Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікаціїі*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Віртуальний помічник для інтернет магазинів в Instagram
(тема)

Виконав:
студент 2 курсу, групи *ІМІм-20-2*
Антонян О.Г.

Спеціальності *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *доц., к.т.н. Омельченко С.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2022 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Антонян О.Г.*
(підпис) (прізвище та ініціали)

Керівник _____ *Омельченко С.В.*
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ІМІ_____
(підпис)
“____” _____2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Антонян Олександр Гришаєвич*
(прізвище, ім’я, по батькові)

1. Тема роботи *Віртуальний помічник для інтернет магазинів в Instagram*
затверджені наказом університету від 14 березня 2022 року № 592 *Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *13 травня 2022 р.*
3. Вихідні дані до роботи
1. Facebook API Graph
2. Facebook Messenger Platform
3. Безкоштовний веб-фреймворк Laravel із відкритим кодом
4. JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов React.
5. Мова програмування PHP

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Огляд та аналіз
2. Структура бази даних
3. Розробка серверної частини
4. Розробка кліенской частини
5. Тестування системи
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Слайди у форматі Power Point (назва, мета і задачі роботи, Facebook messanger, Проектування бази даних, Інтеграція с Facebook, Розробка сервісної частини, Розробка кліенской частини, Панель продавця – авторизація та реєстрація, Панель продавця – домашня сторінка, Панель продавця – сторінка продуктів, Панель продавця – сторінка створення продукту, Панель продавця – сторінка контактів, Панель продавця – сторінка замовлень, Панель продавця – сторінка «Доставка», Панель продавця – сторінка «Оплата», Тестування системи – 1, Тестування системи – 2, Тестування системи – 3, Тестування системи - 4, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	*14.03.22*	*виконано*
2	*Підбір літератури за темою роботи*	*15.03-18.03.22*	*виконано*
3	*Виконання розділу 1*	*19.03-29.03.22*	*виконано*
4	*Виконання розділу 2*	*30.03-09.04.22*	*виконано*
5	*Виконання розділу 3*	*10.04-20.04.22*	*виконано*
6	*Виконання розділу 4*	*21.04-01.05.22*	*виконано*
7	*Виконання розділу 5*	*02.05-08.05.22*	*виконано*
8	*Оформлення пояснювальної записки*	*09.05-11.05.22*	*виконано*
9	*Оформлення презентаційного матеріалу,*	*12.05-13.05.22*	*виконано*
	підготовка до захисту у ЕК		

Дата видачі завдання *14.03.2022 р.*

Студент _____ *Антонян О. Г.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Омельченко С.В.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 68 с., 29 рис., 15 табл., 7 джерел, 2 додатки.

Об'єкт дослідження – сервіс для створення віртуального асистента для інтернет магазинів на площадці соціальної мережі «Instagram».

Мета роботи – розробити систему, яка покращить взаємодію між продавцем та покупцем, зробить процес покупки автоматичним, за рахунок цього зменшить час на оформлення замовлення покупцем, та також зменшити відсоток відмов, за рахунок миттєвої відповіді.

Результатом роботи є створення сервісу для інтернет магазинів в «Instagram», щоб вони усього у пару натискають, могли автоматизувати відповіді с клієнтом, та автомізувати процес оформлення замовлення. Цим самим заощадити час менеджера, збільшити час реагування на повідомлення, та усунути помилки, при оформлені замовлення, які виникають за рахунок людського фактору.

ЧАТ-БОТИ, ВІРТУАЛЬНИЙ АСИСТЕНТ, PHP, LARAVEL, REACTJS

THE ABSTRACT

Explanatory note: 68 p., 29 fig., 15 tabl., 7 sources, 2 app.

The object of study is the service for creating a virtual assistant for online stores on the Instagram social media platform.

Meta robots - expand the system, how to fix the relationship between the seller and the buyer, make the purchase process automatic, change the hour for the buyer’s registration, and also change the amount of credits, for the balance of the meeting.

Results - the creation of a service for online stores on Instagram, so that the stink of a couple is on the way, they could automate the correspondence with the client, and that automize the process of registration of the application. Tsim yourself to spare the manager's hour, increase the time of response to reminders, and use pardons, when making a request, as if blamed for the human factor.

CHAT BOTS, VIRTUAL ASSISTANT, PHP, LARAVEL, REACTJS

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API (Application Programming Interface) – опис способів, якими одна комп'ютерна програма може взаємодіяти з програмою;

HTTP (HyperText Transfer Protocol – протокол прикладного рівня передачі даних, спочатку - у вигляді гіпертекстових документів у форматі HTML, в даний час використовується для передачі довільних даних;

URL (Uniform Resource Locator) – система уніфікованих адрес електронних ресурсів, або одноманітний визначник місцезнаходження ресурсу.

MVC (Model-View-Controller) – схема поділу даних програми та керуючої логіки на три окремі компоненти: модель, уявлення та контролер — таким чином, що модифікація кожного компонента може здійснюватися незалежно.

SPA (single page application) — це веб-додаток або веб-сайт, що використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і організує взаємодію з користувачем через HTML, CSS, JavaScript, що динамічно підвантажуються, зазвичай у вигляді AJAX.

AJAX (Asynchronous Javascript and XML) — підхід до побудови інтерактивних інтерфейсів користувача веб-додатків, що полягає в «фоновому» обміні даними браузера з веб-сервером.

ВСТУП

Електронна комерція відіграє важливу роль у сучасній комерції. Частка електронної комерції в загальному обсязі світової торгівлі в 2020 році зросла на 3 пункти і досягла 18%.

В останні роки обсяги онлайн-торгівлі в усьому світі неухильно зростають. У 2020 році в умовах пандемії Covid-19 онлайн-торгівля пережила новий етап розвитку, який отримав новий виток у різних галузях економіки. За прогнозами, до 2025 року частка онлайн-комерції перевищить 25% [1].

Основну частка підвищення продажів у онлайн-торгівлі обумовлена продажами у соціальних мережах, и це можна пояснити тим, що кількість користувачів соціальних мереж за останній рік зросла більш ніж на 13%. До початку 2021 року в соціальних мережах зареєструвалося майже півмільярда нових користувачів [1].

У середньому протягом 2020 року щодня створювалося понад 1,3 мільйона нових облікових записів, що дорівнює приблизно 15,5 нових користувачів на секунду.

В Україні в 2022 року у сфері соціальних мереж домінує компанія Meta, яка має такі продукти як «Facebook» та «Instagram», які пов’язані в одну велику екосистему. Також Instagram займає друге місце по скачуванням додатку в App Store.

Виходячи з цього Instragram на даний момент є великим торговельним майданчиком, в якому є багато покупців та також продавців, але не зовсім зручна і безпечна влаштована взаємодія між ними. Рішення цього питання стало в 2021 році, коли було випущено API Messenger для Instragram, що дозволяю створювати чат-ботів для соціальної мережи «Instragram».

Метою роботи є розробити сервіс який упросить взаємодію між покупцем та продавцем, та зробить покупку речей в «Instagram» зручнішу для користувача, а для продавця заощадить час на спілкування с покупцем, та також збільшить кількість можливих продаж, за миттєвий час відгуку. Саме тому тема кваліфікаційної роботи є актуальною.

1 ОГЛЯД ТА АНАЛІЗ

1.1 Поняття та функції чат-бота

Чат-бот — це програмне забезпечення або комп'ютерна програма, яка імітує людську розмову за допомогою текстових або голосових взаємодій.

Чат-боти можуть бути такими ж простими, як програми, які відповідають на простий запит однорядковою відповіддю, або такими ж складними, як цифрові помічники, які навчаються та розвиваються, щоб забезпечувати все більший рівень персоналізації у міру збору та обробки інформації [3].

1.1.1 Як працюють чат-боти?

Завдяки штучному інтелекту, автоматизованим правилам та машинно-му навчанню чат-боти обробляють дані для надання відповідей на запити всіх видів.

Існує два основних типи чат-ботів:

– Цілеспрямовані (декларативні) чат-боти - це одно цільові програми, орієнтовані виконання однієї функції. Вони генерують автоматичні, але діалогові відповіді на запити користувачів. Взаємодії з цими чат-ботами дуже специфічні і структуровані і найбільш застосовні для функцій підтримки та обслуговування - подумайте про надійні інтерактивні питання, що часто ставляться [3]. Чат-боти, орієнтовані на завдання, можуть опрацьовувати поширені питання, такі як запити про години роботи або прості транзакції, які не включають безліч змінних. В даний час це найчастіше використовувані чат-боти;

– Чат-боти, керовані даними, та інтелектуальні (розмовні) чат-боти часто називають віртуальними помічниками або цифровими помічниками, і вони набагато складніші, інтерактивні та персоналізовані, ніж чат-боти, орієнтовані виконання завдань. Ці чат-боти враховують контекст і використовують розуміння природної мови, щоб вчитися на ходу. Вони застосовують прогностичний інтелект та аналітику, щоб забезпечити персоналізацію на основі профілів користувачів та минулої поведінки користувачів. Цифрові помічники можуть згодом вивчати уподобання користувача, давати

рекомендації і навіть передбачати потреби. Крім моніторингу даних та намірів, вони можуть ініціювати розмови.

1.1.2 Основні функції чат-ботів

Чат-боти вже багато років використовуються в додатках для обміну миттєвими повідомленнями та інтерактивних онлайн-іграх, і лише недавно вони перейшли у продаж та послуги [3].

Організації можуть використовувати чат-ботів такими способами:

– Покупки в інтернет-магазині. У цих середовищах відділи продажів можуть використовувати чат-ботів, щоб відповідати на нескладні питання про продукти або надавати корисну інформацію, яку споживачі можуть шукати пізніше, включаючи вартість доставки та доступність;

– Обслуговування клієнтів. Відділи обслуговування також можуть використовувати чат-ботів, щоб допомогти агентам обслуговування відповідати на запити, що повторюються. Наприклад, представник служби підтримки може дати чат-боту номер замовлення та запита-ти, коли замовлення буде надіслано. Як правило, чат-бот перекладає дзвінок або текст агенту служби підтримки, коли розмова стає надто складною;

– Віртуальні помічники. Чат-боти також можуть виступати у ролі віртуальних помічників. Apple, Amazon, Google і Microsoft мають фор-ми віртуальних помічників. Програми, такі як Siri від Apple і Cortana від Microsoft, або такі продукти, як Amazon Echo з Alexa або Google Home, відіграють роль особистого чат-бота.

1.1.3 Переваги чат-бота для бізнесу

Давайте розглянемо 5 основних причин, за якими ви повинні сказати «да» чат-ботам:

1) Підвищення якості обслуговування клієнтів. Чат-боти можуть відповідати через кілька секунд після запиту, що є складною задачею для людини-оператора. Вони забезпечують безперебійну розмову з клієнтом і таким чином покращують утримання користувачів та лояльність клієнтів. У міру того, як кожна розмова стає все більш схожою на людську, чат-боти

будуть давати правильні відповіді, щоб покращити якість обслуговування клієнтів;

2) Персоналізація. Чат-боти допомагають у розширеній персоналізації з найпершої взаємодії з клієнтом. Чат-боти можуть швидше отримувати інформацію про клієнтів та відповідним чином адаптувати контент. Він також може давати рекомендації користувачам відповідно до їх уподобань;

3) Підтримка 24/7. Основним недоліком підтримки людини є необхідність постійно тримати співробітника, який працює на зміну. І якість, що забезпечується людьми, буде неоднорідною залежно від їх настрою та ситуації. Щаслива людина, яка відвідує свого першого клієнта, буде сповнена енергії, тоді як змучена людина може забезпечити нижчу якість обслуговування. Тим не менш, чат-боти зі штучним інтелектом цілодобово надають однорідні послуги;

4) Автоматизація процесів. Чат-боти гарантують, що обсяг людського досвіду, необхідного для виконання простих завдань, буде зменшено, щоб ці ресурси можна було переміщувати та використовувати для вирішення складних завдань. Це допомагає підвищити ефективність персоналу та ресурсів;

5) Мінімізуйте витрати на підтримку клієнтів. Чат-боти можуть заощадити до 30% витрат на підтримку клієнтів завдяки якісному обслуговуванню. Він може обробляти безліч квитків одночасно без затримок, зберігаючи при цьому витрати під контролем. Чат-боти здатні до постійної та автоматизованої доробки. Вони навчаються за допомогою запитів користувача і розуміють їх, щоб дати кращу відповідь.

1.2 Соціальна мережа Instagram

Instagram — це безкоштовний онлайн-додаток для обміну фотографіями та платформа соціальної мережі, яку було придбано Facebook у 2012 році.

Instagram дозволяє користувачам редагувати та завантажувати фотографії та короткі відеоролики через мобільний додаток. Користувачі можуть додавати підпис до кожного свого повідомлення та використовувати хештеги та геотеги на основі розташування, щоб індексувати ці повідомлення та зробити їх доступними для пошуку іншими користувачами в програмі. Кожне повідомлення користувача відображається у стрічках його передплатників у Instagram, а також може бути переглянуте публікою, якщо воно позначене за

допомогою хештегів або геотегів. Користувачі також мають можливість зробити свій профіль закритим, щоб тільки їх передплатники могли переглядати їхні повідомлення.

Як і в інших соціальних мережах, користувачі Instagram можуть лайкати, коментувати та додавати в закладки чужі публікації, а також надсилати особисті повідомлення своїм друзям за допомогою функції Instagram Direct.

Instagram це інструмент не тільки для приватних осіб, але і для бізнесу. Додаток для обміну фотографіями пропонує компаніям можливість створити безкоштовний бізнес-аккаунт для просування свого бренду та продуктів. Компанії з бізнес-аккаунтами мають доступ до безкоштовних показників взаємодії та показів. Згідно з веб-сайтом Instagram, понад 1 мільйон рекламодавців по всьому світу використовують Instagram, щоб ділитися своїми історіями та досягати результатів у бізнесі. Крім того, 60% людей кажуть, що дізнаються про нові продукти через програму.

1.3 API Messenger для Instagram

API Messenger для Instagram дозволяє брендам зв'язуватися з клієнта-ми по бажаному ними каналу комунікації. API Messenger для Instagram інтегрується з існуючими інструментами та даними та спрощує управління вели-кою кількістю повідомлень клієнтів, допомагаючи перетворити листування на бізнес-результати [2].

В основі цього API лежить API Graph, яке архітектура аналогічна архітектурі API платформи Messenger.

API Graph – це основний інструмент для завантаження даних на плат-форму Facebook та їх отримання звідти. Він являє собою API на базі HTTP, за допомогою якого програми можуть програмним шляхом вимагати дані, публікувати нові історії, керувати рекламою, завантажувати фото і виконувати безліч інших завдань.

1.3.1 Як працює платформа Messenger?

Коли користувач надсилає повідомлення на сторінку в Messenger, яка використовує програму для часткової або повної автоматизації розмов, відбувається таке (Рисунок 1.1):

1) Наш сервер надсилає повідомлення від користувача на сторінку Messenger з використанням URL-адреси сервера цієї сторінки. Сервер сторінки — це ресурс, на якому розміщено програму, яка надсилає повідомлення;

2) За допомогою API Send програма може відповісти користувачеві Messenger. Використовуючи цю програму, розробники можуть створювати персоналізовані та адаптовані сценарії автоматизованої взаємодії з користувачами.

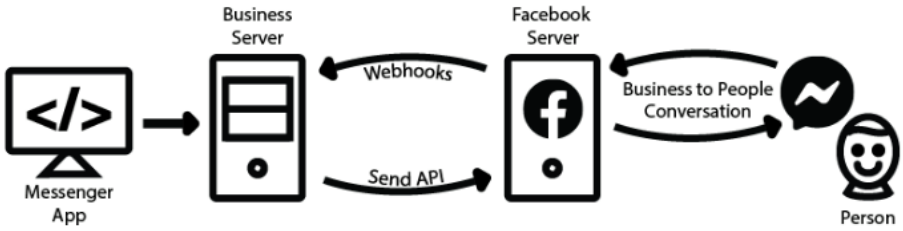

Рисунок 1.1 – Схема роботи платформи Messenger

1.3.2 Send API

Send API — це основний API, використовуваний для відправки повідомлення користувачам, включаючи текст, вкладення, дії відправника та багато іншого [2].

Повідомлення від користувачів за замовчуванням потрапляють у папку «Входить» в Instagram, якщо тільки в інших папках не було попереднього розмови або компанія не підписалася на користувача. При відповіді на діалог через Send API повідомлення будуть автоматично переміщатися з папки запитів у загальну папку.

Вимоги:

– Для взаємодії з цією кінцевою точкою потрібен токен доступу до сторінки з роздільною здатністю instagram_manage_messages. Про-грами в режимі розробки або стандартному доступі обмежені надсиланням повідомлень людям, які мають участь у програмі;

– Використовуйте Graph API v5.0 та вище. Деякі нові функції потребують пізнішої версії API графа;

– Обмін повідомленнями в Instagram повинен мати шлях ескалації до людини. Досвід може починатися з автоматизації для уточнення наміру, але

користувачі повинні мати можливість спілкувати-ся з агентом-людиною в міру необхідності;

– Всі повідомлення, надіслані агентом-людиною поза 24-годинним періодом, повинні використовувати тег HUMAN_AGENT. Тег HUMAN_AGENT дозволяє надсилати повідомлення протягом 7 днів після останнього повідомлення користувача.

1.3.3 Функції Instagram Messaging API

Instagram Messaging API надає набір функцій та API-інтерфейсів, які дозволяють створювати зручні програми для обміну повідомленнями [6]. Доступні функції та подробиці наведено нижче (Таблиця 1.1).

Таблиця 1.1 - Функції Instagram Messaging API

Функція	Опис
Messaging	Надсилайте текст, медіа та інше.
Persistent Menu	Дозволити користувачам відкривати та взаємодіяти з основними функціями сторінки.
Webhooks	Отримуйте сповіщення, коли відбуваються різноманітні взаємодії чи події, зокрема коли людина надсилає пові домлення.
Conversation	Отримати історію розмов з папки "Вхідні".
User Profile	Персоналізуйте розмови та створюйте єдиний досвід.
Story Mention	Найкращі методи роботи з Instagram Stories.

Private Replies	Відповідайте на публікації коментарів одним повідомленням в Instagram.

Продовження табл. 1.1.

Quick Replies	Надайте набір кнопок під час розмови, щоб користувачі могли відповісти.
Ice Breakers	Почніть розмову з компанією зі списку поширених запитань.

1.4 Вибір та аналіз програмного забезпечення

Наш сервіс буде приставляти с себе веб-додаток, для швидкого роботи будем розробляти REST API по принципу Client-Server.

Принцип Client-Server – це відділення клієнта від сервера, де клієнт — це інтерфейс сайту або програми, наприклад, пошуковий рядок відеохостингу. У REST API код запитів залишається на стороні клієнта, а код доступу до даних - на стороні сервера [4]. Це спрощує організацію API, дозволяє легко переносити інтерфейс користувача на іншу платформу і дає можливість краще масштабувати серверне зберігання даних.

1.4.1 REST API

REST API – це архітектурне програмне забезпечення, яке може обмінюватися даними з іншим програмним забезпеченням через мережу віддаленого апарату (Рисунок 1.2) [4].

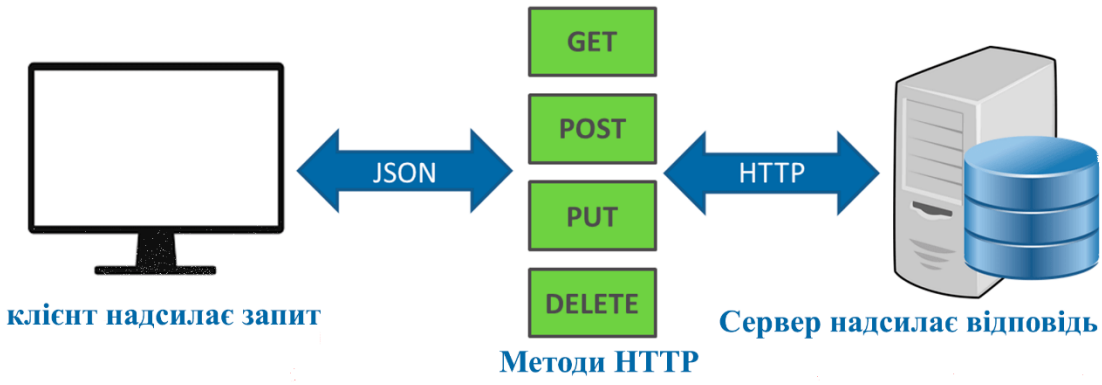

Рисунок 1.2 – Схема REST API

REST API ґрунтується на протоколі передачі гіпертексту HTTP. Кожен об'єкт на сервері в HTTP має свою унікальну URL-адресу у строгому послідовному форматі.

У REST API є 4 методи HTTP, які використовують для дій з об'єктами на серверах:

– GET (отримання інформації про дані або список об'єктів);
– DELETE (видалення даних);
– POST (додавання або заміна даних);
– PUT (регулярне оновлення даних).

1.4.2 Вибір серверного програмного забезпечення

В якості серверної мови ми виберемо мову PHP, а точніше версію 8, для якої в якості каркаса будем використовувати «Laravel». Такий вибір обумовлений високою швидкістю розробки, відрізняється гнучкістю та доброю сумісністю.

«Laravel» є фреймворк веб-додатків із виразним та елегантним синтаксисом. Він спростить вирішення основних проблем, таких як аутентифікація, маршрутизація, сеанси та кешування [5]. «Laravel» доступний, але потужний. Має багато чудових інструментів для великих, надійних додатків:

– як і багато інших фреймворків PHP, «Laravel» використовує архітектурний шаблон MVC для організації коду;

– «Laravel» дотримується філософії виразного, красивого синтаксису. Він призначений для людей, які цінують елегантність, простоту та читабельність;

– Функціональність «Laravel» проста для розуміння та використання.

1.4.3 Вибір клієнтського програмного забезпечення

З боку клієнтської частини, буде веб-додаток, для взаємини клієнта з сервісом. Ядром веб-додатку вибрано фрейморк «ReactJS».

«React» — це декларативна, ефективна та гнучка JavaScript-бібліотека для створення інтерфейсів користувача. Вона дозволяє вам збирати складний UI із маленьких ізольованих шматочків коду, званих «компонентами» [7].

1.5 Основні функції додатку

Головна ідея нашої програми полягає у спрощенні взаємодії між покупцем та продавцем на платформі соціальної мережі «Instagram». Для цього ми розробимо панель продавця і напишемо сценарій для нашого чат-бота, який буде на основі введених даних в панелі давати відповіді.

1.5.1 Панель продавця

Щоб приєднати свій інтернет магазин в instagram до нашої системи, для цього потрібно зареєструватися в нашій панелі продавця. Авторизація буде відбуватися за допомогою входу через facebook сторінку.

У панелі продавця будуть доступні такі функції:

– Увімкнути або вимкнути чат-бота;

– Перегляд списку продуктів;

– Створення, редагування, видалення продуктів;

– Перегляд списку контактів;

– Перегляд списку замовлень;

– Зміна статусу замовлення;

– Перегляд та встановлення методів доставки;

– Перегляд та встановлення методів оплати.

1.5.2 Сценарій чат-боту

Інтернет магазині в «Instagram», публікують пости зі своїми продуктами, які можуть побачити у вільному доступі усі користувачі цієї соціальної мережі. Якщо користувачу сподобався продукт і він бажає его купити, то він пише повідомлення цьому магазину, та відправляю публікацію з даним продуктом.

На основі даної взаємодії між покупцем та магазином, ми пишемо сценарій для нашої системи.

2 СТРУКТУРА БАЗИ ДАНИХ

2.1 Таблиця «users»

В даній таблиці ми будемо зберігати усі дані про наших продавців, в таблиці 2.1 описані усі стовпці таблиці.

Таблиця 2.1 – Таблиця «users»

id	Унікальний ідентифікатор
first_name	Ім'я
last_name	Прізвище
picture	Фото користувача
email	Електронна пошта
email_verified_at	Статус підтвердження електронної пошти
password	Пароль
fb_id	Унікальний ідентифікатор прив’язаної сторінки facebook
fb_access_token	Токен доступу сторінки facebook
fb_account_id	Унікальний ідентифікатор сторінки facebook
fb_account_name	Ім'я сторінки facebook
fb_account_access_token	Токен доступу сторінки facebook

ig_id	Унікальний ідентифікатор instagram
ig_username	Ім'я користувача instagram
ig_picture	Фото користувача instagram
automation_enable	Статус чат-боту
remember_token	Токен користувача
created_at	Дата створення користувача
updated_at	Дата останнього оновлення користувача

2.2 Таблиця «contacts»

В даній таблиці ми будемо зберігати усі дані про наших покупців, в таблиці 2.2 описані усі стовпці таблиці.

Таблиця 2.2 – Таблиця «contacts»

id	Унікальний ідентифікатор
user_id	Унікальний ідентифікатор продавця до якого прив’язаній цей контакт
Ig_id	Унікальний ідентифікатор instagram
username	Ім'я контакту
Automation_enable	Статус чат-боту
Follower_count	Кількість підписників
First_name	Ім'я
Last_name	Прізвище
Phone_number	Номер телефону
email	Електронна пошта
Last_message_sent	Дата останнього відправленого повідомлення
created_at	Дата створення контакту
updated_at	Дата останнього оновлення контакту

2.3 Таблиця «products» та її допоміжні таблиці

В таблиці «products» будуть зберігають усі продукти усіх покупців, в таблиці 2.3 описані усі стовпці таблиці.

Таблиця 2.3 – Таблиця «products»

id	Унікальний ідентифікатор
user_id	Унікальний ідентифікатор продавця до якого прив’язаній цей продукт
name	Ім'я продукту
image	Зображення продукту

Продовження табл. 2.3

sku	Артикль продукту
Status	Кількість підписників
description	Опис продукту
price	Ціна
track_quantity	Відстежувати кількість
quantity	Кількість продуктів
weight	Вага продукту
length	Довжина продукту
width	Ширина продукту
height	Висота продукту
tags	Теги
created_at	Дата створення продукту
updated_at	Дата останнього оновлення продукту

Також продукти можуть мати параметри, такі як колір, та розмір наприклад, то для цих цілей нам потрібно вказувати параметри, та створювати варіації, щоб цю інформацію зберігати ми створимо ще таблицю «product_options», «product_option_values», «product_variants» та «product_variant_values».

Таблиця «product_options» буде зберігати опції продуктів, в таблиці 2.4 описані усі стовпці таблиці.

Таблиця 2.4 – Таблиця «product_options»

id	Унікальний ідентифікатор опції
product_id	Унікальний ідентифікатор продукт до якого прив’язаній це опція
name	Ім'я опції
position	Пріоритет опції

Таблиця «product_option_values» буде зберігати усі значення опції продуктів, в таблиці 2.5 описані усі стовпці таблиці.

Таблиця 2.5 – Таблиця «product_options»

id	Унікальний ідентифікатор значення опції
product_id	Унікальний ідентифікатор продукт до якого прив’язаній це значення
option_id	Унікальний ідентифікатор опції до якого прив’язаній це значення
name	Ім'я значення
position	Пріоритет значення

Таблиця «product_variants» буде зберігати усі варіації продуктів, в таблиці 2.6 описані усі стовпці таблиці.

Таблиця 2.6 – Таблиця «product_variants»

id	Унікальний ідентифікатор варіації
product_id	Унікальний ідентифікатор продукт до якого прив’язана ця варіація
price	Ціна
image	Зображення варіації
sku	Артикль варіації
quantity	Кількість варіації
position	Пріоритет значення

Таблиця «product_variant_values» зберігає в собі усі залежності між

«product_options», «product_option_values» та «product_variants», в таблиці 2.7 описані усі стовпці таблиці.

Таблиця 2.7 – Таблиця «product_variant_values»

id	Унікальний ідентифікатор
product_id	Унікальний ідентифікатор продукт
variant_id	Унікальний ідентифікатор варіації
option_id	Унікальний ідентифікатор опції
option_value_id	Унікальний ідентифікатор значення опції

В нашій системі потрібно також зв’язувати продукти з публікаціями в instagram, и можуть бути у продукту безліч публікації, та у публікації безліч продуктів, тому нам потрібна створити таблиця «product_associated_posts», для збереження публікації та зв'язати їх с продуктам, в таблиці 2.8 описані усі стовпці таблиці.

Таблиця 2.8 – Таблиця «product_associated_posts»

id	Унікальний ідентифікатор
product_id	Унікальний ідентифікатор продукт
caption	Заголовок публікації
media_type	Тип медіа
media_url	URL-адреса медіа
permalink	Постійне посилання

2.4 Таблиця «carts» та її допоміжні таблиці

В таблиці «carts» будуть зберігають кошики усіх контактів, в таблиці 2.9 описані усі стовпці таблиці. Але ми розуміємо, що у кошику можуть бути безліч продуктів, тому ми створимо таблицю «cart_items», яка буде зберігати дані продукту, в таблиці 2.10 описані усі стовпці таблиці «cart_items».

Таблиця 2.9 – Таблиця «carts»

id	Унікальний ідентифікатор
contact_id	Унікальний ідентифікатор контакту

created_at	Дата створення кошику
updated_at	Дата останнього оновлення кошику

Таблиця 2.10 – Таблиця «cart_items»

id	Унікальний ідентифікатор
cart_id	Унікальний ідентифікатор кошику
product_id	Унікальний ідентифікатор продукту
variant_id	Унікальний ідентифікатор варіації
quantity	Кількість

2.5 Таблиця «shipping_methods»

В таблиці «shipping_methods» будуть зберігатися методи доставки, що встановить продавець в панелі продавця, в таблиці 2.11 описані усі стовпці таблиці.

Таблиця 2.11 – Таблиця «shipping_methods»

id	Унікальний ідентифікатор
user_id	Унікальний ідентифікатор продавця
name	Ім'я
status	Статус
payload	Додаткові параметри

2.6 Таблиця «payment_methods»

В таблиці «payment_methods» будуть зберігатися методи оплати, що встановить продавець в панелі продавця, в таблиці 2.12 описані усі стовпці таблиці.

Таблиця 2.12 – Таблиця «payment_methods»

id	Унікальний ідентифікатор
user_id	Унікальний ідентифікатор продавця
name	Ім'я
status	Статус
payload	Додаткові параметри

2.7 Таблиця «orders» та її допоміжні таблиці

Таблиця «orders» буде зберігати усі замовлення наших продавців продавця, в таблиці 2.13 описані усі стовпці таблиці.

Таблиця 2.13 – Таблиця «orders»

id	Унікальний ідентифікатор
user_id	Унікальний ідентифікатор продавця

Продовження табл. 2.13

contact_id	Унікальний ідентифікатор покупця
first_name	Ім'я
Last_name	Прізвище
Phone_number	Номер телефону
Email	Електронна пошта
City	Місто
Status	Статус
Shipping_method	Метод доставки
Shipping_details	Деталі доставки
Payment_method	Метод оплати
Payment_details	Деталі оплати
total	Всього
created_at	Дата створення замовлення
updated_at	Дата останнього оновлення замовлення

Також замовлення може містити безліч продуктів в собі, для цього нам потрібно створи додаткову таблицю «order_items», в якій будемо зберігати деталі продуктів із замовлення продавця, в таблиці 2.14 описані усі стовпці таблиці.

Таблиця 2.14 – Таблиця «order_items»

id	Унікальний ідентифікатор
order_id	Унікальний ідентифікатор замовлення

product_id	Унікальний ідентифікатор продукту
product_name	Ім'я продукту
variant_id	Унікальний ідентифікатор варіації
quantity	Кількість
created_at	Дата створення замовлення
updated_at	Дата останнього оновлення замовлення

На цьому наша база даних спроектована.

3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ

В якості серверної мови програмування ми обрали «php», на каркаси «Laravel», тому наша структура додатку буде по принципу MVC. Структуру сервісу ми обрали REST API, тому наш серверна частина буду виступати в якості API, запити будуть відправляться в форматі JSON.

3.1 Інтеграція з facebook

Для Інтеграція нам потрібно створити додаток на портали розробників facebook. Для цього ми переходимо на портал для розробників facebook, и створюймо сторінку розробника, після переходимо на сторінку «My Apps» та нажимаймо «Create App» для створення нового додатку в «facebook».

На сторінці додатку, в розділі «My products», добавляймо наступні продукти:

– «Facebook Login» – для авторизації через facebook;

– «Instagram Graph API» – для отримання публікацій користувача;

– «Messenger» – для відправки повідомлення в instagram або facebook;

– «Webhooks» – для отримання повідомлення.

Ми ознайомили в попередніх розділах о «facebook API», и зараз на базі цього ми створи наш першій сервіс і назвем його «FacebookApi» цей сервіс буде приймати такі параметри:

– Унікальний ідентифікатор сторінки facebook;

– Токен доступу к сторінці.

И опишемо таки методи для цього сервісу, а саме:

– «setPage» – для оновлення сторонки facebook;

– «request» – функція для відправки запитів до API facebook;

– «me» – отримати дані сторінки;

– «getMePicture» – отримати зображення сторінки;

– «getAccounts» – отримати список прив’язаних instagram сторінок;

– «getAccount» – отримати instagram сторінок, приймає в якості параметрів, унікальний ідентифікатор сторінки;

– «sendMessage» – функція для відправки повідомлені, приймає два параметри, отримувача та саме вміст повідомлення;

– «getMedia» – отримати instagram публікації, приймає в якості параметрів, унікальний ідентифікатор instagram сторінки;

3.2 Авторизація та реєстрація

Автентифікація в системі буде за допомогою facebook сторінку, ми будемо получати токен доступу, провірять його, получать ідентифікатор користувача, та регіструвати в нашій системі.

Для початку ми створимо модель «User» яка буде відповідати нашій таблиці бази даних «users».

Після створення моделі, ми створимо контролер «AuthController», та запишемо до нього наступні методи:

– «me» – отримати поточного користувача;

– «login» – метод для авторизації користувача приймає в якості параметра токен, відправляю его в наш сервіс «FacebookApi» та, викликає метод «me», с параметром нашого токена, та повертає нам об’єкт даних сторінки або помилку;

– «registration» – метод для реєстрації користувача приймає в якості параметра токен, відправляю его в наш сервіс «FacebookApi» та, викликає метод «me», с параметром нашого токена, та повертає нам об’єкт даних сторінки або помилку, якщо повернулась сторінка, то ми створюймо нового користувача;

– «logout» – вийти.

Після проходження автентифікації, треба получити унікальний ідентифікатор instagram сторінки, для цього ми створюємо новий контролер під назвою «UserController», в якому описуємо наступні методи:

– «getInstagramAccounts» – получити instagram сторінки прив’язані до даної facebook сторінки;

«connectInstagramAccount» – підключити instagram сторінки до нашого користувача, приймає параметри ідентифікатор instagram сторінки.

Додамо наші методи в маршрутизатор:

– «login» метод POST – запускає метод login контролера «AuthController»;

– «registration» метод POST – запускає метод registration контролера «AuthController»;

– «logout» метод GET – запускає метод logout контролера «AuthController»;

– «fb-accounts» метод GET – запускає метод «getInstagramAccounts» контролера «UserController»;

– «connect-ig» метод POST – запускає метод «connectInstagramAccount» контролера «UserController».

3.3 Створення, читання, модифікація, видалення продуктів

Для таблиці «products», ми створюймо модель «Product», чиї параметри будуть відповідати колонкам таблиці, та також у нас маються додаткові таблиці для продуктів, і тому ми для них також створюймо моделі, а саме:

– Для таблиці «product_options», ми створюймо модель «ProductOption»;

– Для таблиці «product_option_values», ми створюймо модель «ProductOptionValue»;

– Для таблиці «product_variants», ми створюймо модель «ProductVariant»;

– Для таблиці «product_variant_values», ми створюймо модель «ProductVariantValue».

Після створення усіх додаткових моделей для моделі «Product», нам потрібно зв’язати данні моделі між собою, для цього в «Laravel» маються зручні методи, а саме:

– «HasOne» - є дуже базовим типом зв’язків з базою даних. Наприклад, «User» модель може бути пов’язана з однією «Phone» моделлю. Щоб визначити

цей зв’язок, ми розмістимо «phone» метод на «User» моделі. Метод «phone» повинен викликати «hasOne» метод і повернути його результат;

– «HasMany» - використовується для визначення зв’язків, коли одна модель є батьківською для однієї або кількох дочірніх моделей;

– «BelongsTo» - використовується для дочірніх моделей, щоб отримати батьківську модель.

За допомогою даних методів, ми зв'яжемо усі наші таблиці, для цього в модель «Product» ми добавляймо методи:

– «options» який повертає метод «hasMany» с параметром моделі «ProductOption»;

– «option_values» який повертає метод «hasMany» с параметром моделі «ProductOptionValue»;

– «variants» який повертає метод «hasMany» с параметром моделі «ProductVariant»;

– «variant_values» який повертає метод «hasMany» с параметром моделі «ProductVariantValue».

І також щоб додаткові моделі були зворотне зв’язані с моделлю «Product» в кожну модель потрібно додати метод «product» який буде повертати метод «belongsTo» с параметром моделі «Product».

Перейдемо до зв’язані між собою додаткових моделей, а саме:

– В модель «ProductOption» добавляємо метод «values» який повертає метод «hasMany» с параметром моделі «ProductOptionValue»;

– В модель «ProductOptionValue» добавляємо метод «option» який повертає метод «belongsTo» с параметром моделі «ProductOption»;

– В модель «ProductVariant» добавляємо метод «values» який повертає метод «hasMany» с параметром моделі «ProductVariantValue»;

– В модель «ProductVariantValue» добавляємо метод «variant» який повертає метод «BelongsTo» с параметром моделі «ProductVariant», також метод «option» який повертає метод «BelongsTo» с параметром моделі «ProductOption», та метод «option_value» » який повертає метод «BelongsTo» с параметром моделі «ProductOptionValue».

На цьому наші моделі для збереження продуктів готові, і мають таку структуру зображену на рис. 3.1 .

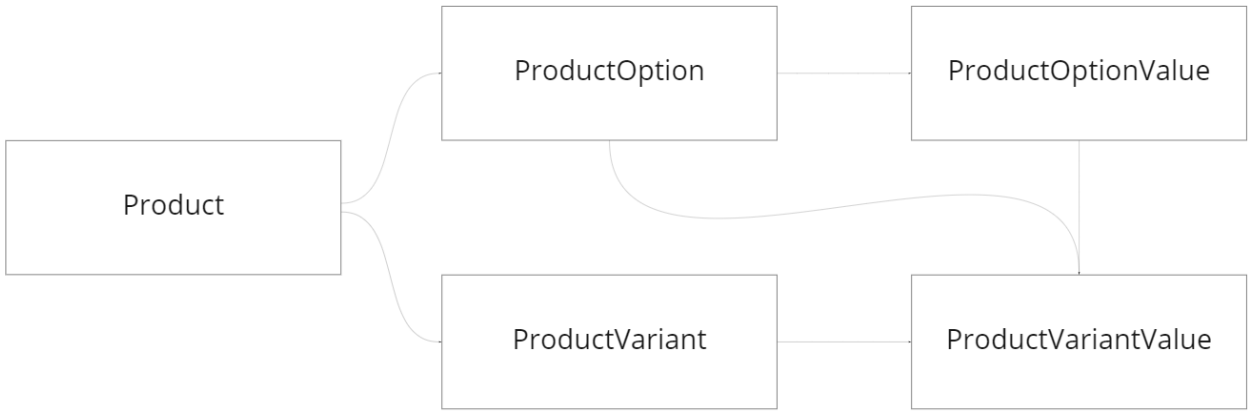

Рисунок 3.1 – Схема відношень між моделями продуктів

Тепер ми створимо контролер «ProductConroller», який буде відповідати за всі маніпуляції с моделями для продуктів.

Опишу основні методи даного контролера:

– «index» - повертаю список продуктів поточного користувача, в якості параметрів приймає:

- «per_page» - кількість відображених продуктів на одній сторінці;
- «page» - номер сторонки;
- «name» - ім’я продукту, слугую для пошуку продукту за ім’ям.

– «store» - приймає модель «Product», та створюю новий продукт;

– «show» - приймає ідентифікатор продукту, та повертає модель «Product»;

– «update» - приймає модель «Product», та оновлюю продукт в базі даних;

– «destroy» - приймає ідентифікатор продукту, та видаляє продукт;

– «saveVariants» - допоміжний метод який викликатися при створені або оновлені, для обробки і збереження нових параметрів варіації продукту.

Додамо наші методи в маршрутизатор:

– «products» метод GET – запускає метод index контролера «ProductConroller»;

– «products» метод POST – запускає метод store контролера «ProductConroller»;

– «products/:product_id» метод GET – запускає метод show контролера «ProductConroller»;

– «products/:product_id» метод POST – запускає метод update контролера «ProductConroller»;

– «products/:product_id» метод DELETE – запускає метод destroy контролера «ProductConroller».

3.4 Створення, читання, модифікація, видалення контактів

Контакти в нашій системи, це покупці, які прив’язані до певного користувача, тобто продавця. Для початку ми створюємо модель «Contact», чиї параметри будуть відповідати колонкам таблиці «contacts».

Так як модель «Contact» прив’язані до користувача, то нам потрібно їх зв'язати, для цього ми добавляємо в модель «User» метод «contacts», який повертаю нам метод «hasMany» с параметром моделі «Contact», та для зворотного зв'язування потрібно в модель «Contact» додати метод «user», який повертаю нам метод «BelongsTo» с параметром моделі «User».

На рис 3.2 наведена візуальна схема відношень між моделями «User» та «Contact», на якій видно, що один користувач, може мати безліч контактів, а контакт лише одного користувача.

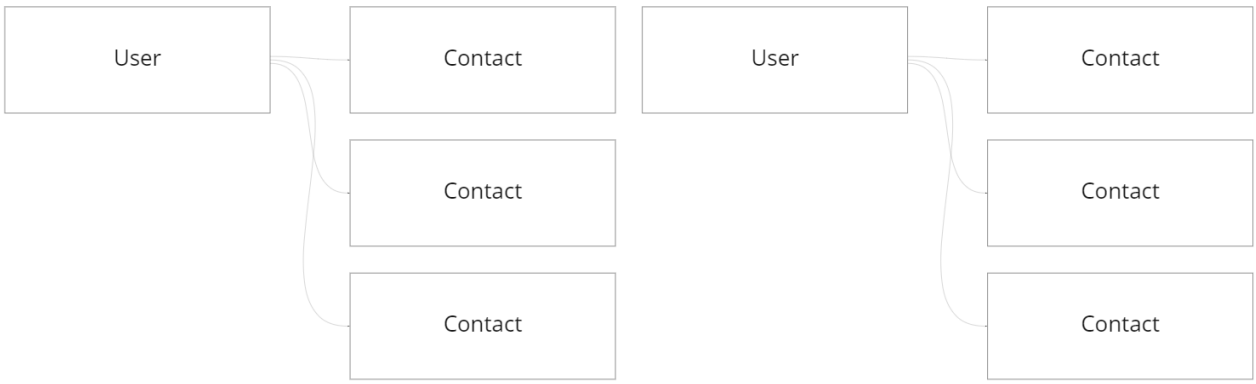

Рисунок 3.2 – Схема відношень між моделями «User» та «Contact»

Тепер ми створимо контролер «ContactConroller», який буде відповідати за всі маніпуляції с моделлю «Contact».

Опишу основні методи даного контролера:

– «index» - повертаю список контактів поточного користувача, в якості параметрів приймає:

- «per_page» - кількість відображених продуктів на одній сторінці;
- «page» - номер сторонки;

- «name» - ім’я продукту, слугую для пошуку продукту за ім’ям.

– «store» - приймає модель «Contact», та створюю новий контакт;

– «show» - приймає ідентифікатор контакту, та повертає модель «Contact»;

– «update» - приймає модель «Contact», та оновлюю дані контакту;

– «destroy» - приймає ідентифікатор контакту, та видаляє продукт.

Додамо наші методи в маршрутизатор:

– «contacts» метод GET – запускає метод index контролера «ContactConroller»;

– «contacts» метод POST – запускає метод store контролера «ContactConroller»;

– «contacts/:contact_id» метод GET – запускає метод show контролера «ContactConroller»;

– «contacts/:contact_id» метод POST – запускає метод update контролера «ContactConroller»;

– «contacts/:contact_id» метод DELETE – запускає метод destroy контролера «ContactConroller».

3.5 Налаштування веб-хуків

Для отримання повідомлення з facebook, нам потрібно налаштувати отримання веб-хуків на наш сервіс, в інтеграції facebook, ми вже додали додаток «Webhooks» для отримання повідомлення, тепер перейдемо до його налаштування.

Перейдемо на сторінку додатку на порталі розробника, далі у пункт меню «Messenger», і в відкритому списку вибираємо «Instagram settings». Далі переходимо в розділ «Webhooks» та нам потрібно вказати «Callback URL» та «Verify token» рис 3.3 .

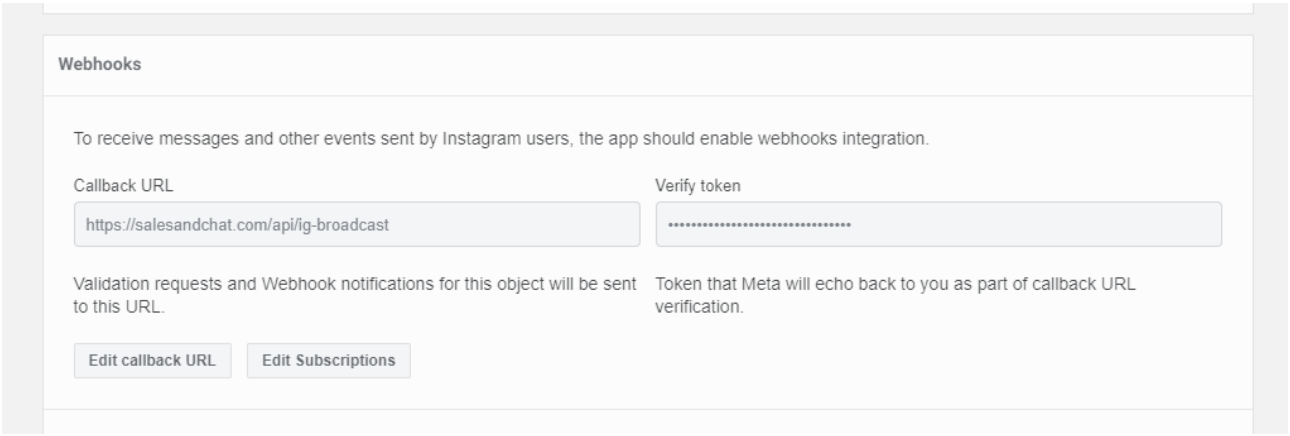

Рисунок 3.3 – Налаштування «Webhooks»

В полі «Callback URL» ми указуймо URL на який нам будуть робитися HTTP запити, в форматі JSON будуть надходити інформація щодо надходження нового повідомлення.

В полі «Verify token» ми указуймо token, який ми самі генерували, і записали у нашій системі, цей токен потрібен для того, щоб facebook підтвердив, що це саме він відправляє нам запити.

Для обробки веб-хуків, ми створимо клас с назвою «Broadcast» до сервісу «FacebookAPI» який буде включати в себе методи:

– «verify» - для перевірки, що саме «Facebook» присилає нам дані;

– «listen» - для читання, та обробки повідомлення надходженні нам с «Facebook».

На основі цього ми створимо маршрут «ig-broadcast» який буде передавати дані в клас «FacebookAPI/Broadcast».

3.6 Класифікація, обробка, аналіз повідомлень

Повідомлення можна класифікувати на текстові повідомлення, вкладені повідомлення, швидкі відповіді, шаблоні.

Текстові повідомлення, це звичайні повідомлення які в контексті себе мають лише текст самого повідомлення рис 3.4 .

Рисунок 3.4 – Текстові повідомлення

Вкладені повідомлення, це повідомлення що мають в собі, переслану публікацію, або відео чи зображення рис 3.5 .

Рисунок 3.5 – вкладені повідомлення

Швидкі відповіді дозволяють отримувати вхідні дані одержувача повідомлення, надсилаючи кнопки в повідомленні. Коли натискається швидка відповідь, значення кнопки надсилається в бесіді, а платформа Messenger надсилає «messanger» подію на веб-хук.

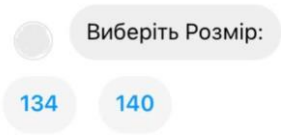

Рисунок 3.6 – швидкі відповіді

Шаблони повідомлень дають змогу запропонувати більш багатий досвід під час розмови, ніж стандартні текстові повідомлення, інтегруючи кнопки, зображення, списки тощо разом із текстом в одне повідомлення рис 3.7 . Шаблони можна використовувати для багатьох цілей, наприклад, для відображення інформації про продукт, прохання одержувача повідомлення вибрати із заздалегідь визначеного набору параметрів та відображення результатів пошуку.

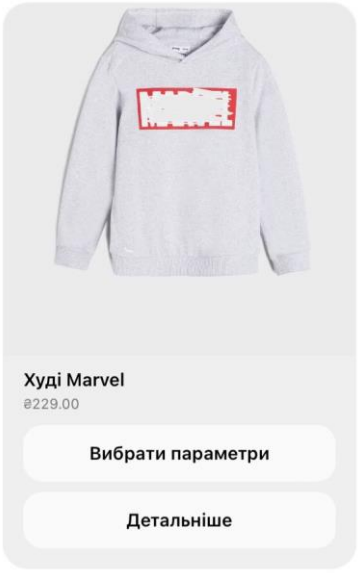

Рисунок 3.7 – Шаблоні повідомлення

Для обробки отриманих повідомлень створюються клас «Recipient» в сервіс «FacebookAPI», який буду отримати повідомлення, та зв’язувати повідомлення з користувачем та контактом за ідентифікатор відправника повідомлення, та ідентифікатором отримувача повідомлення.

Клас «FacebookAPI/Recipient» при ініціалізації приймає параметри ідентифікатор відправника, ідентифікатор отримувача. Властивості даного класу:

– «user» - ідентифікатор користувача;

– «contact» - ідентифікатор контакту.

Даний клас має такі методи:

– «getUser» - отримати модель «User»;

– «handleQuickReply» - цей метод обробляю отримані відповіді на повідомлення «швидкі відповіді»;

– «handleAttachmentMessage» - цей метод обробляю отримані відповіді вкладені повідомлення;

– «handleTextMessage» - цей метод обробляю текстові повідомлення;

– «handlePayload» - цей метод обробляю події після натиску на кнопку в шаблонному повідомленні;

– «greetings» - метод який аналізую дату останнього повідомлення, поточний час, та основі цього додаю до відповіді на повідомлення привітання;

– «productsList» - метод який реагую коли надсилаються публікацію яка зв’язана с продуктом і в відповідь відсилаю шаблон продукту с зображенням продукту, назвою, ціною, та кнопка для взаємодії с продуктом;

– «addToCart» - метод який реагую при натисканні на кнопку «Додати у кошик» та добавляю продукт у кошик.

3.7 Створення, читання, модифікація, видалення кошиків

Для таблиці «carts», ми створюймо модель «Cart», чиї параметри будуть відповідати колонкам таблиці, та також у нас маються додаткова таблиця «cart_items», для якої ми також відповідно створюємо модель «CartItem».

Перейдемо до зв’язування моделі «Cart» с іншими моделями, так як покупець у нас це модель «Contact», тому один контакт може мати лише одну корзину, тому в модель «Contact» ми додамо метод «cart» який буде повертати метод «hasOne» с параметром моделі «Cart», та аналогічно зв’язкою буде додати в модель «Cart» метод «contact», який буде повертати метод «belongsTo» с параметром моделі «Contact».

Кошик може мати безліч в собі продуктів, тому для цього було додано додаткову модель «CartItem», тепер нам потрібно пов’язати ці моделі, для цього ми в модель «Cart» додамо метод «items» який буде повертати метод «hasMany» с параметром моделі «CartItem», та також робимо зворотний зв'язок, додаємо в модель «CartItem» метод «cart» який буде повертати метод «belongsTo» с параметром моделі «Cart».

Модель «CartItem» містить в собі дані о продукті, і нам потрібно зв’язати дану модель с моделюю за методом «belongsTo»

Таким чином ми зв’язали модель «Cart» з іншими моделями рис 3.8 .

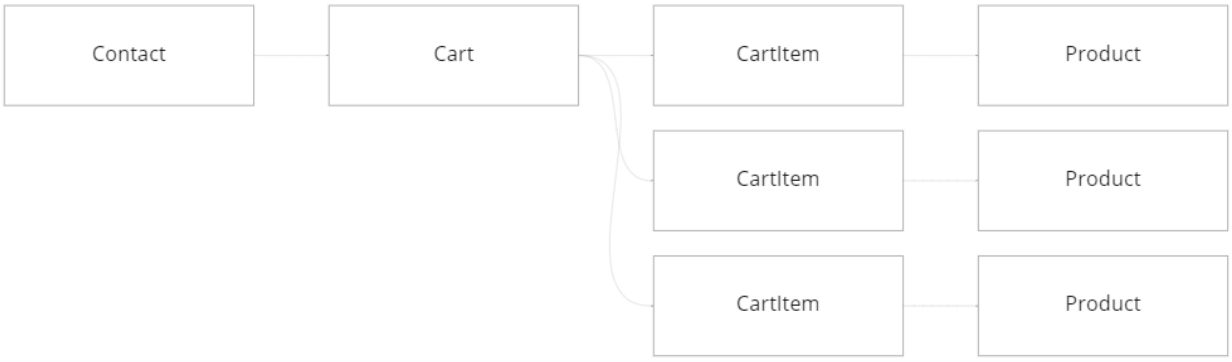

Рисунок 3.8 – Схема відношень моделі «Cart»

Для взаємодії с моделлю «Cart» ми створимо контролер «CartConroller», який буде відповідати за всі маніпуляції с моделлю.

Опишу основні методи даного контролера:

– «view» - повертаю шаблон сторінки кошик;

– «index» - повертаю усі дані кошика;

– «store» - приймає модель «Cart», та оновлюю дані кошику або видаляє;

– «orderCreate» - приймає модель «Cart», та створюю замовлення;

Додамо нові методи в маршрутизатор для нашого API:

– «carts/:cart_id» метод GET – запускає метод index контролера «CartConroller»;

– «carts/:cart_id» метод POST – запускає метод store контролера «CartConroller»;

– «carts/:cart_id/order» метод POST – запускає метод orderCreate контролера «CartConroller».

3.8 Методи доставки

Створюємо модель «ShippingMethod» для таблиці «shipping_methods», модель прив’язані до користувача, то нам потрібно їх зв'язати, для цього ми добавляємо в модель «User» метод «shipping_methods», який повертаю нам метод «hasMany» с параметром моделі «ShippingMethod», та для зворотного зв'язування потрібно в модель «ShippingMethod» додати метод «user», який повертаю нам метод «BelongsTo» с параметром моделі «User».

На рис 3.9 наведена візуальна схема відношень між моделями «User» та «ShippingMethod», на якій видно, що один користувач, може мати безліч контактів, а контакт лише одного користувача.

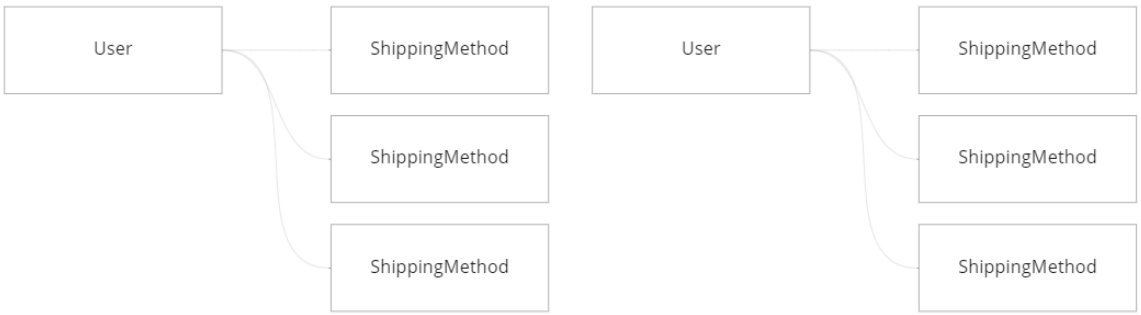

Рисунок 3.9 – Схема відношень моделі «ShippingMethod»

Для маніпулюй с моделлю «ShippingMethod» ми створимо контролер «ShippingMethodController», який буде мати наступні методі:

– «index» - повертаю список методів доставки поточного користувача;

– «store» - приймає модель «ShippingMethod», та створюю новий метод достави;

– «show» - приймає ідентифікатор методу доставки, та повертає модель «ShippingMethod»;

– «update» - приймає модель «ShippingMethod», та оновлюю дані методу;

– «destroy» - приймає ідентифікатор методу, та видаляє метод.

Для кожного користувача будуть доступні три методи доставки, це:

1) Само вивіз;
2) Доставка кур’єром;
3) Доставка сервісом «Нова Пошта».

Ми зробили інтеграцію с сервісом «Нова Пошта», створили клас «NovaPoshtaAPI» для відправлення запитів к API, цей клас має такі методи як:

– «request» - цей метод формую запит к API;

– «searchCity» - робить пошук міст, приймаю в параметрах назву міста;

– «searchWarehouses» - робить пошук відділень, приймаю в параметрах ідентифікатор міста.

Додамо нові методи в маршрутизатор API:

– «shipping-methods» метод GET – запускає метод index контролера «ShippingMethodController»;

– «shipping-methods» метод POST – запускає метод store контролера «ShippingMethodController»;

– «shipping-methods/:shipping_method_id» метод GET – запускає метод show контролера «ShippingMethodController»;

– «shipping-methods/:shipping_method_id» метод POST – запускає метод update контролера «ShippingMethodController»;

– «shipping-methods/:shipping_method_id» метод DELETE – запускає метод destroy контролера «ShippingMethodController»;

– «nova-poshta/cities» метод POST – запускає метод «searchCity» класу «NovaPoshtaAPI»;

– «nova-poshta/warehouses» метод POST – запускає метод «searchWarehouses» класу «NovaPoshtaAPI».

3.9 Методи оплати

Створюємо модель «PaymentMethod» для таблиці «payment_methods», модель прив’язані до користувача, то нам потрібно їх зв'язати, для цього ми добавляємо в модель «User» метод «payment_methods», який повертаю нам метод «hasMany» с параметром моделі «PaymentMethod», та для зворотного зв'язування потрібно в модель «PaymentMethod» додати метод «user», який повертаю нам метод «BelongsTo» с параметром моделі «User».

На рис 3.10 наведена візуальна схема відношень між моделями «User» та «ShippingMethod», на якій видно, що один користувач, може мати безліч контактів, а контакт лише одного користувача.

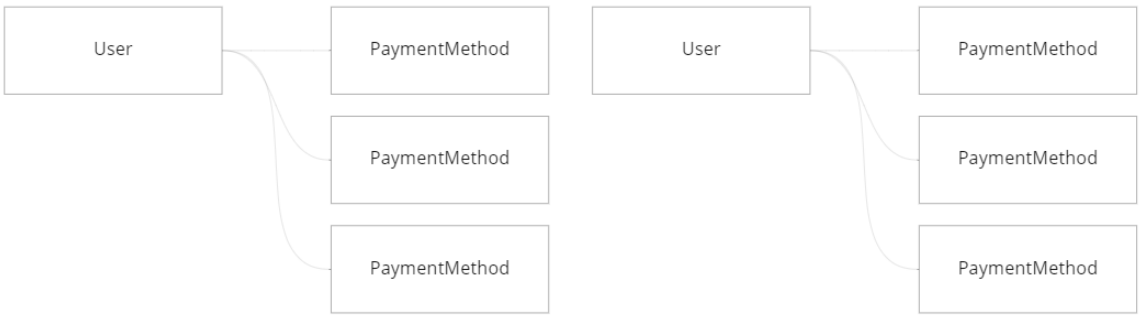

Рисунок 3.10 – Схема відношень моделі «PaymentMethod»

Для маніпулюй с моделлю «PaymentMethod» ми створимо контролер «PaymentMethodController», який буде мати наступні методі:

– «index» - повертаю список методів оплати поточного користувача;

– «store» - приймає модель «PaymentMethod», та створюю новий метод оплати;

– «show» - приймає ідентифікатор методу оплати, та повертає модель «PaymentMethod»;

– «update» - приймає модель «PaymentMethod», та оновлюю дані методу;

– «destroy» - приймає ідентифікатор методу, та видаляє метод.

Для кожного користувача будуть доступні три методи доставки, це:

1) Оплата при доставці;
2) Прямий банківський переказ.

Додамо нові методи в маршрутизатор API:

– «payment-methods» метод GET – запускає метод index контролера «PaymentMethodController»;

– «payment-methods» метод POST – запускає метод store контролера «PaymentMethodController»;

– «payment-methods/:payment_method_id» метод GET – запускає метод show контролера «PaymentMethodController»;

– «payment-methods/:payment_method_id» метод POST – запускає метод update контролера «PaymentMethodController»;

– «payment-methods/:payment_method_id» метод DELETE – запускає метод destroy контролера «PaymentMethodController»;

3.10 Створення, читання, модифікація замовлень

Для таблиці «orders», ми створюймо модель «Order», чиї параметри будуть відповідати колонкам таблиці, та також у нас маються додаткова таблиця «order_items», для якої ми також відповідно створюємо модель «OrderItem».

Перейдемо до зв’язування моделі «Order» с іншими моделями. Замовлень може бути безліч як у продавця, так і у покупця, тому в модель «Contact» ми додамо метод «orders» який буде повертати метод «hasMany» с параметром моделі «Order», та аналогічно зв’язкою буде додати в модель «Order» метод «contact», який буде повертати метод «belongsTo» с параметром моделі «Contact».

Також нам потрібно зв’язати модель «Order» с моделлю «User», для цього додаємо моделі «User» такий же метод «orders» як і у моделі «Contact», а моделі «Order» додаєм ще один метод «user», який буде повертати метод «belongsTo» с параметром моделі «User».

Замовлення може мати безліч в собі продуктів, тому для цього було додано додаткову модель «OrderItem», тепер нам потрібно пов’язати ці моделі, для цього ми в модель «Order» додамо метод «items» який буде повертати метод «hasMany» с параметром моделі «OrderItem», та також робимо зворотний зв'язок, додаємо в модель «OrderItem» метод «order» який буде повертати метод «belongsTo» с параметром моделі «Order».

Таким чином ми зв’язали модель «Order» з іншими моделями рис 3.11 .

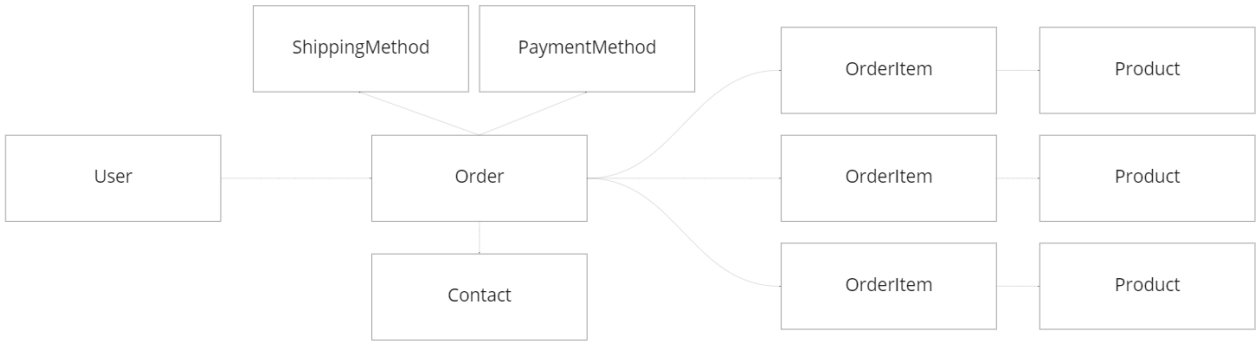

Рисунок 3.11 – Схема відношень моделі «Order»

Для взаємодії с моделлю «Order» ми створимо контролер «OrderConroller», який буде відповідати за всі маніпуляції с моделлю.

Опишу основні методи даного контролера:

– «index» - повертаю список замовлень поточного користувача, в якості параметрів приймає:

- «per_page» - кількість відображених продуктів на одній сторінці;
- «page» - номер сторонки.

– «show» - приймає ідентифікатор замовлення, та повертає модель «Order»;

– «update» - приймає модель «Order», та оновлюю дані замовлення.

Додамо наші методи в маршрутизатор:

– «orders» метод GET – запускає метод index контролера «OrderConroller»;

– «orders/:order_id» метод GET – запускає метод show контролера «OrderConroller»;

– «orders/:order_id» метод POST – запускає метод update контролера «OrderConroller»;

На цьому наша серверна частина закінчена, наше API розроблено, тепер можливо перейти до клієнтської частини.

4 РОЗРОБКА КЛІЕНСКОЇ ЧАСТИНИ

В ядрі нашого SPA буде react js, який завдяки HTTP запитам к API, буде отримувати всю інформацію. В якості додаткових модулів були встановлені:

– «Axios.js» - HTTP-клієнт на основі Promise для браузера;

– «Moment.js» - Бібліотека дат JavaScript для аналізу, перевірки, маніпуляції та форматування дат;

– «Redux» - це передбачуваний контейнер стану для програм JavaScript;

– «React Router» - це легка, повнофункціональна бібліотека маршрутизації для бібліотеки React JavaScript.

– «Facebook SDK» - багатий набір функціональних можливостей на стороні клієнта для додавання соціальних плагінів, входу у Facebook і викликів API Graph.

4.1 Сторінка входу

Ми зробили верстку сторінки, а також підключили авторизацію через facebook, та створили файл «AuthService.js», який відповідаю за всі запити к API, що пов’язані с авторизацією. На рис 4.1 ми бачимо простий дизайн сторінки, та можливість увійти в сервіс за допомогою Facebook сторінки.

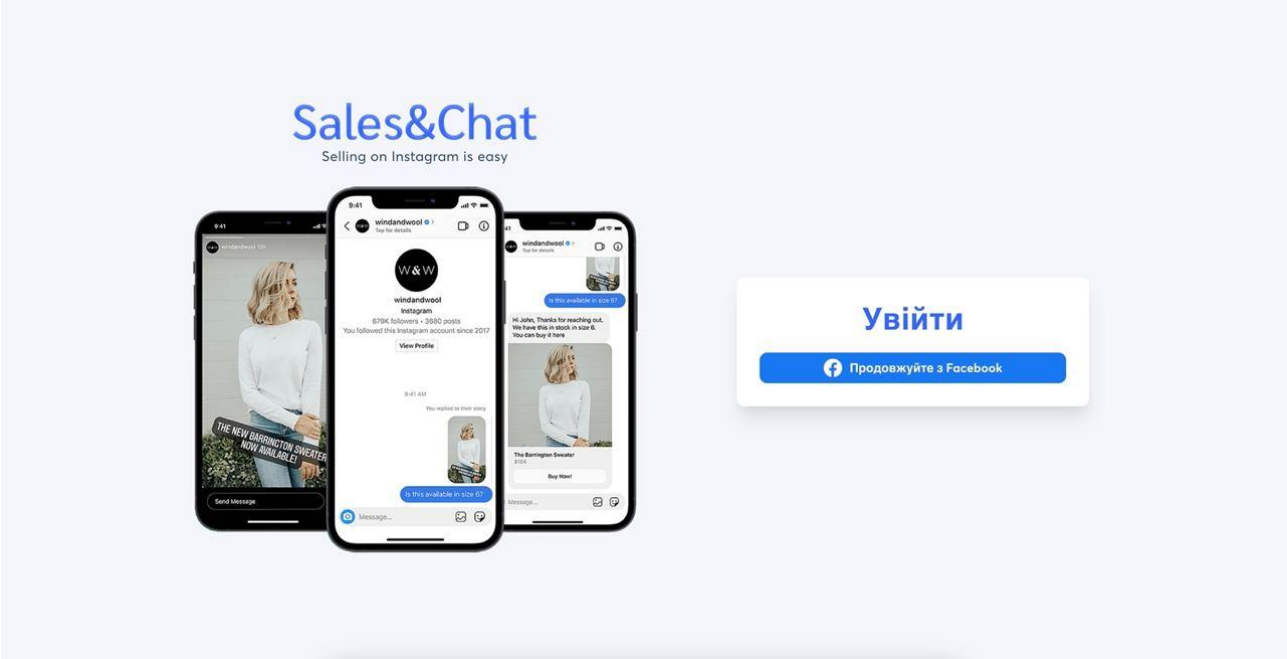

Рисунок 4.1 – Сторінка входу

4.2 Домашня сторінка

На цій сторінці ми бачимо невеличку статистику даного користувача. Після успішної авторизації, користувач попадає на домашню сторінку рис. 4.2 , де йому виводиться наступні елементи:

1) Назва та зображення instagram сторінки;
2) Елемент для увімкнення та вимкнення віртуального асистента;
3) Усього кількість продуктів у даного користувача;
4) Усього кількість контактів у даного користувача;
5) Усього кількість замовлень у даного користувача;

6) Головне меню.

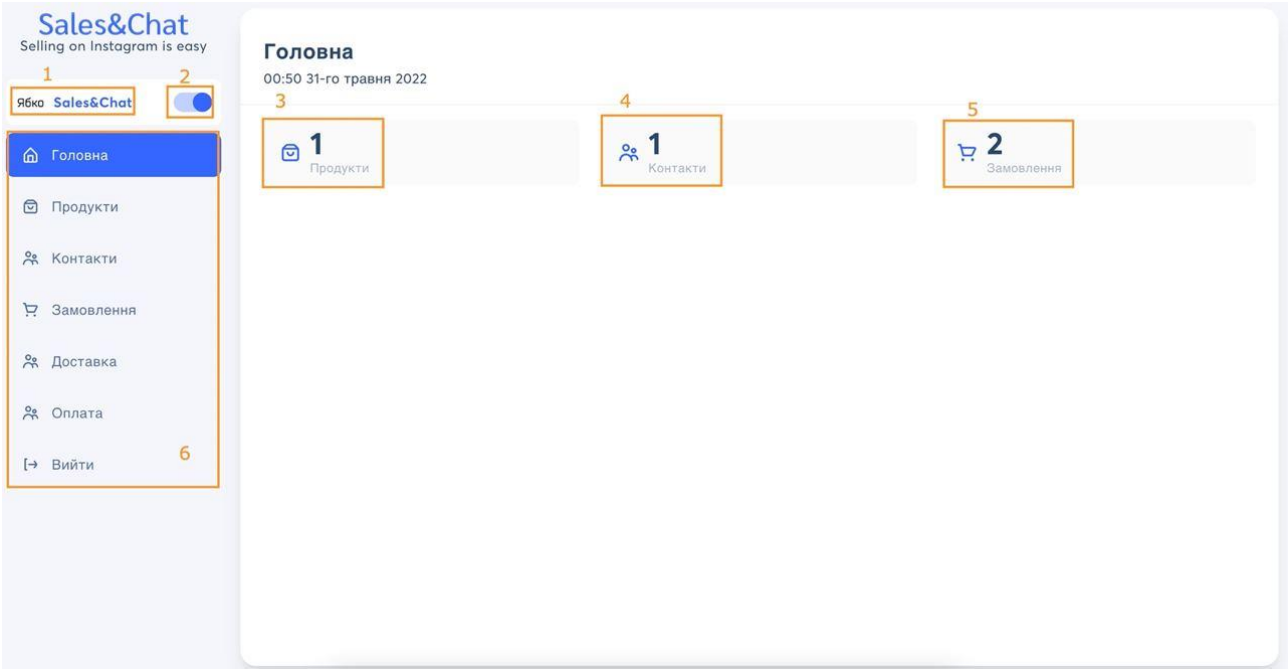

Рисунок 4.2 – Домашня сторінка

4.3 Сторінка продуктів

Якщо в меню натиснути на «Продукти», то користувач попадає на сторінку продуктів рис. 4.3 , де йому виводиться наступні елементи:

1) Дозволяю зробити експорт продуктів;
2) Дозволяю зробити імпорт продуктів;
3) При натискі на кнопку перекидає на сторінку створення продуктів рис. 4.4 ;
4) Поле для пошуку за ім’я продукту;
5) Можливість обрати декілька продуктів;
6) Виводиться зображення продукту;
7) Виводиться назва продукту;
8) Виводиться ціна продукту;
9) Виводиться кількість продукту;
10) При натисканні на елемент переходи до сторінки редагування продукту.

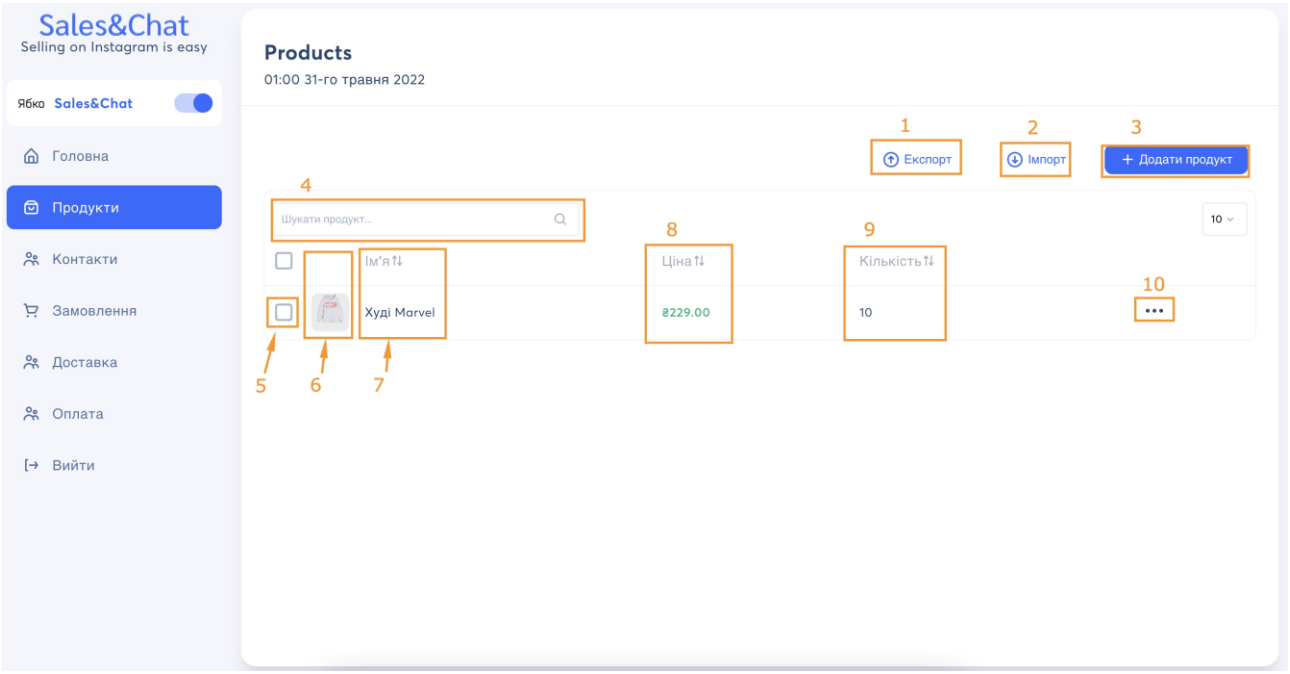

Рисунок 4.3 – Сторінка продуктів

На рис. 4.4 зображено сторінку продукту, де йому виводиться наступні елементи:

1) Елемент для завантаження зображення;
2) Поле для вводу ім’я продукту;
3) Поле для вводу артикля продукту;
4) Поле для вводу опису продукту;
5) Поле для вводу ціну на продукт;
6) Вибір статусу продукту;
7) Можливість добавляти мітки для продукту;
8) Зв’язую публікації із продуктом;
9) Поле для вказання назви нового параметру;
10) Можливість змінити порядок параметрів;
11) Видалити параметр;
12) Додати значення для параметру;
13) Додати новий параметр;
14) Скасувати зміни;
15) Зберегти продукт.

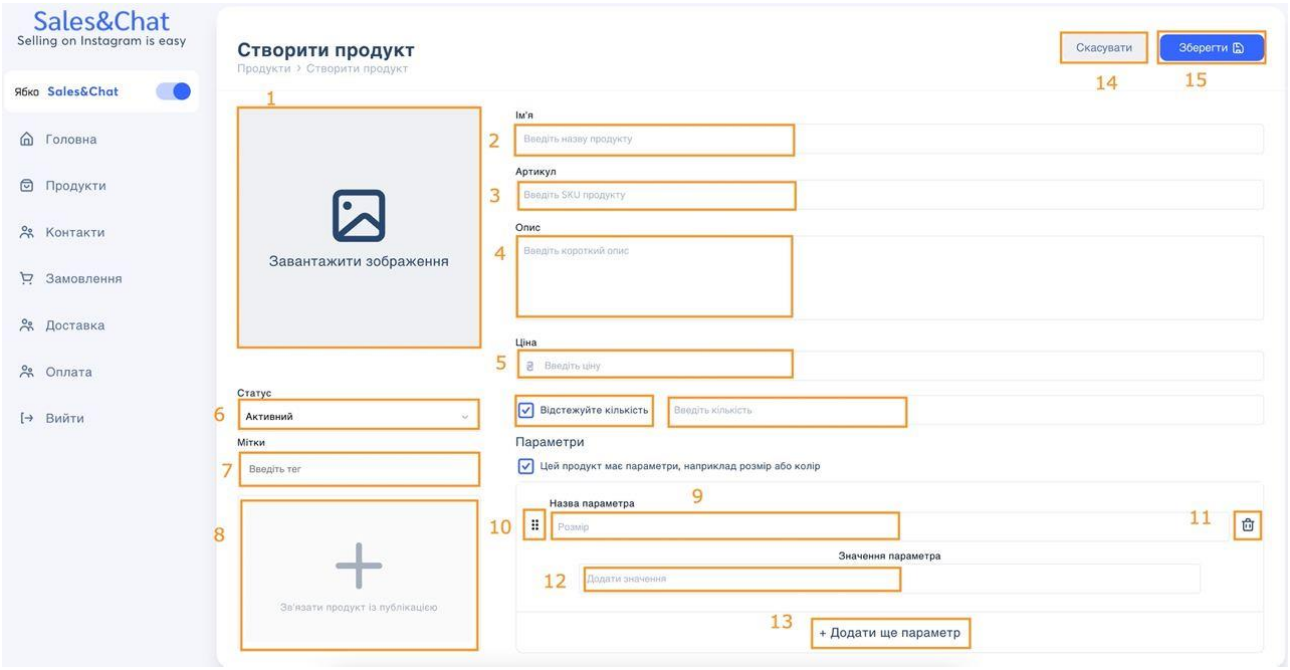

Рисунок 4.4 – Сторінка продукту

4.4 Сторінка контактів

Якщо в меню натиснути на «Контакти», то користувач попадає на сторінку контактів рис. 4.5 , де йому виводиться наступні елементи:

1) Поле для пошуку за ім’я контакту;
2) Можливість обрати декілька контактів;
3) Виводиться зображення контакту;
4) Виводиться ім’я контакту;
5) Виводиться кількість підписників контакту;
6) При натисканні на елемент можливість видали контакт.

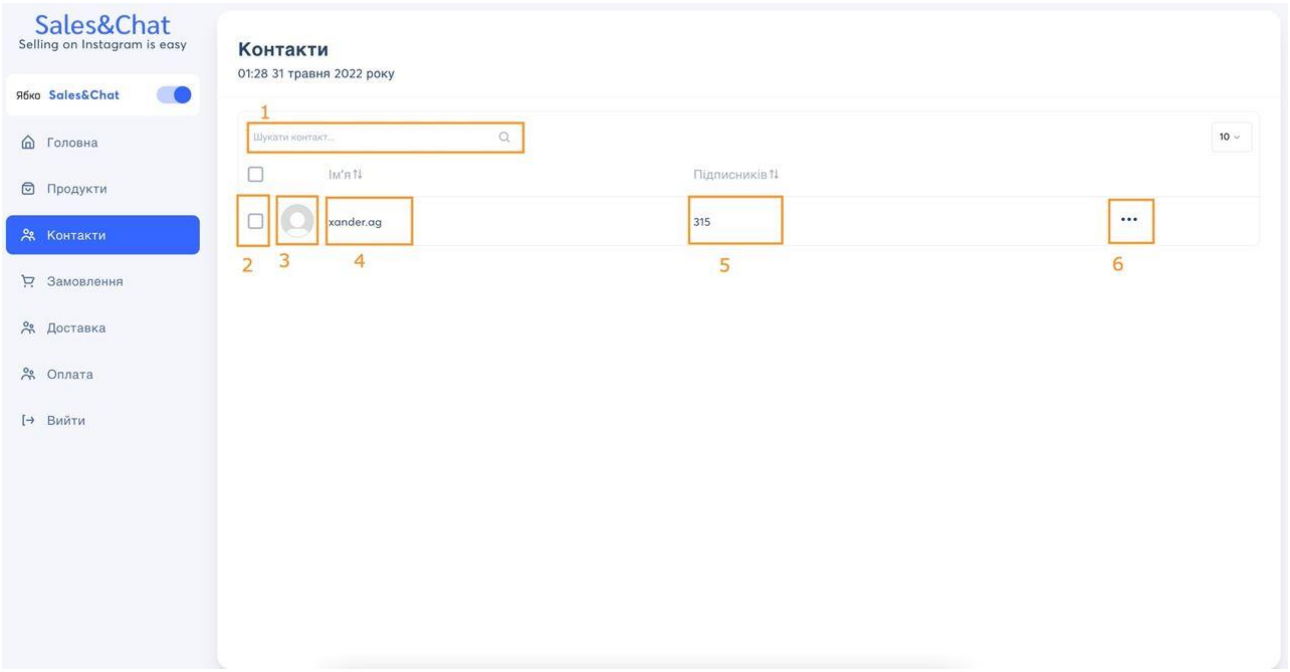

Рисунок 4.5 – Сторінка контактів

4.5 Сторінка замовлень

Якщо в меню натиснути на «Замовлення», то користувач попадає на сторінку замовлення рис. 4.6 , де йому виводиться наступні елементи:

1) Можливість обрати декілька контактів;
2) Виводиться унікальний ідентифікатор замовлення;
3) Виводиться дата замовлення;
4) Виводиться ім’я контакту;
5) Виводиться сума замовлення;
6) Виводиться статус замовлення;
7) Виводиться кількість продуктів в замовленні;
8) Виводиться метод доставки замовлення;
9) При натисканні на елемент можливість змінити статус замовлення.

Рисунок 4.6 – Сторінка замовлення

4.6 Сторінка методів доставки

Якщо в меню натиснути на «Доставка», то користувач попадає на сторінку «Доставка» рис. 4.7 , де йому виводиться наступні елементи:

1) Перемикач для увімкнення або вимкнення методу доставки;
2) Поле для вказані міста доставки;
3) Поле для вказання ціни доставки;
4) Кнопка для додавання ще міст.

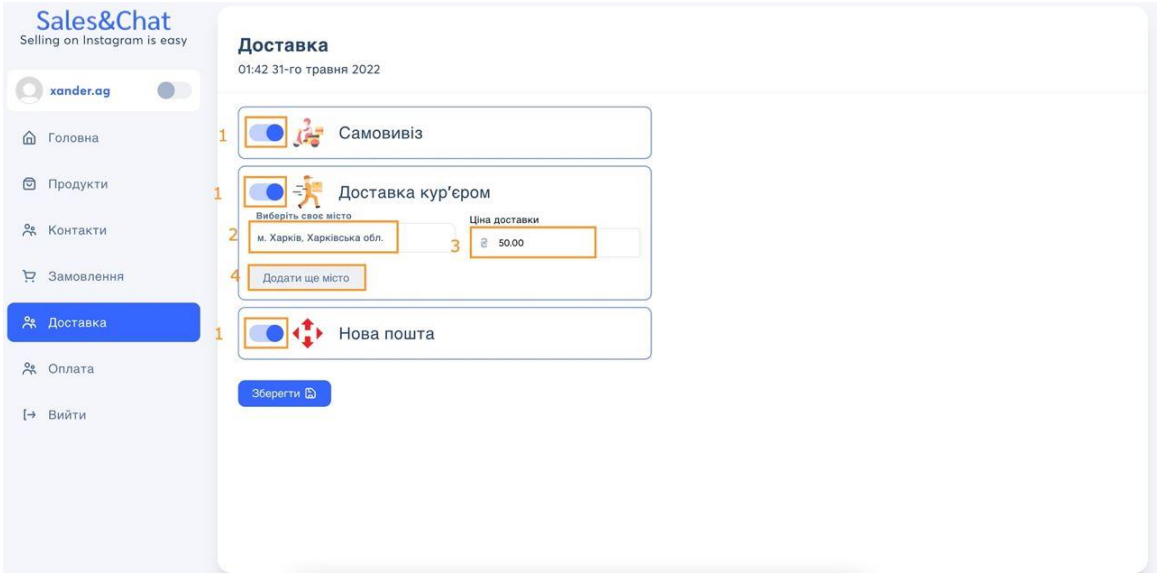

Рисунок 4.7 – Сторінка «Доставка»

4.7 Сторінка методів оплати

Якщо в меню натиснути на «Оплата», то користувач попадає на сторінку «Оплата» рис. 4.8 , де йому виводиться наступні елементи:

1) Перемикач для увімкнення або вимкнення методу оплати;
2) Поле для вказані реквізитів для оплати;
3) Кнопка для збереження змін.

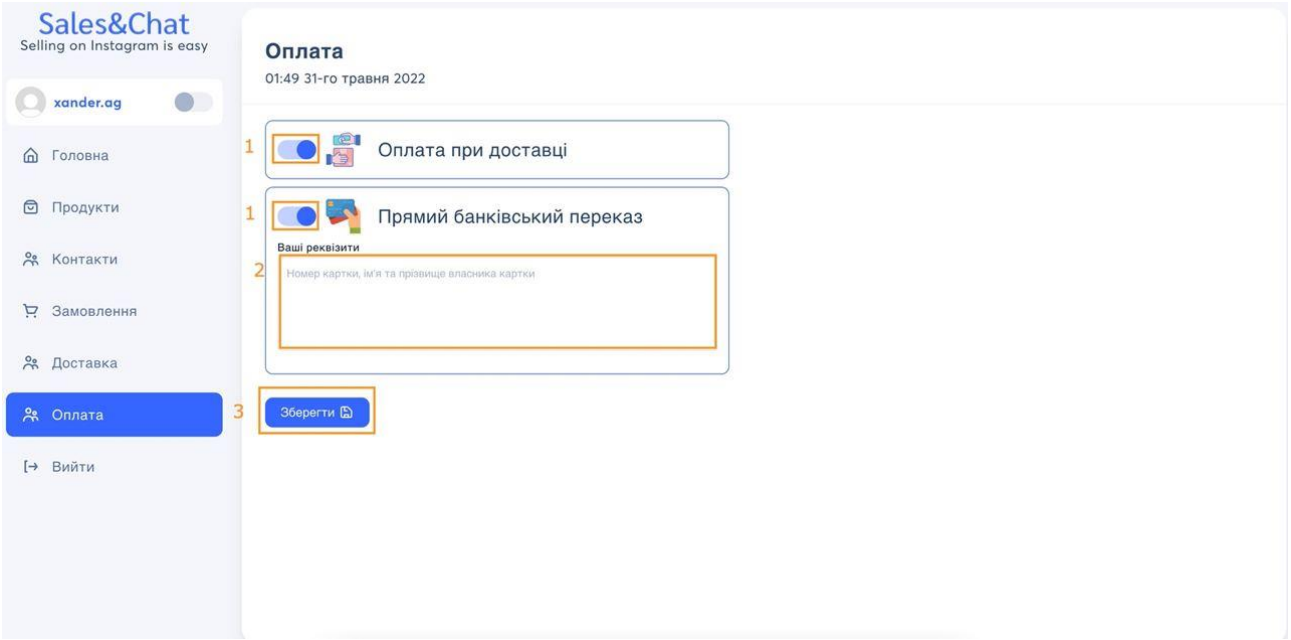

Рисунок 4.8 – Сторінка «Оплата»

5 ТЕСТУВАННЯ СИСТЕМИ

Ми зареєстрували у нашій панелі продавця, створили тестовий продукт с варіаціями, задали методи доставки, та задали методи оплата, увімкнули віртуального асистента.

Заходимо в додаток Instagram, переходимо до сторінку магазину зареєстрованого продавця, та натискаємо кнопку «написати» рис. 5.1 . Як бачимо нам відобразились вже дві кнопки, для допомоги покупцю.

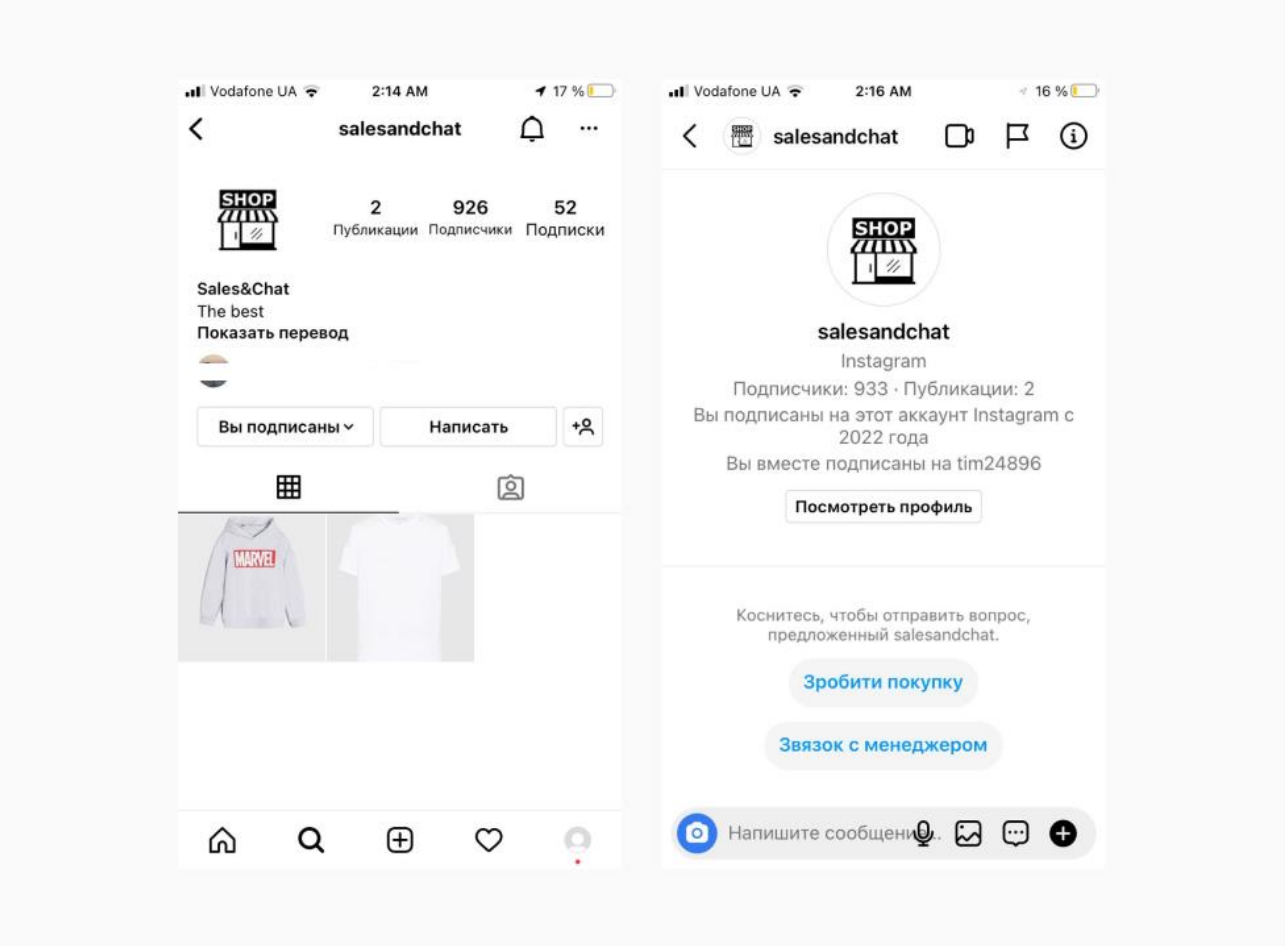

Рисунок 5.1 – Вхід у чат с магазином

Натискаємо на кнопку «Зробити покупку», після цього нас просять скину публікацію с продуктом, ми скидаємо публікацію, і отримує у відповідь шаблон продукту рис 5.2 .

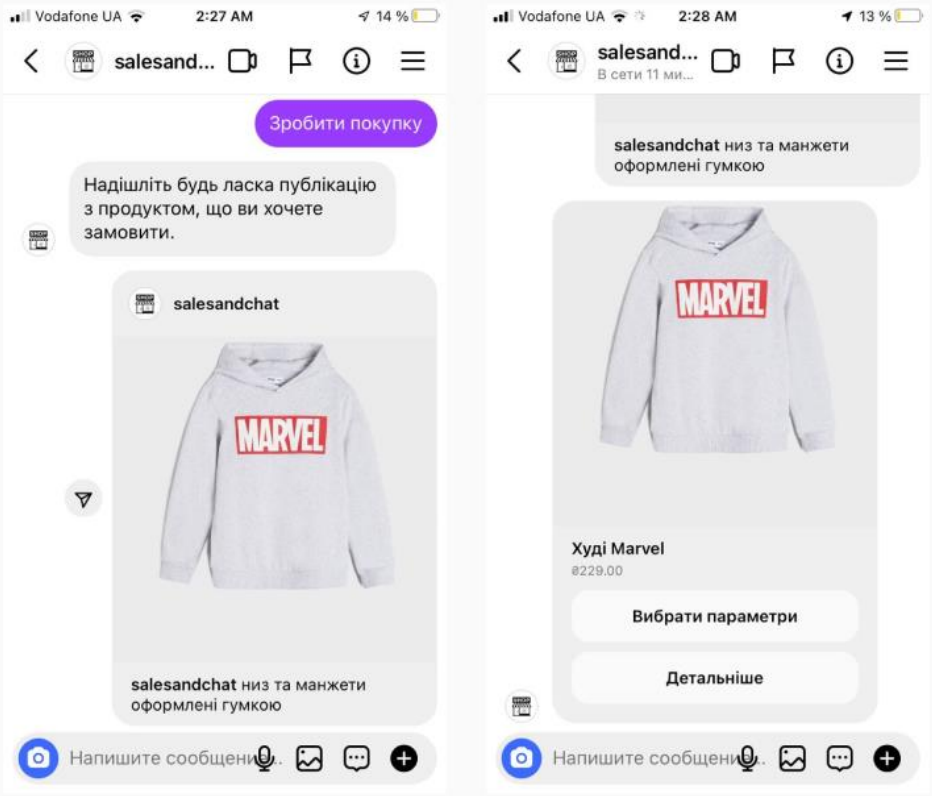

Рисунок 5.2 – Робимо покупку, вибираємо продукт

Натискаємо «Детальніше», нам надсилається опис продукту, та кнопка «Вибрати параметри», нам все подобається, тому нажимає «Вибрати параметри», після цього нам приходить назва параметру, та варіанти параметру рис 5.3 .

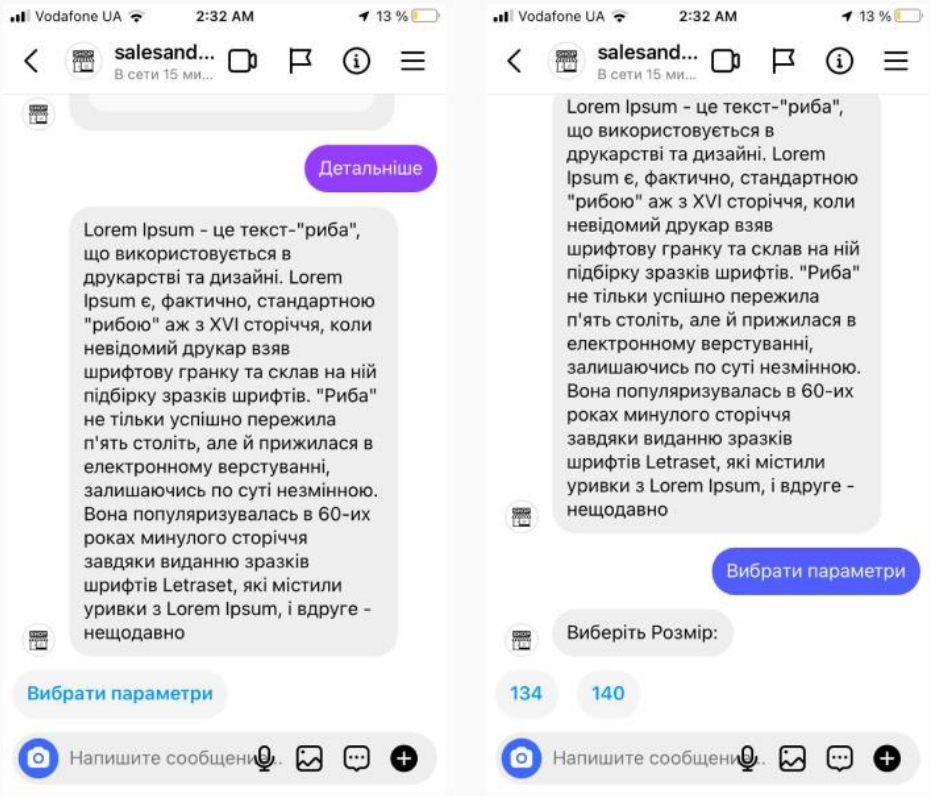

Рисунок 5.3 – Читаємо «Детальніше», та вибираємо параметри

Після вибору параметру, нам додало продукт в кошик, я перейшов у меню, та натиснув «Кошик», після чого прислали мені шаблон кошика, с даними кошика, та кнопка для взаємодії рис 5.4 .

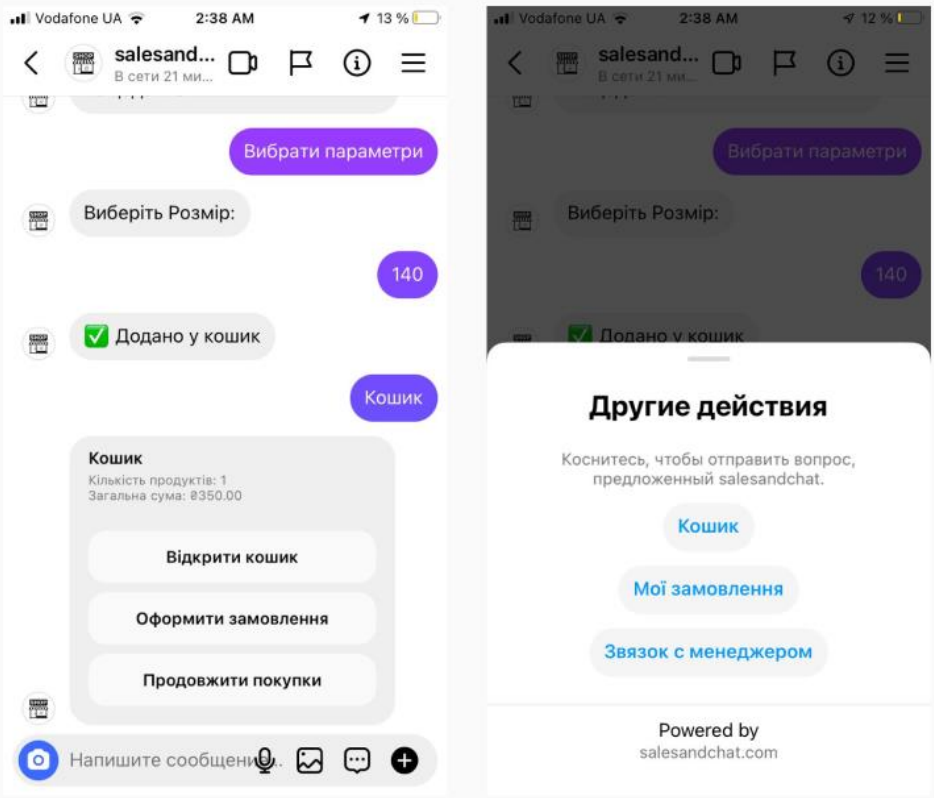

Рисунок 5.4 – Додавання та перегляд кошику

Натискаємо «Оформити замовлення» переходимо по унікальному посиланню. По посиланню відкривається сторінка с моїм кошиком, на якій можливо змінювати кількість продуктів, та видаляти продукту із кошику, ми змінюймо кількість продуктів, сума загальна змінилась рис. 5.5 .

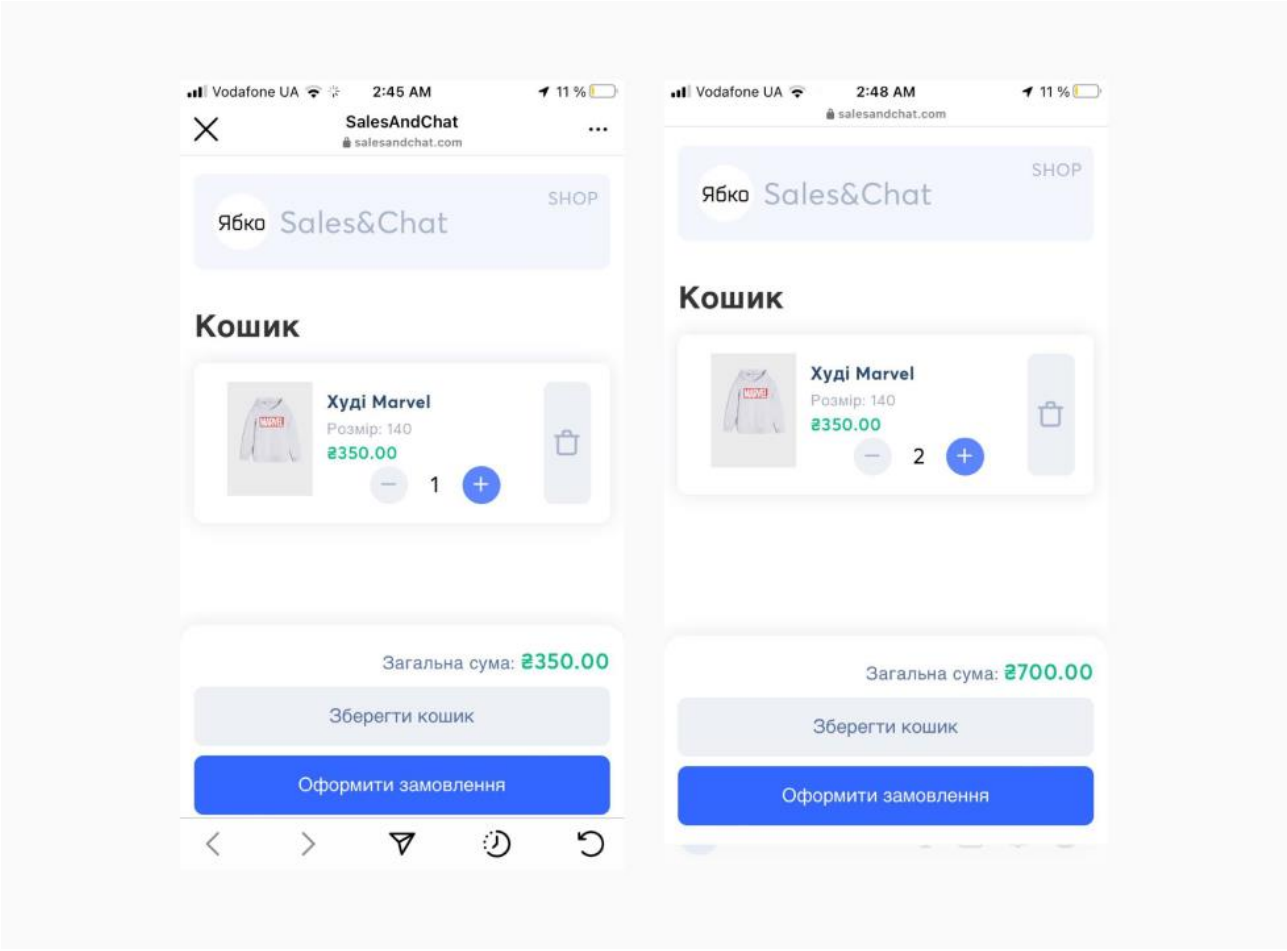

Рисунок 5.4 – Сторінка кошик

Далі ми натискаємо кнопку «Оформити замовлення», переходимо на форму «Оформити замовлення», вказуємо персональні данні, вибираємо метод доставки, та метод оплати рис 5.5 .

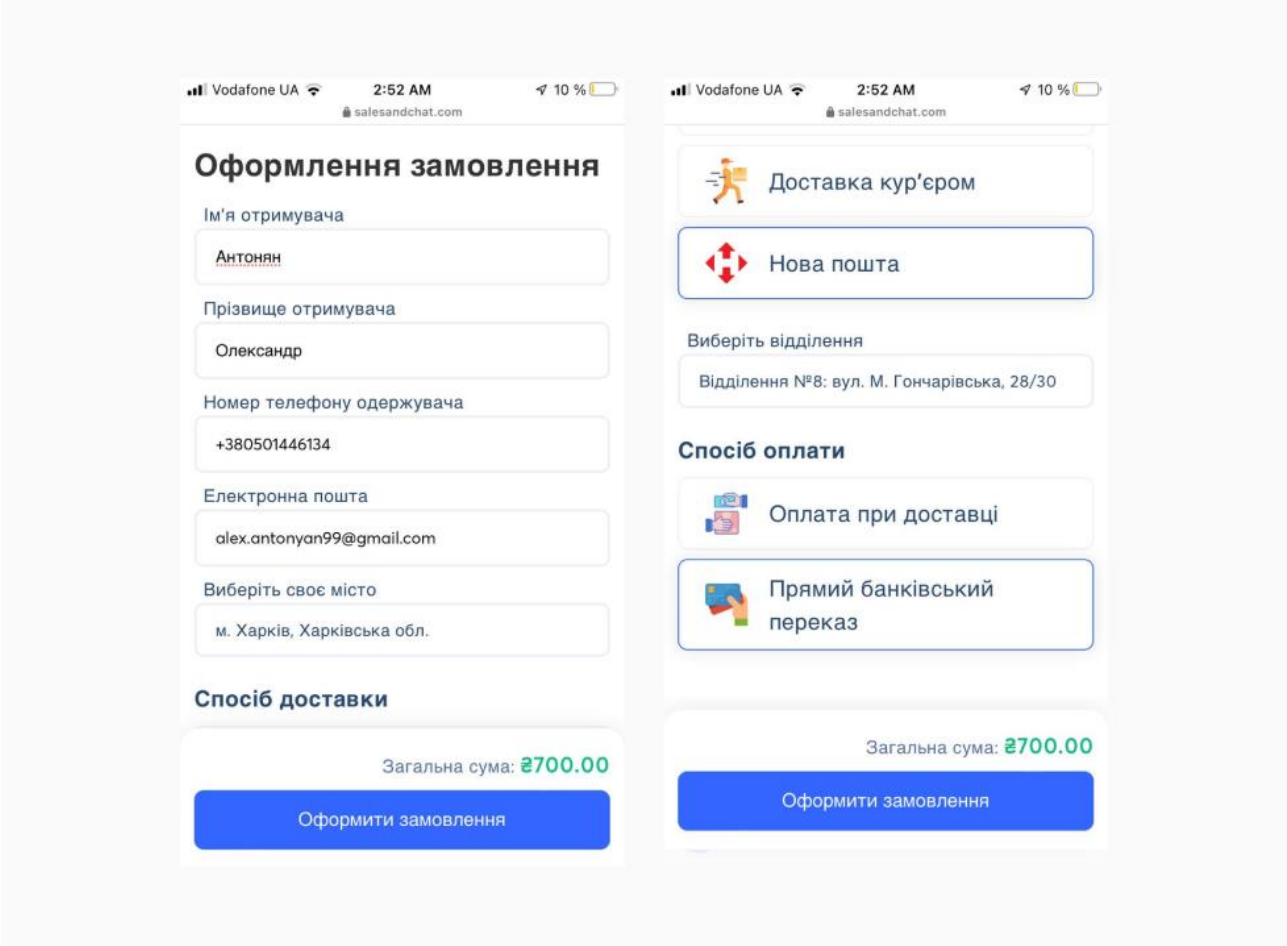

Рисунок 5.5 – Оформлення замовлення

І нарешті ми натискаємо «Оформити замовлення», і получаємо повідомлення, що замовлення успішно оформлено, а так як ми вибрали метод оплати «Прямий банківський переказ», то в повідомленнях в instagram, прийшли реквізити для оплати рис 5.6 .

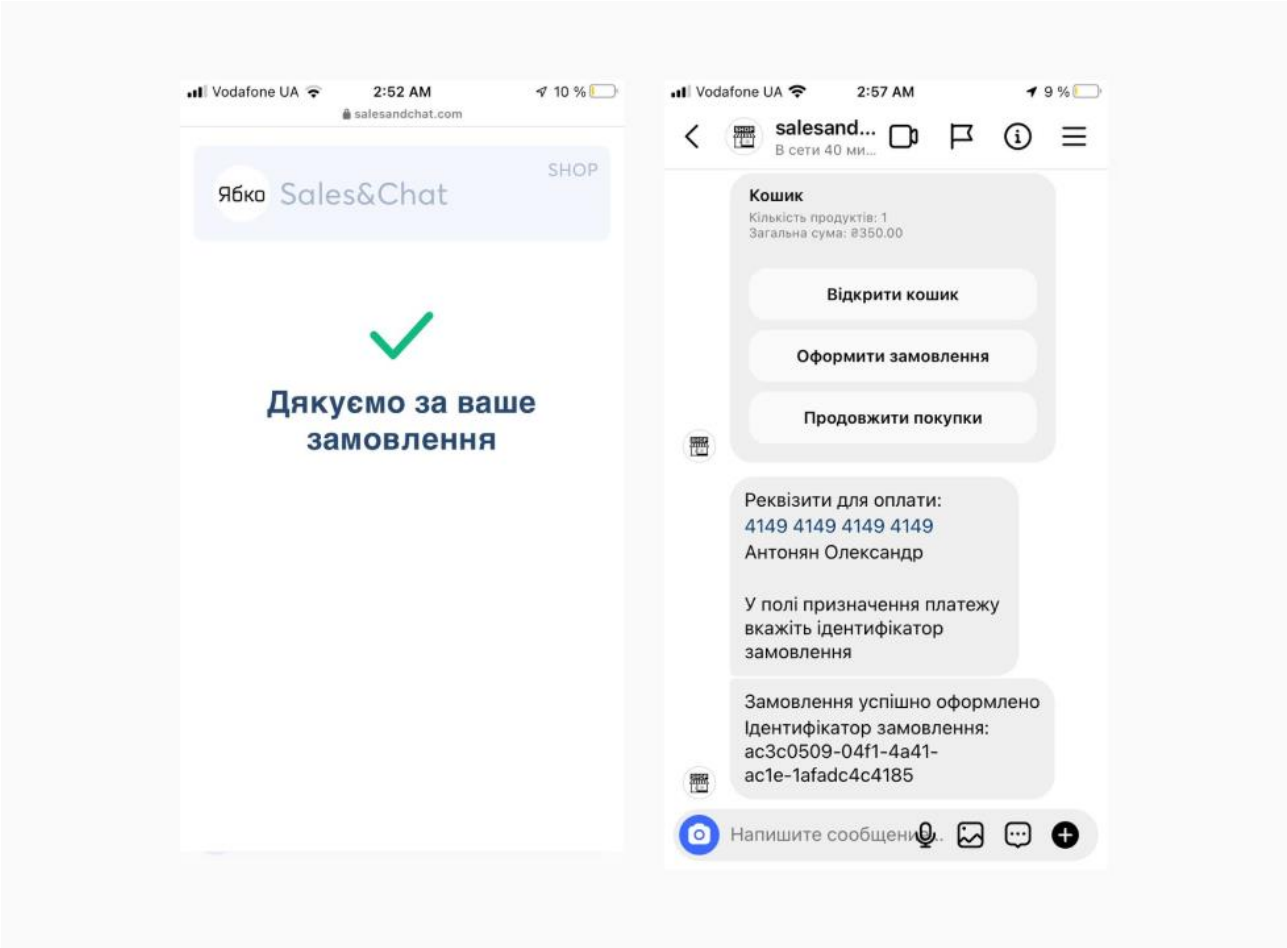

Рисунок 5.6 – Успішне оформлення замовлення

Замовлення оформлено, тепер перейдемо до панелі продавця, і перевіримо нове замовлення. Так замовлення оформлено, усі данні вірні рис. 5.7 .

Рисунок 5.7 – Замовлення в панелі продавця

Тестування успішно пройдено, помилок не виявлено, сервіс робить коректо.

ВИСНОВКИ

На даний момент спосіб взаємодії продавця і покупця в інтернет магазинах в «Instagram» довільно трудомісткий процес, так як запитів на оформлення замовлення можете буду безліч, і один менеджер с цим не впораєтеся над-то швидко, і для прискорення обробки запитів, добавляються велика кількість менеджерів, що не завжди є оптимально.

В роботі було розглянуто ряд питань, що стосуються покращення взаємодії між покупцем та продавцем, та збільшити ефективність, без збільшення витрат.

В першому розділі кваліфікаційної роботи проаналізовано основні можливості для створення чат-ботів в соціальної мережі «Instagram», які типи повідомлень ми можемо відправляти, та отримувати.

У другому розділі ми проектували базу даних для нашого сервісу, який має можливості, збереження даних о продавці, збереження даних о покупці, збереження даних о продуктах, збереження даних о кошиках, та збереження даних замовлень .

Третій розділ присвячено питанням створенні внутрішньої логіки сервісу, а саме, процеси авторизації та реєстрації, створенню та оновлень продуктів, створення та оновлення контактів, взаємодія покупця с кошиком, та процес оформлення замовлення.

В четвертому розділі ми створили SPA додаток, для продавців, через який він може, взаємодіяти с функціями сервісу.

В п’ятому розділі ми проводили тестування з боку покупця, і всі етапи, без помилок виконались, та без втручання продавця.

В роботі підкреслено важливість розробки чат-ботів, для оптимізації процесів. В умовах конкурентної боротьби за клієнта на перший план виходить індивідуальний підхід до кожного, здатність знайти найбільш раціональний баланс інтересів за допомогою програми лояльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 19 New E-Commerce Statistics You Need to Know in 2022 - Режим доступу до ресурсу: https://www.drip.com/blog/e-commerce-statistics.
2. API Messenger для Instagram - Режим доступу до ресурсу: https://developers.facebook.com/products/messenger/messenger-api-instagram/.
3. Oracle What is a chatbot? - Режим доступу до ресурсу: https://www.oracle.com/chatbots/what-is-a-chatbot/#link1.
4. What is a REST API? - Режим доступу до ресурсу: https://www.redhat.com/en/topics/api/what-is-a-rest-api.
5. Laravel - Режим доступу до ресурсу: https://laravel.com/.
6. Facebook Messenger - Режим доступу до ресурсу: https://developers.facebook.com/products/messenger/.
7. React documentation - Режим доступу до ресурсу: https://reactjs.org/docs/getting-started.html.