МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Інтернет магазин в додатку Telegram»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Власенко О.В.**

Студента групи ІН – 62 **Воробйов І.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-62 спеціальності «Інформатика» денної форми навчання Воробйова Івана Олександровича.

Тема: " Інтернет магазин в додатку Telegram "

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) інформаційний огляд інтернет-магазинів та постановка завдання; 2) аналіз та вибір методів вирішення поставленого завдання; 3) опис та аналіз результатів роботи.

Дата видачі завдання "_____"_____2020 р

.

Керівник випускної роботи _____ Власенко О.В.

Завдання прийняв до виконання _____ Воробйов І.О.

РЕФЕРАТ

Записка: 39 стор., 15 рис., 3 додаток, 7 джерел.

Об’єкт дослідження – бот в ”Telegram”.

Мета роботи – дослідження і розробка додатку в месенджері на мові Python із використанням адміністративної панелі.

Предметна область – Інтернет-торгівля в месенджері.

Результати – розроблено бота в “Telegram” із використанням Python, JSON, SQLite та інше. Бота можна знайти по його імені, або QR коду.

БОТ ІНТЕРНЕТ-МАГАЗИН В ДОДАТКУ ТЕЛЕГРАМ

ЗМІСТ

ВСТУП

Інтернет-торгівля – процес реалізації фізичних і нефізичних товарів за допомогою спеціалізованих електронних майданчиків, які надають дистанційне оформлення замовлення. На даний момент інтернет-торгівля в усьому світі розвивається дуже стрімко і досить успішно. За підсумками досліджень ринку інтернет на 2019 рік - в Україні обсяг продажів за 4 період постійно збільшується як і збільшується кількість магазинів електронної торгівлі. А якщо розглядати зарубіжні інтернет майданчики такі як Amazon, то працівники-аналітики з Bank of America прогнозуються збільшення коштів на 22% компанії Amazon.

Отже, роблячи вибір торгового онлайн-майданчика, продавець полегшує своє завдання, а саме доставку товарів до покупців. Сервіси які вже досить давно на ринку мають свою, добре налагоджену, логістику, а значить продавцю залишається тільки домовитися про відправку і координувати процес отримання посилки.

У випускній роботі не будуть розглядатися переваги і недоліки електронної торгівлі в порівнянні з традиційною – на сучасному етапі все значно складніше, ніж це описується в роботах, розміщених в мережі Інтернет. Відмітимо лише, що електронна торгівля – особливий вид торгівлі, який може доповнювати традиційну торгівлю, а може існувати абсолютно самостійно.

Актуальність випускної роботи полягає у новизні розробленого програмного продукту, де онлайн-магазин переноситься у Telegram. Як висновок, можна сказати, шо торгові онлайн-майданчики досить активно розвиваються разом з електронною комерцією. А бізнес, який розвивається, або адаптується, під сучасний ринок e-commerce, має досить великі шанси розвиватись динамічніше, що являється досить важливим фактором в період невизначеності.

Метою випускної роботи є розробка і програмна реалізація bota для ресторану “Simbiosi”.

1. ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Інтернет-магазин – частина торгового підприємства / організації торгівлі або торгова організація, призначена для надання покупцеві за допомогою мережі Інтернет відомостей, необхідних при здійсненні покупки, в тому числі про асортимент товарів, ціни, продавця, способи та умови оплати і доставки, для прийому від покупців за допомогою мережі Інтернет повідомлень про намір придбати товари, а також для забезпечення можливості доставки товарів продавцем або його підрядником, за вказаною покупцем адресою або до пункту самовивозу [1].

Інформаційна система – це взаємозв'язана сукупність засобів, методів і персоналу, яка використовується для зберігання, обробки та видачі інформації для досягнення мети управління. В сучасних умовах основним технічним засобом обробки інформації є персональний комп'ютер. Більшість сучасних інформаційних систем перетворюють не інформацію, а дані. Тому часто їх називають системами обробки даних [2].

1.1 Основні функції інтернет-магазину

Електронний магазин можна розглядати як прикладну систему, яка побудована як елемент технології системи електронної комерції. Подібно до звичайного магазину, електронний магазин реалізує наступні основні функції: представлення товарів (послуг) покупцю, обробку замовлень, продаж і доставку товарів.

У мережі часто зустрічається інше визначення цього поняття, більш примітивне, – сайт, який торгує товарами і послугами в мережі інтернет. Електронний магазин дозволяє людям сформувати замовлення, не виходячи с дому, на покупку, вибрати спосіб оплати і спосіб доставки замовлення [3].

Головною відмінністю електронного магазину від звичайного магазину є його розташування і організація взаємодії з покупцем, використання мережі

Інтернет для здійснення всіх можливих операцій. Інтернет-магазин розглядається як більш прийнятна і комплексна, в той же час, складна в реалізації, система інтернет-торгівлі, яка охоплює всі основні бізнес-процеси організації торгівлі.

У цілому основні функції електронної-торгівлі – інформаційне обслуговування клієнта, обробка замовлення, прийняття/проведення платежів, а також обробка і збір статистичної інформації. Програмний комплекс управління онлайн-торгівлі дає змогу формувати як інтерфейс з покупцем, так і функціональні можливості інтернет-магазину, виходячи з потреб власників.

Виходячи зі сказаного про e-commerce додатку то його структура можна подати наступним чином (рис. 1.1):

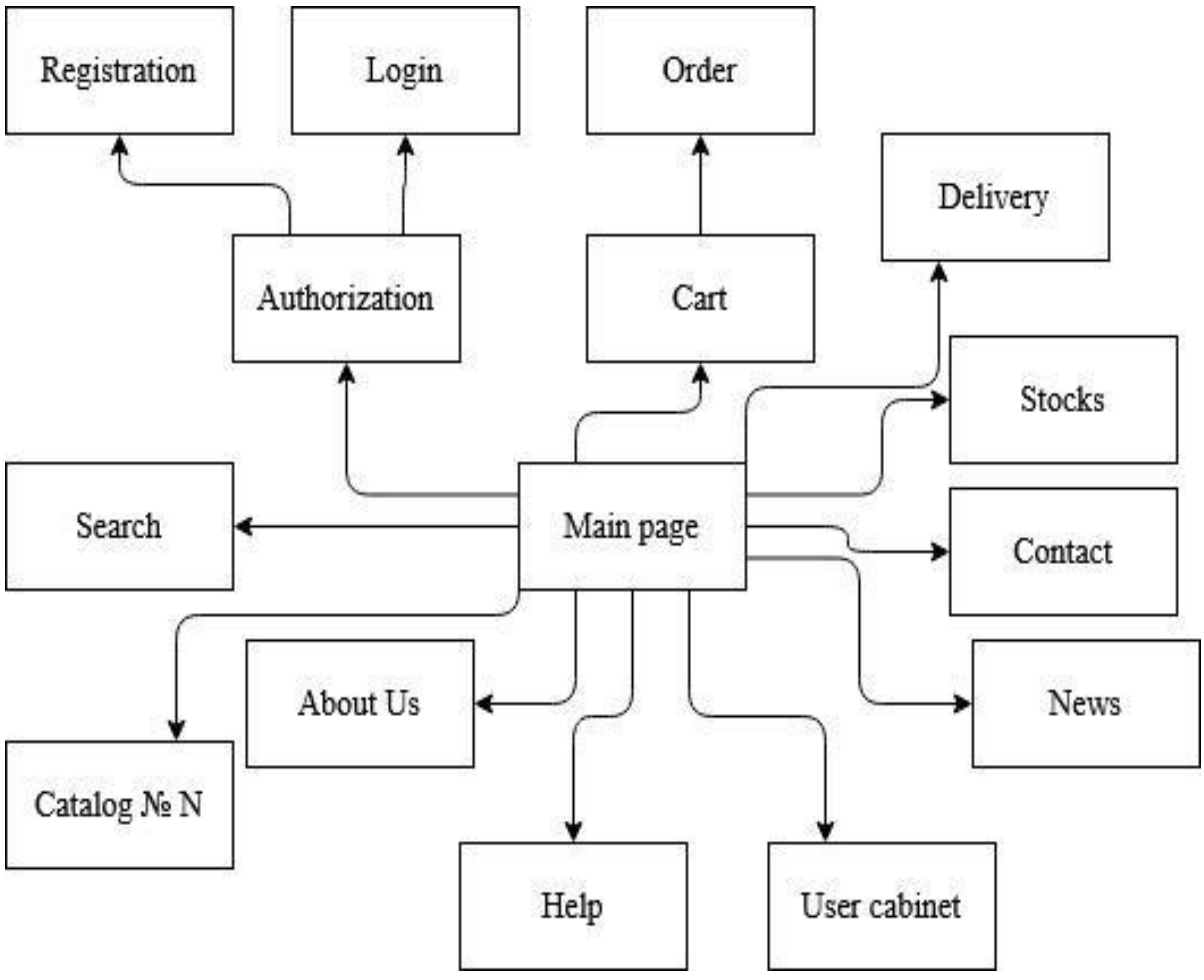

Рисунок 1.1 – Структура інтерфейсної частини e-commerce

1.2 Використання додатку “Telegram”

З кожним днем у додатку «Telegram» реєструються все більше і більше людей. За даними статистики число користувачів по світу наближається до 1 мільярду. Популярність цього додатка не піддається сумніву, так як він забезпечує безпеку і анонімність користувачів. Додаток вже давно перестав бути просто месенджером, у ньому є канали, пабліки, боти. Всі ці фактори дають таку популярність додатком.

1.2.1 API Telegram bot

Варто почати з поняття API. Прикладний програмний інтерфейс (інтерфейс програмування застосунків, інтерфейс прикладного програмування) – набір визначених взаємодій різнотипного програмного забезпечення. API – це зазвичай метод абстракції між низькорівневим та високорівневим програмним забезпеченням.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, API може передавати інформацію(за стандартом JSON) в відмінному від стандартного HTML форматі, завдяки чому ним зручно користуватись при написанні власних програм, більш легко контролювати зміни. Сторонні загальнодоступні API найчастіше віддають дані в одному з двох форматів: XML або JSON. Застосування JSON набагато більш лаконічний и простий в читанні, ніж XML, а сервіси, що Надаються доступ до даних до XML-форматі, поступово відмовляються від останнього.

1.2.2 Telegram-bot як платформа

Telegram-bot − це додаток, розміщений на сервері, що використовує API бота Telegram для підключення до клієнтів Telegram Messenger.

Telegram-bot «розуміє» що розпочався сеанс з користувачем за допомогою спеціальних тестових команд (за замовчування стандартні для будь-якого бота),

або за допомогою кнопок, поміщених в бота (в кнопки вбудовано код JSON). Додаток бота, запущений на сервері, може бути чим завгодно.

Великою перевагою ботів Telegram є те, що вони не вимагають установки і працюють на всіх комп'ютерних платформах, де працює Telegram Messenger (Windows, Mac, Linux, Android, iOS і все веб-браузери).

Боти в «Telegram», вже давно вийшли за межі стандартних роботів соціальних мережах. Один бот, може замінити собою не один десяток сервісів – скачування відео, завантаження книг, музики, можливість грати прямо в «Telegram», робити ставки, або заробляти без вкладень.

У Telegram використовується загальний вид ботів, які діляться на кілька напрямків:

- чат-боти - це API в веб-системах, які можуть виступати у ролі ігр;
- боти-асистенти, використовуються онлайн-сервісами як доповнення до основної веб-версії.

Популярність телеграм ботів обумовлена тим, що все знаходитися під рукою, і доступ до будь-яких популярних сервісів може бути доступна в одному додатку. Люди завжди для зручності робили все в одному, месенджер телеграм - дає таку можливість.

1.3 Використання CMS для створення інтернет-магазину

У сучасному світі існує велика кількість готових рішень для створення інтернет магазинів. Їх установка, модернізація може відрізнятися одна від одної, але як правило це − установка плагіна на движок і використання спеціальних платформ для e-commerce.

Порівняння вимоги реалізації випускної роботи і популярних відкритих рішень не є коректним, так як сфери розгортання відрізняються, але беручи до уваги факт того, що скрізь є перегляд товару, додавання в корзину, сторінка адміністратора, то можна зробити оціночну характеристику.

На даний момент популярними рішеннями є наступні (кінець 2019 − початок 2020):

- WooCommerce;
- Magento;
- OpenCart;
- PrestaShop;
- Shop-Script;
- CS-Cart;
- «1С-Битрикс»;
- Joomla!;
- Drupal;
- MODX.

Розглянемо найбільш поширені з них:

WooCommerce – одна з найпоширеніших систем для електронної торгівлі, яка дозволяє запустити магазин на базі WordPress [3]. Встановлюючи даний рушій отримуємо все в розпорядження, а саме: створення каталогів, формування каталогів для фільтрації при пошуку, підключення онлайн-оплати, система знижок, способи доставки. За стандартом плагін є безкоштовним, при необхідності функціонал можна розширити, але за це доведеться платити;

Magento – рушій з відкритим вихідним кодом, призначений для створення великих інтернет-магазинів. Його головні характеристики − функціональність і гнучкість. У базовій комплектації пропонує повний набір інструментів для управління магазином. Для розширення функціональності доступні доповнення, які можна завантажити з офіційного каталогу Magento;

OpenCart – популярна безкоштовна CMS, тому на більшості хостингів вона розгортається з адміністративної панелі в один клік, можна завантажити з вільною ліцензією з офіційного сайту, після чого отримувати оновлення. Головна особливість OpenCart − будова за принципом MVC, який передбачає поділ даних, інтерфейсу і логіки на три компонента.

Таким розглянувши три найпопулярніших рішення на сьогоднішній день, можна прийти до висновку, що їх код перебуває у відкритому доступі, а значить власники таких рішень можуть бути піддані загрозам втрати даних і хакерським атакам. Навпаки ж, рішення представлене у випускній роботі є унікальним, і не використовує готових CMS.

1.4 Постановка задачі

Для випускної роботи сформовані наступні завдання для роботи:

- Розробити інтернет-магазин;
- Розробити бота у месенджері Telegram;
- Вибрати та налаштувати сервер;
- Вибрати мову програмування для реалізації;
- Обрати тему для формування каталогу в інтернет-магазині;
- Створити адміністративну панель для бота.

У результаті має бути створений додаток, який буде максимально простий у використанні, для охоплення більшої аудиторії.

2 МЕТОДИ ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Клієнт-серверні технології

Для реалізації програми будуть використані наступні технології: мова програмування Python, Telegram bot API, UWSGI, Ubuntu server.

2.1.1 Мова програмування Python

Python − об’єктно-орієнтована інтерпретована мова програмування високого рівня і зі строгою типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв’язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільований, так і у вихідній формі на всіх основних платформах. У мові програмування Python підтримується кілька парадигм програмування, зокрема: об’єктно-орієнтованість, процедурність, функціональна та аспектно-орієнтованість [7, 4].

Серед основних переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносимість програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python у діалоговому режимі (дуже корисне для експериментування та розв’язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище
- розробки IDLE написані на мові Python;

- зручний для розв’язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Таким чином доцільно виконувати розробку Telegram-bot за допомогою мови Python, тому що існує бібліотека, з великою кількістю готових функцій [5, 7].

2.1.2 Серверна операційна система Ubuntu

Операційна система Ubuntu − широко використовується багатьма користувачами, так як вона проста в налаштуванні. Було вирішено використати дану ОС, так як інформація по налаштуванню і роботі знаходиться у відкритому доступі, велика кількість корисних форумів, блогів і всього іншого. Вагома перевага: її найчастіше використовують на десктопах, таким чином спрощується робота з сервером. Крім того Ubuntu Linux з повною упевненістю можна вважати безпечною системою. Вона захищена як від вторгнень, так і від вірусів.

Важлива перевага установки Linux на сервер, тобто ОС, яка не має графічного інтерфейсу, полягає в її відмінній надійності, яка пов'язана з меншою кількістю функціонуючих компонентів. Зокрема, система Windows не буде завантажуватися при невірно встановленому драйвері монітора, чого не можна сказати про операційну систему Linux, яка відмінно працює і без відеокарти [6].

У ОС Linux сервер управління виконується за допомогою утиліт команд «telnet» і «ssh», отже, відсутні такі загрози, які мають місце при локальному адмініструванні. Це одна із вагомих переваг даної системи перед ОС Windows, яка більшій мірі виключає вразливість графічних веб-навігаторів «Internet Explorer», «Mozilla», які є зспільними для обох ОС, поштових клієнтів і інших GUI-додатків.

Здійснення повноцінного віддаленого адміністрування ОС Windows допустимо тільки на локальному рівні із застосуванням графічного інтерфейсу. Це, звичайно, зручніше для користувача, однак, це значніше завантажує ресурси

пам’яті та процесора за рахунок великої кількості процесів. Усе це в значній мірі обмежує можливості додатків серверів. Якщо при цьому сервер розташовується в єдиній мережі з ПК-клієнтом, то взаємодія буде більш-менш швидкою. Однак при низькій швидкості з’єднання робота буде значно ускладнена.

Серед фінансових переваг ОС Linux треба відзначити низьку вартість пристроїв. У свою чергу високий рівень захисту, надійність і керованість системи дозволяє зменшити витрати на обслуговування серверів. Що стосується необхідних сервісів, а саме proxy, веб-, поштового сервера, антивірусних програм, засобів безпеки, сервера IP-телефонії, то все це абсолютно безкоштовно входить в дистрибутив Linux. Оплата здійснюється лише за те, що встановлюється на Linux-сервера і налаштування додатків, а не за ліцензії. Якщо говорити про ОС Windows, то у даному випадку, щоб отримати подібного роду програми, необхідно витратити коштів на порядок більше. Наприклад, сьогодні ціна Windows Server 2003 R2 Standard Edition дорівнює приблизно 1000 доларів.

2.1.3 Nginx як веб сервер

Nginx почав набирати популярність з моменту релізу завдяки незначним розмірам (light-weight resource utilization) і можливості легко масштабуватися. Nginx добре підходить для передачі статичного контенту і спроектований так, щоб передавати динамічні запити іншому ПО призначеному для їх обробки [6].

Адміністратори часто вибирають Nginx через його ефективне споживання ресурсів і швидким відкликом під навантаженням, а також через можливість використовувати його і як веб-сервер, так і як проксі.

Nginx створює воркер-процеси, кожен з яких може обслуговувати тисячі з'єднань. Досягаються такі результату завдяки механізму швидких циклів, у яких перевіряються і оброблюються події. Відокремлення основної роботи від обробки з'єднань дозволяє кожному воркеру займатися своїм процесом і переходити на обробку з'єднань тільки тоді коли створилась нова подія [2].

Кожне з'єднання, яке обробляється воркером, поміщається в event loop разом з іншими з’єднаннями. У цьому циклі події обробляються асинхронно,

коли з'єднання закривається воно видаляється з циклу. Такий підхід до обробки з'єднань дозволяє Nginx'у неймовірно масштабуватися при обмежених ресурсах. Оскільки сервер однопотоковий і він не створює процеси під кожне з'єднання, то і використання пам'яті і CPU відносно рівномірне, навіть при високих навантаженнях.

Так як інтерпретатор не вбудований до кожного воркера, то оверхед, пов'язаний з цим, буде мати місце тільки при запитах до динамічного контенту. Статичний контент буде повернений клієнтові простим способом і запити до інтерпретатора будуть виконуватися тільки тоді коли вони потрібні. Так як Nginx не дозволяє перевизначати конфігурації на рівні директорій, він може обробляти запити швидше, адже йому досить зробити один directory lookup і прочитати один файл конфігурацій на кожен запит (передбачається, що файл знайдений там де він повинен бути за згодою).

Перевага використання також пов'язана з безпекою. Розподілена конфігурація на рівні директорій в Apache покладає відповідальність за безпеку на звичайних користувачів, які навряд чи здатні вирішити цю задачу якісно. Те що адміністратор контролює весь сервер запобігає помилки безпеки, які можуть виникнути якщо дати користувачам доступ до конфігурації.

2.2 Проектування баз даних

У проекті базою даних послужить – SQLite3. Вона є вбудованою для Python-проекту. Дана база даних є лідером в надійності, не вимагає спеціальних налаштувань, і займає мало місця. Саме за це вона прекрасно підійде для старту. Діаграма бази даних виглядає наступним чином (рис. 2.1):

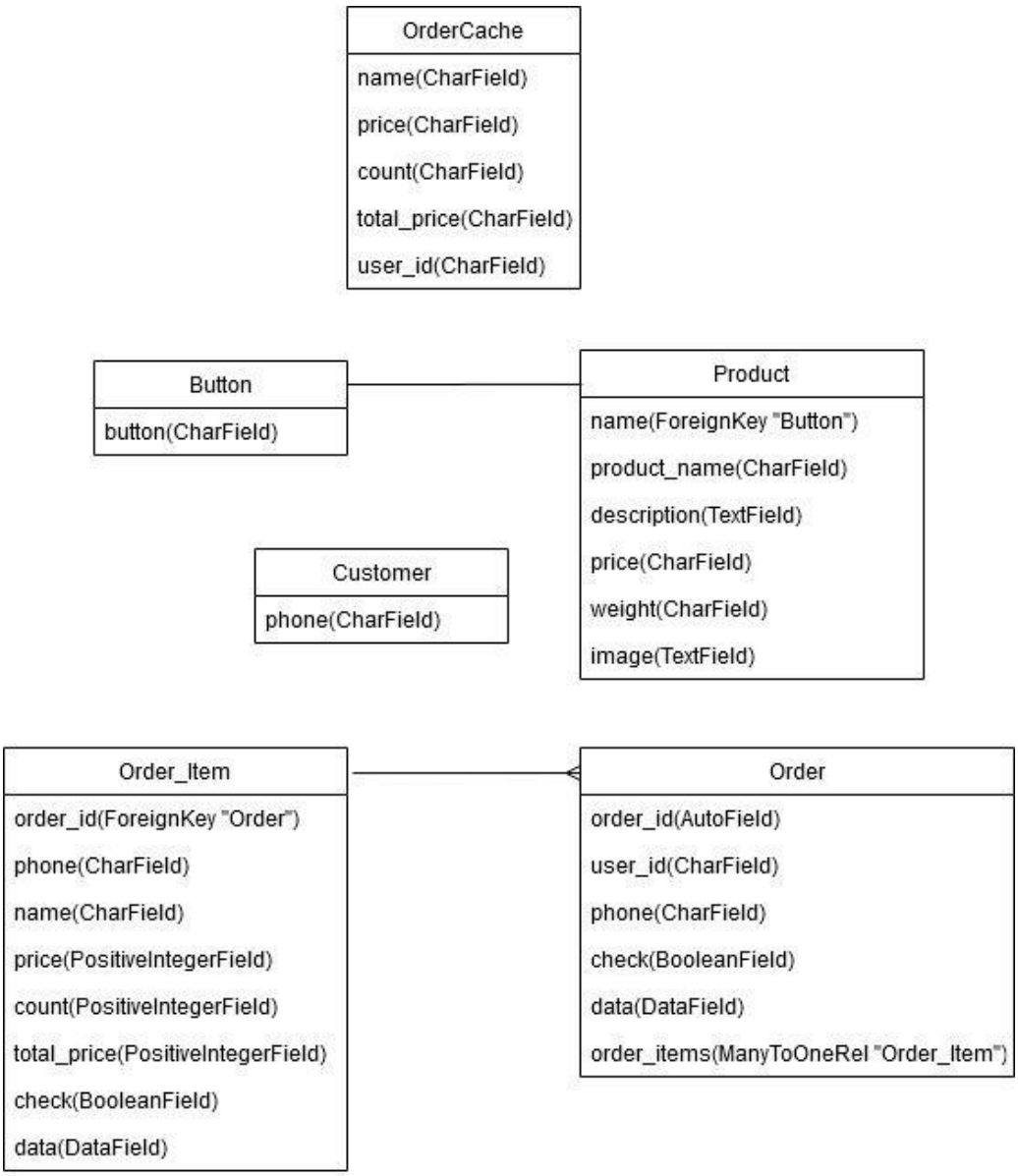

Рисунок 2.1 – Діаграма бази даних

Так як для реалізації було використано Python + Django, то база даних створена у вигляді ORM, її код наведено у Додаток А.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура інтерфейсу випускної роботи

Структура інтерфейсної частини випускної роботи (рис 3.1):

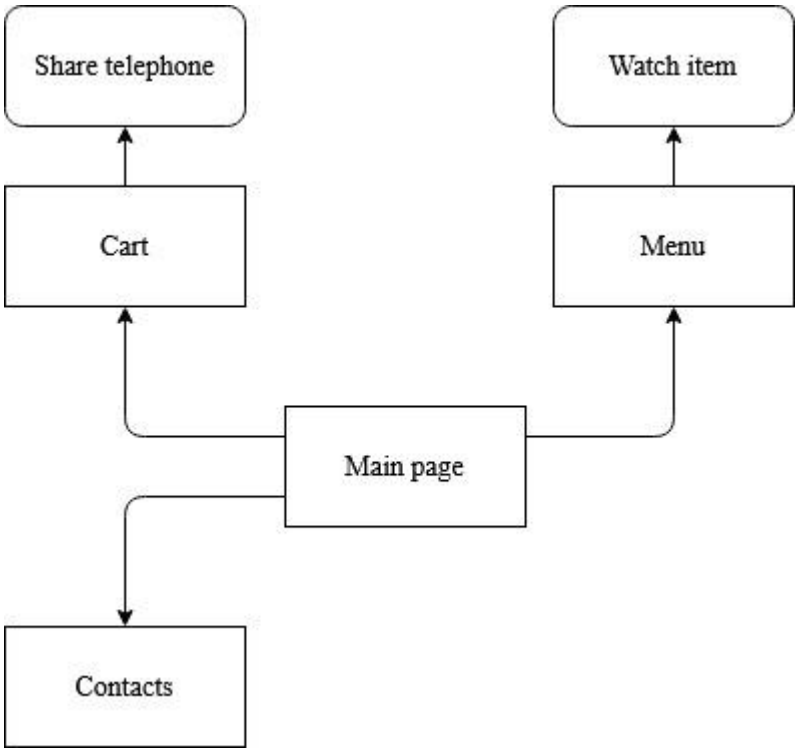

Рисунок 3.1 – Структура інтерфейсу дипломної роботи

Як можна бачити, інтерфейс є менш завантаженим, від чого легше і приємніше для сприйняття.

3.2 Дизайн и структура додатку

Далі приведені скріншоти робочого додатку, які демонструють працездатність системи. Стартова сторінка при вході в бота виглядає наступним чином (рис. 3.2).

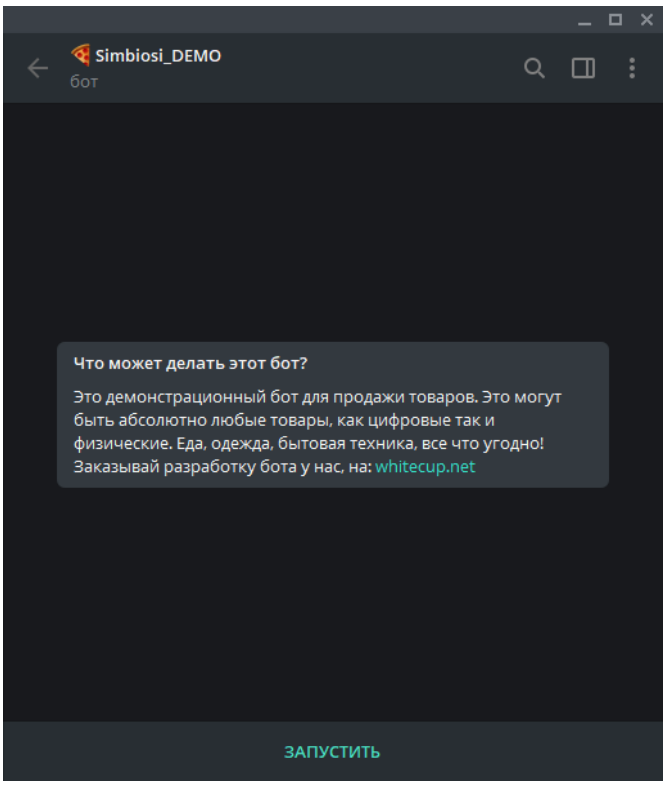

Рисунок 3.2 – Стартова сторінка додатка

Після натискання на кнопку "Запустить" перенаправляють на головну сторінку бота (рис. 3.3).

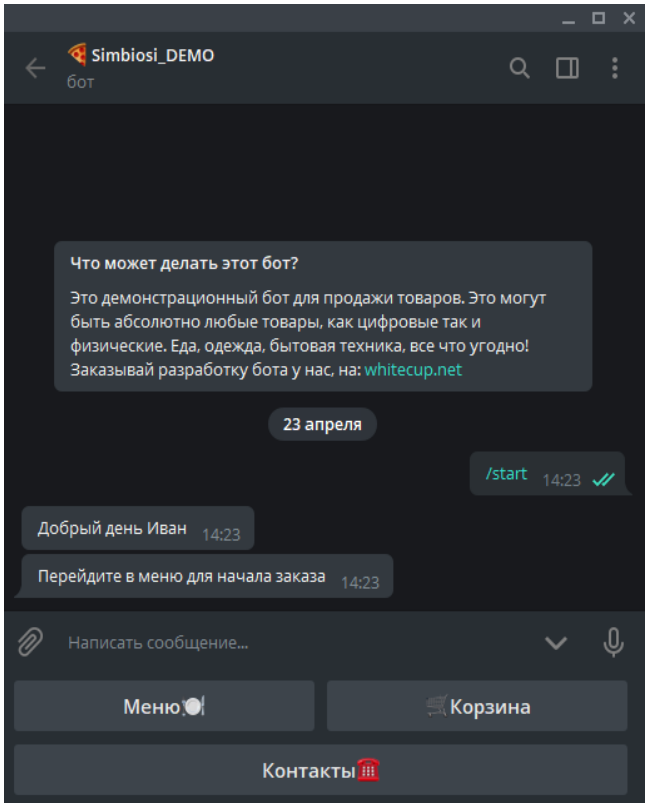

Рисунок 3.3 – Головна сторінка додатка

При натисканні на кнопку "Контакты" зможемо побачити контакти закладу (рис. 3.4)

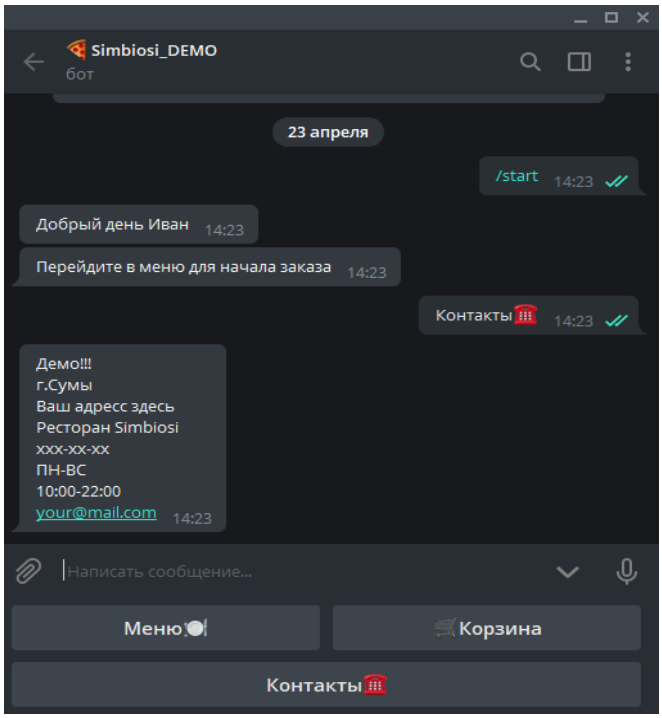

Рисунок 3.4 – Інформація вкладки "Контакты"

При натисканні на кнопку "Меню" відкриється інформація про існуюче меню (рис. 3.5):

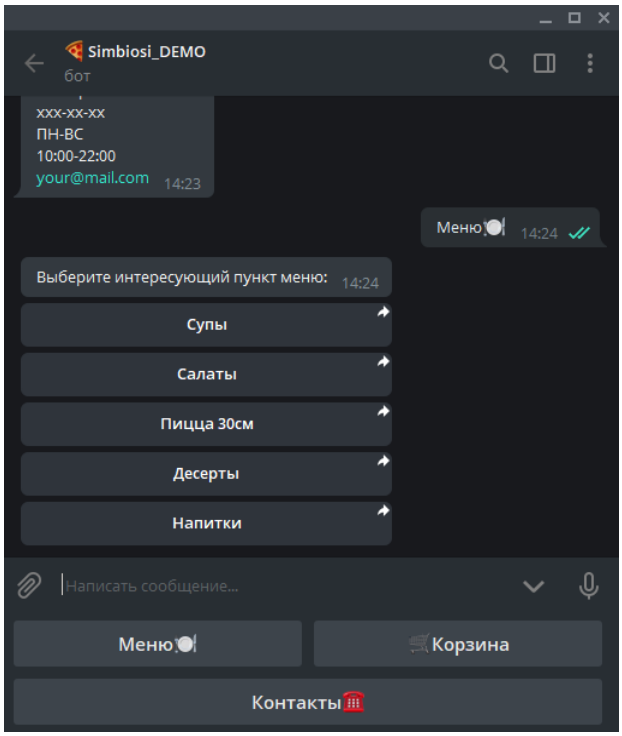

Рисунок 3.5 – Інформація вкладки "Меню"

При виборі категорії меню із запропонованого списку з'являться страви відповідної категорії, для прикладу взята категорія "Десерты" (рис. 3.6):

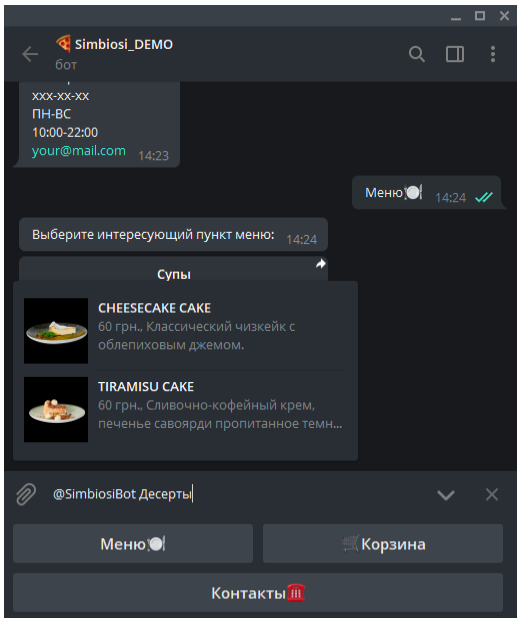

Рисунок 3.6 – Інформація після натискання на кнопку "Десерты"

Для прикладу виберемо страву – "CHEESECAKE CAKE". Після його вибору бачимо товар з можливістю додавання його в кошик (рис. 3.7). Також виведена інформація про його ціну, вагу порції і складовими.

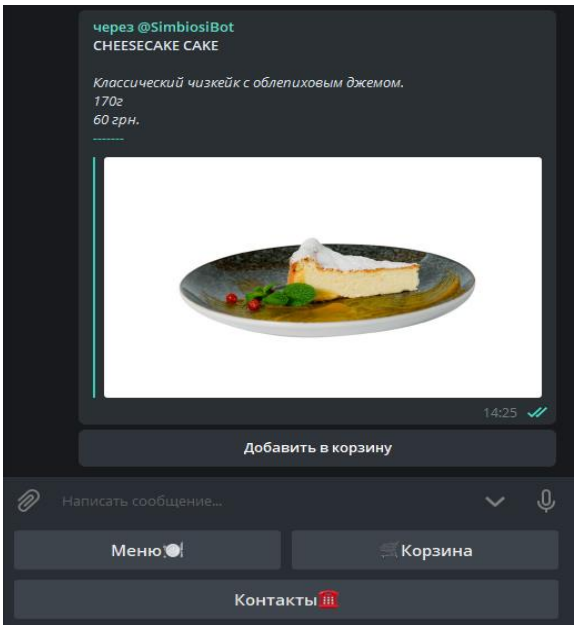

Рисунок 3.7 – Інформація про страву

Після переходу в “Корзину”, буде виведений перелік доданих товарів, для управління замовленням - кількістю порцій. З даного пункту можемо перейти до пункту "Перейти до оформлення замовлення" (рис. 3.8)

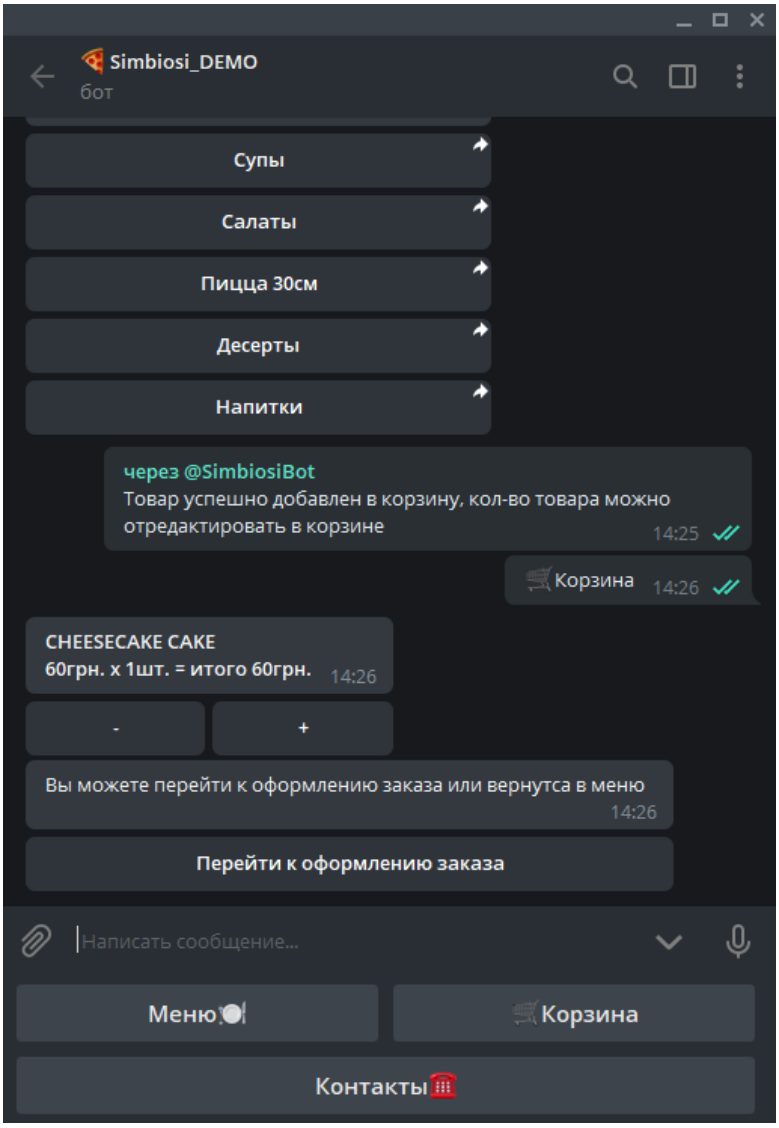

Рисунок 3.8 – Зовнішній вигляд після переходу в “Корзину”

При натисканні на кнопку "Перейти к оформлению заказа" бот пропонує поділитися контактними даними. (рис. 3.9)

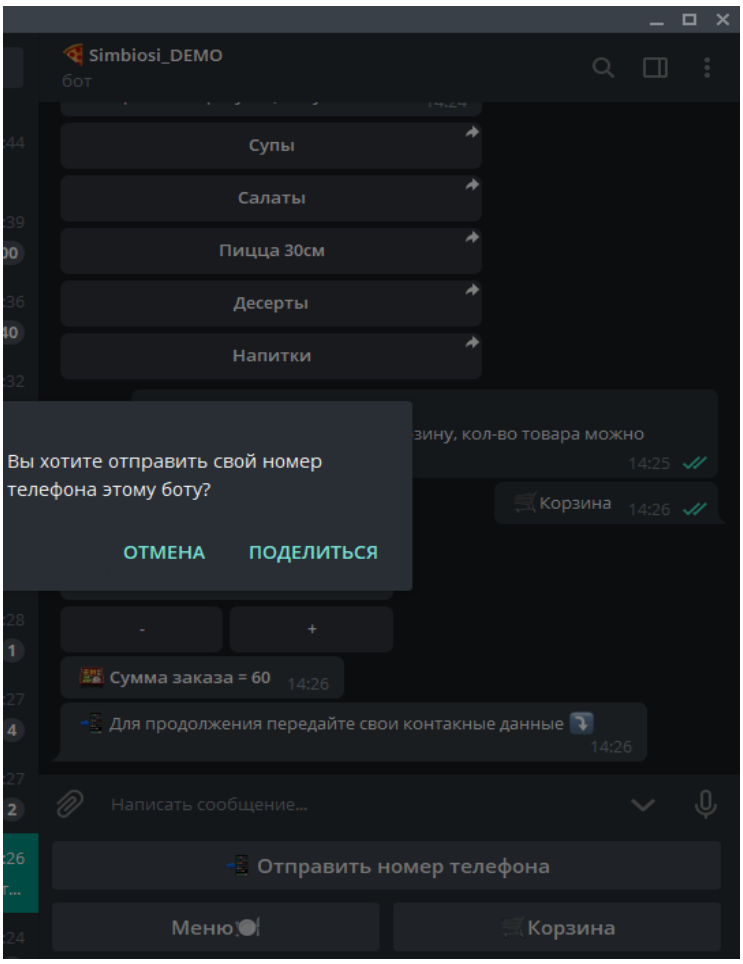

Рисунок 3.9 – Пропозиція поділитися контактними даними

Після чого успішно реєструється замовлення, про що повідомляє бот (рис. 3.10):

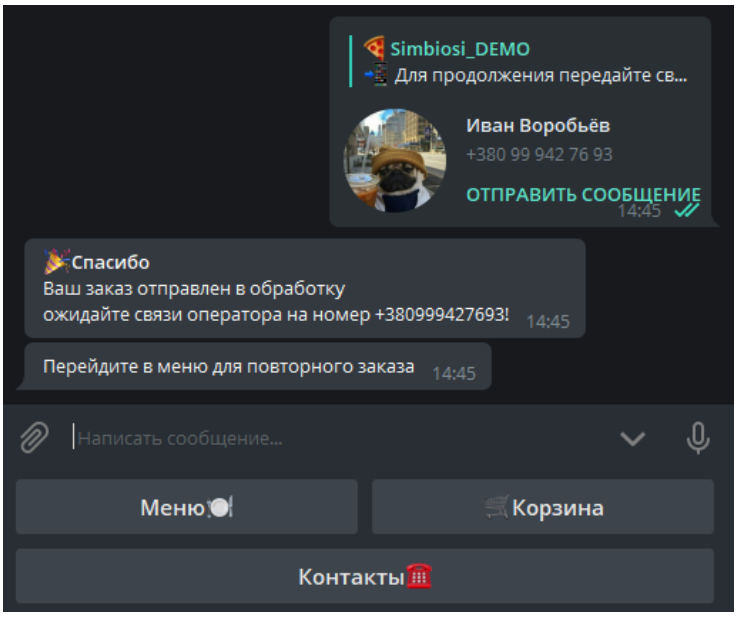

Рисунок 3.10 – Успішне підтвердження замовлення

3.3 Панель адміністратора

Як було сказано раніше, для адміністрування використовується Django. Панель адміністратора відкриває багато можливостей: створення, зміна, видалення інформації. Перевага Django в тому, що навіть не підготовлений адміністратор з легкістю може контролювати процес. Головна сторінка адміністратора виглядає наступним чином (рис. 3.11). На ній можна перейти в будь-який пункт, для перегляду інформації.

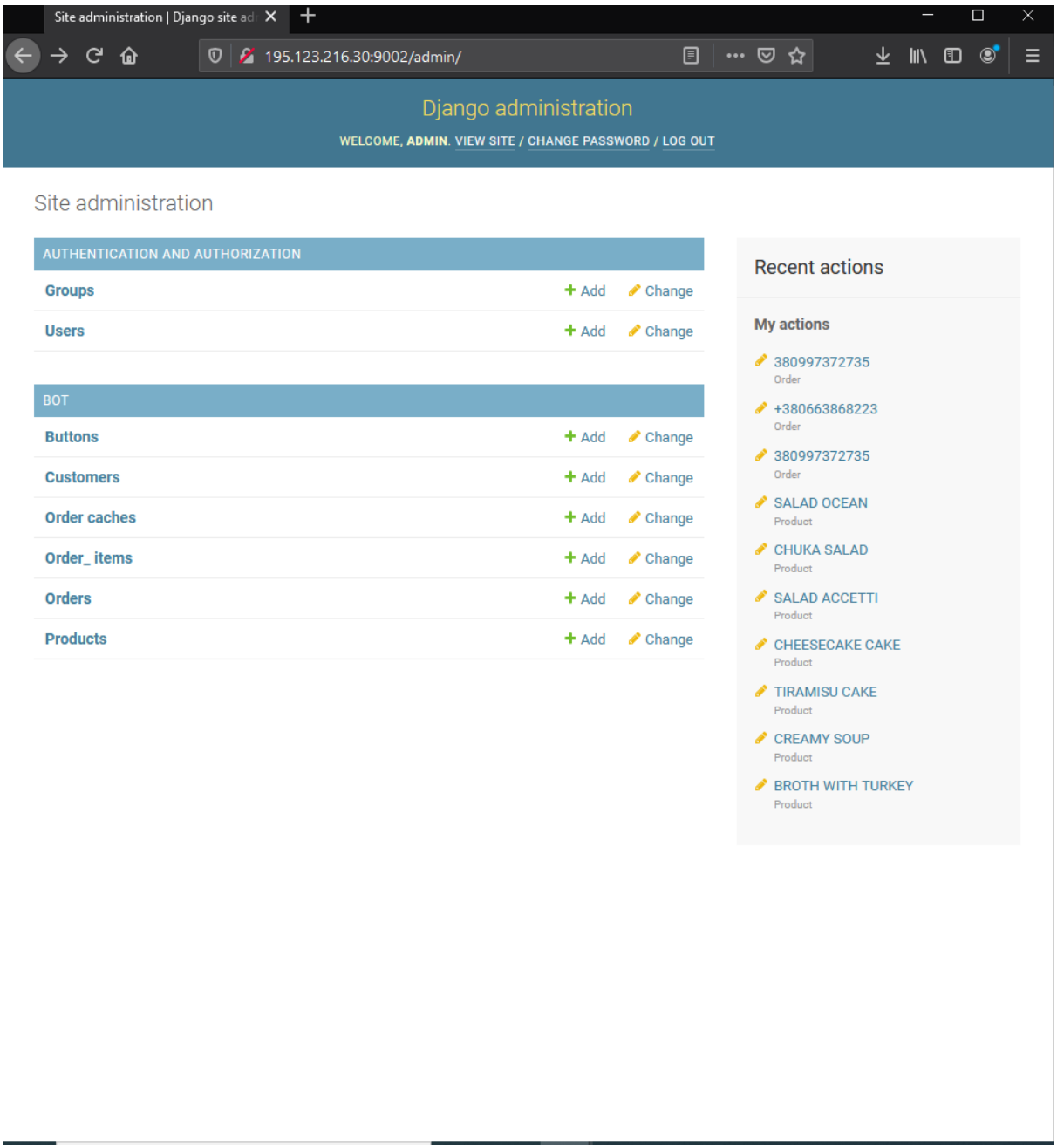

Рисунок 3.11 – Головна сторінка адміністратора

Для прикладу перейдемо до пункту "Buttons". Тут знаходитися інформація про "Меню" ресторану (рис. 3.12).

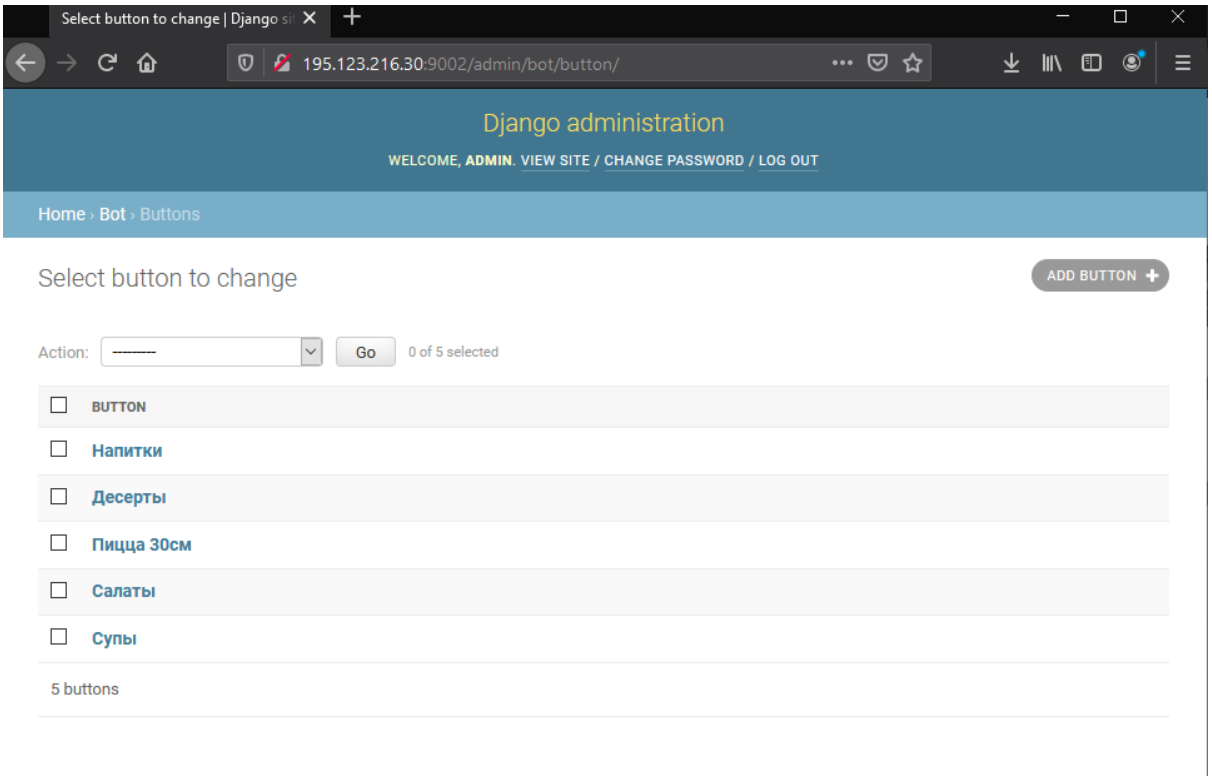

Рисунок 3.12 – Сторінка“Buttons”

Перейшовши в категорію "Напитки", можемо змінити назву, тим самим змінивши даний пункт в Telegram bot (e) (рис. 3.13).

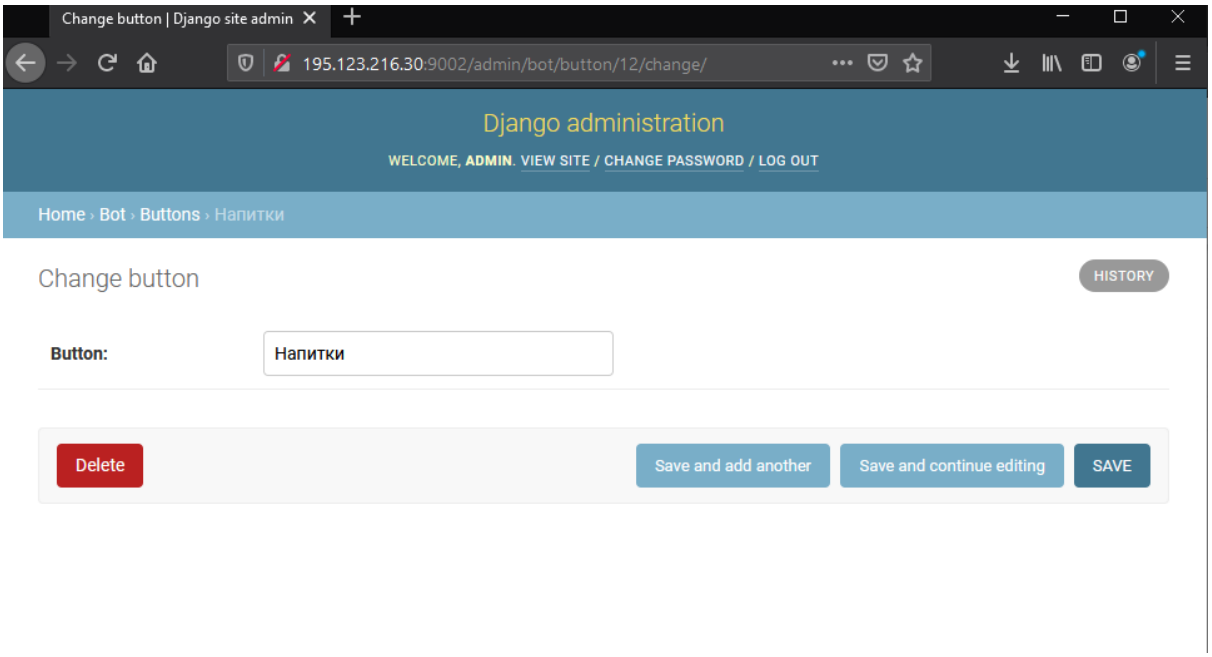

Рисунок 3.13 – Сторінка редагування категорії "Напитки"

3.4 Функції, які використовуються у додатку

Дані функції використовуються у програмі, написаній на мові Python, які використовуються при розробці:

- def start_def – функція яка запускається при відкритті бота.
- def hello_def – “ виводить повідомлення для привітання.
- def get_button_def – “ відповідає за показ навігаційної панелі: “Меню”, “Корзина”, “Контакты”, “Перейдите в меню для начала заказа”.
- def get_again_button_def – повторює функцію “def get_button_def”, при повторному виклику.
- def get_button_def – виклик функції для відображення наявного меню.
- def query – функція для відображення позиції продукту.
- def add_to_order_menu_def – функція для додавання товару в корзину.
- def get_garbage_def – “ для виведення вмісту кошика.
- def add_in_order_def – “ для управління кількістю товару або його видаленням з кошика.
- def get_geophone_def – “ для відправки телефону.
- def contact – “ підтвердження замовлення.
- def get_garbage_def – “ виведення контактної інформації.

3.5 Deploy додатку

Telegram бот вже перебуває в мережі, у Додатку В наведено QR код для підключення. Використовується робоча версія для демонстрації працездатності. На даний час унікальних юзерів більше тридцяти. Під час використання бота дефектів не виявлено, кожний юзер успішно проходить всі етапи, від привітання бота, до підтвердження замовлення.

ВИСНОВКИ

Під час виконання випускної роботи було виконане наступне:

- Був проведений літературний огляд за тематикою “Створення інтернет-магазину у месенджері”.
- Мовою програмування був обраний – Python, за легкість роботи с JSON, велику кількість модулів, те що створення бота саме на цій мові програмування не викликає складностів. Присутній простий, але досить функціональний framework – Django, для створення панелі адміністратора.
- На сервері використовуються Nginx + Ubuntu Server. Ці компоненти не являються “важкими” для придбаного серверу, ця зв’язка є досить легка у налаштуванні, оновленні та підтримці.
- Результатом роботи являється розроблений бот в “Telegram”, знайти його можна за назвою – “@SimbiosiBot”, або просканувати QR код в Додатку В.
- Поставлені задачі до розробки виконані у повній мірі, бот виконує роль інтернет-магазину. Він працездатний, та знаходиться в мережі Інтернет.

СПИСОК ЛІТЕРАТУРИ

1. Інтернет-магазин [Електронний ресурс] URL: https://www.searchengines.ru/v_rossii_poyavilos.html
2. Інформаційна система [Електронний ресурс] URL: https://www.yaklass.ru/materiali?chtid=455&mode=cht
3. Існуючі рішення [Електронний ресурс] URL: https://uguide.ru/rejting-cms-dlja-internet-magazina-vybiraem-luchshij-dvizhok-dlja-onlajn-torgovli
4. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. М. : Финансы и статистика, 2009. 176 с.
5. Maikle Doyson. Programming on Python – 2016 – с.0-40.
6. Кенин Александр Михайлович. Самоучитель системного администратора – 2016 – с.0-60.
7. Python [Електронний ресурс] URL: https://www.python.org/

ДОДАТОК А

Файл - bot_ne_bot.py

```
from typing import Dict, List, Any

import telebot
import requests
import json
import time
import urllib
import re
from telebot import types

bot = telebot.TeleBot('********:**********')
ip = '**************'


@bot.message_handler(commands=['start'])
def start_def(message):
    hello_def(message)


def hello_def(message):
    text = "Добрый день {}".format(message.from_user.first_name)
    bot.send_message(message.chat.id, text)
    get_button_def(message)


def get_button_def(message):
    key = types.ReplyKeyboardMarkup(True, False)
    key.row("Меню", "Корзина")
    key.row("Контакты☎")
    bot.send_message(message.chat.id, "Перейдите в меню для начала заказа", reply_markup=key)


def get_again_button_def(message):
    key = types.ReplyKeyboardMarkup(True, False)
    key.row("Меню", "Корзина")
    key.row("Контакты☎")
    bot.send_message(message.chat.id, "Перейдите в меню для повторного заказа", reply_markup=key)


@bot.message_handler(func=lambda message: message.text == "Меню")
def get_menu_def(message):
    try:
        link = requests.get('http://{ip}/api/button'.format(ip=ip))
        data = json.loads(link.text)
        in_key = types.InlineKeyboardMarkup()
        for i in range(len(data['list'])):
            call_name = data['list'][i]['name']
            button = types.InlineKeyboardButton(text=call_name, switch_inline_query_current_chat=call_name)

            in_key.add(button)

        bot.send_message(message.chat.id, 'Выберите интересующий пункт меню:', reply_markup=in_key)
    except:
        pass


@bot.inline_handler(func=lambda query: True)
def query(query):
    try:
```

```
    link =
'http://{ip}/ap
i/products'.for
mat(ip=ip)
    text =
{'text':
query.query}
    req =
requests.post(l
ink,
data=json.dumps
(text))
    data =
json.loads(req.
text)
    caller =
[]
    for i in
range(len(data[
'list'])):

    call_id =
data['list'][i]
['id']

    call_title
=
data['list'][i]
['name']

    call_desc
=
data['list'][i]
['desc']
        price
=
data['list'][i]
['price']

    weight =
data['list'][i]
['weight']

    image_url
=
data['list'][i]['
image']
        in_key
=
types.InlineKeybo
ardMarkup()

    add_button =
types.InlineKeybo
ardButton(text="Д
обавить в
корзину",

    callback_dat
a='add_product_{c
all_title}'.forma
t(

    call_title=c
all_title))

    in_key.add(a
dd_button)
        try:

    caller.appen
d(types.InlineQue
ryResultArticle(

    id=call_id,

    title=call_t
itle,

    description=
'{price} грн.,
{call_desc}'.form
at(price=price,
call_desc=call_de
sc),

    thumb_url=im
age_url,
thumb_width=48,
thumb_height=48,

    input_message_cont
ent=types.InputTextMess
ageContent(

    message_text='<b>{
call_title}</b>'

    '&#010;&#010;'

    '<i>{call_desc}</i
>'

    '&#010;'

    '<i>{call_weight}<
/i>'

    '&#010;'

    '<i>{call_price}
грн.</i>'

    '&#010;'

    '<a
href="{call_image}">---
----</a>'.format(

    call_title=call_ti
tle,
call_desc=call_desc,
call_weight=weight,

    call_price=price,
call_image=image_url),
parse_mode='html',
disable_web_page_previe
w=False),

    reply_markup=in_ke
y))
        except:
            pass

    bot.answer_inline_
```

```
query(query.id,
caller,
cache_time=0)
     except:
     pass


@bot.callback_q
uery_handler(fu
nc=lambda call:
'add_product_'
in call.data)
def
add_to_order_me
nu_def(call):
     if
call.inline_mes
sage_id:
     product =
re.match(r"(^(a
dd_product_)(.+
))",
call.data).grou
p(3)
     link =
'http://{ip}/ap
i/send_order_ca
che'.format(ip=
ip)
     text =
{'text':
product,
'user_id':
call.from_user.
id}
     req =
requests.post(l
ink,
data=json.dumps
(text))

     bot.edit_m
essage_text(inl
ine_message_id=
call.inline_mes
sage_id,
     text="Товар
успешно добавлен
в корзину, кол-во
товара можно
отредактировать в
корзине")


@bot.message_hand
ler(func=lambda
message:
message.text ==
"Корзина")
def
get_garbage_def(m
essage):
     try:
     link =
'http://{ip}/api/
get_order_cache'.
format(ip=ip)
     text =
{'user_id':
message.from_user
.id}
     req =
requests.post(lin
k,
data=json.dumps(t
ext))
     data =
json.loads(req.te
xt)
     if not
data['list']:

     bot.send_mes
sage(chat_id=mess
age.chat.id,
text='Ваша
корзина пуста
\nНажмите
кнопку: Меню для
выбора
продукции\n ⬇')
     else:
          for i in
range(len(data['list'])
):

     call_name =
data['list'][i]['name']

     call_price =
data['list'][i]['price'
]

     call_count =
data['list'][i]['count'
]

     call_total_price =
data['list'][i]['total_
price']
               in_key =
types.InlineKeyboardMar
kup()

     add_button_minus =
types.InlineKeyboardBut
ton(text="-",
callback_data='count_ch
ange_-
_{call_name}'.format(

     call_name=call_nam
e))

     add_button_plus =
types.InlineKeyboardBut
ton(text="+",
callback_data='count_ch
ange_+_{call_name}'.for
mat(

     call_name=call_nam
e))

     in_key.add(add_but
```

```
ton_minus,
add_button_plus
)

     bot.send_m
essage(message.
chat.id,
text='<b>{call_
title}</b>'

     '&#010;'

     '<b>{call_
price}грн. x
{count}шт. =
итого
{prod_sum}грн.<
/b>'.format(

     call_title
=call_name,
call_price=call
_price,
count=call_coun
t,
prod_sum=call_t
otal_price),

     parse_mode
='html',

     disable_we
b_page_preview=
False,
reply_markup=in
_key)
          key =
types.InlineKey
boardMarkup()

     ad_button
=
types.InlineKey
boardButton(tex
t="Перейти к
оформлению
заказа",
callback_data='ma
ke_order')

     key.add(ad_b
utton)

     bot.send_mes
sage(message.chat
.id, "Вы можете
перейти к
оформлению заказа
или вернутся в
меню",
reply_markup=key)
     except:
     pass



@bot.callback_que
ry_handler(func=l
ambda call:
'count_change_-_'
in call.data)
def
add_in_order_def(
call):
     product =
re.match(r"(^(cou
nt_change_-
_)(.+))",
call.data).group(
3)
     try:
          link =
'http://{ip}/api/
del_order_cache'.
format(ip=ip)
          text =
{'text': product,
'user_id':
call.from_user.id
}
          req =
requests.post(lin
k,
data=json.dumps(text))
          link =
'http://{ip}/api/get_pr
oduct_in_order_cache'.f
ormat(ip=ip)
          text =
{'text': product,
'user_id':
call.from_user.id}
          r =
requests.post(link,
data=json.dumps(text))
          data =
json.loads(r.text)
          if not
data['list']:

     bot.edit_message_t
ext(chat_id=call.messag
e.chat.id,
message_id=call.message
.message_id,
text="Товар успешно
удален из корзины")
          else:
               for i in
range(len(data['list'])
):

     call_name =
data['list'][i]['name']

     call_price =
data['list'][i]['price'
]

     call_count =
data['list'][i]['count'
]

     call_total_price =
data['list'][i]['total_
price']

     in_key =
```

```
types.InlineKeyboardMarkup()

    add_button_minus = types.InlineKeyboardButton(text="-",

    callback_data='count_change_-_{call_name}'.format(

    call_name=call_name))

    add_button_plus = types.InlineKeyboardButton(text="+",

    callback_data='count_change_+_{call_name}'.format(

    call_name=call_name))

    in_key.add(add_button_minus, add_button_plus)

    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,

    text='<b>{call_title}</b>'

    '&#010;'

    '<b>{call_price}грн. x {count}шт. = итого {prod_sum}грн.</b>'.format(

    call_title=call_name, call_price=call_price, count=call_count,

    prod_sum=call_total_price),

    parse_mode='html',

    disable_web_page_preview=False, reply_markup=in_key)
    except:
        pass

@bot.callback_query_handler(func=lambda call: 'count_change_+_' in call.data)
def add_in_order_def(call):
    product = re.match(r"(^(count_change_\+_)(.+))", call.data).group(3)
    try:
        link = 'http://{ip}/api/send_order_cache'.format(ip=ip)
        text = {'text': product, 'user_id': call.from_user.id}
        req = requests.post(link, data=json.dumps(text))
        link = 'http://{ip}/api/get_product_in_order_cache'.format(ip=ip)
        text = {'text': product, 'user_id': call.from_user.id}
        r = requests.post(link, data=json.dumps(text))
        data = json.loads(r.text)
        for i in range(len(data['list'])):

    call_name = data['list'][i]['name']

    call_price = data['list'][i]['price']

    call_count = data['list'][i]['count']

    call_total_price = data['list'][i]['total_price']
```

```
     in_key =
types.InlineKey
boardMarkup()

     add_button
_minus =
types.InlineKey
boardButton(tex
t="-",

     callback_d
ata='count_chan
ge_-
_{call_name}'.f
ormat(

     call_name=
call_name))

     add_button
_plus =
types.InlineKey
boardButton(tex
t="+",

     callback_d
ata='count_chan
ge_+_{call_name
}'.format(

     call_name=
call_name))

     in_key.add
(add_button_min
us,
add_button_plus
)

     bot.edit_m
essage_text(cha
t_id=call.messa
ge.chat.id,
message_id=call
.message.message_
id,

     text='<b>{ca
ll_name}</b>'

     '&#010;'

     '<b>{call_pr
ice}грн. x
{count}шт. =
итого
{prod_sum}грн.</b
>'.format(

     call_name=ca
ll_name,
call_price=call_p
rice,
count=call_count,

     prod_sum=cal
l_total_price),

     parse_mode='
html',

     disable_web_
page_preview=Fals
e,
reply_markup=in_k
ey)
     except:
          pass

@bot.callback_que
ry_handler(func=l
ambda call:
call.data ==
'make_order')
def
get_geophone_def(
call):
     key =
types.ReplyKeyboa
rdMarkup(row_widt
h=1,
resize_keyboard=True)
     button_phone =
types.KeyboardButton(te
xt="□ Отправить номер
телефона",
request_contact=True)
     key.row(button_pho
ne)
     key.row("Меню□",
"□Корзина")
     link =
'http://{ip}/api/get_or
der_cache'.format(ip=ip
)
     text = {'user_id':
call.from_user.id}
     r =
requests.post(link,
data=json.dumps(text))
     data =
json.loads(r.text)
     total_order_price
= 0
     for i in
range(len(data['list'])
):
     call_total_price =
data['list'][i]['total_
price']
     total_order_price
+=
int(call_total_price)

     bot.edit_message_t
ext(chat_id=call.messag
e.chat.id,
message_id=call.message
.message_id,

     text='<b>□ Сумма
заказа =
{totat_order_sum}</b>'.
format(totat_order_sum=
total_order_price),
```

```
    parse_mode
='html',
disable_web_pag
e_preview=False
)
    bot.send_m
essage(call.fro
m_user.id, "
Для продолжения
передайте свои
контакные
данные ",
reply_markup=ke
y)

@bot.message_ha
ndler(content_t
ypes=['contact'
])
def
contact(message
):
    if
message.contact
is not None:
    link =
'http://{ip}/ap
i/get_order_cac
he'.format(ip=i
p)
    text =
{'user_id':
message.from_us
er.id}
    r =
requests.post(l
ink,
data=json.dumps(t
ext))
    data =
json.loads(r.text
)
    link =
'http://{ip}/api/
send_order'.forma
t(ip=ip)
    order =
{'user_id':
message.from_user
.id, 'phone':
message.contact.p
hone_number,
'list':
data['list']}
    r =
requests.post(lin
k,
data=json.dumps(o
rder))

    bot.send_mes
sage(message.from
_user.id,
text='<b>Спасибо
</b>'

    '&#010;'

    'Ваш заказ
отправлен в
обработку'

    '&#010;'

    'ожидайте
связи оператора
на номер
{phone}!'.format(phone=
message.contact.phone_n
umber),parse_mode='html
',
disable_web_page_previe
w=False)

    get_again_button_d
ef(message)

@bot.message_handler(fu
nc=lambda message:
message.text ==
"Контакты☎")
def
get_garbage_def(message
):
    try:

    bot.send_message(m
essage.chat.id,

    'Демо!!!\nг.Сумы\n
Ваш адресс
здесь\nРесторан
Simbiosi\nxxx-xx-
xx\nПН-ВС\n10:00-
22:00\nyour@mail.com')
    except:
    pass


while True:
    try:
    bot.polling()
    except Exception:
    time.sleep(10)
```

Файл - **urls.py**

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
```

```
    path('admin/', admin.site.urls),
    path('api/', include('bot.urls'))
]
```

Файл - **admin.py**

```
from django.contrib import admin
from .models import Button, Product, OrderCache, Order, Customer, Order_Item
# Register your models here.
admin.site.register(Button)
admin.site.register(Customer)

class OrderItemsTabularInline(admin.TabularInline):
    model = Order_Item
    fields = ('name', 'price', 'count', 'check')

class OrderAdmin(admin.ModelAdmin):
    inlines = [OrderItemsTabularInline]
    list_display = ('order_id', 'phone', 'user_id', 'check', 'data')
    list_filter = ('phone', 'check', 'data',)
admin.site.register(Order, OrderAdmin)

class ProductAdmin(admin.ModelAdmin):
    list_display = ('product_name', 'price')
    search_fields = ('product_name',)
    list_filter = ('name',)
admin.site.register(Product, ProductAdmin)

class OrderCacheAdmin(admin.ModelAdmin):
    list_display = ('name', 'count', 'total_price', 'user_id')
    list_filter = ('user_id',)
```

```
admin.site.register(OrderCache, OrderCacheAdmin)

class OrderAdmin(admin.ModelAdmin):
     list_display = ('phone', 'name', 'count', 'data', 'check')
     list_filter = ('phone', 'check', )
admin.site.register(Order_Item, OrderAdmin)
```

Файл - **urls.py**

```
from .views import Menu_Button, Get_Product, Get_Order_Cache, Send_Order_Cache, Del_Order_Cache, Get_Product_in_Order_Cache, Send_Order
from django.urls import path, include
urlpatterns = [
     path('button', Menu_Button.as_view(), name='button'),
     path('products', Get_Product.as_view(), name='products'),
     path('send_order_cache', Send_Order_Cache.as_view(), name='send_order_cache'),
     path('get_order_cache', Get_Order_Cache.as_view(), name='get_order_cache'),
     path('del_order_cache', Del_Order_Cache.as_view(), name='del_order_cache'),
     path('get_product_in_order_cache', Get_Product_in_Order_Cache.as_view(), name='get_product_in_order_cache'),
     path('send_order', Send_Order.as_view(), name='send_order')
]
```

ДОДАТОК Б

```
from django.db import models

class Button(models.Model):
     button = models.CharField(max_length=300)

     def __str__(self):
     return self.button

class Product(models.Model):
     name = models.ForeignKey(Button, on_delete=models.CASCADE)
     product_name = models.CharField('название продукта',
max_length=1000)
     description = models.TextField('краткое описание здесь',
max_length=1000)
     price = models.CharField('стоимость продукта в грн',
max_length=30) #price = models.FloatField(max_length=100)
     weight = models.CharField('вес продукта', max_length=30)
     image = models.TextField('ссылка на картинку', max_length=300)

     def __str__(self):
     return self.product_name

class OrderCache(models.Model):
     name = models.CharField(max_length=300)
     price = models.CharField(max_length=30) #price =
models.FloatField(max_length=100)
     count = models.CharField(max_length=30) #count =
models.PositiveIntegerField(max_length=100)
     total_price = models.CharField(max_length=10000) #total_price =
models.PositiveIntegerField(max_length=100, editable=False)
     user_id = models.CharField(max_length=100)

     def __str__(self):
     return self.name

     #def total_price(self):
     #    self.total_price = self.price * self.count

class Order_Item(models.Model):
     order_id = models.ForeignKey('Order', on_delete=models.CASCADE)
     #order_id = models.CharField(max_length=100)
     phone = models.CharField(max_length=100)
     name = models.CharField(max_length=100)
```

```
     price = models.PositiveIntegerField() #price =
models.FloatField(max_length=100)
     count = models.PositiveIntegerField()
     total_price = models.PositiveIntegerField()
     check = models.BooleanField(default=False)
     data = models.DateField(auto_now=True)

     def total_price(self):
     self.total_price = self.price * self.count
     return self.total_price

     def __str__(self):
     return self.phone

class Order(models.Model):
     order_id = models.AutoField(primary_key=True)
     user_id = models.CharField(max_length=100)
     phone = models.CharField(max_length=100)
     #total_price = models.PositiveIntegerField()
     check = models.BooleanField(default=False)
     data = models.DateField(auto_now=True)
     order_items = models.ManyToOneRel(Order_Item, to=Order_Item,
field_name='order_id')

     def __str__(self):
     return self.phone

class Customer(models.Model):
     phone = models.CharField(max_length=100, null=True)

     def __str__(self):
     return self.phone
```

ДОДАТОК В