МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА КОМПОНЕНТУ ДЛЯ ІНТЕГРАЦІЇ ПЛАТІЖНИХ СЕРВІСІВ ДО СИСТЕМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1219
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.Е. Юровський
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н., Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної математики, доцент, к.ф.-м.н., Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

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

Лісняк А.О.

(підпис)

« » 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Юровському Володимиру Едуардовичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції

керівник роботи (проекту) Лісняк Андрій Олександрович, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » 05 2020 року № 576-с

2. Строк подання студентом роботи 30.11.2020

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

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

1. Постановка задачі.
2. Огляд технологій.
3. Проектування компоненту.
4. Реалізація компоненту.

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

Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 20.05.2020

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	27.05.2020	Виконано
2.	Збір вихідних даних.	10.06.2020	Виконано
3.	Обробка методичних та теоретичних	12.08.2020	Виконано
	джерел.		
4.	Розробка першого і другого розділу.	14.10.2020	Виконано
5.	Розробка третього розділу.	04.11.2020	Виконано
6.	Оформлення і нормоконтроль	30.11.2020	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	16.12.2020	Виконано

Студент _____ В.Е. Юровський
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції»: 50 с., 20 рис., 5 табл., 21 джерело, 2 додатка.

ПЛАТІЖНИЙ ПРОВАЙДЕР, GOCARDLESS, LARAVEL, LIQPAY, PAYPAL, STRIPE.

Об’єкт дослідження – процес розробки платіжного компоненту.

Мета роботи: розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції.

Метод дослідження – об’єктно орієнтований аналіз, методи програмної інженерії.

У кваліфікаційній роботі виконано огляд платіжних сервісів Paypal, Stripe, Liqpay та GoCardless, розглянуто Php-фреймворки для розробки сучасних веб-додатків, створено діаграми у нотації UML, реалізовано компонент використовуючи обраний стек технологій і протестовано функціонал за допомогою виконання тестової покупки.

На основі використання Php-фреймворку Laravel та інтерфейсів прикладного програмування платіжних систем було створено компонент, який за допомогою єдиного інтерфейсу отримує на вхід дані покупця та товару, створює на їх основі запити відповідно до вимог кожного платіжного сервісу та обробляє результати їх виконання. Створений компонент розрахований для використання в інтернет-магазині при оплаті товару.

SUMMARY

Master’s Qualification Thesis «Development of the Payment Providers Integration Component for E-commerce System»: 50 pages, 20 figures, 5 tables, 21 references, 2 supplements.

GOCARDLESS, LARAVEL, LIQPAY, PAYMENT PROVIDER, PAYPAL, STRIPE.

The object of the study is development process of payment component.

The aim of the study is development of the payment providers integration component for E-commerce system.

The method of research are object-oriented analysis, methods of program engineering.

This qualification paper reviewed Paypal, Stripe, Liqpay and GoCardless payment services, examined Php-frameworks for development of modern web applications, created diagrams in UML notation, implemented a component using the selected technology stack and tested the functionality by performing a test purchase.

Based on the use of Php-framework Laravel and payment service’s APIs a component was made. Using single interface, it receives user’s and good’s data, makes a request based on the input according to requirements of every payment service and processes results of their execution. Created component was designed to use in online-stores during payment for the goods.

ЗМІСТ

ВСТУП

Розробка сайтів з функціоналом інтернет-магазину стала необхідністю для сучасного бізнесу пов'язаним з роздрібним продажем товарів. Цей факт обумовлюється тим, що населення віддає перевагу магазинам, які пропонують цифрову платформу для швидких та безпечних покупок в інтернеті. Для того щоб її створити розробники використовують цілий спектр вбудованих компонентів, одним з основних з яких є платіжний провайдер. Він є проміжною ланкою між веб-додатком та платіжним сервісом і допомагає безпечно здійснити грошовий обмін між покупцем та продавцем, що і визначило напрямок дослідження. Цей компонент пропонує єдиний стилізований під платформу інтерфейс для зручного вводу вхідних даних користувача до обраного платіжного сервісу та отримання відповідного повідомлення після спроби проведення грошової транзакції між аккаунтом покупця та бізнес-аккаунтом інтернет-магазину. Маючи на увазі важливість та актуальність цієї технології формулюється мета дослідження кваліфікаційної роботи.

Метою дослідження є розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції.

Для досягнення мети роботи сформульовано наступні завдання:

а) огляд поширених платіжних сервісів;

б) аналіз технологій для реалізації компоненту;

в) проектування компоненту;

г) реалізація компоненту;

д) створення документації по інтеграції платіжного компоненту.

Об’єктом дослідження є процес розробки платіжного компоненту. Предметом дослідження є розробка компоненту для інтеграції платіжних сервісів до систем електронної комерції.

Програмний продукт-результат реалізовує єдиний інтерфейс для обміну даними між інтернет-магазином та представленими платіжними сервісами з ціллю проведення грошової транзакції між аккаунтами покупця та продавця.

Кваліфікаційна робота складається з реферату, вступу, трьох розділів (16 підрозділів), висновків, переліку посилань (21 найменувань) та 2 додатка.

1 ОГЛЯД ТЕХНОЛОГІЙ

1.1 Огляд платіжних провайдерів

Платіжний провайдер [1] – це платформа, яка дозволяє проводити швидкі та безпечні грошові операції всередині системи між її учасниками та між користувачем сервісу та зовнішньою сутністю. Маючи єдиний інтерфейс, він дозволяє отримувати платежі за допомогою цілої низки методів починаючи від кредитної карти і закінчуючи електронним гаманцем.

Таке поняття, як платіжний провайдер з’явилось після того, як коло підприємців зрозуміло, що тримати велику кількість рахунків в різних банках для отримання платежів від покупців ставало нерентабельним, особливо враховуючи що гроші приходили з великою затримкою. Таким чином були створені перші платформи, які пропонували спеціальний торговий аккаунт, що приймав на себе гроші з усіх передбачених джерел, групував та відправляв їх на вказаний банківський рахунок підприємця з меншою затримкою у доставці.

Існує три основні категорії платіжних провайдерів:

а) дистриб’ютор [2] – провайдер, який пропонує декілька методів оплати, з кожним з яких потрібно укласти договір та кожен з яких має окремий рахунок;

б) збирач [2] – провайдер, який пропонує декілька методів оплати, з кожним з яких потрібно укласти договір та кошти з яких накопичуються на одному рахунку торговця;

в) агрегатор [2] – провайдер, який пропонує декілька методів оплати, з кожним з яких вкладено договір та кошти з яких накопичуються на одному рахунку торговця.

Комбінація з цих систем при створенні веб-додатку збільшує ймовірність того, що покупець знайде підходящий метод оплати саме у

цьому інтернет-магазині і не надасть перевагу сайту-конкуренту через їх нестачу. Для того, щоб охопити більшу частину сегменту варіантів оплати в інтернеті для користувачів з України та Європи, компонент включатиме у собі наступні платіжні провайдери:

а) Paypal;

б) Stripe;

в) Liqpay;

г) GoCardless.

Paypal [3] – це американська e-commerce компанія створена у 1998 році, що спеціалізується на проведенні грошових транзакцій онлайн. Станом на 2019 рік, вона працює з 25 валютами та пропонує пакет послуг платіжного провайдеру для більш ніж 300 мільйонів користувачів, що включають в себе звичайних платників і представників великого та малого бізнесу згідно з [4]. Для проведення транзакцій користувачі сервісу мають можливість використовувати кошти з банківських рахунків, дебетових і кредитних карток або електронних гаманців прив’язаних до аккаунту Paypal. Об’єм функціоналу аккаунту платіжного провайдеру неоднорідний, різниться від регіону до регіону та поділяється на чотири основні категорії:

а) Send only [3] – аккаунти з можливістю тільки відправляти кошти (працює у 97 країнах);

б) PayPal Zero [3] – аккаунти з функціоналом відправлення та отримання коштів без можливості їх зберігання на електронному гаманці Paypal (працює у 18 країнах);

в) SRW Send - Receive – Withdrawal [3] – аккаунти, які мають увесь вищезазначений функціонал та можливість зберігання коштів на електронному гаманці (працює у 41 державі);

г) Local Currency [3] – аккаунти з вищезазначеним функціоналом та можливістю тримати кошти у місцевій валюті (працює у 21 державі).

Paypal є платіжним провайдером-збирачем, а отже для автоматизування ведення грошового обороту пропонує спеціальний мерчант-аккаунт [5], який

отримуватиме та оброблятиме платежі з будь-яких вищезазначених джерел за допомогою API, групуватиме їх на одному рахунку та відправлятиме за потребою користувача. На виході, в залежності від країни відправника та отримувача коштів, система стягує приблизно 2,9% + 0,3 доларів США [6].

Stripe [7] – американська компанія, основна спеціалізація якої є надання послуг у сфері обробки платежів та їх автоматизації за допомогою API. Вона працює з більш ніж сотнею валют та приймає платежі зі стандартних джерел, таких як кредитні та дебетові карти так і з мобільних гаманців як Apple та Google Pay згідно з [8]. Stripe реалізує всі можливості платіжного провайдера-агрегатора, що значить що продавцю для ведення свого онлайн бізнесу потрібно подбати лише про методи обробки користувацького платежу. Для цього він надає цілу низку API викликів, які зручно реалізуються за допомогою фреймворку Laravel у пакеті Laravel Cashier [9]. Подбавши про реалізацію платіжного функціоналу веб-додатку продавець матиме можливість отримувати кошти з комісією від 2,9% + 0,3 доларів США [8].

Liqpay [10] – українська компанія, що реалізувала веб-інтерфейс для обробки та отримання платежів на рахунок ПриватБанку. Вона тісно співпрацює з клієнтами вищезазначеної інституції та приймає кошти власників карт Mastercard або Visa зі всього світу згідно з [11]. Будучи платіжним провайдером-збирачем, Liqpay надає спеціальний мерчант-аккаунт [11], у комплекті з яким розробник отримує стислу документацію по основним запитам до платіжної системи, які дуже зручно реалізуються за допомогою інструментів мови PHP та фреймворку Laravel. Liqpay є нативною українською платформою для обробки платежів у гривні, а отже має бути обов’язковою будь якого веб-додатку, який працює з фінансами та функціонує на території України. Маючи це на увазі, комісія при покупці товару за допомогою цього платіжного провайдеру на вищевказаній території складає 2,75% [11].

GoCardless [12] – англійська компанія, розробник платіжного провайдеру-дистриб'ютору, що надає можливість інтеграції функціоналу прямого дебетування [13] у веб-додаток. Це означає що продавець має можливість висилати рахунки для оплати за допомогою обраного споживачем банку. Для реалізації цього процесу розробнику веб-додатку потрібно створити мерчант-аккаунт [14] та скористатись документацією по створенню спеціальної форми для вводу чутливих даних користувача, які потрібні для створення обраного типу платежу. Після інтеграції цієї системи користувачу потрібно лише раз провести платіж для того, щоб сервіс його запам'ятав та ініціалізовував його виконання за потребою покупця. Комісія по транзакціям за допомогою платіжного провайдеру GoCardless складає 1% + 0,2 євро [15] на території Євросоюзу.

Порівняльна характеристика вищезазначених технологій наведена у таблиці 1.1:

Таблиця 1.1 – Порівняльна характеристика платіжних провайдерів

Назва провайдеру	Тип	Спеціалізація	Методи оплати	Комісія по транзакції
Paypal	Збирач	Світ	Банківські картки та рахунки, електронний гаманець	2,9% + 0,3$
Stripe	Агрегатор	Світ	Банківські картки та рахунки, електронні гаманці Apple, Google Pay	2,9% + 0,3$

Продовження таблиці 1.1

Назва провайдеру	Тип	Спеціалізація	Методи оплати	Комісія по транзакції
Liqpay	Збирач	Україна	Банківські картки	2,75%
GoCardless	Дистриб’ютор	Країни ЄС	Банківські картки, електронний гаманець Paypal	1% + 0,2€

Кожен з представлених платіжних провайдерів охоплює певний сегмент потенціальних користувачів і потребує свого інтерфейсу для вводу чутливих даних. Для того щоб вирішити питання захаращення сторінки вищевказаними формами використовується компонент для інтеграції платіжних сервісів до систем електронної комерції. Він має єдиний стилізований інтерфейс для вводу користувацьких даних, який адаптується під потреби певного провайдера та проводить з ним транзакції після схвалення покупки споживачем. Для реалізації цього компоненту потрібно обрати гнучкий Php-фреймворк, що задовольняє потреби усіх представлених сервісів та має готові фрагменти реалізації для швидкої та надійної імплементації заявлених можливостей. Для цього буде проведено аналіз найпопулярніших фреймворків на ринку у результаті якого буде обрано найбільш підходящий засіб програмування у випадку інтеграції платіжного функціоналу у Ecommerce систему.

1.2 Стек технологій для реалізації компоненту

Заявлений функціонал платіжного компоненту можливо реалізувати за допомогою звичайної мови програмування Php, але для того щоб підняти процес розробки на новий рівень, який дозволяє створити гнучкий та простий у масштабуванні програмний продукт використовуються сторонні інструменти, сукупністю яких є Php-фреймворки. Перевагами використання зазначених технологій у порівнянні зі звичайними засобами є:

а) прискорення розробки за рахунок використання готових методів, фрагментів реалізації та унікальних особливостей кожного з фреймворків;

б) спрощення в масштабуванні та підтримці проекту;

в) підвищена ступінь захисту остаточного веб-додатку.

Маючи це на увазі, для реалізації компоненту потрібно обрати сучасний фреймворк, який найбільше підходить для втілення платіжного функціоналу та підтримки його функціонування всередині Ecommerce системи. На основі цих запитів та вживаності при розробці програмних продуктів для аналізу було виокремлено наступні фреймворки:

а) Laravel;

б) Symfony;

в) Yii2.

Laravel [16] – найпопулярніший Php-фреймворк станом на 2020 рік, який дозволяє швидко розгорнути та реалізувати проект за допомогою близько 20000 пакетів від основних розробників і спеціальних рішень від хостерів веб-додатків.

Symfony [17] – популярний модульний Php-фреймворк для реалізації продуктивних веб-додатків, що інтегрується з інструментами фронт-енд розробки, такими як Angular JS та використовується у таких проектах, як Drupal та phpBB.

Yii2 [18] – швидкий Php-фреймворк, що реалізує концепт «ледачого завантаження» сторінки та який широко використовується серед фронт-енд розробників через його інтеграцію з jQuery та Ajax.

На основі можливостей кожного з представлених фреймворків було створено порівняльну характеристику, що зазначена у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика Php-фреймворків

Характеристика	Laravel	Symfony	Yii2
Ліцензія	MIT	MIT	BSD
Менеджер для встановлення пакетів	Composer	Composer	Composer
Шаблонізатор за замовчуванням	Blade	Twig	За замовчуванням не використовується
Використання моделі MVC	Є	Є	Є
Продуктивність у php 7.1 (кількість запитів у секунду)	600	1022	2664
Підтримка ORM	Є	Є	Є
Бази даних	MySQL, PostgreSQL, SQLite и SQL Server	MySQL, PostgreSQL, SQLite	MySQL, MariaDB, SQLite, PostgreSQL, CUBRID, Oracle, MSSQL
Захист від SQL-injection, XSS, CSRF	Є	Є	Є

Продовження таблиці 1.2

Характеристика	Laravel	Symfony	Yii2
Кількість пакетів	44719	36066	14802
Офіційний пакет для інтеграції платіжного функціоналу	Laravel Cashier	Немає	Yii2 Payment Module
Документація та співтовариство	Є, сильне англомовне, сильне російськомовне	Є, сильне англомовне	Є, сильне англомовне
Поріг входу для початку розробки веб-додатку	Низький	Високий	Середній

Маючи на увазі результати порівняльної характеристики можна зробити висновок, що найбільш ефективним фреймворком серед представлених є Yii2. Але оскільки у випадку розробки платіжного компоненту більше значення має швидкість і надійність імплементації функціоналу та підтримка засобів його реалізації з боку співтовариства та офіційних розробників, більш сприятливим варіантом є Laravel, за допомогою інструментів якого і буде реалізовано платіжний компонент.

1.3 Методи та засоби інтеграції провайдерів

Створення платіжного функціоналу у веб-додатку за допомогою Paypal API [19] потребує генерації спеціальних ключів, які даватимуть можливість відрізняти застосунок від інших та користуватись методами для обміну інформацією з серверами відповідного провайдеру. Після їх отримання проводиться ініціалізація застосунку за допомогою функції setConfig, після

виконання якої відкривається можливість для розробки платіжного контролеру, який буде обробляти всі наступні транзакції пов’язані з Paypal.

Основною функцією контролеру є проведення користувацького платежу. Він складається з наміру Intent, покупця Payer, посилання після оплати RedirectUrls та транзакції Transaction. Після заповнення цих полів відповідними даними викликається метод create, який створює платіж у системі Paypal та метод Redirect::away, який перенаправляє користувача на сторінку для перевірки та схвалення оплати відповідного продукту. Після завершення цього процесу веб-додаток виконує функцію execute та кошти користувача резервуються для проведення грошового обміну між продавцем та покупцем.

Розробка платіжного контролеру за допомогою провайдеру Stripe базується на викликах методів з пакету Laravel Cashier [9] і використанні спеціально згенерованих ключів, які дозволяють обмінюватись інформацією між веб-додатком та відповідним сервером. Для цього у розпорядженні розробника є низка методів для роботи з методами оплати та підписками.

Для маніпуляції над методами оплати в Stripe передбачено наступні функції:

а) paymentMethods;

б) findPaymentMethod;

в) addPaymentMethod.

Функція paymentMethods повертає колекцію екземплярів існуючих у поточній системі методів оплати. Функція findPaymentMethod повертає конкретний метод, передбачений ідентифікатором-аргументом. Функція addPaymentMethod додає новий метод оплати, передбачений об’єктом-аргументом.

Для роботи з підписками у системі платіжного провайдеру в пакеті Laravel Cashier передбачено послідовний виклик функцій newSubscription та create. Спочатку вона ініціалізується за допомогою назви підписки та її

спеціального плану, а потім на основі цих даних, об’єкту paymentMethod та функції create вона вступає в силу у платіжній системі Stripe.

Реалізація базового платіжного функціоналу за допомогою інтерфейсу прикладного програмування Liqpay [20] починається з отримання пари ключів public_key та private_key. Вони потрібні для того, щоб ідентифікувати веб-додаток серед інших Liqpay застосунків та отримати доступ до методів для роботи з платежами. Після створення ключів за допомогою них проводиться ініціалізація об’єкту Liqpay, який тепер може представляти веб-додаток у іменному просторі платіжного провайдеру.

Для створення платежу потрібно розробити форму, яка зберігатиме інформацію про майбутню оплату та згенерувати унікальну сигнатуру, яка відповідатиме поточній транзакції. Для цього отримана інформація про покупку спочатку кодується за допомогою функції base64_encode, обгортається з обох сторін ключом private_key та проходить ще один цикл кодування використовуючи методи sha1 та base64_encode. Після закінчення підготовчих робіт формується запит до checkout серверу Liqpay та проводиться відповідна оплата продукту.

Інтеграція базових операцій з платіжним провайдером GoCardless складається з двох етапів: отримання access_token та формування запиту для маніпуляції над транзакцією за допомогою відповідних викликів до API.

Інтерфейс прикладного програмування GoCardless [21] пропонує дуже зручну імплементацію обробки платежу не потребуючу реалізації системних викликів до серверів платіжного провайдеру. Для цього потрібно лише ініціалізувати об’єкт GoCardlessPro\Client, за допомогою вищевказаного токену, і використати відповідний підхід передбачений документацією. У випадку створення платежу ним є послідовний виклик методів payments та create, останній з яких приймає на вхід інформацію про поточну покупку, а саме його опис, деталі про покупця та посилання на веб-додаток, яке спрацьовує після спроби оплати товару.

1.4 Висновки до розділу

Маючи на увазі інформацію отриману у результаті аналізу сфери надання платіжних послуг та стеку технологій, завдяки якому інтегрується товарно-грошовий функціонал у сучасних Ecommerce системах, для розробки компоненту було обрано Php-фреймворк Laravel.

Для того, щоб охопити більшу частину сегменту варіантів оплати, на основі впізнаваності, поширеності та популярності у використанні серед звичайних користувачів, було обрано наступні платіжні провайдери:

а) Paypal;

б) Stripe;

в) Liqpay;

г) GoCardless.

Інтерфейси прикладного програмування вищевказаних провайдерів у комбінації з зазначеним фреймворком формує інструментарій, який буде використовуватись при розробці остаточного програмного продукту. Для того, щоб перейти до реалізації компоненту, потрібно пройти наступний етап створення застосунку – проектування.

2 ПРОЕКТУВАННЯ КОМПОНЕНТУ

Оглянувши всі можливості представлених платіжних провайдерів та інструментів, які використовуються при їх реалізації, наступним кроком у життєвому циклі програмного забезпечення є проектування. За допомогою цього етапу відбувається перехід від абстрактних ідей та вимог замовника до реалізації конкретного макету майбутнього програмного продукту та дорожньої карти розробника, що міститиме кроки для виконання поставленої задачі. Для того, щоб спроектувати додаток існує ціла низка нотацій, кожна з яких описує конкретний рівень реалізації системи і для того щоб найбільш точніше це зробити, перебуваючи в рамках однієї нотації, було обрано мову UML. Вона містить у собі групу діаграм, які дозволяють зазирнути у концептуальний, логічний та фізичний рівень розробки програмного забезпечення і графічно представити його еволюцію від загального концепту до фінального продукту з деталізованими зв’язками між фізичними компонентами. Маючи це на увазі, для проектування платіжного компоненту було обрано діаграми варіантів використання, класів, діяльності, послідовності та компонентів.

2.1 Діаграма варіантів використання

Першим етапом створення дорожньої мапи розробника є реалізація діаграми на концептуальному рівні, якою у мові UML є діаграма варіантів використання. Вона графічно представляє взаємодію між двома основними елементами абстрактного бачення майбутньої системи – акторами та прецедентами. У частному випадку розробки платіжного компоненту, акторами є користувачі Ecommerce системи, що бажають здійснити покупки, та сервіси, які пропонують можливість оплати за обрані товари. Щодо інших

елементів Use Case діаграми – прецедентів, вони представляють собою функціонал поточної системи, який використовується або реалізується акторами на концептуальному рівні. Результат виконання діаграми варіантів використання зазначено на рисунку 2.1.

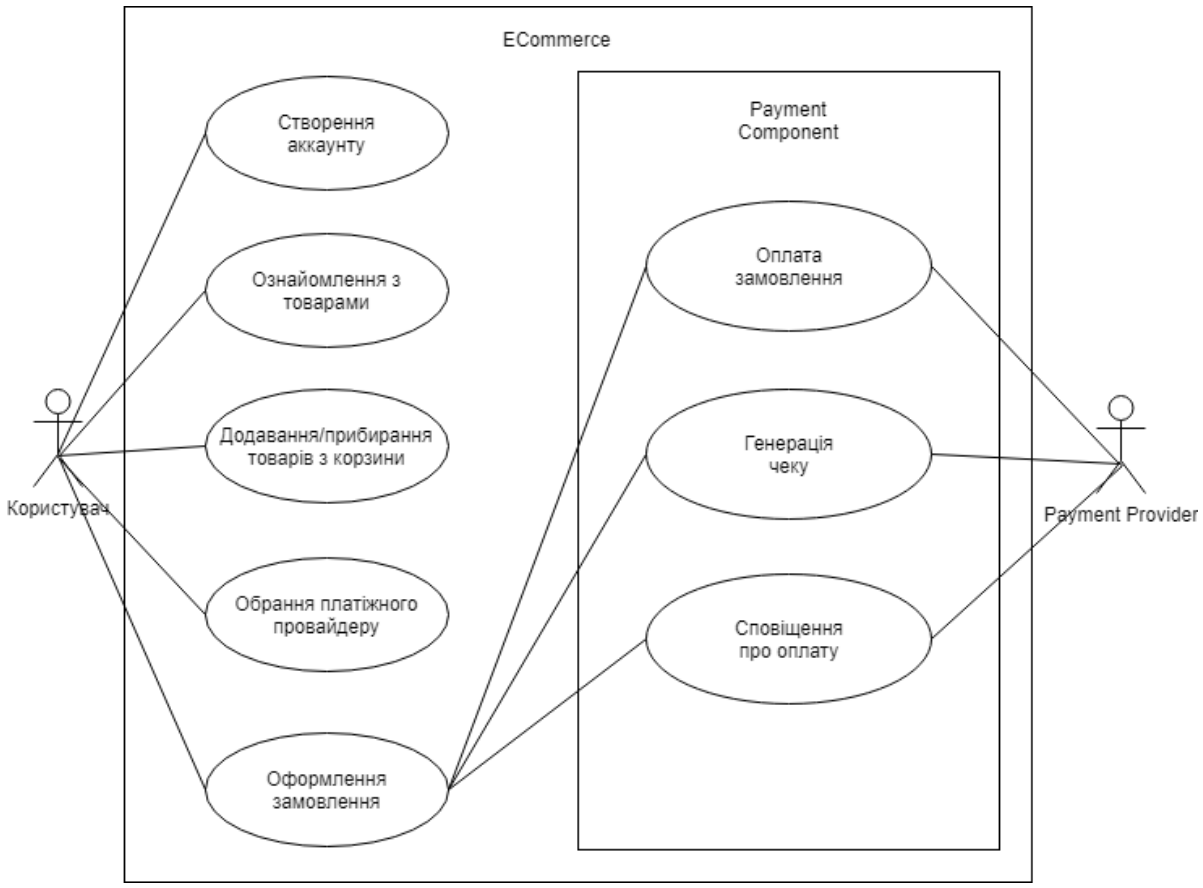

Рисунок 2.1 – Діаграма варіантів використання

2.2 Діаграма класів

Після завершення робіт на концептуальному рівні проекту наступним кроком переходу від абстракції до конкретики є логічний рівень, який графічно представляється діаграмою класів. Вона описує основні будівельні одиниці програмного забезпечення – класи та інтерфейси і показує зв’язки між ними. Крім описання структури проекту, кожен з його елементів має реалізувати певну частину функціоналу веб-додатку, що на діаграмі класів

виражається за допомогою методів. Маючи це на увазі платіжний компонент можна умовно поділити на три частини: контролер, платіжний шлюз та обробник замовлень.

Контролер складається з поєднання класів Payment та інтерфейсу IPayment, використовує інтерфейси IPaymentGateway та IOrderService для обміну даними з зовнішніми сутностями та системами і втілює першочергову обробку користувацьких даних щодо майбутнього платежу. Для реалізації наведеного функціоналу потрібно реалізувати основу, що складається з методів наведених у таблиці 2.1.

Таблиця 2.1 – Опис методів контролеру платіжного компоненту

Назва методу	Опис методу
render	Метод, що використовується для початку генерації форми для вводу чутливих даних користувача щодо оплати товару за допомогою обраного платіжного сервісу
processRefund	Метод, що використовується для початку процесу повертання грошей за зроблену раніше покупку
processRequest	Метод для ініціалізації грошового обміну між користувачем та Ecommerce сервісом
processResponse	Метод для обробки повідомлення, що було згенеровано платіжною системою
processNotification	Метод для генерації повідомлення відповідно до статусу покупки
registerGateway	Метод, що використовується для реєстраціїї нового платіжного шлюзу

Наступний елемент – платіжний шлюз, який втілюється за допомогою вищезазначеного інтерфейсу IPaymentGateway, що унаслідував поля та методи з цілої низки інтерфейсів: IResponse, IRefundResponse,

IPaymentGatewayNotification та IPaymentGatewayResponse. Його основна задача – реалізація грошового обміну між покупцем та Ecommerce системою використовуючи нативні для провайдерів засоби з відповідного інтерфейсу прикладного програмування. Основною вимогою до втілення IPaymentGateway є створення єдиного набору інструментів до кожного платіжного шлюзу для зручної імплементації сегменту варіантів оплати у поточний веб-додаток. Набор інструментів інтерфейсу складається з методів, що зазначені у таблиці 2.2, та які відповідають викликам, зробленими раніше контролером.

Таблиця 2.2 – Опис методів інтерфейсу платіжного шлюзу

Назва методу	Опис методу
render	Метод для генерації форми, що відповідає поточному платіжному сервісу
processRefund	Метод для обробки повернення коштів
processRequest	Метод для обробки платежу
processResponse	Метод для обробки повідомлення від платіжного сервісу
processNotification	Метод для зміни статусу покупки на сайті

Для отримання даних про поточні замовлення у Ecommerce системі використовується обробник замовлень. Він складається з інтерфейсу IOrder, який містить всю інформацію про поточну покупку; інтерфейсів ICustomer та IItem, які є його будівельними одиницями, що містять дані про покупця та товар відповідно і інтерфейс IOrderService який використовується для обміну даними з зовнішніми системами. Втілення основного функціоналу полягає у реалізації основних методів інтерфейсу IOrderService, що зазначені у таблиці 2.3.

Таблиця 2.3 – Опис методів інтерфейсу платіжного шлюзу

Назва методу	Опис методу
getOrder	Метод, який повертає поточне замовлення, що складається з товарів, їх кількості, покупця та валюти, яка використовується у Ecommerce системі;
applyResponse	Метод, який змінює статус замовлення в залежності від того, на якому етапі знаходиться транзакція між балансом інтернет-магазину та користувача.

Приймаючи до уваги вищевказані класи, інтерфейси та зв’язки між ними була створена повнорозмірна діаграма класів, що зображена на рисунку А.1 та зменшений її варіант, який представлено на рисунку 2.2.

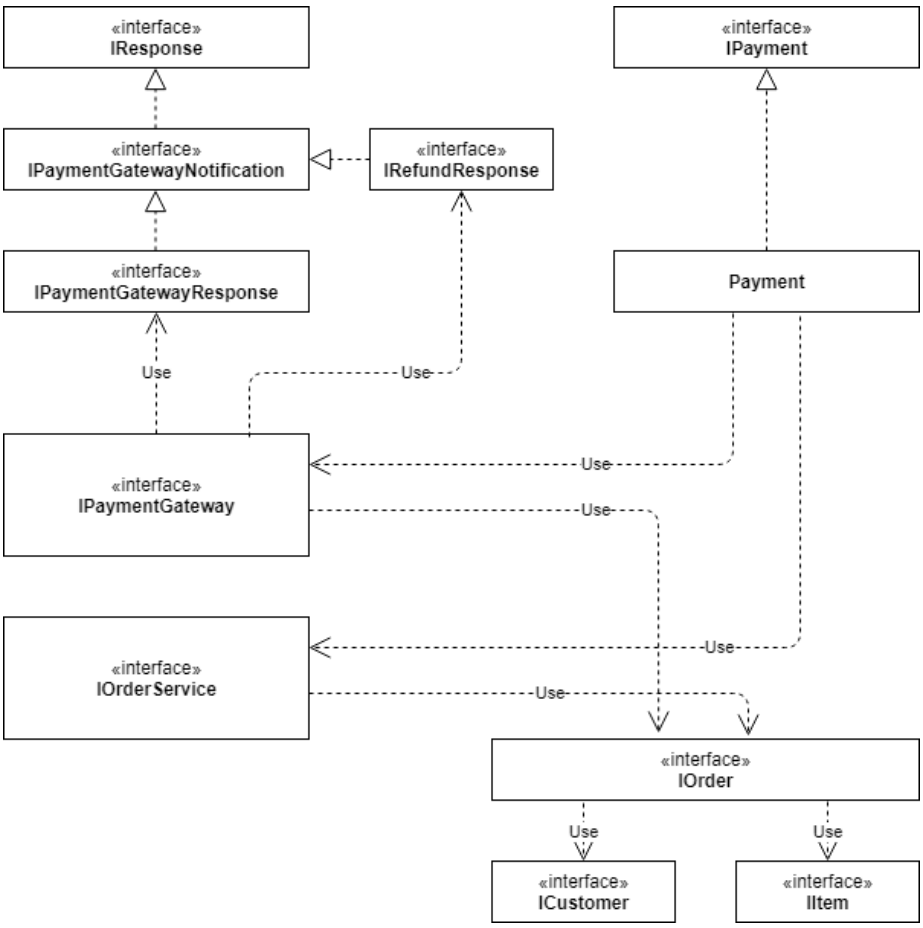

Рисунок 2.2 – Діаграма класів

2.3 Діаграма діяльності

Тепер маючи структуру проекту та перелік методів для реалізації функціоналу потрібно правильно описати порядок їх виклику. Для цього ця задача умовно ділиться на дослівний опис послідовності дій для досягнення результату та їх конкретизація за допомогою інструментів обраної мови програмування. Для виконання першої частини задачі у мові UML передбачено використання діаграми діяльності. Оскільки компонент має широкий спектр функціоналу, першочерговою та основною задачею у проектуванні цього компоненту є графічна візуалізація порядку дій при оплаті продукту, остаточний результат виконання якої зазначено на рисунку 2.3.

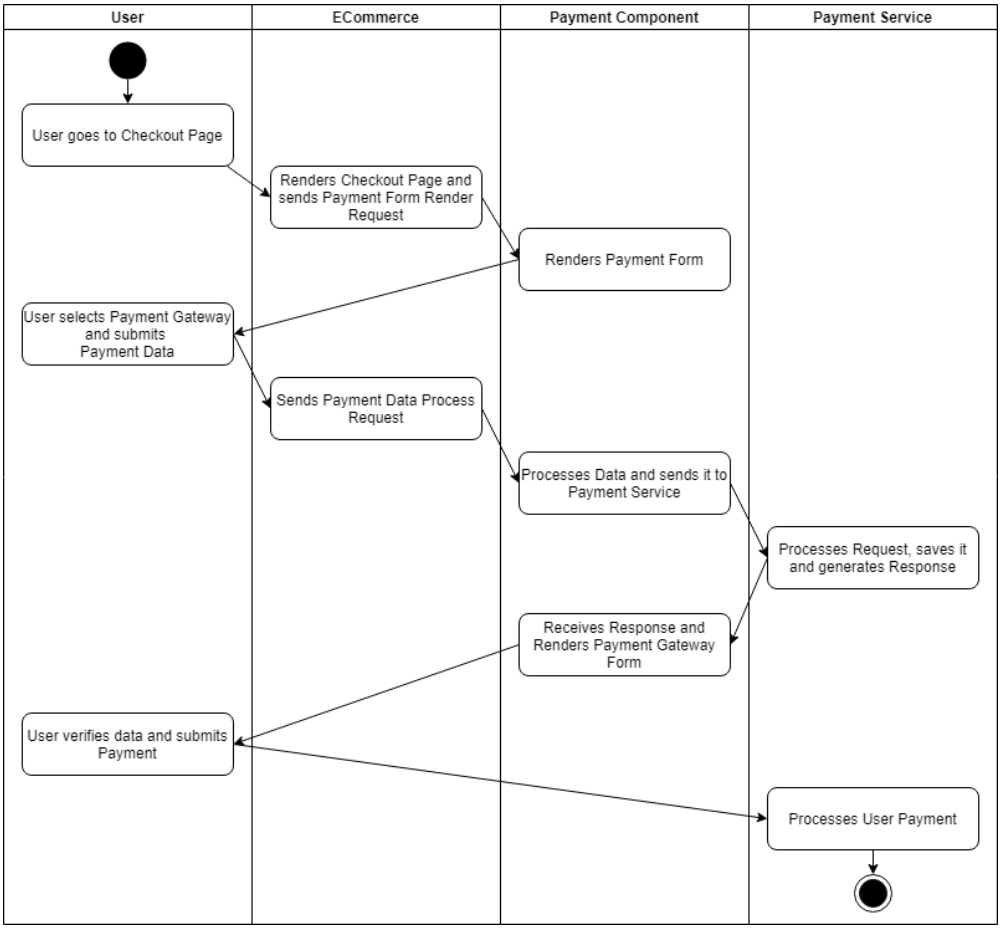

Рисунок 2.3 – Діаграма діяльності

Як можна побачити, для реалізації діаграми її площа поділяється на чотири лінії-сфери впливу: User, Ecommerce, Payment Component та Payment Service. Після цього обирається ініціатор та проводиться опис кожної дії, послідовність яких приводить до бажаного результату, а саме проведення грошової транзакції на сервері платіжного сервісу.

2.4 Діаграма послідовності

Виконавши графічну візуалізацію порядку дій при оплаті продукту було створено фундамент на якому за допомогою діаграми послідовності можна побудувати модель, яка міститиме послідовність конкретних викликів між основними компонентами системи та її користувачами.

Для її виконання площа діаграми умовно поділяється на три частини: актори, інтерфейси, контролери системи та зовнішні сутності, між якими за допомогою направлених стрілок відбуваються виклики API та обмін даними. У даному випадку актором є користувач-покупець, інтерфейсом є Ecommerce система, контролерами є Payment, IPaymentGateway IOrderService і зовнішньою сутністю є обраний Payment Service. Для реалізації оплати продукту спочатку робляться запити до контролерів для рендеру платіжної форми, потім користувач обирає платіжний сервіс, вводить чутливі дані пов’язаного з ним рахунку, переходить на відповідну сторінку оплати, проводить її та повертається на сторінку Ecommerce системи з відповідним до статусу транзакції повідомленням. Результат виконання графічної візуалізації вищевказаного алгоритму зображено на рисунку 2.4.

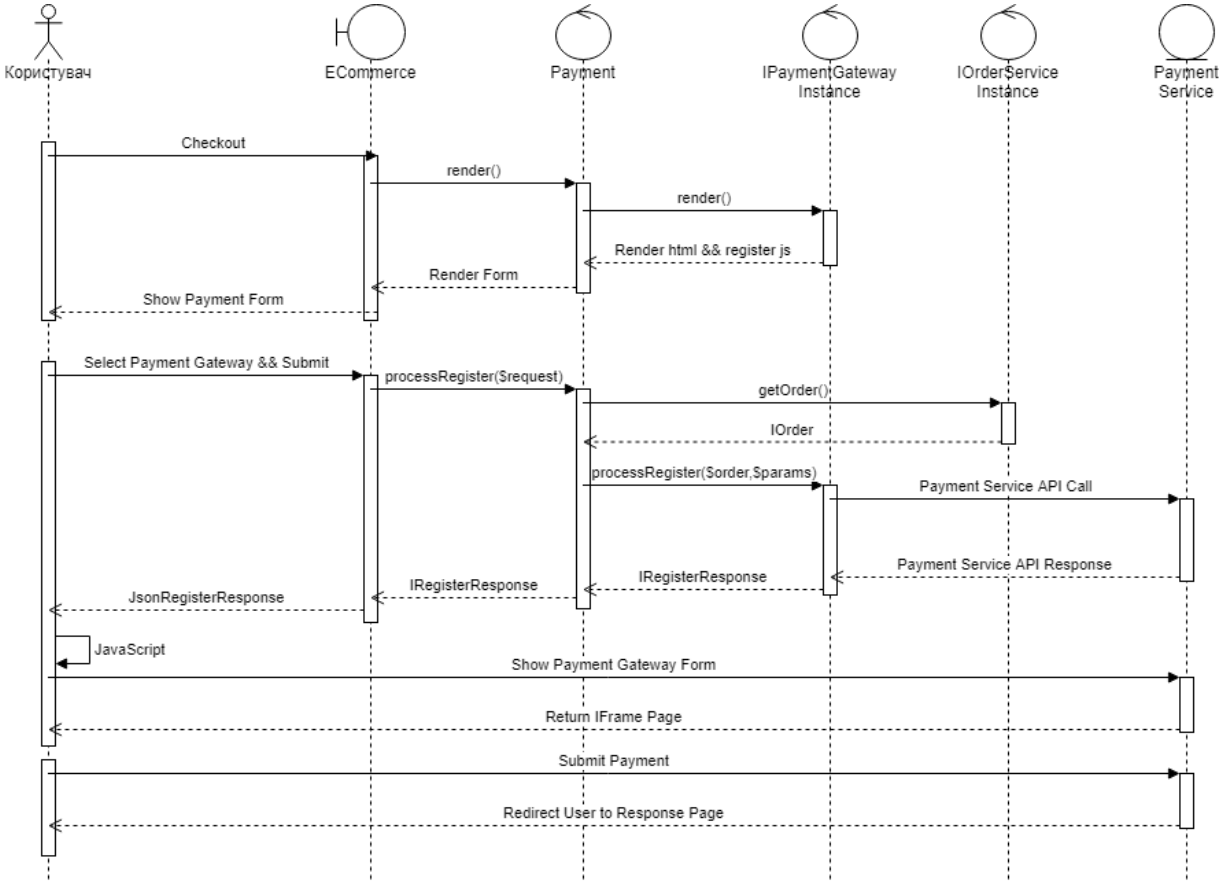

Рисунок 2.4 – Діаграма послідовності

2.5 Діаграма компонентів

Завершуючим етапом проектування за допомогою нотації UML є створення діаграми фізичних компонентів. Тут, маючи інформацію про структуру проекту та взаємодію між його основними елементами, будується модель, яка показуватиме яким чином буде виглядати остаточний продукт та які інтерфейси потрібні його компонентам для правильного функціонування.

У випадку платіжного компоненту для його інтеграції у Ecommerce систему потрібно мати низку платіжних шлюзів, які пропонують свій функціонал за допомогою інтерфейсу IPaymentGateway та компонент системи OrderService, що використовує для обміну даними інтерфейс IOrderService і надає всю інформацію про поточну покупку та конкретного

користувача. Діаграма, що утворилась на основі існуючих даних, які були отримані впродовж проектування програмного забезпечення наведено на рисунку 2.5

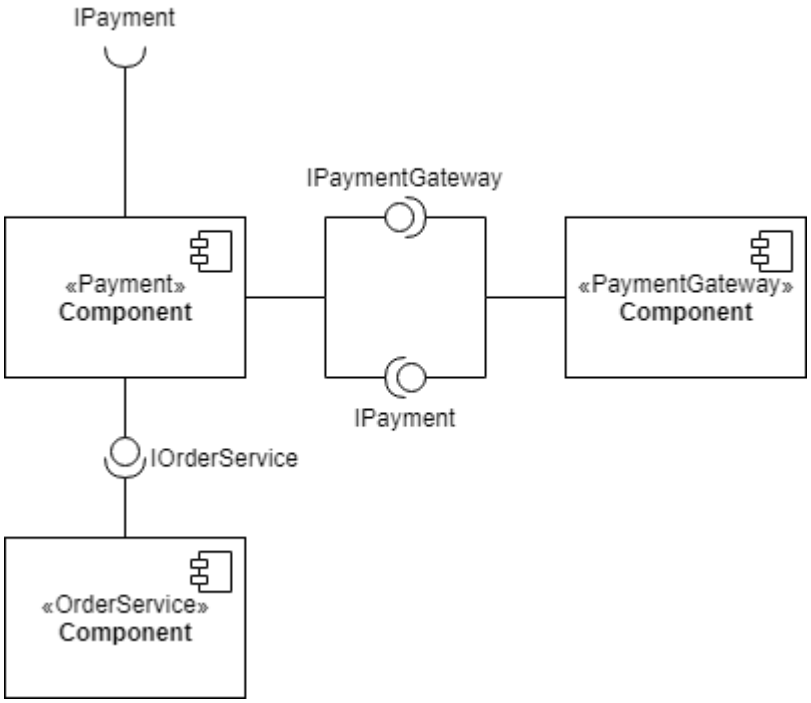

Рисунок 2.5 – Діаграма компонентів

2.6 Висновки до розділу

Для виконання проектування платіжного компоненту було використано нотацію UML. Всі діаграми, які були створені впродовж цього етапу життєвого циклу програмного забезпечення стали частиною дорожньої карти розробника, яка врахувала всі аспекти та вимоги при створенні платіжного функціоналу.

Відповідно до діаграми варіантів використання платіжний компонент має втілити наступні прецеденти:

а) реалізація оплати кожним з платіжних шлюзів;

б) генерація чеку після оплати замовлення;

в) сповіщення користувача про проведення оплати та її статусу;

Для реалізації вищевказаних прецедентів, відповідно до діаграм класів, діяльності, послідовності та компонентів, які утворились у результаті проектування, потрібно виконати три основні задачі:

а) створення основного контролеру Payment, який приймає на вхід дані користувача та робить виклики до обраного платіжного сервісу;

б) реалізація інтерфейсу до кожного з представлених платіжних шлюзів;

в) реалізація інтерфейсу до системи обліку замовлень поточного Ecommerce сервісу.

На основі цих даних час переходити до наступного етапу створення платіжного компоненту – розробки зазначеного функціоналу.

3 РЕАЛІЗАЦІЯ КОМПОНЕНТУ

Наступним етапом життєвого циклу програмного продукту є реалізація платіжного функціоналу використовуючи Php-фреймворк Laravel. Відповідно до результатів проектування основними елементами компоненту є клас Payment та інтерфейси IPayment, IPaymentGateway та IOrderService. Кожен з них реалізує певну частину загального програмного продукту, втілюючи методи зазначені у діаграмі класів. Остаточний додаток потребує здійснення кожного з них, але для проведення аналізу смислового наповнення важливо виокремити метод, реалізація якого є ключовим фактором успіху у розробці платіжного компоненту та є його уособленням, яке потребує додаткового огляду з точки зору його коду. Таким методом у цьому випадку є processRequest інтерфейсу IPaymentGateway. Він відіграє ключову роль при проведенні грошової транзакції між двома платіжними сутностями та має різне втілення у залежності від обраного користувачем сервісу. Тому для ознайомлення з основним функціоналом розробляємого програмного продукту було наведено реалізацію базового шлюзу EmptyPaymentGateway у додатку Б та проведено огляд втілення методу processRequest у кожному обраному PaymentGateway, а саме платіжному шлюзі Paypal, Stripe, Liqpay та GoCardless.

3.1 Імплементація провайдеру Paypal

Проведення оплати за допомогою платіжного сервісу Paypal відбувається за рахунок частини реалізації processRequest, що наведена на рисунку 3.1.

```
$payer = new Payer();
$payer->setPaymentMethod('paypal');

$amount = new Amount();
$amount->setCurrency($currency)
      ->setTotal($amountToBePaid);

$redirect_urls = new RedirectUrls();
$redirect_urls->setReturnUrl(URL::route('status'))
         ->setCancelUrl(URL::route('status'));

$transaction = new Transaction();
$transaction->setAmount($amount)
       ->setItemList($itemList)
       ->setDescription($orderDescription);

$payment = new Payment();
$payment->setIntent($intent)
       ->setPayer($payer)
       ->setRedirectUrls($redirect_urls)
       ->setTransactions(array($transaction));
try {
   $payment->create($this->_api_context);
} catch (\PayPal\Exception\PPConnectionException $ex) {
   if (\Config::get('app.debug')) {
      \Session::put('error', 'Таймаут соединения');
      return Redirect::route('/');
   } else {
      \Session::put('error', 'Возникла ошибка, извините за неудобство');
      return Redirect::route('/');
   }
}
```

Рисунок 3.1 – Реалізація оплати за допомогою Paypal API

Як бачимо, для проведення грошової транзакції за допомогою методу $payment->create потрібно навести низку аргументів таких як:

а) payer, що є поточним користувачем, який здіснює покупку у Ecommerce системі;

б) amount, що є остаточною грошовою сумою, яка вказана у конкретні валюті, що ініціалізується за допомогою методу setCurrency;

в) redirect_urls, що є масивом, який містить усі посилання на поточний сервіс, куди відправлятимуть користувача після проведення оплати;

г) itemList, що є масивом, який містить усі обрані покупцем товари;

д) orderDescription, що має опис поточного чеку, який відображатиметься після переходу до платіжного сервісу;

е) intent, що є наміром Ecommerce системи по відношенню до поточного користувача.

Після створення платежу користувач переноситься на сторінку Paypal з описом усіх вищевказаних аргументів, схвалює грошову транзакцію між своїм та балансом Ecommerce системи та переходить на сторінку інтернет-магазину, що відповідає статусу поточної оплати.

3.2 Імплементація провайдеру Stripe

Для того, щоб користувач мав можливість оплати підписки за допомогою платіжної системи Stripe, було імплементовано метод processRequest, який отримує дані про покупця та обраний ним цифровий товар і виконує послідовність викликів методів з пакету Laravel Cashier які проводять грошовий обмін між двома фінансовими сутностями на стороні вказаного платіжного сервісу. Частина реалізації вищевказаного методу зазначено на рисунку 3.2.

```php
// Ініціалізація користувача
$user = $request->user();

// Ініціалізація токену варіанту оплати
$ccToken = $request->input('cc_token');

// Ініціалізація плану обраного користувачем
$plan = $request->input('plan');

// Створення підписки
try {
    $user->newSubscription('main', $plan)->create($ccToken, [
        'email' => $user->email
    ]);
} catch (\Exception $e) {
    return back()->withErrors(['message' => 'Error creating subscription.']);
}
```

Рисунок 3.2 – Реалізація оплати за допомогою Stripe та Laravel Cashier

Для втілення платіжного функціоналу потрібно виконати наступну послідовність дій:

а) ініціалізувати користувача за допомогою даних вказани у аргументі методу processRequest;

б) ініціалізувати токен, що відповідає обраному методу оплати, серед тих які пропонує Stripe;

в) ініціалізувати план, який було обрано користувачем;

г) використати ініціалізовані дані для оформлення підписки за допомогою послідовності викликів методів newSubscription та create.

3.3 Імплементація провайдеру Liqpay

Інтеграція платіжного шлюзу Liqpay базується на імплементації низки методів інтерфейсу IPaymentGateway, що реалізують засоби для проведення грошових транзакцій у національній валюті між поточним користувачем та Ecommerce системою за допомогою відповідного інтерфейсу прикладного програмування.

Для реалізації оплати товару за допомогою Liqpay API документація пронує створити форму що має наступні поля:

а) version – поточна версія API;

б) sandbox – режим у якому знаходиться поточний веб-додаток (0 якщо веб-додаток знаходиться на стадії розробки і 1 якщо прийшов час релізу);

в) amount – грошова сума поточної покупки;

г) currency – валюта, у якій було розраховано чек;

д) description – опис чеку, який надається Ecommerce системою;

е) order_id – ідентифікатор поточного замовлення;

ж) type – тип транзакції;

з) result_url – посилання на Ecommerce ресурс після проведення оплати користувачем.

На основі документації по реалізації платіжного функціоналу, за допомогою Liqpay API було створено метод processRequest, який втілює у

життя можливість оплати за допомогою відповідного сервісу, реалізація форми якого вказана на рисунку 3.3.

```
$liqpay = new LiqPay($this->getValue('public_key'), $this->getValue('private_key'));
$html = $liqpay->cnb_form(
    array(
        'version' => '3',
        'sandbox' => '0',
        'amount' => $amount,
        'currency' => $currency,
        'description' => $description,
        'order_id' => $orderId,
        'language' => 'en',
        'result_url' => $redirectUrl
    )
);
```

Рисунок 3.3 – Реалізація форми оплати за допомогою Liqpay API

3.4 Імплементація провайдеру GoCardless

Імплементація платіжного функціоналу у GoCardless полягає у автоматизації наступних кроків:

а) створення унікального посилання на ресурс GoCardless, що відповідає обраному товару та користувачу;

б) схвалення покупки користувачем та надання спеціального мандату на зняття коштів з рахунку покупця;

в) зняття грошової суми з вказаного банківського рахунку користувача, що відповідає сумі чеку у Ecommerce системі.

Для створення посилання, що відповідає поточній покупці у GoCardless API передбачено послідовність викликів методів redirectFlows та create, що зображені на рисунку 3.4.

```
$redirectFlow = $client->redirectFlows()->create([
    "params" => [
        // This will be shown on the payment pages
        "description" => $description,
        // Not the access token
        "session_token" => $session_token,
        "success_redirect_url" => $redirectUrl,
        // Optionally, prefill customer details on the payment page
        "prefilled_customer" => [
          "given_name" => $firstName,
          "family_name" => $lastName,
          "email" => $email
        ]
    ]
]);
```

Рисунок 3.4 – Генерація посилання на ресурс GoCardless

Після цього користувач схвалює цю покупку і система отримує її іденфікатор на основі якого генерує мандат, що буде використовуватись при зняття платежу та код отримання якого зображено на рисунку 3.5.

```
$redirectFlow = $client->redirectFlows()->complete(
    $uniqueGoCardlessPaymentID, //The redirect flow ID from above.
    ["params" => ["session_token" => $session_token]]
);
```

Рисунок 3.5 – Генерація мандату на зняття коштів

Для завершення процесу оплати за допомогою GoCardless розробнику потрібно використати послідовність викликів методів payments та create, які приймають за аргументи грошову суму поточного чеку, валюту в якому вона вказана та ідентифікатор мандату, що був наданий користувачем та оброблений платіжним компонентом за допомогою функцій redirectFlows та complete.

Після завершення робіт над платіжними шлюзами та інтерфейсами для початкової обробки платежів та їх обліку потрібно виконати останній етап розробки платіжного компоненту – створення сторінок-повідомлень на які переходимуть користувачі після проходження через вищезазначений функціонал. Остаточна реалізація оплати чеку вказана на рисунку 3.6.

```
$payment = $client->payments()->create([
  "params" => [
      "amount" => $amount,
      "currency" => $currency,
      "links" => [
          "mandate" => $redirectFlow->links->mandate
                       // The mandate ID from last section
      ],
      // Almost all resources in the API let you store custom metadata,
      // which you can retrieve later
      "metadata" => [
          "invoice_number" => $invoiceId
      ]
  ]
]);
```

Рисунок 3.6 – Реалізація оплати чеку за допомогою GoCardless

3.5 Налаштування роботи платіжного компоненту

Для того, щоб додаток мав можливість робити виклики до відповідного інтерфейсу прикладного програмування потрібно ініціалізувати файл середовища .env ключами доступу, що були згенеровані при створенні проектів у кожному з обраних платіжних сервісів. Виявивши та зкомпонувавши ключі в один файл, він прийме вигляд, що зазначено на рисунку 3.7.

```
PAYPAL_MODE=sandbox
PAYPAL_CLIENT_ID=AXOEU2RpuneXRGnzw9JVExfm2R36ozmVVRAY6_aS4yKsoF8QlG3nh84JtXeEItDUXEkkLKOqQ8zQBkQv
PAYPAL_SECRET=XXXXXXXXXXXXXXX
STRIPE_KEY=STRIPE_SECRET=pk_test_51HjPrRDEsA9kGO7AO7aBSoo7qfz7zkGiMLxOVED7zj08WtZarIRZ2bd6ZZhXKrt8gJmVivHeCqKIfLlmFWFF8US100720jY9cm
STRIPE_SECRET=XXXXXXXXXXXXXXX
LIQPAY_PUBLIC_KEY=OBkB3fjqTUQkcPac42GKamFkpWgjCJAiQR61fKAc
LIQPAY_PRIVATE_KEY=XXXXXXXXXXXXXXX
GC_ACCESS_TOKEN=EoAPWc5KH2vKGoaPEQckTi5YlAxm23ETUkSPQ44xkFkTllCnYvkViTkV14E25Ws76Rkc9MAgG5NTi
```

Рисунок 3.7 – Файл середовища .env

Після цього для кожного платіжного провайдеру потрібно створити конфігураційний файл, що матиме налаштування, за допомогою яких у коді буде ініціалізовуватись об’єкт класу середовища, що міститиме усі виклики до відповідного платіжного сервісу. Кожен з них має ідентичну структуру тому для прикладу приводиться конфігураційний файл Paypal, вміст якого зображено на рисунку 3.8.

```php
return [
    'client_id' => env('PAYPAL_CLIENT_ID',''),
    'secret' => env('PAYPAL_SECRET',''),
    'settings' => array(
        'mode' => env('PAYPAL_MODE','sandbox'),
        'http.ConnectionTimeOut' => 30,
        'log.LogEnabled' => true,
        'log.FileName' => storage_path() . '/logs/paypal.log',
        'log.LogLevel' => 'ERROR'
    ),
];
```

Рисунок 3.8 – Конфігураційний файл Paypal

3.6 Використання компоненту у Ecommerce системі

Маючи на руках готовий платіжний компонент, розроблений за допомогою інструментів фреймворку Laravel, прийшов час для вбудовування його функціоналу у Ecommerce систему. Остаточний вигляд, який матимуть конфігураційні файли Ecommerce системи після маніпуляцій над ними наведено на рисунку 3.9.

config/app.php

```php
'providers' => [
    ...
    /*
    * Package Service Providers...
    */
    Payment\PaymentServiceProvider::class,
],
'aliases' => [
    'Payment' => Payment\Facades\Payment::class,
    ...
]
```

config/payment.php

```php
...
'paymentGateways' => [
    Micropayment\StripeGateway::class,
    Micropayment\LiqpayGateway::class,
    Micropayment\GoCardlessGateway::class,
    PayPal\PayPalGateway::class,
    EmptyPayment\EmptyPaymentGateway::class,
],
```

Рисунок 3.9 – Конфігураційні файли app.php та payment.php

Для того, щоб це зробити потрібно виконати наступні кроки:

а) додати до конфігураційного файлу config/app.php повний шлях до класу розробленого обробнику платежів;

б) додати до конфігураційного файлу config/payment.php повні шляжи до класів, що реалізують функціонал платіжних шлюзів;

в) додати виклик методу Payment::render() до реалізації вікна оплати товару.

3.7 Інструкція з інтеграції та використання

Після встановлення платіжного компоненту потрібно впевнитись, що працює увесь заявлений функціонал та після завершення покупки грошова сума перераховується на бізнес-аккаунт Ecommerce системи. Для цього потрібно обрати тестовий товар, що зазначено на рисунку 3.10.

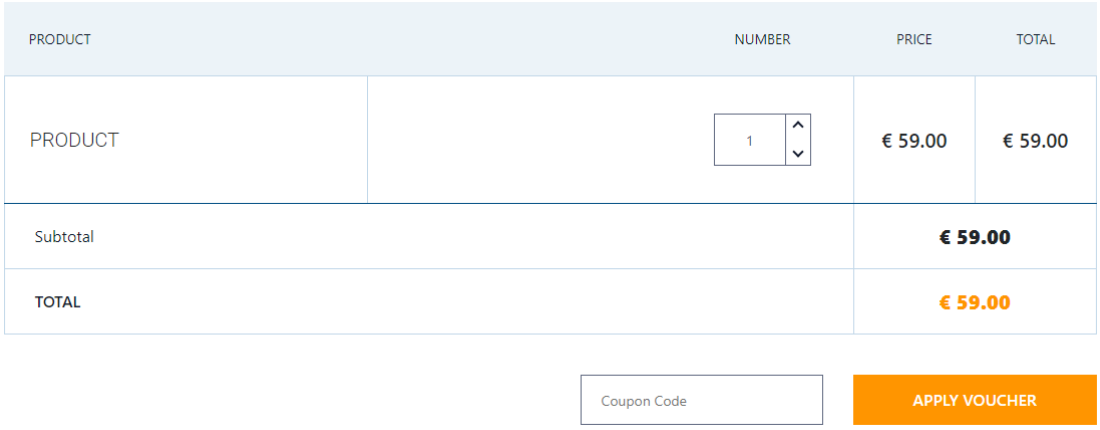

Рисунок 3.10 – Створення замовлення на тестовий продукт

Далі потрібно перейти до форми, яка містить обов’язкові для заповнення поля для вводу чутливих даних користувача щодо оформлення обраного товару. Поточний чек буде оплачено за допомогою сервісу Paypal, а саме тестової банківської картки. Інтерфейс оформлення замовлення показано на рисунку 3.11.

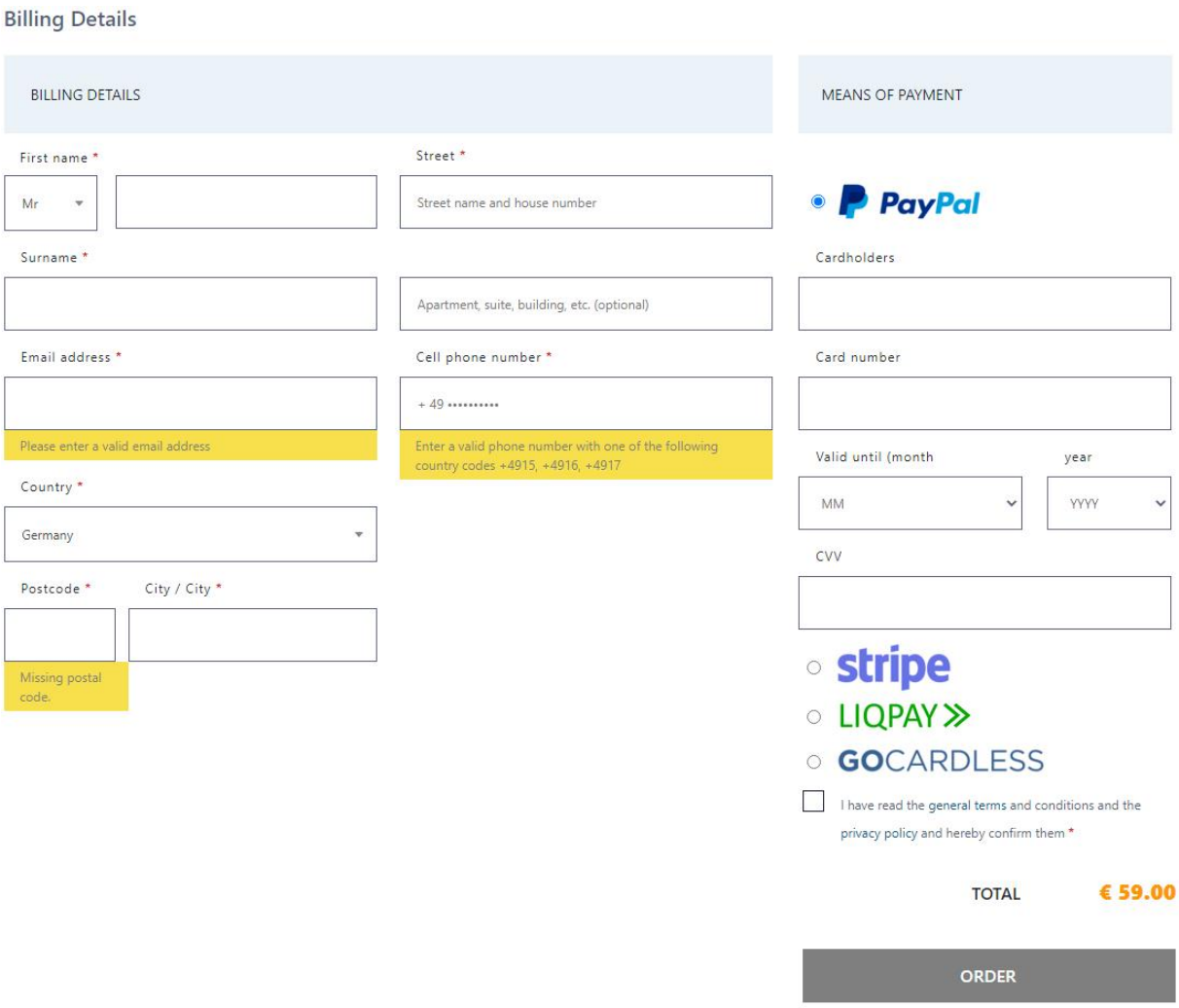

Рисунок 3.11 – Інтерфейс оформлення замовлення

Після заповнення кожного з вищенаведених полів, включаючи дані картки Mastercard переходимо до оплати за допомогою платіжного сервісу Paypal. Після перенесення користувача до нативної форми оплати, там можна побачити грошову суму переведену у долари, поточний курс валют та останні чотири цифри обраної картки. Тут покупець має останній шанс пересвідчитись у доцільності поточної покупки та схвалити її проведення за допомогою клавіші Pay now. Інтерфейс Paypal на цьому етапі транзакції між користувачем та Ecommerce системою наведено на рисунку 3.12.

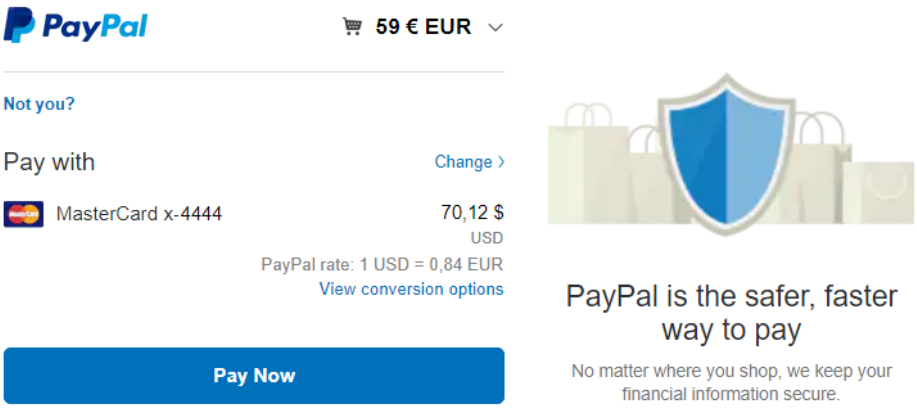

Рисунок 3.12 – Інтерфейс форми оплати Paypal

Після схвалення покупки обраного товару відбувається грошовий обмін між балансом картки користувача та рахунком PayPal інтернет-магазину. Для того щоб переконатись у тому що він пройшов успішно потрібно зайти у тестовий бізнес-аккаунт прив’язаний до Ecommerce системи та пересвідчитись у зміні грошового балансу, що відповідає сумі, яку оплатив користувач. Приборна панель рахунку PayPal інтернет-магазину після проведення тестового платежу зображена на рисунку 3.13.

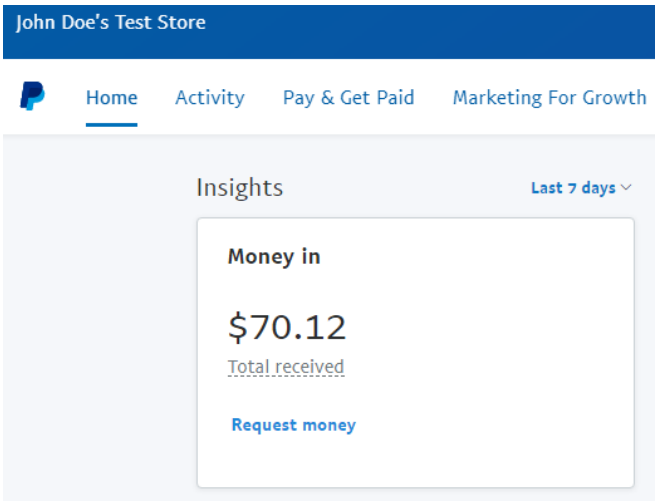

Рисунок 3.13 – Приборна панель Paypal тестового магазину

Крім грошового обміну який відбувся між двома вищезазначеними фінансовими сутностями, користувача перенаправили назад до конкретного

пункту призначення у Ecommerce системі, заздалегідь зазначеного у конфігураційному файлі роутів Laravel. Фінальний етап-сторінку з подякою користувачеві за покупку на сайті зазначено на рисунку 3.14.

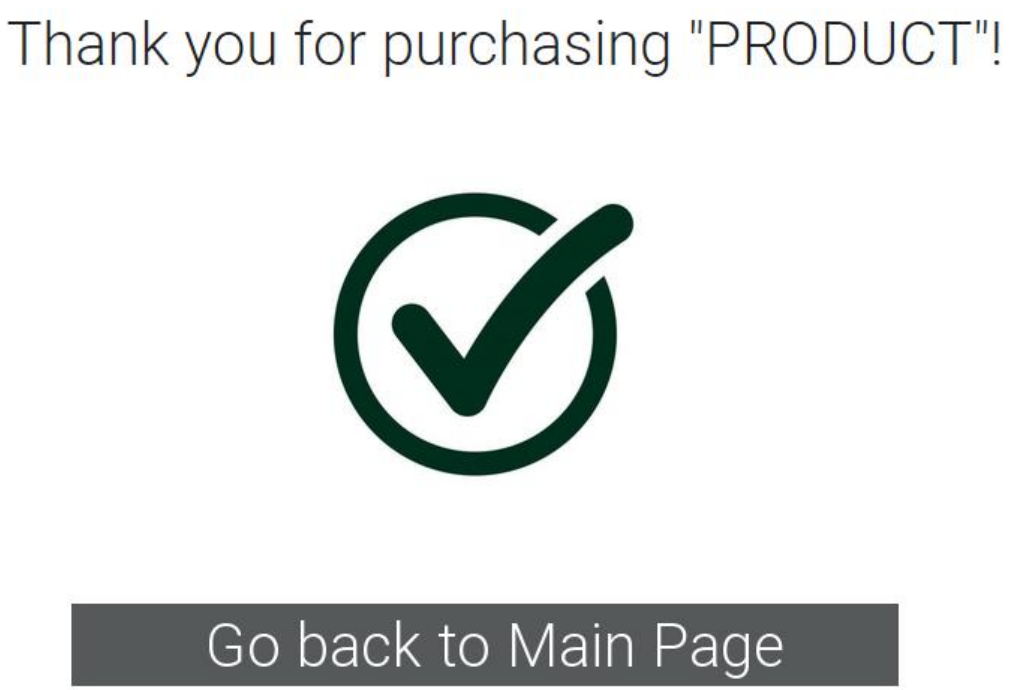

Рисунок 3.14 – Подяка користувачеві за покупку на сайті

ВИСНОВКИ

Беручи за основу матеріал, отриманий під час дослідження процесу розробки платіжного компоненту, можна зробити наступні висновки:

а) існує три основні категорії платіжних провайдерів: дистриб'ютори, збирачі та агрегатори, на основі комбінації яких будується інтеграція товарно-грошового функціоналу у сучасну Ecommerce систему;

б) найбільш поширеними, впізнаваними та використовуваними серед інтернет-покупців представниками категорій платіжних провайдерів є Paypal, Stripe, Liqpay та GoCardless, кожен з яких має детальну документацію по їх імплементації у веб-додаток;

в) платіжний функціонал реалізується за допомогою обраного в результаті аналізу фреймворку Laravel та інтерфейсів прикладного програмування кожного з обраних провайдерів, які використовують дані користувача, поточного чеку, методу оплати та посилання на ресурс Ecommerce системи.

На основі вищевказаних відомостей, аналізу технологій та переліку діаграм UML було розроблено платіжний компонент, який реалізує єдиний інтерфейс для імплементації будь-якого платіжного провайдеру у Ecommerce систему. Для цього було реалізовано шлюзи, що відповідають кожному з наведених провайдерів які втілюють основні функції, такі як оплата, повернення коштів, обробка повідомлень та генерація форми для вводу чутливих даних; сутність, яка міститиме усі дані про поточну покупку та компонент, який є реалізує інтерфейс для управління вищезазначеними елементами застосунку. Результатом виконання є компонент, що інтегрується у вікні схвалення оплати поточного товару у Ecommerce системі.

ПЕРЕЛІК ПОСИЛАНЬ

1. EcommerceWiki. What is a Payment Service Provider? Online Payments. URL : https://www.ecommercewiki.org/index.php/topics/37/online-payments/articles/439/what-is-a-payment-service-provider (дата звернення:14.08.2020).

2. Netguru. 10 Online Payment Service Providers You Need To Know About. Netguru blog. URL : https://www.netguru.com/blog/10-online-payment-service-providers (дата звернення: 19.08.2020).

3. English Wikipedia. PayPal. Wikipedia Article. URL : https://en.wikipedia.org/wiki/PayPal (дата звернення: 24.08.2020).

4. Paypal. PayPal Reports Fourth Quarter and Full Year 2019 Results. Paypal Story. URL : https://www.paypal.com/stories/us/paypal-reports-fourth-quarter-and-full-year-2019-results (дата звернення: 26.08.2020).

5. Paypal. Accept Payments through Paypal. Paypal Article. URL : https://www.paypal.com/ru/webapps/mpp/merchant (дата звернення: 28.08.2020).

6. Paypal. PayPal Merchant Fees. Paypal Article. URL : https://www.paypal.com/us/webapps/mpp/merchant-fees (дата звернення: 02.09.2020).

7. English Wikipedia. Stripe (company). Wikipedia Article. URL : https://en.wikipedia.org/wiki/Stripe_(company) (дата звернення: 09.09.2020).

8. Fundera. Stripe Review: Should You Use This Payment Processor for Your Business? Fundera blog. URL : https://www.fundera.com/blog/stripe-reviews (дата звернення: 11.09.2020).

9. Laravel. Laravel Cashier. Laravel Documentation. URL : https://laravel.com/docs/8.x/billing (дата звернення: 15.09.2020).

10. Ukrainian Wikipedia. LiqPay. Wikipedia Article. URL : https://uk.wikipedia.org/wiki/LiqPay (дата звернення: 18.09.2020).

11. Liqpay. Frequently asked questions. Liqpay Article. URL : https://www.liqpay.ua/ru/faq (дата звернення: 22.09.2020).

12. Kinderpocock. How can I explain GoCardless to my customers? Kinderpocock blog. URL : https://www.kinderpocock.co.uk/how-can-i-explain-gocardless-to-my-customers/ (дата звернення: 24.09.2020).

13. GoCardless. How to accept payments online. GoCardless Guides. URL : https://gocardless.com/guides/posts/online-payment-options/ (дата звернення: 28.09.2020).

14. GoCardless. A guide to merchant accounts for small businesses. GoCardless Guides. URL : https://gocardless.com/guides/posts/merchant-account-for-small-businesses/ (дата звернення: 29.09.2020).

15. GoCardless. Pricing. GoCardless Guides. URL : https://gocardless.com/pricing/ (дата звернення: 30.09.2020).

16. Stauffer M. Laravel: Up and Running: A Framework for Building Modern PHP Apps / Sebastopol : O'Reilly Media, 2016. 795 p. ISBN 1491936088. ISBN 978-1491936085.

17. Potencier F. Symfony 5: The Fast Track / Paris : Symfony SAS leanpub, 2020. 342 p. ISBN-13 978-2-918390-37-4.

18. Portwood C. R. II. Mastering Yii / Birmingham : Packt Publishing, 2016. 380 p. ISBN-10: 1785882422, ISBN-13: 978-1-78588-242-5.

19. Balderas M. PayPal APIs: Up and Running: A Developer's Guide / Sebastopol : O'Reilly, 2011. 120 p. ISBN 1449396127.

20. Liqpay. Checkout. Liqpay Documentation URL : https://www.liqpay.ua/documentation/en/api/aquiring/checkout/doc (дата звернення: 06.10.2020).

21. GoCardless Developers. Taking your first payment. Getting started. URL : https://developer.gocardless.com/getting-started/api/taking-your-first-payment/ (дата звернення: 08.10.2020).

ДОДАТОК А

Діаграма класів

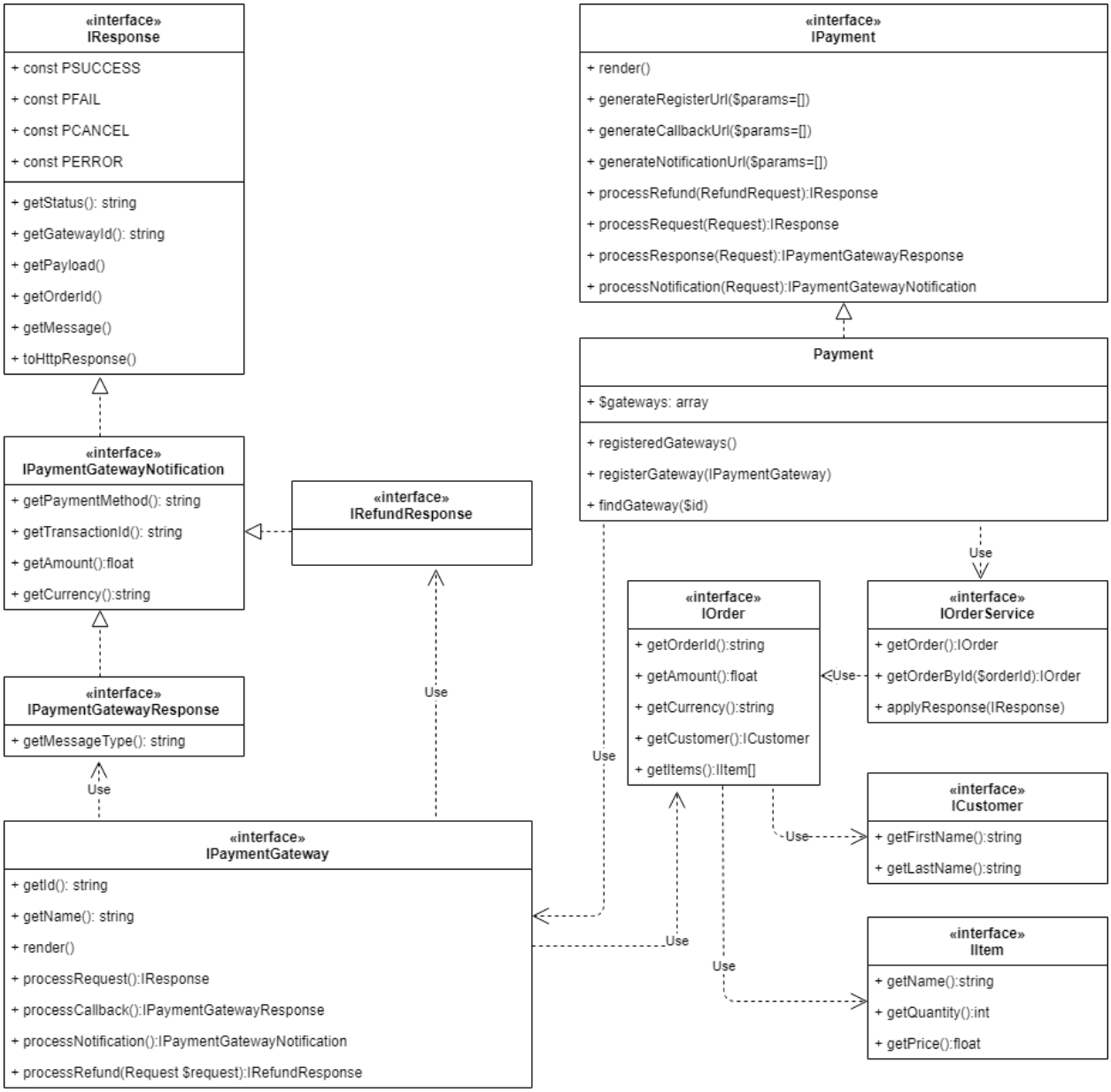

Рисунок А.1 – Діаграма класів

ДОДАТОК Б

Код шлюзу EmptyPaymentGateway

```
<?php


namespace EmptyPayment;


use Illuminate\Http\Request;
use Payment\Facades\Payment;
use Payment\IOrderService;
use Payment\IPaymentGateway;
use Payment\IPaymentGatewayResponse;
use Payment\IRefundResponse;
use Payment\IResponse;
use Payment\Responses\Builder;

class EmptyPaymentGateway implements IPaymentGateway
{
    const PAYMENT_METHOD = "empty";

    /**
     * @var IOrderService
     */
    public $orderService;
    /**
     * @var Request
     */
    public $request;
```

```
/**
 * CreditCardGateway constructor.
 * @param IOrderService $orderService
 * @param Request $request
 */
public function __construct(IOrderService $orderService, Request $request)
{
    $this->orderService = $orderService;
    $this->request = $request;
}

/**
 * @inheritDoc
 */
public function getId(): string
{
    return "empty_payment";
}

/**
 * @inheritDoc
 */
public function getName(): string
{
    return __("EmptyPayment::messages.name");
}

/**
 * @inheritDoc
 */
public function render()
```

```
{
    return view("EmptyPayment::form", [
        'name' => self::getName(),
        'gateway' => self::getId(),
        'paymentMethod' => self::PAYMENT_METHOD,
    ])->render();
}

/**
 * @inheritDoc
 */
public function processCallback(): IPaymentGatewayResponse
{
    $order = $this->orderService->getOrder();
    $builder = Builder::create()
        ->amount($order->getAmount())
        ->gatewayId($this->getId())
        ->orderId($order->getId())
        ->currency($order->getCurrency())
        ->paymentMethod($this->getId())
        ->payload($this->request->all())
        ->transactionId("empty-payment-" . md5(now()));
    if ($order->getAmount() > 0) {
        $builder
            ->status(IResponse::PGR_ERROR)
            ->message(__("EmptyPayment::messages.emptyAmountRequired"))
            ->messageType(IResponse::PGR_ERROR);
    } else {
        $builder
            ->status(IResponse::PGR_SUCCESS)
            ->message(__("EmptyPayment::messages.successPayment"))
            ->messageType(IResponse::PGR_SUCCESS);
```

```
        $this->orderService->applyResponse($builder->getCallbackResponse());
    }

    return $builder->getCallbackResponse();
}

/**
 * @inheritDoc
 */
public function processRequest(): IResponse
{
    Payment::forgetCheckoutMessage();
    Payment::forgetGetawayResponse();
    $order = $this->orderService->getOrder();
    $builder = Builder::create()
        ->response(response()->redirectTo(Payment::generateCallbackUrl([
            'gateway' => $this->getId()
        ])))
        ->amount($order->getAmount())
        ->gatewayId($this->getId())
        ->orderId($order->getId())
        ->currency($order->getCurrency())
        ->payload($this->request->all());
    if ($order->getAmount() > 0) {
        $message = __("EmptyPayment::messages.emptyAmountRequired");
        Payment::setCheckoutMessage($message, IResponse::PGR_ERROR);
        $builder
            ->status(IResponse::PGR_ERROR)
            ->message($message)
            ->messageType(IResponse::PGR_ERROR)
            ->response(redirect(route("checkout") . "#notifications-details"));
```

```
        }
        return $builder->getResponse();
    }

    /**
     * @inheritDoc
     */
    public function processNotification()
    {
        throw new \BadMethodCallException("Notification is not supported!");
    }

    /**
     * @inheritDoc
     */
    public function processRefund(): IRefundResponse
    {
        throw new \BadMethodCallException("Refund is not supported!");
    }
}
```