роботи модуля в режимі Sandbox, користувач повинен мати облікові дані PayPal Sandbox. Користувач може увімкнути режим Sandbox у верхній правій частині у вкладці «Параметри» в адміністративній панелі.

Через секцію «Payment Settings» користувач має можливість налаштовувати режим роботи модуля, тобто режим платіжних дій. Можливі налаштування та принцип роботи було описано в підпункті 3.2.2.

3.4 Тестування розробленого модуля

Для тестування розробленого модуля було використано режим Sandbox та створено обліковий запис PayPal Sandbox середовища з метою створення тестових транзакції, не використовуючи реальних фінансових транзакцій.

Таким чином, в першу чергу було увімкнено тестовий режим модуля, вказавши відповідний параметр у налаштуваннях, та підключено обліковий запис PayPal Sandbox (рис. 3.14).

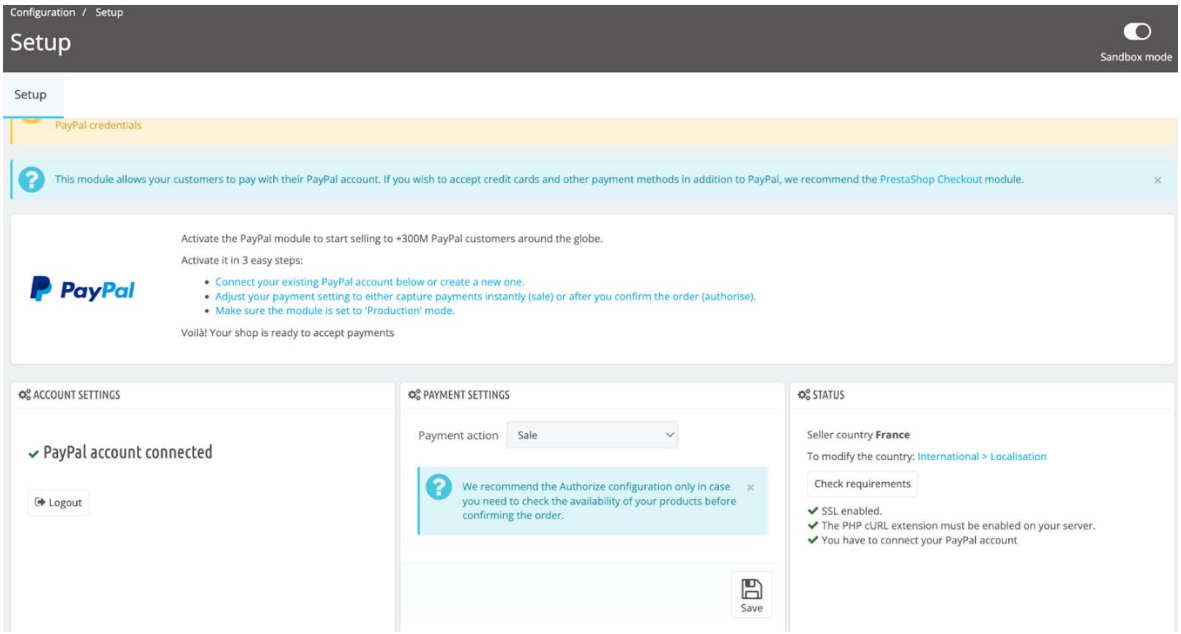

Рисунок 3.14 – Налаштування модуля для тестування в режимі Sandbox

Після налаштування тестового режиму було ретельно перевірено правильність роботи модуля шляхом створення тестових транзакцій. Модуль було протестовано спочатку в режимі платіжних дій «Sale», а потім в режимі платіжних дій «Authorize». Було проведено тестові платежі, перевірено

реакцію модуля на різні статуси транзакцій, включаючи успішні платежі, відхилені платежі та помилки, а також протестовано повернення коштів.

Для тестування модуля потрібно перейти до фронтенду магазину, обрати тестовий продукт та додати його до кошику. Після цього необхідно створити тестове замовлення, вказавши особисті або тестові дані, та провести тестову транзакцію, роблячи покупку з використанням платіжної системи PayPal як способу оплати. На цьому етапу рекомендовано провести тестування різних сценаріїв, наприклад розміщення замовлення із використанням спеціального промокоду магазину.

Після ініціації тестової транзакції на сторінці оформлення замовлення онлайн-магазину, модуль успішно направляє клієнта на платіжну систему PayPal для підтвердження оплати. Під час тестування було використано особистий обліковий запис PayPal Sandbox (рис. 3.15).

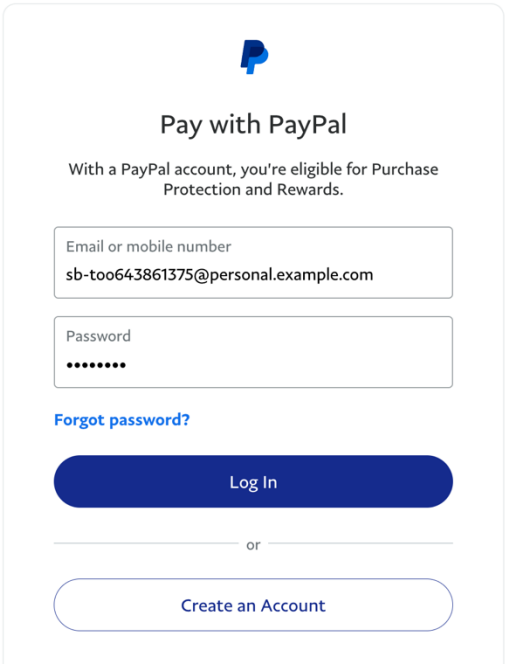

Рисунок 3.15 – Процес авторизації в PayPal Sandbox для підтвердження тестового замовлення

Після авторизації було підтверджено оплату замовлення (рис. 3.16). Модулем було використано PayPal REST API, операція захоплення грошей пройшла успішно.

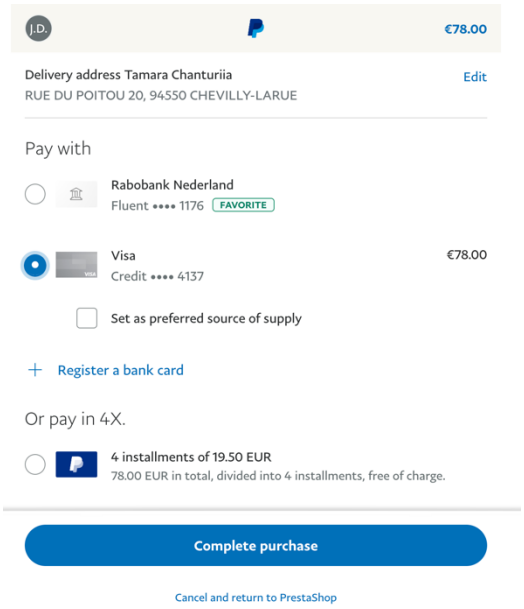

Рисунок 3.16 – Процес підтвердження оплати замовлення

Після підтвердження замовлення відбувається перенаправлення клієнта на сторінку вебсайту з інформацію про замовлення та повідомленням про підтвердження замовлення (рис. 3.17).

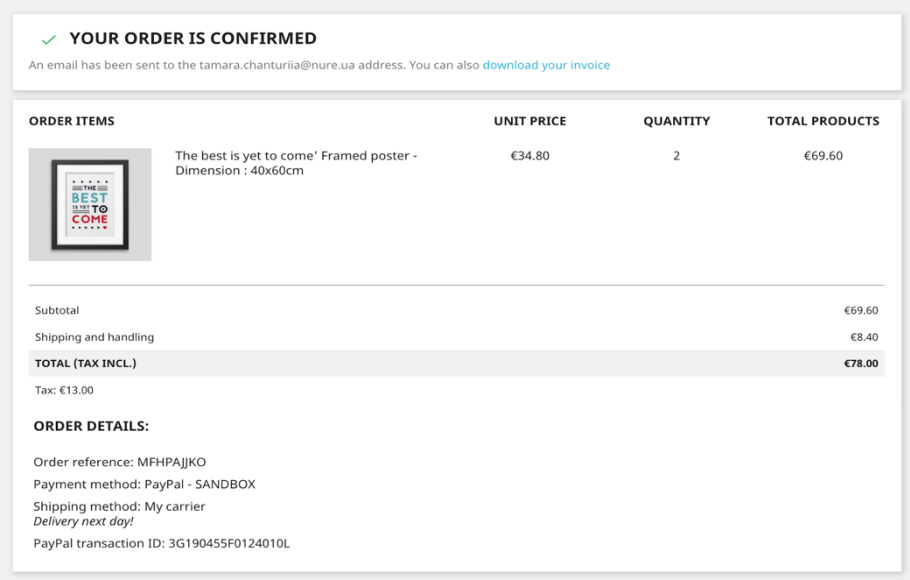

Рисунок 3.17 – Сторінка підтвердження замовлення

Таке саме замовлення було успішно створено в режимі платіжних дій «Authorize». Після цього було протестовано процес повернення коштів шляхом оновлення статусу замовлення в CMS PrestaShop.

На рисунку 3.18 можна побачити тестове замовлення, яке було створено в режимі платіжних дій «Authorize». Як було вказано в підпункті 3.2.2, для авторизації транзакції статус замовлення було вручну оновлено на «Payment Accepted» в адміністративній панелі, після чого операція захоплення грошей пройшла успішно. Після цього було авторизовано повернення коштів шляхом зміни статусу замовлення в адміністративній панелі на «Refunded».

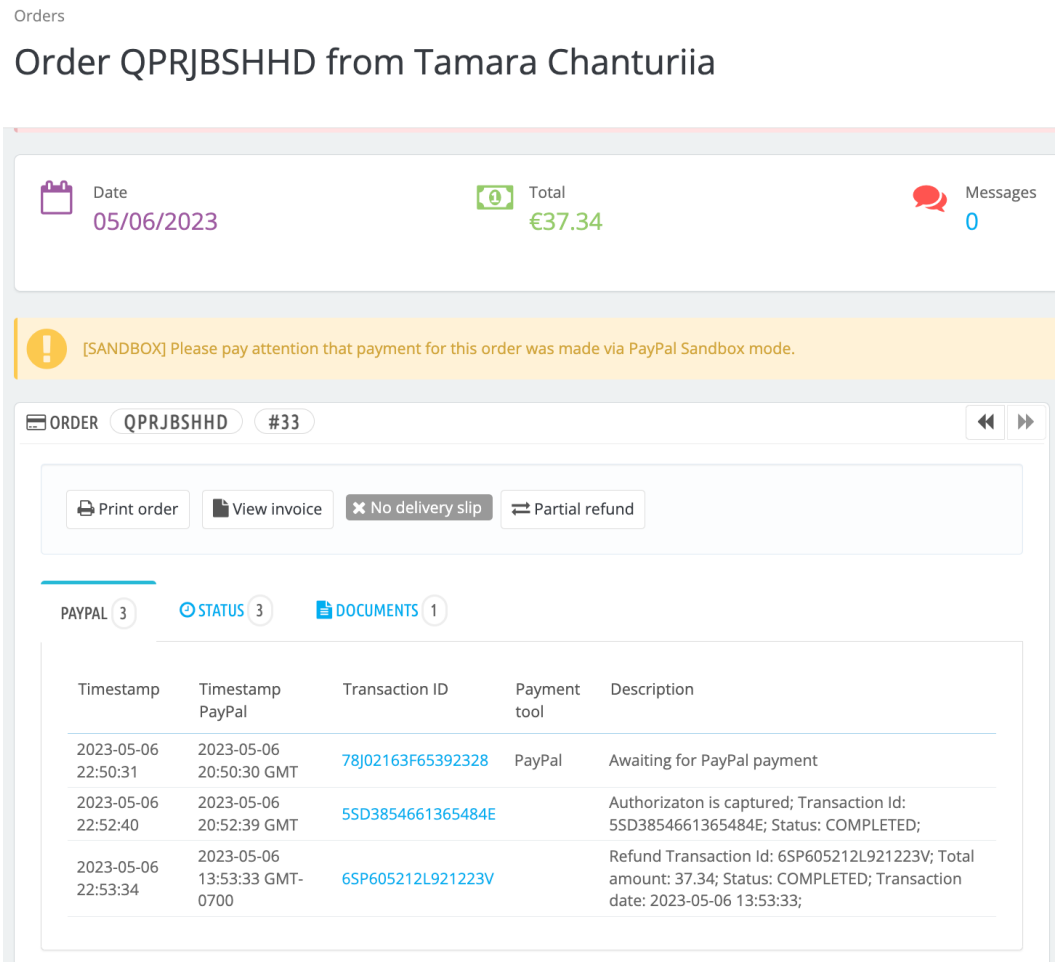

Рисунок 3.18 – Замовлення після авторизації транзакції та повернення коштів

Щоб перевірити, чи було успішно проведено платіж, і переконатися, що дані про транзакцію відображаються вірно, потрібно увійти до використаного Sandbox облікового запису PayPal. Обидві тестові транзакції, створені в режимі платіжних дій модуля «Sale» та «Authorize», було успішно проведено та авторизовано повернення коштів (після оновлення статусу замовлення на Refunded). Усі тестові платіжні операції доступні для перегляду в Sandbox

обліковому запису PayPal, який було використано під час тестування розробленого модуля (рис. 3.19).

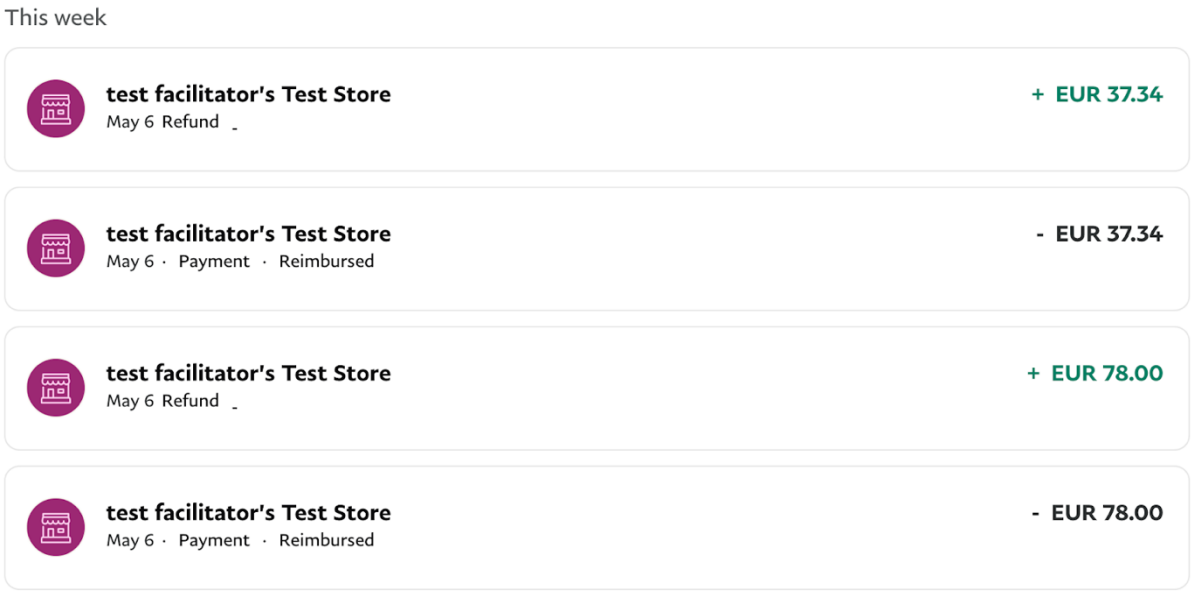

Рисунок 3.19 – Тестові платіжні операції

Підсумовуючи, створення тестових транзакції в Sandbox середовищі для перевірки і валідації інтеграції PayPal з магазином PrestaShop пройшло успішно. В цілому, розроблений модуль та інтеграція працюють коректно.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований модуль, який дозволяє інтегрувати платіжну систему PayPal з e-commerce системами керування вмістом PrestaShop для інтернет-магазинів.

В час глобалізації та інтернету, електронна комерція у всіх на слуху, і з кожним роком інтерес до неї стає дедалі більшим. Бізнес стає усе більше зацікавленим у можливості просувати та надавати свої товари та послуги через інтернет.

Найважливішим етапом для електронної комерції і бізнесу є процес оформлення замовлення та отримання оплати онлайн. Саме тому методи розробки різноманітних платіжних систем та методів їх інтеграцій є дуже актуальними.

Розроблений у рамках кваліфікаційної роботи модуль дозволяє зручно налаштувати платіжну систему PayPal і забезпечує її інтеграцію з інтернет-магазином PrestaShop. При використанні розробленого модуля, покупці можуть обирати PayPal як спосіб оплати при оформленні замовлення.

PayPal – всесвітньо відомий платіжний інструмент, з яким знайома більшість людей у світі. Інтеграція такої платіжної системи з e-commerce системами керування вмістом PrestaShop для інтернет-магазинів є важливим кроком для бізнеса. Оскільки PayPal дуже часто використовується при оформленні замовлення, можна сказати, що розроблений модуль інтеграція має чималий вплив на прибуток інтернет-магазину.

Розроблений модуль володіє наступними характеристиками:

– модуль дозволяє не тільки здійснювати оплату замовлення, а й повертає кошти при скасуванні замовлення;

– модуль має режим Sandbox, який дозволяє здійснювати тестові покупки;

– модуль має два доступні режими платіжних дій;

– параметри конфігурації модуля доступні із серверної частини.

Підсумовуючи, ця кваліфікаційної роботи дозволила мені значно поглибити свої навички вебінтеграції. Під час виконання роботи було вдосконалено навички в галузі обчислювальної техніки загалом, а також знання з різних мов програмування (html, css, php, JavaScript).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ю. В. Машика (2011). Електронна комерція – платіжні системи мережі Інтернет. Науковий вісник НЛТУ України , 21 (8), 344-348.

2. ІГ, Х., & ЄО, С. (2021). ЕЛЕКТРОННІ ПЛАТІЖНІ СИСТЕМИ: ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ ТА ОЦІНКА ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ. *Problems of Economy*, *48*(4).

3. Stripe Payments. URL: https://stripe.com/docs/payments (дата звернення 24.04.2023).

4. Stripe API Reference. URL: https://stripe.com/docs/api (дата звернення 24.04.2023).

5. Braintree Developer Documentation. URL: https://developer.paypal.com/braintree/docs (дата звернення 24.04.2023).

6. Юровський, В. Е. (2020). Розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції.

7. PayPal Developer. URL: https://developer.paypal.com/home (дата звернення 24.04.2023).

8. Get started with PayPal APIs. URL: https://developer.paypal.com/api/rest/ (дата звернення 24.04.2023).

9. PayPal JavaScript SDK. URL: https://developer.paypal.com/sdk/js/ (дата звернення 24.04.2023).

10. Таценко, М. І. (2020). *Методи організації даних у розподілених платіжних системах* (Master's thesis, КПІ ім. Ігоря Сікорського).

11. Kobylin,O.A., Gorokhovatskyi,V.O., Tvoroshenko,I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

12. Колпаков, М. О., & Петренко, А. Б. Масштабування та посилення захисту даних веб-додатку відповідно до вимог стандартів PCI DSS, HIPAA/HITECH, FEDRAMP. *Захист інформації*, *20*(4), 215-220.

13. Lyashenko, V., Kobylin, O., & Ahmad, M. A. (2014). General methodology for implementation of image normalization procedure using its wavelet transform.

14. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

15. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy-clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 930-935). IEEE.

16. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative Based Clustering of Long Multivariate Sequences with Different Lengths. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 545-548). IEEE.

17. Бодянский, Е. В., Винокурова, Е. А., Пелешко, Д. Д., Кобылин, И. О., & Кобылин, О. А. (2017). Нечёткая кластеризация временных рядов с неравномерными и асинхронными тактами квантования. *Системи обробки інформації,* (5), 47-54.

18. Васянович, Є. А., & Катєльніков, Д. І. (2021). *Використання шаблонів проектування у сучасному підході до розробки програмного забезпечення* (Doctoral dissertation, ВНТУ).

19. Работягов, А. В., Ляшенко, В. В., & Кобылин, О. А. (2016). Сегментация сложных изображений цитологических препаратов.

20. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

21. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

22. Lyashenko, V., Mohammad, A., & Kobylin, O. (2015). Experiments with Fusion of Images with Use of Wavelet Transformation in Problems of the Text Information Analysis.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

25. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

26. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

27. Kobylin, O., & Lyashenko, V. (2014). Comparison of standard image edge detection techniques and of method based on wavelet transform.

28. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

29. Gorokhovatskiy, V. A., Kobylin, O. A., & Kulikov, Y. A. (2015). Application of Granulation of Feature Descriptions in Structural Image Recognition. *Telecommunications and Radio Engineering*, *74*(6).

30. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications* (pp. 210-230). Springer, Cham.

31. Lyashenko, V., Matarneh, R., Kobylin, O., & Putyatin, Y. (2016). Contour detection and allocation for cytological images using Wavelet analysis methodology.

32. PrestaShop tutorials and advice for e-commerce merchants and developers. URL: http://prestadb.com/ (дата звернення 24.04.2023).

33. Пурський, О. І., & Мазоха, Д. П. (2017). Метод побудови мережі вітрин інтернет-магазинів на основі архітектури MVC. *Бизнес Информ*, (10 (477)), 319-324.

34. Що таке рукостискання SSL/TLS? Пояснимо за 3 хвилини. URL: https://ssl.com.ua/blog/ukr/what-is-ssl-tls-handshake/ (дата звернення 24.04.2023).

35. Pérez Castaño Arnaldo. (2017). Prestashop recipes : a problem-solution approach.

36. Bevilacqua A. (2014). Redmine plugin extension and development : build stunning extensions quickly and efficiently by leveraging redmine's plugin facilities.

37. PrestaShop DevDocs. PrestaShop Modules. URL: https://devdocs.prestashop-project.org/8/modules/ (дата звернення 24.04.2023).

38. OAuth. URL: https://auth0.com/docs/protocols/protocol-oauth2.

39. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In *Eleventh International Conference on Machine Vision (ICMV 2018)* (Vol. 11041, pp. 176-183). SPIE.

40. Величко, С. В. (2022). *Засоби та механізми протидії DDoS-атакам* (Doctoral dissertation, Одеса).