МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна система керування продажами онлайн магазину «Чай Чорногір'я»»
здобувача групи ІН – 02 Горпинченка Андрія Васильовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Андрій ГОРПИНЧЕНКО
(підпис)

Керівник доцент, к.ф.-м.н., доцент Галина ОЛЕКСІЄНКО
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЄХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Горпинченко Андрій Васильович

1. Тема роботи: « Інформаційна система керування продажами онлайн магазину «Чай Чорногір'я»»
затверджую наказом по СумДУ від *22 квітня 2024 року № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 29 травня 2024 року*

3. Вхідні дані до кваліфікаційної роботи Телеграм бот як метод електронної комерції
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Оцінка предметної області по обраній темі/
2) Вибір технологій для реалізації бота для продажу чайної продукції.
3) Моделювання системи.
4) Програмна реалізація за допомогою обраних технологій.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, постановка задачі, аналіз аналогів, технологііі структура проєкту, структура бази даних, схема меню.
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024 р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Оцінка предметної області	06.05.24-12.05.24	
2	Вибір технологій для реалізації бота для продажу чайної продукції	12.05.24-15.05.24	
3	Моделювання системи	15.05.24-20.05.24	
4	Програмна реалізація за допомогою обраних технологій	21.05.24-29.05.24	
5	Оформлення пояснювальної записки до кваліфікаційної роботи	29.05.24-31.05.24	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 68 стр., 29 рисунків, 3 додатоки, 28 використаних джерел.

Обґрунтування актуальності теми роботи – у сучасному цифровому суспільстві технологічний прогрес змінив спосіб взаємодії бізнесу зі споживачами. Інтернет та соціальні мережі стали потужними інструментами для залучення, продажу та купівлі клієнтів. Однією з інноваційних технологій, яка допомагає бізнесу стати ближчим до споживачів, є використання чат-ботів.

Об'єкт дослідження — телеграм-бот та його можливість інтеграції в електронну комерцію.

Предмет дослідження — Процес автоматизації продажів та обслуговування клієнтів за допомогою Telegram-бота.

Мета роботи — розробка та впровадження Telegram-бота, який допоможе автоматизувати продажі, покращити обслуговування клієнтів та надати їм зручний засіб для здійснення покупок.

Методи дослідження — аналіз літератури та інших джерел, конкурентних додатків, моделювання систем, технології по розробці Телеграм-ботів в бізнесі для продажу онлайн продукції.

Результати — проведено інформаційний огляд по даній темі та проаналізовано додатки аналоги. Вибрано найкращі методи для розв'язання поставленої задачі. В результаті чого було розроблено інформаційну систему, для замовлення чайної продукції та підтримки онлайн-продажів зі зручним інтерфейсом для адміністраторів із можливостями ведення записів та керування продуктами. Що наддасть можливість автоматично працювати з клієнтами через інтерфейс та зручно працювати адміністраторам з замовленнями.

ІНФОРМАЦІЙНА СИСТЕМА, ОНЛАЙН-ПРОДАЖІ, TELEGRAM, БОТИ, AIOGRAM, POSTGRESQL, PYTHON, SQLALCHEMY.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми. Сучасний технологічний прогрес суттєво змінив спосіб взаємодії бізнесу зі споживачами. Інтернет та соціальні мережі стали потужними інструментами для залучення, продажу та купівлі клієнтів. Однією з інноваційних технологій, яка допомагає бізнесу стати ближчим до споживачів, є використання чат-ботів.

Актуальність. Використання Telegram-ботів в електронній комерції та обслуговуванні клієнтів стає все більш популярним. Боти можуть автоматизувати процес продажу, надавати інформацію про товари та послуги в режимі реального часу, а також взаємодіяти з клієнтами через популярні платформи для обміну повідомленнями, такі як Telegram.

Об’єкт дослідження. телеграм-бот та його можливість інтеграції в електронну комерцію.

Предмет дослідження. Процес автоматизації продажів та обслуговування клієнтів за допомогою Telegram-бота.

Гіпотеза. Використання Telegram-бота для продажу чайної продукції може підвищити ефективність роботи, скоротити час обслуговування клієнтів та підвищити їхню задоволеність.

Новизна. Розробка Telegram-бота для продажу чайної продукції з використанням мови програмування Python та бібліотеки Aiogram є інноваційним рішенням, яке може надати клієнтам зручні та персоналізовані послуги, а компаніям - допомогти збирати та аналізувати дані про вподобання клієнтів для оптимізації роботи.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв'язання задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Веб-застосунки для продажів

Паралельно з розвитком веб-технологій починає розвиватися ніша онлайн-продажів. Веб-додатки стали невід'ємною частиною бізнесу в сучасному світі. За допомогою веб-додатків підприємства тепер можуть легко розвиватися і набагато швидше досягати своїх цілей. Ці програми допомагають охопити велику кількість клієнтів одночасно. Організації швидко освоюють цей аспект Інтернету, створюючи веб-додатки за допомогою розробників для задоволення потреб свого бізнесу. Веб-додатки важливі з кількох причин.

1) Реклама та брендинг: компанії більше не можуть розраховувати на збільшення частки ринку, якщо у них немає відповідного веб-додатка. Хоча великі компанії можуть дозволити собі утримувати власні команди розробників для таких цілей, малий бізнес може звернутися до веб-розробників, щоб отримати ті самі переваги за менших витрат.
2) Підтримка клієнтів: веб-додатки також надають можливості для покращення підтримки клієнтів. Хороший додаток може стати першою лінією зв'язку між потенційним клієнтом та компанією. Краса такого додатка полягає в тому, що до нього можна отримати доступ у будь-який час. З їх допомогою навіть місце більше не є обмеженням. Звичайно, тільки якісна компанія з веб-розробки може надати таку можливість своїм додатком.
3) Конкурентна перевага: ці додатки можуть стати важливим інструментом залучення клієнтів, оскільки в поточній ситуації в діловому світі стає все більш важливим наявність веб-додатків, присвячених бізнесу організації. Розробляючи додатки для IOS та Android, компанії можуть використовувати смартфони для виведення своїх продуктів на ринок з величезною конкурентною перевагою.[1]

Для того щоб розпочати бізнес у сфері електронної комерції, насамперед потрібно вирішити, яким типом електронної комерції збираємося займатися і який тип інтернет-магазину хочемо відкрити.

Для цього давайте розберемося які є платформи для продажу продукції:

1) Дошки оголошень: хороший спосіб продати невелику кількість товарів або протестувати нішу. Тут також легко створювати та розміщувати оголошення. Для того щоб продавати товари, не потрібно створювати юридичну особу або офіційно ставати підприємцем. Оплата від покупців може прийматися банківською картою або післяплатою, хоча це не рекомендується.
2) Маркетплейси: веб-сайт або додаток, який полегшує здійснення покупок з багатьох різних джерел. Оператор маркетплейсу не володіє жодним товаром, його бізнес полягає в тому, щоб представити користувачеві чужі товари та сприяти здійсненню транзакції. eBay є найкращим прикладом онлайн-маркетплейсу, вони продають все і всім.[2]
3) Власний веб-сайт: платформа для демонстрації особистого бренду і завоювання довіри аудиторії. Це місце, де можна розповісти свою історію, продемонструвати свої навички та досягнення і надати аудиторії цінний контент. Персональний сайт дає змогу контролювати свою присутність у мережі. Він також може дати професійну перевагу перед тими, у кого немає особистого сайту.
4) Соціальні мережі: Ще одне місце, де можна продавати свої товари онлайн, - це соціальні мережі. Не потрібно знати технічні аспекти, необхідні для створення інтернет-магазину. Достатньо просто створити акаунт в соціальный мережі і почати продавати.
5) Месенджери: Продажі через месенджери - це використання платформ для обміну повідомленнями, таких як Facebook Messenger, Telegram або інших чат-додатків, для здійснення торгових операцій, спілкування в реальному часі та встановлення

персоналізованих зв'язків з потенційними клієнтами та покупцями.

Обираючи платформу для продажу товарів онлайн, важливо ретельно вивчити особливості кожного варіанта і врахувати цілі та потреби вашого бізнесу. Який би напрямок ви не обрали, успішний бізнес у сфері електронної комерції вимагає не тільки вибору правильної платформи, а й ефективного маркетингу, обслуговування клієнтів і постійного вдосконалення.[3]

1.2 Маркетинг в Telegram

Через величезну кількість оголошень, публікацій, рекламних кампаній та анкет, спрямованих на підвищення обізнаності про бренд та залучення трафіку на веб-сайти, конкуренція настільки висока, що новачки змушені шукати альтернативні канали для охоплення своєї цільової аудиторії. Це гарна ідея, багато з них знаходять таку можливість в месенджері, який розвивається навіть швидше, ніж соціальні мережі минулого. Telegram на сьогоднішній день є одним з найбільш швидкозростаючих новачків і вже встиг завоювати свої позиції в таблиці лідерів.[4]

Telegram - це програма для обміну миттєвими повідомленнями зі швидко зростаючою базою користувачів. Незважаючи на те, що він відносно новий, це один з найпопулярніших месенджерів у світі.[5]

Перш за все, варто знати, що в Telegram немає реклами. Така офіційна політика розробників, яка вже привернула величезну кількість прихильників, які відразу ж перейшли в Telegram з усіх інших месенджерів. Таким чином, можна бути в курсі останніх інтернет-тенденцій і отримати активну аудиторію, яка не прив'язана до старомодної настирливої реклами. Це клієнти, які вже відмовилися від покупки у конкурентів за допомогою спонсорської реклами і можуть залучити їх за допомогою фантазії і нестандартних рішень.[4]

Він ідеально підходить для маркетингових цілей, включаючи групові чати, обмін медіа та підтримку ботів. Крім того, Telegram є абсолютно

безкоштовним, що робить його надзвичайно економічно ефективним способом охоплення потенційних клієнтів.

Коли справа доходить до маркетингу в Інтернеті, для цього є багато різних інструментів і платформ. Деякі з них, такі як Facebook або Twitter, добре зарекомендували себе і відносно прості у використанні. Однак, якщо потрібен додаток для обміну повідомленнями, який надає додаткові функції для бізнесу та маркетологів, Telegram може бути правильним вибором.

1.3 Сервіси для маркетингу в Telegram

За допомогою Telegram можна створювати групи та канали для обговорення ваших продуктів і послуг з потенційними клієнтами або покупцями. Ви також можете інтегрувати ботів у свій обліковий запис Telegram для автоматизації певних завдань, заощаджуючи час і зусилля в процесі.

Завдяки потужним функціям безпеки та хмарній інфраструктурі Telegram чудово підходить для захисту конфіденційної інформації, дозволяючи легко охопити велику аудиторію.

Телеграм-канал - це платформа, де можна транслювати повідомлення великій аудиторії без обмежень щодо кількості учасників каналу. Цей інструмент ідеально підходить для організацій, підприємств та громадських діячів, які хочуть миттєво ділитися новинами, оновленнями чи вмістом із широкою аудиторією. Канал Telegram є універсальним, оскільки підтримує різні форми вмісту, включаючи текст, зображення, відео, файли та опитування. Крім того, менеджери каналів можуть отримати доступ до інструменту аналітики для відстеження взаємодії та активності повідомлень, щоб вони могли оптимізувати свої комунікаційні стратегії.

Лише адміністратори можуть публікувати повідомлення в каналі, але кожен учасник отримує сповіщення про кожну нову публікацію. Це робить його ідеальним для обміну анонсами, оновленнями або просто тим, що ви хочете сказати.[6]

А оскільки учасники каналу не бачать один одного і адміністраторів, це також чудовий спосіб зберегти анонімність вашої аудиторії.

Групи в Telegram, що налічують до 200 000 учасників, ідеально підходять для бізнесу та організацій, які шукають спосіб спілкування з великою кількістю людей.

Учасники можуть надсилати повідомлення, фотографії, відео та голосові записи, що робить групи Telegram універсальним каналом комунікації. Групи можуть бути як публічними, так і приватними.[7]

Групи в Telegram є чудовим маркетинговим інструментом, оскільки дозволяють бізнесу легко охопити велику аудиторію. Створюючи публічну групу, підприємці можуть полегшити потенційним клієнтам пошук і приєднання до своєї групи.

За допомогою приватної групи компанії можуть контролювати, хто має доступ до їхньої групи, і використовувати посилання-запрошення як спосіб викликати інтерес до своїх продуктів або послуг. Так чи інакше, групи є одним з найпотужніших маркетингових каналів.

Кожному каналу та групі "потрібен" чат-бот, щоб переконатися, що він функціонує належним чином. З тисячами користувачів адміністратори не можуть зробити так багато.

Чат-боти в Telegram - це зручні маленькі програми, які можуть допомогти вам у виконанні різноманітних завдань: від збору потенційних клієнтів до надання клієнтської підтримки або навіть розсилки новин існуючим клієнтам.

BotFather слугує шаблоном для всіх чат-ботів Telegram. Можна модифікувати його, щоб створити бота, який точно відповідає поставленому завданню. Чат-боти корисні тим, що вони автоматизують деякі взаємодії, які інакше вам довелося б виконувати вручну.

Використання бота економить час та енергію і може допомогти забезпечити кращий досвід для клієнтів. Крім того, чат-боти можуть

допомогти проводити маркетингові кампанії або збирати відгуки від ваших користувачів

1.4 Аналіз аналогів

@MagicCoffee_bot – MagicCoffee — інтернет-магазин оптового та роздрібного продажу кави, чаю та аксесуарів. Цей бот повторює функціонал інтернет-магазину компанії. Онлайн-оплата замовлень чат-ботами з використанням платіжних чат-ботів, синхронізація даних з базою даних сайту і інтеграція з зовнішньої crm-системою компанії[8] рис1.1.

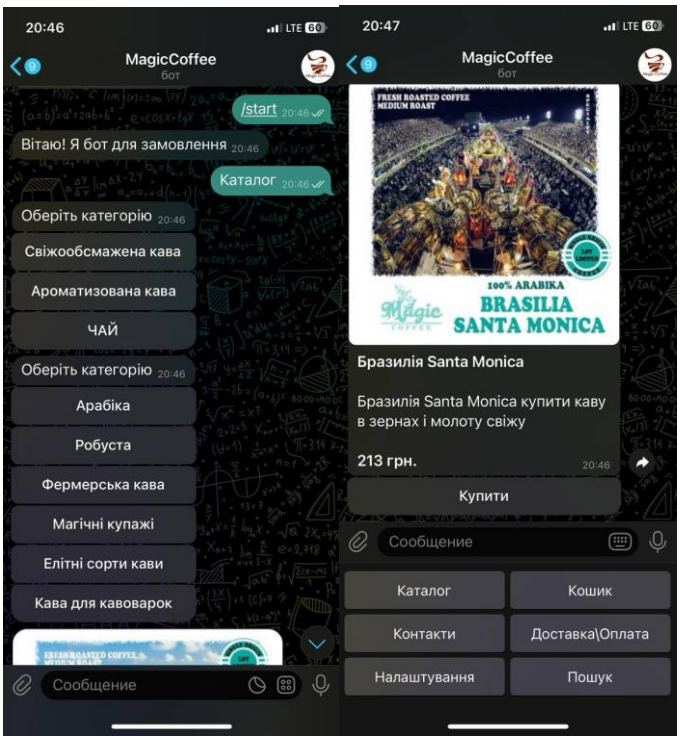

Рисунок 1.1 – Інтерфейс MagicCoffee бота

@railwaybot - Цей бот дуже зручний для тих, хто часто користується послугами "Укрзалізниці". Бот здійснює пошук залізничних квитків по Україні і показує розклад поїздів за номерами. Його особливістю є моніторинг квитків за вказаним напрямком з урахуванням обраних параметрів кожні 5 хвилин. За допомогою @railwayboy ви також можете приступити до процесу покупки

квитків через сервіс ПриватБанку.[9] Бот також розклад поїздів та автобусів між станціями рис1.2.

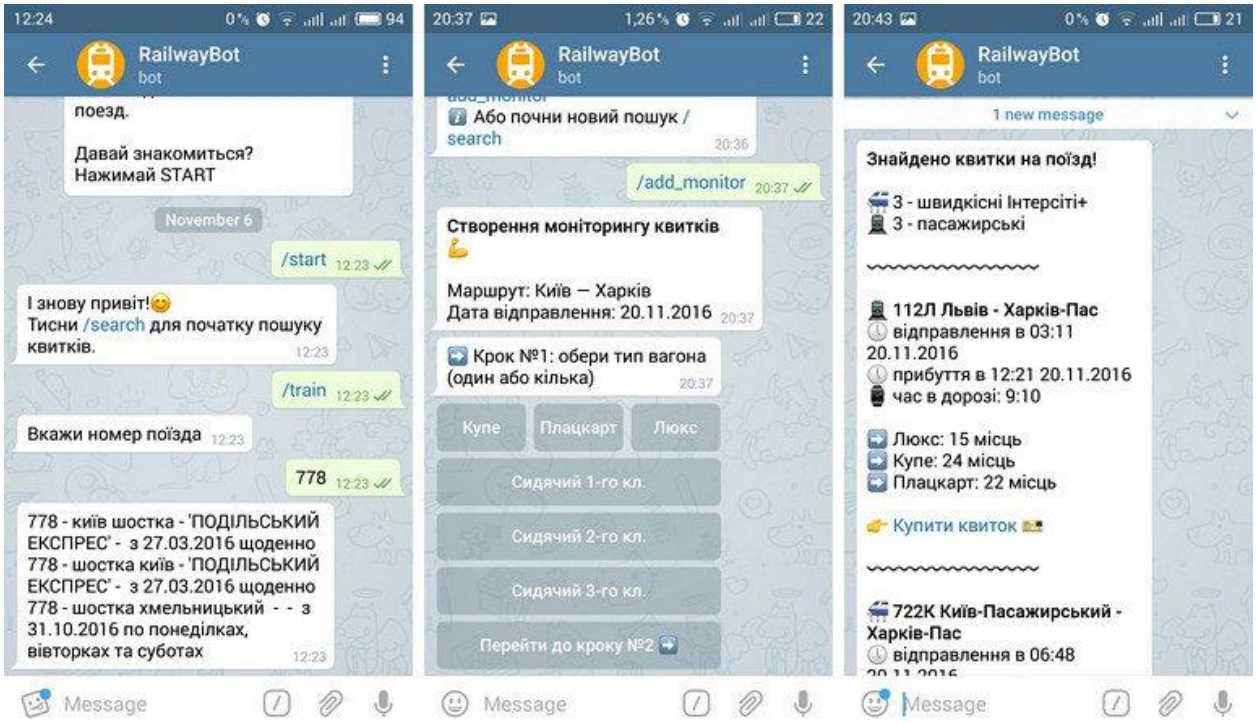

Рисунок 1.2 – Інтерфейс RailwayBot

@epicentrk_digitalbot - Хелп-бот від Епіцентр К – персональний помічник, який працює в зручних месенджерах. Як пояснюють розробники, за допомогою бота клієнт мережі «Епіцентр К» зможе керувати онлайн-замовленнями, шукати товар або перевіряти його ціну, переглядати історію покупок, отримувати якісну підтримку з будь-якого питання.

Help-бот значно розширює сервіси для офлайн-клієнтів мережі, і, окрім звернення до служби підтримки клієнтів та отримання професійної консультації щодо товарів та послуг від експертів, покупець зможе[10]:

- Звернутися на інформаційну стійку будь-якого торговельного центру
- Переглянути актуальний графік роботи торговельних центрів, знайти найближчі торговельні центри та прокласти маршрут до торговельного центру

- Відсканувати товар у торговому залі за допомогою чат-бота, не чекаючи на вільного продавця, дізнатися ціну та характеристики товару, переглянути відгуки про товар
- Дізнатися торгові центри, в яких товар є в наявності;
- Отримати картку "Вигода" або прив'язати існуючу картку до чат-бота в 1 клік, а також використовувати картку для сканування коду на касі;
- Отримати сповіщення про час, коли замовлення буде сформовано та підготовлено до доставки;
- Зареєструватися в електронній черзі для отримання онлайн-замовлення в центрі видачі замовлень без очікування в черзі;
- Продовжити термін зберігання онлайн-замовлення в пункті видачі замовлень;

Проаналізувавши вище написані аналоги було створено порівняльну таблицю:

Таблиця 1.1 Аналіз аналогів

Пункти	**@MagicCoffee_bot**	**@railwaybot**	**@epicentrk_digitalbot**
Призначення	Продаж кави, чаю та аксесуарів	Перегляд залізничних та автобусних квитків	Перегляд каталогу товарів Епіцентру
Категорія	Напої	Подорожі	Рітейл
Наявність корзини	Так	Ні	Так
Можливість покупок	Через платіжного чат-бота	На сайтах перевізників	На сайті компанії
Трекінг замовлення	Ні	Ні	Так
Інтеграція	MagicCoffee API	API ж/д компаній	Інтеграція з epicentrk API
Мова	Українська	Українська	Українська

1.5 Постановка задачі

Після проведеного аналітичного огляду, завданням цієї роботи є розробка та розгортання бота Telegram для підтримки й оптимізації процесу онлайн-продажів стартапів і компаній. Основна увага буде зосереджена на використанні програми обміну повідомленнями Telegram як платформи для ефективного бізнесу та продажів.

Завдання роботи:

1. Провести аналіз методів для створення чат-боту.
2. Прототипування функіональних можливостей боту.
3. Розробити функціонал Telegram-бота для продажу товарів і послуг.
4. Налаштувати взаємодію бота з користувачем з урахуванням функціоналу месенджера.
5. Розробити зручний інтерфейс для адміністраторів ботів, щоб відстежувати замовлення та керувати продуктами.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Вибір методу створення Чат-боту в Telegram

Створення телеграм-бота - це завдання, яке може бути виконане різними способами. Кожен з яких надає свій функціонал для розробки який має свої переваги з недоліками.

1. Використання Telegram Bot API:
   - Telegram надає спеціальний API для розробки ботів. Можна використовувати різні мови програмування, такі як Python, Node.js, Java, і багато інших, для взаємодії з цим API.
   - Після вибору мови програмування, використовуєте відповідну бібліотеку або клієнт Telegram Bot API для спрощення комунікації з API.
   - Цей метод надає повний контроль над ботом та дозволяє реалізувати різноманітні функції.
2. Використання платформ для розробки ботів:
   - Існують різні платформи, такі як Rowy, Dialogflow (від Google) і багато інших, які дозволяють створювати ботів без програмування.
   - Ці платформи надають графічний інтерфейс для налаштування бота та визначення його функціональності.
   - Вони можуть бути корисними для швидкого створення простих ботів, але можуть бути обмежені в можливостях порівняно з власною розробкою.
3. Використання бот-будівель:
   - Існують інструменти, які дозволяють створювати ботів без програмування, використовуючи візуальний інтерфейс.
   - Зазвичай це вимагає мінімальних навичок в програмуванні, і ви можете використовувати такі інструменти для створення ботів для різних цілей, таких як обслуговування клієнтів, автоматизація завдань тощо.

Після аналізу цих методів я зупинився на розробці через Telegram Bot API. Тому що він забезпечує високу гнучкість і широкий спектр можливостей для

інтерактивної взаємодії з користувачами. Таким чином, Telegram Bot API є потужним і корисним інструментом для створення інтерактивних рішень, здатних ефективно задовольняти потреби користувачів. Використовуючи його, можна швидко та ефективно реалізувати заплановані функції та забезпечити високий рівень взаємодії з користувачами.

2.2 Вибір мови програмування

Наступним кроком потрібно вибрати мову програмування для написання логіки чат бота. Вибираючи мову програмування для розробки ботів Telegram, слід враховувати кілька факторів, зокрема простоту використання, доступні бібліотеки та фреймворки, підтримку спільноти та особисті переваги[11].

Ось кілька популярних мов програмування, які зазвичай використовуються для створення ботів Telegram:

1) Python: Це одна з найпоширеніших мов програмування. Мова має простий синтаксис і легка для розуміння. Крім того, вона має велику бібліотеку для додатків ML і NLP, що спрощує процес створення чатботів.[12]
2) Java: Об'єктно-орієнтована мова загального призначення, що робить її ідеальною для програмування чат-ботів. Чат-боти, запрограмовані на Java, можуть працювати на будь-якій системі з встановленою віртуальною машиною Java (JVM). Мова також дозволяє використовувати багатопотоковість, що забезпечує кращу продуктивність, ніж інші мови програмування в цьому списку.
3) Ruby: Високорівнева об'єктно-орієнтована мова програмування. Мова підтримує динамічне програмування, що дає змогу змінювати код під час виконання програми в міру зміни потреб системи; як і Python, Ruby є дуже доброзичливою до користувача і має добре документовані бібліотеки ML і NLP.
4) C++: Одна з найшвидших мов програмування, що використовуються для створення чат-боті. Вона також підходить для високопродуктивних

чатботів. C++, з іншого боку, є низькорівневою мовою програмування і потребує певного досвіду для ефективного написання коду.[13] Крім того, у ній обмежена кількість високорівневих бібліотек ML і NLP, а це означає, що чатботи потрібно створювати з нуля.

Після огляду мов програмування було вибрано Python. Через те що він має простий і зрозумілий синтаксис і безліч бібліотек, які спрощують розробку. Python також відомий своєю великою спільнотою розробників та наявністю безлічі навчальних ресурсів, які швидко вирішують проблеми, які можуть виникнути в процесі розробки, а його інтеграція з іншими сервісами та гнучкість роблять його ідеальним вибором для створення складних та функціональних ботів.

2.3 Вибір інтегрованого середовища розробки

Після визначення мови програмування постає питання вибору середовища розробки або редактору коду. IDE та редактор коду-це інструменти, які розробники програмного забезпечення використовують для написання та редагування коду. Інтегроване середовище розробки, як правило, є більш функціональним і містить інструменти для налагодження, написання та розгортання коду. [14]

Редактор коду простіший і, як правило, орієнтований на редагування коду. Багато розробників використовують IDE та редактори коду залежно від своїх завдань.

PyCharm:

Це широке середовище розробки Python, створене JetBrains. Ця IDE підходить для професійних розробників і полегшує розробку великих проектів на Python.[14] Найбільш помітними функціями PyCharm є:

- Підтримка JavaScript, CSS та TypeScript
- Зручна навігація по коду
- Швидкий і безпечний рефакторинг коду

- Підтримка таких функцій, як доступ до бази даних безпосередньо з IDE
- Це хороше середовище розробки Python для Windows

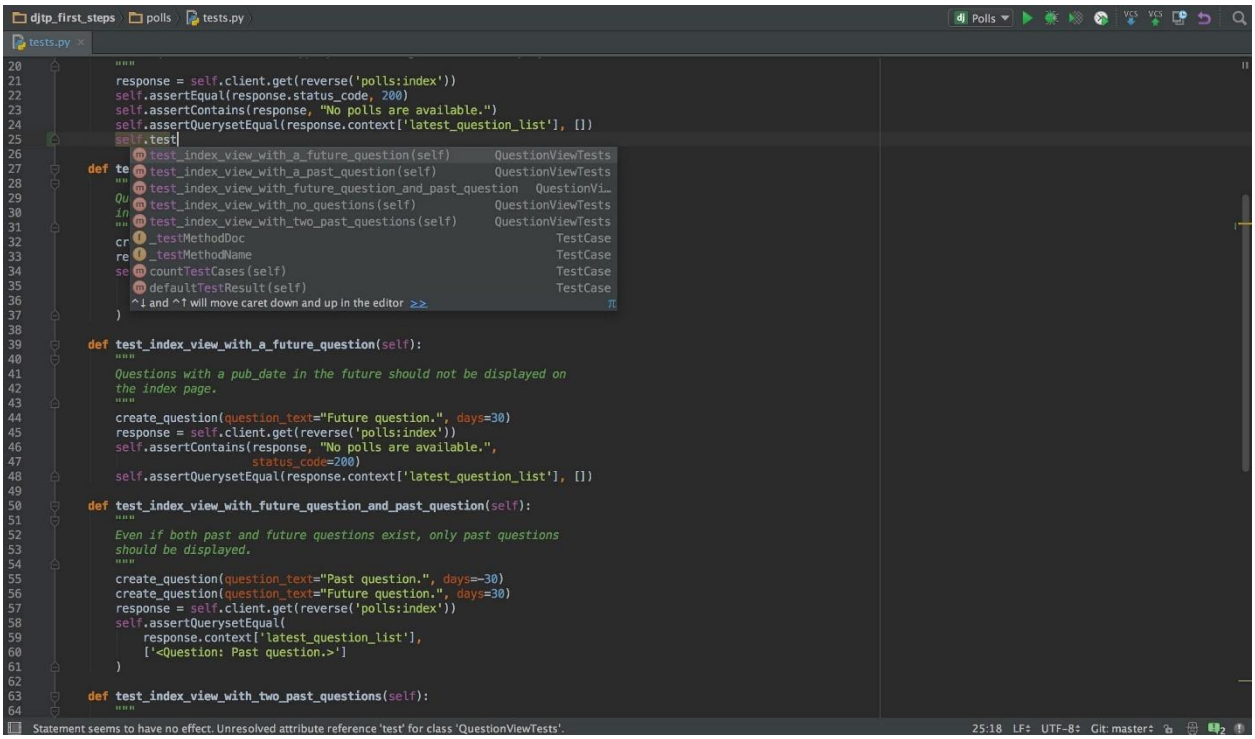

Рисунок 2.1 – Інтерфейс середовища розробки PyCharm

Visual Studio Code

Відкрите (і безкоштовне) середовище розробки, створене Microsoft. Вона широко використовується при розробці на Python

VS Code-це легке та потужне середовище розробки, яке пропонують лише деякі платні Ide.[14] Найбільш помітними особливостями Visual Studio code є:

- Один з найкращих інтелектуальних засобів завершення коду, заснований на різних факторах
- Інтеграція з Git
- Налагодження коду в редакторі
- Розширення для додавання додаткових функцій, таких як лінчування коду, теми та інші послуги.

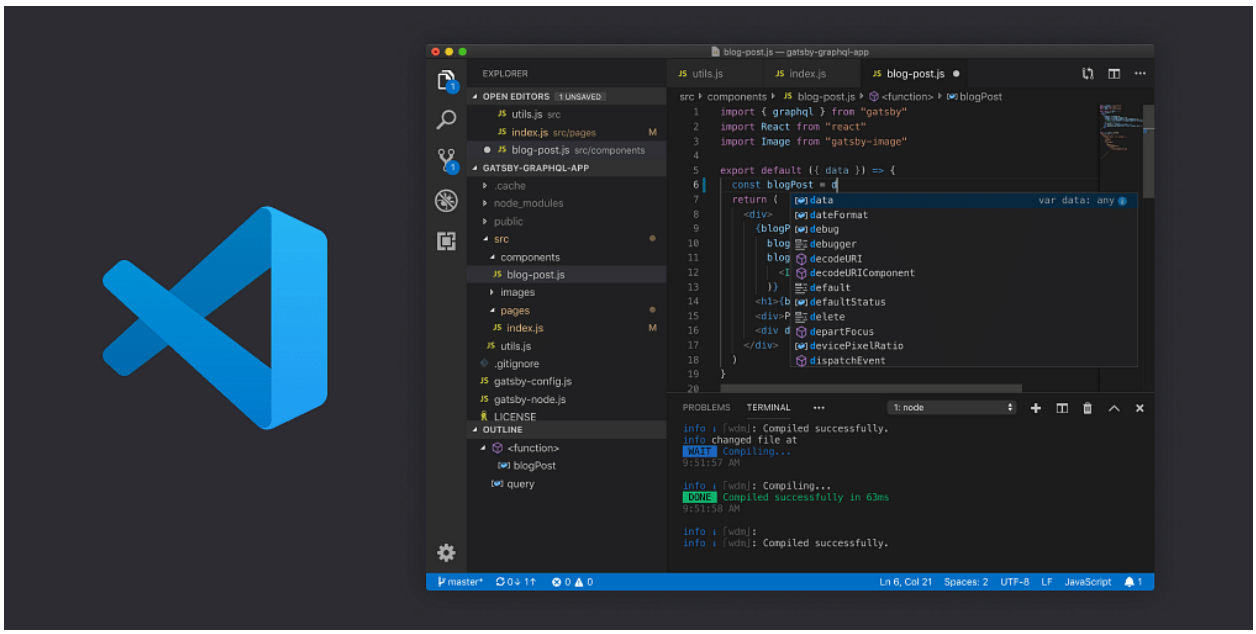

Рисунок 2.2 – Інтерфейс середовища розробки Visual Studio code

Sublime Text 3

Дуже популярний редактор коду. Він підтримує багато мов, включаючи Python.Він добре налаштовується, а також пропонує швидку швидкість розробки та надійність. Найбільш помітні функції Sublime Text 3 включають:

- Підсвічування синтаксису
- Користувацькі команди для роботи з IDE
- Ефективне управління каталогами проектів
- Підтримує додаткові пакети для веб- та наукової розробки на Python
- Це чудове середовище розробки Python для Windows

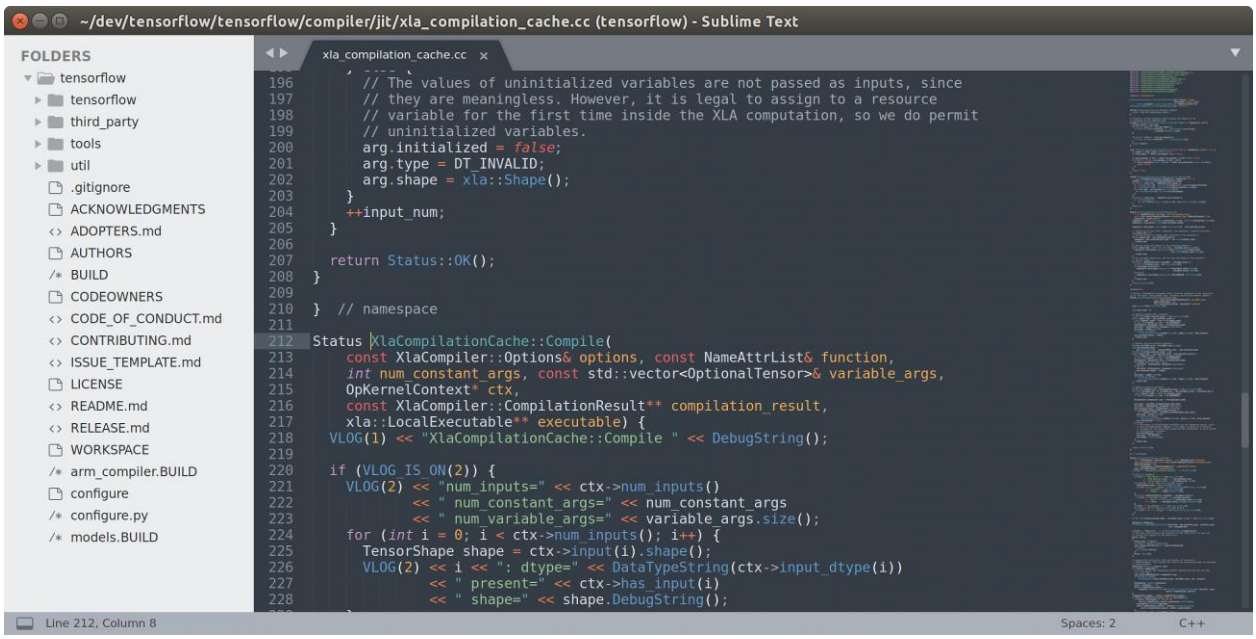

Рисунок 2.3 – Інтерфейс середовища розробки Sublime Text 3

Jupyter

Широко використовується в області обробки даних. Він простий у використанні, інтерактивний і дозволяє ділитися своїм кодом і візуалізувати його в режимі реального часу. Найбільш помітними особливостями Jupiter є:

- Підтримка робочих процесів чисельного та машинного навчання
- Комбінуйте код, текст і зображення, щоб поліпшити взаємодію з користувачем
- Взаємодія з науковими бібліотеками даних, такими як NumPy, Pandas та Matplotlib

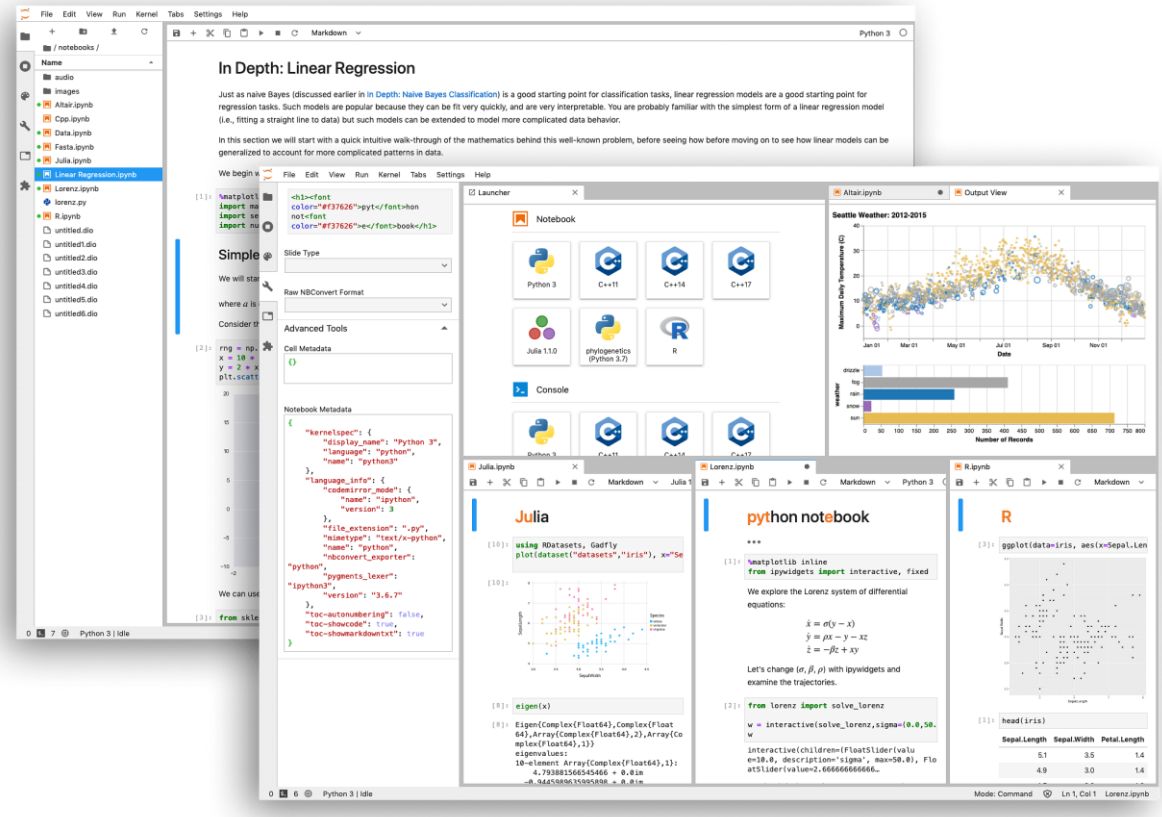

Рисунок 2.4 – Інтерфейс середовища розробки Jupyter

Після аналізу наведених середовищ розробки та редакторів коду було обрано IDE PyCharm. Оскільки він спеціалізований для написання коду на мові Python, має зручний інтерфейс для написання коду, роботи з git та перегляду баз даних. Також він не потребує додаткових налаштувань розробки проекту.

2.3 Вибір фреймворку для створення боту

Обравши мову програмування Python, мій наступний крок - вибір фреймворку для ефективної розробки чат-бота.

1) Aigram:

Це сучасний та повністю асинхронний фреймворк для API Telegram Bot. Пакет Aiogram Python завантажується 217 770 разів на тиждень. Таким чином, популярність aiogram дозволяє віднести його до числа впливових проектів.[15]

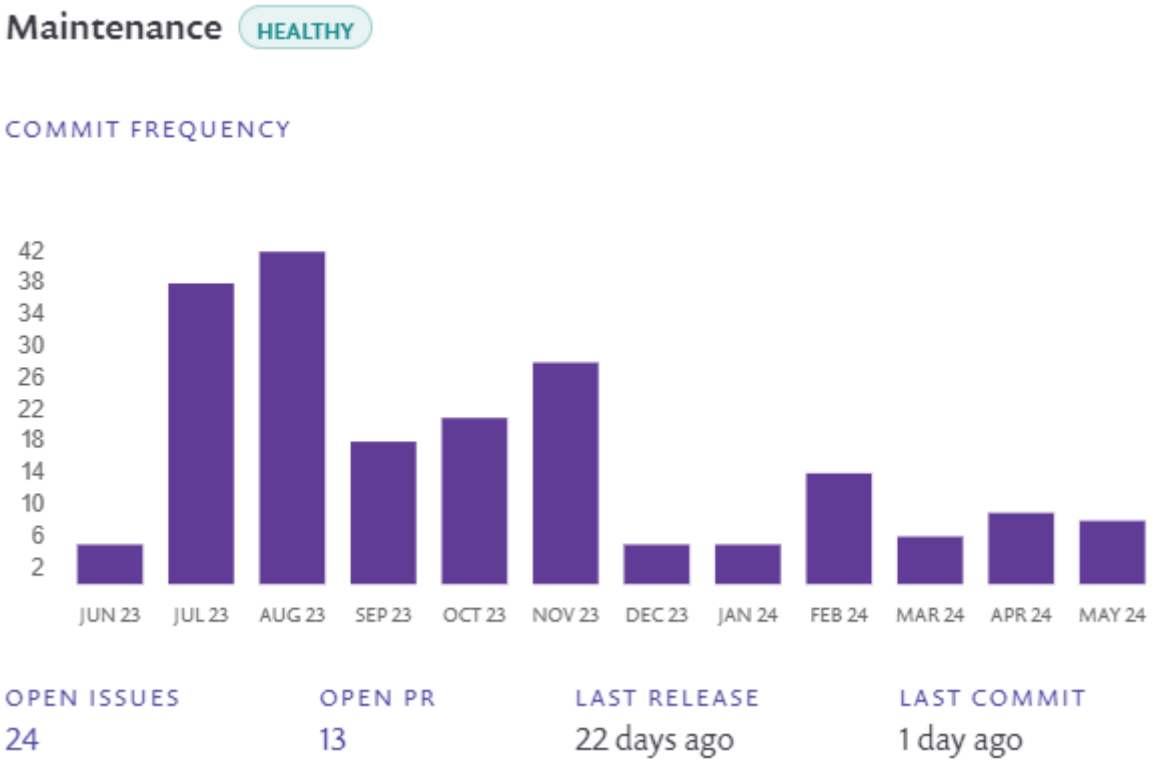

Рисунок 2.5 – Технічне обслуговування Aiogram

Плюси:

- Потужний і гнучкий фреймворк із безліччю можливостей для створення чат-ботів. Надає підтримку декількох платформ (Telegram, Discord, Slack).
- Забезпечує підтримку асинхронних запитів, що може істотно прискорити продуктивність бота.

Мінуси:

- Використання aiogram вимагає знання Python і асинхронного програмування.
- Розробка бота в Aiogram може зайняти більше часу, ніж в інших фреймворках.

2) Telebot:

Бот-фреймворк для Telegram bot API. Цей пакет забезпечує найкращий api для маршрутизації команд, вбудованих запитів, клавіатури та зворотних викликів. Всі методи API Telebot дуже легко запам'ятати і звикнути до них.

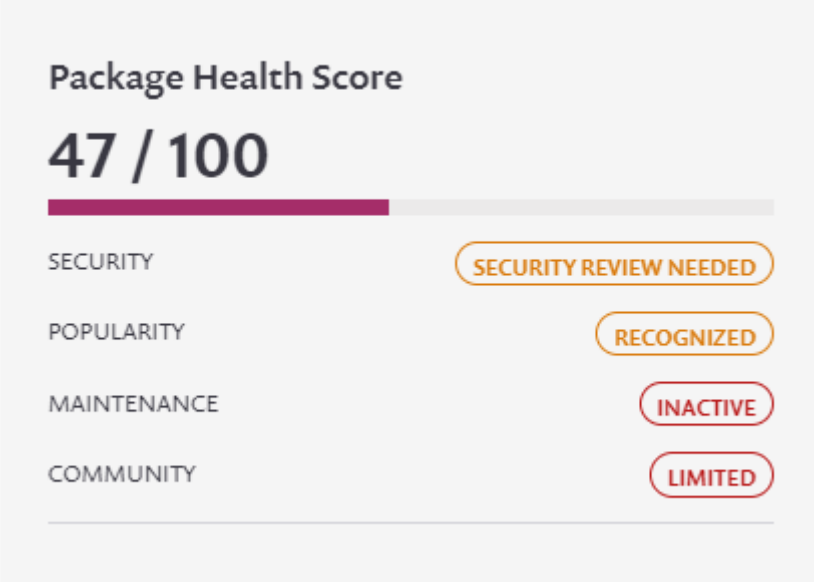

Рисунок 2.6 – Технічне обслуговування Telebot

Плюси:

- Telebot - це простий і зрозумілий фреймворк, який підходить для початківців; він підтримує API Telegram і дозволяє створювати різні типи ботів.
- Telebot предоставляет хорошую документацию.

Мінуси:

- Telebot не підтримує асинхронні запити.
- Деякі можливості Telegram API можуть бути недоступні.

3) Flask:

Мікрофреймворк з відкритим вихідним кодом, написаний на Python. Його називають мікро веб-фреймворком, оскільки за замовчуванням Flask надає лише основні функції веб-розробки, такі як обробка HTTP-запитів, веб-сервер та управління файлами cookie.[16]

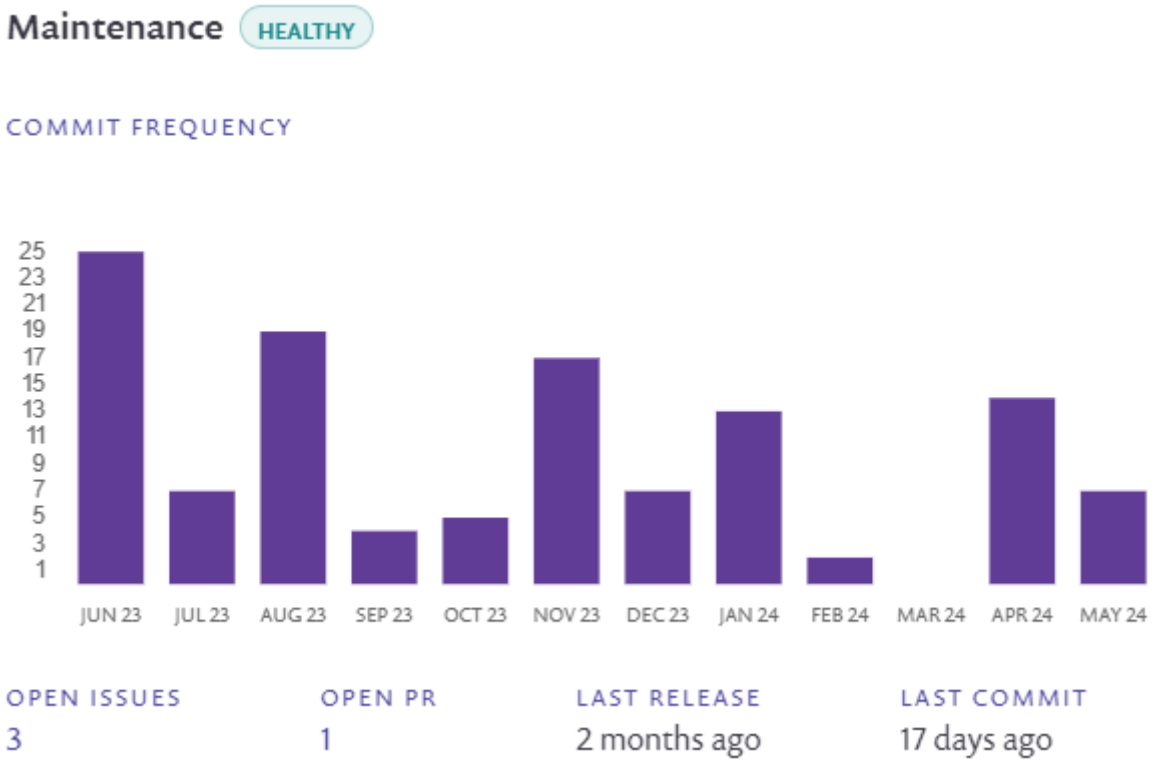

Рисунок 2.7 – Технічне обслуговування Flask

Плюси:

- Легкий фреймворк.
- Дозволяє створювати ботів з більш складной логікой.

Мінуси:

- Може бути складним для початківців.
- Може бути не ефективним для ботів з високим навантаженням.

Проаналізувавши все вище написане я зупинився на фреймворці Aiogram. Так як він асинхронний, це забезпечить одночасно обробляти велику кількість запитів, що значно підвищить ефективність роботи бота під час навантаження користувачами.

2.4 Вибір СУБД

Система управління базами даних (СУБД) - це тип інтерфейсу або програмного забезпечення для роботи з базами даних, що дає змогу додавати, видаляти та модифікувати дані в базі.

Для мого проекту проаналізуємо наступні СУБД:

1) OracleDB:

Переваги:

- Вона підтримується компанією Oracle і вирізняється високою надійністю. Розробники стверджують, що OracleDB рідко виходить з ладу і регулярно отримує оновлення.
- Вона вважається високомасштабованою базою даних, що ідеально підходить для роботи з великими масивами даних. Наразі Oracle переводить усі свої продукти та послуги в хмару, що забезпечує більшу гнучкість.
- Вона надійна, скрупульозно дотримується сучасних стандартів безпеки (включаючи відповідність стандарту PCI) і пропонує хороше шифрування конфіденційних даних.
- Вона дуже ефективно керує пам'яттю і з легкістю виконує складні операції. Крім того, вона ефективно керує та організовує різні сторонні інструменти.
- Вона перевершує інші рішення за швидкістю доступу до даних по мережі.

Недоліки:

- Найпопулярніша СУБД, OracleDB, також є однією з найдорожчих. Ліцензії на процесор для стандартної версії коштують до 17 500 доларів США за одиницю.
- Oracle має складну документацію та бракує хороших посібників. Підтримка клієнтів є корисною, але деякі розробники скаржаться на довгий час відповіді.

- Ці фактори роблять OracleDB типом бази даних, який найкраще підходить для зберігання великих обсягів даних. Малому та середньому бізнесу варто шукати більш економічно ефективні альтернативи.[17]

2) MySQL:

Переваги:

- MySQL складається з надійного рівня захисту даних, який захищає конфіденційні дані від зловмисників, а паролі MySQL зашифровані, що робить їх більш безпечними.
- MySQL можна безкоштовно завантажити та використовувати з офіційного сайту MySQL.
- Сумісна з більшістю операційних систем, включаючи Windows, Linux, NetWare, Novell, Solaris та інші варіації UNIX.
- Клієнт і сервер можуть працювати на одному або різних комп'ютерах через Інтернет або локальну мережу.
- Має унікальну архітектуру зберігання, яка є швидшою, дешевшою та надійнішою.
- Підвищує продуктивність розробників завдяки використанню представлень, тригерів і збережених процедур.
- Є простим і зручним у використанні, вимагаючи лише базових знань MySQL та кількох простих SQL-запитів для створення та роботи з MySQL.
- Використовує архітектуру клієнт-сервер. Будь-яка кількість клієнтських або прикладних програм може взаємодіяти з сервером бази даних (MySQL), запитуючи дані та зберігаючи зміни.
- Масштабується і здатна обробляти понад 50 мільйонів рядків. Цього достатньо для обробки практично будь-якої кількості даних. Хоча за замовчуванням обмеження на розмір файлу становить 4 ГБ, його можна збільшити до 8 ТБ.
- Дозволяє відкочувати транзакції.

- Він дуже гнучкий, оскільки підтримує багато вбудованих додатків.

Недоліки:

- Не дуже ефективна при роботі з дуже великими базами даних.
- Не має таких же чудових інструментів для розробки та налагодження, як платні бази даних.
- Версії MySQL нижче 5.0 не підтримують COMMIT, збережені процедури та ROLE.
- Не ефективно обробляє транзакції, що робить їх схильними до пошкодження даних.
- Не підтримує обмеження валідації SQL.[18]

3) PostgreSQL:

Преваги:

- Він надає програмні інтерфейси для найпопулярніших мов, таких як Java, Python, C# та C/C++.
- Він підтримує користувацькі типи даних.
- Що стосується цілісності даних, він підтримує механізми явного блокування разом із рекомендованим блокуванням.
- Для підвищення продуктивності він підтримує розширену індексацію, таку як Nist, SP-Gist, KNN-Gist, GIN, GIN, Індекс покриття та фільтр Bloom
- Реалізовано мультиспортивний контроль паралелізму (MVCC).
- З точки зору безпеки, він підтримує безпеку на рівні стовпців і рядків.
- Багатофакторна аутентифікація з використанням сертифікатів і додаткових методів.
- Підтримка міжнародних наборів символів, таких як ICU comparison.
- Він має сортування без урахування регістру і наголосів, що корисно в разі інтернаціоналізації.

Недоліки:

- Postgres не відноситься до однієї організації. Тому, незважаючи на достатню функціональність і порівнянність з іншими системами СУБД, виникли проблеми з розповсюдженням.
- PostgreSQL більше орієнтована на сумісність, тому зміни для підвищення швидкості є більш трудомісткими, ніж в MySQL.
- Важче здійснення реплікацій.
- По продуктивності вона повільніша за MySQL.[19]

4) MongoDb:

Переваги:

- MongoDB, база даних без схеми, вимагає міграції схеми.
- Запити до документів використовуються тому, що це документ-орієнтована мова, що важливо для забезпечення динамічних запитів.
- Порівняно з іншими реляційними базами даних, оптимізувати продуктивність дуже просто.
- Внутрішня пам'ять, що використовується для зберігання даних, гарантує швидкий доступ до них.
- Завдяки шардингу MongoDB масштабується горизонтально (розподіляючи дані між різними серверами). Дані розбиваються на шарди і розподіляються рівномірно. Це полегшує читачеві пошук даних.
- Легше підтримувати, ніж звичайні бази даних.
- Реплікація даних і розподілені робочі навантаження використовуються для підвищення доступності даних і забезпечення високої продуктивності.[20]

Недоліки:

- Вона не підтримує об'єднання так само, як реляційні бази даних. Однак можна вручну додати функцію об'єднання на

сторінку за допомогою відповідного коду. Однак це може сповільнити виконання і вплинути на продуктивність.

- MongoDB зберігає ім'я ключа для кожної пари значень. Крім того, обмеження об'єднання призводить до надмірності даних. Це призводить до того, що пам'ять використовується більше, ніж потрібно.
- Максимальний розмір документа - 16 МБ.
- Документи не можуть бути вкладені більш ніж на 100 рівнів.

Після аналізу відомих СУБД мій вибір зупинився на PostgreSQL через його популярність та хорошу взаємодію з мовою програмування Python.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Створення бот токену

Перед тим як почати розробку продукту потрібно створити самого бота та отримати бот токен для підключення його до проекту.

Знайти телеграм-бота під назвою "@botfarther", він допоможе створити та керувати вашим ботом.

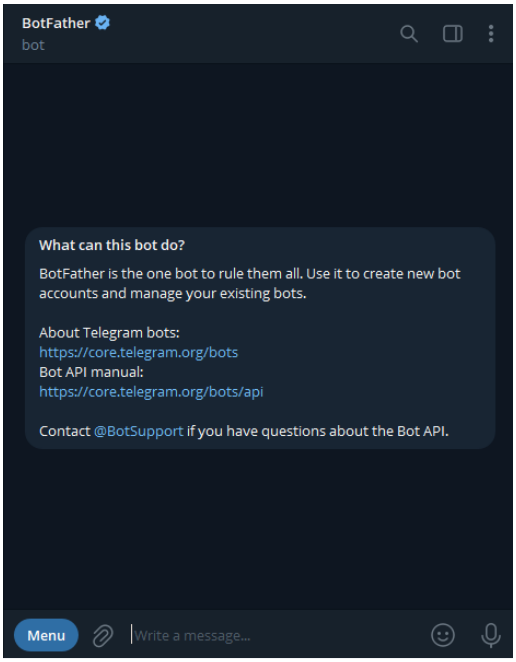

Рисунок 0.1 – Діалогове вікно BotFather

Запустить бота натиснувши на кнопку розпочати.

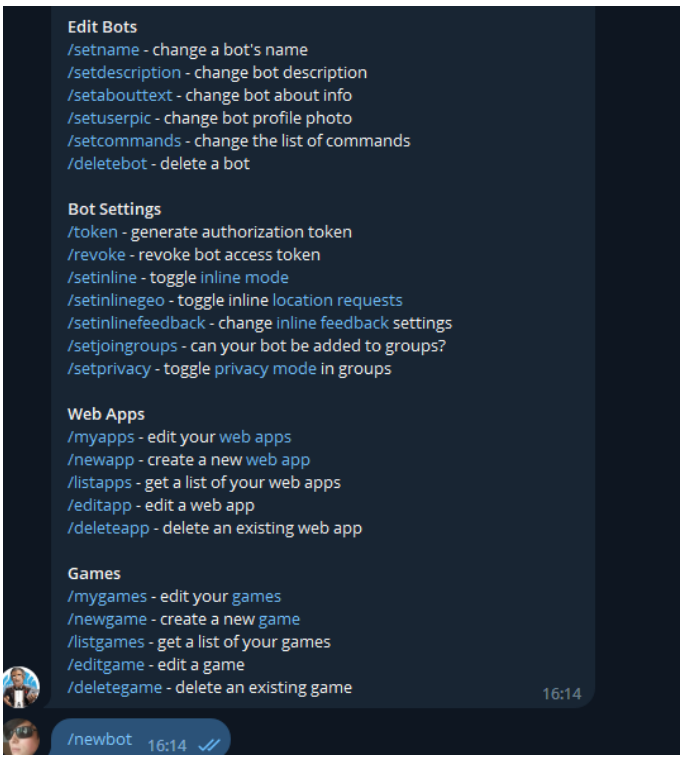

Рисунок 0.2 – Список команд для редагування ботів

Після вибору команди newbot потрібно придумати ім’я боту та username яке повинно закінчуватися на bot та починатися з “@”.

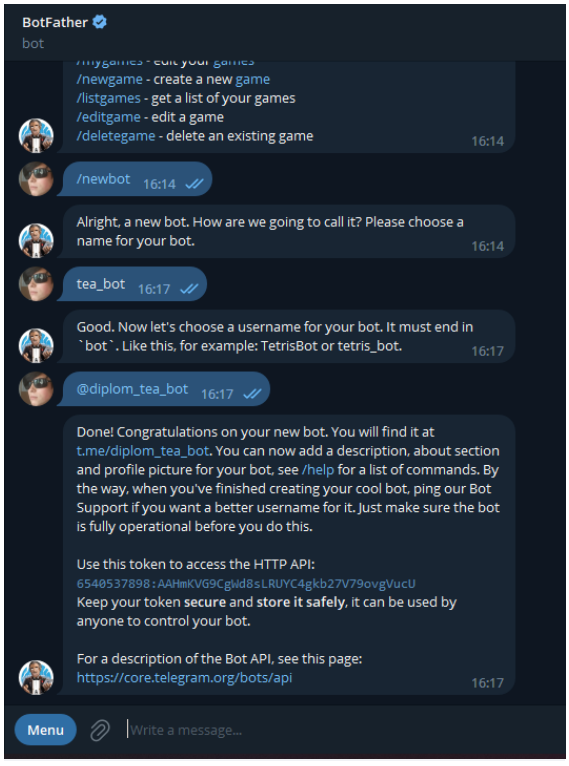

Рисунок 0.3 – Заповнення інформації та отримання токену

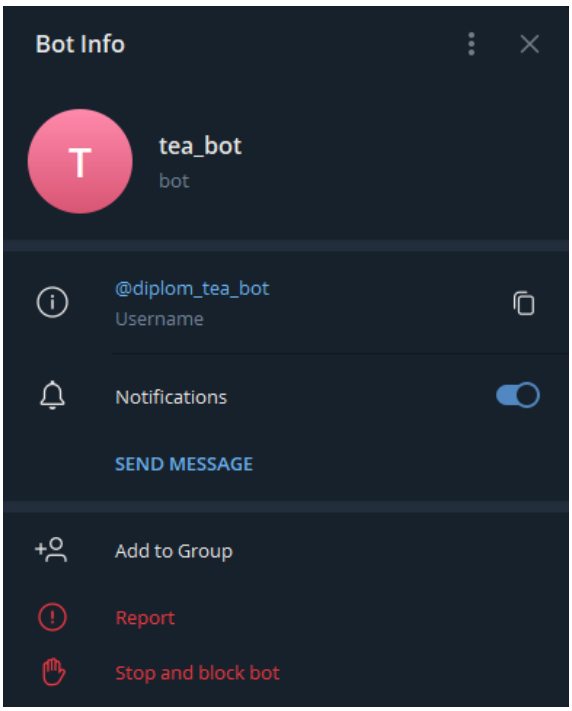

Рисунок 0.4 – Сторінка готового бота

Таким чином було створено нового бота та отримано його токен. Тепер коли у нас є токен його можна інтегрувати з різними платформами використовуючи API. З ним можна надсилати запити на сервера, отримувати відповіді та виконувати завдання задані функціями.

3.1 Структура проекту

Структура проекту телеграм-бота для продажу товарів організована таким чином, щоб максимально оптимізувати розробку та зробити код легким для розуміння та обслуговування.

- **db**: Директорія в якій виконується підключення до бази даних, написані методи для отримання з неї даних та моделі самих таблиць.[21]
- **filters**: Директорія для створення фільтрів користувачів.
- **handlers**: Директорія в якій написані хендлери для обробки запитів користувача.
- **kb**: Директорія з клавіатурами для адмін та клієнтської панелі.
- **core.py**: Файл ініціалізації бота.[22]

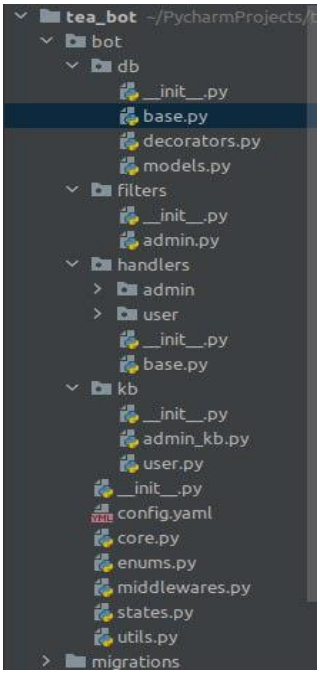

Рисунок 0.5 – Структура проект

Проектування структури бази даних. На цьому етапі визначаються зв'язки між таблицями, встановлюються обмеження цілісності даних та оптимізуються структури для підвищення продуктивності. Особлива увага приділяється нормалізації даних, щоб уникнути їх дублювання. Проектування структури бази даних також включає створення індексів для прискорення запитів і визначення зовнішніх ключів для підтримки цілісності. Все це забезпечує ефективне зберігання та обробку даних відповідно до потреб користувачів та бізнес-процесів.

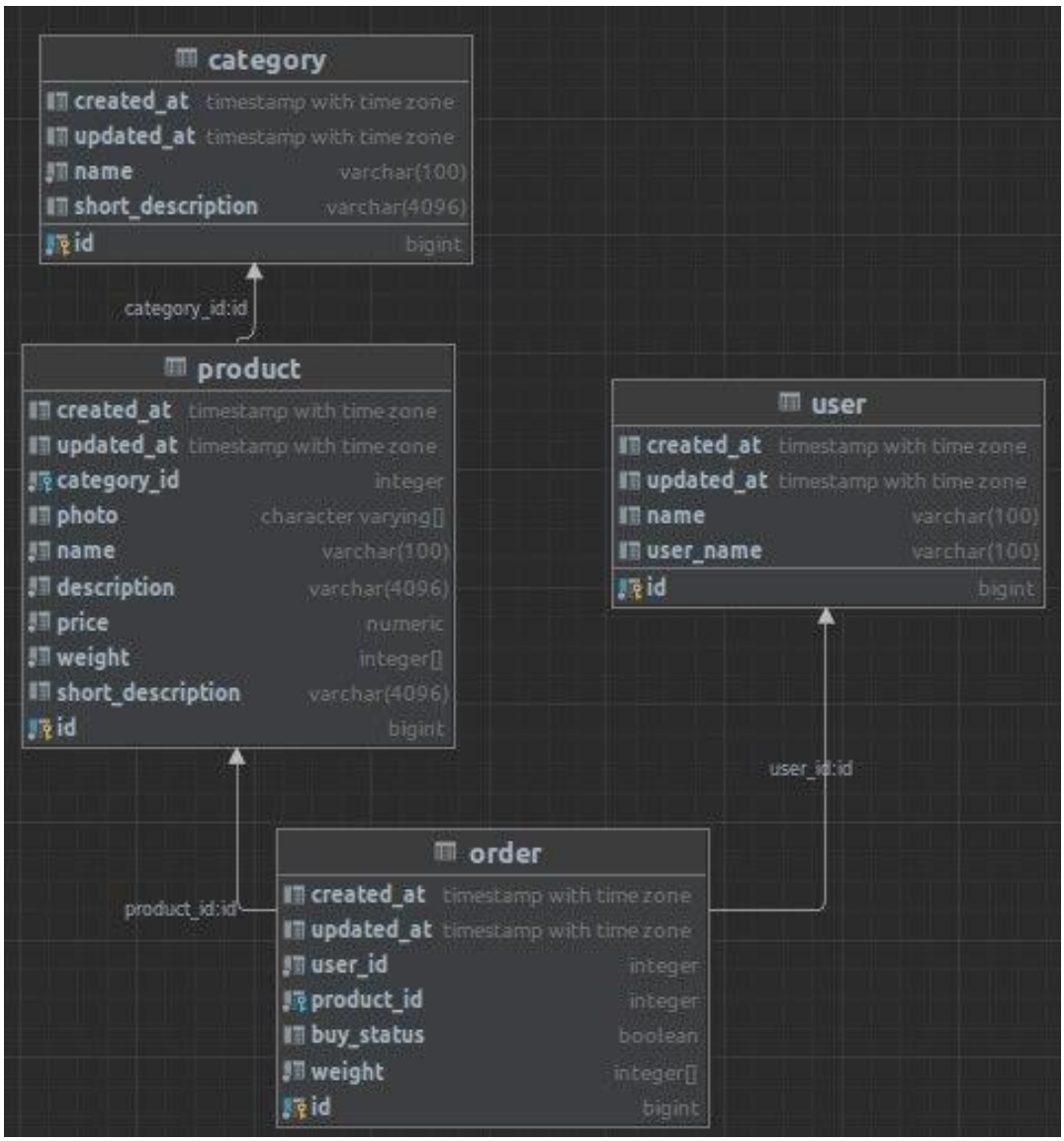

Рисунок 0.6 – Схема бази даних

Дана база даних використовується для керування категоріями, товарами клієнтами та їх замовленнями.

Таблиця 3.1 – Опис таблиці «User»

№	Назва поля	Тип даних	Опис	Ключ	Обмеження
1	name	varchar	Поле збереження імені користувача		< 100 символів

Продовження таблиці 3.1 – Опис таблиці «User»

2	user_name	varchar	Поле збереження юзернейму користувача		< 100 символів
3	id	integer	Ідентифікатор користувача в базі даних	Primary_key	
4	created_at	datatime	Дата створення		
5	updated_at	datatime	Дата останнього оновлення		

Таблиця 3.2 – Опис таблиці «Category»

№	Назва поля	Тип даних	Опис	Ключ	Обмеження
1	name	varchar	Поле збереження назви категорії		< 100 символів
2	short_description	varchar	Поле збереження короткого опису категорії		< 4096 символів
3	id	integer	Ідентифікатор категорії в базі даних	Primary_key	
4	created_at	datatime	Дата створення		
5	updated_at	datatime	Дата останнього оновлення		

Таблиця 3.3 – Опис таблиці «Product»

№	Назва поля	Тип даних	Опис	Ключ	Обмеження
1	photo	array	Массив збереження фото		
2	name	varchar	Назва продукту		< 100 символів
3	description	varchar	Опис продукту		< 4096 символів
4	price	decimal	Ціна		
5	weight	array(inte-ger)	Варіанти ваги		
6	short_description	varchar	Поле збереження короткого опису продукту		< 4096 символів
7	id	integer	Ідентифікатор продукту	Pri-mary_key	
8	category_id	integer	Ідентифікатор категорії в якій знаходиться продукт	Foreign_key	
9	created_at	datatime	Дата створення		
10	updated_at	datatime	Дата останнього оновлення		

Таблиця 3.4 – Опис таблиці «Order»

№	Назва поля	Тип даних	Опис	Ключ	Обмеження
1	user_id	integer	Id користувача в телеграмі		
2	weight	array(integer)	Варіанти ваги в замовленні		
3	buy_status	boolean	Статус замовлення		
4	id	integer	Ідентифікатор замовлення	Primary_key	
5	product_id	integer	Ідентифікатор продукту для замовлення	Foreign_key	
6	created_at	datatime	Дата створення		
7	updated_at	datatime	Дата останнього оновлення		

3.2 Структурна схема Адміністративного меню

Відповідно до технічного завдання, потрібно створити меню, яке допоможе адміністраторам керувати роботою телеграм бота. Вимогою для нього є комфортність використання та зрозумілий інтерфейс.

Так як меню повинне працювати лише в адміністратора, воно повинно включати наступний функціонал:

- Перегляд категорій;

- Додавання та видалення категорії;
- Редагування категорії;
- Перегляд списку продуктів категорії;
- Можливість додавання, видалення та редагування продукту;

Відповідно до функціоналу меню було складено структурну схему «Робота меню адміністратора» рис. 3.7.

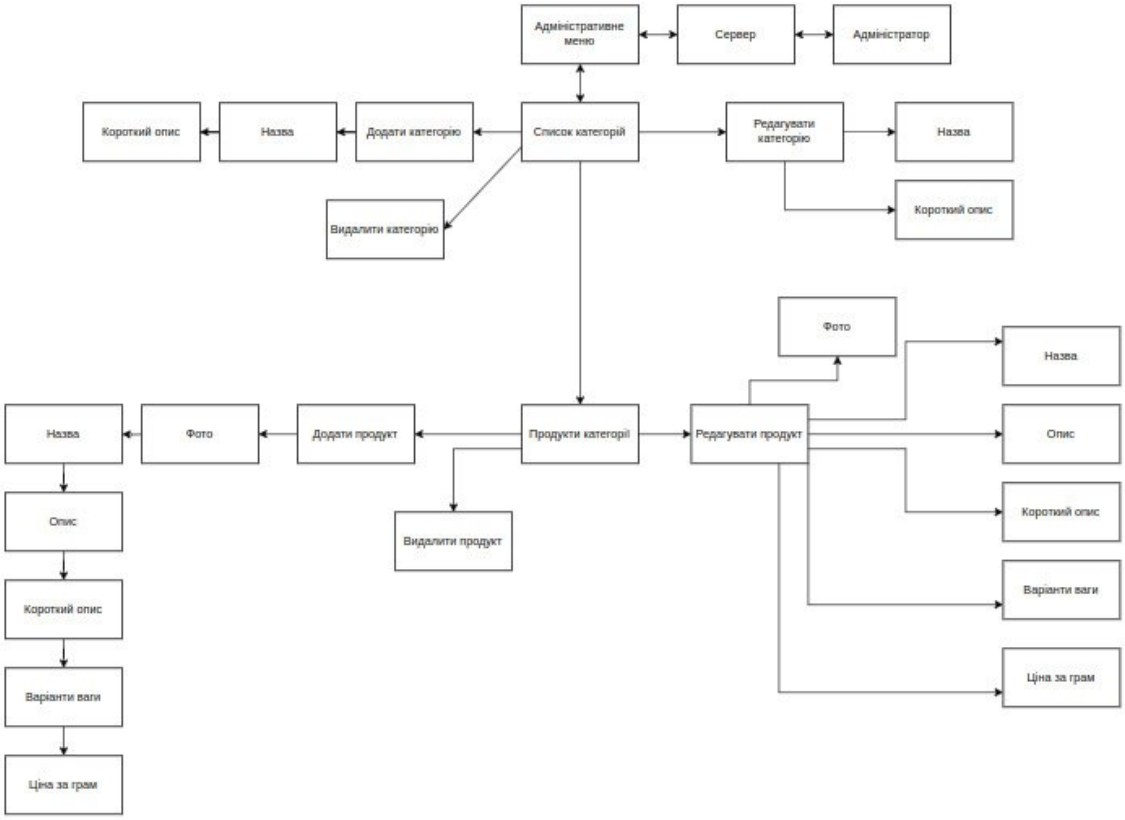

Рисунок 0.7 – Структура адміністративного меню

Для отримання доступу до адмінської панелі потрібно використовувати команду /admin. Ця команда буде активована тільки в тому випадку, якщо ваш ID користувача відповідає заздалегідь підготовленому списку адміністраторів.

Якщо категорії ще немає, адміністратор можете створити її, натиснувши на кнопку "Додати категорію" і ввівши відповідні дані. Після цього категорія з'явиться у списку, а поруч з нею будуть дві кнопки: "Видалити" та "Редагувати". рис. 3.8.

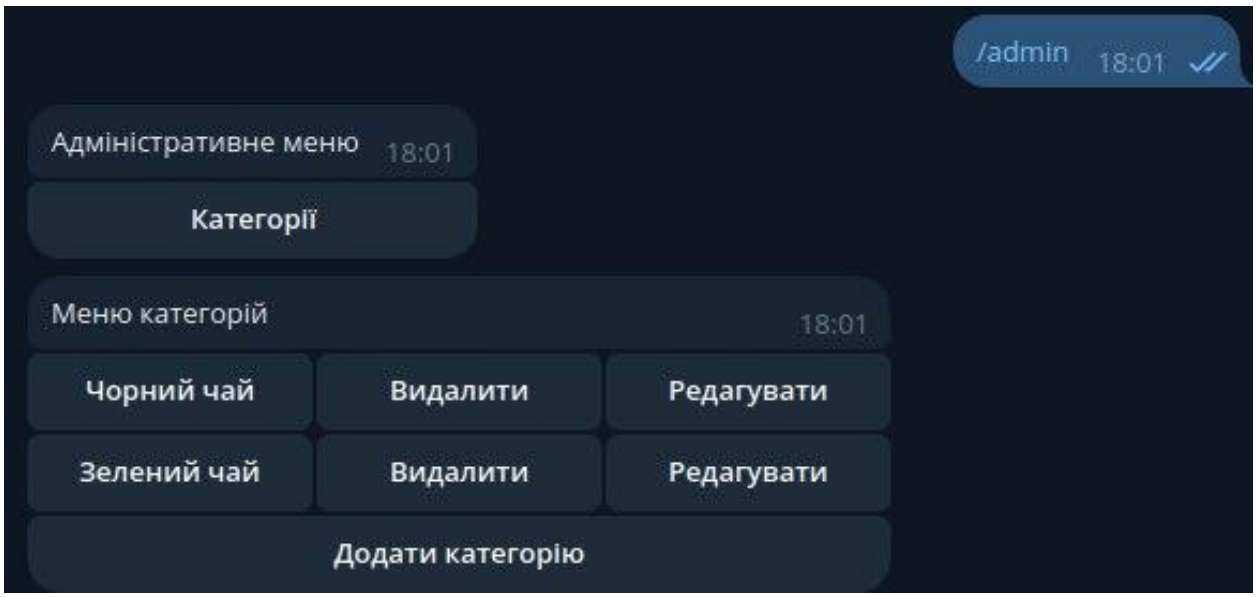

Рисунок 0.8 – Категорії адміністративного меню

Натиснувши на назву категорії, адміністратор потрапляє в меню продуктів цієї категорії, де відображається назва продукту та його опис. Як і у випадку з категоріями, можна додавати нові продукти, а також видаляти або редагувати відповідні елементи продукту. рис. 3.9-3.10.

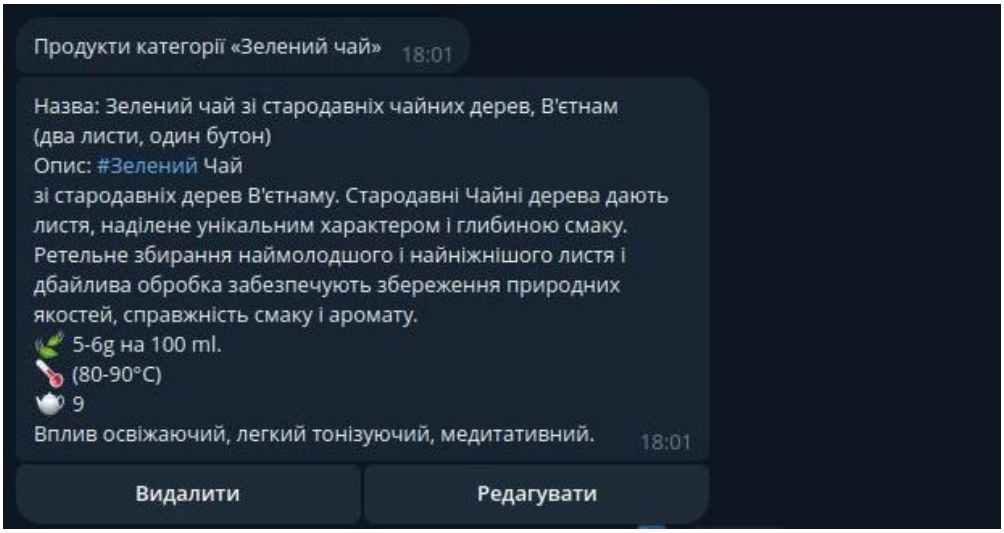

Рисунок 0.9 – Продукти адміністративного меню

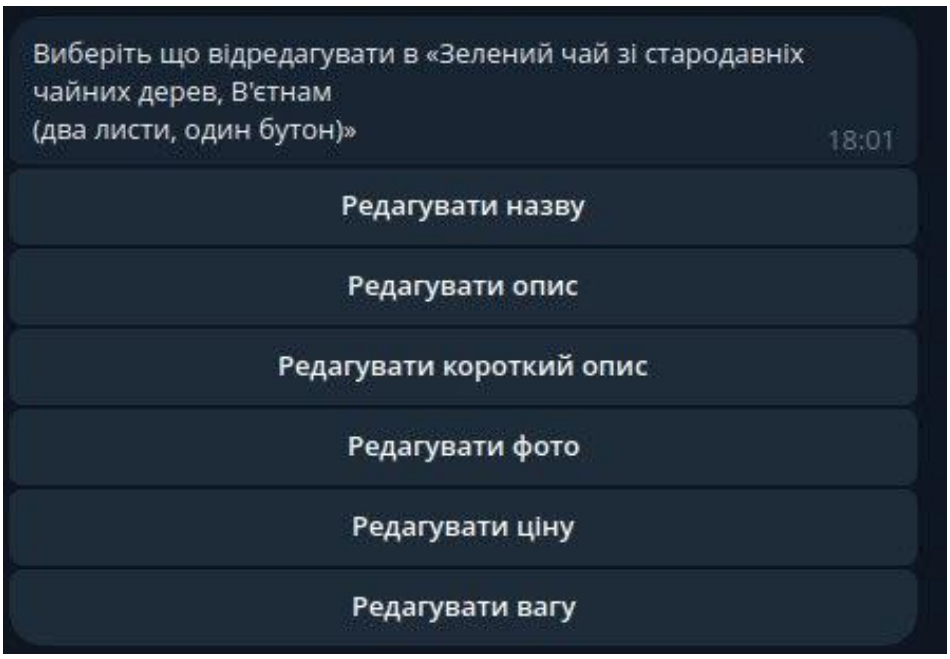

Рисунок 0.10 – Редагування продукту

3.3 Структурна схема Клієнтського меню

Для створення клієнтського меню, необхідно розглянути функції, доступні звичайному користувачеві. Основне завдання клієнтського меню - надати зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з ботом. Нижче наведено список функцій клієнтського меню:

- Перегляд списку категорій;
- Перехід до категорії;
- Перегляд списку продуктів;
- Перегляд детальної інформації про продукт;
- Додавання продукту до кошика;
- Перегляд кошика;
- Видалення продукту з кошика;
- Очищення кошика;
- Створення замовлення;

Відповідно до функціоналу меню було складено структурну схему «Робота меню клієнта» рис. 3.11.

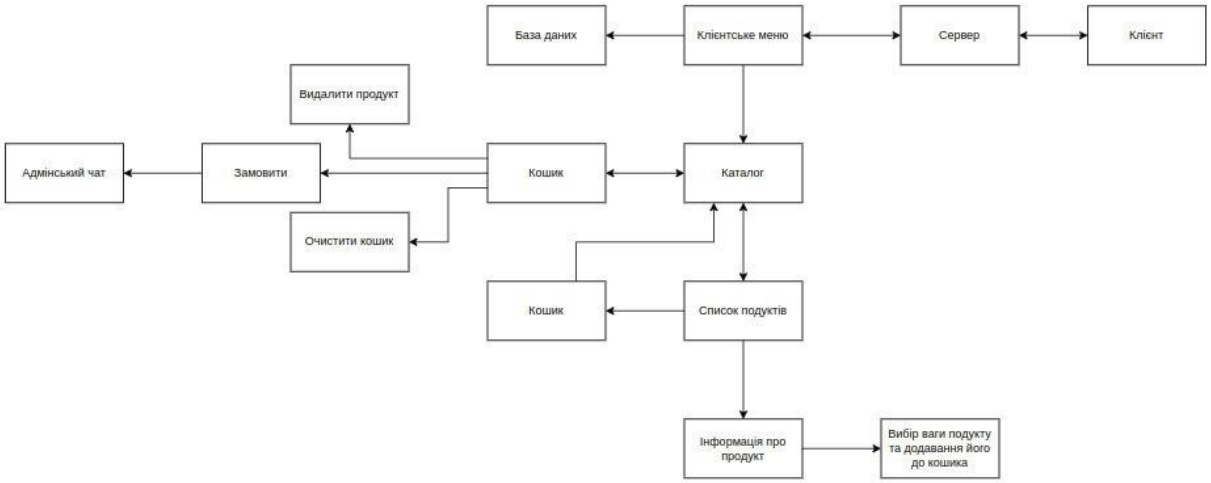

Рисунок 0.11 – Структура клієнтського меню

Для отримання доступу до клієнтської панелі потрібно використати команду /start. Після чого з’явиться клавіатура з можливістю вибрати категорію та перегляду кошика. рис. 3.12.

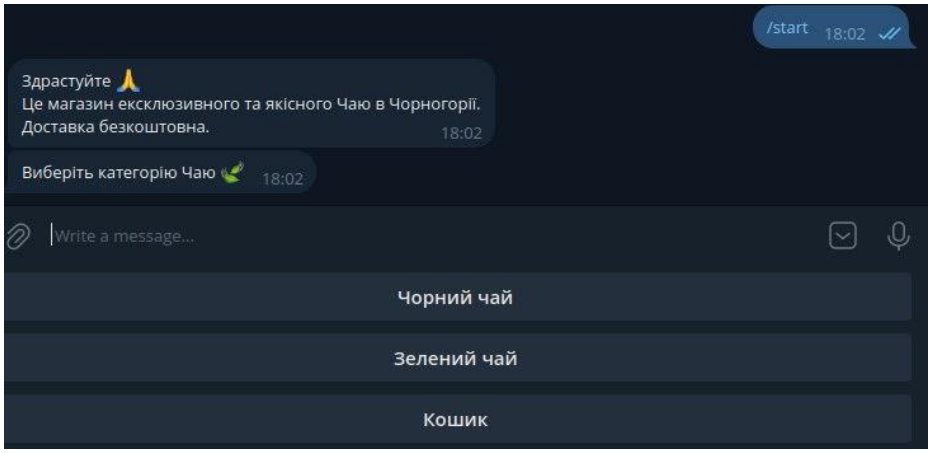

Рисунок 0.12 – Меню вибору категорії

При натисканні на категорію меню змінюється на список продуктів даної категорії. Також залишається кнопка перегляду кошика та з’вляється можливість повернутися до попереднього меню. рис. 3.13.

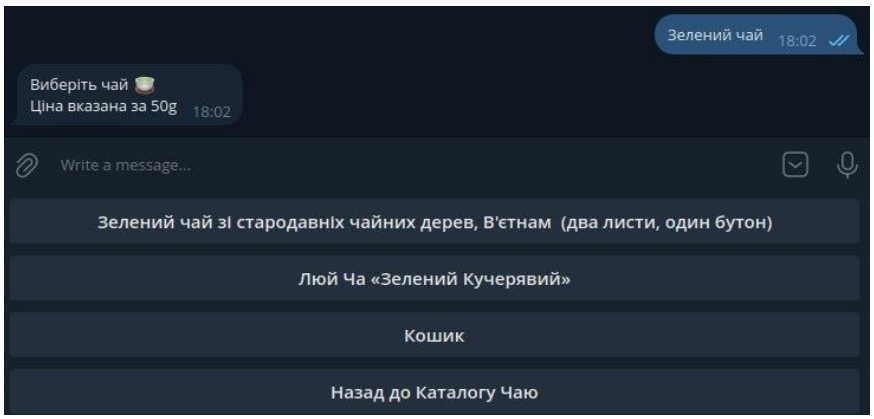

Рисунок 0.13 – Меню вибору продукту

Після вибору товару відображається детальна інформація про нього, включно з фотографією, описом і ціною за 50 грам. Разом з інформацією користувач може додати товар до кошика. рис. 3.14.

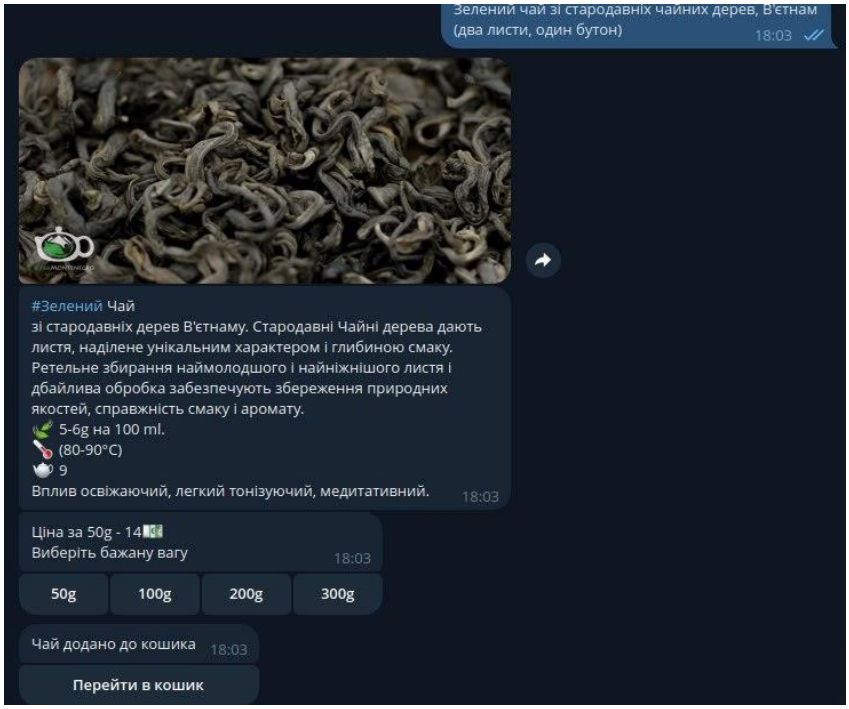

Рисунок 0.14 – Детальна інформація про продукт

В кошику відображається весь товар який хоче замовити клієнт. Він може видаляти зайвий товар, повністю очищати кошик та створити замовлення. рис. 3.15.

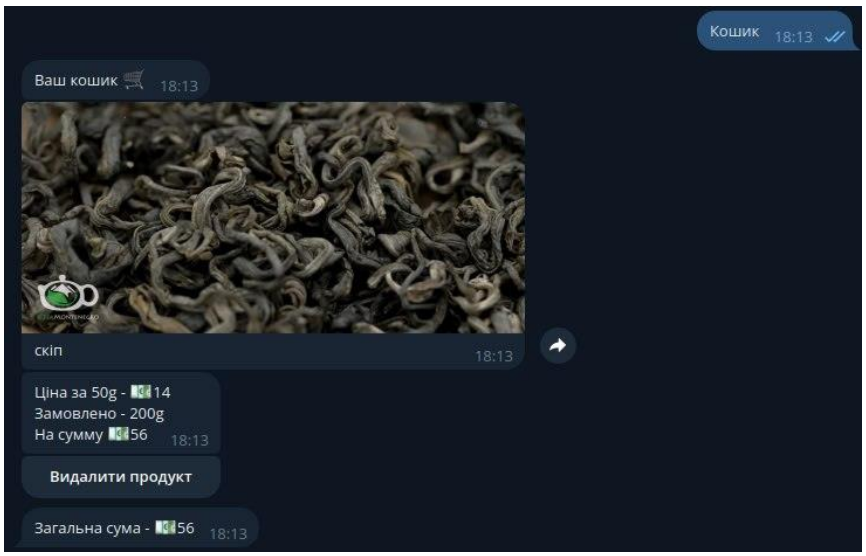

Рисунок 0.15 – Кошик

При підтвердженні замовлення дані про клієнта та продукти передаються в окремий й чат. Де адміністратори зможуть перевірити та підтвердити його у самого замовника. рис. 3.16.

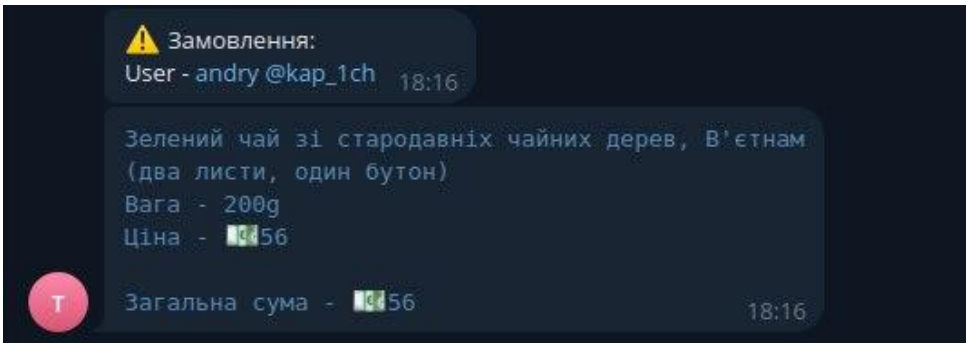

Рисунок 0.16 – Повідомлення в адмінський чат

3.5 Реалізація адміністративної та клієнтської панелі

Після визначення задач, які буде виконувати бот для продажів, необхідно провести його ініціалізацію та заповнення функціоналом. Ініціалізація включає в себе налаштування основних параметрів для бота, таких як інтеграція з телеграм ботом, підключення до бази даних. Впровадження цього функціоналу забезпечує розробку скриптів для обробки запитів клієнтів та адміністраторів, автоматизацію процесів продажів.

Першим що потрібно зробити - це створити load.yaml файл. Він використовується в проектах для безпечного зберігання параметрів конфігурації, змінних середовища та конфіденційної інформації. Це надає зручний спосіб керувати різними параметрами, необхідні проекту, і впорядковує їх, не вводячи їх безпосередньо у код.[23]

Поля .yaml файлу:

- bot_token: поле в якому зберігається токен бота
- admins: список адміністраторів бота
- database_uri: посилання на базу даних

```
bot_token: ""
admins: [ ]
chat:
database_uri: postgresql+asyncpg://postgres:post-
gres@localhost:5432/tea_bot
```

Наступним етапом є сама ініціалізація бота. Для цього потрібно створити об’єкт бота який приймає його токен взятий із .yaml файлу та об’єкт Dispatcher який потрібен для обробки даних, опрацьовування подій перед тим як вони потраплять до обробника.

```
from aiogram.types import ParseMode

from bot.middlewares import AlbumMiddleware

dp: Dispatcher
bot: Bot


def create_dp(config: dict):
    global dp, bot
    bot = Bot(config["bot_token"], parse_mode=Parse-
Mode.HTML, validate_token=True)
```

```
storage = RedisStorage2()
dp = Dispatcher(bot, storage=storage)

dp.setup_middleware(AlbumMiddleware())

return dp
```

Далі потрібно налаштувати підключення до бази даних, створення таблиць в ній та написання CRUD. Підключення відбувається за допомогою ORM SQLAlchemy який спрощує роботу з базою даних та Python автоматично викликаючи класи в Python перетворюючи їх в SQL запити.[24] В ньому використовуємо функцію create_async_engine() в яку першим аргументом передається URL-адресу зазвичай це рядок, який вказує діалект бази даних і аргументи підключення.[25]

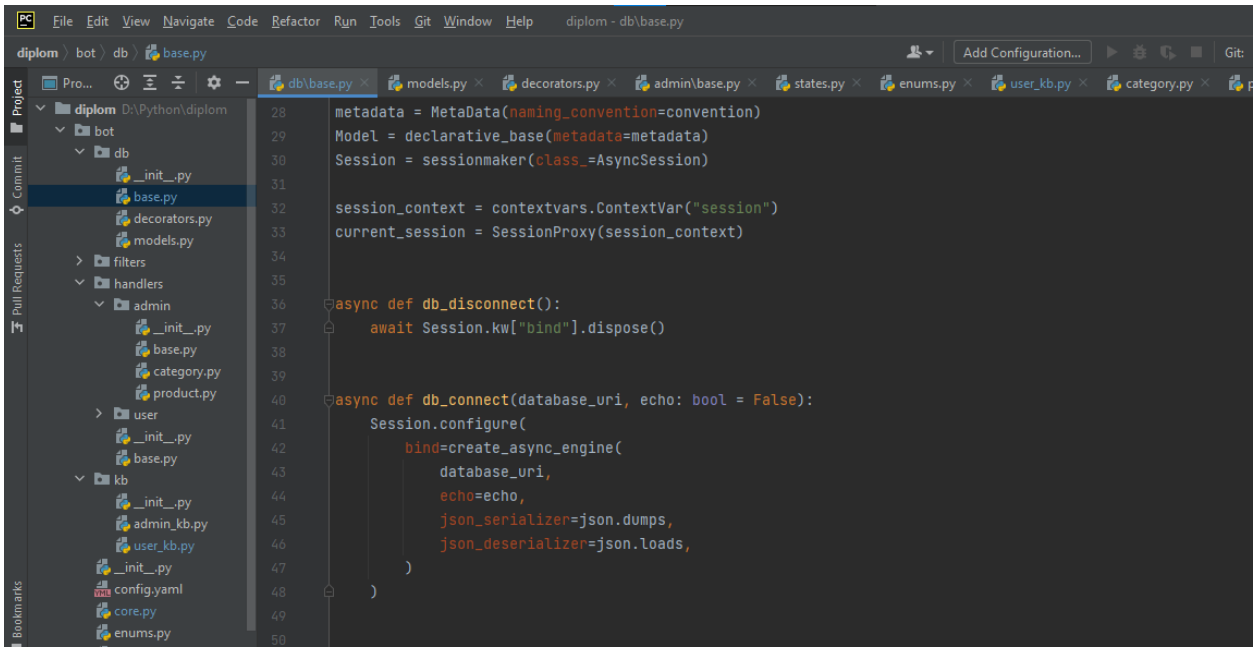

Рисунок 0.17 – Підключення до бази даних

Після підключення створено базовий клас моделей від якого будуть наслідуватися всі класи моделей таблиць де визначено методи CRUD такі як:

- get_or_create(): отримання об’єкту або створення якщо його не існує

- get(): отримання об’єкту
- get_list(): отримання списку об’єктів
- create(): створення
- update(): оновлення
- delete(): видалення

```
from sqlalchemy import Column, ForeignKey, Integer,
String, Boolean, DECIMAL
from sqlalchemy.dialects.postgresql import ARRAY
from sqlalchemy.orm import relationship

from .base import Base, CreatedMixin, UpdatedMixin


class User(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "user"

    name = Column(String(100), nullable=True)
    user_name = Column(String(100), nullable=True)


class Category(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "category"

    name = Column(String(100), nullable=False)
    short_description = Column(String(4096))
    products = relationship(
        "Product",
        back_populates="category",
        cascade="all, delete",
        passive_deletes=True,
    )


class Product(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "product"

    category_id = Column(Integer, ForeignKey("cate-
gory.id", ondelete="CASCADE"), nullable=False)
    photo = Column(ARRAY(String), nullable=True)
```

```
    name = Column(String(100), nullable=False)
    description = Column(String(4096), nullable=False)
    short_description = Column(String(4096))
    price = Column(DECIMAL, nullable=False)
    weight = Column(ARRAY(Integer), nullable=False)

    category = relationship("Category", back_popu-
lates="products")
    orders = relationship(
        "Order",
        back_populates="product",
        cascade="all, delete",
        passive_deletes=True,
    )
```

Надалі відбувається сама реалізація функціоналу за допомогою хендлерів – це асинхронний виклик, який приймає подію з контекстними даними і повертає відповідь.[26]

```
@dp.message_handler(IsSuperAdmin(), commands="admin",
state="*")
async def admin_menu(msg: types.Message, state: FSMCon-
text) -> None:
    await state.finish()
    await msg.answer("Адміністративне меню", re-
ply_markup=admin_kb)
```

Для створення нового продукту та категорії використовується FSM(Кінцевий автомат). Він може перебувати в будь-якому заданому із заданому списку станів.[27] В даному випадку він потрібен для того щоб можна було поетапно заповнювати інформацію продукту, категорії та можливість подорожування клієнта по меню.

```
class ProductState(StatesGroup):
    photo = State()
    name = State()
    description = State()
    short_description = State()
```

```
    weight = State()
    price = State()
    edit_name = State()
    edit_price = State()
    edit_photo = State()
    edit_weight = State()
    edit_description = State()
    edit_short_description = State()


@dp.callback_query_handler(
    IsSuperAdmin(), product_callback.filter(ac-
tion=ProductActionEnum.create.value)
)
@session_decorator()
async def create_product(cq: types.CallbackQuery,
state: FSMContext, callback_data: dict) -> None:
    async with state.proxy() as data:
        data["category_id"] = int(callback_data["cate-
gory_id"])
    await ProductState.photo.set()
    await cq.message.answer("Відправте фото", re-
ply_markup=skip_photo_kb())


@dp.message_handler(state=ProductState.photo, re-
gexp="Пропустити фото")
async def skip_photo(msg: types.Message):
    await ProductState.next()

    await msg.answer("Додавання фото пропущено", re-
ply_markup=ReplyKeyboardRemove())
    await msg.reply("Введіть назву:")
```

Одним з найпоширеніших способів взаємодії з ботами Telegram є використання кнопок на клавіатурі. Ці кнопки забезпечують зручний і інтуїтивно зрозумілий спосіб введення даних і виконання певних дій з ботом. Без кнопок на клавіатурі вам доведеться вручну вводити команди та відповіді, що може зайняти багато часу та призвести до помилок.

Кнопки на клавіатурі також дозволяють нам зробити роботу користувачів більш зручною і цікавою. Надаючи користувачам чіткі та впорядковані можливості, боти можуть направляти користувачів у складні робочі процеси або надавати їм швидкий доступ до необхідної інформації та функцій.[28] Телеграм боти мають два види клавіатур:

ReplyKeyboardButton: Кнопка цього типу на клавіатурі використовується для надання Користувачеві заздалегідь визначеного набору параметрів на клавіатурі відповіді. Коли користувач натискає на кнопку, виділений текст відправляється у вигляді повідомлення боту telegram. Це забезпечує простий і зручний спосіб взаємодії між користувачами та ботами telegram. В даній роботі віна потрібна для того щоб клієнти могли переглядати меню категорій, продуктів та працювати з корзиною.

InlinekeyboardButton: Це кнопка на вбудованій клавіатурі. Ці кнопки використовуються для відображення набору параметрів, які користувач може вибрати на вбудованій клавіатурі для виконання певних завдань. Після натискання на кнопку бот отримає зворотний дзвінок. В нашому випадку воня потрібна для того щоб адміністратори могли взаємодіяти зі своєю панелю.

```
async def get_user_category_kb():
    kb = ReplyKeyboardMarkup(row_width=1, resize_key-
board=True)
    categories = await Category.get_list()
    for category in categories:
        kb.add(KeyboardButton(category.name))
    kb.add(KeyboardButton("Кошик"))

    return kb


async def create_product_kb(category_id: int):
    kb = InlineKeyboardMarkup()
    kb.add(InlineKeyboardButton(
        "Додати продукт",
        callback_data=product_callback.new(
```

```
            action=ProductActionEnum.create.value,
            category_id=category_id)),
    )
    return kb
```

ВИСНОВКИ

У рамках даної роботи було проведено огляд телеграм додатків для продажу продукції. Він допоміг оцінити переваги і недоліки таких додатків, що сприяло подальшій роботі.

Також було проаналізовано та обрано найбільш перспективні технології для розробки продукту. Ці технології мають першорядне значення для успішного виконання проекту. Вони надають необхідні інструменти та платформи для розробників. Створено структуру проекту та бази даних які є найбільш оптимальною для даної задачі. Розроблено схеми роботи адмінської та клієнтської панелі, по яким реалізовано логіку взаємодії з користувачами, адміністраторами та інтеграцію з базою даних.

В результаті чого було створено Ефективний Telegram-бот для підтримки онлайн-продажів який має зручний інтерфейсом для адміністраторів із можливостями ведення записів, керування продуктами та працювати з замовленнями. Клієнти в свою чергу можуть зручно подорожувати по меню, замовляти свої ордера що були додані в кошик та редагувати їх. Це розкриває основні можливості телеграм-ботів в електронній комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Why is web application must in today's business?. quora. [Електронний ресурс] Режим доступу до ресурсу: https://www.quora.com/What-language-to-choose-for-Telegram-bots#:~:text=Python:Python%20is%20one%20of,-telegram-bot%20and%20Telepot. (date of access: 31.05.2024).
2. Kestenbaum R. What Are Online Marketplaces And What Is Their Future?. Forbes. [Електронний ресурс] Режим доступу до ресурсу: https://www.forbes.com/sites/richardkestenbaum/2017/04/26/what-are-online-marketplaces-and-what-is-their-future/?sh=20200aa3284b
3. Платформи для продажу товарів через інтернет. weblium. [Електронний ресурс]Режим доступу до ресурсу: https://ua.weblium.com/blog/platformi-dlya-prodazhu-tovariv (дата звернення: 23.05.2024).
4. Black H. Telegram Marketing. Netpeak Journal – best articles and case studies on marketing and business development. [Електронний ресурс] Режим доступу до ресурсу: https://netpeak.net/blog/telegram-marketing-for-online-business-promotion-how-messenger-app-can-bring-you-money/ (date of access: 31.05.2024).
5. Telegram Marketing: A Guide For Beginners | Brand24. Brand24 Blog. [Електронний ресурс] Режим доступу до ресурсу: https://brand24.com/blog/telegram-marketing-guide/?psafe_param=1&adgr=performance-max-ii&keyword-ext=&placement=&location=1012864&adgr=performance-max-ii&gad_source=1&gclid=CjwKCAjw9cCyBhBzEiwAJTUWNThC62oNK0F9aZ9mZDqi7RbRZT7DJwlNRvVZcnKNcbdFDKA7ZcndsBoCVQIQAvD_BwE (date of access: 24.05.2024).

6. Telegram Channels. Telegram. [Електронний ресурс] Режим доступу до ресурсу: https://telegram.org/tour/channels#:~:text=Channels%20are%20a%20tool%20for,have%20the%20right%20to%20post. (date of access: 24.05.2024).
7. Group Chats on Telegram. Telegram. [Електронний ресурс] Режим доступу до ресурсу: https://telegram.org/tour/groups#:~:text=Telegram%20groups%20are%20a%20powerful,in%20touch%20with%20their%20investors. (date of access: 24.05.2024).
8. Чат-бот інтернет-магазину Magiccoffee. gerabot. [Електронний ресурс] Режим доступу до ресурсу: https://gerabot.com/en/case/magiccoffeebot (дата звернення: 23.05.2024).
9. Найкорисніші боти Telegram для українських користувачів. ITC.ua. [Електронний ресурс] Режим доступу до ресурсу: https://itc.ua/articles/samyie-poleznyie-botyi-telegram-dlya-ukrainskih-polzovateley/ (дата звернення: 23.05.2024).
10. Replacing sales consultants: Epicentr K has launched a unique Help-bot. Corezoid. [Електронний ресурс] Режим доступу до ресурсу: https://corezoid.com/blog/epicentr-has-launched-help-bot/ (date of access: 23.05.2024).
11. What language to choose for Telegram bots?. Quora. [Електронний ресурс] Режим доступу до ресурсу: https://www.quora.com/What-language-to-choose-for-Telegram-bots#:~:text=Python:Python%20is%20one%20of,-telegram-bot%20and%20Telepot. (дата звернення: 31.05.2024).
12. Hillard D. Practices of the Python Pro. Manning Publications Co. LLC, 2020. [Друкований ресурс]
13. Addepto. Addepto. [Електронний ресурс] Режим доступу до ресурсу: https://addepto.com/blog/quick-guide-to-programming-an-ai-chatbot/#:~:text=Java%20is%20a%20general-purpose,programming%20languages%20on%20the%20list.

14. Gupta A. 17 Most Popular Python IDEs in 2024: Code Like a Pro. Simplilearn.com. [Електронний ресурс] Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/python-tutorial/python-ide (дата звернення: 31.05.2024).
15. View aiogram on Snyk Open Source Advisor. Snyk Advisor. [Електронний ресурс] Режим доступу до ресурсу: https://snyk.io/advisor/python/aiogram (дата звернення: 31.05.2024).
16. Flask - One of the most popular Python frameworks. Data Science Courses | DataScientest. [Електронний ресурс] Режим доступу до ресурсу: https://datascientest.com/en/flask-one-of-the-most-popular-python-frameworks#:~:text=What%20is%20Flask?,web%20server,%20and%20cookie%20management. (дата звернення: 31.05.2024).
17. Oracle Database Advantages, Disadvantages and Features [Guide 2024]. The NineHertz:. [Електронний ресурс] Режим доступу до ресурсу: https://theninehertz.com/blog/advantages-of-using-oracle-database (дата звернення: 24.05.2024).
18. MySQL Advantages and Disadvantages - Blue Claw Database Developer Resource. Blue Claw Database Developer Resource. [Електронний ресурс] Режим доступу до ресурсу: https://blueclawdb.com/mysql/advantages-disadvantages-mysql/ (дата звернення: 24.05.2024).
19. PostgreSQL Tutorials | What is PostgreSQL, Advantages, Disadvantages. Code Topology . [Електронний ресурс] Режим доступу до ресурсу: https://codetopology.com/database/what-is-postgresql/ (дата звернення: 31.05.2024).
20. MongoDB : Advantages and Disadvantages - Infraveo Technologies. Infraveo Technologies. [Електронний ресурс] Режим доступу до ресурсу: https://www.infraveo.com/blog/mongodb-advantages-and-disadvantages/
21. Garcia-Molina H., Ullman J. D., Widom J. Database Systems : The Complete Book: International Edition. Pearson Education, Limited, 2008. 1248 p. [Друкований ресурс]

22. Welcome to aiogram’s documentation .[Електронний ресурс] Режим доступу до ресурсу: https://docs.aiogram.dev/en/v2.25.1/
23. Mudadla S. What is the use of .env. Medium. [Електронний ресурс] Режим доступу до ресурсу:https://medium.com/@sujathamudadla1213/what-is-the-use-of-env-8d6b3eb94843#:~:text=env%20file%20is%20used%20in,them%20directly%20into%20your%20code. (дата звернення: 31.05.2024).
24. SQLAlchemy: What is it? What's it for?. Data Science Courses | DataScientest. . [Електронний ресурс] Режим доступу до ресурсу: https://datascientest.com/en/sqlalchemy-what-is-it-whats-it-for#:~:text=SQLAlchemy%20is%20a%20tool%20based,in%20this%20case,%20Python). (дата звернення: 31.05.2024).
25. Engine Configuration – SQLAlchemy 2.0 Documentation. SQLAlchemy Documentation – SQLAlchemy 2.0 Documentation. [Електронний ресурс] Режим доступу до ресурсу: https://docs.sqlalchemy.org/en/20/core/engines.html#sqlalchemy.create_engine (дата звернення: 31.05.2024).
26. Class based handlers - aiogram 3.7.0 documentation. aiogram 3.7.0 documentation. [Електронний ресурс] Режим доступу до ресурсу: https://docs.aiogram.dev/en/latest/dispatcher/class_based_handlers/index.html (дата звернення: 31.05.2024).
27. Кінцевий автомат (FSM) - aiogram 3.7.0 documentation. aiogram 3.7.0 documentation. [Електронний ресурс] Режим доступу до ресурсу: https://docs.aiogram.dev/uk-ua/latest/dispatcher/finite_state_machine/index.html (дата звернення: 31.05.2024).
28. Keyboard buttons in Telegram bot Using Python - GeeksforGeeks. GeeksforGeeks. [Електронний ресурс] Режим доступу до ресурсу: https://www.geeksforgeeks.org/keyboard-buttons-in-telegram-bot-using-python/ (дата звернення: 31.05.2024).

ДОДАТОК А

core.py

```
import aiogram
from aiogram.contrib.fsm_storage.redis RedisStorage2
from aiogram.types import ParseMode

from bot.middlewares import AlbumMiddleware

dp: aiogram.Dispatcher
bot: aiogram.Bot


def create_dp(config: dict):
    global dp, bot
    bot = aiogram.Bot (config["bot_token"], parse_mode=Parse-
Mode.HTML, validate_token=True)
    storage = RedisStorage2()
    dp = aiogram.Dispatcher (bot, storage=storage)

    dp.setup_middleware(AlbumMiddleware())

    return dp
```

ДОДАТОК Б

models.py

```
from sqlalchemy import Column, ForeignKey, Integer, String, Boolean,
DECIMAL
from sqlalchemy.dialects.postgresql import ARRAY
from sqlalchemy.orm import relationship

from .base import Base, CreatedMixin, UpdatedMixin


class User(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "user"

    name = Column(String(100), nullable=True)
    user_name = Column(String(100), nullable=True)


class Category(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "category"

    name = Column(String(100), nullable=False)
    short_description = Column(String(4096))
    products = relationship(
        "Product",
        back_populates="category",
        cascade="all, delete",
        passive_deletes=True,
    )


class Product(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "product"

    category_id = Column(Integer, ForeignKey("category.id",
ondelete="CASCADE"), nullable=False)
    photo = Column(ARRAY(String), nullable=True)
    name = Column(String(100), nullable=False)
    description = Column(String(4096), nullable=False)
    short_description = Column(String(4096))
    price = Column(DECIMAL, nullable=False)
    weight = Column(ARRAY(Integer), nullable=False)

    category = relationship("Category", back_populates="products")
    orders = relationship(
        "Order",
        back_populates="product",
        cascade="all, delete",
        passive_deletes=True,
    )


class Order(Base, CreatedMixin, UpdatedMixin):
    __tablename__ = "order"
```

```
    user_id = Column(Integer, nullable=False)
    product_id = Column(Integer, ForeignKey("product.id",
ondelete="CASCADE"), nullable=False)
    buy_status = Column(Boolean, default=False)
    weight = Column(ARRAY(Integer), nullable=False)

    product = relationship("Product", back_populates="orders")
```

ДОДАТОК В

category.py

```
from bot.core import dp
from bot.db.decorators import session_decorator
from bot.db.models import Category
from bot.enums import CategoryActionEnum
from bot.filters.admin import IsSuperAdmin
from bot.kb.admin_kb import category_callback, cate-
gory_edit_callback, get_category_kb, edit_category_kb
from bot.states import CategoryState


@dp.callback_query_handler(
    IsSuperAdmin(), category_callback.filter(action=CategoryAc-
tionEnum.view_sub_category.value)
)
@session_decorator(add_param=False)
async def category_list(cq: types.CallbackQuery) -> None:
    kb = await get_category_kb()
    await cq.message.answer(
        "Меню категорій", reply_markup=kb
    )
    await cq.answer()


@dp.callback_query_handler(
    IsSuperAdmin(), category_callback.filter(action=CategoryAc-
tionEnum.create.value)
)
@session_decorator(add_param=False)
async def create_category_btn(cq: types.CallbackQuery) -> None:
    await CategoryState.name.set()
    await cq.message.answer("Введіть назву категорії:")
    await cq.answer()


@dp.message_handler(IsSuperAdmin(), state=CategoryState.name)
@session_decorator(add_param=False)
async def load_name(msg: types.Message, state: FSMContext):
    if await Category.exists(None, name=msg.text):
        return await msg.answer("Чай з такою назвою уже існує,
введіть іншу")
    async with state.proxy() as data:
        data["name"] = msg.text
    await CategoryState.short_description.set()
    await msg.reply("Введіть короткий опис:")


@dp.message_handler(IsSuperAdmin(), state=Catego-
ryState.short_description)
@session_decorator(add_param=False)
```

```
async def load_short_description(msg: types.Message, state:
FSMContext):
    data = await state.get_data()
    short_description = msg.text
    await Category.create(**data, short_description=short_de-
scription)
    await state.finish()
    await msg.answer("Категорію створено ✅")


@dp.callback_query_handler(
    IsSuperAdmin(), category_callback.filter(action=CategoryAc-
tionEnum.delete.value)
)
@session_decorator(add_param=False)
async def delete_category(cq: types.CallbackQuery,
callback_data: dict) -> None:
    category = await Category.get(int(callback_data["cate-
gory_id"]))
    await category.delete()
    await cq.message.answer("Категорію видалено ✅")


@dp.callback_query_handler(
    IsSuperAdmin(), category_callback.filter(action=CategoryAc-
tionEnum.edit.value)
)
@session_decorator()
async def category_edit_btn(
        cq: types.CallbackQuery, callback_data: dict, state:
FSMContext
) -> None:
    category_id = callback_data.get("category_id")
    async with state.proxy() as data:
        data["category_id"] = category_id

    category = await Category.get(int(data["category_id"]))

    await cq.message.answer(
        f"Виберіть що відредагувати в «{category.name}»",
        reply_markup=await edit_cate-
gory_kb(int(callback_data["category_id"]))
    )


@dp.callback_query_handler(
    IsSuperAdmin(), category_edit_callback.filter(action=Cate-
goryActionEnum.edit_name.value)
)
@session_decorator()
async def category_edit_name(
        cq: types.CallbackQuery, callback_data: dict, state:
```

```
FSMContext
) -> None:
    await CategoryState.edit_name.set()

    category_id = callback_data.get("category_id")
    async with state.proxy() as data:
        data["category_id"] = category_id

    category = await Category.get(int(data["category_id"]))

    await cq.message.answer(f"Поточна назва
категорії:\n{category.name}")
    await cq.message.answer("Введіть нову назву\nДля скасування
- /cancel")


@dp.message_handler(IsSuperAdmin(), state=Catego-
ryState.edit_name)
@session_decorator()
async def category_edit_name(msg: types.Message, state: FSMCon-
text):
    async with state.proxy() as data:
        category_id = int(data["category_id"])
    edited_category = await Category.get(category_id)

    await edited_category.update(name=msg.text)
    await state.finish()
    await msg.answer("Назву категорії відредаговано ✅")


@dp.callback_query_handler(
    IsSuperAdmin(), category_edit_callback.filter(action=Cate-
goryActionEnum.edit_short_description.value)
)
@session_decorator()
async def category_edit_short_description(
        cq: types.CallbackQuery, callback_data: dict, state:
FSMContext
) -> None:
    await CategoryState.edit_short_description.set()

    category_id = callback_data.get("category_id")
    async with state.proxy() as data:
        data["category_id"] = category_id

    category = await Category.get(int(data["category_id"]))

    await cq.message.answer(f"Поточний короткий
опис:\n{category.short_description}")
    await cq.message.answer("Введіть новий короткий опис \nДля
скасування - /cancel")
```

```
@dp.message_handler(IsSuperAdmin(), state=Catego-
ryState.edit_short_description)
@session_decorator()
async def product_edit_short_description(msg: types.Message,
state: FSMContext):
    async with state.proxy() as data:
        category_id = int(data["category_id"])
    edited_category = await Category.get(category_id)

    await edited_category.update(short_description=msg.text)
    await state.finish()
    await msg.answer("Короткий опис категорії відредаговано ✅")
```

admin_kb.py

```
from     aiogram.utils.callback_data     import     CallbackData

from           bot.db.models           import           Category
from   bot.enums   import   CategoryActionEnum,   ProductActionEnum

category_callback  =  CallbackData("category",  "action",  "cate-
gory_id")
category_edit_callback = CallbackData("category", "action", "cat-
egory_id")
product_callback  =  CallbackData("product",  "action",  "cate-
gory_id")
product_edit_callback = CallbackData("product", "action", "prod-
uct_id")

admin_kb              =              InlineKeyboardMarkup().add(
    InlineKeyboardButton(
        "Категорії",
        callback_data=category_callback.new(
            action=CategoryActionEnum.view_sub_category.value,
category_id=0,
        ),
    ),
)


async        def        get_product_kb(category_id:        int):
    kb                  =                  InlineKeyboardMarkup()
    kb.row(InlineKeyboardButton(
        "Видалити",
        callback_data=product_callback.new(
            action=ProductActionEnum.delete.value,
            category_id=category_id)),
        InlineKeyboardButton(
```

```
        "Редагувати",
        callback_data=product_callback.new(
            action=ProductActionEnum.edit.value,
            category_id=category_id))
    )
    return                                                    kb


async       def       create_product_kb(category_id:       int):
    kb                  =                  InlineKeyboardMarkup()
    kb.add(InlineKeyboardButton(
        "Додати                                          продукт",
        callback_data=product_callback.new(
            action=ProductActionEnum.create.value,
            category_id=category_id)),
    )
    return                                                    kb


async        def        edit_product_kb(product_id:        int):
    kb            =            InlineKeyboardMarkup(row_width=1)
    kb.add(InlineKeyboardButton(
            "Редагувати                                    назву",
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_name.value,
                product_id=product_id)),
        InlineKeyboardButton(
        "Редагувати                                        опис",
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_description.value,
                product_id=product_id)),
        InlineKeyboardButton(
            "Редагувати              короткий              опис",
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_short_descrip-
tion.value,
                product_id=product_id)),
        InlineKeyboardButton(
            "Редагувати                                    фото",
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_photo.value,
                product_id=product_id)),
        InlineKeyboardButton(
            "Редагувати                                    ціну",
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_price.value,
                product_id=product_id)),
        InlineKeyboardButton(
            "Редагувати                                    вагу",
```

```
            callback_data=product_edit_callback.new(
                action=ProductActionEnum.edit_weight.value,
                product_id=product_id)),
    )
    return kb


async def get_category_kb():
    categories = await Category.get_list()
    kb = InlineKeyboardMarkup()
    for category in categories:
        kb.row(InlineKeyboardButton(category.name,
                                    callback_data=cate-
gory_callback.new(action=CategoryActionEnum.detail.value,

category_id=category.id)),
               InlineKeyboardButton("Видалити",
                                    callback_data=cate-
gory_callback.new(action=CategoryActionEnum.delete.value,

category_id=category.id)),
               InlineKeyboardButton("Редагувати",
                                    callback_data=cate-
gory_callback.new(action=CategoryActionEnum.edit.value,

category_id=category.id)))
    kb.add(InlineKeyboardButton("Додати категорію",
                                callback_data=cate-
gory_callback.new(action=CategoryActionEnum.create.value,

category_id=0)))

    return kb


async def edit_category_kb(category_id: int):
    kb = InlineKeyboardMarkup(row_width=1)
    kb.add(InlineKeyboardButton("Редагувати назву",
                                callback_data=cate-
gory_edit_callback.new(action=CategoryAc-
tionEnum.edit_name.value,

category_id=category_id)),
           InlineKeyboardButton("Редагувати короткий опис",
                                callback_data=cate-
gory_edit_callback.new(action=CategoryActionEnum.edit_short_de-
scription.value,

category_id=category_id)),
```

```
    )
    return                                                    kb


def                                                skip_photo_kb():
    kb        =        ReplyKeyboardMarkup(resize_keyboard=True)
    kb.add(KeyboardButton("Пропустити                          фото"))
    return kb
```

user_kb.py

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardMarkup, InlineKeyboardButton
from aiogram.utils.callback_data import CallbackData

from bot.db.models import Category, Product
from bot.enums import BuyActionEnum

buy_callback = CallbackData("buy", "action", "product_id")
delete_callback = CallbackData("buy", "action", "order_id",
"msg_id")
add_basket_callback = CallbackData("buy", "action", "prod-
uct_id", "weight")
cart_callback = CallbackData("go_to_the_cart", "action")


def delete_order_kb(order_id: int, msg_id: int):
    kb = InlineKeyboardMarkup()
    kb.add(InlineKeyboardButton(
        "Видалити продукт",
        callback_data=delete_callback.new(
            action=BuyActionEnum.delete_basket.value,
            order_id=order_id,
            msg_id=msg_id)))
    return kb


async def add_basket_product_kb(product_id: int):
    product = await Product.get(None, id=product_id)

    kb = InlineKeyboardMarkup()

    kb.row(InlineKeyboardButton(f"{product.weight[0]}g",
                                callback_data=add_bas-
ket_callback.new(action=BuyActionEnum.add_basket.value,

product_id=product_id,

weight=product.weight[0])),
```

```
            InlineKeyboardButton(f"{product.weight[1]}g",
                                 callback_data=add_bas-
ket_callback.new(action=BuyActionEnum.add_basket.value,

product_id=product_id,

weight=product.weight[1])),
            InlineKeyboardButton(f"{product.weight[2]}g",
                                 callback_data=add_bas-
ket_callback.new(action=BuyActionEnum.add_basket.value,

product_id=product_id,

weight=product.weight[2])),
            InlineKeyboardButton(f"{product.weight[3]}g",
                                 callback_data=add_bas-
ket_callback.new(action=BuyActionEnum.add_basket.value,

product_id=product_id,

weight=product.weight[3])))

    return kb


async def get_user_category_kb():
    kb = ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    categories = await Category.get_list()
    for category in categories:
        kb.add(KeyboardButton(category.name))
    kb.add(KeyboardButton("Кошик"))

    return kb


async def get_category_products_kb(category_id: int):
    kb = ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    products = await Product.get_list(category_id=category_id)
    for product in products:
        kb.add(KeyboardButton(product.name))
    kb.add(KeyboardButton("Кошик"))
    kb.add(KeyboardButton("Назад до Каталогу Чаю"))

    return kb


async def get_basket_products_kb():
    kb = ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    kb.add(KeyboardButton("Замовити"))
    kb.add(KeyboardButton("Очистити кошик"))
    kb.add(KeyboardButton("Назад до Каталогу Чаю"))

    return kb
```

```
async def go_to_cart_kb():
    kb = InlineKeyboardMarkup()
    kb.row(InlineKeyboardButton("Перейти в кошик",
callback_data=cart_callback.new(action=BuyAc-
tionEnum.go_cart.value)))

    return kb
```