Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Шевченка Іллі Віталійовича*
	(ПІБ)
академічної групи	*122М-22-2*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart*

_____І.В. Шевченко

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*доц.Спірінцев В.В.*			
економічний				

Рецензент				

Нормоконтролер	*проф. Лактіонов І.С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-2* *Шевченко Іллі Віталійовичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес практичного застосування мобільного додатку для підвищення продуктивності роботи бізнес моделі Ecommerce рішень (Opencart).

Предмет досліджень – методи і моделі взаємодії мобільного застосунку з RESTful API в контексті використання системи управління контентом Opencart. Дослідження спрямоване на вивчення оптимальних стратегій інтеграції, забезпечення ефективної взаємодії та оптимізації робочих процесів, пов'язаних із використанням мобільних застосунків та API у сфері електронної комерції.

Мета НДР – удосконалення ефективності взаємодії мобільного застосунку з RESTful API у рамках CMS системи Opencart, за рахунок розділення запитів до API та переписування запитів до бази даних.

Вихідні дані для проведення роботи – аналіз та систематизація теоретичних знань з області взаємодії мобільних застосунків з RESTful API та функціоналу CMS системи Opencart.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що вперше запропоновано систему взаємодії мобільного додатку з RESTful API при використанні системи управління контентом Opencart. Розроблений підхід базується на використанні

декларативних засад фреймворку Flutter для побудови інтерфейсу мобільного додатку та ефективного взаємодії з API Opencart.

Практична цінність полягає в розробці та впровадженні мобільного застосунку, який оптимізує взаємодію з RESTful API при роботі з CMS системою Opencart. Одним із ключових аспектів є створення високодинамічного мобільного додатку, який дозволить ефективно взаємодіяти з API Opencart та забезпечить користувачам швидку та зручну навігацію.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути представлені у формі функціонального та ефективного мобільного додатку, який ілюструє оптимальну взаємодію з RESTful API при інтеграції з CMS системою Opencart. Розроблений програмний продукт буде надавати користувачам зручний та інтуїтивно зрозумілий інтерфейс для роботи з онлайн-магазином, використовуючи сучасні технології та підходи.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2023-30.09.2023
Визначення основних вимог та функціональності мобільного застосунку, який буде інтегровано з CMS Opencart	01.10.2023-31.10.2023
Проведення тестування прототипу для визначення ефективності та стабільності взаємодії з API.	01.11.2023-06.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи в області дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart може бути значний і виявитися позитивним в контексті оптимізації бізнес-процесів. Очікується зменшення кількості розробників, задіяних у створенні та підтримці мобільних додатків, завдяки використанню єдиної бази коду та декларативного підходу до побудови інтерфейсу через фреймворк Flutter.

Соціальний ефект від реалізації результатів роботи в галузі дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart очікується позитивним завдяки поліпшенню процесів проектування та розробки мобільного додатку.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Спірінцев В.В.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Шевченко І.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *09.10.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *07.12.2023*

РЕФЕРАТ

Пояснювальна записка: 111 стор., 37 рис., 4 додатки, 37 джерело.

Об’єкт дослідження: процес практичного застосування мобільного додатку для підвищення продуктивності роботи бізнес моделі Ecommerce рішень (Opencart).

Предмет дослідження: методи і моделі взаємодії мобільного застосунку з RESTful API в контексті використання системи управління контентом Opencart. Дослідження спрямоване на вивчення оптимальних стратегій інтеграції, забезпечення ефективної взаємодії та оптимізації робочих процесів, пов'язаних із використанням мобільних застосунків та API у сфері електронної комерції.

Мета роботи: удосконалення ефективності взаємодії мобільного застосунку з RESTful API у рамках CMS системи Opencart, за рахунок розділення запитів до API та переписування запитів до бази даних.

Методи дослідження. Для вирішення поставлених задач використані методи: використання засобів крос-платформного програмування, декларативного програмування та об’єктно-орієнтованого програмування.

Новизна отриманих результатів визначається тим, що вперше запропоновано систему взаємодії мобільного додатку з RESTful API при використанні системи управління контентом Opencart. Розроблений підхід базується на використанні декларативних засад фреймворку Flutter для побудови інтерфейсу мобільного додатку та ефективного взаємодії з API Opencart.

Практична цінність результатів полягає в розробці та впровадженні мобільного застосунку, який оптимізує взаємодію з RESTful API при роботі з CMS системою Opencart. Одним із ключових аспектів є створення високодинамічного мобільного додатку, який дозволить ефективно взаємодіяти з API Opencart та забезпечить користувачам швидку та зручну навігацію.

Область застосування. Розроблена інформаційна система може бути ефективно використана у сфері електронної комерції, де її інтеграція з CMS системою Opencart сприятиме оптимізації та автоматизації процесів управління контентом та обслуговуванням мобільного додатку.

Значення роботи та висновки. Вдосконалений підхід до взаємодії мобільного застосунку з RESTful API в контексті CMS системи Opencart дозволяє покращити ефективність та надає нові можливості для управління вмістом та обслуговуванням мобільних додатків.

Прогнози щодо розвитку досліджень. Майбутні дослідження можуть фокусуватися на вдосконаленні взаємодії мобільних додатків з RESTful API в операційному середовищі Opencart, з метою додавання нових функцій, покращення процесів управління та підтримки.

Список ключових слів: Flutter, Dart, RESTful API, CMS Opencart, плагін, адмін-панель, івент, MVC, MySQLi, Apache, MVVM, PHP, XML.

ABSTRACT

Explanatory note: 111 pages, 37 figures, 4 appendices, 37 sources.

The object of the study: the process of practical application of a mobile application to increase the productivity of the business model of Ecommerce solutions (Opencart).

Research subject: methods and models of mobile application interaction with RESTful API in the context of using the Opencart content management system. The study is aimed at studying optimal integration strategies, ensuring effective interaction and optimizing work processes related to the use of mobile applications and APIs in the field of e-commerce.

The goal of the work: improving the efficiency of the interaction of the mobile application with the RESTful API within the Opencart CMS system, due to the separation of requests to the API and the rewriting of requests to the database.

Research methods. To solve the problems, the following methods are used: the use of cross-platform programming, declarative programming, and object-oriented programming.

The novelty of the obtained results is determined by the fact that for the first time the development of a mobile application interaction system with a RESTful API when using the Opencart content management system is proposed. The developed approach is based on the use of the declarative principles of the Flutter framework for building the mobile application interface and effective interaction with the Opencart API.

The practical value of the results lies in the development and implementation of a mobile application that optimizes interaction with the RESTful API when working with the Opencart CMS system. One of the key aspects is the creation of a highly dynamic mobile application that will allow efficient interaction with the Opencart API and provide users with quick and convenient navigation.

Field of application. The developed information system can be effectively used in the field of e-commerce, where its integration with the Opencart CMS system will contribute to the optimization and automation of content management processes and mobile application maintenance.

Value of work and conclusions. An improved approach to the interaction of a mobile application with a RESTful API in the context of the Opencart CMS allows for improved efficiency and provides new opportunities for content management and maintenance of mobile applications.

Forecasts regarding the development of research. Future research can focus on improving the interaction of mobile applications with the RESTful API in the Opencart operating environment, with the aim of adding new features, improving management and support processes.

List of keywords: Flutter, Dart, RESTful API, CMS Opencart, plugin, admin panel, event, MVC, MySQLi, Apache, MVVM, PHP, XML.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CMS Opencart – сиcетма управління контентом;

Plugin – модуль або розширення;

Admin Panel – веб-інтерфейс для управління;

Event – подія чи сповіщення;

MVC – Model-View-Controller;

MySQLi – розширення для взаємодії з MySQL;

Apache – веб-сервер;

MVVM – (Model-View-ViewModel);

PHP – скриптова мова програмування;

XML – eXtensible Markup Language.

ЗМІСТ

ВСТУП

Актуальність теми. В сучасному світі мобільні додатки стали необхідним елементом повсякденного життя, що використовується для різних цілей, від соціальних мереж і онлайн-покупок до ведення бізнесу. Один із важливих аспектів успішної розробки мобільних додатків - це взаємодія з серверами через RESTful API. CMS система Opencart широко використовується для онлайн-торгівлі, і інтеграція мобільного додатка з нею за допомогою RESTful API має великий потенціал для розширення бізнесу та полегшення користувачам процесу покупок.

Мета і задачі дослідження. Метою даного дослідження є аналіз, розробка та оцінка ефективності взаємодії мобільного додатку з RESTful API у роботі з CMS системою Opencart. Для досягнення цієї мети ставляться такі задачі:

1. Провести огляд і аналіз існуючих методів взаємодії мобільних додатків з RESTful API.

2. Дослідити особливості CMS системи Opencart і її можливості щодо інтеграції з мобільними додатками.

3. Розробити мобільний додаток, який використовує RESTful API для взаємодії з CMS системою Opencart.

4. Провести тестування та оцінку продуктивності розробленого додатку.

Об’єктом досліджень є процес практичного застосування мобільного додатку для підвищення продуктивності роботи бізнес моделі Ecommerce рішень (Opencart).

Предметом досліджень є методи та технології, використовувані для розробки та оптимізації мобільного додатку, яке взаємодіє з CMS системою Opencart через RESTful API.

Методи дослідження. Для досягнення поставлених цілей та вирішення задач дослідження використовуватимуться наступні методи:

– Функціональний аналіз: Розбір функцій мобільного застосунку та RESTful API Opencart. Визначення ключових взаємодій та інтеграцій між

системними компонентами. Розробка функціональних моделей для кращого розуміння взаємодії.

— Теорія баз даних: Аналіз структури та організації бази даних Opencart. Визначення оптимальних методів взаємодії мобільного застосунку з базою даних через RESTful API.Оцінка ефективності використання бази даних у відповідь на запити мобільного застосунку.

Наукова новизна роботи визначається тим, що вперше запропоновано систему взаємодії мобільного додатку з RESTful API при використанні системи управління контентом Opencart. Розроблений підхід базується на використанні декларативних засад фреймворку Flutter для побудови інтерфейсу мобільного додатку та ефективного взаємодії з API Opencart. Результати дослідження дозволять визначити оптимальні методи та технології для розробки мобільних додатків, які використовують Opencart як основну платформу для онлайн-торгівлі.

Практичне значення. Результати дослідження можуть бути корисними для розробників мобільних додатків, які працюють з CMS системою Opencart. Вони нададуть змогу покращити взаємодію між мобільними додатками та Opencart, що сприятиме підвищенню продуктивності та зручності користувачів під час онлайн-покупок.

Особистий внесок автора. Автор проведе всі етапи дослідження, включаючи аналіз літературних джерел, розробку мобільного додатку, тестування та аналіз результатів. Автор також зробить висновки та рекомендації щодо використання RESTful API у роботі з CMS системою Opencart для мобільних додатків.

Структура та обсяг кваліфікаційної роботи. Відповідно до мети, задач і предмета дослідження, кваліфікаційна робота складається з реферату, вступу, трьох основних розділів і висновків, списку використаних джерел та 4 додатків. Загальний обсяг роботи містить 111 сторінок друкованого тексту, із них основна частина - 15 сторінок, спеціальна – 34 сторінок з 14 рис., списку використаних джерел з 37 найменуванням на 3 сторінках, 4 додатках на 30 сторінках.

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз сучасного стану взаємодії мобільних додатків з CMS системами

Аналіз сучасного стану взаємодії мобільних додатків з CMS системами є критично важливою задачею в сучасному цифровому середовищі. Зростання використання мобільних пристроїв та CMS систем, таких як Opencart, призвело до необхідності глибокого розуміння, як ці дві платформи можуть ефективно взаємодіяти між собою.

Аналізуються різні підходи до інтеграції та обміну даними між мобільними додатками та CMS системами, і особлива увага приділяється питанням безпеки, оптимізації та масштабованості. Важливі виклики та перспективи розвитку цієї взаємодії розглядаються з точки зору розробників мобільних додатків.

Аналіз сучасного стану взаємодії мобільних додатків з CMS системами має велике практичне значення. Він допомагає вдосконалювати обслуговування клієнтів, підвищувати продуктивність бізнесу та зміцнювати конкурентоспроможність. Актуальність цього аналізу постійно зростає в умовах швидкозмінюваного інформаційного середовища, що вимагає подальших досліджень та розробки ефективних рішень у цій області.

Роль CMS (Content Management System) систем в сучасному бізнесі є надзвичайно важливою і стратегічною. CMS системи є інструментами, які дозволяють організаціям створювати, редагувати, керувати та публікувати вміст на своєму веб-сайті без необхідності глибокого знання програмування або веб-розробки [1]. Роль CMS систем в бізнесі може бути розкрита наступними ключовими аспектами:

Управління контентом: CMS системи дозволяють бізнесам з легкістю створювати, редагувати та оновлювати контент на своєму веб-сайті. Це включає в себе публікацію новин, статей, товарів, послуг, зображень, відео та іншого важливого вмісту.

Ефективний електронний бізнес: CMS системи, такі як Opencart, надають можливість створювати онлайн-магазини та електронні платформи для продажу товарів та послуг. Це спрощує процес створення та управління електронним бізнесом.

Персоналізація контенту: CMS системи дозволяють адаптувати контент для різних категорій аудиторії. Це допомагає створити більш персоналізовані та релевантні веб-сайти для користувачів, що підвищує залученість та конверсію.

SEO оптимізація: Багато CMS систем включають в себе інструменти для пошукової оптимізації (SEO), які допомагають веб-сайтам підвищувати свій рейтинг у пошукових системах. Це робить бізнес більш видимим у мережі.

Аналітика та звіти: CMS системи надають можливість відстежувати та аналізувати веб-аналітику, що дозволяє бізнесам робити обгрунтовані рішення на основі даних.

Мультиплатформенність: Деякі CMS системи дозволяють оптимізувати контент для різних пристроїв і платформ, забезпечуючи гладку роботу на комп'ютерах, планшетах і смартфонах [2].

Загалом, CMS системи відіграють критичну роль у спрощенні управління контентом, покращенні електронного бізнесу та забезпеченні взаємодії бізнесів з їх аудиторією в сучасному цифровому світі. Вони є невід'ємною частиною успішного онлайн-присутності та бізнес-розвитку.

1.1.1. Визначення основних викликів та проблем, пов'язаних із взаємодією мобільних застосунків та Opencart через RESTful API

Один з основних викликів полягає у забезпеченні безпеки та конфіденційності даних під час передачі інформації між мобільним додатком і системою Opencart через RESTful API. Наприклад, при передачі особистих даних користувачів (наприклад, ім'я, адреса, дані кредитних карт) через API, існує ризик перехоплення цих даних зловмисниками. Для вирішення цієї проблеми, необхідно використовувати шифрування даних та інші заходи безпеки.

Ще однією проблемою може бути нестабільність чи недоступність API системи Opencart. Це може призвести до незручностей для користувачів мобільного додатку, оскільки вони не зможуть здійснювати покупки чи звертатися до інших функцій системи в разі відмови API. Наприклад, якщо API Opencart не працює коректно, користувачі можуть зіткнутися з проблемами при додаванні товарів до кошика або оплаті замовлень.

Прикладом ще однієї проблеми є несумісність форматів даних між мобільним додатком і системою Opencart через RESTful API. Наприклад, мобільний додаток може відправляти дані у JSON форматі, а система Opencart очікувати XML. Це може призвести до помилок у передачі даних і непрацездатності взаємодії між додатком і системою.

Загалом, взаємодія мобільних застосунків та Opencart через RESTful API може стикатися з різними викликами та проблемами, які потребують уважного аналізу та розробки ефективних рішень для їх вирішення.

1.2. Методи та технології взаємодії мобільних застосунків з RESTful API

Взаємодія між мобільними застосунками та RESTful API в сучасному програмуванні є надзвичайно важливою. REST (Representational State Transfer) є архітектурним стилем, який надає можливість взаємодії між різними компонентами системи через стандартні HTTP-запити. Мобільні застосунки зазвичай використовують RESTful API для звернення до серверів, отримання даних, відправлення оновлень і здійснення інших дій, необхідних для їх роботи. Розуміння методів та технологій взаємодії є важливим аспектом розробки мобільних застосунків, оскільки це впливає на продуктивність, безпеку та користувацький досвід [3].

Основні методи та технології взаємодії мобільних застосунків з RESTful API включають в себе [4]:

— HTTP-запити та методи: Мобільні застосунки використовують HTTP-запити (GET, POST, PUT, DELETE і т.д.) для взаємодії з RESTful API. Наприклад, GET-запит використовується для отримання ресурсів, POST - для створення нових ресурсів, PUT - для оновлення існуючих ресурсів, а DELETE - для видалення;

— формати обміну даними: Дані, які передаються між мобільним застосунком і RESTful API, зазвичай форматуються в стандартних форматах, таких як JSON (JavaScript Object Notation) або XML (Extensible Markup Language). JSON є більш популярним форматом через його легкість в читанні та обробці;

— аутентифікація та авторизація: Для забезпечення безпеки та конфіденційності даних між мобільним застосунком і RESTful API використовуються методи аутентифікації та авторизації. Це може включати в себе використання токенів, API-ключів або базової аутентифікації;

— робота зі статусами та помилками: Мобільні застосунки повинні правильно обробляти HTTP-статуси відповідей від RESTful API, щоб реагувати на успішні або невдали дії. Наприклад, статус 200 означає успішну операцію, а статус 404 вказує на відсутність ресурсу;

— кешування даних: Мобільні застосунки можуть використовувати кешування для збереження копій раніше отриманих даних, щоб покращити продуктивність та зменшити навантаження на сервер;

— синхронізація даних: У разі втрати зв'язку з сервером мобільний застосунок повинен мати механізм синхронізації даних при відновленні зв'язку.

— тестування та налагодження: Розробники мобільних застосунків повинні використовувати інструменти для тестування взаємодії з RESTful API та налагодження коду для вирішення можливих проблем;

— документація API: Розробники повинні мати доступ до документації RESTful API, яка описує доступні ресурси, методи та параметри.

Продуктивна і безпечна взаємодія між мобільними застосунками та CMS системами, як Opencart, залежить від глибокого розуміння цих методів та

технологій. Особливу увагу слід приділити забезпеченню безпеки даних, ефективності передачі та правильному обробці помилок.

1.2.1. Аналіз різних методів та технологій, що застосовуються при інтеграції мобільних додатків з RESTful API

Розглядаючи різні методи та технології інтеграції мобільних додатків з RESTful API, важливо враховувати, що існують численні підходи до цього завдання. Вибір конкретного методу чи технології може суттєво вплинути на продуктивність, надійність та зручність взаємодії між мобільними додатками та CMS системами.

Один із можливих методів - це використання бібліотек та фреймворків, які спрощують інтеграцію RESTful API в мобільних додатках. Наприклад, використання Retrofit для Android або Alamofire для iOS може значно полегшити роботу з HTTP-запитами та дозволити зручно обробляти дані, отримані з CMS системи.

Інший метод - це використання архітектурних підходів, таких як Model-View-ViewModel (MVVM) чи Model-View-Controller (MVC), для структурування мобільного додатка та розділення логіки взаємодії з RESTful API від користувацького інтерфейсу. Це дозволяє зберігати код чистим і підтримуваним, а також спрощує тестування.

Технології, такі як OAuth або JWT (JSON Web Tokens), можуть бути використані для забезпечення безпеки під час аутентифікації та авторизації користувачів в мобільних додатках, що взаємодіють з CMS системами.

Важливим елементом є також вибір формату даних для обміну інформацією між мобільним додатком і CMS системою. JSON часто є стандартним форматом для передачі даних через RESTful API, оскільки його легко розуміти і обробляти як на стороні сервера, так і на стороні мобільного додатка.

Залежно від конкретних потреб та особливостей проекту, може бути використано інші методи та технології, такі як GraphQL, WebSockets або кешування даних на стороні клієнта. Важливо обирати ті рішення, які найкращим чином відповідають конкретним вимогам і завданням мобільного додатка та CMS системи.

Приклади які бувають методи для розробки інтеграції:

BaaS (Backend as a Service): Це підходить, коли розробники хочуть спростити розробку серверної частини свого додатка. BaaS-платформи, такі як Firebase або AWS Amplify, надають готовий серверний функціонал, включаючи базу даних, аутентифікацію та сховище файлів.

SDK (Software Development Kit): Багато популярних мобільних платформ, таких як Android і iOS, надають SDK для спрощення взаємодії з RESTful API. Це дозволяє розробникам використовувати готові бібліотеки та інструменти для відправки запитів і обробки відповідей.

OAuth і авторизація: Один з ключових аспектів безпеки та авторизації в мобільних додатках - це використання протоколу OAuth для забезпечення доступу до API. Цей протокол дозволяє користувачам надавати дозвіл на доступ до своїх облікових записів і ресурсів.

Swagger/OpenAPI: Swagger (тепер відомий як OpenAPI) - це інструмент для створення документації API та генерації клієнтських бібліотек. Використання цього інструмента допомагає розробникам зрозуміти, як працює API та які доступні ресурси.

WebSockets: У деяких сценаріях взаємодії в режимі реального часу (наприклад, чат в мобільному додатку), використання технології WebSockets може бути важливим. Вона дозволяє бідбивати зв'язок між клієнтом та сервером для негайного обміну даними.

REST-клієнти: Деякі розробники створюють REST-клієнти для спрощення взаємодії з API. Ці клієнти надають зручний спосіб виконувати запити та отримувати дані, що відповідають структурі об'єктів у програмі.

Мікросервісна архітектура: У випадках, коли мобільний додаток взаємодіє з декількома API або сервісами, може виникнути необхідність використовувати мікросервісну архітектуру [6]. Це дозволяє розділити функціонал на невеликі сервіси, які легко масштабувати та оновлювати окремо.

Тестування API: Для забезпечення надійності та відповідності специфікаціям RESTful API використовуються інструменти для автоматизованого тестування. Це допомагає виявляти помилки та недоліки під час розробки та інтеграції.

Розробники мобільних додатків повинні ретельно вибирати методи та технології для взаємодії з RESTful API, враховуючи вимоги щодо безпеки, продуктивності та функціональності свого додатка. Коректний вибір інструментів і стратегій дозволить створити ефективний та надійний мобільний застосунок, який відповідає потребам користувачів.

1.2.2. Оцінка ефективності вже існуючих підходів до взаємодії мобільних застосунків із CMS системою Opencart

Аналіз ефективності існуючих методів взаємодії мобільних додатків з CMS системами, зокрема з системою Opencart через RESTful API, є важливим етапом дослідження. На сучасному ринку існує багато підходів до цієї взаємодії, і вони мають свої переваги та обмеження.

Проведення оцінки ефективності включає аналіз різних аспектів. Важливо визначити, які підходи найбільш підходять для конкретних завдань і цілей. Перш за все, ми детально досліджуємо продуктивність. Це включає вимірювання часу завантаження сторінок мобільного додатка, часу обробки запитів до CMS системи та загальної продуктивності системи в цілому.

Другий важливий аспект - це надійність. Ми аналізуємо стійкість підходів до помилок та завад, які можуть виникнути під час взаємодії, і визначаємо можливості автоматичного відновлення роботи системи в разі виникнення проблем.

Безпека - ще один важливий аспект. Ми вивчаємо рівень захищеності підходів і переконуємося, що дані користувачів залишаються захищеними від несанкціонованого доступу.

Зручність використання - це також ключовий критерій. Ми аналізуємо, наскільки зручно та інтуїтивно користуватися мобільним додатком при взаємодії з CMS системою, особливо якщо остання має складну структуру.

Масштабованість, ресурсозбереження та вартість розробки і підтримки - це інші фактори, які ми враховуємо при оцінці ефективності. Кожен з цих аспектів має велике значення для визначення оптимального підходу [7].

Варто зауважити, що оцінка ефективності є постійним процесом, оскільки технології та потреби користувачів постійно змінюються. Тому ми продовжуємо вивчати нові методи та технології для досягнення найкращих результатів у взаємодії мобільних додатків з CMS системами. Значимість оцінки ефективності існуючих методів взаємодії мобільних додатків з CMS системами, зокрема з системою Opencart через RESTful API, найбільше проявляється в сучасному бізнес-середовищі. Розвиток інформаційних технологій та зростання конкуренції вимагають від компаній та підприємств максимальної ефективності і оптимізації їхніх операцій.

Мобільні додатки стали необхідною складовою бізнесу, оскільки вони дозволяють підприємствам залучати більше клієнтів, підвищувати лояльність і полегшувати спілкування зі своїми користувачами. Взаємодія з CMS системами через RESTful API надає можливість ефективно керувати контентом, продуктами і послугами в реальному часі.

Але, на жаль, існують численні виклики та проблеми, пов'язані з цією взаємодією. Серед них - забезпечення безпеки, оптимізація продуктивності, підтримка різних платформ і пристроїв, а також стійкість до великої кількості користувачів. Оцінка ефективності допомагає ідентифікувати найкращі рішення та методи для подолання цих викликів.

У реальному бізнес-середовищі, де кожна секунда і кожен клієнт мають значення, зрозуміння, як досягти оптимальної ефективності взаємодії мобільних

додатків з CMS системами, стає ключовим фактором успіху. Такий аналіз допомагає підприємствам не лише заощаджувати ресурси, але й забезпечує їхню конкурентоспроможність та здатність адаптуватися до змін в сучасному бізнес-середовищі.

1.3. Оцінка продуктивності та надійності RESTful API в Opencart

Оцінка продуктивності та надійності RESTful API в системі Opencart є ключовим етапом аналізу взаємодії мобільних додатків з цією CMS-системою. Вона визначає, наскільки ефективно система сприймає і обробляє запити, надсилані з мобільних додатків, і наскільки надійно вона функціонує в різних умовах та обставинах.

Оцінка продуктивності включає в себе дослідження швидкості відповіді системи на запити, завантаження сервера під час інтенсивного використання, а також вимоги до ресурсів, які необхідно виділити для забезпечення нормальної роботи API. Необхідно виміряти метрики продуктивності, такі як час відповіді, пропускна спроможність, кількість запитів на одиницю часу та інші, щоб визначити, наскільки ефективно API взаємодіє з мобільними додатками.

Оцінка надійності передбачає вивчення стійкості системи до помилок і перебоях в мережі. Це включає в себе аналіз механізмів обробки помилок, забезпечення безпеки даних під час передачі через API, а також можливості резервного копіювання та відновлення даних. Важливо враховувати, як система реагує на велику кількість запитів та які заходи приймаються для запобігання відмовам і збоїв.

Оцінка продуктивності та надійності RESTful API в Opencart вимагає проведення серії тестів та досліджень, а також враховування факторів навантаження та обсягу даних. Результати цього аналізу допоможуть визначити, як можна покращити ефективність і надійність API для забезпечення оптимальної взаємодії з мобільними додатками.

1.4. Розробка мобільного застосунку для тестування взаємодії з Opencart

В ході дослідження взаємодії мобільних застосунків з CMS системою Opencart через RESTful API було розроблено спеціальний мобільний додаток для тестування цієї взаємодії. Даний додаток імітував дії реальних користувачів та взаємодію з Opencart через RESTful API, відтворюючи різні сценарії, такі як додавання товарів до кошика, оформлення замовлення, пошук товарів тощо.

Під час розробки та використання цього додатку було проведено аналіз різних показників ефективності та надійності RESTful API в Opencart. Специфічні метрики включали швидкість відповіді сервера, обсяг передачі даних, кількість успішних та неуспішних запитів, а також виявлення можливих помилок та проблем взаємодії.

Проведення тестування на великій кількості користувачів, які одночасно взаємодіють з системою через мобільний додаток, дало змогу визначити, як система масштабується та впорається з навантаженням. Також було виділено питання безпеки та ресурсомісткості під час взаємодії.

Наприклад, в одному з тестових сценаріїв було виявлено помилку при зверненні до RESTful API. Це дозволило команді дослідження детально проаналізувати цю помилку та розробити шляхи її виправлення. Отже, отримана інформація про ефективність, надійність та швидкодію взаємодії мобільних застосунків з CMS системою Opencart через RESTful API стала важливим внеском у вдосконалення цієї взаємодії та забезпечення її оптимальної роботи.

1.4.1. Оцінка ефективності та можливостей розробленого застосунку для тестування

Після розробки мобільного застосунку для тестування взаємодії мобільних додатків з CMS системою Opencart через RESTful API, була проведена оцінка

його ефективності та можливостей. Дана оцінка полягала в аналізі різних аспектів функціональності та результативності застосунку.

Перш за все, важливо було визначити, наскільки добре застосунок відтворює реальні сценарії взаємодії користувачів з CMS системою Opencart через RESTful API. Для цього були розроблені тестові сценарії, що імітували дії користувачів, та були зіставлені результати їх виконання.

Далі, була проведена оцінка швидкості виконання тестових сценаріїв застосунком. Це включало в себе аналіз часу, необхідного для виконання кожного сценарію, а також часу реакції застосунку на запити. Завдяки цій оцінці, було визначено, чи відповідає швидкодія застосунку вимогам до продуктивності [8].

Також важливою була оцінка стійкості застосунку до помилок та надійність його роботи. В ході тестування були відтворені ситуації, коли RESTful API могло відповісти помилкою або іншим неправильним результатом. Важливо було визначити, як застосунок реагує на такі ситуації та чи вміє він коректно обробляти помилки.

В результаті оцінки ефективності та можливостей розробленого застосунку для тестування було виявлено його сильні та слабкі сторони. Ця інформація допомагає команді розробників покращити застосунок та забезпечити його найкращу продуктивність та надійність під час тестування взаємодії мобільних додатків з Opencart через RESTful API.

1.5. Переваги та недоліки крос-платформної розробки мобільних застосунків для взаємодії з OpenCart

Переваги крос-платформної розробки включають [9]:

1. Зменшення затрат: Крос-платформні рішення дозволяють розробляти один код, який працює на різних платформах. Це може значно скоротити витрати на розробку та підтримку;

2. Загальна база коду: Код може бути перевикористаний між платформами, що полегшує підтримку та оновлення;

3. Один інструментарій: Крос-платформні фреймворки та інструменти надають єдиний набір засобів для розробки, що спрощує роботу розробників;

4. Швидкість розробки: Зменшення необхідності в розробці окремих версій для кожної платформи дозволяє розробникам прискорити випуск нових функцій та оновлень;

5. Можливість залучення широкого аудиторії: Крос-платформні застосунки можуть бути доступні для більшої аудиторії користувачів, оскільки вони підтримують кілька платформ.

Однак існують і недоліки:

1. Зменшена продуктивність: Крос-платформні рішення можуть бути менш продуктивними у порівнянні з нативними застосунками, оскільки вони працюють на абстрактних шарах;

2. Обмежені функції: Деякі функції та можливості, доступні на конкретних платформах, можуть бути важко реалізовані в крос-платформних застосунках;

3. Залежність від інструментів розробки: Крос-платформна розробка може вимагати використання конкретних фреймворків та інструментів, що обмежує вибір для розробників;

4. Вимоги до навчання: Розробка крос-платформних застосунків може вимагати додаткового навчання для розробників, які раніше працювали лише з нативними платформами;

5. Проблеми з використанням ресурсів: Використання апаратних ресурсів, таких як камера або сенсори, може бути складнішим в крос-платформних застосунках.

Отже, оцінка переваг та недоліків крос-платформної розробки мобільних застосунків для взаємодії з Opencart допомагає прийняти рішення про вибір оптимального підходу до розробки, враховуючи конкретні вимоги та обмеження проекту.

1.5.1. Розгляд підходів до крос-платформної розробки мобільних додатків та їх вплив на взаємодію з Opencart

Під час розгляду підходів до крос-платформної розробки мобільних додатків та їх впливу на взаємодію з CMS системою Opencart, важливо розглянути кілька ключових аспектів:

— використання крос-платформних фреймворків: Один із підходів - використання спеціалізованих крос-платформних фреймворків, таких як React Native, Flutter або Xamarin. Ці фреймворки дозволяють розробляти мобільні додатки, які працюють на різних платформах. Вони забезпечують доступ до апаратних можливостей пристроїв та інтеграцію з RESTful API Opencart;

— гібридні додатки: Гібридні додатки розробляються з використанням веб-технологій, таких як HTML, CSS та JavaScript, і запускаються в нативних контейнерах. Цей підхід може бути ефективним для простих додатків, але він може обмежити доступ до деяких функцій пристроїв та взаємодію з Opencart через RESTful API;

— використання веб-технологій: Іншим підходом є розробка мобільних додатків як веб-сторінок, що відкриваються у мобільних браузерах. Цей метод може бути менш витратним у розробці, але може обмежити доступ до нативних можливостей пристроїв;

— розгляд нативних рішень: В окремих випадках, де важлива висока продуктивність та доступ до специфічних функцій платформи, розробка окремих нативних додатків для кожної платформи може бути вибором. Це може бути обґрунтованим, якщо потрібна максимальна інтеграція з Opencart та оптимізована взаємодія з RESTful API;

— вплив на продуктивність і відповідь: Вибір підходу до крос-платформної розробки може вплинути на продуктивність додатків та швидкість відповіді на запити до Opencart. Важливо ретельно вивчити можливості та обмеження кожного методу для забезпечення оптимального результату.

Розглядаючи ці підходи, розробники можуть зробити обгрунтований вибір, який найкраще відповідає вимогам проекту та забезпечує ефективну взаємодію мобільного застосунку з CMS системою Opencart через RESTful API [10].

1.6. Висновки до першого розділу

Аналіз сучасного стану взаємодії мобільних додатків з CMS системою Opencart через RESTful API розкриває низку важливих аспектів, які впливають на результативність та успіх таких рішень. За ретельного розгляду пунктів, наведених у роботі, можна зробити кілька ключових висновків:

— вибір технологій та підходів: Вибір між крос-платформними та нативними методами розробки мобільних додатків важливий і повинен базуватися на конкретних вимогах проекту. Крос-платформні рішення, такі як React Native та Flutter, можуть бути корисними для зменшення витрат на розробку на різних платформах;

— продуктивність та надійність RESTful API: Оцінка продуктивності та надійності API, яке використовується для взаємодії мобільних додатків та Opencart, є важливим етапом проекту. Швидкість завантаження сторінок та реакція на запити мають велике значення для користувачів;

— тестування та відлагодження: Розробка мобільних додатків вимагає високоякісного тестування та відлагодження, особливо в контексті крос-платформних рішень. Це дозволяє виявити та виправити помилки, забезпечуючи якість продукту;

— плюси та мінуси крос-платформної розробки: Використання крос-платформних рішень дозволяє заощадити час та кошти при розробці, але може вимагати компромісів у доступі до нативних функцій. Недоліком є необхідність тестування для різних платформ;

— аналіз методів та технологій: Ретельний аналіз методів та технологій для інтеграції додатків з Opencart через RESTful API допомагає вибрати найбільш підходящий підхід для конкретного проекту.

Загалом, успішна взаємодія мобільних додатків з CMS системою Opencart вимагає обгрунтованого вибору технологій та методів розробки, уважності до продуктивності та якості API, а також систематичного тестування та відлагодження. Правильний підхід допомагає створити додатки, які задовольняють потреби користувачів та бізнес-вимоги.

РОЗДІЛ 2
АНАЛІЗ ТА РОЗБІР РОБОТИ ВЗАЄМОДІЇ RESTFUL API З CMS СИСТЕМОЮ OPENCART

2.1. Аналіз системи Opencart

Так як створення плагіну для взаємодії по RestFul API буде на базі Opencart, системною та застосованою мовою розробки плагіну буде PHP.

Створення розширення потребую встановити CMS систему, потрібно розташувати на локальному хості місце для розгортання файлів. Мова програмування яка використовується в CMS – PHP.

На даний час PHP є версії з 1 і закінчуються на 8.1. Платформа Opencart останньої версії функціонує з PHP5.6 і до 8.1. Тому з технічних характеристик, потрібно встановити останній дистрибутив. З використаних веб серверів буде Apache. Так, як розроблятися буде в ОС Ubuntu, він використовується за стандартом. База даних, буде створена MySql, так як це вимоги платформи.

Назва PHP означає Hypertext Preprocessor і позначає мову сценаріїв на стороні сервера , що означає, що програми, написані на ньому, працюють на веб-серверах і не залежать від веб-браузера . Проте з роками сфера його використання змінилася, і сьогодні мова кодування PHP входить до числа найкращих і найпопулярніших інструментів програмування для веб-розробки завдяки багатьом перевагам, які будуть у центрі уваги цієї статті. Це вважається дуже ефективною технологією, яка пропонує зручний процес розробки з багатьма додатковими інструментами, які допомагають йому. Насправді, згідно з індексом популярності мови програмування (PYPL), PHP є п'ята за популярністю мова кодування в світі [11].

Популярність PHP є логічним результатом його численних переваг, які роблять його потужним та ефективним інструментом розробки. Нижче наведено короткий список причин, чому PHP є чудовим вибором для веб-додатка, який згодом буде описано більш детально. Переваги мови:

– багато доступних спеціалістів;

- велика база довідкових та навчальних матеріалів;
- краща швидкість завантаження веб-сайтів;
- більше можливостей підключення до бази даних ;
- велика колекція доповнень з відкритим кодом ;
- недорогий хостинг веб-сайтів;
- велика синергія з HTML;
- відмінна гнучкість і скомбінованість;
- різноманітні переваги, які надають хмарні рішення.

Використання PHP прискорює завантаження сторінок веб-сайтів у порівнянні з багатьма іншими технологіями веб-розробки. Наприклад, на даний момент PHP приблизно втричі швидший за Python для більшості сценаріїв використання. У свою чергу, менший час завантаження є важливим фактором рейтингу SEO, який сприяє подальшому просуванню веб-сайту, забезпечуючи конкурентні переваги. Вища швидкість додатків забезпечує задоволення клієнтів і, у поєднанні з іншими перевагами, допомагає створити та зберегти клієнтську базу.

PHP дозволяє підключатися практично до будь-якого типу бази даних. Найпоширенішим вибором є MySQL, головним чином тому, що він безкоштовний, ефективний і популярний серед розробників. Інші надійні варіанти систем керування базами даних, сумісні з PHP, це mSQL, MS-SQL, SQLite, PostgreSQL тощо. Крім того, PHP можна однаково добре використовувати з ElasticSearch, Redis, MongoDB та іншими нереляційними базами даних. Таким чином, розробники не обмежуються використанням конкретної бази даних і можуть вибрати найбільш оптимальну для майбутнього додатка, враховуючи всі важливі фактори.

Найпоширенішим сценарієм роботи веб-сайту PHP є стек LAMP. Це означає, що веб-сайт працює на веб-сервері Apache HTTP, розгорнутому в системі Linux, і використовує MySQL як базу даних. Всі ці компоненти безкоштовні, а стек добре перевірений, що передбачає скорочення часу та коштів на розробку [11].

Хоча PHP, безсумнівно, корисний у сфері веб-розробки, він також має кілька недоліків, які не дозволяють йому домінувати в цій області. Для неупередженого огляду давайте розглянемо ці недоліки та дізнаємося, як вони можуть бути шкідливими для майбутнього програмного забезпечення та його бізнес-реалізації. Недоліки мови:

- зниження популярності;
- відсутність спеціалізованих бібліотек;
- проблеми безпеки.

SQL використовується в СУБД, призначених для різних обчислювальних систем: від персональних комп'ютерів і робочих станцій до локальних мереж, міні-комп'ютерів і великих ЕОМ. За допомогою SQL творець бази може зробити так, що різні користувачі бази даних будуть бачити різні уявлення її структури і вмісту; повноцінність як мови, призначеного для роботи з базами даних.

Ядро Apache включає в себе основні функціональні можливості, такі як обробка конфігураційних файлів, протокол HTTP і система завантаження модулів. Ядро (на відміну від модулів) повністю розробляється Apache Software Foundation, без участі сторонніх програмістів.

Теоретично, ядро apache може функціонувати в чистому вигляді, без використання модулів. Однак, функціональність такого рішення вкрай обмежена.

Частина модулів використовує в своїй роботі конфігураційні файли операційної системи (наприклад / etc / passwd і / etc / hosts).

Існують зовнішні засоби забезпечення безпеки, наприклад mod_security.

За базу даних використовується MySQL з веб-інтерфейсом phpMyAdmin, потрапити в який можна включивши сервер Apache.

phpMyAdmin - веб-додаток з відкритим кодом, написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування СУБД MySQL. PHPMyAdmin дозволяє через браузер і не тільки здійснювати адміністрування сервера MySQL, запускати команди SQL і переглядати вміст таблиць і баз даних. Додаток користується великою популярністю у веб-розробників, так як дозволяє

управляти СУБД MySQL без безпосереднього введення SQL команд, надаючи дружній інтерфейс.

Opencart використовує MVC модель приведена на рис. 2.1, MVC означає модель-перегляд-контролер. Ось що означає кожен із цих компонентів:

– модель : бекенд, який містить всю логіку даних;
– перегляд : інтерфейс або графічний інтерфейс користувача (GUI);
– контролер : мозок програми, який контролює відображення даних.

Шаблон MVC допомагає розбити інтерфейси і бекенд-код на окремі компоненти. Таким чином, набагато легше керувати та вносити зміни до будь-якої сторони, не заважаючи одна одній.

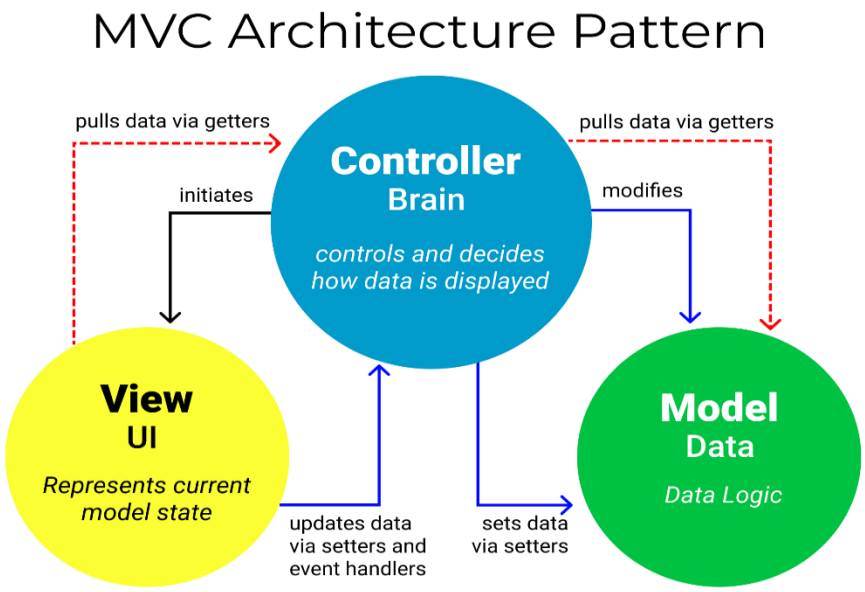


Рис. 2.1. Модель MVC

2.2. Опис структури Opencart та взаємодії її з мобільним додатком

Сформована система для якої розроблений додаток має струкутру MVC. Реалізація і файлова система сайту показана на рис. 2.2.

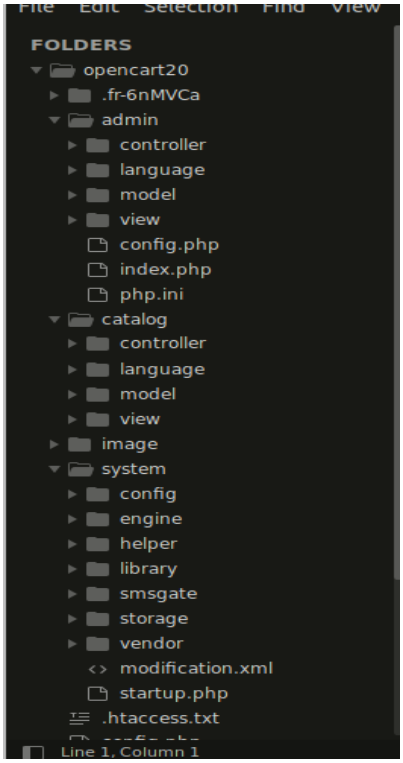


Рис. 2.2. Файлова структура CMS Opencart

OpenCart — це безкоштовна платформа електронної комерції з відкритим кодом для онлайн-торгівців. OpenCart забезпечує професійну та надійну основу для створення успішного інтернет-магазину. Ця основа підходить для широкого кола користувачів; від досвідчених веб-розробників, які шукають зручний інтерфейс для використання, до власників магазинів, які вперше відкривають свій бізнес в Інтернеті. OpenCart має велику кількість функцій, які дають сильний контроль над налаштуванням магазину. За допомогою інструментів OpenCart можна допомогти інтернет-магазину реалізувати весь свій потенціал [12].

OpenCart вимагає виконання певних технічних вимог для належної роботи магазину. По-перше, необхідно створити веб-сервер, щоб магазин OpenCart був загальнодоступним у мережі. Доменні імена та послуги хостингу можна легко придбати за доступною ціною.

Вибираючи послугу хостингу, потрібно переконатися, що ці вимоги до сервера надаються та встановлені на їхніх веб-серверах:

Ці розширення мають бути ввімкнені, щоб OpenCart правильно встановлювався на веб-сервері.

– Веб-сервер (пропонується Apache)
– PHP 5.4+
– База даних (пропонується MySQLi)

Необхідні бібліотеки / модулі PHP:

– Curl
– ZIP
– Zlib
– GD Library
– Mcrypt
– Mbstrings
– Xml

Наведені вище розширення PHP повинні бути доступні майже всім провайдерам хостингу, під час процесу встановлення він перевірить, чи всі вони включені. Потрібно звернутися до хостинг-провайдера, якщо він відсутній.

Для того, щоб створити плагін треба розділити файлові структури, для розміщення в адмін-панелі та фронтовій. В каталозі admin, розташовані файлі контролеру, моделі та відображення на рис. 2.3 показані файли, які відображають конфігурації адміністрування модулю.

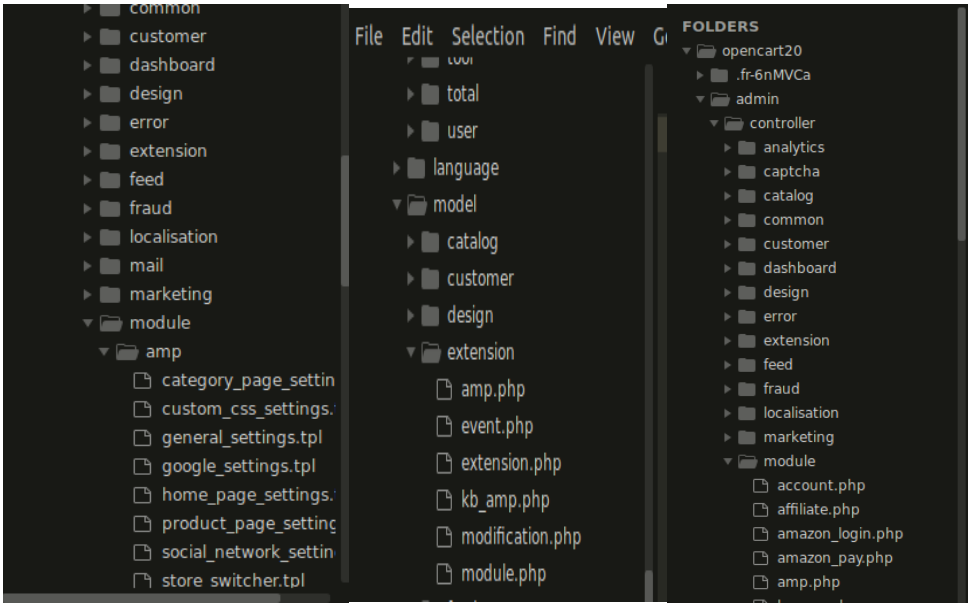


Рис. 2.3. Розташування фалів модулю в адмін панелі

Також, для того щоб зробити відображення сторінок Frontend, для мобільних сторінок, зроблене відповідно основних сторінок и сбору інформації о продуктах чи інших взаємодій. Відповідно сторінки модулю розволожені за структурою, це показано на рис. 2.4.

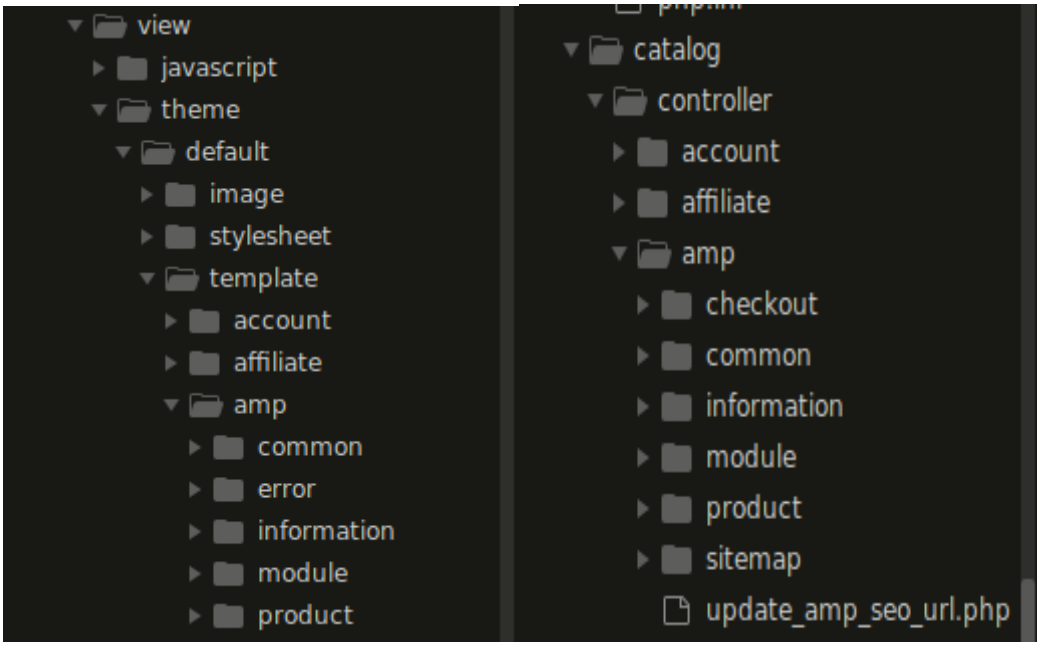


Рис. 2.4. Розташування Frontend частини плагіну

В Opencart замість основного відтворення сторінок за допомогою HTML використовується шаблонизітор Twig, який дуже зручно інтегруються з мовою на які розміщена платформа. Звичайно використання коду в шаблонізаторах забезпечує добру взаємодію з відтворенням веб сторінок.

Шаблон — це звичайний текстовий файл. Він може генерувати будь-який текстовий формат (HTML, XML, CSV, LaTeX тощо). Він не має спеціального розширення, .html або .xmlпросто добре [13].

Шаблон містить змінні або вирази, які замінюються значеннями, коли шаблон оцінюється, і теги, які керують логікою шаблону.

Нижче наведено мінімальний шаблон, який ілюструє кілька основ.

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption
}}</a></li>
        {% endfor %}
        </ul>

        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

Існує два види роздільників: {% ... %}і {{ ... }}. Перший використовується для виконання таких операторів, як цикли for, останній виводить результат виразу.

2.3. Характеристика та архітектура побудови системи взаємодії в Opencart

Для взаємодії з мобільними пристроями Opencart використовує RESTful API. RESTful API - це стиль архітектури, який використовує стандартні HTTP

методи для взаємодії з ресурсами. Це дозволяє розробникам створювати мобільні додатки, які легко інтегруються з Opencart.

Характеристика RESTful API в Opencart

RESTful API Opencart складається з наступних ресурсів:

– **/products:** Цей ресурс містить інформацію про всі товари, які доступні в магазині;

– **/categories:** Цей ресурс містить інформацію про всі категорії товарів, які доступні в магазині;

– **/customers:** Цей ресурс містить інформацію про всіх клієнтів, які зареєстровані в магазині;

– **/orders:** Цей ресурс містить інформацію про всі замовлення, які були оформлені в магазині.

Для взаємодії з кожним ресурсом використовується відповідний HTTP метод:

– **GET:** Метод GET використовується для отримання інформації про ресурс;

– **POST:** Метод POST використовується для створення нового ресурсу;

– **PUT:** Метод PUT використовується для оновлення існуючого ресурсу;

– **DELETE:** Метод DELETE використовується для видалення існуючого ресурсу.

Архітектура системи взаємодії в Opencart

Система взаємодії в Opencart складається з наступних компонентів:

– **Мобільний додаток:** Мобільний додаток використовує RESTful API Opencart для взаємодії з магазином;

– **RESTful API:** RESTful API Opencart надає доступ до інформації про магазин;

– **Веб-сервер:** Веб-сервер обробляє HTTP запити від мобільного додатка;

– **База даних:** База даних зберігає інформацію про магазин.

Приклад взаємодії мобільного додатку з RESTful API Opencart

Для того щоб отримати список всіх товарів, які доступні в магазині, мобільний додаток може виконати наступний HTTP запит:

GET /products

Відповідь на цей запит показана на рис. 2.5.

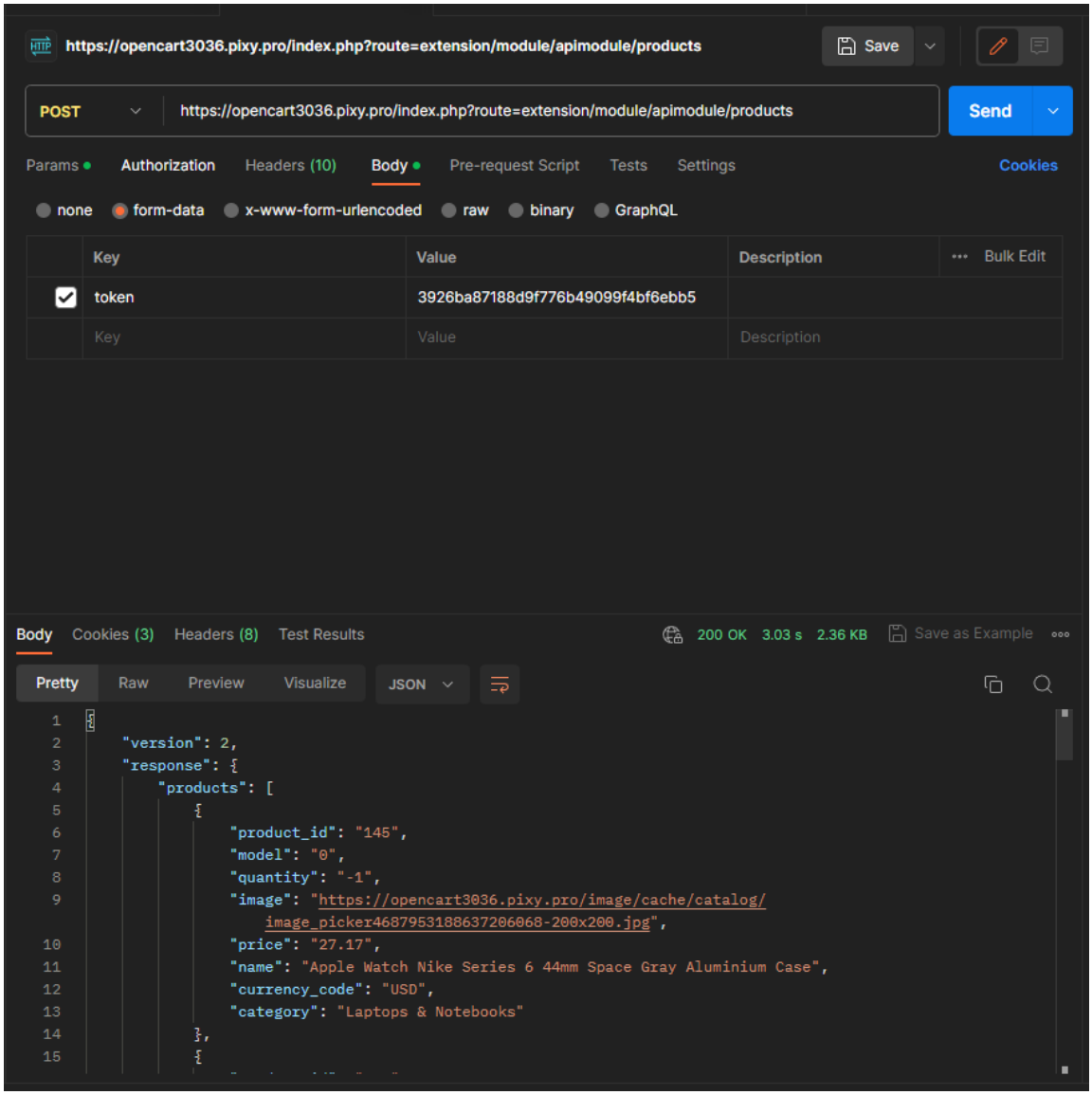


Рис. 2.5. Приклад запросу і відповіді від API

Цей приклад показує, як мобільний додаток може використовувати RESTful API Opencart для отримання інформації про магазин.

RESTful API Opencart використовується для створення різних функцій в мобільному додатку. Наприклад для наступних цілей:

– перегляд каталогу товарів: RESTful API Opencart можна використовувати для отримання інформації про товари, які доступні в магазині. Це дозволяє користувачам мобільного додатку переглядати каталог товарів, знаходити потрібні товари та отримувати додаткову інформацію про них;

– оформлення заказів: RESTful API Opencart можна використовувати для створення нових замовлень та управління ними. Це дозволяє користувачам мобільного додатку оформляти замовлення, переглядати статус своїх замовлень та отримувати інформацію про доставку;

– управління обліковим записом клієнта: RESTful API Opencart можна використовувати для управління обліковим записом клієнта. Це дозволяє користувачам мобільного додатку реєструватися в магазині, вхід в свою обліковий запис, обновлять свою інформацію;

– відправлення повідомлень клієнтам: RESTful API Opencart можна використовувати для відправки повідомлень клієнтам.

RESTful API Opencart використовується для поліпшення клієнтського досвіду. Наприклад :

– забезпечення більш швидкої та плавної взаємодії з користувачем: RESTful API Opencart може використовуватися для забезпечення більш швидкої та плавної взаємодії з користувачем. Це досягається за рахунок того, що RESTful API використовує стандартні методи HTTP, які добре оптимізовані для роботи в мережі;

– забезпечення більш персоналізованого досвіду для користувачів: RESTful API Opencart може використовуватися для забезпечення більш персоналізованого досвіду для користувачів. Це досягається за рахунок того, що RESTful API дозволяє одержувати інформацію про користувачів з їх облікових записів;

– забезпечення зручнішого доступу до інформації для користувачів: RESTful API Opencart може використовуватися для більш зручного доступу до інформації для користувачів. Це досягається за рахунок того, що RESTful API

визначає стандартний формат даних, який може бути легко прочитаний та оброблений мобільними пристроями.

2.4. Аналіз ефективності та переваг мобільного додатку на основі Flutter для взаємодії з системою Opencart

Мобільний додаток на основі Flutter для взаємодії з системою Opencart має ряд переваг, які роблять його ефективним інструментом для користувачів інтернет-магазинів [14].

Гібридність

Одним з основних переваг Flutter є його гібридність. Це означає, що додаток, написаний на Flutter, може запускатися на пристроях iOS і Android без необхідності створення окремих версій для кожної платформи. Це може заощадити розробникам час і гроші.

Швидкість і продуктивність

Flutter використовує віртуальний рушій для рендерингу графіки. Це дозволяє створювати додатки з високою частотою кадрів, а також поліпшити продуктивність. Це особливо важливо для додатків, які використовують багато графіки, таких як додатки для інтернет-магазинів.

Простота використання

Dart - це проста мова програмування, яка легко вивчається. Це може допомогти розробникам скоротити час, необхідний для розробки додатків.

Ефективність

Мобільний додаток на основі Flutter для взаємодії з системою Opencart може бути ефективним інструментом для користувачів інтернет-магазинів завдяки наступним факторам:

– гібридність: Додаток може запускатися на будь-якому пристрої, що працює на iOS або Android. Це дозволяє користувачам використовувати додаток незалежно від того, який пристрій у них є;

– швидкість і продуктивність: Додаток може відображати графіку з високою частотою кадрів, що забезпечує плавний і приємний досвід користувача. Це також може допомогти поліпшити продуктивність додатка, особливо для додатків, які використовують багато графіки;

– простота використання: Додаток може бути легко вивчений і використаний користувачами з різними рівнями навичок. Це може допомогти залучити більше користувачів і підвищити рівень їх задоволеності.

Flutter – це популярний та безкоштовний мобільний фреймворк від Google, який дозволяє створювати крос-платформні програми для iOS та Android. Він використовує мову програмування Dart, яка є простою та гнучкою.

Основи архітектури Flutter

Flutter складається з двох основних частин:

– фреймворк – це бібліотека інтерфейсу користувача на основі віджетів. Вона містить різноманітні елементи інтерфейсу користувача, такі як повзунки, кнопки, введення тексту та інші;

– SDK – це набір інструментів для розробки додатків і компіляції коду у рідний машинний код для Android та iOS;

Переваги Flutter:

Flutter має ряд переваг, які роблять його популярним вибором для розробників мобільних додатків:

– крос-платформність – Flutter дозволяє створювати додатки для iOS та Android за допомогою єдиного коду;

– висока продуктивність – додатки, написані на Flutter, працюють з високою частотою кадрів і забезпечують плавний досвід користувача;

– простота використання – Dart – це проста мова програмування, яка легко вивчається.

Мова програмування Dart

Dart – це нова мова програмування, яка розроблялася Google. Вона використовує об’єктно-орієнтований підхід і C-подібний синтаксис.

Особливості мови програмування Dart

– підтримка ООП – Dart підтримує класифікації, інтерфейси, абстрактні класи, generics, mixins та статичну типізацію;

– динамічна та статична типізація – у Dart є можливість декларувати типи змінних або використовувати змінні без явного вказання типу;

– компіляція в JavaScript – код на Dart можна скомпілювати в JavaScript для виконання в веб-браузері;

– підтримка потоків – Dart підтримує потоки ізольованих об’єктів, які не розділяють пам’ять і спілкуються за допомогою повідомлень.

Нижче на рис. 2.6 показана архітектура Flutter.

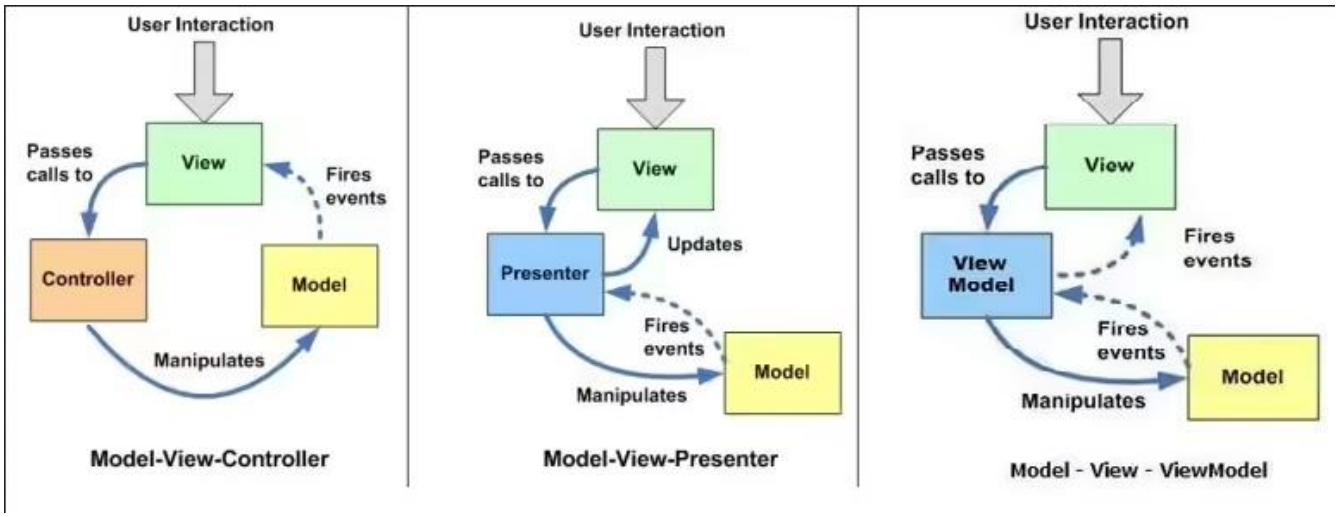


Рис. 2.6. Архитектура додатку на Flutter

Для розробки додатку була використана архітектура MVVM.

Архітектура MVVM (Model-View-ViewModel) - це шаблон для розробки програмного забезпечення, який розділяє додаток на три основні компоненти:

– модель відповідає за зберігання даних та бізнес-логіку додатку;

– перегляд відповідає за відображення даних на екрані;

– ViewModel відповідає за взаємодію між моделлю та переглядом.

Особливості архітектури MVVM:

– розділення відповідальності. Модель, перегляд та ViewModel відповідають за різні аспекти додатку, що полегшує розробку та обслуговування;

– легкість тестування. Кожен компонент архітектури може бути протестований незалежно від інших, що покращує якість коду;

– підтримка реактивного програмування. Архітектура MVVM добре підходить для використання з реактивним програмуванням, що дозволяє створювати додатки з високою продуктивністю та динамічністю.

Переваги архітектури MVVM:

– зробити код більш зрозумілим і організованим. Модель, перегляд та ViewModel відповідають за різні завдання, що полегшує розуміння коду;

– полегшити тестування. Кожен компонент архітектури може бути протестований незалежно від інших, що покращує якість коду;

– зробити додаток більш динамічним. Архітектура MVVM добре підходить для використання з реактивним програмуванням, що дозволяє швидко реагувати на зміни даних.

Розглянемо детальніше особливості та переваги архітектури MVVM:

Розділення відповідальності

Одна з основних переваг архітектури MVVM - це розділення відповідальності між різними компонентами додатку. Модель відповідає за зберігання даних та бізнес-логіку, перегляд відповідає за відображення даних на екрані, а ViewModel відповідає за взаємодію між моделлю та переглядом.

Це розділення відповідальності має ряд переваг:

– збільшує зрозумілість коду. Кожен компонент архітектури відповідає за певний аспект додатку, що полегшує розуміння коду;

– полегшує тестування. Кожен компонент архітектури може бути протестований незалежно від інших, що покращує якість коду;

– полегшує масштабування додатку. Додавання нових функцій до додатку можна здійснювати, не вносячи змін до існуючих компонентів архітектури.

Легкість тестування

Ще одна перевага архітектури MVVM - це легкість тестування. Кожен компонент архітектури може бути протестований незалежно від інших. Це покращує якість коду та дозволяє виявляти помилки на ранніх етапах розробки.

Архітектура MVVM добре підходить для реактивного програмування, оскільки ViewModel відповідає за взаємодію між моделлю та переглядом. ViewModel може використовувати сповіщення для інформування перегляду про зміни даних.

Загалом, архітектура MVVM є хорошим вибором для розробки мобільних додатків. Вона забезпечує ряд переваг, які роблять розробку та обслуговування додатків більш ефективними. Нижче на рис. 2.6 приведена структура MVVM [15].

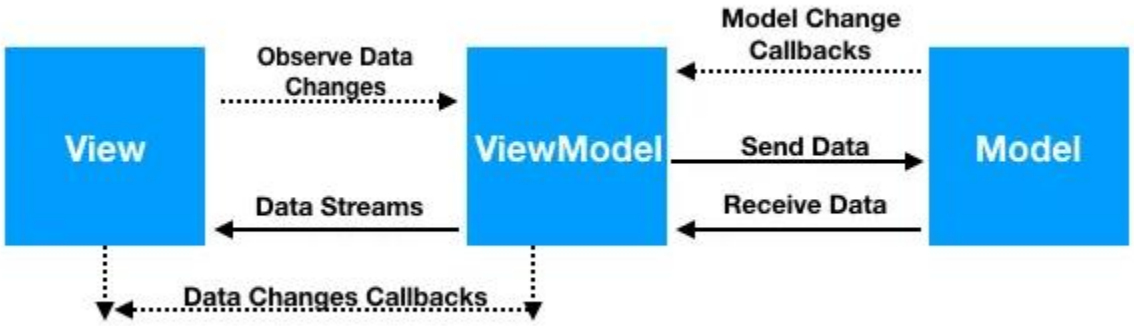


Рис. 2.6. Структура MVVM

Модель відповідає за зберігання даних та бізнес-логіку додатку. У випадку з додатком для перегляду списку товарів модель представляє собою клас ProductModel, який має такі поля:

– id - ідентифікатор товару;
– name - назва товару;
– price - ціна товару.

Перегляд відповідає за відображення даних на екрані. У цьому випадку перегляд представляє собою клас ProductListPage, який має такий код:

Dart

```
class ProductListPage extends StatelessWidget {
  const ProductListPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Product List"),
      ),
      body: ListView.builder(
        itemCount: productViewModel.products.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text(productViewModel.products[index].name),
            subtitle:
Text(productViewModel.products[index].price.toString()),
          );
        },
      ),
    );
  }
}
```

Цей код використовує Provider для інжектування ViewModel-у в перегляд. ViewModel надає список товарів, який перегляд використовує для відображення на екрані.

ViewModel відповідає за взаємодію між моделлю та переглядом. У цьому випадку ViewModel представляє клас ProductViewModel, який має такий код:

Dart

```
class ProductViewModel extends ChangeNotifier {
  ProductViewModel({required this.products});

  final List<ProductModel> products;

  void addProduct(ProductModel product) {
    products.add(product);
    notifyListeners();
  }
}
```

Цей код отримує список товарів від моделі і використовує його для відображення на екрані. Також ViewModel надає метод addProduct(), який дозволяє користувачам додавати нові товари до списку.

2.5. Використані розширення для мобільного додатку

Розширення, які використовувалися для розробки мобільного додатку на Flutter:

- flutter_svg - розширення для роботи з векторною графікою SVG;
- firebase - розширення для інтеграції додатку з Firebase;
- image_picker - розширення для вибору зображень з галереї пристрою;
- path_provider - розширення для доступу до файлової системи пристрою;
- shared_preferences - розширення для зберігання налаштувань додатку;
- webview_flutter - розширення для впровадження веб-вмісту в додаток.

flutter_svg

Розширення flutter_svg дозволяє працювати з векторною графікою SVG. Воно надає такі функції, як:

- завантаження та рендеринг SVG-файлів;
- створення SVG-контейнерів;
- використання SVG-контейнерів у макетах.

Розширення flutter_svg можна використовувати для створення красивих і складних інтерфейсів для мобільних додатків.

firebase

Розширення firebase дозволяє інтегрувати додаток з Firebase. Firebase - це платформа для розробки мобільних додатків, яка надає такі послуги, як [17]:

- хмарне сховище;
- бази даних;
- розсилання сповіщень;
- аналіз даних.

Розширення firebase можна використовувати для розширення функціональності мобільних додатків, таких як:

- зберігання даних користувачів;
- розсилання сповіщень користувачам;
- аналіз поведінки користувачів.

image_picker

Розширення image_picker дозволяє вибирати зображення з галереї пристрою. Воно надає такі функції, як:

- завантаження зображень з галереї;
- збереження зображень у галереї;
- обробка зображень.

Розширення image_picker можна використовувати для таких завдань, як:

- завантаження зображень профілю користувачів;
- редагування зображень;
- додавання зображень до додатків.

path_provider

Розширення path_provider дозволяє отримати доступ до файлової системи пристрою. Воно надає такі функції, як:

- отримання шляху до кореневої папки пристрою;
- отримання шляху до папки користувача;
- створення та видалення файлів та папок.

Розширення path_provider можна використовувати для таких завдань, як:

- збереження даних користувачів;
- завантаження файлів з Інтернету;
- зберігання журналів помилок.

shared_preferences

Розширення shared_preferences дозволяє зберігати налаштування додатку. Воно надає такі функції, як:

- зберігання ключ-значення;
- отримання ключ-значення;
- видалення ключ-значення.

Розширення shared_preferences можна використовувати для таких завдань, як:

- збереження мови додатку;
- збереження темного режиму додатку;

– збереження підписки користувача.

webview_flutter

Розширення webview_flutter дозволяє впроваджувати веб-вміст в додаток. Воно надає такі функції, як:

– відображення веб-сторінок;

– взаємодія з веб-сторінками;

– керування веб-браузером.

Архітектура зробленого мобільного за стосунку показана на рис. 2.7.

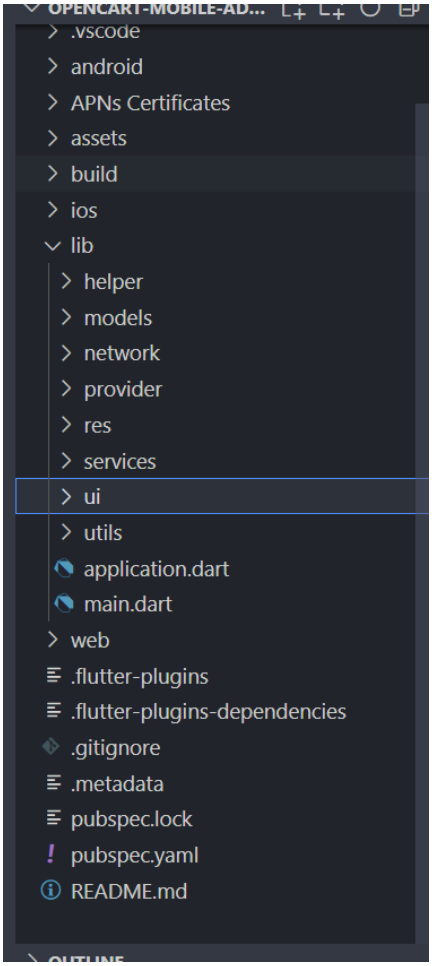


Рис. 2.7. Архітектура мобільного додатку

На рис. 2.8 показано основний клас провайдер в якому відбувається взаємодія з системою Opencart.

Даний класс зроблений по принципу SingeTone тому, що даний клас постійно викликається в програмі. І для того, щоб додаток працював швидше він зберігається в пам’яті.

Цей клас використовує бібліотеку Dio, для відправки та обробки HTTP запитів.

Обробка помилок робиться також за допомоги бібліотеки Dio і це дає можливість більш детально відслідковувати роботу запитів і всіх помилок.

Для того, щоб взаємодіяти з API безпечно, використовується бібліотека Flutter Secure Storage, яка реалізовує доступ до Secure Storage в Android і IOS. В цій бібліотеці зберігається токен, для авторизації користувача [18].

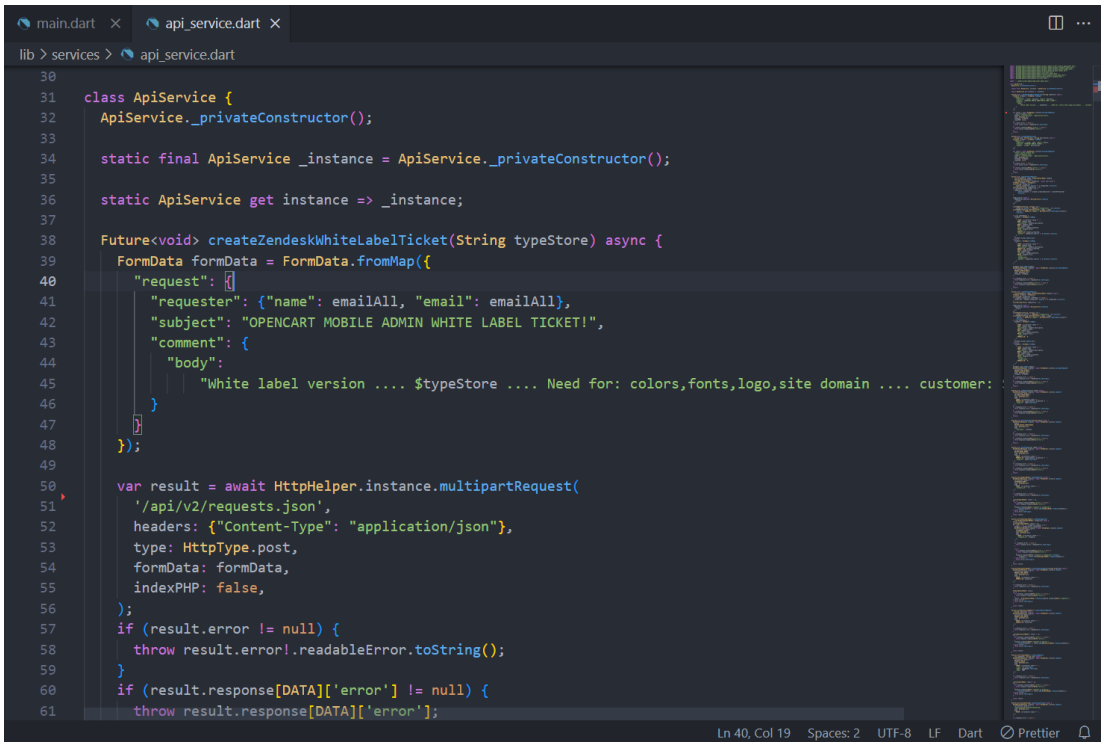


Рис. 2.8. Клас провайдер для API

На рис. 2.9 показано основний клас проекту. В даному класі є обгортка, яка називається runZonedGuarded(). Вона дозволяє відловлювати всі критичні помилки, які можуть зупинити роботу додатку. Завдяки цьому додаток не буде припиняти свою роботу при критичній помилці.

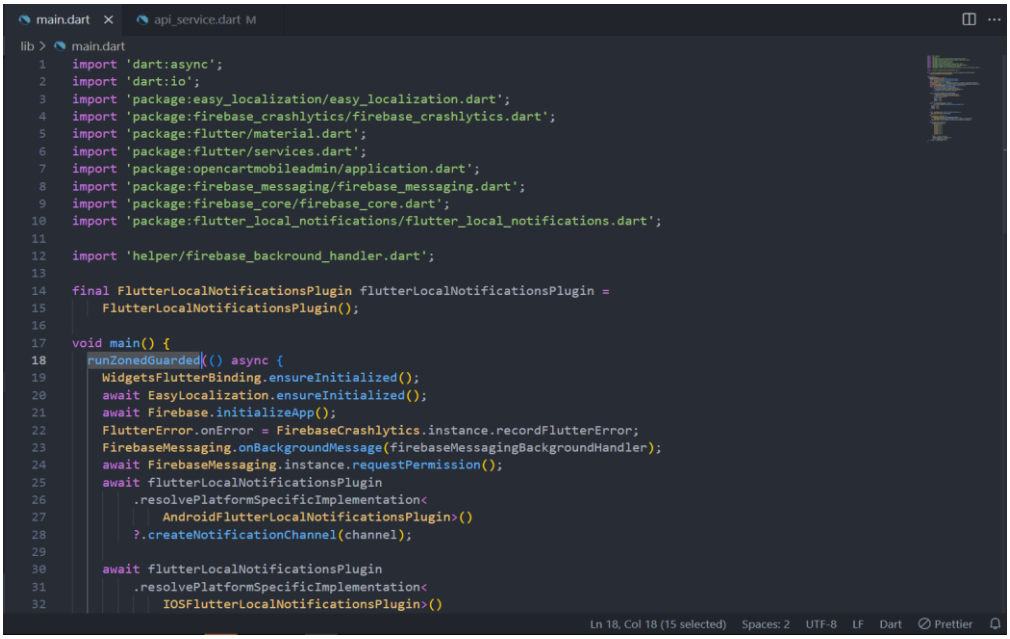


Рис. 2.9. Основний клас входу в проєкт

На рис. 2.10 показано реалізацію MVVM у мобільному додатку. У моделі є функція onReady(), яка виконується один раз при відкритті сторінки. У модулі представлені все методі які працюють з моделлю даних. По такій аналогії зроблені всі View-Model, для кожної реалізації View компоненту у застосунку.

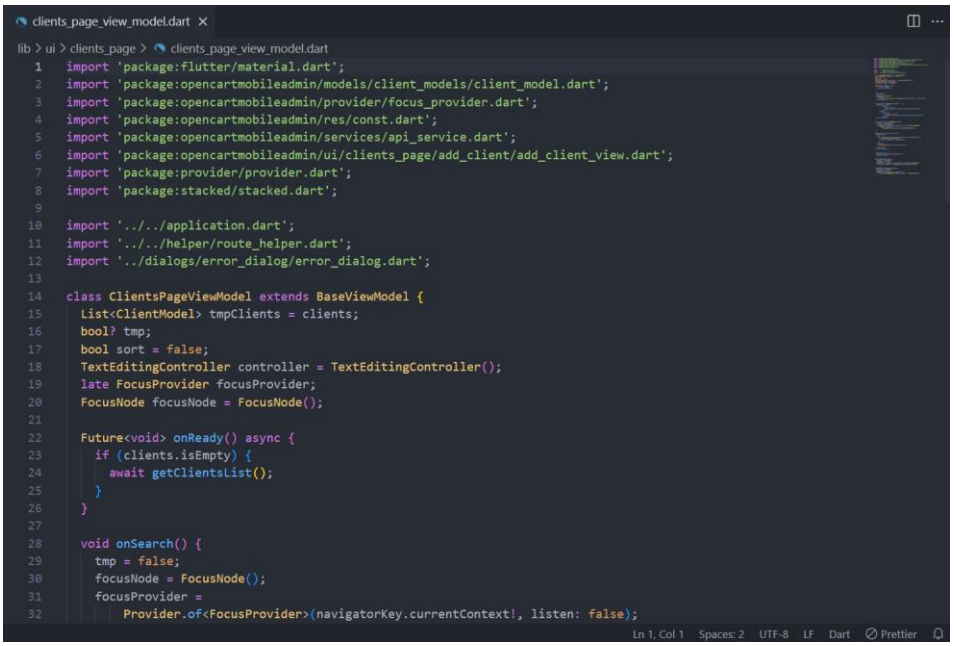


Рис. 2.10. View-Model для компоненту clients

На рис. 2.11 показана модель clients, яка описує всі данні моделі з якою взаємодіє View-Model. Також використовується фабричний метод, для створення моделей даних від API, які приходять в форматі JSON.

Всі поля моделі є nullable, для забезпечення сумісності з різними версіями API.

clients_page_view_model.dart × client_model.dart ×

lib > models > client_models > client_model.dart

```dart
class ClientModel {
  final String? clientId, fio, quantity, currencyCode;
  final double? total;

  ClientModel({
    this.clientId,
    this.fio,
    this.total,
    this.quantity,
    this.currencyCode,
  });

  factory ClientModel.fromJson(Map<String, dynamic> json) {
    return ClientModel(
          clientId: json["client_id"].toString(),
          fio: json["fio"].toString(),
          total: double.parse(json["total"]?.toString() ?? '0.0'),
          quantity: json["quantity"].toString(),
          currencyCode: json["currency_code"].toString(),
        );
  }
}
```

Рис. 2.11. Модель client

На рис. 2.12 показано View clients. В ньому використовується клас ViewModelBuilder для забезпечення взаємодії View з View-Model.

```dart
class ClientsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ViewModelBuilder<ClientsPageViewModel>.reactive(
        viewModelBuilder: () => ClientsPageViewModel(),
        onViewModelReady: (model) => model.onReady(),
        builder: (context, model, child) {
          return KeyboardDismissOnTap(
            dismissOnCapturedTaps: true,
            child: FocusLayout(
              child: MainLayout(
                  bottomBar: NavigationBottomBar(),
                  endDrawer: DrawerMenu(),
                  appBar: AppBar(
                    centerTitle: false,
                    elevation: 0.0,
                    actions: <Widget>[
                      ClientSearchIcon(
                          search: (value) =>
                              model.searchClient(searchText: value)),
```

Рис. 2.12. View сторінки client

На рис. 2.13 показані всі сервіси проекту. Всі сервіси є Single Tone, тому що вони часто викликаються у різних місцях проекту. Ці сервіси винесені в головний Layout проекту тому, що вони уніфіковані.

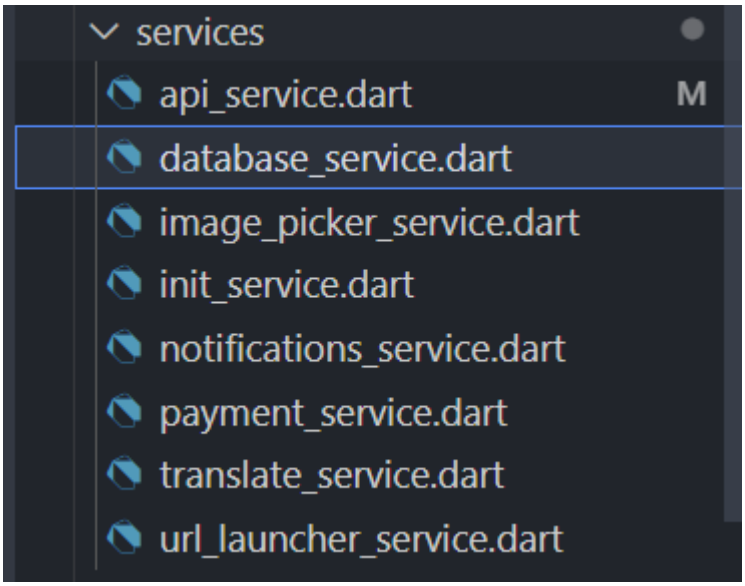


Рис. 2.13. Сервіси мобільного додатку

Весь застосунок збудований по принципу Clean архітектури, де всі головні компоненті розбиті на Layouts. Що дозволяє краще вносити зміну до коду, без

завдання шкоди вже працюючому функціоналу. Ця архітектура показана на рис. 2.14.

Рис. 2.14. Архітектура Clean

2.6. Висновки до другого розділу

У другому розділі було проведено аналіз та розбір роботи взаємодії RESTful API з CMS системою Opencart. Були розглянуті такі аспекти, як:

- методи взаємодії мобільних додатків із CMS системами;
- особливості взаємодії RESTful API з Opencart;
- ефективність різних підходів до взаємодії мобільних додатків із CMS системами Opencart.

В результаті аналізу було встановлено, що існує кілька методів взаємодії мобільних додатків із CMS системами. Найпростішим способом є використання асинхронних HTTP запитів. Однак цей метод може бути менш ефективним, ніж інші підходи, оскільки вимагає додаткового коду для обробки запитів та відповідей [19].

Більш ефективним способом є використання бібліотек для RESTful API. Ці бібліотеки часто надають такі функції, як:

- автоматичне форматування запитів та відповідей;

- підтримка асинхронних запитів;
- обробка помилок.

Також можна використовувати віртуальні HTTP-сервери для імітації RESTful API в локальному середовищі розробки.

Найбільш ефективним підходом до взаємодії мобільних додатків із CMS системами Opencart є використання бібліотек для RESTful API. Ці бібліотеки забезпечують високу продуктивність, а також пропонують широкий набір функцій, які спрощують розробку та тестування мобільних додатків.

Важливою особливістю взаємодії RESTful API з Opencart є те, що Opencart підтримує RESTful API за замовчуванням. Це означає, що для взаємодії з CMS системою Opencart не потрібно налаштовувати додаткові параметри.

У цілому, взаємодія RESTful API з CMS системами Opencart є ефективним та простим способом для розробників мобільних додатків.

РОЗДІЛ 3
ОПИСАННЯ ФУНКЦІЙ У СИСТЕМІ OPENCART ТА У МОБІЛЬНОМУ ДОДАТКУ

3.1. Інтерфейс і функціонал плагіну для Opencart

Для того, щоб розробити API для Opencart треба було зробити зручний функціонал, щоб можна було використовувати його. Інтерфейс користувача для адмін-панелі адміністратора показано на рис. 3.1.

По функціональності в панелі є включення, виключення самого плагіну. Також так, як платформа може буде з декількома сайтами і для кожного є свої продукті і індивідуальна інформація, тому у налаштуваннях є вибір сайту, з якого будуть віддаватися інформація о сайті [20].

На рис. 3.2 показано функціонал в якому виводяться всі девайсі котрі підключені до сайту. В таблиці можна відслідкувати коли і який девайс підключався до сайту і забирав інформацію. І якщо побачити девайс котрій не повинен маті доступу, його можна видалити.

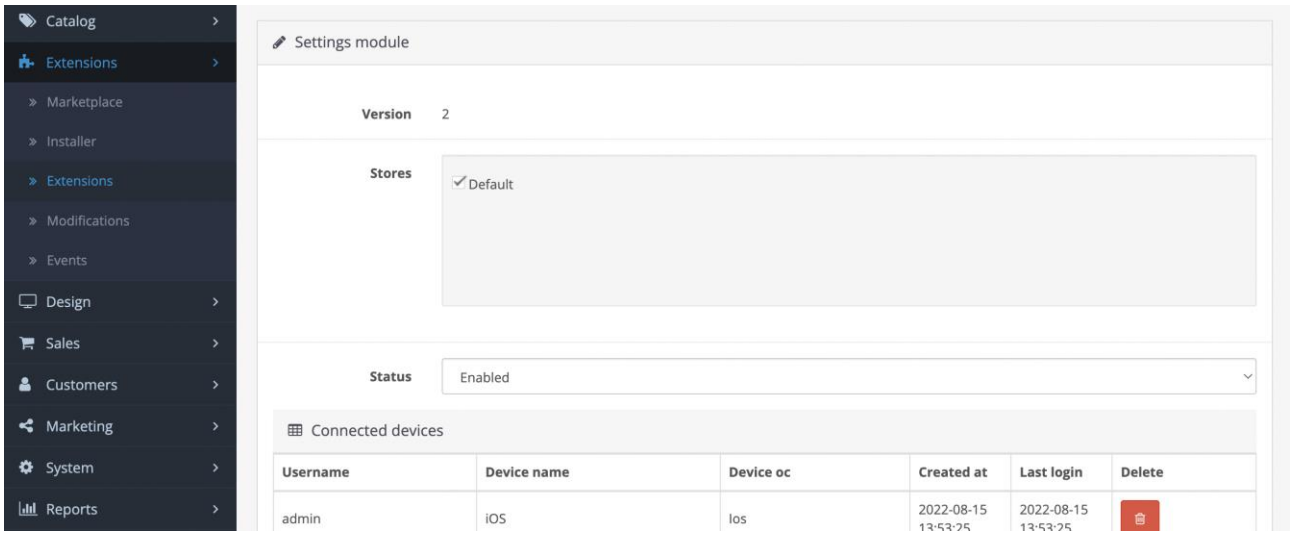


Рис. 3.1. Інтерфейс доступу до плагіну в Opencart

Для того, щоб забезпечити безпечне підключення, до сайту було використане генерування токену, для кожного користувача і коли мобільний додаток підключаєтеся до платформи він отримує, згенерований 32 символьний токен.

Connected devices

Username	Device name	Device oc	Created at	Last login	Delete
admin	iOS	Ios	2022-08-15 13:53:25	2022-08-15 13:53:25	
admin		android	2022-08-17 11:29:19	2022-08-17 11:29:21	
admin		android	2022-08-19 10:25:01	2022-08-22 18:31:01	
admin		android	2022-09-28 06:47:55	2022-09-28 06:47:59	
admin		android	2022-09-28 08:04:49	2022-09-28 08:52:46	
admin		ios	2022-09-30 07:34:56	2022-09-30 07:35:01	
admin		ios	2022-09-30 10:52:02	2022-09-30 10:52:04	
admin		ios	2022-09-30 10:55:23	2022-09-30 10:55:27	
admin		ios	2022-09-30 11:03:44	2022-09-30 11:03:53	
admin		android	2022-09-30	2022-10-03	

Рис. 3.2. Підключені до сайту додатки

Для того, щоб відтавати інформацію про статистику треба було при установці зареєструвати Event функції, всі івенти показані на рис. 3.3. Вони булі зареєстровані при інсталяції плагіну.

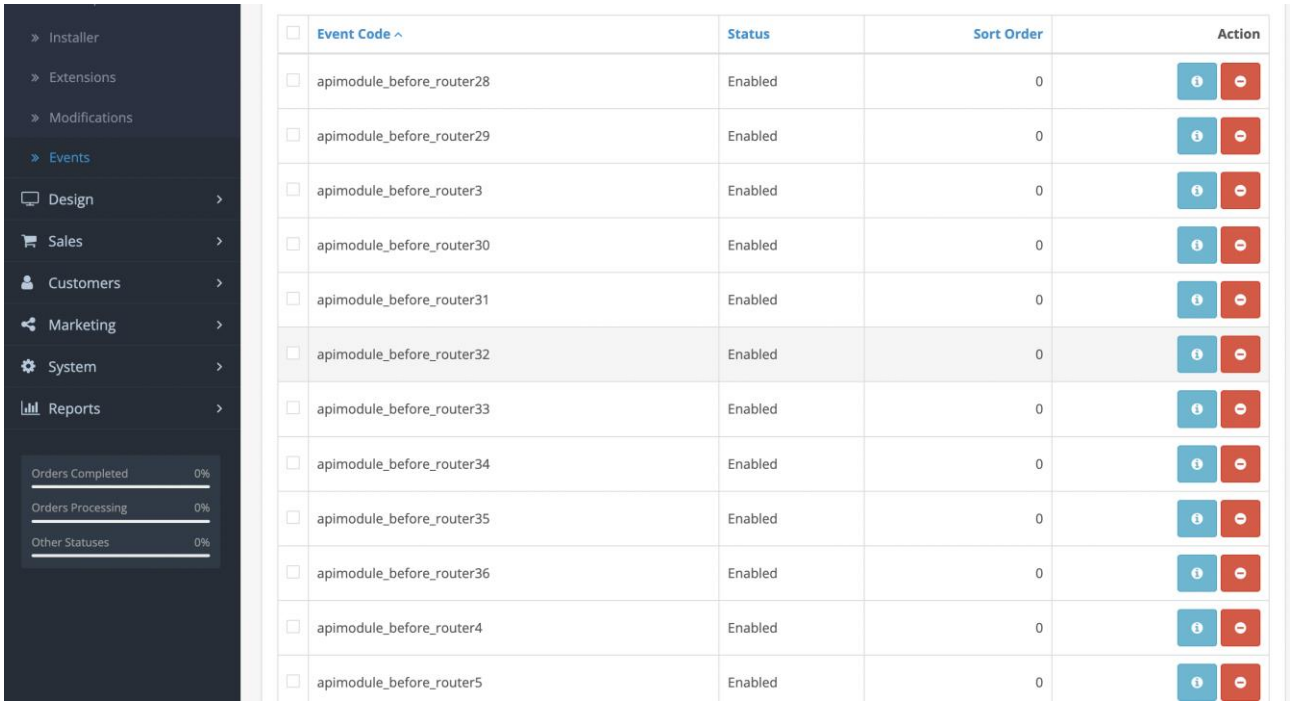


Рис. 3.3. Зареєстровані функції в івентах

Оголошення основних методів, які буде відтавати API зазначені в основному контролері плагіну, приклад методів виглядає так:

```
private $error = [];
private $API_VERSION = 0;
private $OPENCART_VERSION = 0;
const API_METHODS = [
    'login',
    'getOption',
    'addOrder',
    'statistic',
    'orders',
    'getorderinfo',
    'paymentanddelivery',
    'orderproducts',
    'orderhistory',
    'orderstatuses',
    'changestatus',
    'delivery',
    'clients',
    'products',
    'productinfo',
    'getSubstatus',
    'clientinfo',
    'clientorders',
    'updatedevicetoken',
    'deletedevicetoken',
    'updateProduct',
    'getReview',
    'mainImage',
    'deleteImage',
    'getCategories',
    'newOrder',
```

```
        'getClient',
        'getCountry',
        'setZone',
        'getCustomerGroup',
        'getMethodShipping',
        'getMethodPayment',
        'getStore',
        'getCurrency',
        'getProducts',
        'addReview',
        'editReview',
    ];
```

Для кращої взаємодії користувача з своїми клієнтами, у додаткових функціях було додано функції відправку пуш повідомлень. Щоб реалізувати цю функцію прі інсталяції плагіну була зроблена модифікація, яка додає в основній код застосунку, екшн який спрацьовує при оформлення клієнтом замовлення. Приклад цього коду приведений нижче:

```
private function sendCurl($fields)
{
    $API_ACCESS_KEY = '{$key} ';
    $headers = [
        'Authorization: key=' . $API_ACCESS_KEY,
        'Content-Type: application/json'
    ];

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, 'https://fcm.googleapis.com/fcm/send');
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($fields));
    curl_exec($ch);
    curl_close($ch);

}
```

Для того, щоб передавати інформацію про замовлення було використано технологію FireBase. Ця технологія дає можливість в реальному часі передавати повідомлення на мобільні пристрої за допомогою токена, який генерується при відкритті додатку.

Firebase - це платформа для розробки мобільних додатків, яка надає такі послуги, як:

– хмарне сховище;

– бази даних;

– розсилання сповіщень;

– аналіз даних.

FireBase Cloud Messaging (FCM) - це послуга, яка дозволяє розробникам мобільних додатків надсилати пуш-повідомлення користувачам. Пуш-повідомлення - це сповіщення, які можуть бути отримані користувачем, навіть якщо додаток не запущено.

Для використання FCM у Flutter потрібно додати наступні пакети в свій проект:

```
dependencies:
  firebase_core: ^8.0.0
  firebase_messaging: ^9.0.0
Після цього потрібно ініціалізувати FCM у своєму додатку:
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  // Ініціалізувати FCM
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    // Обробити повідомлення
  });
}
```

Для надсилання пуш-повідомлення потрібно створити об'єкт RemoteMessage і вказати в ньому наступні поля:

– title - заголовок повідомлення;

– body - тіло повідомлення;

– data - додаткові дані, які можна використовувати для обробки повідомлення.

Наприклад, наступний код надсилає пуш-повідомлення з заголовком "Новий товар" і тілом "Сьогодні в продажу з'явився новий товар".

```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  // Ініціалізувати FCM
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    // Обробити повідомлення
  });

  // Створити повідомлення
  RemoteMessage message = RemoteMessage(
    title: "Новий товар",
    body: "Сьогодні в продажу з'явився новий товар",
```

```
  );

  // Надіслати повідомлення
  FirebaseMessaging.send(message);
}
```

Для отримання пуш-повідомлень потрібно реалізувати обробник FirebaseMessaging.onMessage. У цьому обробнику можна отримати інформацію про повідомлення, наприклад, його заголовок, тіло і додаткові дані.

Наприклад, наступний код відображає заголовок і тіло пуш-повідомлення в полі Text().

```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  // Ініціалізувати FCM
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    // Обробити повідомлення
    Text(
      "${message.notification?.title}: ${message.notification?.body}",
    );
  });
}
```

Для того, щоб відсилати пуш повідомлення були створені таблиці для збереження та перевірки даних які приходять від запитів. Приклад таблиць і їх наповнення приведене на рис. 3.4 і 3.5.

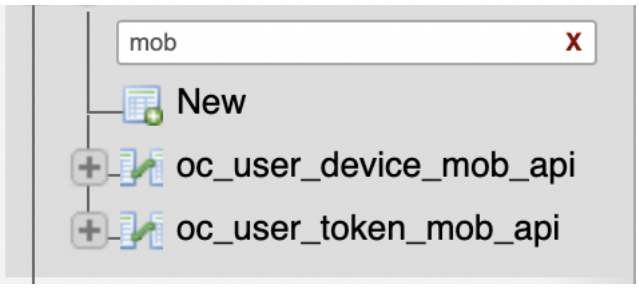
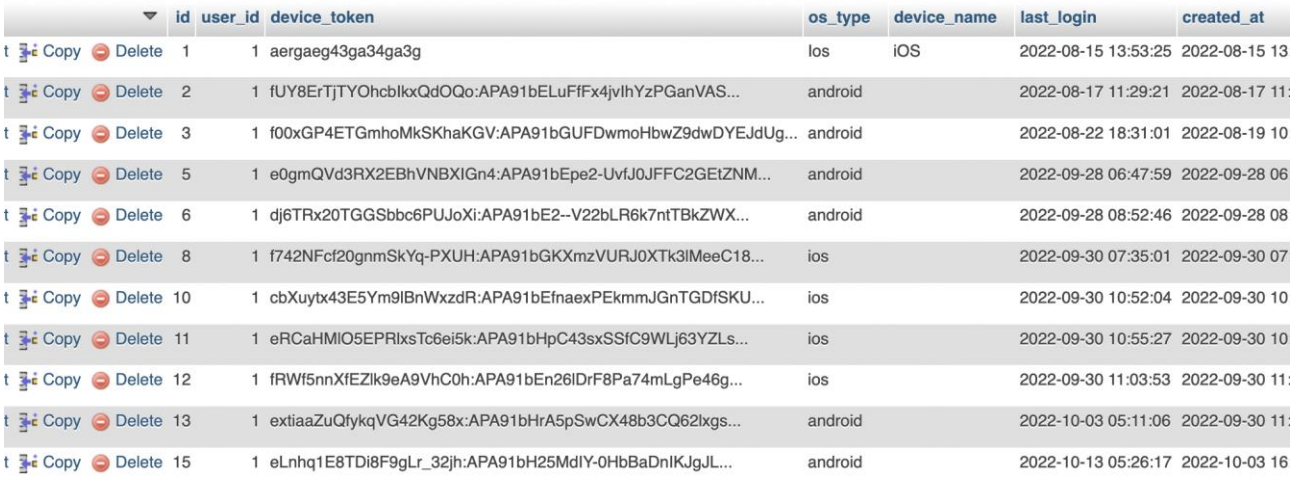


Рис. 3.4. Структура таблиць плагіну



	id	user_id	device_token	os_type	device_name	last_login	created_at
Copy Delete	1	1	aergaeg43ga34ga3g	los	iOS	2022-08-15 13:53:25	2022-08-15 13
Copy Delete	2	1	fUY8ErTjTYOhcbIkxQdOQo:APA91bELuFfFx4jvIhYzPGanVAS...	android		2022-08-17 11:29:21	2022-08-17 11
Copy Delete	3	1	f00xGP4ETGmhoMkSKhaKGV:APA91bGUFDwmoHbwZ9dwDYEJdUg...	android		2022-08-22 18:31:01	2022-08-19 10
Copy Delete	5	1	e0gmQVd3RX2EBhVNBXIGn4:APA91bEpe2-UvfJ0JFFC2GEtZNM...	android		2022-09-28 06:47:59	2022-09-28 06
Copy Delete	6	1	dj6TRx20TGGSbbc6PUJoXi:APA91bE2--V22bLR6k7ntTBkZWX...	android		2022-09-28 08:52:46	2022-09-28 08
Copy Delete	8	1	f742NFcf20gnmSkYq-PXUH:APA91bGKXmzVURJ0XTk3lMeeC18...	ios		2022-09-30 07:35:01	2022-09-30 07
Copy Delete	10	1	cbXuytx43E5Ym9lBnWxzdR:APA91bEfnaexPEkmmJGnTGDfSKU...	ios		2022-09-30 10:52:04	2022-09-30 10
Copy Delete	11	1	eRCaHMlO5EPRlxsTc6ei5k:APA91bHpC43sxSSfC9WLj63YZLs...	ios		2022-09-30 10:55:27	2022-09-30 10
Copy Delete	12	1	fRWf5nnXfEZlk9eA9VhC0h:APA91bEn26lDrF8Pa74mLgPe46g...	ios		2022-09-30 11:03:53	2022-09-30 11
Copy Delete	13	1	extiaaZuQfykqVG42Kg58x:APA91bHrA5pSwCX48b3CQ62lxgs...	android		2022-10-03 05:11:06	2022-09-30 11
Copy Delete	15	1	eLnhq1E8TDi8F9gLr_32jh:APA91bH25MdlY-0HbBaDnIKJgJL...	android		2022-10-13 05:26:17	2022-10-03 16

Рис. 3.5. Наповнення таблиці oc_user_device_mob_api

Також з цих таблиць береться інформація, для відображення таблиці підключених девайсів і токен для перевірки, що користувач підключений до API.

Вся структура API зроблена з урахуванням плагінів та додаткових змін на сайті і використовує основні методі и функції системи.

3.2. Оптимізація взаємодії API з основною структурою бази даних та запитів

Оптимізація структури бази даних є першим кроком до оптимізації взаємодії API з нею. Для цього необхідно використовувати такі підходи, як:

– нормальний розподіл даних - це процес організації даних у таблицях таким чином, щоб уникнути повторень та дублювання;

– кешування даних - це зберігання часто використовуваних даних у пам'яті, щоб уникнути повторних запитів до бази даних;

– індексування даних - це створення структури, яка дозволяє швидко знаходити необхідні дані в базі даних.

Так, як можна б було використати основні функції для роботі з даними в API, але в них є зайві залежності, які уповільнюють роботу. Було прийнято рішення оптимізувати ці запиті і зробити індексацію потрібних таблиць.

Приклад використання оптимізації запитів наведений нижче:

```
public function getProducts($page)
{
    $sql = "SELECT p.`product_id` "
          ."FROM `" . DB_PREFIX . "product` p "
          ."LEFT JOIN `" . DB_PREFIX . "product_description` pd ON
(p.`product_id` = pd.`product_id`) "
          ."LEFT JOIN `" . DB_PREFIX . "product_to_store` p2s ON
(p.`product_id` = p2s.`product_id`) "
          ."LEFT JOIN `" . DB_PREFIX . "stock_status` ss ON
(p.`stock_status_id` = ss.`stock_status_id`) "
          ."WHERE pd.`language_id` = " . (int) $this->config-
>get('config_language_id') . " "
          ."AND p.`status` = 1 "
          ."AND p.`date_available` <= NOW() "
          ."AND p2s.`store_id` = " . (int) $this->config-
>get('config_store_id') . " "
          ."GROUP BY p.`product_id` "
          ."ORDER BY p.`product_id` ASC "
          ."LIMIT 5 OFFSET " . (int) $page;
    $query = $this->db->query($sql);
    $this->load->model('catalog/product');
    foreach ($query->rows as $result) {
        $product_data[$result['product_id']] = $this-
>model_catalog_product->getProduct($result['product_id']);
    }
    return $product_data;
}
```

Ця функція визначена для отримання всіх продуктів які має користувач. А нижче наведена стандартна функція для отримання продуктів на сайті.

```
public function getProduct($product_id) {
    $query = $this->db->query("SELECT DISTINCT *, pd.name AS name,
p.image, m.name AS manufacturer, (SELECT price FROM " . DB_PREFIX .
"product_discount pd2 WHERE pd2.product_id = p.product_id AND
pd2.customer_group_id = '" . (int)$this->config-
>get('config_customer_group_id') . "' AND pd2.quantity = '1' AND
((pd2.date_start = '0000-00-00' OR pd2.date_start < NOW()) AND
(pd2.date_end = '0000-00-00' OR pd2.date_end > NOW())) ORDER BY
pd2.priority ASC, pd2.price ASC LIMIT 1) AS discount, (SELECT price FROM "
. DB_PREFIX . "product_special ps WHERE ps.product_id = p.product_id AND
ps.customer_group_id = '" . (int)$this->config-
>get('config_customer_group_id') . "' AND ((ps.date_start = '0000-00-00'
OR ps.date_start < NOW()) AND (ps.date_end = '0000-00-00' OR ps.date_end >
NOW())) ORDER BY ps.priority ASC, ps.price ASC LIMIT 1) AS special,
(SELECT points FROM " . DB_PREFIX . "product_reward pr WHERE pr.product_id
= p.product_id AND pr.customer_group_id = '" . (int)$this->config-
>get('config_customer_group_id') . "') AS reward, (SELECT ss.name FROM " .
DB_PREFIX . "stock_status ss WHERE ss.stock_status_id = p.stock_status_id
AND ss.language_id = '" . (int)$this->config->get('config_language_id') .
"') AS stock_status, (SELECT wcd.unit FROM " . DB_PREFIX .
"weight_class_description wcd WHERE p.weight_class_id =
wcd.weight_class_id AND wcd.language_id = '" . (int)$this->config-
>get('config_language_id') . "') AS weight_class, (SELECT lcd.unit FROM "
. DB_PREFIX . "length_class_description lcd WHERE p.length_class_id =
lcd.length_class_id AND lcd.language_id = '" . (int)$this->config-
>get('config_language_id') . "') AS length_class, (SELECT AVG(rating) AS
total FROM " . DB_PREFIX . "review r1 WHERE r1.product_id = p.product_id
AND r1.status = '1' GROUP BY r1.product_id) AS rating, (SELECT COUNT(*) AS
total FROM " . DB_PREFIX . "review r2 WHERE r2.product_id = p.product_id
```

```
AND r2.status = '1' GROUP BY r2.product_id) AS reviews, p.sort_order FROM
" . DB_PREFIX . "product p LEFT JOIN " . DB_PREFIX . "product_description
pd ON (p.product_id = pd.product_id) LEFT JOIN " . DB_PREFIX .
"product_to_store p2s ON (p.product_id = p2s.product_id) LEFT JOIN " .
DB_PREFIX . "manufacturer m ON (p.manufacturer_id = m.manufacturer_id)
WHERE p.product_id = '" . (int)$product_id . "' AND pd.language_id = '" .
(int)$this->config->get('config_language_id') . "' AND p.status = '1' AND
p.date_available <= NOW() AND p2s.store_id = '" . (int)$this->config-
>get('config_store_id') . "'");

    if ($query->num_rows) {
       return array(
          'product_id'       => $query->row['product_id'],
          'name'             => $query->row['name'],
          'description'      => $query->row['description'],
          'meta_title'       => $query->row['meta_title'],
          'meta_description' => $query->row['meta_description'],
          'meta_keyword'     => $query->row['meta_keyword'],
          'tag'              => $query->row['tag'],
          'model'            => $query->row['model'],
          'sku'              => $query->row['sku'],
          'upc'              => $query->row['upc'],
          'ean'              => $query->row['ean'],
          'jan'              => $query->row['jan'],
          'isbn'             => $query->row['isbn'],
          'mpn'              => $query->row['mpn'],
          'location'         => $query->row['location'],
          'quantity'         => $query->row['quantity'],
          'stock_status'     => $query->row['stock_status'],
          'image'            => $query->row['image'],
          'manufacturer_id'  => $query->row['manufacturer_id'],
          'manufacturer'     => $query->row['manufacturer'],
          'price'            => ($query->row['discount'] ? $query-
>row['discount'] : $query->row['price']),
          'special'          => $query->row['special'],
          'reward'           => $query->row['reward'],
          'points'           => $query->row['points'],
          'tax_class_id'     => $query->row['tax_class_id'],
          'date_available'   => $query->row['date_available'],
          'weight'           => $query->row['weight'],
          'weight_class_id'  => $query->row['weight_class_id'],
          'length'           => $query->row['length'],
          'width'            => $query->row['width'],
          'height'           => $query->row['height'],
          'length_class_id'  => $query->row['length_class_id'],
          'subtract'         => $query->row['subtract'],
          'rating'           => round($query->row['rating']),
          'reviews'          => $query->row['reviews'] ? $query-
>row['reviews'] : 0,
          'minimum'          => $query->row['minimum'],
          'sort_order'       => $query->row['sort_order'],
          'status'           => $query->row['status'],
          'date_added'       => $query->row['date_added'],
          'date_modified'    => $query->row['date_modified'],
          'viewed'           => $query->row['viewed']
       );
    } else {
       return false;
    }
}
```

Стандартна функція для отримання товарів, використовує багато залежності, які зроблені для отриманні інформації від якої залежить формування продукту, але щоб використати цю функцію в результаті буде сформована сторінка продукту. А для API, ця інформація є зайвою, тому для архітектурного рішення було прийнято розбити отримання даних на різні запити.

До прикладу оптимізації нижче приведені рис. 3.6 і 3.7, де на рисунку 3.6 показна продуктивність в хвилинах, від запиту в якому використовується написаний оптимізований запит до бази даних і на рис. 3.7 зроблений такий ж самий запит, але з використанням основної функції для отримання продуктів в системі.

Також написаний запит для API має ліміти і пагінацію для того, щоб не заламати структуру та працюючий MySQL на сервері. Так як, вигризка великої кількості продуктів може, зупинити роботу системи.

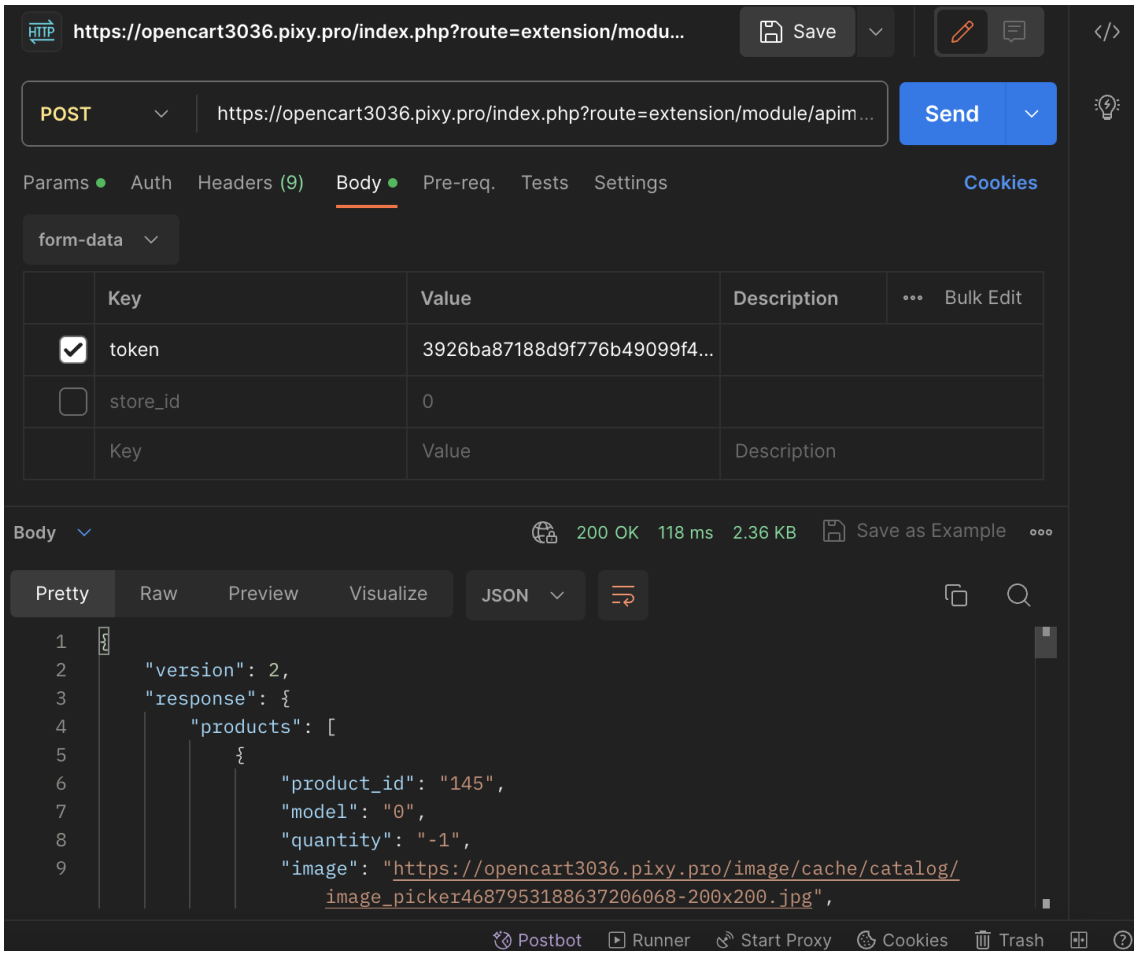


Рис. 3.6. Запит до API з використанням написаного запиту до бази даних

Якщо зробити висновок з цих двох рисунків, то можна виділити, що запит по API з використанням написаної функції зайняв по часу 118 мс, а запит з використанням основної функції зайняв 10.17 мс. Хоча об’єм даних був однаковий 2.36 кб.

Тому спосіб розбивки запитів до сервера є більш надійним рішенням чим використовування все написаних системою.

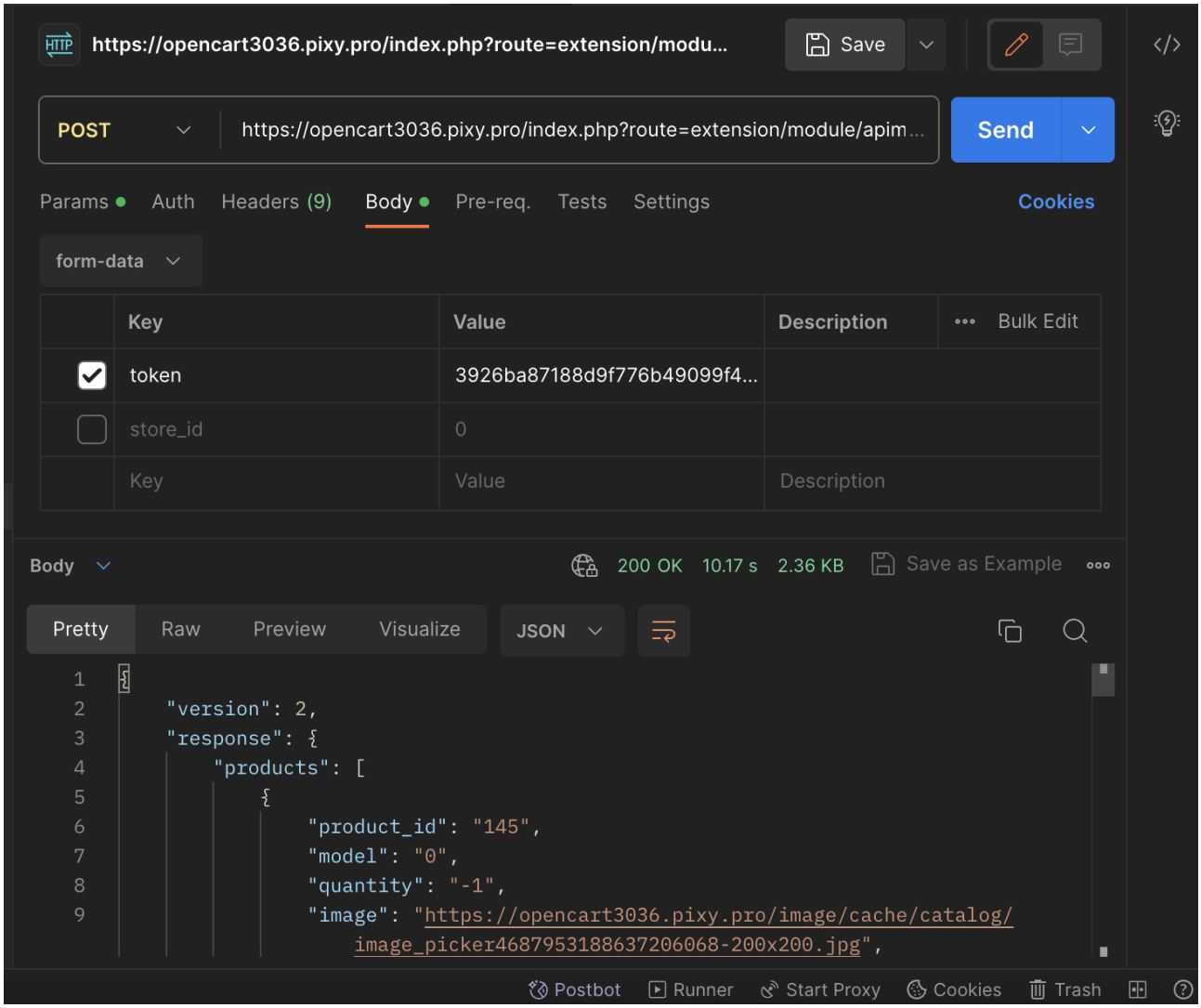


Рис. 3.7. Запит до API з використанням основної функції системи

3.3. Інтерфейс мобільного додатку

Для того, щоб взаємодіяти з API було створено мобільний застосунок, в якому поєднуються зручний інтерфейс і швидка інтеграція.

На початку розробки було створено інтеграцію з FireBase для того, щоб передавати ідентифікатор пристрою по якому будуть приходити пуш повідомлення та зроблена система авторизації користувача. Доступ до сайт буде знати тільки адміністратор і він же зможе зайти через додаток на нього. Щоб додати свій сайт у додаток треба ввести посилання на свій сайт, ввести логін та пароль до адмін-панелі.

Після того, як данні будуть збережені в додатку, буде зроблений запит на сервер і якщо данні введені не вірно, то користувач побачить повідомлення про це. А якщо все підходить, то його переведе на головну сторінку всіх доданих їм сайтів. На рис. 3.8 показано головну сторінку додатку, а на рис. 3.9 показано сторінку з додаванням сайту.

Рис. 3.8 Головна сторінка додатку

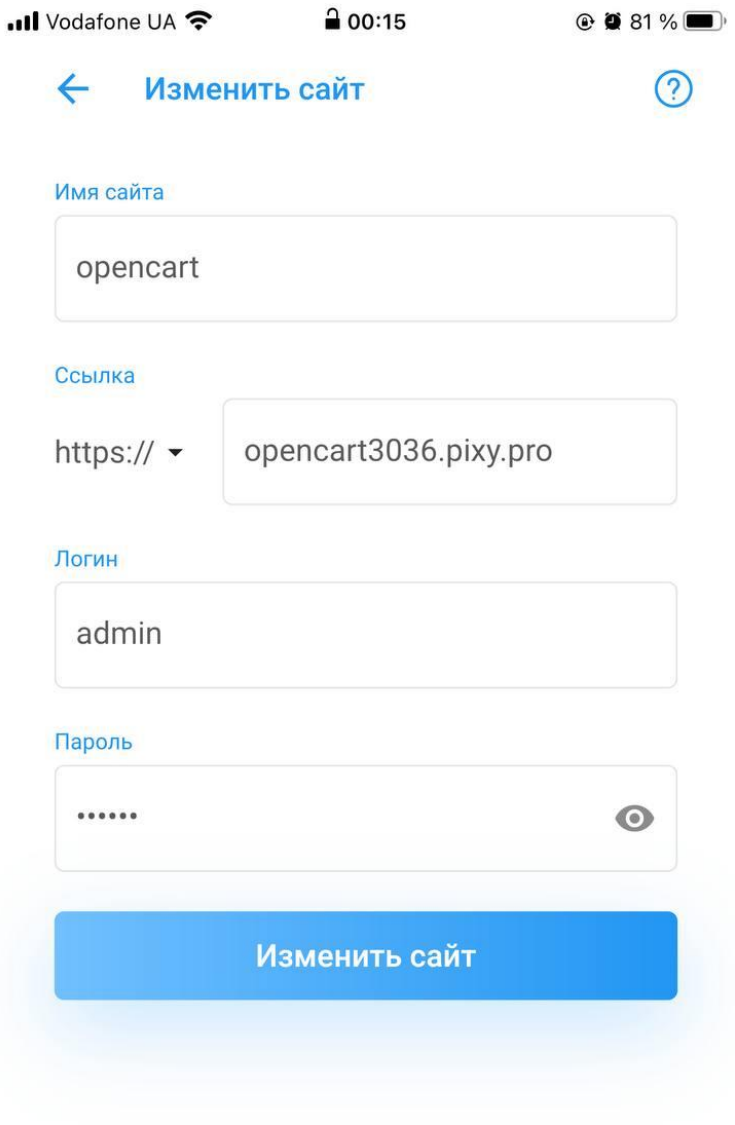


Рис. 3.8. Сторінка додавання сайту у проєкт

Коли користувач до дасть свій сайт він може перейти на нього і побачить першу сторінку проєкту, на якій буде показано статистика сайту. Тобто це кількість товарів, сума заробітку і кількість клієнтів. Також цю інформацію можна відфільтрувати за день, за тиждень, за місяць і за рік. Детально перша сторінка показана на рис. 3.10.

Рис. 3.10. Сторінка статистики сайту

Так як, користувач зможе приймати пуш повідомлення, при реєстрації сайту в додатку завдяки бібліотеці Fire Base генерується токен девайсу і зберігається в базі даних.

Коли клієнти на сайті зроблять замовлення або зареєструються, в такому випадку завдяки функції плагіну відпрацює івент і на мобільному пристрої можна буде побачити повідомлення. Приклад повідомлення показаний на рис. 3.11.

Ця функція є дуже важливою складовою додатку, так як це зможе допомагати бізнесу швидко реагувати на замовлення і мати зворотній зв’язок з своїми покупцями.

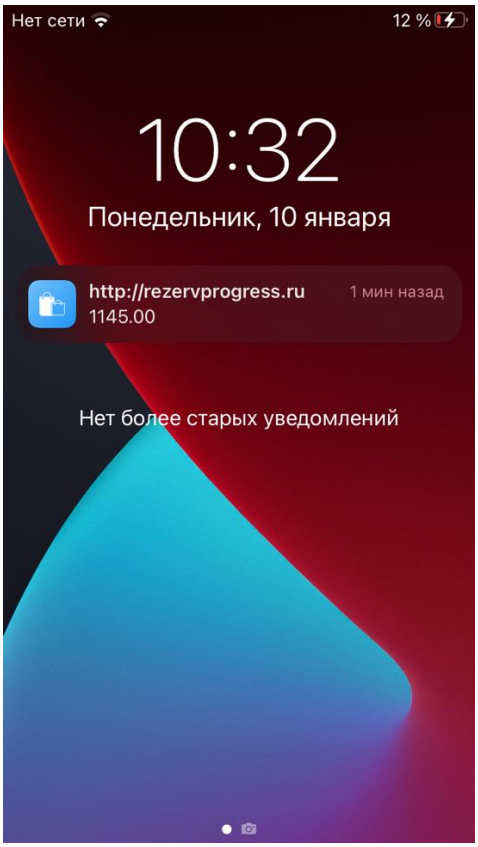


Рис. 3.11. Надходження пуш повідомлення на мобільний пристрій

Для зручності в інтерфейсі є вкладка для того, щоб відстежувати замовлення. На вкладці замовлення, користувач може переглядати всі замовлення які були і переглядати звіт замовлення, який товар замовив клієнт, також міняти статус замовлення і виконувати всі ті функції які є в адмін-панелі сайту. Приклад того, як виглядає сторінка зі списком замовлення показана на рис. 3.12, а приклад історії замовлення показано на рис. 3.13.

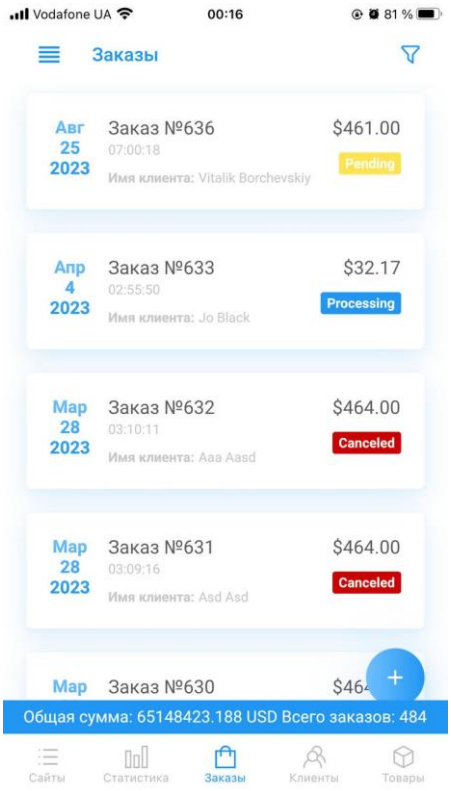


Рис. 3.12. Сторінка замовлень

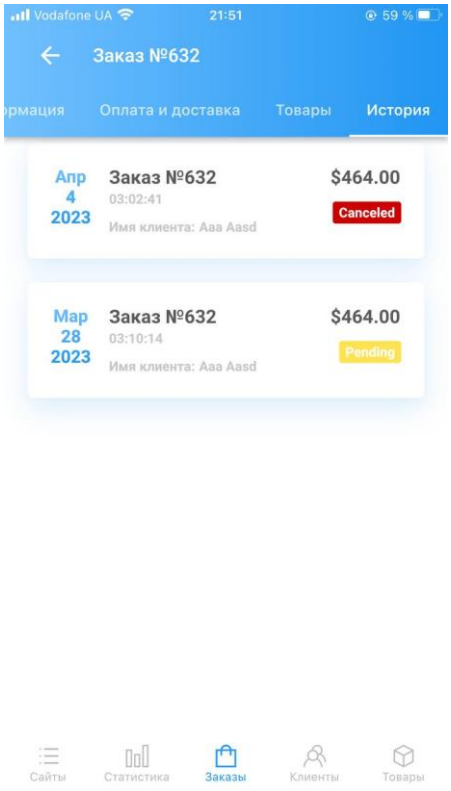


Рис. 3.13. Сторінка історії замовлень

Далі користувач може бачити сторінку клієнти, на ній можна подивитися список всіх клієнтів які є в системі. Якщо є потрібним то змінити інформацію про користувача. А також в додатковому функціоналі, якщо перейти на користувача в вкладці історія відображається вся інформація про його замовлення які він робив.

На рис. 3.14. показана сторінка з виведеними всіма користувачами системи, а на рис. 3.15 показана історія користувача.

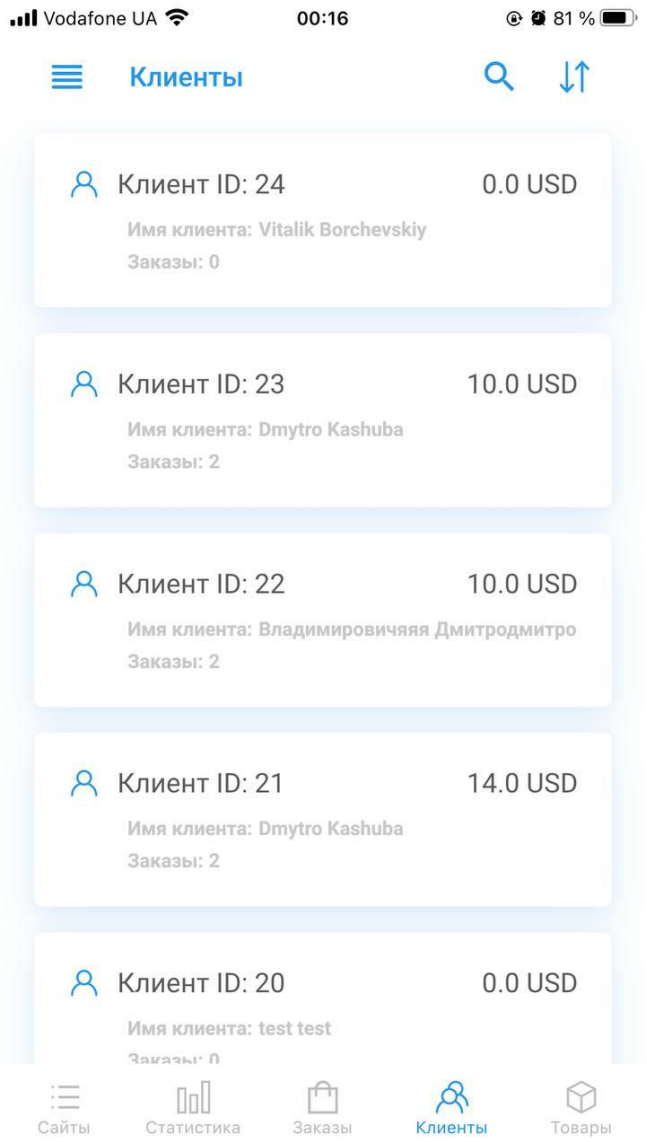


Рис. 3.14. Вкладка клієнти

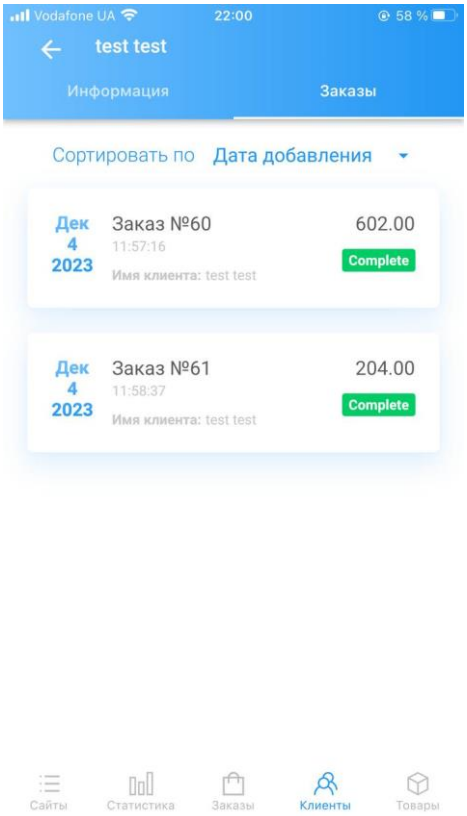


Рис. 3.15. Владка історія користувача

Якщо перейти на сторінку товари, користувач може побачити всі товари своїй системи, так як їх може бути велика кількість в додатку використовується пагінація, щоб не зупиняти роботу користувача і не зламати MySQL на сервері. Тому, що велика кількість даних може призупинити роботу баз даних і із-за цього сама система і сайт припинять роботу.

На сторінці товари окрім того, що можна переглядати товари, ще користувач може їх змінювати і додавати нові. В системі є функція з додаванням опцій на продукт, але поки що додаток не має можливості такого функціоналу і це планується на розвиток. Окрім цього, основний функціонал додавання продукту збережений і створити продукт з основними важливими параметрами можливо. Приклад сторінки товарів і додавання товарів приведений на рис. 3.16 і рис. 3.17. Приклад сторінки редагування товару приведений на рис. 3.18.

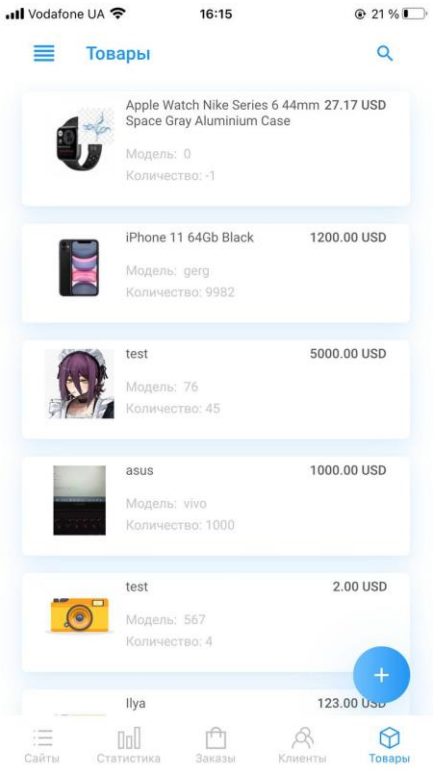


Рис. 3.16. Сторінка товарів

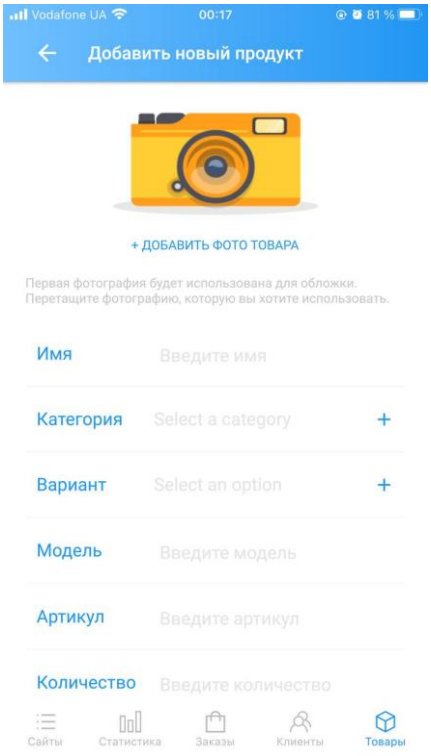


Рис. 3.17. Сторінка додавання товару

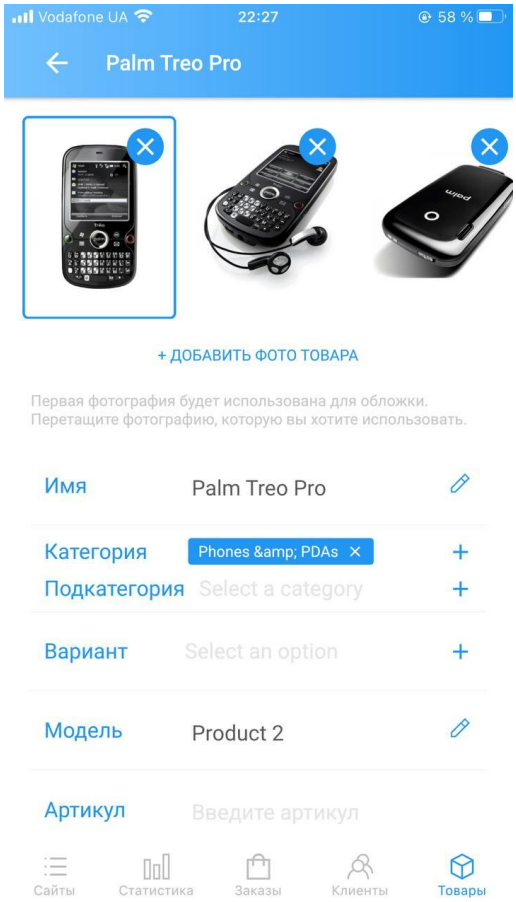


Рис. 3.18. Сторінка редагування товару

До додаткової функції на сторінці замовлень також можна зробити редагування замовлення, та додавання нового. Але при додаванні нового не всі поля замовлення присутні, по прикладу з адмін-панелі не було допрацьовано додавання на нове замовлення способів доставки та оплати, це планується додати при оновлені.

При цьому основні функції присутні і додати можна замовлення, а якщо потрібно добавити потрібну інформацію, потрібно заходити на сайт та правити ці дані.

Приклад вигляду сторінок редагування і додавання показані на рис. 3.19 та 3.20.

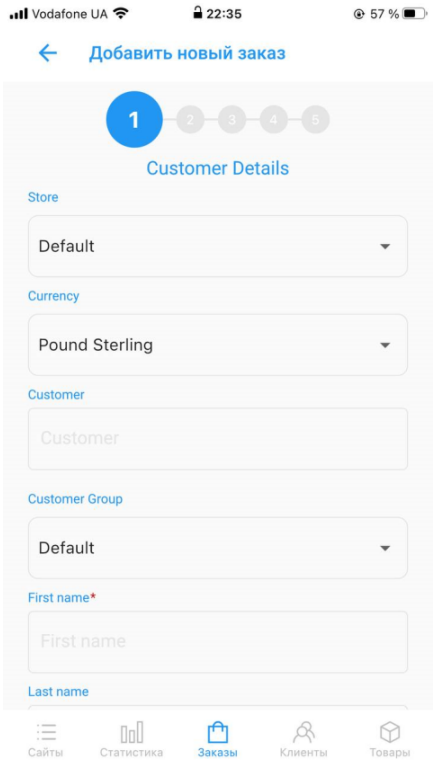


Рис. 3.19. Сторінка редагування замовлення

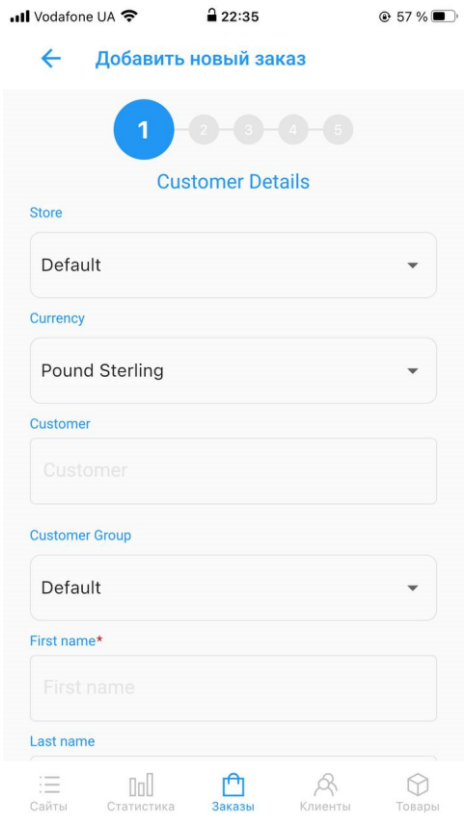


Рис. 3.20. Сторінка додавання замовлення

Також додаток, має ще одну сторінку яка показується в боковому меню і вона називається відгуки приклад показано на рис 3.21. На цій сторінці користувач може побачити коментарі, які клієнти пишуть на товарах. Так, як всі відгуки проходять модерацію користувач може через мобільний додаток позначати коментарі які можна відобразити, а які ні. Сторінку зі списком коментарів можна побачити на рис. 3.22, а приклад відгуку з його редагуванням можна побачити на рис. 3.23.

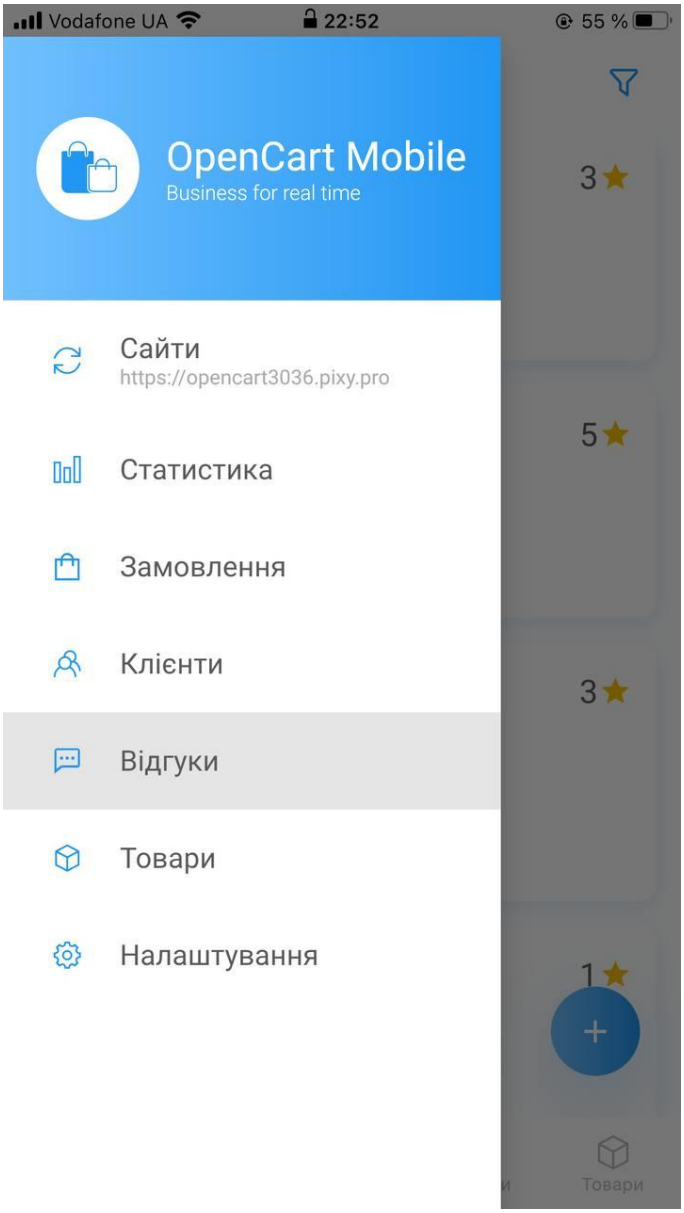


Рис. 3.21 Бокове вікно меню додатку

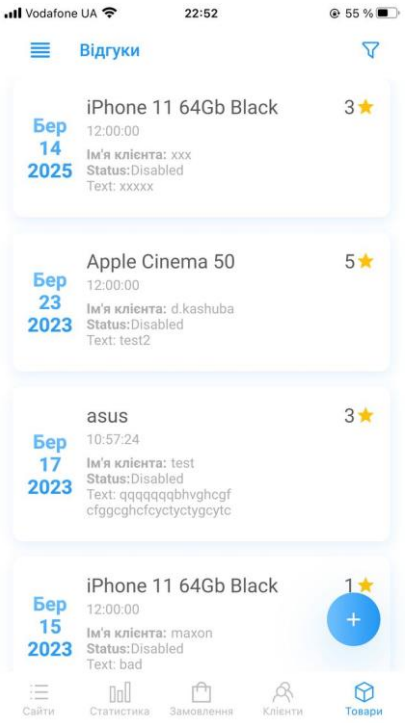


Рис. 3.22. Сторінка з відгуками

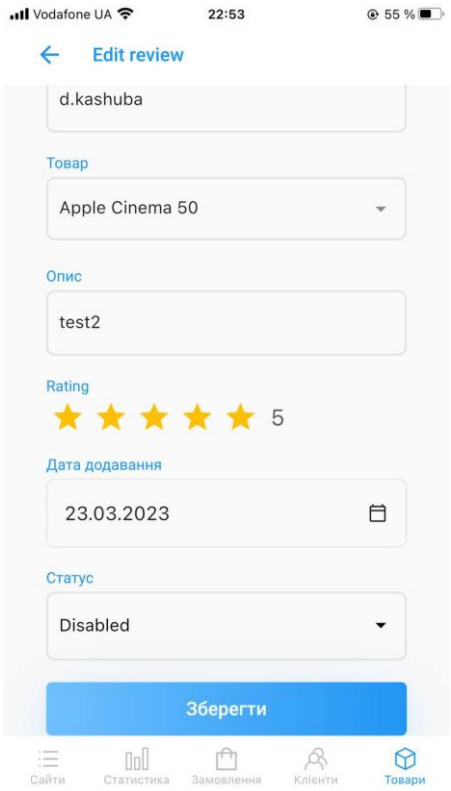


Рис. 3.23. Сторінка редагування відгука

В зробленому мобільному додатку є майже всі основні функції котрі є і в адмін панелі. Але для його розширення множна враховувати ті зазначені вище функції, які потрібні для створення ордерів і продуктів.

3.3. Висновки до третього розділу

В цьому розділі на практиці і теоретично були досліджено і розроблено плагін для Opencart (API) і мобільний додаток, а також було проведено оптимізацію запитів до бази даних. Була взята до уваги мала швидкість відповіді запита від основної функції системи і зроблений висновок розбити на під запити, які написані власноруч.

За проведеним аналізом можна зробити висновки, що інтеграція між системою і мобільним додатком була успішна, і може похвалитися вищою продуктивністю роботи і використаними функціями.

Крім цього, у процесі проведення розробки були додані додаткові функції, які можуть допомогти бізнесу і покращити взаємодію клієнта і замовника завдяки швидкому реагуванню на замовлення через сповіщення клієнта пуш повідомленням.

Даний мобільний додаток має 4 основних сторінки які показують користувачу основну інформацію по його системі. Також додаток має функції на редагування та додавання товарів і замовлень. Всі запити і додавання сайтів мають захищений вид взаємодії через токени, які за часом можуть змінюватися і це надає можливість захистити користувача від втраті своїх особливих даних.

Створений додаток можна використовувати навіть у великих зроблених системах, коли Opencart буде має велику кількість складів по всій країні і працівники в системі завдяки повідомленням про замовлення можуть швидко реагувати, та надавати відповідь замовнику про час отримання товару.

На підставі проведеного дослідження та розробки можна визначити, що Flutter виявляється ефективним інструментом для розробки високодинамічних та ефективних мобільних додатків. За використанням Dart та Flutter можна

створити масштабовані, швидкі та ефективні мобільні додатки, які працюють на різних платформах, таких як iOS та Android. Такий підхід може бути особливо корисним для розробки програмного забезпечення на замовлення бізнесу, де важлива продуктивність, узгодженість та масштабованість.

ВИСНОВОК

Підсумковий висновок кваліфікаційної роботи полягає в створенні та розгортанні інформаційної системи для оптимізації взаємодії мобільного застосунку з RESTful API в контексті управління вмістом CMS системи Opencart.

В даному дослідженні було розроблено структуровану архітектуру для ефективної взаємодії мобільного застосунку із RESTful API при використанні CMS системи Opencart. Клієнтська частина застосунку дозволяє здійснювати різноманітні операції з контентом, надаючи користувачам зручний інтерфейс без необхідності втручання в код. Застосування асинхронного програмування дозволяє відкладати виконання змін за розкладом або обраним інтервалом, забезпечуючи оперативність та безпеку операцій.

Адміністративна частина розроблена з урахуванням простоти та інтуїтивної зрозумілості, не вимагаючи від користувачів спеціальної підготовки для управління ролями та статусами. Використання бази даних з підтримкою мови SQL забезпечує високу швидкодію системи.

Розроблений мобільний застосунок націлений на підвищення продуктивності та швидкодії, а також на оптимізацію сценаріїв повного тестування інформаційної системи, що включає в себе взаємодію з RESTful API у роботі з CMS системою Opencart. Отримані результати дослідження можуть знайти широке застосування як у сфері розробки програмного забезпечення, де поліпшення продуктивності є важливим фактором, так і в освітньому процесі, сприяючи розвитку галузі та оптимізації бізнес-процесів.

У процесі створення застосунку використовувалися передові технології програмування, зокрема PHP, MySQL, HTML5, CSS3, Flutter, Dart, JavaScript та Bootstrap. Цей проект може стати корисним для корпоративних систем, які використовують CMS систему Opencart, сприяючи подальшому розвитку електронної комерції та оптимізації бізнес-процесів в контексті взаємодії з мобільними пристроями через RESTful API.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Amundsen, M. (2016). RESTful Web Clients: Enabling Reuse Through Hypermedia. O'Reilly Media.

2. Balter, B. (2014). The REST API Design Handbook. Leanpub.

3. Bhasin, J. (2016). Rails API: Building a RESTful API in Ruby on Rails. Apress.

4. Chauhan, S. (2018). Learning Opencart 3 Theme Development. Packt Publishing Ltd.

5. Chauhan, V. (2012). Opencart 1.5 Beginner's Guide. Packt Publishing Ltd.

6. Chauhan, V. (2013). RESTful Rails Development: Building Open Service-Oriented Applications and Services. Packt Publishing Ltd.

7. Choudhury, M. (2013). Rails 4 Application Development: Hotshot. Packt Publishing Ltd.

8. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Електронний ресурс – Режим доступу: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm.

9. Fielding, R. T. (2008). RESTful Web Services: Principles, Patterns, Emerging Technologies. O'Reilly Media. Електронний ресурс – Режим доступу: https://rest.elkstein.org/.

10. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. Електронний ресурс – Режим доступу: https://www.researchgate.net/publication/338514531_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks .

11. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. Електронний ресурс – Режим доступу: https://arxiv.org/abs/1312.6114.

12. Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. Електронний ресурс – Режим доступу: https://arxiv.org/abs/1906.02691.

13. Li, J. (2012). Opencart 1.5 User Manual. Packt Publishing Ltd.

14. Li, J. (2018). Learning Opencart 3: Theme and Module Development. Apress.

15. Masse, M., & Hohpe, G. (2004). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.

16. Ojala, T., & Suomalainen, T. (2019). RESTful API Design. O'Reilly Media. Електронний ресурс – Режим доступу: https://www.oreilly.com/library/view/restful-api-design/9781449337934/.

17. Ramalho, L. (2015). Fluent Python. O'Reilly Media.

18. Richardson, L., & Amundsen, M. (2013). RESTful Web APIs: Services for a Changing World. O'Reilly Media.

19. Roy Fielding's Dissertation (2000). Architectural Styles and the Design of Network-based Software Architectures. Електронний ресурс – Режим доступу: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm.

20. Stanishev, S. (2020). API Design for Beginners: A Pragmatic Approach. Apress.

21. Vukovic, V. (2021). RESTful Web Services with Spring Boot: A Guide to Implementing RESTful APIs using Spring Boot. Apress.

22. Wang, Z., Liu, R., & Xu, J. (2020). Hands-On RESTful API Design Patterns and Best Practices: A Practical Guide to Streamlining Your REST Services and Making Them Future-Proof. Packt Publishing Ltd.

23. White, G. (2017). RESTful Web Services in Action. Manning Publications.

24. Yegge, S. (2008). Good Agile, Bad Agile. Електронний ресурс – Режим доступу: https://sites.google.com/site/steveyegge2/they-are-not-you.

25. Zhang, W. (2019). Pro RESTful APIs: Design, Build, and Integrate with REST, JSON, XML, and JAX-RS. Apress.

26. Ziegler, J., & Lorch, M. (2013). REST: From Research to Practice. Springer.

27. Zhu, J. (2013). Spring REST. O'Reilly Media.

28. Zoltners, A. A., Sinha, P., & Lorimer, S. E. (2005). Building a Winning Sales Force: Powerful Strategies for Driving High Performance. AMACOM Div American Mgmt Assn.

29. Zomaya, A. Y. (1999). Parallel and Distributed Computing Handbook. McGraw-Hill.

30. Zongker, D. (2018). Django for APIs: Build web APIs with Python & Django. William S. Vincent.

31. Zuill, W. (2013). No Estimates: How to Measure Project Progress Without Estimating. Електронний ресурс – Режим доступу: https://leankit.com/blog/2013/01/noestimates-part1/.

32. Zygmuntowicz, D. (2008). Deploying Rails Applications: A Step-by-Step Guide. Pragmatic Bookshelf.

33. Reynolds, A., & Breddels, M. A. (2005). Mind the Gap: Towards a Unified Web Query Language. In Web Information Systems – WISE 2005 (pp. 54–67). Springer.

34. Kerremans, M. (2022). The One-Person Business: Great Work from the Comfort of Your Home. Chronicle Books.

35. Wilson, G. (2022). Seven Steps to Mastering Business Analysis. Sage Publications.

36. Karras, T., Laine, S., & Aila, T. (2020). Training Generative Adversarial Networks with Limited Data. Електронний ресурс – Режим доступу: https://arxiv.org/abs/2006.06676.

37. Google. (2015). Inceptionism: Going Deeper into Neural Networks. Електронний ресурс – Режим доступу: https://blog.research.google/2015/06/inceptionism-going-deeper-into-neural.html.

ЛІСТИНГ ПРОГРАМИ

Код основного контролера в плагіні API

```
<?php

class ControllerExtensionModuleApimodule extends Controller {
    private $API_VERSION = 2;
    private $PUSH_ACTIONS = [
        0 => 'new_order',
        1 => 'new_customer',
        2 => 'new_review',
        3 => 'order_status_changed'
    ];

    public function __construct($registry) {
        parent::__construct($registry);
        $this->load->model('extension/module/apimodule');
    }

    public function getVersion() {
        return $this->API_VERSION;
    }

    // catalog/controller/module/apimodule/*/before
    public function routerIndex(&$route, &$args) {
        $route = 'extension/' . $route;
    }

    // catalog/model/checkout/order/addOrderHistory/after
    public function afterAoh(&$route, &$args, &$output) {
        if (!empty($args) && (count($args) >= 2)) {
            if (count($this->model_extension_module_apimodule-
>getOrderHistory($args[0])) == 1) {
                $this->sendNotifications([
                    'actionId' => 0,
                    'orderId' => $args[0],
                    'orderStatusId' => $args[1],
                ]);
            } else {
                $this->sendNotifications([
                    'actionId' => 3,
                    'orderId' => $args[0],
                    'orderStatusId' => $args[1],
                ]);
            }
        }
    }

    // catalog/model/account/customer/addCustomer/after
    public function afterAddCustomer(&$route, &$args, &$output) {
        if (!empty($output)) {
            $this->sendNotifications([
                'actionId' => 1,
                'customerId' => (int) $output,
            ]);
        }
```

```
    }

    // catalog/model/catalog/review/addReview/after
    public function afterAddReview(&$route, &$args, &$output) {
        $this->sendNotifications([
            'actionId' => 2,
            'productId' => (int) $args[0], // $args[1] это массив с
информацией о ревью, может ее тоже передавать?
        ]);
    }

    /**
     * @api {post} index.php?route=module/apimodule/orders  Orders List
     * @apiName  GetOrders
     * @apiDescription  List of user orders
     * @apiGroup Order
     *
     * @apiParam {Token}     token your unique token.
     * @apiParam {Number}    page number of the page.
     * @apiParam {Number}    limit limit of the orders for the page.
     * @apiParam {Array[]}   filter Array of the filters.
     * @apiParam {String}    filter.fio full name of the client.
     * @apiParam {Number}    filter.order_status_id unique id of the
order.
     * @apiParam {Number}    filter.min_price min price of order.
     * @apiParam {Number}    filter.max_price max price of order.
     * @apiParam {Date}      filter.date_min min date adding of the order.
     * @apiParam {Date}      filter.date_max max date adding of the order.
     *
     * @apiSuccess {Number}  version                          Current API
version.
     * @apiSuccess {Bool}    status                           Response
status.
     * @apiSuccess {Array[]} response                         Array with
content response.
     * @apiSuccess {String}  response.total_quantity          Total
quantity of the orders.
     * @apiSuccess {String}  response.currency_code           Default
currency of the shop.
     * @apiSuccess {Number}  response.total_sum               Total amount
of orders.
     * @apiSuccess {String}  response.max_price               Maximum
order amount.
     * @apiSuccess {Number}  response.api_version             Current API
version.
     *
     * @apiSuccess {Array[]} response.orders                  Array of the
orders.
     * @apiSuccess {Array[]} response.statuses                Array of the
order statuses.

     * @apiSuccess {String} response.orders.order_id          ID of the
order.
     * @apiSuccess {String} response.orders.order_number      Number of
the order.
     * @apiSuccess {String} response.orders.fio               Client's
FIO.
     * @apiSuccess {String} response.orders.status            Status of
the order.
     * @apiSuccess {String} response.orders.total             Total sum of
the order.
     * @apiSuccess {String} response.orders.date_added        Date added
of the order.
     * @apiSuccess {String} response.orders.currency_code     Currency of
```

```
the order.
     *
     * @apiSuccess {String} response.statuses.name            Status Name.
     * @apiSuccess {String} response.statuses.order_status_id Status id.
     * @apiSuccess {String} response.statuses.language_id     Language id.
     *
     *
     *
     * @apiSuccessExample Success-Response:
     *     HTTP/1.1 200 OK {
     *     "Response" {
     *        "orders": {
     *            {
     *             "order_id" : "1",
     *             "order_number" : "1",
     *             "fio" : "Anton Kiselev",
     *             "status" : "Сделка завершена",
     *             "total" : "106.00",
     *             "date_added" : "2016-12-09 16:17:02",
     *             "currency_code": "RUB"
     *             },
     *            {
     *             "order_id" : "2",
     *             "order_number" : "2",
     *             "fio" : "Vlad Kochergin",
     *             "status" : "В обработке",
     *             "total" : "506.00",
     *             "date_added" : "2016-10-19 16:00:00",
     *             "currency_code": "RUB"
     *             }
     *         },
     *         "statuses" : {
     *              {
     *               "name": "Отменено",
     *               "order_status_id": "7",
     *               "language_id": "1"
     *               },
     *              {
     *               "name": "Сделка завершена",
     *               "order_status_id": "5",
     *               "language_id": "1"
     *               },
     *               {
     *                "name": "Ожидание",
     *                "order_status_id": "1",
     *                "language_id": "1"
     *                }
     *       },
     *       "currency_code": "RUB",
     *       "total_quantity": 50,
     *       "total_sum": 2026.00,
     *       "max_price": "1405.00"
     *   },
     *   "Status" : true,
     *   "version": 2.0
     * }
     * @apiErrorExample Error-Response: {
     *      "version": 2.0,
     *      "Status" : false
     * }
     *
     *
     */
    public function orders()
```

```
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {

            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION, 'error' => $error, 'status' => false]));
            return;
        }

        if (isset($_REQUEST['page']) && (int)$_REQUEST['page'] != 0 &&
isset($_REQUEST['limit']) && (int)$_REQUEST['limit'] != 0) {
            $page = ($_REQUEST['page'] - 1) * $_REQUEST['limit'];
            $limit = $_REQUEST['limit'];
        } else {
            $page = 0;
            $limit = 9999;
        }

        $this->load->model('extension/module/apimodule');

//        var_export($_REQUEST['filter']);
//        die();
        if (isset($_REQUEST['filter'])) {
            $orders = $this->model_extension_module_apimodule-
>getOrders(['filter' => $_REQUEST['filter'], 'page' => $page, 'limit' =>
$limit]);
        } elseif (isset($_REQUEST['platform']) && $_REQUEST['platform'] ==
'android') {
            $filter = [];
            if (isset($_REQUEST['order_status_id'])) {
                $filter['order_status_id'] = $_REQUEST['order_status_id'];
            }
            if (isset($_REQUEST['fio'])) {
                $filter['fio'] = $_REQUEST['fio'];
            }
            if (isset($_REQUEST['min_price']) && $_REQUEST['min_price'] !=
0) {
                $filter['min_price'] = $_REQUEST['min_price'];
            } else {
                $filter['min_price'] = 1;
            }
            if (isset($_REQUEST['max_price'])) {
                $filter['max_price'] = $_REQUEST['max_price'];
            } else {
                $filter['max_price'] = $this-
>model_extension_module_apimodule->getMaxOrderPrice();
            }

            $filter['date_min'] = $_REQUEST['date_min'];
            $filter['date_max'] = $_REQUEST['date_max'];

            $orders = $this->model_extension_module_apimodule-
>getOrders(['filter' => $filter, 'page' => $page, 'limit' => $limit]);

        } else {
            $orders = $this->model_extension_module_apimodule-
>getOrders(['page' => $page, 'limit' => $limit]);
        }
        $response = [];
        $orders_to_response = [];
```

```
        $currency = $this->model_extension_module_apimodule-
>getUserCurrency();
        if (empty($currency)) {
            $currency = $this->model_extension_module_apimodule-
>getDefaultCurrency();
        }
        $response['currency_code'] = $currency;

        foreach ($orders->rows as $order) {
            $data['order_number'] = $order['order_id'];
            $data['order_id'] = $order['order_id'];
            if (isset($order['firstname']) && isset($order['lastname'])) {
                $data['fio'] = $order['firstname'] . ' ' .
$order['lastname'];
            } else {
                $data['fio'] = $order['payment_firstname'] . ' ' .
$order['payment_lastname'];
            }
            $data['status'] = $order['name'];

            $data['total'] = $this->currency->format($order['total'],
$order['currency_code'], $order['currency_value']);
            $data['date_added'] = $order['date_added'];
            $data['currency_code'] = $order['currency_code'];
            $orders_to_response[] = $data;
        }

        $response['total_quantity'] = $orders->quantity;
        $response['currency_code'] = $currency;
        $response['total_sum'] = $this->calculatePrice($orders->totalsumm,
$currency);
        //$response['total_sum'] = number_format($orders->totalsumm, 2,
'.', '');
        $response['orders'] = $orders_to_response;
        $response['max_price'] = $this->model_extension_module_apimodule-
>getMaxOrderPrice();
        $statuses = $this->model_extension_module_apimodule-
>OrderStatusList();
        $response['statuses'] = $statuses;
        $response['api_version'] = $this->API_VERSION;

        $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION, 'response' => $response, 'status' => true]));
        return;
    }


    /**
     * @return void
     */
    public function getReview()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }
```

```
        $product_id = $_REQUEST['product_id'];

        $this->load->model('extension/module/apimodule');
        $reviews = $this->model_extension_module_apimodule-
>getReviewComment($product_id);

        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $reviews
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getOption()
    {

        ini_set('display_errors', 1); ini_set('display_startup_errors',
1); error_reporting(E_ALL);

        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $value = $_REQUEST['value'];

        if(isset($option_id)) {
            $option_id = $_REQUEST['option_value_id'];
        } else {
//            $option_id = '';
        }

        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule-
>getOption($value,$option_id = null);

        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
```

```
    public function getCustomerGroup()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule-
>getCustomerGroups();


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getStore()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule->getStores();


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }
```

```
    /**
     * @return void
     */
    public function getCountry()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule->getCountrys();


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function setZone()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $id_country = $_REQUEST['country_id'];

        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule-
>getZone($id_country);


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
```

```
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getClient()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }


        $client_info = $_REQUEST['store_id'];


        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule-
>getClientStore($client_info);


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getCurrency()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }
```

```
        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule->getCurency();


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getProducts()
    {


        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }


        $client_info = $_REQUEST['store_id'];


        $this->load->model('extension/module/apimodule');
        $option = $this->model_extension_module_apimodule-
>getProd($client_info);


        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $option
                ]
            ]
        ));
    }


    /**
     * @return void
     */
    public function getMethodPayment()
    {

        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');
```

```
        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }


        $this->load->language('checkout/checkout');

        // Totals
        $totals = array();
        $taxes = $this->cart->getTaxes();
        $total = 0;

        // Because __call can not keep var references so we put them into
an array.
        $total_data = array(
            'totals' => &$totals,
            'taxes'  => &$taxes,
            'total'  => &$total
        );

        $this->load->model('setting/extension');

        $sort_order = array();

        $results = $this->model_setting_extension->getExtensions('total');

        foreach ($results as $key => $value) {
            $sort_order[$key] = $this->config->get('total_' .
$value['code'] . '_sort_order');
        }

        array_multisort($sort_order, SORT_ASC, $results);


        foreach ($results as $result) {
            if ($this->config->get('total_' . $result['code'] .
'_status')) {
                $this->load->model('extension/total/' . $result['code']);

                // We have to put the totals in an array so that they pass
by reference.
                $this->{'model_extension_total_' . $result['code']}-
>getTotal($total_data);
            }
        }

        // Payment Methods
        $method_data = array();

        $this->load->model('setting/extension');

        $results = $this->model_setting_extension-
>getExtensions('payment');

        $recurring = $this->cart->hasRecurringProducts();

        foreach ($results as $result) {
```

```
            $this->load->model('extension/payment/' . $result['code']);
            $address = array();

            $method = $this->{'model_extension_payment_' .
$result['code']}->getMethod($address, $total);

            if ($method) {

                $method_data[] = $method;
            }
        }


        $sort_order = array();

        foreach ($method_data as $key => $value) {

            $sort_order[$key] = $value['sort_order'];
        }

        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $method_data
                ]
            ]
        ));

    }


    /**
     * @return void
     */
    public function getMethodShipping()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $this->load->language('checkout/checkout');


        // Shipping Methods
        $method_data = array();

        $this->load->model('setting/extension');

        $results = $this->model_setting_extension-
>getExtensions('shipping');

        foreach ($results as $result) {
```

```
                $this->load->model('extension/shipping/' .
$result['code']);

                $quote = $this->{'model_extension_shipping_' .
$result['code']}->getQuote($this->session->data['shipping_address']);




                    $method_data[] = $quote['quote']['flat'];


            }

//        var_dump($method_data);
//        die();



        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>
                     $method_data

            ]
        ));
    }


    /**
     * @return void
     */
    public function addReview()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }


        $data = $_REQUEST['data'];


        $this->load->model('extension/module/apimodule');
        $new_review = $this->model_extension_module_apimodule-
>addReview($data);




        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>
```

```
                    $new_review

            ]
        ));
    }


    /**
     * @return void
     */
    public function editReview()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }


        $data = $_REQUEST['data'];
        $review_id = $_REQUEST['review_id'];



        $this->load->model('extension/module/apimodule');
        $new_review = $this->model_extension_module_apimodule-
>editReview($review_id,$data);
       // $this->model_catalog_review->editReview($this->request-
>get['review_id'], $this->request->post);




        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>
                    $new_review

            ]
        ));
    }


    /**
     * @return void
     */
    public function newOrder()
    {

        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');

        $this->load->model('checkout/order');

        $order_data = array();
        $totals = array();
```

```
        $order_data['invoice_prefix'] = $this->config-
>get('config_invoice_prefix');
        $order_data['store_id'] = $this->config->get('config_store_id');
        $order_data['store_name'] = $this->config->get('config_name');

        if ($order_data['store_id']) {
            $order_data['store_url'] = $this->config->get('config_url');
        } else {
            if ($this->request->server['HTTPS']) {
                $order_data['store_url'] = HTTPS_SERVER;
            } else {
                $order_data['store_url'] = HTTP_SERVER;
            }
        }

        $this->load->model('account/customer');

        $order_customer = $_REQUEST['client'];

        $order_customer_data = $_REQUEST['client'];



        if (isset($order_customer_data)) {
            $customer_info = $this->model_account_customer-
>getCustomer($order_customer['customer_id']);



            if(isset($customer_info)) {
                $order_data['customer_id'] =
$customer_info['customer_id'];
                $order_data['customer_group_id'] =
$customer_info['customer_group_id'];
                $order_data['firstname'] = $customer_info['firstname'];
                $order_data['lastname'] = $customer_info['lastname'];
                $order_data['email'] = $customer_info['email'];
                $order_data['telephone'] = $customer_info['telephone'];
                $order_data['custom_field'] =
json_decode($customer_info['custom_field'], true);
            } else {
                $order_data['customer_id'] =
$order_customer['customer_id'];
                $order_data['customer_group_id'] =
$order_customer['customer_group_id'];
                $order_data['firstname'] = $order_customer['firstname'];
                $order_data['lastname'] = $order_customer['lastname'];
                $order_data['email'] = $order_customer['email'];
                $order_data['telephone'] = $order_customer['telephone'];
                $order_data['custom_field'] =
json_decode($order_customer['custom_field'], true);
            }

        } elseif (isset($this->session->data['guest'])) {
            $order_data['customer_id'] = 0;
            $order_data['customer_group_id'] = $this->session-
>data['guest']['customer_group_id'];
            $order_data['firstname'] = $this->session-
>data['guest']['firstname'];
            $order_data['lastname'] = $this->session-
>data['guest']['lastname'];
            $order_data['email'] = $this->session->data['guest']['email'];
            $order_data['telephone'] = $this->session-
>data['guest']['telephone'];
```

```
            $order_data['custom_field'] = $this->session-
>data['guest']['custom_field'];
        }



        $payment_data = $_REQUEST['payment'];

        $order_data['payment_firstname'] =
$payment_data['payment_firstname'];
        $order_data['payment_lastname'] =
$payment_data['payment_lastname'];
        $order_data['payment_company'] = $payment_data['payment_company'];
        $order_data['payment_address_1'] =
$payment_data['payment_address_1'];
        $order_data['payment_address_2'] =
$payment_data['payment_address_2'];
        $order_data['payment_city'] = $payment_data['payment_address_2'];
        $order_data['payment_postcode'] =
$payment_data['payment_postcode'];
        $order_data['payment_zone'] = $payment_data['payment_zone'];
        $order_data['payment_zone_id'] = $payment_data['payment_zone_id'];
        $order_data['payment_country'] = $payment_data['payment_country'];
        $order_data['payment_country_id'] =
$payment_data['payment_country_id'];
        $order_data['payment_address_format'] =
$payment_data['payment_address_format'];
        $order_data['payment_custom_field'] = (isset($this->session-
>data['payment_address']['custom_field']) ? $this->session-
>data['payment_address']['custom_field'] : array());


        if (isset($payment_data['payment_method'])) {
            $order_data['payment_method'] =
$payment_data['payment_method'];
        } else {
            $order_data['payment_method'] = '';
        }

        if (isset($pyment_order['payment_code'])) {
            $order_data['payment_code'] = $pyment_order['payment_code'];
        } else {
            $order_data['payment_code'] = '';
        }

        $shipping_order = $_REQUEST['shipping'];



        if (isset($shipping_order)) {
            $order_data['shipping_firstname'] =
$shipping_order['shipping_firstname'];
            $order_data['shipping_lastname'] =
$shipping_order['shipping_lastname'];
            $order_data['shipping_company'] =
$shipping_order['shipping_company'];
            $order_data['shipping_address_1'] =
$shipping_order['shipping_address_1'];
            $order_data['shipping_address_2'] =
$shipping_order['shipping_address_2'];
            $order_data['shipping_city'] =
$shipping_order['shipping_city'];
            $order_data['shipping_postcode'] =
$shipping_order['shipping_postcode'];
```

```
            $order_data['shipping_zone'] =
$shipping_order['shipping_zone'];
            $order_data['shipping_zone_id'] =
$shipping_order['shipping_zone_id'];
            $order_data['shipping_country'] =
$shipping_order['shipping_country'];
            $order_data['shipping_country_id'] =
$shipping_order['shipping_country_id'];
            $order_data['shipping_address_format'] =
$shipping_order['shipping_address_format'];
            $order_data['shipping_custom_field'] =
$shipping_order['shipping_custom_field'];

            if (isset($shipping_order['shipping_method'])) {
                $order_data['shipping_method'] =
$shipping_order['shipping_method'];
            } else {
                $order_data['shipping_method'] = '';
            }

            if (isset($shipping_order['shipping_code'])) {
                $order_data['shipping_code'] =
$shipping_order['shipping_code'];
            } else {
                $order_data['shipping_code'] = '';
            }
        } else {
            $order_data['shipping_firstname'] = '';
            $order_data['shipping_lastname'] = '';
            $order_data['shipping_company'] = '';
            $order_data['shipping_address_1'] = '';
            $order_data['shipping_address_2'] = '';
            $order_data['shipping_city'] = '';
            $order_data['shipping_postcode'] = '';
            $order_data['shipping_zone'] = '';
            $order_data['shipping_zone_id'] = '';
            $order_data['shipping_country'] = '';
            $order_data['shipping_country_id'] = '';
            $order_data['shipping_address_format'] = '';
            $order_data['shipping_custom_field'] = array();
            $order_data['shipping_method'] = '';
            $order_data['shipping_code'] = '';
        }

        $order_product = array();

        $order_product['products'] = $_REQUEST['products'];

      //  $order_products = json_decode($order_product, true);

//        var_dump(json_decode($order_product, true));
//
//        var_dump($order_product);


//        foreach ($order_product['products'] as $key => $product) {
//            foreach ($product['option'] as $option) {
//
//
//                $option_data[] = array(
//                    'product_option_id'       =>
$option['product_option_id'],
//                    'product_option_value_id' =>
$option['product_option_value_id'],
```

```
//                    'option_id'               => $option['option_id'],
//                    'option_value_id'         =>
$option['option_value_id'],
//                    'name'                    => $option['name'],
//                    'value'                   => $option['value'],
//                    'type'                    => $option['type']
//                );
//            }
//
////            $order_product['product'][] = array(
////                'product_id' => $product['product_id'],
////                'name'       => $product['name'],
////                'model'      => $product['model'],
////                'option'     => $option_data,
////                'download'   => $product['download'],
////                'quantity'   => $product['quantity'],
////                'subtract'   => $product['subtract'],
////                'price'      => $product['price'],
////                'total'      => $product['total'],
////                'tax'        => $this->tax->getTax($product['price'],
$product['tax_class_id']),
////                'reward'     => $product['reward']
////            );
//
//
//        }
        $order_data['products'] = $_REQUEST['products'];
//        var_dump($order_data['products']);
//        die();


        // Gift Voucher
        $order_data['vouchers'] = array();

        if (!empty($this->session->data['vouchers'])) {
            foreach ($this->session->data['vouchers'] as $voucher) {
                $order_data['vouchers'][] = array(
                    'description'      => $voucher['description'],
                    'code'             => token(10),
                    'to_name'          => $voucher['to_name'],
                    'to_email'         => $voucher['to_email'],
                    'from_name'        => $voucher['from_name'],
                    'from_email'       => $voucher['from_email'],
                    'voucher_theme_id' => $voucher['voucher_theme_id'],
                    'message'          => $voucher['message'],
                    'amount'           => $voucher['amount']
                );
            }
        }

        $order_data['comment'] = $_REQUEST['comment'];

        $sub_total = $_REQUEST['total'] - $_REQUEST['total_ship'];


        $totals[0] = [
            'code' => 'sub_total',
            'title' => 'Sub-Total',
            'value' => $sub_total,
            'sort_order' => 1
        ];

        $totals[1] = [
            'code' => 'shipping',
```

```
            'title' => 'Flat Shipping Rate',
            'value' => $_REQUEST['total_ship'],
            'sort_order' => 3
        ];

        $totals[2] = [
            'code' => 'total',
            'title' => 'Total',
            'value' => $_REQUEST['total'],
            'sort_order' => 9
        ];


        $order_data['totals'] = $totals;
        $order_data['total'] = $_REQUEST['total'];


        if (isset($this->request->cookie['tracking'])) {
            $order_data['tracking'] = $this->request->cookie['tracking'];

            $subtotal = $this->cart->getSubTotal();

            // Affiliate
            $affiliate_info = $this->model_account_customer-
>getAffiliateByTracking($this->request->cookie['tracking']);

            if ($affiliate_info) {
                $order_data['affiliate_id'] =
$affiliate_info['customer_id'];
                $order_data['commission'] = ($subtotal / 100) *
$affiliate_info['commission'];
            } else {
                $order_data['affiliate_id'] = 0;
                $order_data['commission'] = 0;
            }

            // Marketing
            $this->load->model('checkout/marketing');

            $marketing_info = $this->model_checkout_marketing-
>getMarketingByCode($this->request->cookie['tracking']);

            if ($marketing_info) {
                $order_data['marketing_id'] =
$marketing_info['marketing_id'];
            } else {
                $order_data['marketing_id'] = 0;
            }
        } else {
            $order_data['affiliate_id'] = 0;
            $order_data['commission'] = 0;
            $order_data['marketing_id'] = 0;
            $order_data['tracking'] = '';
        }

        $order_data['order_status_id'] = $_REQUEST['status_id'];
        $order_data['language_id'] = $this->config-
>get('config_language_id');
        $order_data['currency_id'] = $this->currency->getId($this-
>session->data['currency']);
        $order_data['currency_code'] = $this->session->data['currency'];
        $order_data['currency_value'] = $this->currency->getValue($this-
>session->data['currency']);
        $order_data['ip'] = $this->request->server['REMOTE_ADDR'];
```

```
        if (!empty($this->request->server['HTTP_X_FORWARDED_FOR'])) {
            $order_data['forwarded_ip'] = $this->request-
>server['HTTP_X_FORWARDED_FOR'];
        } elseif (!empty($this->request->server['HTTP_CLIENT_IP'])) {
            $order_data['forwarded_ip'] = $this->request-
>server['HTTP_CLIENT_IP'];
        } else {
            $order_data['forwarded_ip'] = '';
        }

        if (isset($this->request->server['HTTP_USER_AGENT'])) {
            $order_data['user_agent'] = $this->request-
>server['HTTP_USER_AGENT'];
        } else {
            $order_data['user_agent'] = '';
        }

        if (isset($this->request->server['HTTP_ACCEPT_LANGUAGE'])) {
            $order_data['accept_language'] = $this->request-
>server['HTTP_ACCEPT_LANGUAGE'];
        } else {
            $order_data['accept_language'] = '';
        }




        $order_info = $this->model_extension_module_apimodule-
>addOrder($order_data);

        $data_history = [
            'order_id' => $order_info,
            'order_status_id' => $order_data['order_status_id'],
            'notify' => 0,
            'comment' => $order_data['comment'],
            'date_added' => date("Y-m-d H:i:s"),
        ];


        $add_hisory_order = $this->model_extension_module_apimodule-
>historyorder($data_history);


        $error = $this->valid();
        if ($error != null) {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION,
                'error' => $error,
                'status' => false]));
            return;
        }

        $this->response->setOutput(json_encode([
                'version' => $this->API_VERSION,
                'status' => true,
                'response' =>[
                    'attributes' => $order_info,
                    'history' => $add_hisory_order,
                ]
            ]
        ));
```

```
    }



    /**
     * @api {post} index.php?route=module/apimodule/getorderinfo  Order
Info
     * @apiName getOrderInfo
     * @apiDescription Order details
     * @apiGroup Order
     *
     * @apiParam {Token}       token                               Your
unique token.
     * @apiParam {Number}      order_id                            Unique
order ID.
     *
     * @apiSuccess {Number}    version                             Current
API version.
     * @apiSuccess {Bool}      status                              Response
status.
     * @apiSuccess {Array[]}   response                            Array
with content response.
     *
     * @apiSuccess {String}    response.order_number               Number
of the order.
     * @apiSuccess {String}    response.fio                        Client's
FIO.
     * @apiSuccess {String}    response.status                     Status
of the order.
     * @apiSuccess {String}    response.email                      Client's
email.
     * @apiSuccess {String}    response.telephone                  Client's
phone.
     * @apiSuccess {String}    response.total                      Total
sum of the order.
     * @apiSuccess {String}    response.currency_code              Default
currency of the shop.
     * @apiSuccess {String}    response.date_added                 Date
added of the order.
     * @apiSuccess {Array[]}   response.statuses                   Statuses
list for order.
     *
     * @apiSuccess {String}    response.statuses.language_id       Language
id
     * @apiSuccess {String}    response.statuses.name              Status
name
     * @apiSuccess {String}    response.statuses.order_status_id   Status
id
     *
     *
     *
     *
     * @apiSuccessExample Success-Response:
     *     HTTP/1.1 200 OK {
     *      "response" :
     *          {
     *              "order_number" : "6",
     *              "currency_code": "RUB",
     *              "fio" : "Anton Kiselev",
     *              "email" : "client@mail.ru",
     *              "telephone" : "056 000-11-22",
     *              "date_added" : "2016-12-24 12:30:46",
     *              "total" : "1405.00",
```

```
     *              "status" : "Сделка завершена",
     *              "statuses" :
     *                  {
     *                          {
     *                              "name": "Отменено",
     *                              "order_status_id": "7",
     *                              "language_id": "1"
     *                          },
     *                          {
     *                              "name": "Сделка завершена",
     *                              "order_status_id": "5",
     *                              "language_id": "1"
     *                           },
     *                           {
     *                               "name": "Ожидание",
     *                               "order_status_id": "1",
     *                               "language_id": "1"
     *                            }
     *                      }
     *          },
     *      "status" : true,
     *      "version": 1.0
     * }
     *
     * @apiErrorExample Error-Response: {
     *       "error" : "Can not found order with id = 5",
     *       "version": 1.0,
     *       "Status" : false
     *     }
     */
    public function getorderinfo()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');
        if (isset($_REQUEST['order_id']) && $_REQUEST['order_id'] != '') {
            $id = $_REQUEST['order_id'];

            $error = $this->valid();
            if ($error != null) {
                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'error' => $error, 'status' => false]));
                return;
            }

            $this->load->model('extension/module/apimodule');
            $order = $this->model_extension_module_apimodule-
>getOrderById($id);

            if (count($order) > 0) {
                $data['order_number'] = $order[0]['order_id'];

                if (isset($order[0]['firstname']) &&
isset($order[0]['lastname'])) {
                    $data['fio'] = $order[0]['firstname'] . ' ' .
$order[0]['lastname'];
                } else {
                    $data['fio'] = $order[0]['payment_firstname'] . ' ' .
$order[0]['payment_lastname'];
                }
                if (isset($order[0]['email'])) {
                    $data['email'] = $order[0]['email'];
                } else {
                    $data['email'] = '';
                }
```

```
                if (isset($order[0]['telephone'])) {
                    $data['telephone'] = $order[0]['telephone'];
                } else {
                    $data['telephone'] = '';
                }

                $data['date_added'] = $order[0]['date_added'];

                if (isset($order[0]['total'])) {
                    $data['total'] = $this->currency-
>format($order[0]['total'], $order[0]['currency_code'],
$order[0]['currency_value']);
                }
                if (isset($order[0]['name'])) {
                    $data['status'] = $order[0]['name'];
                } else {
                    $data['status'] = '';
                }
                $statuses = $this->model_extension_module_apimodule-
>OrderStatusList();
                $data['statuses'] = $statuses;
                $data['currency_code'] = $order[0]['currency_code'];
                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'response' => $data, 'status' => true]));
            } else {
                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'error' => 'Can not found order with id = ' . $id,
'status' => false]));
            }
        } else {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION, 'error' => 'You have not specified ID', 'status' =>
false]));
        }
    }

    /**
     * @api {post} index.php?route=module/apimodule/paymentanddelivery
Payment and delivery by order
     * @apiName getOrderPaymentAndDelivery
     * @apiDescription Receive payment and delivery by order
     * @apiGroup Order
     *
     * @apiParam {Number} order_id                                  Unique
order ID.
     * @apiParam {Token}  token your                                Unique
token.
     *
     * @apiSuccess {Number}    version                              Current
API version.
     * @apiSuccess {Bool}      status                               Response
status.

     * @apiSuccess {Array[]}   response                             Array
with content response.
     * @apiSuccess {String}    response.payment_method              Payment
method.
     * @apiSuccess {String}    response.shipping_method             Shipping
method.
     * @apiSuccess {String}    response.shipping_address            Shipping
address.
     *
     * @apiSuccessExample Success-Response:
     *     HTTP/1.1 200 OK {
```

```
     *          "response":
     *              {
     *                  "payment_method" : "Оплата при доставке",
     *                  "shipping_method" : "Доставка с фиксированной
стоимостью доставки",
     *                  "shipping_address" : "проспект Карла Маркса 1,
Днепропетровск, Днепропетровская область, Украина."
     *              },
     *          "status": true,
     *          "version": 1.0
     *      }
     * @apiErrorExample Error-Response:
     *
     *    {
     *      "error": "Can not found order with id = 90",
     *      "version": 1.0,
     *      "Status" : false
     *   }
     *
     */
    public function paymentanddelivery()
    {
        header("Access-Control-Allow-Origin: *");
        $this->response->addHeader('Content-Type: application/json');
        if (isset($_REQUEST['order_id']) && $_REQUEST['order_id'] != '') {
            $id = $_REQUEST['order_id'];

            $error = $this->valid();
            if ($error != null) {
                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'error' => $error, 'status' => false]));
                return;
            }

            $this->load->model('extension/module/apimodule');
            $order = $this->model_extension_module_apimodule-
>getOrderById($id);



//var_dump($data);
//die();
            if (count($order) > 0) {
                $data['shipping_address'] = '';
//                var_dump($order);
//                die();
                if (isset($order[0]['payment_method']) &&
$order[0]['payment_method'] != '') {
                    $data['payment_method'] = $order[0]['payment_method'];
                }
                if (isset($order[0]['shipping_method']) &&
$order[0]['shipping_method'] != '') {
                    $data['shipping_method'] =
$order[0]['shipping_method'];
                }

//                var_dump();
//                die();
                if (isset($order[0]['shipping_address_1']) &&
$order[0]['shipping_address_1'] != '') {
                    $data['shipping_address'] .=
$order[0]['shipping_address_1'];
                }
                if (isset($order[0]['shipping_address_2']) &&
```

```
$order[0]['shipping_address_2'] != '') {
                    $data['shipping_address'] .= ', ' .
$order[0]['shipping_address_2'];
                }
                if (isset($order[0]['shipping_city']) &&
$order[0]['shipping_city'] != '') {
                    $data['shipping_address'] .= ', ' .
$order[0]['shipping_city'];
                }
                if (isset($order[0]['shipping_country']) &&
$order[0]['shipping_country'] != '') {
                    $data['shipping_address'] .= ', ' .
$order[0]['shipping_country'];
                }
                if (isset($order[0]['shipping_zone']) &&
$order[0]['shipping_zone'] != '') {
                    $data['shipping_address'] .= ', ' .
$order[0]['shipping_zone'];
                }

                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'response' => $data, 'status' => true]));
            } else {
                $this->response->setOutput(json_encode(['version' =>
$this->API_VERSION, 'error' => 'Can not found order with id = ' . $id,
'status' => false]));
            }
        } else {
            $this->response->setOutput(json_encode(['version' => $this-
>API_VERSION, 'error' => 'You have not specified ID', 'status' =>
false]));
        }
}
```

Додаток Б

ВІДГУК КЕРІВНИКА

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Спірінцев В.В., к.т.н., доцент, доцент кафедри ПЗКС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

На кваліфікаційну роботу

студента Шевченко Іллі Віталійовича
(прізвище, ім'я, по батькові)

курсу II групи 122м-22-2
спеціальності 122 Комп'ютерні науки
на тему Дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart

Актуальність теми Представлена магістерська кваліфікаційна робота присвячена вивченню взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart. З урахуванням поширеності використання мобільних додатків та значущості впливу API на їхню ефективність, дослідження цієї теми стає доречним та важливим завданням. В сучасному світі, де електронна комерція та мобільні технології швидко розвиваються, робота з CMS системами через мобільні застосунки стає стратегічно важливою.

Мета досліджень полягає в вивченні та оптимізації взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart на платформах Android та iOS. Шляхом застосування принципів моделювання взаємодії та розробки плагінів, дослідження спрямоване на створення ефективного інструментарію для оптимізації роботи з мобільним застосунком на основі вимог бізнесу.

Коротка характеристика розділів роботи Перший розділ присвячений огляду проблематики взаємодії мобільного застосунку з RESTful API у контексті роботи з CMS системою Opencart, а також розгляду можливих шляхів її вирішення.

Другий розділ включає аналітичний огляд архітектури, композиції та передачі даних у фреймворках, необхідних для розробки мобільного застосунку – Flutter та Dart.

Третій розділ націлено на розробку та аналіз ефективності мобільного додатку, спроектованого для оптимізованої взаємодії з RESTful API та CMS системою Opencart, з використанням фреймворку Flutter.

Практичне значення роботи проведених досліджень виявляється у можливості використання розробленої системи для створення високодинамічних мобільних додатків, орієнтованих на взаємодію з RESTful API та вбудованою системою управління контентом (CMS) Opencart. Розробка додатків з використанням цієї системи дозволяє зробити процес розробки більш гнучким і ефективним, зменшити витрати грошових та управлінських ресурсів, що робить її ідеальним інструментом для команди розробників.

Зауваження та недоліки. Додані не всі функції в систему, так як в адмін панелі, ще є багато функціональності. Звістно, що це можна віднести до подальшого розвитку роботи і це не як не впливає на загальну схвальну оцінку роботи.

Висновки та оцінка Магістром було проведено аналіз та розробка мобільного застосунку і плагіну для CMS системи Opencart. Під час виконання магістерської кваліфікаційної роботи було в повній мірі виконане завдання. Вважаю, що магістерська кваліфікаційна робота заслуговує оцінку 95 «відмінно», а Шевченко І.В. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Науковий керівник _____

(прізвище, ім'я, по батькові, посада, місце роботи)

«___»_____20__ р. _____

(підпис)

Додаток В

Додаток В

РЕЦЕНЗІЯ

на кваліфікаційну роботу

студента Шевченка Іллі Віталійовича

(прізвище, ім’я, по батькові)

курсу II групи 122м-22-2

кафедри програмного забезпечення комп’ютерних систем

спеціальності 122 Комп’ютерні науки

Тема роботи Дослідження взаємодії мобільного застосунку з RESTful API у роботі з CMS системою Opencart

Стисла характеристика розділів роботи Вступний розділ: Подано огляд проблем та їх актуальність у вирішенні завдань, пов'язаних з інтеграцією мобільного застосунку з RESTful API в контексті CMS системи Opencart.

Аналітичний огляд фреймворків: Розглянуто архітектуру, композицію та механізми передачі даних у фреймворках Flutter та Dart, з фокусом на їхню ефективність у високодинамічних мобільних додатках.

Розробка мобільного додатка: У цьому розділі представлено розробку мобільного застосунку, спрямованого на визначення ефективності фреймворку Flutter в умовах високоінтенсивних мобільних додатків, зокрема в контексті взаємодії з RESTful API та CMS системою Opencart.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування У даній кваліфікаційній роботі студент вніс кілька обґрунтованих пропозицій щодо вирішення поставлених завдань. Кожна із запропонованих ідей є науково підтвердженою та підкріплена відповідними даними, що вказує на високий рівень наукового обґрунтування студентських внесків у роботу.

Практичне значення роботи виявляється в можливості використання розробленої системи для створення високодинамічних мобільних додатків з спільною кодовою базою, спрощуючи процес розробки та зменшуючи витрати фінансових та управлінських ресурсів на велику команду розробників. За допомогою Flutter як базової платформи, створена система дозволяє зібрати додаток, який працює

на більш ніж п'яти різних платформах, що робить її вельми ефективним і універсальним інструментом для розробки мобільних додатків.

Якість оформлення роботи Магістерська кваліфікаційна робота, яку подано на рецензію, виконана у повному обсязі у встановлений термін. Робота є добре структурованою та достатньо проілюстрованою. Викладена основна суть проблеми, що вирішується в ході виконання роботи, і шляхів її вирішення.

Недоліки в роботі в роботі використовувалися пуш повідомлення, які відправлялися при оформлені замовлення, але студентом було не в повній мірі описана структура взаємодії.

Загальний висновок Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

Оцінка магістерської роботи Робота заслуговує оцінки «відмінно», а студент Шевченко І.В. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Рецензент _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____20__ р. _____
(підпис)

Додаток Г

ПЕРЕЛІК ДОКУМЕНТІВІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Коваленко.doc	Пояснювальна записка роботи. Документ Word.
Диплом_Коваленко.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Коваленко.ppt	Презентація роботи