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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Воробйова Данила Дмитровича*
(ПІБ)

академічної групи *122-19ск-2*
(шифр)

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

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

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

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19ск-2* *Воробйова Данила Дмитровича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram*

затверджена наказом ректора НТУ «ДП» від *12 травня 2022* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Воробйов Д.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 67 сторінок, 34 рисунки, 3 додатку, 24 джерела, 1 таблиця.

Об'єкт розробки: Telegram-bot для замовлення їжі у закладі.

Мета проекту: створити бота зі зрозумілим легким інтерфейсом, який надасть можливості легко замовити їжу з ресторану, отримати меню чи контакти ресторану та використовувати інтерактивні можливості.

В графі вступу розглянуто аналіз та сучасний стан проблеми, сформульовано мету проекту та сферу застосувань. Описано актуальні та пов’язані теми, скореговано завдання.

В першому розділі наведено аналіз предметної області, визначена актуальність завдання і призначення розробки. Розроблено постановку завдання, задано вимоги програмної реалізації, вимоги технології і програмних засобів.

В другому розділі виконано аналіз існуючих рішень, мову та фреймворк для розробки. Виконано проектування та розробка програми. Наведений опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

У економічному розділі визначено трудомісткість розробленої системи, проведено підрахунок вартості роботи створення застосунку і розраховано час на виконання.

Актуальність бота визначена потребами користувачів до планування свого графіку і комфорту. Окрім цього, потребою в зручному і зрозумілому інтерфейсі.

Список ключових слів: TELEGRAM-BOT, AIOGRAM, SQLITE3, ДОКУМЕНТАЦІЯ TELEGRAM, ДОКУМЕНТАЦІЯ AIOGRAM, ІНТЕРФЕЙС, API, БІБЛІОТЕКА, HTTPS.

ABSTRACT

Explanatory note: 67 pages, 34 figures, 3 appendices, 24 sources, 1 table.

Object of development: Telegram-bot for ordering food in the institution.

The purpose of the project: to create a bot with a clear easy interface that will allow you to easily order food from a restaurant, get menus or restaurant contacts and use interactive features.

The introduction column considers the analysis and current state of the problem, formulates the purpose of the project and the scope. Current and related topics are described, tasks are adjusted.

In the first section the analysis of the subject area is given, the urgency of the task and purpose of development is defined. The statement of the task is developed, requirements of software realization, requirements of technology and software are set.

The second section analyzes the existing solutions, language and framework for development. The design and development of the program is performed. The description of algorithm and structure of functioning of system is given, the input and output data are defined, characteristics of structure of parameters of technical means are resulted, the call and loading of application is described, work of the program is described.

In the economic section, the complexity of the developed system is determined, the cost of creating the application is calculated and the execution time is calculated.

The relevance of the bot is determined by the needs of users to plan their schedule and comfort. In addition, the need for a user-friendly and intuitive interface.

Keyword list: TELEGRAM-BOT, AIOGRAM, SQLITE3, TELEGRAM DOCUMENTATION, AIOGRAM DOCUMENTATION, INTERFACE, API, LIBRARY, HTTPS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

МП – мова програмування.

ОКВ – одноразові капітальні витрати.

ОС – операційна система.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

UI – user interface.

HTML - HyperText Markup Language.

HTTPS – HyperText Transfer Protocol Secure.

API – Application Programming Interface.

SCIM – Smart Common Input Method.

OS – Operating system.

COM – Component Object Model.

DRAM – Dynamic Random Acces Memory.

ВСТУП

Інтернет став невід'ємною частиною в суспільстві людей. Більшість речей потребують для свого використання цієї мережі, а люди – тим паче. З цього витікає, що пошук інформації, навчання, робота або ще щось дедалі більше йде у бік Інтернету, а люди - залежні від нього. Через ці умови все більш актуальним повстає питання розробки програм, роботів та просто ботів які б допомогли людям прискорити час виконання роботи, зробити її ліпшою і комфортнішою, наприклад - telegram-bot.

То що ж з себе уявляє Інтернет? Це є супер-глобальна міжнародна мережа. Вона пов'язує у собі безліч пристроїв для передачі тих чи інших даних, інакше кажучи - інформації. Інформація в свою чергу являє собою ті чи інші свідчення про процеси, об'єкти, явища тощо. Робота в мережі заснована на використанні спеціальних протоколів. Для пов'язання пристроїв використовують оптичні, бездротові та дротові технології.

У склад самого Інтернету входить багато інших глобальних мереж, локальних, підмереж тощо. Він являє з себе плато на якому розміщені рекламні або інші послуги, товарні продукти, ті чи інші вакансії. За допомогою Інтернету також люди можуть вільно спілкуватися один з одним за допомогою електронних пошт, різних веб-сторінок та меседжерів з комфортом. Комфортними для людини можна назвати ті умови, коли вона задоволена або знаходиться в своєму затишку, задовільняє усі свої потреби для самовдоволення, має доступ до усіх необхідних ресурсів та відчуває безпеку.

Таким чином можна зробити висновок, що Інтернет, а також використання програм та ботів на його основі можуть значно полегшити життя людини, надати їй більше часу на власні розваги та забезпечити комфорт.

У вирішенні проблем з питання часу людині також допоможе таке поняття як керування часом або «time management». То є сполучення різноматнітних технік з організації часу аби знайти і витратити оптимальний проміжок часу на

заплановані події, роботу, проекти тощо. Для слідкування за часом та виконаннями роботи використовують такі зручності як календар або щоденник. При виконнанні більш складних завдань їх я правило поділять на більш менші та розподіляють кожну окрему частину на одну людину для збільшення ефективності та зменшення використаного часу. Більшість технік керування часом полягають у плануванні, пріоритизації та структуруванні, а за сам винахід маємо дякувати Клаусу Меллеру, що розробив особливий комплексний щоденник.

Задачею кваліфікаційної роботи постає створення telegram-bot для ресторану, який дає можливість користувачу швидко та зручно замовити їжу за власним вибором та смаком. Для виконання завдання було використано мову програмування Python з використанням асинхронного фреймворку aiogram. Мова є високорівневою та має строгу динамічну типізацію. З поміж усього вона має автоматичние управління пам'яттю та здатна забезпечити переносиміть програми. Дана мова дуже легка у розумінні та спроможна підвищити ефективність роботи. Для розробки проекту використано асинхронний фреймворк aiogram.

Таким чином була обрана тема дипломної роботи: «Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram ».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості с предметної галузі

Для виконання роботи людині потрібен час, для навчання та вдосконалення навичок або розваг – також час. Виходячи з цього можна зробити висновок, що вирішення питання часу та його менеджменту є одним з головних питань для людей, адже вони від цього залежні.

Потрібно планувати свій власний час таким чином, щоб зберігаючи якість праці досягнути цілей вчасно, інашке нічого не досягнеш. Для цього необхідно чітко визначити виділений на роботу час, поділивши усю працю на малі підпункти. Окрім цього також потрібно враховувати можливі сторонні чинники та випадкові події, вони можуть вплинути на продовженість виконання. Перелічені зауваження зменшать навантаження на виділений і конкретні проміжки часу. Буде добре сконцентруватися на одній меті, виділити певний час, не жертвуючи якістю роботи.

В навколишньому світі є дуже багато речей які могли б затримати виконання того чи іншого процесу. Втім ми живемо в епоху цифрових технологій і це один з багатьох плюсів. Інтернет дає людині можливість спланувати для себе час зручним чином, завчасно про все повідомляти людей тощо. Як приклад можна навести зменшення витраченого часу на ті чи інші дії коли ми користуємося онлайн послугами. Так нам не потрібно самим зайвий раз витрачати час на те аби щось зробити чи кудись поїхати.

Одним із таких прикладів є онлайн замовлення їжі. Це допоможе не витрачати зайвий раз наш найцінніший ресурс. Через це у багатьох сервісах є свої власні спеціальні додатки, а також боти.

Бот являє собою спеціальну програму яка імітує дії користувача, виконуючи ту чи іншу роботу за заданим алгоритмом у заданий час тощо.

Взагалом такі програми дуже часто використовують як раз в Інтернеті або іграх. Вони виконують певну роботу та дії замість людини, в основному це є нескладні алгоритми чи лінійна робота, а також обчислення.

Для врегулювання таких програм та підвищення безпеки на більшості веб-серверів використовується стандарт(robots.txt) вийнятків для ботів який цим займається. Він допомогає боту зрозуміти які частини вебсайту він може обробити. Пошукові системи використовують ботів для визначення категорій сайтів, проте не всі вони підпорядковуються стандартам.

Існують шкідливі програми спам-боти. Такі можуть надавати велику кількість непотрібної інформації, засмітити систему або отримувати заборонену інформацію. Запропонував ці стандарти Мартін Костер 1994 року коли працюва у Nexor. Того часу ідея прийшла до голови у результаті написання некоректного вебоглядачу. Стандарт швидко отримав визнання та використовується у багатьох системах пошуку, наприклад AltaVista і Lycos. Використовувати це можна для захисту інформацій, коректного відображення сайту чи каталогу, правильного спрацювання того чи іншого скрипту, програми тощо.

Важливо врахувати – стандарт діє окремо для різних протоколів, портів чи субдоменів. За таким принципом працюють системи «Google», «Baidu», «Yandex» чи «Yahoo!».

Для заощадження ресурсів та збільшення ефективності спочатку програма виконує скан сайту, після чого йде його аналіз. Проте у будь-якому випадку це лише умовність, програма-шкідник так чи інашке буде слідкувати лише своїм власним інструкціям, коду чи навіть намагатися отримати власну користь з цього стандарту, адже він ні до чого не забов’язує і розрахований з більшого боку на доброчесність.

Актуальне також використання розширень для стандарту. Для прикладу візьмемо crawl-delay – очікування перед повторним завантаженням сторінки. Втім частиною стандарту воно не є, через це кожна система або бот може розуміти його для себе по-різному. Використовувати це можна для задання

швидкості загрузки веб-сторінок для зменшення навантаження, прискорення тощо.

Якщо потрібно відображати лише конкретні директорії зі зберіганням видимості та індексування файлів html, використовується вказівка «disallow». Для сайтів з дзеркалами використовується директива «Host» для того аби окреслити конкретний домен.

Повертаючись до ботів та способів їй застосування, необхідно вказати, що дуже популярними вони є і у іграх. Оскільки боти імітують дії людини, їх використовують у тих сферах де потрібно виконувати довгу одноманітну роботу або повторювати одні й ті ж самі кроки щоразу, чи у іграх де потрібна хороша реакція.

У іграх зі змаганнями таких можна використати для тренінгу або ж для заміщення купи витрачаємого часу на виконання різного роду завдань. Не стане вийнятком також використання ботів для проведення розіграшів тощо.

Звичайно кожні програми та боти мають свій власний інтерфейс, у кожного свій. Сам по собі інтерфейс є сумішшю різних характеристик, правил, методик, засобів тощо. Не буде помилкою сказати, що це поняття використовується майже будь-де а не тільки у комп’ютерній чи технічній сфері. Термін використовується у медицині, біології, природі, повсякденному життю людини тощо. Слід зазначити, що також мається на увазі взаємодія між елементами інтерфейсу, наприклад робота в мережі Інтернет та її протоколи.

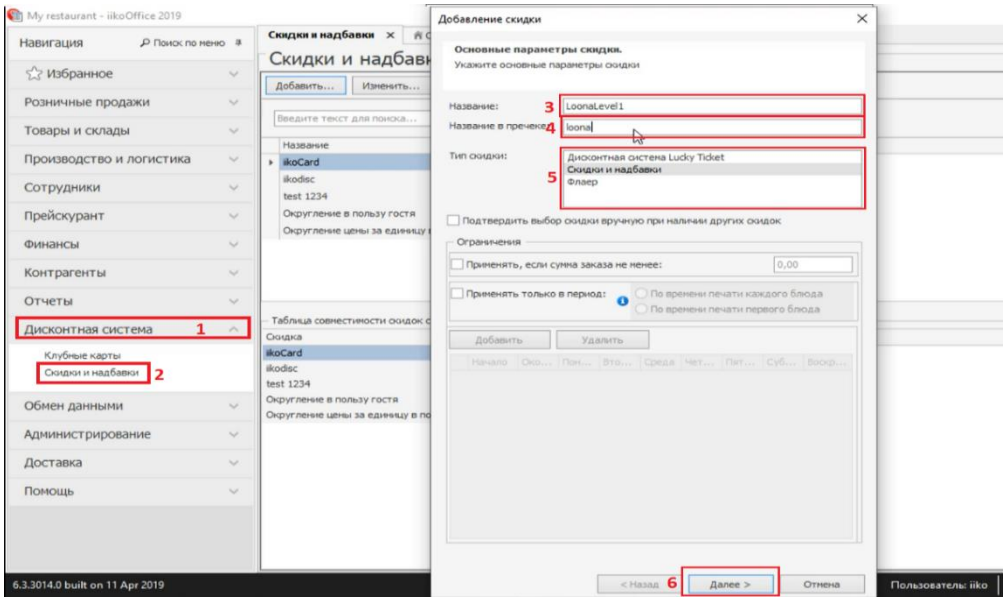

Рис. 1.1. Приклад інтерфейсу програми iiko.

Якщо ж казати про комп’ютерну та технічні сфери інтерфейс можна описати як сполучення деяких правил і засобів для забезпечення взаємодії пристроїв з людиною, програми чи декількох програм з іншими програмами, одного пристрою з іншим, підпорядкування коду в програмі тощо.

Для більшої зрозумілості можна навести приклади інтерфейсів. У кожному з наших домів є розетки у які втикають вили, це і є приклад інтерфейсу. Окрім цього у побуті у нас є дуже багато електроприладів на кшталт годинників, телевізорів, комп’ютерів, радіо тощо. Елементи цих приладів такі як кнопки, ручки, перемикачі, їх дисплей тощо і є прикладом інтерфейсу.

Для користування комп’ютером ми використовуємо клавіатури та миші. У такому випадку маємо приклад інтерфейсу людини і машини, споміж іншого усередині них є також власний інтерфейс, який дозволяє взаємодіяти з самим комп’ютером. В цілому усе це залежить від ситуацій, контексту, та елементів інтерфейсів про які йде мова у конкретному випадку. В деяких випадках користувачу важливо надати лише елементи керування приладом тощо. Йому зовсім не обов’язково знати як він працює та налаштований. Власне це те до чого і прямують досвідчені розробники – полегшити використання для звичайної людини.

У сучасних інформаційних системах інтерфейси стають основою майже усього. В більшості випадків використовуються стандартні інтерфейси які використовуються довгі роки та зручні для використання звичайній людині. Лише інколи це змінюється, коли йде мова про конкретну компанію чи сферу діяльності або можливість корекції інтерфейсу надається самому користувачу. У таких випадках є можливість працювати з комфортом над потрібним проектом в цілому нічого не змінюючи, основну роботу за людину робить сама програма, а користувач лише корегує усе на власний смак.

В нагоді зручний інтерфейс може стати на підприємстві чи при симуляції бізнесу. Для легкості кервуання надається максимальна кілкість інформаціє майже про усі події, чинники тощо. При цьому є багато функцій регулювання інформації або взаємодії з нею. Це дозволяє не витрачати зайвий час на непотрібні речі та виключити зайву або непотрібну у даний момент інформацію.

Якщо ж ми кажемо про безпосередньо людину та її біологічні особливості, то тут також є приклади інтерфейсів, наприклад деякі мозкові структури. Частина з них регулює нашу поведінку при фізіологічних та психічних процесах.

Для взаємодії людини з комп'ютером використовують спеціальний термін UI - user interface. Інтерфейс, котрий сприяє передачі інформації між комп'ютерною системою, програмами та звичайно що людиною. Він є важливою складовою у сфері дизайну, оскільки постійно проходить взаємодія інтерфейсу програми з людиною, використовується багато елементів кожен з яких має значення.

За допомогою висококласного інтерфейсу людині не потрібно зайвий раз витрачати сили, вона лише відтворює потрібні їй дії, а комп’ютер виконує усе необхідне та допомогає, надає рекомендації. Дуже важливими у таких програмах є інтерактиви, інструменти що дозволяють виконувати усе необхідне, засоби керування і контролю. Для створення висококласного інтерфейсу потрібно враховувати такі важливі чинники як психологія, адже людина не буде

налаштована на працю і буде витрачати більше енергії ніж потрібно у незручному становищі. Окрім цього враховуються і ергономічні моменти, які зберігають і фізичні сили також, покращуючи моральний стан працівника.
При створенні інтерфейсу дуже важливо аби він був нескладним для користувача, впорядкованим, усі необхідні та найважливіші елементи мають бути у зручному місці. Більше того, повинна зберігатись ефективність такого інтерфейсу та нейтральність для користувача чи можливість його корекції. Це потрібно для того щоб кожен користувач міг комфортно себе відчувати, інтерфейс не має відторгати тебе від нього. В ідеалі треба отримати використання найменших зусиль з максимальною ефективністю для роботи.

В промисловоті зазвичай використовуються інші типи інтерфейсів, але так чи інакше це взаємодія людини з машиною. Від так можна зробити висновок, що інтерфейс є дуже важливою складовою не тільки при роботі, але й у всьому житті людини.

При розробці ботів для соціальних мереж та меседжерів вони зазвичай отримують мініатюрний, легкий та дуже простий для розуміння інтерфейс, надаючи таким чином користувачу лише необхідну йому інформацію та можливості взаємодії. Більшість ботів вбудовані в чат платформи, унікальний та швидкий простір для користувачів, де вони можуть обінюватись тим чи іншим типом інформації між собою або надавати її за допомогою бота, наприклад для відправки даних до компанії.

Одним із найпоширеніших меседжерів на даний момент є Telegram. Його поважають за захищеність приватної інформації, надійність та дуже легкий і нейтральний інтерфейс.

Як і в кожному меседжері, у Telegram розповсюджене явище ботів.

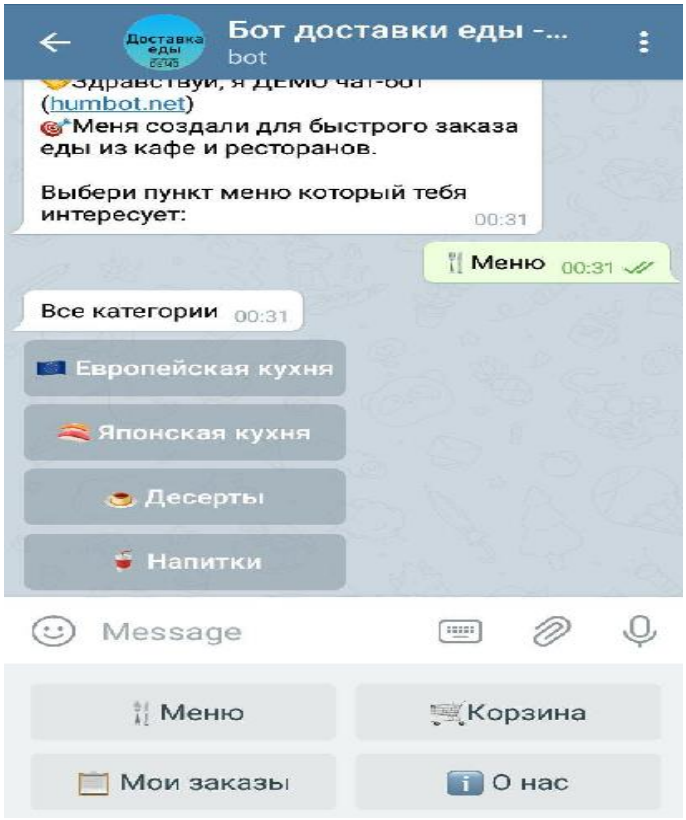

Рис.1.2. Приклад чат-боту Telegram.

Не дивно, що станом на зараз такі боти є дуже релевантними. В першу чергу це пов’язано із самою популярністю платформи. По-друге усе це є можливим завдяки якості. Існують навіть спеціальні вбудовані боти, що дозволяють прямо на платформі шукати той чи інший контент в Інтернеті або грати в ігри. Зокрема є основний бот – BotFather, завдяки йому можна створювати інших ботів та налаштовувати їх так як потрібно.

Та що ж конкретно уявляють з себе боти Telegram? По суті вони існують як окремі облікові записи, що слугують інструментом автоматизації прийняття, обробки та відправлення інформації, а також повідомлень. Взаємодія з ботом відбувається у групових чатах, а також в особистих переписках завдяки повідомленням. За допомогою https-запитів до спеціального API Telegram реалізовано логіку ботів.

API дешифрується як Application Programming Interface. У своїй суті він об'єднує набір різних функцій та процедур, а також скопичення класів, констант чи структур і їх опису. Тобто він пояснює взаємодію програми з програмою. За його допомогою можна описати програмний каркас, а тобто - фреймворк, будь-який протокол такий як SCIM чи стандарт виклику функцій ОС.

Фреймворк в свою чергу є свого рода програмою, що дозволяє об’єднувати в один проект різні компоненти інших проектів. Це може бути програмний код написаний на певній мові програмування, інша програма, різні системи та їх синтаксиси, мова інтерфейсу тощо.

Більшість ботів мають свою власну базу даних(БД). При написанні бота для Telegram часто використовують SQLite3. По суті це легка вбудована система управління базами даних з динамічною типізацією. SQLite не є окремим виконуємим процесом з яким має взаємодіяти програма, він стає її частиною. Протокол обміну використовує виклики функцій з SQLite бібліотеки. Таким чином зменшуються час відгуку, витрати, а сама програма спрощується.

Уся основна інформація накопичується та зберігається на поточному пристрої, що виконує роботу. Записи до БД виконуються при вимогах того, що в даний момент часу нічого не виконується, а от читати дані можуть одразу декілька процесів.

Для наочного розуміння можна навести приклади інструментів чи утиліт. Є спеціально орієнтовані боти, які можуть надати вам можливість перекласти текст, знайти і використовувати певну інформацію з Інтернету, підказати погоду тощо. Бота можна інтегрувати до інших сервісів. За допомогою цього він може отримувати керування чимось та регулювати процеси. У бота буде можливість надсилати вам повідомлення у разі, якщо відбулася та чи інша подія, наприклад повідомлення від YouTube або Google, чи наприклад стримінгових платформ, що сповіщають про старт трансляції.

Як вже було сказано раніше, бот здатний грати з вами у класичні або прості игрі, проводити конкурси, чи бути наприклад вашим співрозмовником, виходячи з особових інтересів.

При створенні свого власного бота слід починати з того, аби написати основному боту команду /newbot. Після цього він запропонує вам дати йому ім’я та можливість налаштувати основні його функції. У результаті ви отримаєте свій власний ключ доступу бота, котрий також зветься token. Далі вам потрібно буде звернутися до документації Telegram стосовно ботів – API ботів. Це дозволить вести подальшу розробку бота та його налаштування.

До унікальних можливостей телеграм-ботів також входять додаткові інтерфейси для дефолтних команд. Поміж іншим є клавіатури з кастомізацією, приватні режими для груп або зовнішні зв’язки. Відповідно до усього цього існує згадана документація Telegram, що дозволяє наводити корекцію потрібних моментів.

Для гри у ігри існують спеціальні групи та приватні чати. Там можна знайти таблицю рекордів у цих іграх, про зміну якої вас може повідомляти бот. Виконується усе це за допомогою спецального додатку html5.

У більшості випадків кожен бот має власну клавіатуру. При натисканні на неї виконуються закладені дії та команди, після чого йде запит до серверу. Це дозволяє налагодити комунікацію з користувачами та значно спрощує її. Для більшої різноманітності станом на зараз підтримується відображення на клавіатурі не тільки тексту, але й емодзі.

Рис.1.3. Приклад клавіатури бота.

Для передачі під час запуску додаткового параметру використовують механізм зовнішнього зв'язвуання, так це працює з токеном авторизації користувача при з'єднанні зовнішнього серверу.

Для початку розмови з ботом у них є власний унікальний лінк. До нього можна додати параметри start та startgroup при довжині до 64 символів. Таким чином при переході за посиланням бота у полі вводу меседжу відобразиться кнопка start.

Унікальні можливості ботів використовують для надання або отримання різної інформації з новин чи будь-чого ще, а не тільки з повідомлень. З використанням ботів люди починають турбуються про свою безпеку і приватність. Кожна переписка чи інформація має бути надійно захищена. Для цього існує спеціальний приватний режим (або режим приватності). За замовчуванням він завжди увімкнений. Він працює таким чином, що при його робочому режимі боти отримують тільки повідомлення з задовільненими вимогами. Повідомлення має містити командний символ «/» або символ згадування «@». Також до цих умов включені відповіді боту та службові повідомлення, наприклад зміна назви, фотографії профілю або що-небудь ще.

Таким чином люди перестають піклуватися про свою приватність та безпеку, а розробнику не потрібно зайвий раз витрачати час та сили на обробку купи повідомлень чатів груп.

1.2. Призначення розробки та область застосування

Темою бакалаврської дипломної роботи виступає: «Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram». В якості мови програмування було обрано Python та фреймворк aiogram. Головною метою роботи є розробка телеграм-бота, що по суті являє з себе інструмент за допомогою якого користувачі, а тобто клієнти ресторану зможуть дивитись розклад, контакти, перевірити меню, спілкуватись з ботом тощо.

Головні критерії розробки:

1) Комфорт у використанні.
2) Легкість для розуміння користувача.
3) Надання користувачу потрібних йому функцій.

Бот призначений для:

1) Взаэмодії клієнта та ресторану.
2) Перевірити меню.
3) Можилвості продивитсь графік роботи.

Програма позиціонується як бот для меседжеру Telegram, що дає можливість використовувати клавіатуру та кнопки бота для взаємодії з користувачем, надання відповідних послуг та інформації.

1.3. Підстава для розробки

Згідно програми освіти, відповідно навчального плану і графіків навчального процесу, під кінець навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Підстави виконання та розробки кваліфікаційної роботи:

- програма освіти для спеціальності 122 «Комп’ютерні науки»;

- Графік і план навчального процесу;
- Наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 12.05.2022 р;
- завдання на дипломний проект на тему «Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram».

1.4. Постановка завдання

Метою дипломного проекта є розробка Telegram-bot для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram, основою якого є мова Python та фреймворк aiogram. Бот призначено для комфортної взаємодій з кліентом та надання йому необхідних послуг.

Бот має реалізувати такі речі:

- Взаємодія з клієнтами.
- Спілкуватись з користувачами.
- Надавати можливість використовувати його клавіатуру і кнопки.
- Мати власне меню, а також телефон і розклад закладу, що можна передивитись.
- Комфорт і зрозумілість у використанні.

Для розробки бота ми маємо:

1) Зробити аналіз існуючих ботів даного плану.
2) Дослідити документацію про ботів від Telegram.
3) Створити комфортну та зрозумілу клавіатуру з кнопками.
4) Навчити бота адмініструвати групу закладу.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Створений бот аби досягнути мети повинен мати:

1) Швидка відповідь на дії клієнта.
2) Надавати впершу чергу необхідну інформацію.
3) Реакція на ненормативну лексику.
4) Зрозумілі та коректні кнопки і клавіатура.

Для досягнння задачі потрібно:

1) Отримати токен нашого боту від BotFather.
2) Уважно дотримуватись документації Telegram.
3) Налагодити вихід нашого боту в онлайн.
4) Написати код для реалізацій поставлених цілей.

1.5.2. Вимоги до інформаційної безпеки

Згідно документації про ботів від Telegram – користувачі захищені від зливу інформації спецрежимом «приватності», саму ж приватну інформацію налаштувати можна безпосередньо у меню Telegram. При увімкненому режимі бот отримає тільки меседжі які починалися зі знаку «/». Також у меседжах, де бота згадали за допомогою «@», службових повідомленнях та відповідей до самого бота. Це надає змогу захистити приватну власність людей та не витрачати купу часу аби редагувати купу повідомлень власноруч.

Таким чином головна вимога інфомаційної безпеки надходить лише до самих користувачів, які надають про себе ту чи іншу інформацію у Telegram, боту тощо.

Самий же токен для боту надійно захищений у Telegram.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для забезпечення функціонування боту потрібні персональний комп’ютер або ноутбук та наступні вимоги:

1) доступ до Інтернет та безпосередньо Telegram.

2) Прилади вводу та виводу(комп’ютерна миш, клавіатура, монітор)

3) Місце на сховищі інформації(краще за все аби індикатор заповненості був синій або диск мав 20 гб вільного місця).

4) Для роботи з програмою – центральний процесор(вищій від шостого покоління Intel).

5) Для комфортної роботи за пам’яттю - 8 гб DRAM.

Наведені вимоги дозволять розробити та вести підтримку бота без значних ускладнень.

1.5.4. Вимоги інформаційної та програмної сумісності

Більшість мов програмування потребують конкретних OS та більш нових версій програмних продуктів, але Python в свою чергу є кросплатформовим і наприклад Windows XP підтримується версією Python 3.4, а Windows Vista – версією 3.8 та вище.

Що стосується інформаційних вимог для бота, слід зазначити важливість оформлення кнопок боту і його наповнення на комфортній для закладу мові та в першу чергу - для користувачів.

Таким чином аби уникнути зайвих проблем потрібна версія OS є не нижчою за Windows 7, а мова наповнення бота повинна відповідати мові користувачів. Окрім цього слід подбати про версію Python вищу за 3.7.

Даний бот має бути виконаний на мові Python з використанням фреймворку aiogram. Виконання та налаштування боту проведено за допомогою команд з документації Telegram і Python.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

В процесі виконання дипломного проекту створено Telegram-bot на основі мови програмування Python з використанням фреймворку aiogram. За допомогою бота можна замовити їжу з ресторану, а окрім цього він надає комфортні для цього можливості. Усе це є можливим завдяки документації Telegram та програмному коду вкладеному до бота.

Розроблений бот виконує:

1) Вбудовані у програмний код команди, які надають можливість переглянути меню бота, графік роботи закладу, контакти та його розташування.

2) Дає можливість використання клавіатури бота для зпрощення набору команд.

3) Спеціальну окрему клавіатуру для адміністратора групи, який може додавати та змінювати елементи меню ресторана.

4) Власну БД з використанням SQLite3 для збереження інформації щодо меню закладу.

5) Inline кнопку для до якої можна надати набір функції або закріпити посилання на ресурси ресторану.

6) Налаштовано parsing боту, це дає змогу клієнту котрий не знає команд отримати підказку.

7) Створено список команд.

Щоб мати всі функції використання що були згадані, бот має :

1) Бути розроблений та налаштований з допомогою документації Telegram.

2) Отримати власний token.

3) Бот має виходити в онлайн.

4) Повинен бути написаний відповідний програмний код, який дозволить реалізувати можливості.

5) Повинна бути створена і налаштована БД бота.

6) Аби бот не засмітчував бесіду та був доцільним слід написати команду, що вимикає отримання «оновлень».

2.2. Опис застосованих математичних методів

Оскільки виконуємий проект спрямований на написання Telegram-bot для замовлення їжі було використано лише програмні засоби та засоби самого Telegram. В ході виконання були застосовані асинхронні функції, що являють з себе тільки програмну частину проекту. Окрім як присвоєння або передачі тих чи інших значень, математичні функції використані не були. Вийнятком може стати лише lambda функція, що закладена у мову Python. Це безіменная функція, яка використовується для роботи з різними типами даних. Наразі це єдине у проекті, що пов’язано з використанням будь-яких математичних методів. Більше ніяких методів використано не було.

2.3. Опис використаних технологій та мов програмування

Telegram-bot розроблений з допомогою мови програмування Python при використанні фреймворку aiogram. Меню для перегляду продукції базується на використанні SQLite3, окрім цього, оскільки частина налаштувань бота та технології його реалізації залежить від ресурсів самого Telegram, їх використано також.

Мова Python

Мова програмування високого рівня загального призначення, якій властива динамічна строга типізація. Це дозволяє програмісту швидко та комфортно вивчати мову або виконувати з її допомогою свою роботу, проекти

тощо. Динамічна типазція дозволяє легко оперувати типом змінної у коді, оскільки він присвоюється лише у момент прив'язування значення. Таким чином змінна у коді завжди буде мати потрібний для розробника тип, що значно спрошує концепцію використання. Для ще більшого полегшення роботи розробника у мові передбачено автоматичне управління пам'яттю, це дозволяє не піклуватися про пам'ять, не витрачати на неї час та автоматично долати проблеми засмітнення пам'яті. Python є об'єктно-орієнтованим, об'єктами являється усе. Кодові блоки при використанні виділяються відступами з прогалинами. Мова високого рівня, на практиці є купа спрощень, а отже і синтаксис мови є простий, має масштабованість, продуману модульність та послідовність. На даній мові часто пишуть скрипти. Це пов'язано з тим що вона є інтерпретованою, усі алгоритми програми виконуються один за одним та не потребують попередньої компіляції. У якості недоліків мови зазвичай згадують використання збільшеної кількості пам'яті або малу швидкість виконання. Втім зустріти на практиці ці недоліки майже ніколи не доведеться.

Python здатен підтримувати багато видів програмувань. Структуроване, функціональне програмування, об'єктно-орієнтоване, метапрограмування тощо. За допомогою декораторів також частково можливе аспектно-орієнтирований варіант. Додаткові фреймворки забезпечують більшу підтримку. Інтроспекція та багатопоточні обчислення дозволяють значно скоротити час виконання роботи. Підтримується поділення на модулі програм, а також високорівневі структури даних.

Інтерпретатор CPython – стандарт та еталон мови, підтримує багато з платформ, що активно використовуються. Він робить компіляцію початкового тексту у високорівневому байт-коді, що виконує стекова віртуальна машина. Окрім цього Python має багато своїх версій для інших МП. Серед них PyPy, Jython чи IronPython. Якщо розбирати PyPy, то побачимо що його написано на RPython, підмножини мови Python. Ціллю було розробити щось схоже на CPython та підвищити швидкість виконання програми завдяки JIT-компіляції.

В стандартній бібліотеці пайтона купа корисних функій. Наприклад засоби написання мережевих аплікацій, робота з текстом тощо. Написання веб-додатків, розробка ігор, математичне моделювання та інші проекти також придатні до реалізації. Це можливо завдяки купі інших додатніх бібліотек та їх інтеграції. Також можливо інтегрувати пайтон в інші мови програмування, а точніше в проекти.

Сам по собі пайтон портовано ледь не на усі можливі платформи. Звичайно він портований до Windows, UNIX, Android, MAC OS, IPhone, IPad тощо. Старі версії платформ в більш нових версіях Python підтримуються все менше. Починаючи з версії Python 3.9 закінчилася підтримка Windows7. Python здатний підтримувати технології класичні до відповідної платформи.

До стандартних типів у Python відноситься ціле число з довільною точністю, тип Буля, комплексні числа, числа з плавоючою комою. З контейнерних типів є списки, словарі, строки, множини та кортежі. Усе це, а також методи, класи, функції і модулі є об’єктами. Для того щоб створити новий тип використовують class або можна визначити його в модулі розширення. Класи підтримують спадкування, воно можливе від більшості вбудованих розширень та типів.

Python має бібліотеки котрі надають на діферентних платформах інтерфейс для усіх викликів системи. Для Win32 існує підтримка усіх викликів Win32 API, COM. Існує специфікація UI для БД DB-API 2 та відповідні пакети доступу до СУБД накшталт MySQL чи Oracle. Для роботи з багатовимірними масивами розроблено бібліотеку NumPy з хорошою ефективністю. При створенні графічного інтерфейсу кросплатформової програми допоможе бібліотека Tkinter.

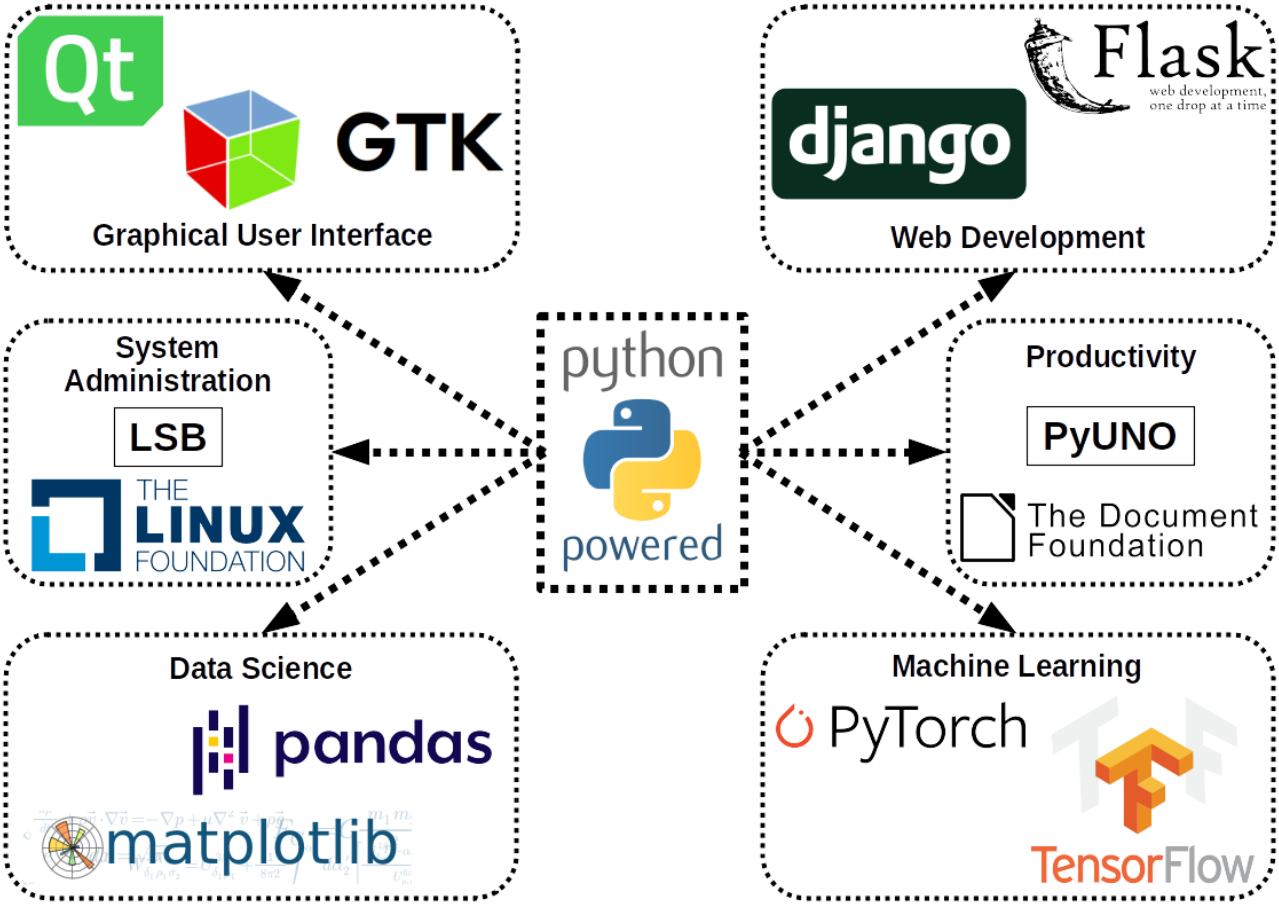

Рис. 2.1. Приклад використання мови Python.

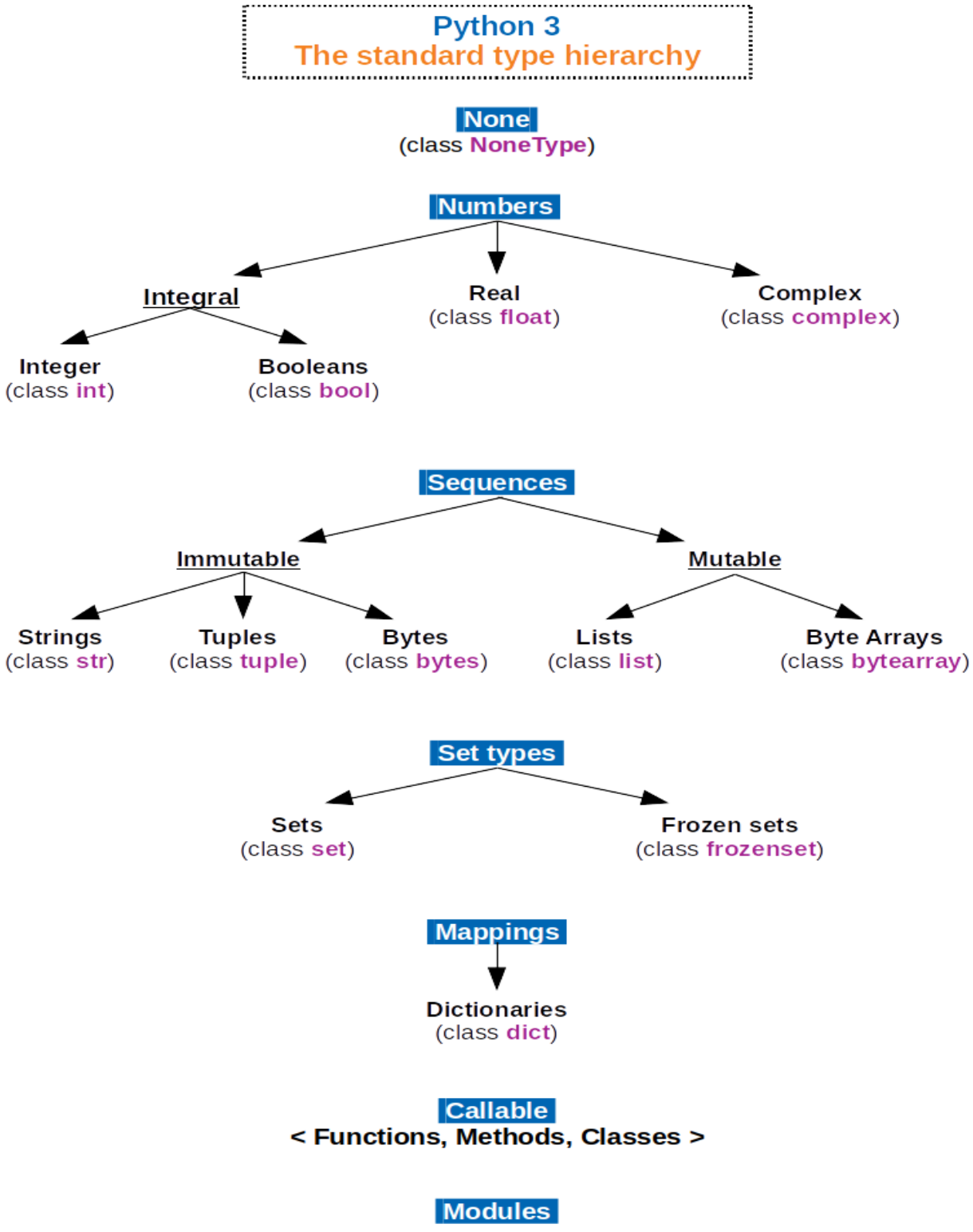

Рис. 2.2. Ієрархія стандартних типів Python.

Фреймворк aiogram та технології Telegram

Доволі легкий асинхронний міні фреймворк призначений спеціально для API телеграм-ботів. Основними вимогами при створенні aiogram було надання можливості швидкого написання боту та спрощення коду і його розуміння. Aiogram було написано на Python версії 3.7 за допомогою aiohttp та asyncio. Детальний опис бібліотеки розробники не залишили, втім як працює aiogram та що до нього входить можна дізнатись на сайті офіційної документації aiogram.

Для початку розробки бота за допомогою aiogram його необхідно спочатку імпортувати, з самої бібліотеки. По суті отриманий бот вже буде мати усі першопочаткові налаштування і розробнику потрібно лише створити його екземпляр. Далі залишається надати боту його токен, котрий отримується від BotFather – бота усіх ботів.

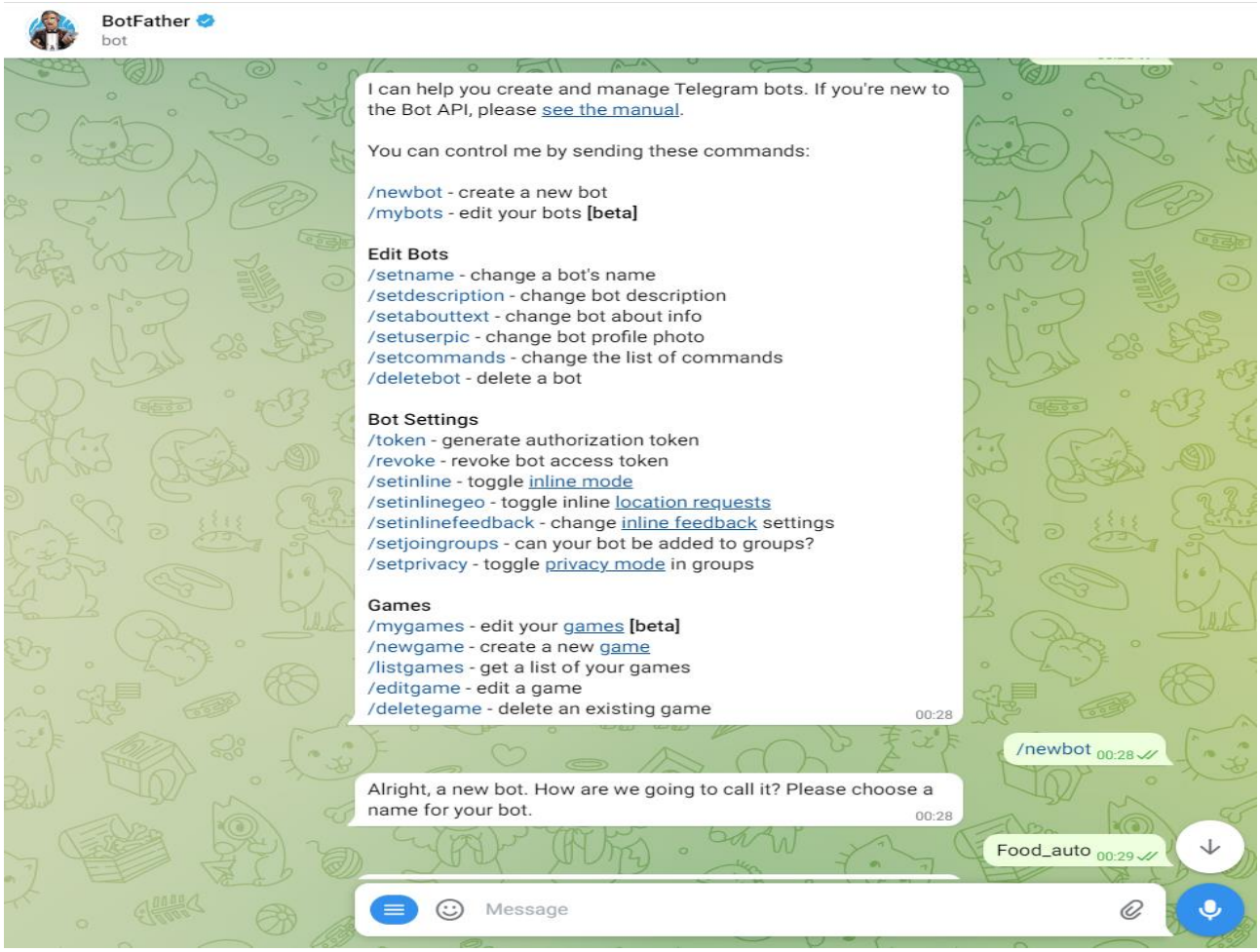

Рис 2.3. Приклад команд налаштування бота від Telegram.

Для продовження налаштування бота необхідно також імпортувати клас Dispatcher. Він відповідає за обробку надходячих «оновлень». Такими є пости у каналі, inline запроси, меседжі тощо. Також потрібно буде імпортувати клас types, це дозволить працювати з типами даних у Python. Спеціально для створення баз даних в Python одразу вшито SQLite3, тож для бота з БД її теж необхідно імпортувати. Надалі уся подальша розробка та імпорти цілком залежать від суті проекту та того, що до нього хоче внести розробник.

SQLite3

База даних SQLite є компактною БД(database engine), яку можна вбудувати в розроблюємий проект. Написано базу даних було на мові програмування C.

Використовують SQLite топові веб-браузери, мобільні телефони, ОС або інші системи які вбудовуються. Пов’язана з більшістю мов програмування. Синтаксис в цілому такий як у PostgreSQL, але перевірка типу не нав’язана. Вона працює на поточному комп’ютері та записує все у файл, тож підключатися до віддаленої БД не потрібно і це дуже зручно. Тобто бібліотека SQLite3 буде частиною програми і буде використовувати виклики функцій API з бібліотеки в якості протоколу обміну. За рахунок цього скорочується час відклику та кількість необхідних ресурсів. Під час запису у файл БД інформації він увесь блокується. Для ACID функцій створюється журнал-файл. В один і той же час з БД можуть зчитувати інформацію кілька процесів, при цьому під час роботи записа не є можливий.

Підтримується динамічний тип даних з варіантами значень text, integer, blob, real та спеціальний – null. Файли типів blob та text не мають верхньої границі розміру. Є лише одна константа, яка зветься sqlite_max_length та дорівнює один мільярд. При цьому вне залежності від типу, який було вказано при оголошенні, кожне окреме значення може мати будь-який тип у будь-якому полі. Коли виникає оголошення типу, SQLite зберігає його для себе в початковому стані для використання у якості справки при неявній трансформації типів, якщо назва є схожою на те, що є відомим для БД(SQLite).

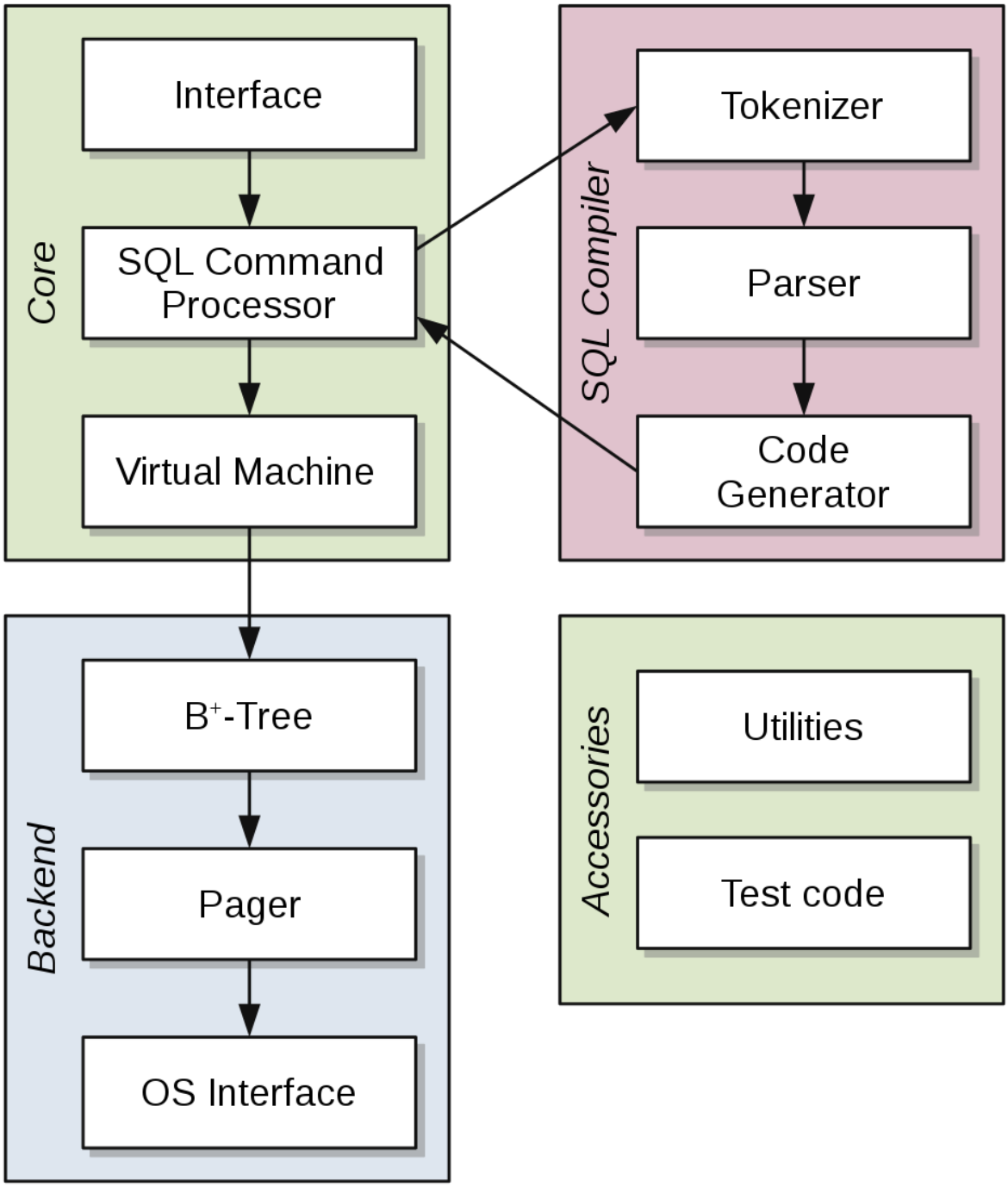

Рис. 2.4. Архітектура БД SQLite.

2.4. Опис структури системи та алгоритмів її функціонування

Розроблена система являє собою Telegram-bot, що написаний з використанням фреймворку aiogram і мови Python. Відповідно до цього було використано усі необхідні компоненти в процесі розробки для створення боту.

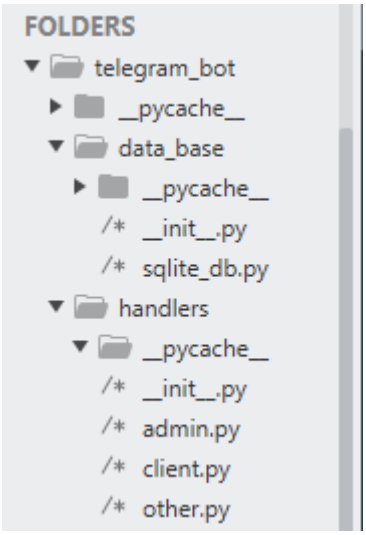

Рис.2.5. Файлова структура боту 1.

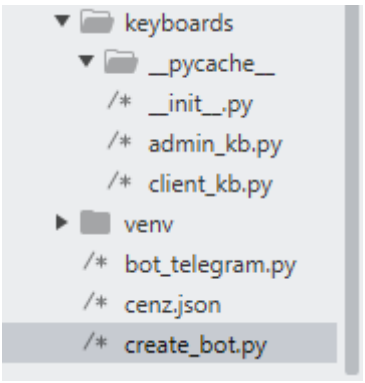

Рис. 2.6. Файлова структура боту 2.

В цілях менеджменту та «правильної» розробки усю систему(бота) було розміщено до папки telegram_bot. Після чого код розбито, а ті частини коду, які відповідають за конкретні дії бота, були поміщені в окремі папки з відповідними назвами.

Уся система вцілому складається з написаного коду, що розбито по папкам та пов’язано між собою в єдиний механізм. Також було створено database, яка використовується для реалізації можливостей «меню» для закладу. База даних створюється за спеціальним скриптом, тож у разі запуску програми та відсутності файла з відповідною назвою його буде створено автоматично. Database як і усі інші елементи були імпортовані до відповідних файлів, якщо вони в них використовуються. Створено машину станів для заповнення меню.

create_bot.py | client.py

```
1  from aiogram import Bot, types
2  from aiogram.dispatcher import Dispatcher
3
4  import os
5
6  from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton
7  from aiogram.contrib.fsm_storage.memory import MemoryStorage
8
9  storage=MemoryStorage()
10
11 bot = Bot(token=os.getenv('TOKEN'))
12 dp = Dispatcher(bot, storage=storage)
13
```

Рис. 2.7. Введення імпортів.

Як можна побачити з рисунку 2.7, було імпортовано два основні класи з бібліотеки aiogram. Безпосередньо імпортовано клас Bot та клас types. Також для того аби бот реагував на події в чаті потрібен клас Dispatcher. Окрім цього, оскільки означений бот в ході розробки мав отримати власну клавіатуру, було імпортовано також і інші потрібні для виконання задачі модулі, включаючи модуль, що дозволить прочитати token(ключ) даного боту. Як вже було зазначено, ключ отримано від BotFather за допомогою налаштувань самого Telegram. Розробка бота велася з використанням класичних для aiogram асинхронних функцій та ключових слів.

```
create_bot.py  ×   client.py  ×
from aiogram import types, Dispatcher
from create_bot import dp, bot
from keyboards import kb_client
from data_base import sqlite_db
```

Рис. 2.8. Приклад імпортів файлу client.py.

Запуск бота реалізовано за допомогою команди наведеної на рис. 2.9.

```
18
19  executor.start_polling(dp, skip_updates=True, on_startup=on_startup)
```

Рис. 2.9. Команда запуску бота.

Частину налаштувань бота, наприклад ім’я, отримання токену чи зміна фотографії профілю, проводимо безпосередньо з ботом BotFather у Telegram.

Рис.2.10. Зміна фотографії бота.

Боти Telegram можуть працювати у двох режимах – long polling та webhook. Приклади використання обох технологій розглянуто на малюнках.

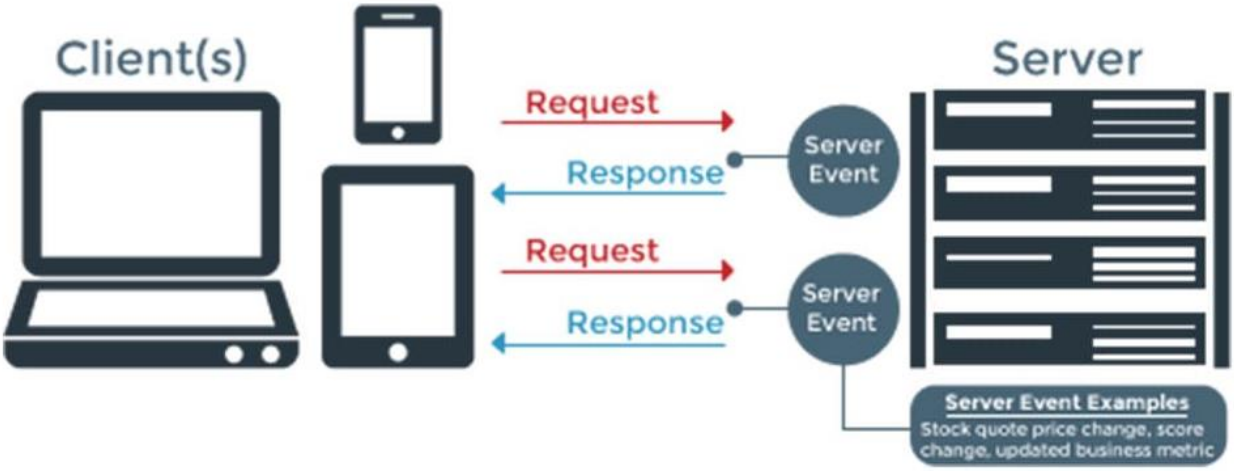

Рис. 2.11. Технологія long polling.

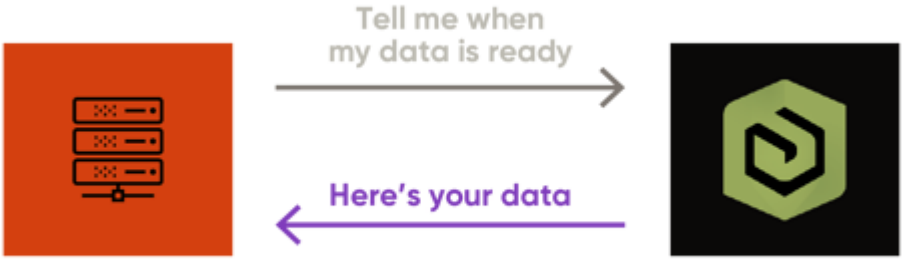

Рис. 2.12. Webhook.

Вся взаємодія бота з Telegram проходить за допомогою Telegram Bot API. Як ми бачимо, технологія webhook відрізняється тим, що програма(бот) чекає поки сервер щось спитає. І навпаки, polling змушує бота самому постійно опитувати сервер. Тобто при роботі з webhook бот може спокійно відпочивати допоки йому не надійде сигнал від сервера. Така технологія може допомогти зменшити навантаження сервера, втім вона потребує розміщення(deployment) бота на віддаленій(облачний) сервер. Зазвичай, сервіси що надають такі можливості, беруть за це кошти, а на безкоштовних платформах ви навряд чи

знайдете достатньо місця чи потрібний термін дії. В той час як використовуючи polling задіяні лише локальні ресурси робочої машини.

При створенні бота для наочного прикладу було використано long polling. Це обгрунтовано меншою витратою ресурсів та доцільністю використання в конкретному проекті.

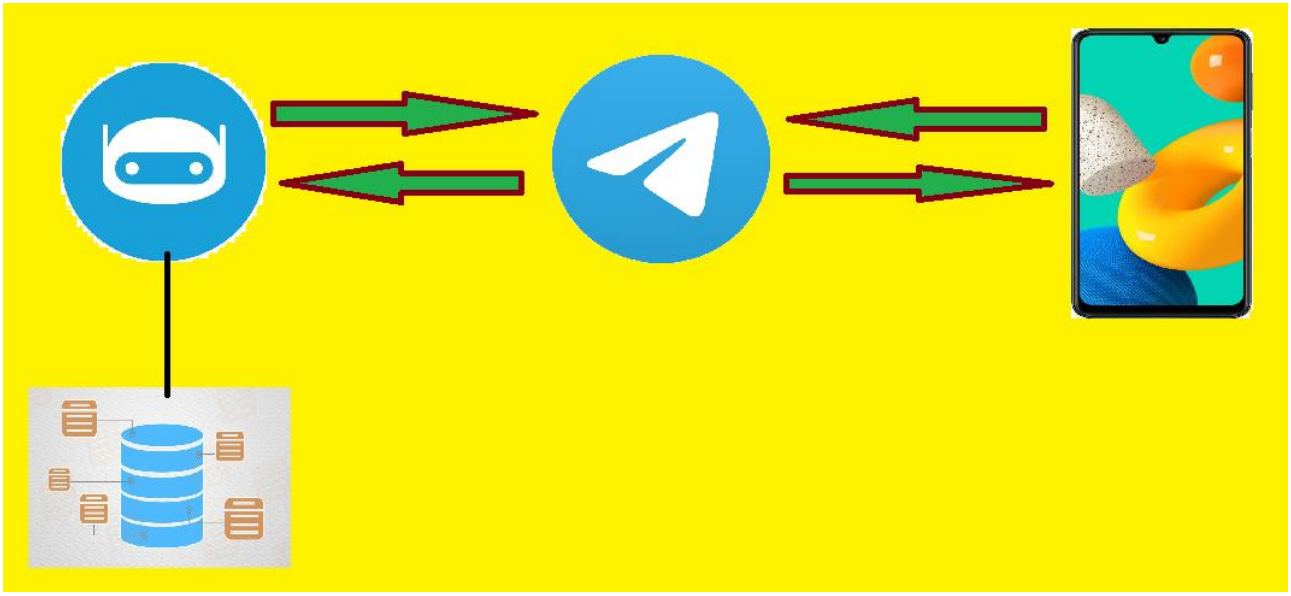

Рис. 2.13. Легкий приклад роботи бота з Telegram Bot API.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Telegram-bot призначено для застосування в сфері ресторанів з ціллю замовлення їжі, таким чином вхідними даними будуть слугувати повідомлення і команди в чаті від користувачів, а також інформація, яку адміністратор вноситиме в БД. Окрім цих двох більше вхідних даних не передбачено.

Вихідними даними відповідно будуть відповіді на повідомлення користувачів чи команди, наприклад меню ресторану, яке завантажив адміністратор. Також вихідними даними слугуватимуть інтерактивні елементи інтерфейсу - інлайн кнопки та звичайні кнопки.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Робота по створенню бота виконувалась на одному комп’ютері. Для виконання роботи по створенню бота для Telegram були використані наступні технічні засоби:

1) Монітор LG W2486L.
2) Материнська плата GIGABYTE Z370P D3.
3) Оперативна пам’ять GOODRAM DDR4 12 гігабайт.
4) Відеокарта MANLI GeForce GTX 1060 6GB.
5) Процесор Intel Core i7-8700.
6) SSD диск GIGABYTE 240GB 2.5 SATA.
7) Блок живлення CHIEFTEC Smart GPS-600A8.
8) Клавіатура Defender Metal Hunter GK-140L.
9) Комп’ютерна миша Razer DeathAdder Essential.

За допомогою використання вище перерахованих технічних засобів велася та була здійснена розробка Telegram-bot для ресторану.

2.6.2. Використані програмні засоби

Бота реалізовано на високорівневій мові програмування Python і задіяно використання асинхронного фреймворку aiogram. Окрім цього використана спеціальна бібліотека мови Python – sqlite3. Вона дозволила не під’єднувати бота до віддаленого серверу та не завантажувати зайві додатки на комп’ютер і проводити налаштування, написання ще більшої кількості коду. Кількість та специфікація роботи перелічених програмних засобів задовільняє потреби по створенню і реалізації бота для ресторану.

Серед інших програмних засобів слід зазначити:

- Використання OS Windows 10.
- Використання редактору документів Word 2016.
- Текстовий редактор коду SublimeText build 4126.
- Веб-браузер для пошуку інформації та перевірки бота Google Chrome.
- Використання незначного функціоналу Windows.

Для використання бота користувачами їм потрібно мати доступ до самого Telegram та Інтернет. Потрібна версія Windows має дорівнювати 7 або бути вищою за цю. Звичайно необхідно встановити веб-браузер, наприклад Chrome.

2.6.3. Виклик та завантаження програми

Оскільки розроблена програма це Telegram bot, для взаємодії з ним потрібний веб-браузер або версія Telegram для робочого столу. Запуск програми виконується за допомогою окремого файлу з розширенням bat та відповідною назвою – bot_run. Реалізація наведена на рис. 2.9. та рис. 2.14.

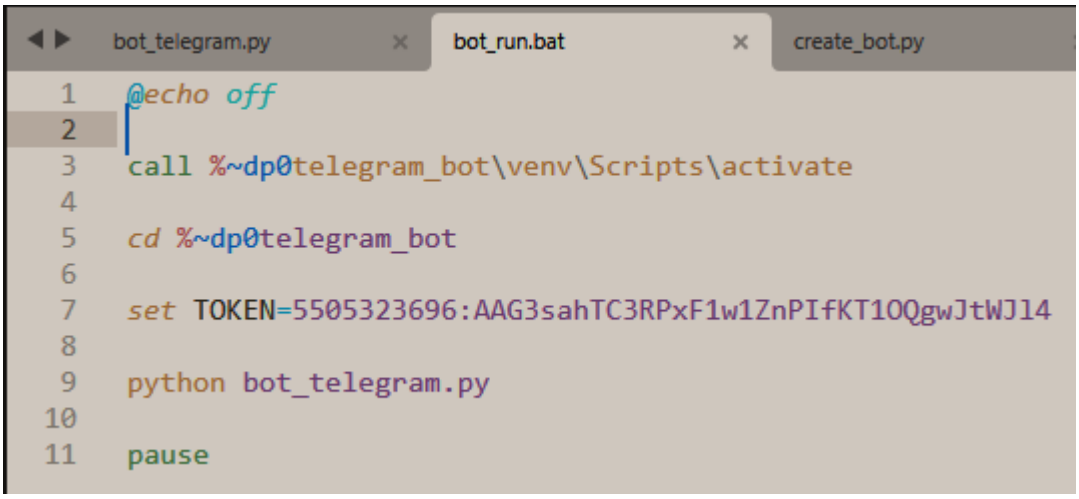

Рис. 2.14. Код файлу bot_run.

2.6.4. Опис інтерфейсу користувача

Коли користувач приєднується до телеграм-групи бот не знає id користувача. Тому для вирішення цієї проблеми було створено хендлер у який закладена функція з рис. 2.15.

```
# @dp.message_handler(commands=['start'])
async def command_start(message : types.Message):
    try:
        await bot.send_message(message.from_user.id,'Welcome, and bon appetit', reply_markup=kb_client)
        await message.delete()
    except:
        await message.reply('Talk to bot -> type to him /help :\n https://t.me/Food_autoBot')
```

Рис. 2.15. message.reply бота до незнайомого id.

Як бачимо, по команді «start» бот надсилає повідомлення з привітанням та побажанням. Але тільки у випадку коли йому відомий ваш id. Для альтернативи була написана строка у якій бот згадує користувача в групі та надає відповідні вказівки.

Після цього можно перейти до самого бота в особову переписку. Там можна розпочати спілкування з ним за допомогою команд, втім це не все.

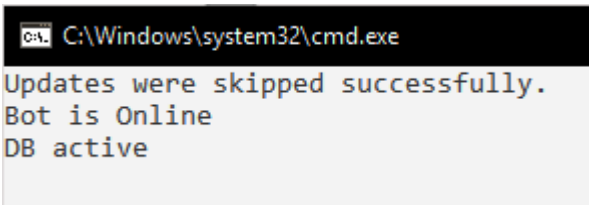

Рис. 2.16. Приклад роботи бота у консолі.

Для початку подивимося на класичні для даного бота команди та їх виконання.

Рис. 2.17. Команда /start.

З малюнку 2.17 видно як реагує бот на команду /start. Саму команду він видаляє з чату як і більшість інших за допомогою коду, наведеного на малюнку 2.15. Бачимо, що за видалення відповідає строка await message.delete().

Розглянемо приклад використання команди /help.

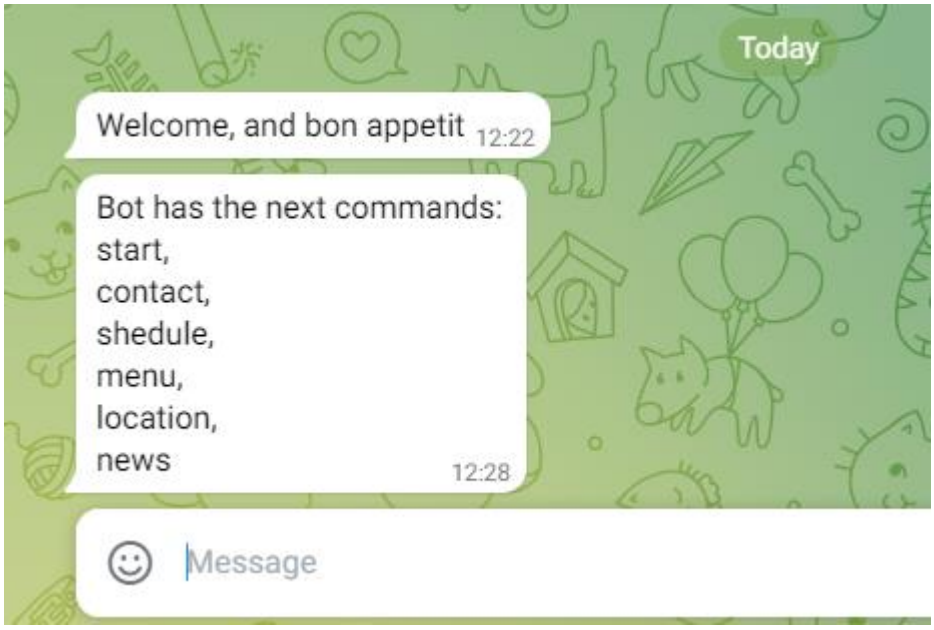

Рис. 2.18. Команда /help.

У відповіді від бота бачи список команд, які дозволяють спілкуватися з ботом користувачу і надавати йому усю необхідну інформацію. Сама команда help є класичною не тільки для більшості ботів, вона використовується майже де завгодно для надання користувачу справки, адже це легко та комфортно.

На рисунку 2.19 можна бачити клавіатуру бота до якої включені деякі команди зі списку, які є найнеобхідніші. А також відповідні до команд повідомлення.

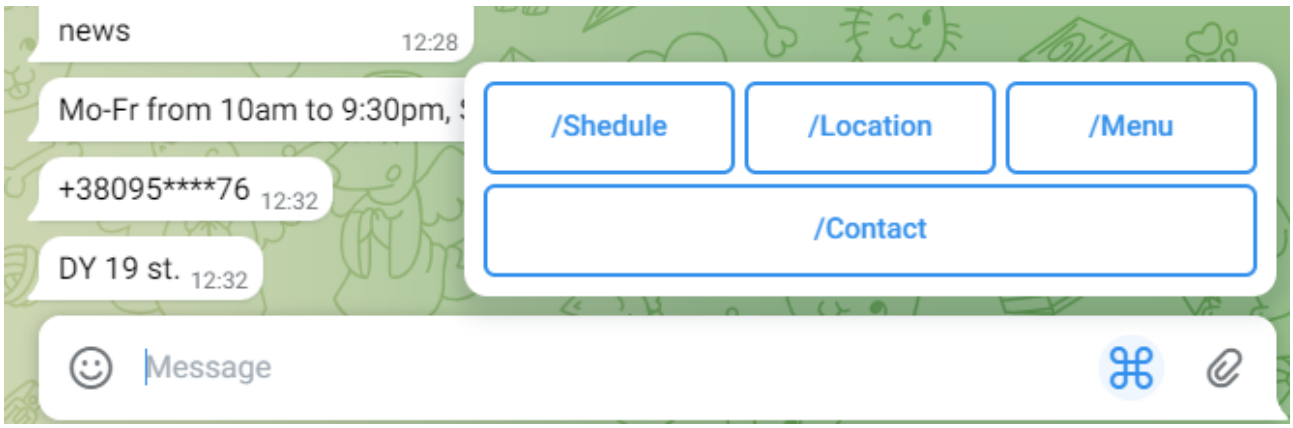

Рис. 2.19. Клавіатура бота.

Оскільки це телеграм-бот для ресторану і замовлень йому потрібне власне меню, яке бачимо на рис.2.19 та рис.2.20.

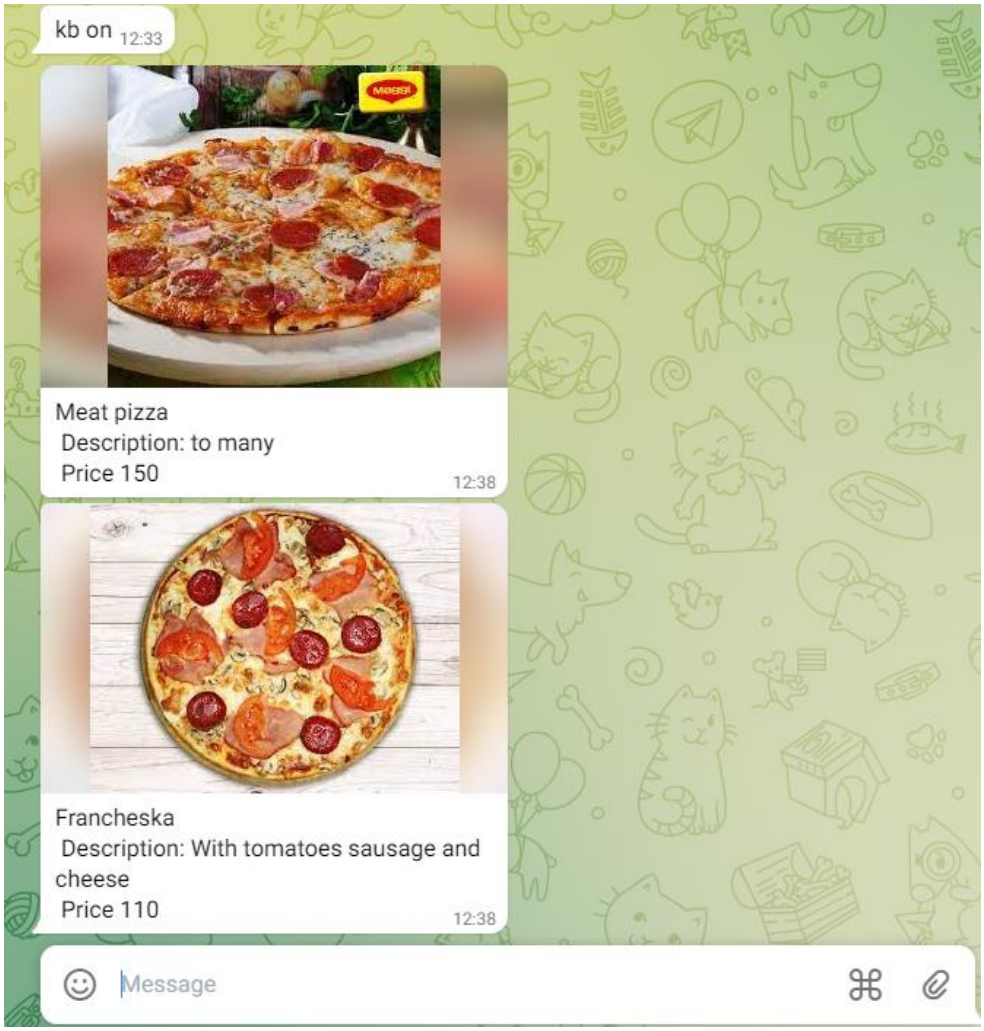

Рис. 2.20. Команда /menu.

Зокрема, на цьому ж малюнку можна побачити повідомлення «kn on» від бота. Воно повідомляє про те, що було увімкнено клавіатуру бота. Код для цього бачили на малюнку 2.15. Команда невідома користувачам, але вони можуть її використати «/». Так, це звичайний «слеш». Це дуже легко і зручно по-перше для самого розробника, а по-друге - серед користувачів деколи з’являються ті, які не знаючи команд намагаються писати цей самий слеш або команду /help. В цілому, як бачимо, вони праві.

Команда /news було розроблена як прототип. Вона викликає інлайн-кнопку в яку вбудовано посилання. Це може стати внагоді закладу для «промоушену» власного бізнесу, надання інших контактів, ресурсів тощо. Також інлайн-кнопки використовуються в режимі адміністратора для корегування меню. В цілому використання таких кнопок буває самим різним, наприклад для голосувань або ігор.

Рис. 2.21. Команда /news.

Як щойно було зазначено є режим адміністратора. Для цього користувач повинен пройти перевірку на адміністратора. Це вберігає від несанкціонованої зміни меню ресторану. Такі режими можна використовувати по-різному в залежності від напрямку та мети.

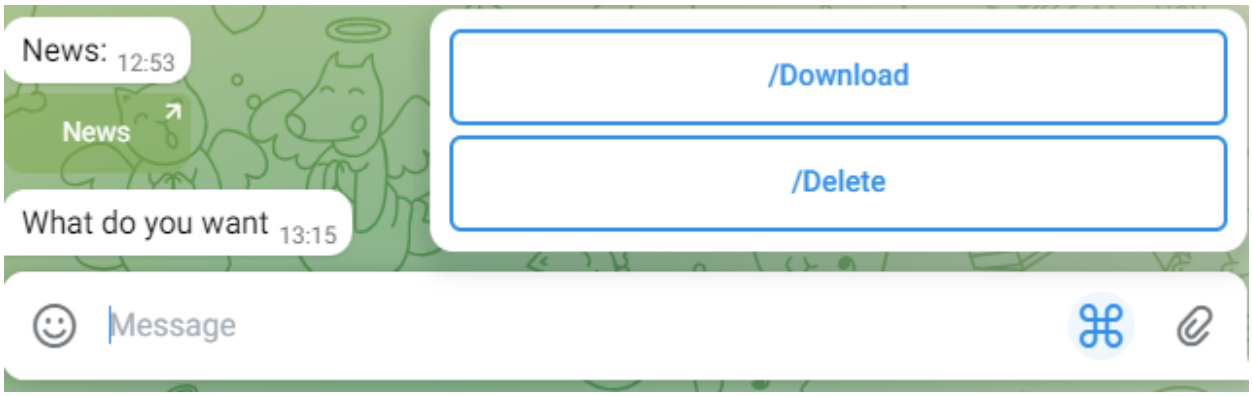

Рис.2.22. Клавіатура модератора.

Після введення спеціальної команди адміністратор групи має доступ до клавіатури бота. Вона дозволяє додати нові пункти меню або видалити непотрібні.

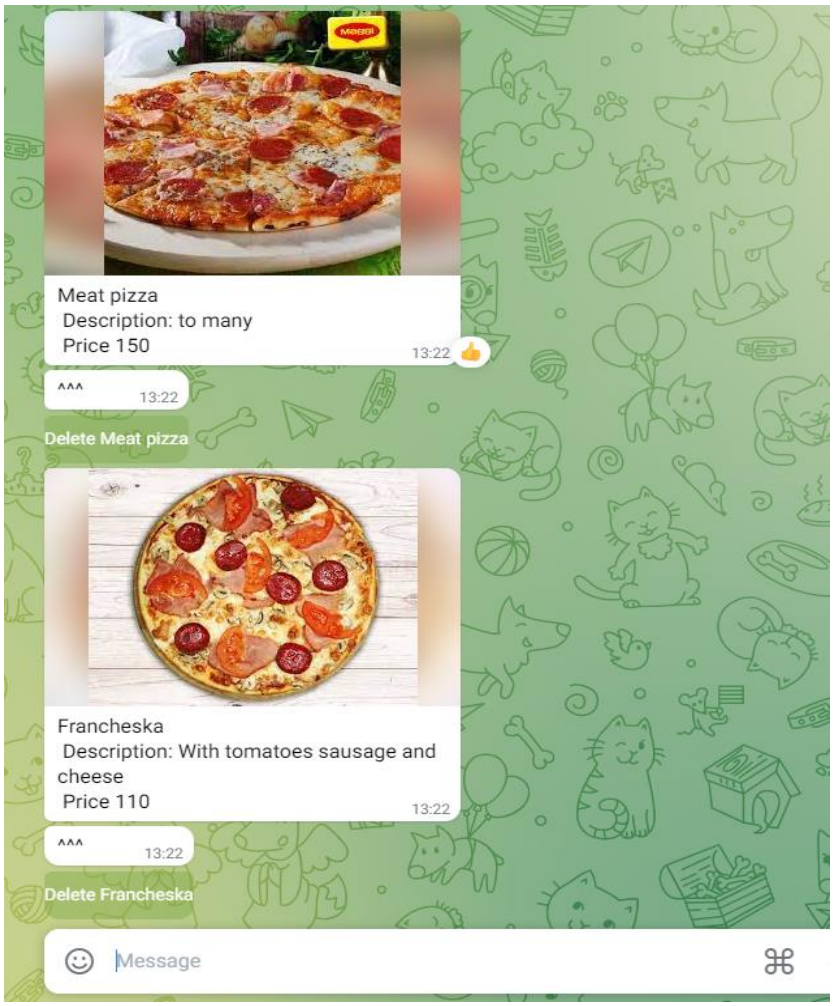

Рис. 2.23. Приклад видалення меню.

Рис. 2.24. Приклад роботи команди Download.

На малюнку 2.23 бачимо інлайн-кнопки з підказкою для видалення пунктів меню. В прикладі з командою Download видно як адміністратор додає нові пункти меню.

Інколи трапляється ситуація з помилковим натисканням кнопки або рішення було відхилено. Для цього є спеціальна команда яка виводить з режиму заповнення анкети(машини станів). На малюнку 2.25 можна побачити конкретний приклад спрацювання команди(/revoke).

Рис. 2.25. Робота команди revoke.

Зпоміж усього цього також було створено insult-filtr, який забороняє написання поганих слів та видаляє їх.

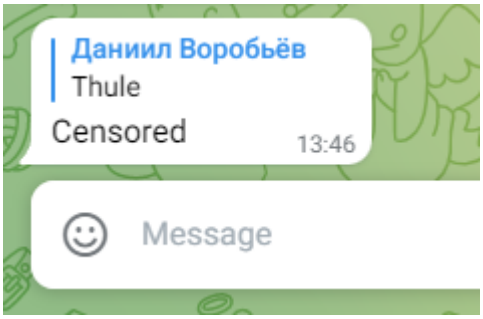

Рис. 2.26. Censorship.

Приклад 2.26 наоково демонструє роботу фільтру. В якості забороненого слова до файлу cenz.txt додана назва острова Thule. Після чого за допомогою скрипту з малюнку 2.27 створено json-файл, що лежить в директиві telegram_bot і є тим самим фільтром заборонених слів.

```
to_json.py - C:\Users\Nacho\Desktop\to_json.py (3.9.13)
File  Edit  Format  Run  Options  Window  Help
import json

ar = []

with open('cenz.txt', encoding='utf-8') as r:
    for i in r:
        n = i.lower().split('\n')[0]
        if n != '':
            ar.append(n)

with open('cenz.json', 'w', encoding='utf-8') as e:
    json.dump(ar, e)
```

Рис. 2.27. Скрипт на Python для створення фільтра слів.

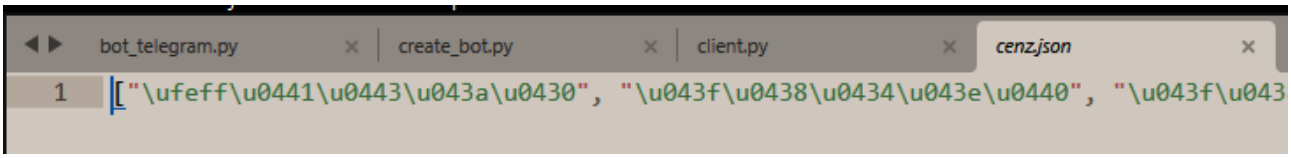

Рис. 2.28. Запис слів в створеному файлі на мові машини.

Також при розробці бота було враховано той факт, що деякі користувачі взагалі погано знайомі з ботами і командами. Тому було прийняте рішення створити «розумного бота». Це означає, що коли користувач буде писати боту меседж, бот буде реагувати на конкретні слова, якщо вони передбачені. Завдяки цьому було реалізовано «парсинг» боту, приклад чого наведено на рис. 2.29.

Рис. 2.29. Реакція боту на слово command.

Наведемо ще декілька прикладів реалізації коду.

```python
@dp.message_handler(commands='News')
async def url_command(message : types.Message):
    await message.answer('News:', reply_markup=urlkb)


@dp.message_handler(lambda message: 'command' in message.text)
async def commands(message: types.Message):
    await message.answer('Type /help')
```

Рис. 2.30. Реалізація команди news та парсингу.

```python
#Current moderator id receiving
# @dp.message_handler(commands=['moder'], is_chat_admin=True)
async def make_changes_command(message: types.Message):
    global ID
    ID = message.from_user.id
    await bot.send_message(message.from_user.id, 'What do you want', reply_markup=admin_kb.button_case_admin)
    await message.delete()

# Start new menu chapter downloading
# @dp.message_handler(commands='Download', state=None)
async def cm_start(message : types.Message):
    if message.from_user.id == ID:
        await FSMAdmin.photo.set()
        await message.reply('Download photo')

# Status Exit
# @dp.message_handler(state="*", commands='revoke')
# @dp.message_handler(Text(equals='revoke', ignore_case=True), state="*")
async def cancel_handler(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        current_state = await state.get_state()
        if current_state is None:
            return
        await state.finish()
        await message.reply('ok')

# 1st Answer Catch up
# @dp.message_handler(content_types=['photo'], state=FSMAdmin.photo)
async def load_photo(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['photo'] = message.photo[0].file_id
        await FSMAdmin.next()
        await message.reply("Enter name")

# 2nd Answer
# @dp.message_handler(state=FSMAdmin.name)
async def load_name(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['name'] = message.text
        await FSMAdmin.next()
        await message.reply("Enter description")
```

Рис. 2.31. Частина коду машини станів, виходу з нього, та режиму адміна.

Більш детальна інформація про код буде у відповідному розділі.

РОЗДІЛ 3
ЕКОНОМІЧНА ЧАСТИНА

3.1 Розрахунок трудомісткості і вартості розробки програмного продукту

Розробка Telegram-bot безумовно потребує від себе розрахунків витрат на неї, адже програмний застосунок має власну ціну та трудомісткість.

Отримали наступні дані:

1) Заробітна плата (з/п) розробника 100 гривень за одну годину.

2) Передбачені оператори 500.

3) Машино-годинні витрати техніки – 30грн/година.

4) Коефіцієнт корекції при розробці 0.2.

5) Коефіцієнт кваліфікації розробника в залежиності від опиту – 1.

6) Коефіцієнт збільшення витрат в разі нечіткої постанови задачі – 1.1.

7) Коефіцієнт складності програми 1.

Рівень нормалізації при створені програми залежний від багатьох чинників, в тому числі непередбачених, моральних тощо. Спираючись на ці факти будемо розраховувати трудомісткість таким чином:

Трудомісткість розробки, формула у людино годинах(л/г):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ л/г.} \quad (3.1)$$

t_u – витрати праці (в/п) дослідження алгоритму рішення задачі.

t_a - в/п розробки блок схеми алгоритму.

t_n - в/п програмування за блок схемою.

$t_{отл}$ - в/п налагодження програми.

$t_д$ - в/п підготовки документації.

t_o - в/п підготовки і опису поставленої задачі, зазвичай 50.

Через умовну кількість операторів програмного забезпечення визначаються складові витрати праці.

Умовна кількість операторів:

$$Q = q * C * (1 + p), \tag{3.2}$$

де q – передбачуване число операторів

C – коефіцієнт складності програми

p – коефіцієнт корекції програми в ході її розробки

$$Q = 500 * 1 *(1 + 0,2) = 600$$

В/п дослідження алгоритму розв'язку задачі:

$$t_u = \frac{Q * B}{(75…85) * k} \text{ л/г} \tag{3.3}$$

B – Коефіцієнт збільшення витрат в разі нечіткої постанови задачі

k – коефіцієнт кваліфікації розробника в залежиності від опиту

$$t_u = \frac{600 * 1,1}{75 * 1} = 8.8 \text{ л/г}$$

Витрати розробки блок схеми алгоритму:

$$t_a = \frac{Q}{(20…25) * k} \text{ л/г} \tag{3.4}$$

$$t_a = \frac{600}{24 * 1} = 25 \text{ л/г}$$

В/п програмування за блок схемою:

$$t_n = \frac{Q}{(20\ldots25) * k} \text{ л/г} \quad (3.5)$$

$$t_n = \frac{600}{25 * 1} = 24 \text{ л/г}$$

Витрати налагодження програми:

1) Налагодження звичайного завдання:

$$t_{отл} = \frac{Q}{(4..5) * k} \text{ л/г} \quad (3.6)$$

$$t_{отл} = \frac{600}{5 * 1} = 120 \text{ л/г}$$

2) Налагодження комплексного завдання:

$$t^k_{отл} = 1{,}5 * t_{отл}, \text{ л/г} \quad (3.7)$$

$$t^k_{отл} = 1{,}5 * 120 = 180, \text{ л/г}$$

Витрати на підготовку документації:

$$t_д = t_{др} + t \quad \text{л/г} \quad (3.8)$$

$t_{др}$ – трудомісткість підготовки матеріалів

$$t_{др} = \frac{Q}{15..20 * k} \text{ л/г} \quad (3.9)$$

$$t_{др} = \frac{600}{20 * 1} = 30 \text{ л/г}$$

$t_{до}$ – загальна трудомісткість оформлення документації:

$$t_{до} = 0{,}75 * t_{др} \text{ л/г} \quad (3.10)$$

$$t_{до} = 0{,}75 * 30 = 22.5 \text{ л/г}$$

$$t_{д} = 30 + 22.5 = 52.5 \text{ л/г}$$

Трудомісткість розробки бота:

$$t = 50 + 8{,}8 + 25 + 24 + 120 + 52{,}5 = 280{,}3 \text{ л/г}$$

Отримано трудомісткість програми.

3.2. Рахунок витрат на створення програми

Необхідні гроші при розробці програмного забезпечення:

$$К_{по} = З_{зп} + З_{мв} \text{ гривень} \quad (3.11)$$

З/п розробника програми:

$$З_{зп} = t * С_{пр} \text{ гривень} \quad (3.12)$$

t – загальна трудомісткість програми

$С_{пр}$ – середня з/п (у гривнях на годину)

$$З_{зп} = 280,3 * 100 = 28030 \text{ гривень}$$

Необхідний час налагодження програми виражений у вартості:

$$З_{мв} = t_{отл} * С_{мч} \text{ гривень} \quad (3.13)$$

$t_{отл}$ – трудомісткість налагодження програми

$С_{мч}$ – вартість машинних годин (у гривнях на годину)

$$З_{мв} = 180 * 30 = 5400 \text{ гривень}$$

Затрати на створення ПЗ є частиною ОКВ в даному випадку:

$$К_{по} = 28030 + 5400 = 33530 \text{ гривень}$$

Очікуваний період створення програми:

$$Т = \frac{t}{B_k * F_р} \text{ місяць} \quad (3.14)$$

B_k – Число виконавців, зазвичай береться один

$F_р$ – Часова норма роботи на місяць 176 годин

$$Т = \frac{280,3}{1 * 176} = 1,5 \text{ місяць}$$

Висновок:

Розроблене ПЗ(бот) призначено для автоматизації та замовлення їжі у ресторані, а також кофортного використання. Однією з важливих цілей було надання користувачам легкого та зручного інтерфейсу, а також усі необхідні та найважливіші можливості. Вартість даного продукту становить 33530 грн. Окрім цих витрат більше їх не потрібно. Приблизний час розробки - 1,5 місяці. Усі дані пов'язані зі значним числом операторів, і включають у себе час на дослідження, розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми та підготовку документації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено Telegram-bot для замовлення їжі у ресторані.

Бот призначено для того, щоб користувач міг легко та зручно замовити їжу з ресторану, витрачаючи при цьому мінімально часу за рахунок цього. До того ж бот пропонує для цього необхідні функції. Завдяки цьому користувач може без проблем зробити замовлення корегуючи це паралельно зі своїми іншими цілями. Допомогає у всьому цьому інтерфейс бота. Візуалізація корегується завдяки документації Telegram з передачою параметрів у коді. Використання такого дозволить підвищити прибуток закладу за рахунок нових клієнтів, що пов’язано зі зручним інтерфейсом та людьми які роблять замовлення тільки в онлайн.

За час виконання дипломного проекту було реалізовано:

- Зроблено аналіз предметної області.
- Налаштовано інтерфейс боту.
- Написані основні алгоритми програми.
- Створені адмінська, клієнтська та спільна частини боту.
- Створено локальне сховище даних.
- Визначено трудомісткість розробленої системи.
- Підрахована вартість додатку.

Розроблений Telegram-bot може:

- Реагувати на команди та слова з повідомлень.
- Надає користувачу інтерактивний інтерфейс.
- Змогу дізнатися інмормацію про заклад, контакти, меню тощо.
- Локально зберігати інформацію.
- Забороняє використовувати погані слова.

Програмне забезпечення реалізовано за допомогою мови програмування Python та фреймворку aiogram. Завдяки цьому реалізовані програмні частини боту та створена база даних на sqlite3, яку вшито в Python. При створенні бота

використовувалася інформація з офіційної документації Telegram. Необхідні додаткові налаштування робилися за допомогою бота BotFather. Мова програмування Python з великою кількістю її бібліотек дозволила значно спростити код та отримати додатковий для розробки і корегувань час. Фінальна вартісь продукту становить 33530 грн з часом виконання 1.5 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Значення мережі Інтернет / URL: https://sites.google.com/site/httpssiteinternetnovaalina/znacenna-merezi-internet-dla-ludini (дата звернення 20.05.2022).
2. Інтернет / URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82 (дата звернення 20.05.2022).
3. Керування часом / URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%87%D0%B0%D1%81%D0%BE%D0%BC (дата звернення 20.05.2022).
4. Тайм-менеджмент / URL: https://trends.rbc.ru/trends/education/606335659a7947a191c4b092 (дата звернення 20.05.2022).
5. Комфорт / URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%84%D0%BE%D1%80%D1%82 (дата звернення 20.05.2022).
6. Бот / URL: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B1%D0%BE%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) (дата звернення 20.05.2022).
7. Стандарт для роботів / URL: https://uk.wikipedia.org/wiki/Robots.txt (дата звернення 20.05.2022).
8. Боти в іграх / URL: https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D1%82_(%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D1%80%D0%B8) (дата звернення 20.05.2022.).

9. Інтерфейс / URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81 (дата звернення 25.05.2022).

10. Лекція Python / URL: https://www.youtube.com/watch?v=KdZ4HF1SrFs (дата звернення 25.05.2022).

11. Практика з Python / URL: https://www.youtube.com/watch?v=us7y0UhTq0s (дата звернення 25.05.2022).

12. Мова програмування Python та проекти / URL: https://www.youtube.com/watch?v=8FK5BWLnEP8 (дата звернення 25.05.2022).

13. Вивчення Python / URL: https://www.youtube.com/watch?v=m20LejnxUZ8 (дата звернення 25.05.2022).

14. Створення Telegram-bot / URL: https://www.youtube.com/watch?v=LJdu68ro-rU&ab_channel=GeekCode (дата звернення 25.05.2022).

15. SQLite урок / URL: https://youtu.be/Ohj-CqALrwk (дата звернення 25.05.2022).

16. Фреймворк / URL: https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA (дата звернення 25.05.2022).

17. Опис SQLite / URL: https://ru.wikipedia.org/wiki/SQLite (дата звернення 25.05.2022).

18. API / URL: https://en.wikipedia.org/wiki/API (дата звернення 26.05.2022).

19. Документація Telegram / URL: https://docs.python.org/3/faq/programming.html (дата звернення 26.05.2022).

20. Функції Python / URL: https://www.programiz.com/python-programming/anonymous-function#:~:text=What%20are%20lambda%20functions%20in,are%20also%20called%20lambda%20functions (дата звернення 26.05.2022).

21. Python / URL: https://ru.wikipedia.org/wiki/Python (дата звернення 26.05.2022).

22. Документація фреймворку aiogram / URL: https://docs.aiogram.dev/en/latest/ (дата звернення 26.05.2022).

23. Економічний розділ / URL: https://jak.bono.odessa.ua/articles/ekonomika-viznachennja-trudomistkosti-rozrobki.php (дата звернення 27.05.2022).

24. Розроблений бот / URL: https://youtu.be/ePy-MZb6rUg (дата звернення 30.05.2022).

ДОДАТОК А

КОД ПРОГРАМИ

bot_run.bat # файл запуску бота.

```
@echo off
call %~dp0telegram_bot\venv\Scripts\activate
cd %~dp0telegram_bot
set TOKEN=5505323696:AAG3sahTC3RPxF1w1ZnPIfKT1OQgwJtWJl4
python bot_telegram.py
pause
```

bot_telegram.py # збирає клієнтську частину, адмінську та головну

```
from aiogram.utils import executor
from create_bot import dp
from data_base import sqlite_db

async def on_startup(_):
        print('Bot is Online')
        sqlite_db.sql_start()

from handlers import client, admin, other

client.register_handlers_client(dp)
admin.register_handlers_admin(dp)
other.register_handlers_other(dp)

executor.start_polling(dp, skip_updates=True, on_startup=on_startup)
```

cenz.json # insult-filtr

```
["\ufeff\u0441\u0443\u043a\u0430", "\u043f\u0438\u0434\u043e\u0440", "\u043f\u0438\u0434\u0430\u0440",
"\u0448\u043b\u044e\u0445\u0430", "\u0431\u043b\u044f\u0442\u044c",
"\u0432\u044b\u0431\u043b\u044f\u0434\u043e\u043a", "\u0433\u043d\u0438\u0434\u0430",
"\u0443\u0451\u0431\u043e\u043a", "\u0443\u0435\u0431\u043e\u043a\u0430", "\u0443\u0435\u0431\u0430\u043d",
"\u0443\u0451\u0431\u0438\u0449\u0435", "\u0445\u0443\u0439", "\u043f\u0438\u0437\u0434\u0430",
"\u043f\u0438\u0437\u0434\u0435\u0446", "\u043d\u0438\u0445\u0443\u044f",
"\u0439\u043e\u0431\u0430\u043d\u044b\u0439", "\u043f\u0438\u0434\u043e\u0440\u0430\u0441",
"\u0445\u0443\u0435\u0441\u043e\u0441", "\u0435\u0431\u0430\u043d\u0443\u044e",
"\u043d\u0430\u0445\u0443\u0439", "pidoras", "pidor", "thule", "\u043f\u0438\u0434opac",
"\u043f\u0438\u0434o\u0440\u0430c", "\u043f\u0456\u0434\u043e\u0440", "\u0435\u0431\u0430\u043b\u043e",
"\u043f\u0438\u0434\u043e\u0440\u0430c", "\u0435\u0431\u0430\u043b\u043e\u043c",
"\u0435\u0431\u0430\u0442\u044c", "\u0435\u0431\u0430\u0442\u044c"]
```

client_kb.py # клавіатура бота для клієнта

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

b1 = KeyboardButton('/Shedule')
b2 = KeyboardButton('/Location')
```

```
b3 = KeyboardButton('/Menu')
b4 = KeyboardButton('/Contact')
#b5 = KeyboardButton('Share number', request_contact=True)
#b6 = KeyboardButton('Share location', request_location=True)

kb_client = ReplyKeyboardMarkup(resize_keyboard=True)

kb_client.row(b1, b2, b3).add(b4)
```

admin_kb.py # клавіатура бота для адміна

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

#Admin keys
button_load = KeyboardButton('/Download')
button_delete = KeyboardButton('/Delete')

button_case_admin = ReplyKeyboardMarkup(resize_keyboard=True).add(button_load)\
        .add(button_delete)
```

keyboard >>> __init__.py # файл ініціалізації клавіатури в папці клавіатури

```
from keyboards.client_kb import kb_client
```

other.py # головна частина

```
from aiogram import types, Dispatcher
from create_bot import dp
import json, string

# @dp.message_handler()
async def echo_send(message : types.Message):
        if {i.lower().translate(str.maketrans('', '', string.punctuation)) for i in message.text.split(' ')}\
            .intersection(set(json.load(open('cenz.json')))) != set():
            await message.reply('Censored')
            await message.delete()

def register_handlers_other(dp : Dispatcher):
    dp.register_message_handler(echo_send)
```

client.py # клієнтська частина

```
from aiogram import types, Dispatcher
from create_bot import dp, bot
from keyboards import kb_client
from data_base import sqlite_db

# @dp.message_handler(commands=['start'])
async def command_start(message : types.Message):
        try:
            await bot.send_message(message.from_user.id,'Welcome, and bon appetit', reply_markup=kb_client)
            await message.delete()
        except:
                await message.reply('Talk to bot -> type to him /help :\n https://t.me/Food_autoBot')

# @dp.message_handler(commands=['help'])
```

```
async def command_help(message : types.Message):
        await bot.send_message(message.from_user.id, 'Bot has the next commands: \nstart, \ncontact, \nshedule,
\nmenu, \nlocation, \nnews')
        await message.delete()

# @dp.message_handler(commands=['Shedule'])
async def pizza_shedule_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'Mo-Fr from 10am to 9:30pm, Sa-Su 10-9')
        await message.delete()

# @dp.message_handler(commands=['Location'])
async def pizza_location_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'DY 19 st.')
        await message.delete()

# @dp.message_handlers(commands=['Contact'])
async def pizza_contact_command(message : types.Message):
        await bot.send_message(message.from_user.id, '+38095****76')
        await message.delete()

async def test_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'kb on', reply_markup=kb_client)
        await message.delete()

#@dp.message.handler(commands=['Menu'])
async def pizza_menu_command(message : types.Message):
        await sqlite_db.sql_read(message)
        await message.delete()

def register_handlers_client(dp : Dispatcher):
    dp.register_message_handler(command_start, commands=['start'])
    dp.register_message_handler(command_help, commands=['help'])
    dp.register_message_handler(pizza_shedule_command, commands=['Shedule'])
    dp.register_message_handler(pizza_location_command, commands=['Location'])
    dp.register_message_handler(pizza_contact_command, commands=['Contact'])
    dp.register_message_handler(test_command, commands=[''])
    dp.register_message_handler(pizza_menu_command, commands=['Menu'])
```

admin.py # адмінська частина

```
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram import types, Dispatcher
from create_bot import dp, bot
from aiogram.dispatcher.filters import Text
from data_base import sqlite_db
from keyboards import admin_kb
from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton

ID = None

class FSMAdmin(StatesGroup):
        photo = State()
        name = State()
        description = State()
        price = State()

#Current moderator id receiving
# @dp.message_handler(commands=['moder'], is_chat_admin=True)
```

```
async def make_changes_command(message: types.Message):
	global ID
	ID = message.from_user.id
	await bot.send_message(message.from_user.id, 'What do you want',
reply_markup=admin_kb.button_case_admin)
	await message.delete()

# Start new menu chapter downloading
# @dp.message_handler(commands='Download', state=None)
async def cm_start(message : types.Message):
	if message.from_user.id == ID:
	    await FSMAdmin.photo.set()
	    await message.reply('Download photo')

# Status Exit
# @dp.message_handler(state="*", commands='revoke')
# @dp.message_handler(Text(equals='revoke', ignore_case=True), state="*")
async def cancel_handler(message: types.Message, state: FSMContext):
	if message.from_user.id == ID:
	    current_state = await state.get_state()
	    if current_state is None:
		return
	    await state.finish()
	    await message.reply('ok')

# 1st Answer Catch up
# @dp.message_handler(content_types=['photo'], state=FSMAdmin.photo)
async def load_photo(message: types.Message, state: FSMContext):
	if message.from_user.id == ID:
	    async with state.proxy() as data:
		data['photo'] = message.photo[0].file_id
	    await FSMAdmin.next()
	    await message.reply("Enter name")

# 2nd Answer
# @dp.message_handler(state=FSMAdmin.name)
async def load_name(message: types.Message, state: FSMContext):
	if message.from_user.id == ID:
	    async with state.proxy() as data:
		data['name'] = message.text
	    await FSMAdmin.next()
	    await message.reply("Enter description")

# 3rd
# @dp.message_handler(state=FSMAdmin.description)
async def load_description(message: types.Message, state: FSMContext):
	if message.from_user.id == ID:
	    async with state.proxy() as data:
		data['description'] = message.text
	    await FSMAdmin.next()

#	    await sqlite_db.sql_add_command(state)
	    await message.reply("Enter price")
#	    await state.finish()

# The last answer
# @dp.message_handler(state=FSMAdmin.price)
async def load_price(message: types.Message, state: FSMContext):
	if message.from_user.id == ID:
	    async with state.proxy() as data:
```

```
                data['price'] = message.text
#           await FSMAdmin.next()

            await sqlite_db.sql_add_command(state)
            await message.reply("Its done")
            await state.finish()

@dp.callback_query_handler(lambda x: x.data and x.data.startswith('del '))
async def del_callback_run(callback_query: types.CallbackQuery):
        await sqlite_db.sql_delete_command(callback_query.data.replace('del ', ''))
        await callback_query.answer(text=f'{callback_query.data.replace("del ", "")} Deleted.', show_alert=True)

@dp.message_handler(commands='Delete')
async def delete_item(message: types.Message):
        if message.from_user.id == ID:
                read = await sqlite_db.sql_read2()
                for ret in read:
                        await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n Description: {ret[2]}\n Price
{ret[-1]}')
                        await bot.send_message(message.from_user.id, text='^^^',
reply_markup=InlineKeyboardMarkup().\
                                add(InlineKeyboardButton(f'Delete {ret[1]}', callback_data=f'del {ret[1]}')))

# Register handlers
def register_handlers_admin(dp : Dispatcher):
        dp.register_message_handler(cm_start, commands=['Download'], state=None)
        dp.register_message_handler(cancel_handler, state="*", commands='revoke')
        dp.register_message_handler(cancel_handler, Text(equals='revoke', ignore_case=True), state="*")
        dp.register_message_handler(load_photo, content_types=['photo'], state=FSMAdmin.photo)
        dp.register_message_handler(load_name, state=FSMAdmin.name)
        dp.register_message_handler(load_description, state=FSMAdmin.description)
        dp.register_message_handler(load_price, state=FSMAdmin.price)
        dp.register_message_handler(make_changes_command, commands=['moder'], is_chat_admin=True)
#       dp.register_message_handler(del_callback_run, lambda x: x.data and x.data.startswith('del '))
#       dp.register_message_handler(delete_item, commands='Delete')
```

handlers >>> **__init__.py** # файл ініціалізації хендлерів в папці з хендлерами

```
from handlers import client
from handlers import admin
from handlers import other
```

sqlite_db.py # файл для опису роботи БД

```
import sqlite3 as sq
from create_bot import bot

def sql_start():
        global base, cur
        base = sq.connect('pizza_cool.db')
        cur = base.cursor()
        if base:
                print('DB active')
        base.execute('CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY KEY,
description TEXT, price TEXT)')
        base.commit()
async def sql_add_command(state):
        async with state.proxy() as data:
```

```
            cur.execute('INSERT INTO menu VALUES (?, ?, ?, ?)', tuple(data.values()))
            base.commit()

async def sql_read(message):
    for ret in cur.execute('SELECT * FROM menu').fetchall():
        await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n Description: {ret[2]}\n Price {ret[-1]}')

async def sql_read2():
    return cur.execute('SELECT * FROM menu').fetchall()

async def sql_delete_command(data):
    cur.execute('DELETE FROM menu WHERE name == ?', (data,))
    base.commit()
```

data_base >>> **__init__.py** # ініціалізація БД

```
from data_base import sqlite_db
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

«Розробка «Telegram-bot» для автоматизації підбору та замовлення харчової продукції з ресторану з використанням фреймворку aiogram»

студента групи 122-19ск-2 Воробйова Данила Дмитровича

Керівник економічного розділу

доцент каф. ПЕП та ПУ, к.е.н. **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Воробйов_диплом_ап.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Воробйов_диплом_ап.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
telegram_bot	Папка. Містить код програми та файли
Презентація	
Воробйов_презентація.ppt	Презентація кваліфікаційної роботи